frontiers in Applied Mathematics and Statistics

TYPE Technology and Code
PUBLISHED 26 March 2026
DOI 10.3389/fams.2026.1762084

OPEN ACCESS

EDITED BY
Felix Krahmer,
Technical University of Munich, Germany

REVIEWED BY
Christian Fiedler,
Technical University of Munich, Germany
Huiming Zhang,
Beihang University, China

*CORRESPONDENCE
Hao Zhu
zhuh@cs.uni-freiburg.de

RECEIVED 06 December 2025
REVISED 28 February 2026
ACCEPTED 10 March 2026
PUBLISHED 26 March 2026

CITATION
Zhu H, Hoffmann J, Zhang B and Boedecker J (2026) Fitting reinforcement learning model to behavioral data under bandits.
Front. Appl. Math. Stat. 12:1762084.
doi: 10.3389/fams.2026.1762084

Fitting reinforcement learning model to behavioral data under bandits

Hao Zhu[1,2]*, Jasper Hoffmann[1,2], Baohe Zhang[1,2] and Joschka Boedecker[1,2]

[1]IMBIT/BrainLinks-BrainTools, Freiburg, Germany, [2]Department of Computer Science, University of Freiburg, Freiburg, Germany

We consider the problem of fitting a reinforcement learning (RL) model to some given behavioral data under a multi-armed bandit environment. These models have received much attention in recent years for characterizing human and animal decision-making behavior. We provide a generic mathematical optimization problem formulation for the fitting problem of a wide range of RL models that appear frequently in scientific research applications. We then provide a detailed theoretical analysis of its convexity properties. Based on the theoretical results, we introduce a novel solution method for the fitting problem of RL models based on convex relaxation and optimization. Our method is then evaluated in several simulated and real-world bandit environments to compare with some benchmark methods that appear in the literature. Numerical results indicate that our method achieves comparable performance to the state-of-the-art, while significantly reducing computation time. We also provide an open-source Python package for our proposed method to empower researchers to apply it in the analysis of their datasets directly, without prior knowledge of convex optimization.

KEYWORDS
behavior modeling, convex optimization, convex relaxation, multi-armed bandits, reinforcement learning

1 Introduction

We consider the problem of fitting a reinforcement learning (RL) model to some given behavioral data under a multi-armed bandit environment.

1.1 Bandits and reinforcement learning

A *bandit problem* is a sequential game between a player and an environment, where the player may choose one of several actions at each time step, and receives some reward from the environment that depends on the selected action. The actions are often referred to as *arms* in the literature, so a k-armed bandit refers to a bandit problem with k possible actions [1]. The player's goal is to maximize the cumulative reward across the whole episode, which requires the player to learn the reward structure of the environment and make informed decisions based on the learned information. Obviously, the player under the bandit setting can only select their action based on the history of their past actions and the received rewards, which forms a typical setting for RL problems [2].

Multi-armed bandits have been a very active research area of engineering, including computer science, operations research, economics, and statistics [3].

1.2 Multi-armed bandit behavioral tasks

Apart from the applications in engineering, the class of multi-armed bandit tasks is also an experimental paradigm used to investigate a wide range of animals' decision-making processes. These tasks have been widely used in neuroscience [4–16], psychology [17–19], and medical [20] researches. In these tasks, the animal or human subject is faced with multiple choices with different rewards, and may choose one of them for each trial. Some variants also include shuffling the reward assigned to each choice regularly or randomly (which are sometimes referred to as *dynamic bandits*), or introducing an external cue, *e.g.*, image, sound, etc., to individual choices. Under the latter type of environment, subjects can select their action according to some contextual information, instead of just based on trial and error. Nevertheless, the common goal of the bandit task for the subject is to maximize the cumulative reward across the whole episode (i.e., experimental session).

1.3 RL model for decision making under bandits

In the context of engineering, RL is a class of algorithms that can be used for solving multi-armed bandit problems. Meanwhile, in behavioral science, RL also serves as one of the mathematical models for characterizing the decision-making behavior of animals and humans under a bandit task (see, *e.g.*, Beron et al. [21] for a review of different models for animal behavior characterization). The following procedure describes the most basic instance of the class of RL models, namely the *forgetting Q-learning* model [21, 22]: Let $m \in \mathbf{Z}_{++}$ be the number of possible *actions* (choices) in the bandit task, and let $t \in \mathbf{Z}_{+}$ be the discrete time step. After the choice at time step $t-1$, the subject receives a *reward signal* $u(t) \in \mathbf{R}^m$ that depends on the selected action, given by

$$u_i(t) = \begin{cases} 1 & \text{if action } i \text{ was selected } \textit{and} \text{ rewarded} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

for $i = 1, \ldots, m$. To maximize the cumulative reward, the subject formulates some *value function* (or really, vector) $x(t) \in \mathbf{R}^m$ for each time step $t \geq 1$, and recursively updates it according to

$$x(t) = x(t-1) + \alpha(\beta u(t) - x(t-1)), \tag{2}$$

where the parameters $\beta \in [0, \infty)$ can be interpreted as the *sensitivity* to the reward signal $u(t)$, and $\alpha \in [0, 1]$ is the *learning rate* of the value estimation error $(\beta u(t) - x(t-1))$. By convention, the initial value function at $t = 0$ is set to $x(0) = 0$. Let $a(t) \in \{1, \ldots, m\}$ denote the subject's action at the tth time step, which is then assumed to be selected according to:

$$\text{prob}(a(t) = i) = \frac{\exp(x_i(t))}{\sum_{j=1}^{m} \exp(x_j(t))}, \quad i = 1, \ldots, m, \quad t = 0, \ldots, n. \tag{3}$$

In addition to the basic example above, several extensions to the forgetting Q-learning model have been developed to capture more subtle behavioral properties in modeling (see Section 2.2 for more details).

1.4 RL model fitting problem

Behavioral science researchers, i.e., the users of the RL models, are interested in obtaining an individual subject's behavior characterization, given the observed behavior outcome. In particular, for the case of the forgetting Q-learning model defined by Equations 1–3, the goal is to recover the model parameters α and β as well as the value functions $x(t)$, $t = 1, \ldots, n$, given the dataset $\{(u(t), a(t))\}_{t=1}^{n}$ (although in practice the value functions are generally of more interest). Roughly speaking, this leads to solving an optimization problem with the objective function being the likelihood of observing the subject's behavior under the model assumptions. The variables for these problems are then the model parameters and the value function at each time step. (A formal definition of the RL model fitting problem will be introduced in Section 2).

1.5 This paper

Despite the fast-growing of RL behavior model applications in the scientific research community, a generic formal definition and analysis of the properties for the RL model fitting problem have not yet been well established. As a partial consequence, regarding the practical aspect, current solution methods for fitting RL models are either very slow or difficult to implement and debug (see Section 6 for more detail). In this study, we aim to fill these gaps with the following three-fold contributions:

- Firstly, we formalize the mathematical optimization problem corresponding to the fitting problem of a wide range of the most commonly used RL models in Section 2.
- Then, in Section 3, we evaluate the convexity properties of the RL model fitting problems according to our problem formulation.
- Finally, based on the theoretical results, in Section 4, we introduce a novel solution method for fitting RL models to bandit behavioral data via convex optimization.

Our proposed solution method is then evaluated in several simulated and real-world bandit environments to compare with some benchmark methods that appear in the literature. Numerical results indicate that our method achieves comparable performance with significantly decreased computing time. The implementation of our method is fully open-sourced as a Python package under https://github.com/nrgrp/rlfit, such that it can be easily applied by users not well versed in convex analysis and optimization.

2 The fitting problem of RL models

2.1 Basic forgetting Q-learning model

We start from the fitting problem of the basic forgetting Q-learning model given by Equations 1–3. Recall that here the objective is to maximize the likelihood of observing the given data $a(t)$ and $u(t)$ for $t = 1, \ldots, n$, with the variables being the value functions $x(t)$ and the model parameters α and β. First, notice that Equation 2 can be written as

$$x(t) = (1-\alpha)x(t-1) + \alpha\beta u(t).$$

For simplicity of notation, we transform the actions $a(t) \in \{1, \ldots, m\}$ into one-hot representations, given by $y(t) \in \{e_1, \ldots, e_m\} \subseteq \mathbf{R}^m$, where e_i is the ith standard basis vector, i.e.,

$$y_i(t) = \begin{cases} 1 & a(t) = i \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

for all $i = 1, \ldots, m$. Then the log-likelihood of observing $a(t)$ at time step t is

$$\ell(x(t), y(t)) = \log\left(y(t)^T \left(\frac{\exp(x(t))}{\sum_{i=1}^m \exp(x_i(t))}\right)\right). \tag{5}$$

Put together, the fitting problem of the forgetting Q-learning model can be written as

$$\begin{array}{ll} \text{minimize} & -\sum_{t=1}^n \ell(x(t), y(t)) \\ \text{subject to} & x(t) = (1-\alpha)x(t-1) + \alpha\beta u(t), \quad t = 1, \ldots, n \\ & x(0) = 0, \quad 0 \le \alpha \le 1, \quad \beta \ge 0, \end{array} \tag{6}$$

where the problem variables are $\alpha, \beta \in \mathbf{R}$ and $x(1), \ldots, x(n) \in \mathbf{R}^m$, the problem data are $y(t), u(t) \in \mathbf{R}^m$, and each log-likelihood term in the objective is given by Equation 5.

2.2 Extensions

2.2.1 Extension on reward signals

As a simple extension to the forgetting Q-learning model, one may assume a different reward signal $u(t)$ as in Equation 1. For example, some kind of "punishment" could possibly be integrated to the unrewarded choices, *e.g.*, replacing all the zeros in Equation 1 with -1 [23, 24]. This type of extension only changes the problem data of Equation 6, which does not influence the properties of the fitting problem itself.

2.2.2 Multiple learning rates and reward sensitivity

One of the widely applied extensions to the basic forgetting Q-learning model is to incorporate different learning rates α and reward sensitivity β for individual actions of the bandit [12, 15]. In other words, for each entry $x_i(t)$ of the value function $x(t) \in \mathbf{R}^m$, $i = 1, \ldots, m$, we have

$$x_i(t) = x_i(t-1) + \alpha_i(\beta_i u_i(t) - x_i(t-1)).$$

In this case, the model parameters α and β are not just two real numbers, but two vectors in $\mathbf{R}^m$ with each entry satisfying $\alpha_i \in [0,1]$ and $\beta_i \in [0, \infty)$, $i = 1, \ldots, m$. Hence, the model fitting problem (Equation 6) is now extended to be

$$\begin{array}{ll} \text{minimize} & -\sum_{t=1}^n \ell(x(t), y(t)) \\ \text{subject to} & x(t) = \mathbf{diag}(1-\alpha)x(t-1) + \mathbf{diag}(\alpha)\,\mathbf{diag}(\beta)u(t), \\ & \quad t = 1, \ldots, n \\ & x(0) = 0, \quad 0 \preceq \alpha \preceq 1, \quad \beta \succeq 0 \end{array} \tag{7}$$

with variables $\alpha, \beta \in \mathbf{R}^m$ and $x(1), \ldots, x(n) \in \mathbf{R}^m$.

2.2.3 Sub-reward signals and sub-value functions

Another type of extension to the basic forgetting Q-learning model that appears frequently in practice takes the following assumption: There exist multiple *sub-reward signals* $u^{(1)}(t), \ldots, u^{(k)}(t) \in \mathbf{R}^m$ [21], corresponding to multiple *sub-value functions* $z^{(1)}(t), \ldots, z^{(k)}(t) \in \mathbf{R}^m$. These sub-value functions are then updated individually with parameters $\alpha^{(i)}, \beta^{(i)} \in \mathbf{R}$, according to

$$z^{(i)}(t) = z^{(i)}(t-1) + \alpha^{(i)}(\beta^{(i)}u^{(i)}(t) - z^{(i)}(t-1))$$

for all $i = 1, \ldots, k$. The value function $x(t)$ used in Equation 3 for action selection is then a linear combination of $z^{(1)}(t), \ldots, z^{(k)}(t)$ under some *given* weight vector $w \in \mathbf{R}^k$ (which is commonly assumed to be $w = \mathbf{1}$), i.e., $x(t) = w_1 z^{(1)}(t) + \cdots + w_k z^{(k)}(t)$. In this setup, the problem (Equation 6) now becomes

$$\begin{array}{ll} \text{minimize} & -\sum_{t=1}^n \ell(x(t), y(t)) \\ \text{subject to} & x(t) = \begin{bmatrix} z^{(1)}(t) & \cdots & z^{(k)}(t) \end{bmatrix} w \\ & z^{(i)}(t) = (1-\alpha^{(i)})z^{(i)}(t-1) + \alpha^{(i)}\beta^{(i)}u^{(i)}(t) \\ & z^{(i)}(0) = 0, \quad 0 \le \alpha^{(i)} \le 1, \quad \beta^{(i)} \ge 0 \\ & i = 1, \ldots, k, \quad t = 1, \ldots, n, \end{array} \tag{8}$$

where the variables are $\alpha^{(i)}, \beta^{(i)} \in \mathbf{R}$, $z^{(i)}(1), \ldots, z^{(i)}(n) \in \mathbf{R}^m$ for all $i = 1, \ldots, k$, and $x(1), \ldots, x(n) \in \mathbf{R}^m$; the problem data are $w \in \mathbf{R}^k$ and $y(t), u^{(i)}(t) \in \mathbf{R}^m$, $t = 1, \ldots, n$, $i = 1, \ldots, k$.

2.3 RL model fitting problems in general form

It is easily seen that the RL model fitting problems, given by Equations 6–8, can be written in the following general form:

$$\begin{array}{ll} \text{minimize} & -\sum_{t=1}^n \ell(x(t), y(t)) \\ \text{subject to} & x(t) = \begin{bmatrix} z^{(1)}(t) & \cdots & z^{(k)}(t) \end{bmatrix} w \\ & z^{(i)}(t) = \mathbf{diag}(1-\alpha^{(i)})z^{(i)}(t-1) + \mathbf{diag}(\alpha^{(i)}) \\ & \mathbf{diag}(\beta^{(i)})u^{(i)}(t) \\ & z^{(i)}(0) = 0, \quad 0 \preceq \alpha^{(i)} \preceq 1, \quad \beta^{(i)} \succeq 0 \\ & i = 1, \ldots, k, \quad t = 1, \ldots, n, \end{array} \tag{9}$$

where the variables are $\alpha^{(i)}, \beta^{(i)} \in \mathbf{R}^m$, $z^{(i)}(1), \ldots, z^{(i)}(n) \in \mathbf{R}^m$ for all $i = 1, \ldots, k$, and $x(1), \ldots, x(n) \in \mathbf{R}^m$; the problem data are $w \in \mathbf{R}^k$, $y(t), u^{(i)}(t) \in \mathbf{R}^m$, $t = 1, \ldots, n$, $i = 1, \ldots, k$. Assuming

$w = \mathbf{1}$, by taking $k = 1$, the problem (9) reduces to (7); by adding additional constraints $\alpha_1^{(i)} = \cdots = \alpha_m^{(i)}$ and $\beta_1^{(i)} = \cdots = \beta_m^{(i)}$, $i = 1, \ldots, k$, the problem (Equation 9) reduces to Equation 8; and by combining the two additional requirements above together, we have the basic forgetting Q-learning model fitting problem (Equation 6). The time complexity of *evaluating* the objective and constraints of Equation 9 is $O(mnk)$.

3 Convexity properties

To analyze the convexity properties of the general RL model fitting problem (Equation 9), we start by eliminating the recursive expression for $z^{(i)}(t)$. Notice that for each entry of the vector $z^{(i)}(t)$, we have

$$z_j^{(i)}(t) = \sum_{\tau=1}^{t} (1-\alpha_j^{(i)})^{t-\tau} \alpha_j^{(i)} \beta_j^{(i)} u_j^{(i)}(\tau)$$

for all $j = 1, \ldots, m$. Hence, the sub-value functions $z^{(i)}(t)$ for all $t = 1, \ldots, n$ can be expressed as

$$z^{(i)}(t) = \mathbf{diag}\left(\begin{bmatrix} \alpha_1^{(i)}\beta_1^{(i)} & (1-\alpha_1^{(i)})^1 \alpha_1^{(i)}\beta_1^{(i)} & \cdots & (1-\alpha_1^{(i)})^{n-1}\alpha_1^{(i)}\beta_1^{(i)} \\ \vdots & \vdots & & \vdots \\ \alpha_m^{(i)}\beta_m^{(i)} & (1-\alpha_m^{(i)})^1 \alpha_m^{(i)}\beta_m^{(i)} & \cdots & (1-\alpha_m^{(i)})^{n-1}\alpha_m^{(i)}\beta_m^{(i)} \end{bmatrix} \tilde{U}^{(i)}(t) \right),$$

where the matrices $\tilde{U}^{(i)}(t)$ are defined as

$$\tilde{U}^{(i)}(t) = \begin{bmatrix} U^{(i)}(t) \\ 0 \end{bmatrix} \in \mathbf{R}^{n\times m}, \quad U^{(i)}(t) = \begin{bmatrix} u^{(i)}(t)^T \\ \vdots \\ u^{(i)}(1)^T \end{bmatrix} \in \mathbf{R}^{t\times m}. \tag{10}$$

Define the transformation $F\colon \mathbf{R}^m \times \mathbf{R}^m \to \mathbf{R}^{m\times n}$, given by

$$F\colon (a,b) \mapsto \begin{bmatrix} a_1 b_1 & (1-a_1)^1 a_1 b_1 & \cdots & (1-a_1)^{n-1} a_1 b_1 \\ \vdots & \vdots & & \vdots \\ a_m b_m & (1-a_m)^1 a_m b_m & \cdots & (1-a_m)^{n-1} a_m b_m \end{bmatrix}, \quad a, b \in \mathbf{R}^m, \tag{11}$$

then the problem (Equation 9) can be written as

$$\begin{array}{ll} \text{minimize} & -\sum_{t=1}^n \ell(x(t), y(t)) \\ \text{subject to} & x(t) = \begin{bmatrix} z^{(1)}(t) & \cdots & z^{(k)}(t) \end{bmatrix} w \\ & z^{(i)}(t) = \mathbf{diag}(F(\alpha^{(i)}, \beta^{(i)})\tilde{U}^{(i)}(t)) \\ & z^{(i)}(0) = 0, \quad 0 \preceq \alpha^{(i)} \preceq 1, \quad \beta^{(i)} \succeq 0 \\ & i = 1, \ldots, k, \quad t = 1, \ldots, n, \end{array} \tag{12}$$

where the variables are $\alpha^{(i)}, \beta^{(i)} \in \mathbf{R}^m$, $z^{(i)}(1), \ldots, z^{(i)}(n) \in \mathbf{R}^m$ for all $i = 1, \ldots, k$, and $x(1), \ldots, x(n) \in \mathbf{R}^m$. The problem data of Equation 12 are $w \in \mathbf{R}^k$, $y(1), \ldots, y(n) \in \mathbf{R}^m$, and $\tilde{U}^{(i)}(1), \ldots, \tilde{U}^{(i)}(n) \in \mathbf{R}^{m\times n}$ for all $i = 1, \ldots, k$ given by Equation 10.

We can now easily check the convexity of Equation 9 via the equivalent form (Equation 12). Note that the objective function in Equation 12 can be written as

$$\begin{aligned} -\sum_{t=1}^n \ell(x(t), y(t)) &= -\sum_{t=1}^n \log\left(\frac{y(t)^T \exp(x(t))}{\sum_{i=1}^m \exp(x_i(t))} \right) \\ &= -\sum_{t=1}^n \left(y(t)^T x(t) - \log \sum_{i=1}^m \exp(x_i(t)) \right), \end{aligned} \tag{13}$$

where the second equality is from the fact that, by Equation 4, the vector $y(t)$ is a standard basis vector for all $t = 1, \ldots, n$. Since in the last expression of Equation 13, the $y(t)^T x(t)$ is affine and the second log-sum-exp term is convex [25, Section 3.1], we conclude that the objective of Equation 12 is convex in the variables $x(t)$ [this follows from basic convex analysis [25, 26]]. Then we immediately see that the problem (Equation 12) is convex if and only if the equality constraints are all affine and the inequality constraints are all convex. However, this condition is violated by the second constraint

$$z^{(i)}(t) = \mathbf{diag}(F(\alpha^{(i)}, \beta^{(i)})\tilde{U}^{(i)}(t)), \quad i = 1, \ldots, k, \quad t = 1, \ldots, n,$$

since the transformation F given by Equation 11 is *not* affine. Hence, we conclude that the RL model fitting problem (Equation 12) is *not* convex. As a result, even if the time complexity of evaluating the objective and constraints of Equation 9 is only $O(mnk)$, the complexity of *solving* it to global optimality in the worst case can be exponential in mnk [27, 28].

4 Solution method

4.1 The convex surrogate

Analysis in Section 3 indicates that by relaxing the transformation F given by Equation 11 to be affine, the RL model fitting problem can then be convexified. To make such relaxation more explicit, we consider an equivalent formulation of Equation 12 given as follows. For all $i = 1, \ldots, k$, let

$$\eta^{(i)} \in \mathbf{R}^m \quad \text{and} \quad G^{(i)} = \begin{bmatrix} g_1^{(i)} & \cdots & g_n^{(i)} \end{bmatrix} \in \mathbf{R}^{m\times n},$$

where $g_1^{(i)}, \ldots, g_n^{(i)} \in \mathbf{R}^m$ are the columns of the matrix $G^{(i)}$. The problem (Equation 12) is then equivalent to

$$\begin{array}{ll} \text{minimize} & -\sum_{t=1}^n \ell(x(t), y(t)) \\ \text{subject to} & x(t) = \begin{bmatrix} z^{(1)}(t) & \cdots & z^{(k)}(t) \end{bmatrix} w \\ & z^{(i)}(t) = \mathbf{diag}(G^{(i)}\tilde{U}^{(i)}(t)), \quad z^{(i)}(0) = 0 \\ & g_{j+1}^{(i)} = \mathbf{diag}(\eta^{(i)}) g_j^{(i)}, \quad 0 \preceq \eta^{(i)} \preceq 1, \quad g_n^{(i)} \succeq 0 \\ & i = 1, \ldots, k, \quad j = 1, \ldots, n-1, \quad t = 1, \ldots, n, \end{array} \tag{14}$$

where the variables are $\eta^{(i)} \in \mathbf{R}^m$, $G^{(i)} \in \mathbf{R}^{m\times n}$, $z^{(i)}(1), \ldots, z^{(i)}(n) \in \mathbf{R}^m$, $i = 1, \ldots, k$, and $x(1), \ldots, x(n) \in \mathbf{R}^m$. The problem data of Equation 14 are $w \in \mathbf{R}^k$, $y(1), \ldots, y(n) \in \mathbf{R}^m$, and $\tilde{U}^{(i)}(1), \ldots, \tilde{U}^{(i)}(n) \in \mathbf{R}^{m\times n}$ for all $i = 1, \ldots, k$ given by Equation 10.

The equivalence between the formulations (Equations 12, 14) can be easily verified: Notice that the constraints

$$g_{j+1}^{(i)} = \mathbf{diag}(\eta^{(i)}) g_j^{(i)}, \quad 0 \preceq \eta^{(i)} \preceq 1, \quad g_n^{(i)} \succeq 0 \tag{15}$$

for all $i = 1, \ldots, k$ and $j = 1, \ldots, n-1$ essentially enforce the columns of the matrix $G^{(i)}$ in Equation 14 to be nonnegative and decay *geometrically* with factor $\eta^{(i)}$ along each respective row. This is exactly the structure of the transformation F given by Equation 11. In particular, the vectors $\eta^{(i)}$ and $g_1^{(i)}$ in Equation 14 correspond to $(1-\alpha^{(i)})$ and $\mathbf{diag}(\alpha^{(i)})\,\mathbf{diag}(\beta^{(i)})$ in Equation 11, respectively.

Now we can see that the non-convexity of Equation 14 follows directly from the first equality constraint in Equation 15. Therefore, to convexify (Equation 14), we relax the constraints in Equation 15 to

$$g_1^{(i)} \succeq \cdots \succeq g_n^{(i)}, \quad g_n^{(i)} \succeq 0$$

and remove the variables $\eta^{(i)}$ for all $i = 1, \ldots, k$. This relaxation can be interpreted as simply requiring the entries of the matrices $G^{(i)}$ to decay along each respective row, but not necessarily geometrically. The resulting relaxed problem

$$\begin{array}{ll} \text{minimize} & -\sum_{t=1}^{n} \ell(x(t), y(t)) \\ \text{subject to} & x(t) = \left[\, z^{(1)}(t) \;\; \cdots \;\; z^{(k)}(t) \,\right] w \\ & z^{(i)}(t) = \mathbf{diag}(G^{(i)} \tilde{U}^{(i)}(t)) \\ & z^{(i)}(0) = 0, \quad g_1^{(i)} \succeq \cdots \succeq g_n^{(i)}, \quad g_n^{(i)} \succeq 0 \\ & i = 1, \ldots, k, \quad t = 1, \ldots, n \end{array} \tag{16}$$

with variables $G^{(i)} \in \mathbf{R}^{m \times n}$, $z^{(i)}(1), \ldots, z^{(i)}(n) \in \mathbf{R}^m$, $i = 1, \ldots, k$, and $x(1), \ldots, x(n) \in \mathbf{R}^m$, is now a convex optimization problem since the objective is convex and the inequality and equality constraints are all affine. We can then solve (Equation 16) efficiently in many ways, *e.g.*, via interior-point methods [25, 29, 30]. We should note that the time complexity of *evaluating* (Equation 16) is $O(mn^2k)$, which is higher than that of evaluating (Equation 9) by a factor of n. However, since Equation 16 is convex, it can be solved to global optimality in polynomial time. Specifically, the time complexity of *solving* (Equation 16) is $O(mn^2k)$ if a first-order method is used, and $O((mn^2k)^3)$ if a second-order method is used. In both cases, the time complexity of solving (Equation 16) is expected to be less than that of solving the nonconvex problem (Equation 9), which can be exponential in mnk in the worst case. See Section 6 for some numerical results on the solution time of Equations 9, 16 in applications.

Solving the relaxed problem (Equation 16) as a convex surrogate to the RL model fitting problem (Equation 12) (or equivalently to Equation 9) has several useful properties. Let $\alpha^{(i)}$ and $\beta^{(i)}$, $i = 1, \ldots, k$, be in the feasible set of Equation 12. Since Equation 16 is a relaxation of Equation 12, then there must exist feasible point $G^{(i)}$, $i = 1, \ldots, k$, to the problem (Equation 16), such that $G^{(i)} = F(\alpha^{(i)}, \beta^{(i)})$. Hence, the optimal value of the relaxed problem (Equation 16) gives a lower bound on the optimal value of the RL model fitting problem (Equation 12). In particular, suppose the problem (Equation 16) attains its optimum at $G^{(i)\star}$, $i = 1, \ldots, k$. If there exist $\eta^{(i)} \in \mathbf{R}^m$ such that $0 \preceq \eta^{(i)} \preceq 1$ and $g_{j+1}^{(i)\star} = \mathbf{diag}(\eta^{(i)}) g_j^{(i)\star}$ (where $g_j^{(i)\star}$ denotes the jth column of $G^{(i)\star}$) for all $i = 1, \ldots, k$ and $j = 1, \ldots, n-1$, i.e., the constraints (Equation 15) are satisfied, then such a lower bound to Equation 12 obtained from solving (Equation 16) is tight. In general, of course, this does not happen—at least some rows of $G^{(i)\star}$ do not decrease geometrically. For these cases, we could not say much regarding the tightness of such a lower bound since it is then dependent on the problem data (at least partially). Nevertheless, numerical examples show that fitting an RL model via Equation 16 has very similar performance to via Equation 12, but has the advantage of tractability. This can be partially explained as follows. When the RL model fitting problem is "hard," for example, when the subject's behavior is quite stochastic so the noise levels in the data are high, no fitting method (and, in particular, neither Equation 12 nor Equation 16) can do a good job at recovering the targeted variables. When the estimation problem is "easy," for example, when the subject's behavior is close to deterministic so the noise levels are low, even simple estimation methods (including via Equation 16) can do a good job at estimating the (sub)value functions and the RL model parameters. Therefore it is only problems in between "hard" and "easy" where we could possibly see a significant difference in fitting performance between Equations 12, 16. In this region, however, we observe from numerical experiments that they achieve very similar performance.

4.2 Recovering RL model parameters

Let $G^{(i)\star}$, $i = 1, \ldots, k$, be the optimal point of the relaxed RL model fitting problem (Equation 16). It is then sufficient for most applications to compute the corresponding (sub)value functions, i.e., $x^\star(t)$ and $z^{(i)\star}(t)$, which are the variables of most interest to researchers. However, it is sometimes still required to recover the full set of RL model parameters. Informally, we consider this step as finding a group of feasible $\alpha^{(i)\star}$ and $\beta^{(i)\star}$, such that the difference between the matrices $F(\alpha^{(i)\star}, \beta^{(i)\star})$ and $G^{(i)\star}$ for all $i = 1, \ldots, k$ is minimized. Note that if the resulting $\alpha^{(i)\star}$ and $\beta^{(i)\star}$ satisfy $F(\alpha^{(i)\star}, \beta^{(i)\star}) = G^{(i)\star}$ for all $i = 1, \ldots, k$, we can conclude that $\alpha^{(i)\star}$ and $\beta^{(i)\star}$ are the (globally) optimal point to Equation 12, although, again, this does not happen in general.

The process described above can be formulated mathematically as follows: Let $\bar{g}_j^{(i)\star} \in \mathbf{R}^n$ be the vector consisting of the jth row of the matrix $G^{(i)\star} \in \mathbf{R}^{m \times n}$, $i = 1, \ldots, k$, then the jth entry to the vectors $\alpha^{(i)\star}, \beta^{(i)\star} \in \mathbf{R}^m$ can be recovered by solving the problem

$$\begin{array}{ll} \text{minimize} & \left\| f(\alpha_j^{(i)}, \beta_j^{(i)}) - \bar{g}_j^{(i)\star} \right\|_2^2 \\ \text{subject to} & 0 \le \alpha_j^{(i)} \le 1, \quad \beta_j^{(i)} \ge 0 \end{array} \tag{17}$$

individually for all $i = 1, \ldots, k$, $j = 1, \ldots, m$. The problem (Equation 17) has variable $\alpha_j^{(i)}, \beta_j^{(i)} \in \mathbf{R}$ and data $\bar{g}^{(i)\star} \in \mathbf{R}^n$, and the transformation $f\colon \mathbf{R} \times \mathbf{R} \to \mathbf{R}^n$ is given by

$$f\colon (a, b) \mapsto (ab,\ (1-a)^1 ab,\ \cdots,\ (1-a)^{n-1} ab), \quad a, b \in \mathbf{R}.$$

Using a similar argumentation as in Section 3, we may see that the problem (Equation 17) is *not* convex, and hence, we consider finding a solution to Equation 17 via local minimization with

repeated initialization. In other words, we find several local minima of Equation 17 from different initial points, and select the one with the least objective value.

One may notice that by choosing a different penalty function for the difference between $f(\alpha_j^{(i)}, \beta_j^{(i)})$ and $\bar{g}_j^{(i)\star}$ in Equation 17, the problem of recovering the RL model parameters can be formulated as a convex program. We leave the corresponding discussion for this approach in Supplementary Section A, and will not consider it further in this study for the reasons listed there.

4.3 Truncation of the horizon

Notice that for all time steps $t = 1, \ldots, n$, we can approximate the calculation of each entry of the sub-value function $z^{(i)}(t)$ as

$$\begin{aligned} z_j^{(i)}(t) &= \sum_{\tau=1}^{t} (1-\alpha_j^{(i)})^{t-\tau} \alpha_j^{(i)} \beta_j^{(i)} u_j^{(i)}(\tau) \\ &\approx \sum_{\tau=t-p+1}^{t} (1-\alpha_j^{(i)})^{t-\tau} \alpha_j^{(i)} \beta_j^{(i)} u_j^{(i)}(\tau), \end{aligned} \tag{18}$$

for all $i = 1, \ldots, k$, $j = 1, \ldots, m$, since the term $(1-\alpha_j^{(i)})^{t-\tau}$ can be very close to zero for small τ. The approximation (Equation 18) can be interpreted as truncating the horizon to the last p steps when accumulating the sub-reward signals, instead of using the full history until the start of the episode. That is, the current sub-value function $z^{(i)}(t)$ is only dependent on the last p sub-reward signals $u^{(i)}(t-p+1), \ldots, u^{(i)}(t)$ (zero padding when $\tau \leq 0$). As two extreme examples, if $p = n$, no truncation is applied; if $p = 1$, the sub-value function $z^{(i)}(t)$ can be determined only from $u^{(i)}(t)$, i.e., there is no "memory" in the decision process.

The approximation (Equation 18) can be easily integrated into the RL model fitting problem (Equation 12) by replacing the transformation F defined in Equation 11 with $F_p \colon \mathbf{R}^m \times \mathbf{R}^m \to \mathbf{R}^{m \times p}$ given by

$$F_p \colon (a,b) \mapsto \begin{bmatrix} a_1 b_1 & (1-a_1)^1 a_1 b_1 & \cdots & (1-a_1)^{p-1} a_1 b_1 \\ \vdots & \vdots & & \vdots \\ a_m b_m & (1-a_m)^1 a_m b_m & \cdots & (1-a_m)^{p-1} a_m b_m \end{bmatrix}, \quad a, b \in \mathbf{R}^m.$$

Correspondingly, we replace $\tilde{U}^{(i)}(t)$ defined by Equation 10 with the submatrices $\tilde{U}_p^{(i)}(t) \in \mathbf{R}^{m \times p}$, which consist of the first p rows of $\tilde{U}^{(i)}(t)$ for all $t = 1, \ldots, n$. Finally, the relaxed Equation 14 and the Equation 17 for recovering RL model parameters can be easily adapted.

In practice, the horizon length p is a hyperparameter chosen by the user a priori (see Section 7.2.1 for some discussion on how to select the value of p in practice). When n is large and $p \ll n$, introducing the approximation (Equation 18) can significantly decrease the solving time, since the number of (scalar) variables in Equation 14 is reduced from kmn to kmp. This corresponds to a reduction of the time complexity of evaluating (Equation 16) from $O(mn^2k)$ to $O(mp^2k)$.

5 Implementation

In this section we describe our implementation of the ideas described in Section 4 for fitting RL model to behavioral data under multi-armed bandits. The source code has been collated into an open-source Python package `rlfit`, which is freely available online at https://github.com/nrgrp/rlfit.

The core module in the `rlfit` package is the `RLFit` class. At initialization, `RLFit` takes an integer and a boolean to specify the horizon length p (see Section 4.3) and whether the model parameters are shared across bandits (as described in Section 2.1), respectively. To fit the RL model to some data via solving the relaxed problem (Equation 16), the user calls the `fit` method. This method implements a solver for Equation 16 based on the domain-specific language `CVXPY` [31, 32] for convex optimization problems, and takes mainly the following arguments:

- `rewards`: A `numpy` array that has the shape `(n, m)` or a list of such `numpy` arrays (with each array representing a subreward signal), corresponding to the data $u^{(i)}(t) \in \mathbf{R}^m$, $i = 1, \ldots, k$, $t = 1, \ldots, n$, in Equation 9.
- `actions`: A `numpy` array with shape `(n, m)`, corresponding to the problem data $y(t) \in \{e_1, \ldots, e_m\} \subseteq \mathbf{R}^m$, $t = 1, \ldots, n$, in Equation 9.
- `w`: A number or a `numpy` array with shape `(k,)`, corresponding to the data $w \in \mathbf{R}^k$ in Equation 9. If a number is given, it is automatically transformed into a k-dimensional `numpy` array by repeating the same given number k times.

Then, if the user would like to recover the RL model parameters $\alpha^{(i)\star}$ and $\beta^{(i)\star}$, the method `fit_param` will be called subsequently. This method finds a solution to the problem (Equation 17) via repeated local minimization using `SciPy` [33]. Note that the argument `concurrent` for this method is set to `True` by default. This allows the problem (Equation 17) with different data, i.e., individual rows of $G^{(i)\star}$, to be minimized in parallel on multiple CPU cores. As a result, all entries of the parameter vectors $\alpha^{(i)\star}$ and $\beta^{(i)\star}$ can be recovered semi-simultaneously.

Once the RL model is fit, it can be used via either the `predict` or `score` method. The two arguments of `predict` are the data `rewards` and `w`, as in the `fit` method. This function returns the predicted action selection probability for all time steps, according to Equation 3, as well as the corresponding underlying (sub)value functions $x^\star(t)$ and $z^{(i)\star}(t)$, $i = 1, \ldots, k$, $t = 1, \ldots, n$. The `score` method takes the same first three arguments as the `fit` method, and evaluates the log-likelihood of the dataset, i.e., the negative objective of the problem (Equation 9). Note that the only prerequisite for using the `predict` and `score` method is to call the `fit` method during model fitting, and calling the `fit_param` method is not mandatory. If the RL model is fit only via the `fit` method, the functions implemented in `predict` and `score` will be based on the optimal point $G^{(i)\star}$, $i = 1, \ldots, k$, for the relaxed problem (Equation 16); if both `fit` and `fit_param` are called, the methods `predict` and `score` will instead use $\alpha^{(i)\star}$ and $\beta^{(i)\star}$, $i = 1, \ldots, k$, from the problem (Equation 17).

6 Empirical experiments

In this section, we evaluate our method proposed in Section 4 for fitting RL models under several popular multi-armed bandit environments. We compare our approach with two other solution methods that appear most commonly in the literature.

6.1 Environment setup

We consider the following three multi-armed bandit environment setups, with each assigned a three-capital-letter tag which we will refer to during subsequent discussion.

- BSC: The basic bandit environment defined according to the basic forgetting Q-learning model, given by Equations 1–3. The model fitting problem corresponds to Equation 6.
- IND: Extend the BSC setup by incorporating different learning rate α and reward sensitivity β for individual choices. The model fitting problem corresponds to Equation 7.
- SUB: Extend the IND setup by further incorporating two sub-reward signals and sub-value functions. The first subreward signal is the same as the reward signal used for BSC and IND, given by Equation 1. The second subreward signal is equal to $y(t)$ given by Equation 4 for all $t = 1, \ldots, n$ [this setup is sometimes considered to model the subject's behavior of repeating the last action under bandit tasks [21]]. The coefficient w used to combine the subvalue functions is defined as the most general case, i.e., $w = \mathbf{1}$. Then the model fitting problem corresponds to Equation 8 with $k = 2$.

Each of the three setups consists of a smaller (2-armed, $m = 2$) and a larger (10-armed, $m = 10$) version. The smaller version has a reward probability $(0.9, 0.1)$ for each possible action, and after each action selection, there is a 0.02 chance of shuffling the reward probabilities. Similarly, the larger version has the reward probability

$$(0.30, 0.27, 0.95, 0.67, 0.69, 0.29, 0.42, 0.05, 0.73, 1.00)$$

for each action, but there is no reward shuffling. The 2-armed bandit environment targets at simulating the animal behavior experiment task widely used for rodents, *e.g.*, in Hattori et al. [12] and Hamaguchi et al. [13], while the larger version aims at those tasks designed for humans, *e.g.*, in Kleespies et al. [19]. Although even the larger version "only" consists of 10 arms, it indeed covers almost all environments that appear in real-world behavioral experiments. For simplicity in description, we assign the tag "2AB" to the 2-armed bandit environment and "10AB" to the 10-armed setup.

For each environment setup, we collected a dataset consisting of 1,000 episodes, where each episode has 200 time steps (i.e., $n = 200$). For each episode, the model parameters α and β were randomly sampled from a uniform distribution defined on the intervals (or boxes) given by Table 1.

TABLE 1 Range of model parameters.

Environment	BSC	IND	SUB
2AB	$\alpha \in [0,1]$, $\beta \in [0,5]$	$\alpha \in [0,1]^2$, $\beta \in [0,5]^2$	$\alpha^{(1)} \in [0,1]^2$, $\beta^{(1)} \in [0,5]^2$ $\alpha^{(2)} \in [0,1]^2$, $\beta^{(2)} \in [0,2]^2$
10AB	$\alpha \in [0,1]$, $\beta \in [5,10]$	$\alpha \in [0,1]^{10}$, $\beta \in [5,10]^{10}$	$\alpha^{(1)} \in [0,1]^{10}$, $\beta^{(1)} \in [5,10]^{10}$ $\alpha^{(2)} \in [0,1]^{10}$, $\beta^{(2)} \in [0,5]^{10}$

6.2 Benchmarks

6.2.1 Direct local minimization with repeated initiation

As shown in Section 3, fitting an RL model to behavioral data consists of solving some instance of the non-convex optimization problem (Equation 9). In practice, one of the most direct approaches is to just apply local minimization methods repeatedly from different initial points [21]. Then, the locally optimal point with the best performance (in the case of Equation 9, this corresponds to the least cumulative negative log-likelihood value) is returned as the final (approximate) solution. Although there is a huge range of local minimization algorithms, one should note that directly minimizing (Equation 9) needs to include bound constraints on the model parameters α and β. For this purpose, the following solvers are widely considered and easily accessible via the Python library `SciPy` [33]: Nelder-Mead [34, 35], L-BFGS-B [36], TNC [37], SLSQP [38], Powell [39], trust region with constraints (Trust-Region) [40, 41], COBYLA [42], and COBYQA [43–45]. Note that the Nelder-Mead algorithm is a simplex method that does not support constraints inherently, but simply handles the box constraints by just clipping all vertices in the simplex based on the bounds. All the aforementioned solvers are evaluated in our empirical experiments (see Section 6.5 and Supplementary Tables S1–S6 for numerical results).

6.2.2 Bayesian modeling

Another popular approach for fitting RL models to behavioral data is to use Bayesian modeling. This method aims at estimating the posterior distribution of the model parameters given the observed subject actions $y(t)$ for all $t = 1, \ldots, n$. Then, the parameters corresponding to the highest posterior probability are selected as an approximate solution to the fitting problem.

With slight abuse of notation, let $\alpha = (\alpha^{(1)}, \ldots, \alpha^{(k)}) \in \mathbf{R}^{mk}$ and $\beta = (\beta^{(1)}, \ldots, \beta^{(k)}) \in \mathbf{R}^{mk}$, then the target distribution of the Bayesian estimation process is written as

$$p(\alpha, \beta \mid y(1), \ldots, y(n)) \propto p(y(1), \ldots, y(n) \mid \alpha, \beta) p(\alpha, \beta). \quad (19)$$

In general, the prior distribution $p(\alpha, \beta) = p(\alpha)p(\beta)$ is given by uniform distributions on the feasible set for the respective parameters, i.e.,

$$p(\alpha) = \mathcal{U}(0, \mathbf{1}), \quad p(\beta) = \mathcal{U}(0, \beta_{max}),$$

where $\beta_{max} \in \mathbf{R}^{mk}_{++}$ is the prior information for an upper bound on β. Under these uniform priors, roughly speaking, the goal of the Bayesian estimation process is equivalent to solving the RL model fitting problem (Equation 9) with an additional box constraint $\beta \preceq \beta_{\max}$ [25, Section 7.2]. Although one may consider some distribution with support $[0, \infty)$ to make $p(\beta)$ consistent with the corresponding constraints in Equation 9, it is then difficult to choose the parameters that control the shape of the priors. Besides, it may take more effort to accurately estimate the target posterior. Nevertheless, it has been reported empirically in the literature that the Bayesian modeling approach for fitting RL models is not very sensitive to the choice of the prior distribution [13, 19]. Finally, after obtaining the posterior $p(\alpha, \beta \mid y(1), \ldots, y(n))$, one may choose the parameters $\alpha^\star$ and $\beta^\star$ corresponding to the highest density as a solution to Equation 9.

In practice, the posterior distribution $p(\alpha, \beta \mid y(1), \ldots, y(n))$ given in Equation 19 is often estimated via Monte Carlo methods. This approach for fitting RL models is considered in several studies [13, 19], but it is generally less used in practice than direct local minimization methods. This is probably due to the difficulties in the implementation and debugging of a Monte Carlo method solver, even though the Python library PyMC [46] has significantly simplified this procedure. On the other hand, since the posterior distribution of the model parameters provides some global information about the solution space of Equation 9, RL model fitting via Bayesian modeling is expected to be more robust to the local minima issue than direct local minimization methods.

6.3 Solver configurations

In this section, we list the configuration details for our solution method introduced in Section 4, as well as the two benchmarks introduced in Section 6.2. When evaluating each solver, the RL model fitting was performed individually for each episode collected under all environments. The name tags used in the subsequent discussion and the configurations corresponding to each solver are listed as follows:

- MC: Bayesian estimation via Monte Carlo. The prior distributions for model parameters are set as uniform distributions on the range given in Table 1. For the fitting of all episodes, the number of sampled Markov chains was set to 4, with each consisting of 2,000 burn-in samples and 5,000 estimation samples.
- D-LOC: Directly solve the model fitting problem (Equation 9) via local minimization methods with repeated initializations. For the fitting of each episode under different environments, the initial values of the model parameters (i.e., the optimization variables) were sampled from a uniform distribution on the range given in Table 1. The number of repeated initializations is set to 5.
- CVX: Solve the corresponding convex relaxation problem (Equation 16). Note that this approach does not recover the RL model parameters $\alpha^{(i)\star}$ and $\beta^{(i)\star}$, $i = 1, \ldots, k$. Hence, the evaluation was only performed based on the estimated value functions $x^\star(t)$, $t = 1, \ldots, n$.
- CVX-T: The same as CVX, but with a truncated horizon (Equation 18), where $p = 5$.
- CVX-LOC: First perform CVX and then recover the model parameters by finding a solution to Equation 17 via local minimization methods. For the second local minimization step, when fit for each episode under different environments, the initial values of the model parameters were sampled from a uniform distribution on the range given in Table 1. The number of repeated initializations is again set to 5. Note that the concurrent computing feature for the parameter recovery problem (Equation 17) with different data, as introduced in Section 5, is enabled.
- CVX-LOC-T: The same as CVX-LOC, but with the truncated horizon approximation (Equation 18), where $p = 5$, for the first CVX step.

Note that to make the prior information compatible across different methods (in particular, between MC and the the other methods), in the numerical experiments, we adapted the constraints about $\beta^{(i)}$ in Equation 9 from $\beta^{(i)} \succeq 0$ to $\beta^{(i)}_{min} \preceq \beta^{(i)} \preceq \beta^{(i)}_{max}$ for all $i = 1, \ldots, k$. The bounds $\beta^{(i)}_{min}$ and $\beta^{(i)}_{max}$ are defined according to Table 1. In addition, when a local minimization step is required, all algorithms listed in the first paragraph of Section 6.2 were applied individually.

6.4 Evaluation metrics

To evaluate the performance of different solution methods, we consider the following two metrics. Firstly, the performance of recovering the value functions $x(t)$, $t = 1, \ldots, n$, is commonly measured by an indirect metric—the KL-divergence of the corresponding true and recovered action selection probability. Specifically, let

$$\pi(t) = \frac{\exp(x(t))}{\sum_{i=1}^{m} \exp(x_i(t))} \in \mathbf{R}^m, \quad t = 1, \ldots, n,$$

which is the probability of selecting individual actions at the tth time step. Then for each episode, we calculate the mean KL-divergence between the ground truth $\pi^{\mathrm{gt}}(t)$ and the estimated $\pi^\star(t)$ [obtained using the ground truth $x^{\mathrm{gt}}(t)$ and the recovered $x^\star(t)$ respectively], across $t = 1, \ldots, n$, i.e.,

$$\mathbf{E}\, D_{\mathrm{kl}}(\pi^{\mathrm{gt}}(t), \pi^\star(t)) = \frac{1}{n} \sum_{t=1}^{n} D_{\mathrm{kl}}(\pi^{\mathrm{gt}}(t), \pi^\star(t)).$$

Secondly, to measure the error of the recovered model parameters, we use $\|\alpha^{\mathrm{gt}} - \alpha^\star\|_2$ and $\|\beta^{\mathrm{gt}} - \beta^\star\|_2$, where α^{gt} and β^{gt} are the ground truth model parameters, and $\alpha^\star$ and $\beta^\star$ are the corresponding estimations. Note that here (as well as in the subsequent discussion) the notation α and β can refer to real numbers (for BSC setup), or vectors in $\mathbf{R}^m$ (for IND setup), or even the concatenated real vectors (for SUB setup), given by $\alpha = [\alpha^{(1)}, \ldots, \alpha^{(k)}] \in \mathbf{R}^{mk}$, $\beta = [\beta^{(1)}, \ldots, \beta^{(k)}] \in \mathbf{R}^{mk}$. The exact meaning of this notation can be determined from the context (or

the text). For the BSC setup, such a metric is simply the absolute value of the difference.

6.5 Numerical results

According to our experiments, the major difference between different local minimization solvers as listed in the first paragraph of Section 6.2 only appears in computing time (as shown in Supplementary Tables S1–S6). Therefore, we select the Trust-Region algorithm as the default solver in the following discussion (Figures 1, 2) because of its robustness, numerical stability, and broad applicability across different problem scales. Readers may refer to Supplementary Section B for the detailed numerical values corresponding to the (Figures 1, 2, as well as those results from the other local minimization solvers that are not included in the figures.

6.5.1 The 2AB environment

In general, under the 2AB environment and across all three setups (BSC, IND, and SUB), as expected, the MC method had the best performance both in recovering the value functions and the model parameters. All other methods had similar performance but slightly below MC (Figure 1, columns 1–3). Surprisingly, the additional truncated horizon approximation (Equation 18) in CVX-T and CVX-LOC-T did not result in a significant loss of solution accuracy. In terms of computational efficiency, our convex surrogate-based methods, CVX, CVX-LOC, CVX-T, and CVX-LOC-T, had faster computing time compared to D-LOC and MC methods across all setups. Specifically, under the BSC setup, the CVX-T method had the fastest solving time within 10^{-2} second; CVX, CVX-LOC, and CVX-LOC-T were slightly slower at the level of 10^{-1} second, while D-LOC and MC required even more computing time, for approximately 4×10^{-1} and 1.4 second(s), respectively. As the setup got more complicated from BSC to SUB, the required solving time also increased for all methods. In particular, the D-LOC method took even longer (in median) than MC for solving the fitting problem under the SUB setup (Figure 1, last column). This indicates that the D-LOC method has the highest sensitivity to the RL model scale, whereas our convex surrogate-based methods are the least influenced ones (which is a direct result of the concurrent implementation as introduced in Section 5).

6.5.2 The 10AB environment

The performance of different solution methods in terms of the fitting accuracy under the 10AB environment (Figure 2) is more or less similar to that from the 2AB environment. The slight differences only appear under the IND and SUB setups (Figures 2b, c). Specifically, the accuracy of recovering the subjects' action selection probability via CVX and CVX-T, measured by the mean KL-divergence $\mathbf{E}D_{\mathrm{kl}}(\pi^{\mathrm{gt}}(t), \pi^{\star}(t))$, has a larger median value and variance compared to the other methods. However, such a minor decrease does not have much influence in practice, since the real-world environments that subjects encounter are unlikely to be as complicated. One may notice that under the most complex setup SUB, our convex surrogate-based method also resulted in a larger fitting error regarding the accuracy of recovering the RL model parameters (Figure 2c, two middle columns). On the one hand, since the accuracy level of the subject's action selection probability does not vary much across different solution methods, it is reasonably expected that there exist multiple groups of RL model parameters that could lead to the same observed behavior. On the other hand, noticing that in this case the vectors for evaluating $\|\alpha^{\mathrm{gt}} - \alpha^{\star}\|_2$ and $\|\beta^{\mathrm{gt}} - \beta^{\star}\|_2$ are all in $\mathbf{R}^{20}$, the estimation error for each parameter is hence still below 1, which does not really influence the real-world applications either. The results for the computing time of different solution methods under the 10AB environment are almost exactly the same as those observed under 2AB, except that the solving time increased significantly for all methods (Figure 2, last column). Notably, with setups IND and SUB, the D-LOC method may take much longer than the MC method, but may result in even worse performance in recovering the value functions and model parameters.

7 Example

This section provides an example of using our provided `rlfit` package for fitting RL models to real-world behavioral data under bandits. We also compare our methods with the two benchmarks introduced in Section 6.2 in terms of the fitting performance and computing time.

7.1 The dataset

We consider the *two-armed bandit reversal learning task*, which is one of the standard protocols widely used in animal behavioral neuroscience research [12–14, 16, 21]. In particular, the dataset used here is from the previous study by De La Crompe et al. [14] and Zhu et al. [16].

The dataset consists of the behavior of 9 different mice performing the reversal learning task, with a total of 64 sessions (where each session corresponds to one episode in our previous discussion). In this task, the animal subject is presented with two water spouts, and it can select one of the spouts to receive a water reward. At the beginning of each session, a random spout is assigned a water reward, and the other spout is not rewarded. The animal may freely select either of the two spouts at each time step. As the session proceeds, the animal needs to learn which spout is currently rewarded. If the animal was able to collect 75% of the rewards in the last 15 time steps, the reward contingency will be reversed, i.e., the previously unrewarded spout becomes rewarded, and the previously rewarded spout becomes unrewarded. After each reversal, the animal needs to adapt its behavior to the new reward contingency. In this dataset, each session consists of 3 reversals, i.e., 4 blocks with different reward contingencies in total, and the total number of time steps in each session is roughly 100.

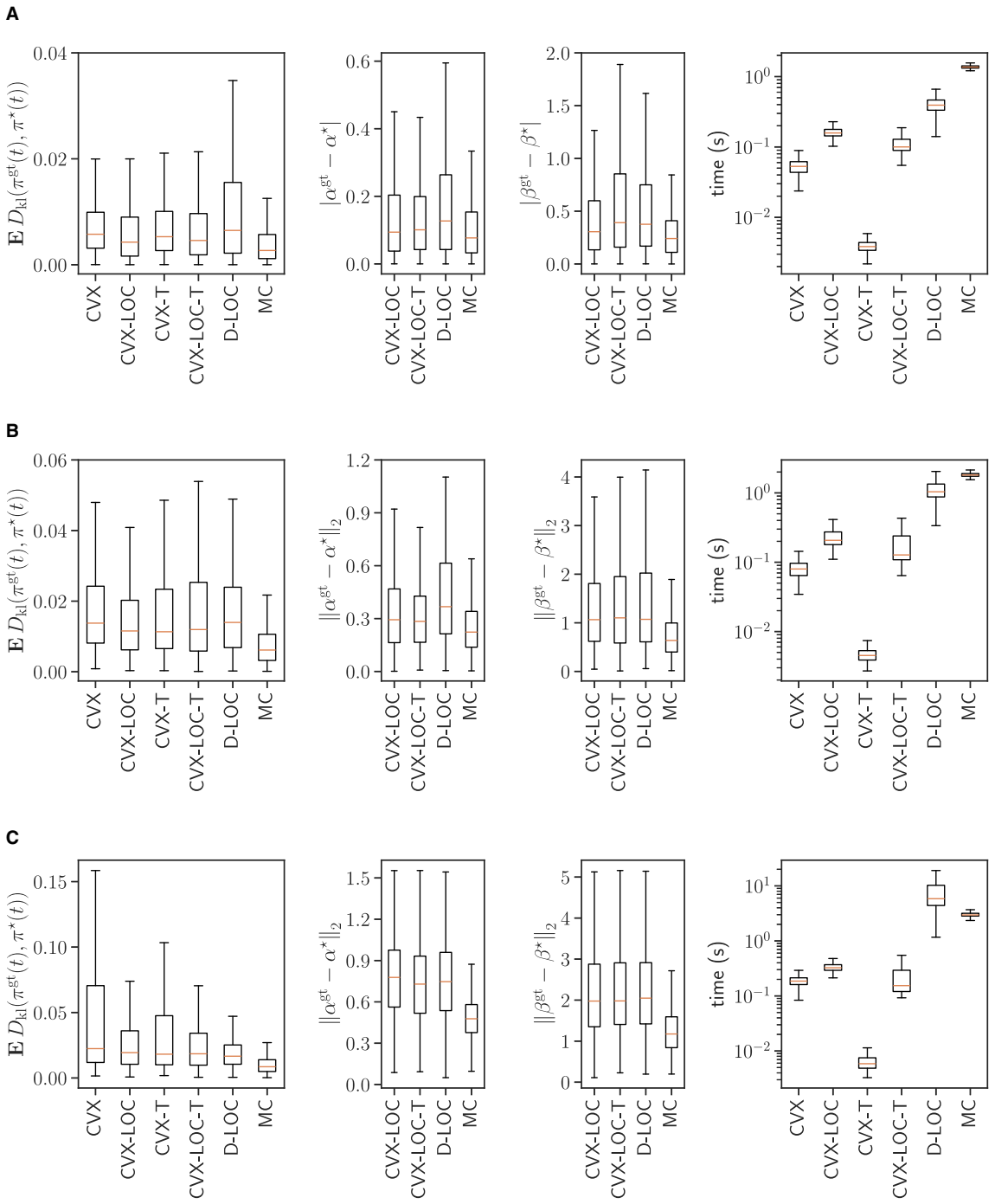

FIGURE 1
Performance of different solution methods for fitting RL models in the 2AB environment, under the **(A)** BSC, **(B)** IND, and **(C)** SUB setups. The left column shows the mean KL-divergence between the ground truth and recovered action selection probability; the middle column shows the error of the recovered model parameters; and the right column shows the computing time required for fitting the RL model.

7.2 Models and fitting

We consider fitting three different RL models under the setup assumptions BSC, IND, and SUB as presented in Section 6.1 to the given dataset, and compare the final results between the solution methods MC, D-LOC, CVX-T, and CVX-LOC-T as listed in Section 6.3.

7.2.1 Selecting the horizon length

The solution methods CVX-T and CVX-LOC-T require the user to specify a horizon length p for the truncated horizon approximation (Equation 18). There are several ways to select the value of p in practice. Empirically, it has been reported that a horizon length of 3 to 5 is often sufficient for fitting RL models

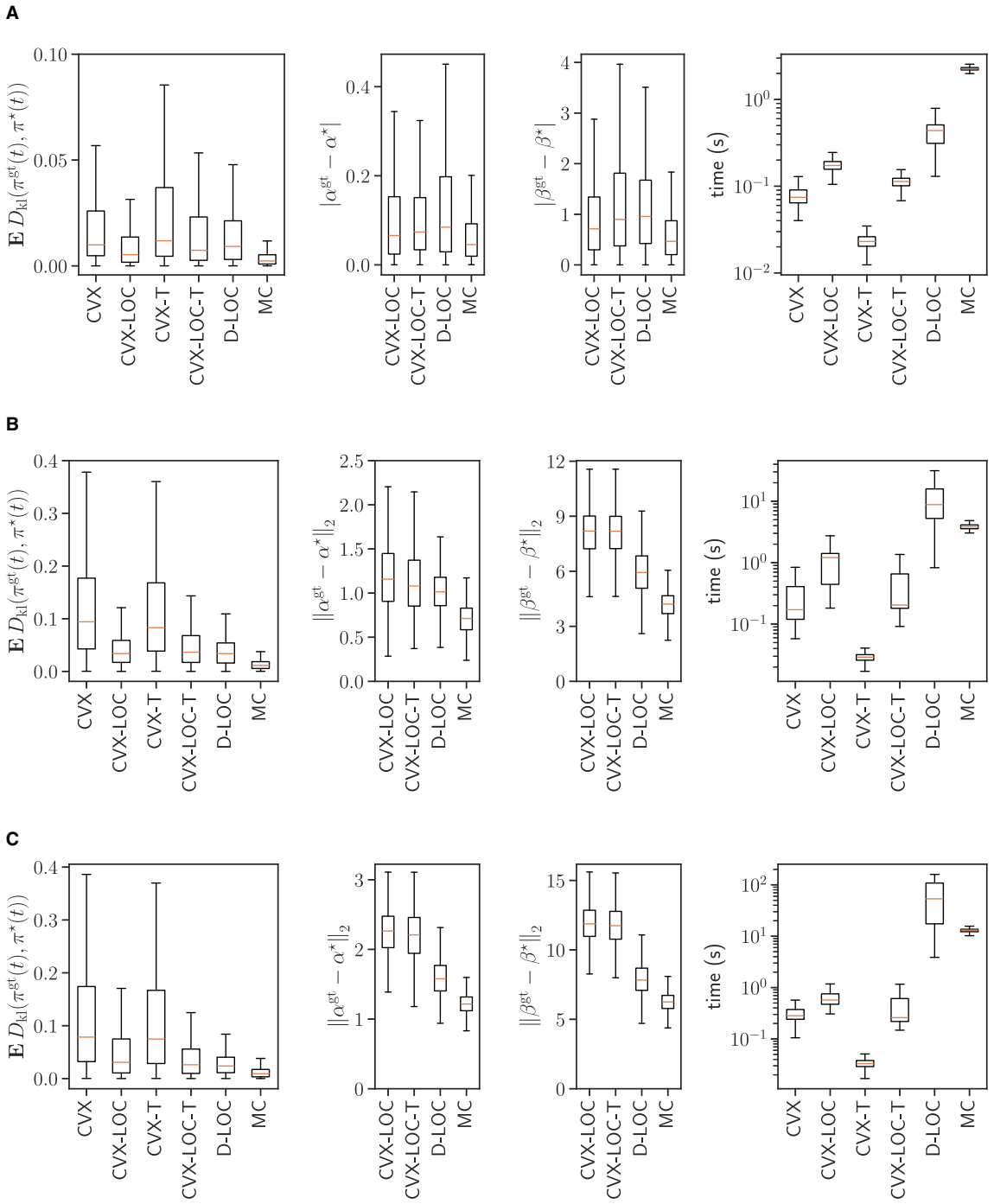

FIGURE 2
Performance of different solution methods for fitting RL models in the 10AB environment, under the **(A)** BSC, **(B)** IND, and **(C)** SUB setups. The left column shows the mean KL-divergence between the ground truth and recovered action selection probability; the middle column shows the error of the recovered model parameters; and the right column shows the computing time required for fitting the RL model.

to behavioral data under such tasks [12, 16, 21]. Quantitatively (or at least semi-quantitatively), one can also solve the problem (Equation 16) with approximation (Equation 18) under different horizon length p, and select the one that achieves the best balance between the optimal value of the RL model fitting problem (i.e., the best log-likelihood of the observed data) and the total number of problem variables.

Here, we consider the second approach as an example to select the horizon length p. We apply the method CVX-T with different horizon lengths p varying from 2 to 20 to fit the RL model under the BSC setup to the dataset. Then, the fitting performance is evaluated by the log-likelihood of the observed data, i.e., the negative of the objective of the problem (Equation 16). The results are shown in Figure 3. It is observed that, in general, the fitting performance gets

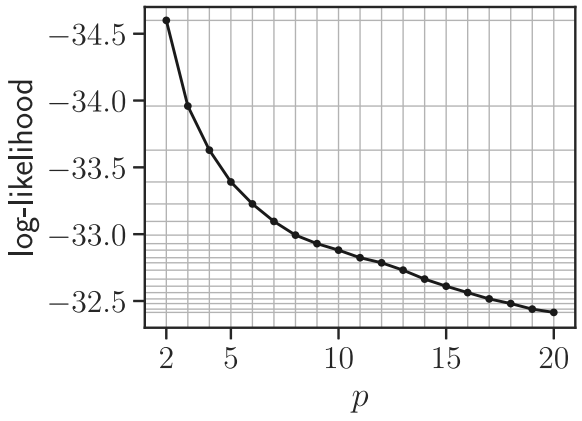

FIGURE 3
Log-likelihood values of the observed data, obtained from solving the convex surrogate (Equation 16) with approximation (Equation 18), under different horizon lengths p. Each point in the figure corresponds to the mean log-likelihood across all episodes in the dataset (with standard error approximately 1.9, not shown).

better as the horizon length p grows, but the improvement becomes less significant when p is larger than 5. This is consistent with the empirical observations in the literature, and hence we may select $p = 5$ as a reasonable value for the subsequent data analysis.

7.2.2 Model fitting

The following code snippet shows how to fit the RL model under the three setups using our package `rlfit`. For each session in the dataset, suppose the data is loaded into the arrays `rews` and `acts`, to fit the RL model under the BSC setup via CVX-T, we can use the following code:

```
import rlfit as rf
# define the BSC model with horizon length p
model = rf.RLFit(horizon_len=p, share_param=True)
# fit the model to the data
model.fit(rews, acts)
# evaluate the log-likelihood of the data under
    the fitted model
model.score(rews, acts)
```

To fit the RL model under the IND setup, we can simply set the argument `share_param` to `False` when defining the model, and to fit the RL model under the SUB setup, we need to pass `[rews, acts]` as the first argument to the `fit` method. Finally, to analyze the whole dataset, we can just repeat the above procedure for each session.

The code snippet above for the CVX-T method can be easily adapted to the CVX-LOC-T method by simply adding the following code after the `fit` method:

```
# recover the model parameters
model.fit_param(max_beta, number_repeats)
```

Here, the argument `max_beta` is a hyperparameter that defines an upper bound for the model parameter β, and `number_repeats` is the number of repeated initializations for the local minimization solver. All values of the hyperparameters used in this example (for all applied solution methods) remain the same as those described in Section 6.3. The implementation of the other two benchmarks roughly involves several tens of lines of Python code, and hence is not included here. We refer interested readers to the companion code repository of this article for the details.

7.3 Results

The best log-likelihood of the observed data under the fitted models via different solution methods, and the corresponding computing time, are shown in the left and right of Figure 4, respectively.

For all three setups, the CVX-T method achieves the best fitting performance in terms of the log-likelihood. This follows from the feature that this method [roughly, because of the approximation (Equation 18)] finds a lower bound for the original RL model fitting problem (Equation 9) via the convex surrogate (Equation 16). The corresponding compromise is then not fully recovering the model parameters and slightly violating the RL model assumptions. After recovering the model parameters via the local minimization step, the CVX-LOC-T method achieves a slightly worse fitting performance than CVX-T. This aligns with our expectation since the relaxations introduced by CVX-T are removed and the RL model assumptions are fully enforced. Nevertheless, the fitting performance of CVX-LOC-T is still comparable to the MC and D-LOC methods.

In terms of the computing time, the observations from Figure 4 are more or less the same as those observed in the numerical experiments with synthetic data: The CVX-T method is the fastest one, followed by CVX-LOC-T, while the D-LOC and MC methods, again, require significantly more computing time.

8 Conclusion and discussion

8.1 Summary of numerical results

Numerical results as listed in Sections 6.5, 7.3 suggest that, in general, our convex surrogate-based method for fitting RL models to behavioral data under multi-armed bandits achieves comparable performance as the other two benchmarks, but with significantly decreased computing time. Although the sampling-based Bayesian estimation method is most likely to result in the best performance in terms of fitting accuracy, its sampling process may last quite long. This disadvantage can be problematic when the behavioral dataset consists of a large number of episodes and the computing time is constrained. In comparison, our method achieves a good balance between the solution accuracy and the computing time.

Our observations also suggest that, although more as a byproduct, the most commonly used local minimization approach may not be ideal, as it achieves only moderate solution accuracy while requiring comparable or even greater computational time than the other methods. To the best of our knowledge, the preference for this direct method in the literature may be due to

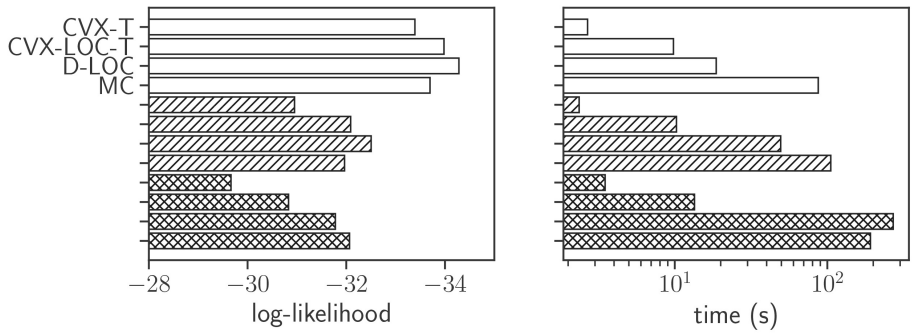

FIGURE 4
(Left) Best model fitting performance in terms of the log-likelihood of the observed data, obtained from different solution methods. The empty, line-hatched, and cross-hatched bars correspond to the BSC, IND, and SUB setups, respectively. The name of each method is only shown for the BSC setup, but the same order applies to the other two setups as well. Each bar in the figure shows the mean log-likelihood across all episodes in the dataset (with standard error approximately 1.9, 1.8, and 1.7 for BSC, IND, and SUB, respectively; not shown). **(Right)** Total computing time required for fitting different RL models to the whole dataset via different solution methods.

the relative simplicity of implementing and debugging, compared to the Monte Carlo sampling procedures. Our convex surrogate-based method, however, offers a relatively clean and straightforward procedure that can be easily implemented by users familiar with convex optimization. Moreover, we also provide a generic, well-documented Python package implementing our proposed solution method, allowing scientific researchers without prior knowledge about convex optimization to apply it in the analysis of their datasets directly.

8.2 Judging a heuristic fitting

Recall that our method, by solving the convex surrogate (Equation 16), computes a lower bound for the original problem (Equation 9). In particular, such a lower bound is computed for each specific problem instance (i.e., RL model structure) and data (i.e., observed subject behavior). This property can be applied to evaluate the sub-optimality of any other heuristic fitting results (at least semi-quantitatively).

Let J be a heuristic objective value of Equation 9 (obtained via any solution method), and let $J^\star$ be the global minimum. Suppose that by solving the convex program (Equation 16), we obtain a lower bound J^{lb} to Equation 9, then we have the inequalities

$$J^{\text{lb}} \leq J^\star \leq J.$$

If $J - J^{\text{lb}}$ is small, then we may conclude that such a heuristic solution is nearly (globally) optimal, and the bound J^{lb} is nearly tight. If $J - J^{\text{lb}}$ is big, then for this problem instance and data, either the fitting is poor, or the bound is poor (or both).

8.3 Parameter recovery

One may notice that our convex surrogate-based method via the problem (Equation 16) does not directly recover the parameters $\alpha^{(i)}$ and $\beta^{(i)}$ of the original RL model, but only estimates the value functions $x(t)$. Then the parameter recovery step is performed by solving the problem (Equation 17), which is a non-convex optimization problem. Although such a two-step method does not fully resolve the non-convexity issue in the original RL model fitting problem (Equation 9), it still has several advantages compared to directly solving (Equation 9).

Firstly, in many practical scenarios, researchers' main interest in fitting RL models to behavioral data is to recover the subjects' value functions [12, 15]. This can be directly obtained from a solution of the convex surrogate (Equation 16), without the need for the next step of parameter recovery. According to our numerical results, the accuracy of recovering the value functions from the convex surrogate in the most commonly used two-armed bandit settings is already comparable to the other two benchmarks (Figure 1, first column), but the computational cost is significantly reduced. Secondly, the RL model parameters recovery step via Equation 17 allows parallel computing of $\alpha^{(i)}$ and $\beta^{(i)}$ for different $i = 1, \ldots, k$. In contrast, directly solving Equation 9 requires all model parameters to be optimized together. Empirical results suggest that this concurrent feature of our method significantly improves computational efficiency, especially when the scale of the environment gets larger (Figures 1, 2, last column).

To sum up, the major contribution of our proposed method is not in finding a better (approximate) solution to the original non-convex RL model fitting problem that is closer to the global optimum, but in providing a computationally efficient approach that can be applied to larger-scale model setups and datasets.

8.4 Previous and related studies

8.4.1 Generalized linear models

Readers who are familiar with generalized linear models might notice that our proposed solution method for fitting RL models

can be interpreted as transforming the RL model into a multi-label logistic regression model, whose fitting problem is well known to be a convex optimization problem. A similar connection between these two types of models was previously observed and discussed by Beron et al. [21], although their focus was primarily on comparing different models of behavior under bandit settings, particularly in terms of interpretability and how well various behavioral characteristics were captured. Based on empirical data and observations, they argued that the generalized logistic regression model and the original RL model are approximately equivalent. In contrast, our convex analysis in this study offers a deeper theoretical insight, showing that the generalized logistic regression model is, in fact, a convex relaxation of the original RL model.

8.4.2 Inverse reinforcement learning

Inverse RL is a well-known problem in the field of machine learning, which aims at recovering the reward function or/and the policy of an agent based on its observed behavior [47, 48]. It is closely related to the problem of fitting RL models to behavioral data, as both problems involve inferring the underlying decision-making process of an agent from its demonstrations. Specifically, the problem considered in this study can be seen as a special case of inverse RL, applied to the setting of multi-armed bandits, where the state space and transition dynamics are trivial.

Inverse RL has been a very popular research topic in the machine learning community and has a wide range of applications in robotics, autonomous driving, and human-computer interaction. We refer the interested readers to Shao and Er [49, 50], Arora and Doshi [51], and Adams et al. [52] for some reviews on the recent developments and applications. Moreover, inverse RL has also been applied in characterizing animal and human behavior in neuroscience and psychology research [53–55]. One of the most notable recent advances in this direction is the *hierarchical* or *multi-intention* inverse RL framework, where the agent's behavior is assumed to be generated by multiple latent intentions or subgoals. These approaches have so far provided many novel explanations for the observed subject behavior in complex environments [16, 56–59]. Besides, hierarchical inverse RL has also attained much attention in the fields of, *e.g.*, engineering [60] and medicine [61, 62]. On the other hand, the model fitting problem of such hierarchical models is often very challenging, since it often involves solving a hierarchical optimization problem, where the inner and outer loop problems correspond to inverse RL and the latent intention estimation, respectively. The proposed method in this study can be used as a computationally efficient building block for these hierarchical models on multi-armed bandits. In particular, via the convex relaxation (Equation 16), the inner loop inverse RL problem of these hierarchical models can be solved efficiently. Most importantly, these ideas can be implemented and prototyped via the recently developed multi-convex programming frameworks [63, 64], and are hence easily actionable by practitioners.

8.4.3 Bandits and the inverse problems

The current major research focus on bandits is on the design of algorithms for solving the forward problem, i.e., how to find an optimal policy for a given bandit environment. Some good textbooks and reviews on this topic include those by Bubeck and Cesa-Bianchi [65], Slivkins [3], Lattimore and Szepesvári [1], and Zhou et al. [66]; just to mention a few. The inverse problem of bandits, in contrast, has received much less attention and is often considered as a special case of inverse RL. To the best of our knowledge, the only work that has directly considered the inverse problem of multi-armed bandits is from Hüyük et al. [67]. Hence, we believe that our study can be seen as a complementary contribution to this line of research.

Data availability statement

The datasets presented in this study can be found in online repositories. The names of the repository/repositories and accession number(s) can be found in the article/Supplementary material.

Author contributions

HZ: Conceptualization, Investigation, Software, Visualization, Writing – original draft, Writing – review & editing. JH: Validation, Writing – original draft, Writing – review & editing. BZ: Validation, Writing – original draft, Writing – review & editing. JB: Funding acquisition, Project administration, Resources, Supervision, Validation, Writing – original draft, Writing – review & editing.

Funding

The author(s) declared that financial support was received for this work and/or its publication. This work has been supported by the BrainLinks-BrainTools, funded by the Ministry of Science, Research and the Arts Baden-Württemberg within the sustainability program for projects of the Excellence Initiative II; by CRC/TRR 384 "IN-CODE," and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) -Project-ID 499552394 - SFB 1597. We further acknowledge support by the Open Access Publication Fund of the University of Freiburg.

Acknowledgments

An early version of this study appears as a preprint on *arXiv* [68], which is based on a very old working draft by the authors [69]. We thank Yuan Zhang for helpful discussions during the early stage of this work.

Conflict of interest

The author(s) declared that this work was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Generative AI statement

The author(s) declared that generative AI was not used in the creation of this manuscript.

Any alternative text (alt text) provided alongside figures in this article has been generated by Frontiers with the support of artificial intelligence and reasonable efforts have been made to ensure accuracy, including review by the authors wherever possible. If you identify any issues, please contact us.

Publisher's note

Supplementary material

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fams.2026.1762084/full#supplementary-material

References

1. Lattimore T, Szepesvári C. *Bandit Algorithms*. Cambridge: Cambridge University Press (2020). doi: 10.1017/9781108571401

2. Sutton RS, Barto AG. *Reinforcement Learning: An Introduction*. 2nd ed New York: MIT Press. (2018).

3. Slivkins A. Introduction to multi-armed bandits. *Found Trends Mach Learn*. (2019) 12:1–286. doi: 10.1561/2200000068

4. Samejima K, Ueda Y, Doya K, Kimura M. Representation of action-specific reward values in the striatum. *Science*. (2005) 310:1337–40. doi: 10.1126/science.1115270

5. Tai LH, Lee AM, Benavidez N, Bonci A, Wilbrecht L. Transient stimulation of distinct subpopulations of striatal neurons mimics changes in action value. *Nat Neurosci*. (2012) 15:1281–9. doi: 10.1038/nn.3188

6. Parker NF, Cameron CM, Taliaferro JP, Lee J, Choi JY, Davidson TJ, et al. Reward and choice encoding in terminals of midbrain dopamine neurons depends on striatal target. *Nat Neurosci*. (2016) 19:845–54. doi: 10.1038/nn.4287

7. Donahue CH, Liu M, Kreitzer AC. Distinct value encoding in striatal direct and indirect pathways during adaptive learning. *bioRxiv*. (2018). p. 277855. doi: 10.1101/277855

8. Ebitz RB, Albarran E, Moore T. Exploration disrupts choice-predictive signals and alters dynamics in prefrontal cortex. *Neuron*. (2018) 97:450–61. doi: 10.1016/j.neuron.2017.12.007

9. Miller KJ, Botvinick MM, Brody CD. From predictive models to cognitive models: Separable behavioral processes underlying reward learning in the rat. *bioRxiv*. (2018). doi: 10.1101/461129

10. Bari BA, Grossman CD, Lubin EE, Rajagopalan AE, Cressy JI, Cohen JY. Stable representations of decision variables for flexible behavior. *Neuron*. (2019) 103:922–33. doi: 10.1016/j.neuron.2019.06.001

11. Costa VD, Mitz AR, Averbeck BB. Subcortical substrates of explore-exploit decisions in primates. *Neuron*. (2019) 103:533–45. doi: 10.1016/j.neuron.2019.05.017

12. Hattori R, Danskin B, Babic Z, Mlynaryk N, Komiyama T. Area-specificity and plasticity of history-dependent value coding during learning. *Cell*. (2019) 177:1858–72. doi: 10.1016/j.cell.2019.04.027

13. Hamaguchi K, Takahashi-Aoki H, Watanabe D. Prospective and retrospective values integrated in frontal cortex drive predictive choice. *Proc Nat Acad Sci*. (2022) 119:e2206067119. doi: 10.1073/pnas.2206067119

14. De La Crompe B, Schneck M, Steenbergen F, Schneider A, Diester I. FreiBox: a versatile open-source behavioral setup for investigating the neuronal correlates of behavioral flexibility via 1-photon imaging in freely moving mice. *eNeuro*. (2023) 10:0469. doi: 10.1523/ENEURO.0469-22.2023

15. Hattori R, Hedrick NG, Jain A, Chen S, You H, Hattori M, et al. Meta-reinforcement learning via orbitofrontal cortex. *Nat Neurosci*. (2023) 26:2182–91. doi: 10.1038/s41593-023-01485-3

16. Zhu H, De La Crompe B, Kalweit G, Schneider A, Kalweit M, Diester I, et al. Multi-intention inverse q-learning for interpretable behavior representation. *arXiv preprint arXiv:2311.13870* (2024).

17. Daw ND, O'doherty JP, Dayan P, Seymour B, Dolan RJ. Cortical substrates for exploratory decisions in humans. *Nature*. (2006) 441:876–9. doi: 10.1038/nature04766

18. Vertechi P, Lottem E, Sarra D, Godinho B, Treves I, Quendera T, et al. Inference-based decisions in a hidden state foraging task: differential contributions of prefrontal cortical areas. *Neuron*. (2020) 106:166–76. doi: 10.1016/j.neuron.2020.01.017

19. Kleespies K, Paulus PC, Zhu H, Pargent F, Jakob M, Werle J, et al. Sleep resolves competition between explicit and implicit memory systems. *bioRxiv*. (2025). p. 2025-02. doi: 10.1101/2025.02.21.639581

20. Seymour B, Barbe M, Dayan P, Shiner T, Dolan R, Fink GR. Deep brain stimulation of the subthalamic nucleus modulates sensitivity to decision outcome value in Parkinson's disease. *Sci Rep*. (2016) 6:32509. doi: 10.1038/srep32509

21. Beron CC, Neufeld SQ, Linderman SW, Sabatini BL. Mice exhibit stochastic and efficient action switching during probabilistic decision making. *Proc Nat Acad Sci*. (2022) 119:e2113961119. doi: 10.1073/pnas.2113961119

22. Ito M, Doya K. Validation of decision-making models and analysis of decision variables in the rat basal ganglia. *J Neurosci*. (2009) 29:9861–74. doi: 10.1523/JNEUROSCI.6157-08.2009

23. Aylward J, Valton V, Ahn WY, Bond RL, Dayan P, Roiser JP, et al. Altered learning under uncertainty in unmedicated mood and anxiety disorders. *Nat Hum Behav*. (2019) 3:1116–23. doi: 10.1038/s41562-019-0628-0

24. MaBouDi H, Marshall JAR, Barron AB. Honeybees solve a multi-comparison ranking task by probability matching. *Proc R Soc B Biol Sci*. (2020) 287:20201525. doi: 10.1098/rspb.2020.1525

25. Boyd S, Vandenberghe L. *Convex Optimization*. Cambridge: Cambridge University Press. (2004). doi: 10.1017/CBO9780511804441

26. Rockafellar RT. *Convex Analysis*. Princeton: Princeton University Press. (1970).

27. Nesterov Y. *Introductory Lectures on Convex Optimization: A Basic Course*. Cham: Springer. (2004). doi: 10.1007/978-1-4419-8853-9

28. Houska B, Chachuat B. Global optimization in Hilbert space. *Mathem Progr*. (2019) 173:221–49. doi: 10.1007/s10107-017-1215-7

29. Nesterov Y, Nemirovskii A. *Interior-point Polynomial Algorithms in Convex Programming*. Philadelphia: SIAM. (1994). doi: 10.1137/1.9781611970791

30. Nocedal J, Wright SJ. *Numerical Optimization*. Cham: Springer. (1999). doi: 10.1007/b98874

31. Diamond S, Boyd S, CVXPY. A Python-embedded modeling language for convex optimization. *J Mach Learn Res*. (2016) 17:1–5.

32. Agrawal A, Verschueren R, Diamond S, Boyd S. A rewriting system for convex optimization problems. *J Control Decis*. (2018) 5:42–60. doi: 10.1080/23307706.2017.1397554

33. Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nat Methods*. (2020) 17:261–72. doi: 10.1038/s41592-020-0772-5

34. Nelder JA, Mead R. A simplex method for function minimization. *Comput J*. (1965) 7:308–13. doi: 10.1093/comjnl/7.4.308

35. Gao F, Han L. Implementing the Nelder-Mead simplex algorithm with adaptive parameters. *Comput Optim Appl*. (2012) 51:259–77. doi: 10.1007/s10589-010-9329-3

36. Zhu C, Byrd RH, Lu P, Nocedal J. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans Mathem Softw*. (1997) 23:550–60. doi: 10.1145/279232.279236

37. Nash SG. Newton-type minimization via the Lanczos method. *SIAM J Numer Anal*. (1984) 21:770–88. doi: 10.1137/0721052

38. Kraft D. *A Software Package for Sequential Quadratic Programming*. DLR German Aerospace Center-Institute for Flight Mechanics (1988). DFVLR-FB 88-28.

39. Powell MJD. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *Comput J*. (1964) 7:155–62. doi: 10.1093/comjnl/7.2.155

40. Branch MA, Coleman TF Li Y. A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems. *SIAM J Sci Comput*. (1999) 21:1–23. doi: 10.1137/S1064827595289108

41. Conn AR, Gould NIM, Toint PL. *Trust Region Methods*. Philadelphia: SIAM. (2000). doi: 10.1137/1.9780898719857

42. Powell MJD. A direct search optimization method that models the objective and constraint functions by linear interpolation. In: Gomez S, Hennart JP, editors. *Advances in Optimization and Numerical Analysis*. Cham: Springer (1994). p. 51–67. doi: 10.1007/978-94-015-8330-5_4

43. Powell MJD, UOBYQA. Unconstrained optimization by quadratic approximation. *Mathem Progr*. (2002) 92:555–82. doi: 10.1007/s101070100290

44. Ragonneau TM. *Model-Based Derivative-Free Optimization Methods and Software*. Hong Kong, China: Department of Applied Mathematics, The Hong Kong Polytechnic University. (2022).

45. Ragonneau TM, Zhang Z. *COBYQA Version 1.1.2* (2024). Available online at: https://www.cobyqa.com (Accessed March 19, 2026).

46. Abril-Pla O, Andreani V, Carroll C, Dong L, Fonnesbeck JJ, Kochurov M, et al. PyMC: A modern, and comprehensive probabilistic programming framework in Python. *PeerJ Comput Sci*. (2023) 9:e1516. doi: 10.7717/peerj-cs.1516

47. Ng AY, Russell S. Algorithms for inverse reinforcement learning. In: *International Conference on Machine Learning*. (2000).

48. Abbeel P, Ng AY. Apprenticeship learning via inverse reinforcement learning. In: *International Conference on Machine Learning*. (2004). doi: 10.1145/1015330.1015430

49. Shao Z, Er MJ. A survey of inverse reinforcement learning techniques. *Int J Intell Comput Cybern*. (2012) 5:293–311. doi: 10.1108/17563781211255862

50. Shao Z, Er MJ. A review of inverse reinforcement learning theory and recent advances. In: *IEEE Congress on Evolutionary Computation*. IEEE (2012). p. 1–8. doi: 10.1109/CEC.2012.6256507

51. Arora S, Doshi P. A survey of inverse reinforcement learning: challenges, methods and progress. *Artif Intell*. (2021) 297:103500. doi: 10.1016/j.artint.2021.103500

52. Adams S, Cody T, Beling PA. A survey of inverse reinforcement learning. *Artif Intell Rev*. (2022) 55:4307–46. doi: 10.1007/s10462-021-10108-x

53. Kalweit G, Kalweit M, Alyahyay M, Jaeckel Z, Steenbergen F, Hardung S, et al. NeuRL: Closed-form inverse reinforcement learning for neural decoding. *arXiv preprint arXiv:2204.04733*. (2022).

54. Alyahyay M, Kalweit G, Kalweit M, Karvat G, Ammer J, Schneider A, et al. Mechanisms of premotor-motor cortex interactions during goal directed behavior. *bioRxiv*. (2023). doi: 10.1101/2023.01.20.524944

55. Ruiz-Serra J, Harré MS. Inverse reinforcement learning as the algorithmic basis for theory of mind: current methods and open problems. *Algorithms*. (2023) 16:68. doi: 10.3390/a16020068

56. Ke J, Wu F, Wang J, Markowitz J, Wu A. Inverse reinforcement learning with switching rewards and history dependency for characterizing animal behaviors. *arXiv preprint arXiv:2501.12633*. (2025).

57. Wang J, Ke J, Dai B, Wu A. Learning task-agnostic motifs to capture the continuous nature of animal behavior. *arXiv preprint arXiv:2506.15190*. (2025).

58. Shehab ML, Aspeel A, Ozay N. Learning reward machines from partially observed policies. *arXiv preprint arXiv:2502.03762* (2025).

59. Hausladen CI, Schubert MH, Engel C. Identifying latent intentions via inverse reinforcement learning in repeated linear public good games. *arXiv preprint arXiv:2601.08803*. (2026).

60. Feng H, Jiang K, Zhao Y, Tian K, Tang B. Deep multi-intentional inverse reinforcement learning for cognitive multi-function radar inverse cognition. *Expert Syst Appl*. (2025) 293:128599. doi: 10.1016/j.eswa.2025.128599

61. Kalweit G, Ullrich E, Boedecker J, Mertelsmann R, Kalweit M. AI in optimized cancer treatment: Laying the groundwork for interdisciplinary progress. *Oxford Open Immunol*. (2025) 6:iqaf004. doi: 10.1093/oxfimm/iqaf004

62. Vogt Y, Kalweit M, Alieva M, Ullrich E, Boedecker J, Kalweit G. AI in modular concepts of natural killer cell therapy. In: *Natural Killer Cells: At the Forefront of Modern Immunology*. Springer (2025). p. 1–33. doi: 10.1007/978-3-662-68816-8_68-1

63. Shen X, Diamond S, Udell M, Gu Y, Boyd S. Disciplined multi-convex programming. In: *29th Chinese Control and Decision Conference*. IEEE (2017). p. 895–900. doi: 10.1109/CCDC.2017.7978647

64. Zhu H, Boedecker J. Disciplined biconvex programming. *arXiv preprint arXiv:2511.01813*. (2025).

65. Bubeck S, Cesa-Bianchi N. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Found Trends Mach Learn*. (2012) 5:1–122. doi: 10.1561/2200000024

66. Zhou P, Wei H, Zhang H. Selective reviews of bandit problems in AI via a statistical view. *Mathematics*. (2025) 13:665. doi: 10.3390/math13040665

67. Hüyük A, Jarrett D, van der Schaar M. Inverse contextual bandits: learning how behavior evolves over time. In: *International Conference on Machine Learning* (2022). p. 9506–9524.

68. Zhu H, Hoffmann J, Zhang B, Boedecker J. Fitting reinforcement learning model to behavioral data under bandits. *arXiv preprint arXiv:2511.04454*. (2025).

69. Zhu H, Boedecker J. Solving inverse problem for multi-armed bandits via convex optimization. *arXiv preprint arXiv:2511.04454*. (2025).