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
[2]Department of Computer Science, University of Freiburg

November 6, 2025

Abstract

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, followed by 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 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.

Contents

1 Introduction

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

Multi-armed bandit behavioral tasks. The class of multi-armed bandit tasks is an experimental paradigm used to investigate a wide range of animals' decision making processes that has been widely used in neuroscience [SUDK05, TLB+12, PCT+16, DLK18, EAM18, MBB18, BGL+19, CMA19, HDB+19, HTAW22, DLCSS+23, HHJ+23, ZCK+24], psychology [DOD+06, VLS+20, KPZ+25], and medical [SBD+16] 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 randomly after some trials or regularly according to some criteria (which are sometimes referred to as *dynamic bandits*), or introducing an external cue, *e.g.*, image, sound, *etc.*, to individual choices, so that 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).

RL model for decision making under bandits. To quantitatively characterize the decision making behavior under bandit tasks, RL has emerged as one of the most popular mathematical models during the last decade. The following procedure describes the most basic instance of the class of RL models, namely the *forgetting Q-learning* model [ID09, BNLS22]: 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

$$u_i(t) = \begin{cases} 1 & \text{if action } i \text{ was selected } \textit{and} \text{ rewarded} \\ 0 & \text{otherwise,} \end{cases} \tag{1.1}$$

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

$$x(t) = x(t-1) + \alpha(\beta u(t) - x(t-1)), \tag{1.2}$$

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:

$$\mathbf{prob}(a(t) = i) = \frac{\exp x_i(t)}{\sum_{j=1}^m \exp x_j(t)}, \quad i = 1, \ldots, m, \quad t = 0, \ldots, n. \tag{1.3}$$

(Define $\mathbf{prob}(a(t) = i) = 1/m$ for all $i = 1, \ldots, m$ if $\sum_{i=1}^m \exp x_i(t) = 0$.) Except for the aforementioned basic example, several extensions to the forgetting Q-learning model have also been incorporated such that more subtle behavior properties can be included in the modeling (see §2.2 for more details).

RL model fitting problem. Behavioral science researchers, *i.e.*, the users of the RL models, are interested in obtaining individual subject's behavior characterization, given the observed behavior outcome. In particular, for the case of the forgetting Q-learning model defined by (1.1) to (1.3), the goal is to recover the model parameters α and β as well as the value functions $x(t)$, $t = 1, \dots, n$, 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, and the variables being the model parameters and the value function at each time step. (A formal definition of the RL model fitting problem will be introduced in §2.)

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 §6 for more detail). In this paper, we aim at filling these gaps with the following three folds of contributions:

- Firstly, we formalize the mathematical optimization problem corresponding to the fitting problem of a wide range of the most widely used RL models (*cf.*, §2).
- Then, in §3, we provide theoretical analysis about the convexity properties of the RL model fitting problems according to our problem formulation.
- Finally, based on the theoretical results, in §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 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 to 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 (1.1) to (1.3). Recall that here the objective is to maximize the likelihood of observing the given data $a(t)$ and $u(t)$ for $t = 1, \dots, n$, with the variables being the value functions $x(t)$ and the model parameters α and β. First, notice that (1.2) can be written as

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

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

$$y_i(t) = \begin{cases} 1 & a(t) = i \\ 0 & \text{otherwise}, \end{cases} \tag{2.1}$$

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

$$\ell(x(t), y(t)) = \log \left(y(t)^T \left(\frac{\exp x(t)}{\sum_{i=1}^m \exp x_i(t)} \right) \right). \tag{2.2}$$

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

$$\begin{array}{ll} \text{minimize} & -\sum_{t=1}^n \ell(x(t), y(t)) \\ \text{subject to} & x(t) = (1-\alpha)x(t-1) + \alpha\beta u(t), \quad t = 1, \dots, n \\ & x(0) = 0, \quad 0 \le \alpha \le 1, \quad \beta \ge 0, \end{array} \tag{2.3}$$

where the problem variables are $\alpha, \beta \in \mathbf{R}$ and $x(1), \dots, x(n) \in \mathbf{R}^m$, the problem data are $y(t), u(t) \in \mathbf{R}^m$, and each log-likelihood term in the objective is given by (2.2).

2.2 Extensions

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 (1.1). For example, some models add 'punishment' to the unrewarded choices, *i.e.*, replacing all the zeros in (1.1) with -1. This type of extensions only changes the problem data of (2.3), which does not influence the properties of the fitting problem itself.

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 [HDB+19, HHJ+23], *i.e.*, for each entry $x_i(t)$ of the value function $x(t) \in \mathbf{R}^m$, $i = 1, \dots, m$, 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 $\beta_i \in [0, \infty)$, $i = 1, \dots, m$. Hence, the model fitting problem (2.3) is now extended to be

$$\begin{array}{ll} \text{minimize} & -\sum_{t=1}^n \ell(x(t), y(t)) \\ \text{subject to} & x(t) = \mathbf{diag}(1-\alpha)x(t-1) + \mathbf{diag}(\alpha)\,\mathbf{diag}(\beta)u(t), \quad t = 1, \dots, n \\ & x(0) = 0, \quad 0 \preceq \alpha \preceq 1, \quad \beta \succeq 0 \end{array} \tag{2.4}$$

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

Subreward signals and subvalue functions. Another type of extension to the basic forgetting Q-learning model that appears in applications assumes that there exist multiple *subreward signals* $u^{(1)}(t), \dots, u^{(k)}(t) \in \mathbf{R}^m$ [BNLS22], corresponding to multiple *subvalue functions* $z^{(1)}(t), \dots, z^{(k)}(t) \in \mathbf{R}^m$, which are 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 (1.3) 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 (2.3) now becomes

$$\begin{array}{ll} \text{minimize} & -\sum_{t=1}^{n} \ell(x(t), y(t)) \\ \text{subject to} & x(t) = \left[\begin{array}{ccc} z^{(1)}(t) & \cdots & z^{(k)}(t) \end{array}\right] w \\ & z^{(i)}(t) = (1-\alpha^{(i)}) z^{(i)}(t-1) + \alpha^{(i)} \beta^{(i)} u^{(i)}(t) \\ & z^{(i)}(0) = 0, \quad 0 \le \alpha^{(i)} \le 1, \quad \beta^{(i)} \ge 0 \\ & i = 1, \ldots, k, \quad t = 1, \ldots, n, \end{array} \tag{2.5}$$

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 (2.3), (2.4), and (2.5), can be written as the following general form:

$$\begin{array}{ll} \text{minimize} & -\sum_{t=1}^{n} \ell(x(t), y(t)) \\ \text{subject to} & x(t) = \left[\begin{array}{ccc} z^{(1)}(t) & \cdots & z^{(k)}(t) \end{array}\right] w \\ & z^{(i)}(t) = \mathbf{diag}(1-\alpha^{(i)}) z^{(i)}(t-1) + \mathbf{diag}(\alpha^{(i)})\, \mathbf{diag}(\beta^{(i)}) u^{(i)}(t) \\ & z^{(i)}(0) = 0, \quad 0 \preceq \alpha^{(i)} \preceq 1, \quad \beta^{(i)} \succeq 0 \\ & i = 1, \ldots, k, \quad t = 1, \ldots, n, \end{array} \tag{2.6}$$

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 (2.6) reduces to (2.4); 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 (2.6) reduces to (2.5); and by combining the two additional requirements above together, we have the basic forgetting Q-learning model fitting problem (2.3). The time complexity of *evaluating* the objective and constraints of (2.6) is $O(mnk)$.

3 Convexity properties

To analyze the convexity properties of the general RL model fitting problem (2.6), we start by eliminating the recursive expression about $z^{(i)}(t)$. Consider the jth entry of the vectors $z^{(i)}(0), \ldots, z^{(i)}(n)$, we have

$$z_j^{(i)}(0) = 0$$

$$
\begin{aligned}
z_j^{(i)}(1) &= (1-\alpha_j^{(i)})z_j^{(i)}(0) + \alpha_j^{(i)}\beta_j^{(i)}u_j^{(i)}(1) = \alpha_j^{(i)}\beta_j^{(i)}u_j^{(i)}(1) \\
z_j^{(i)}(2) &= (1-\alpha_j^{(i)})z_j^{(i)}(1) + \alpha_j^{(i)}\beta_j^{(i)}u_j^{(i)}(2) = (1-\alpha_j^{(i)})\alpha_j^{(i)}\beta_j^{(i)}u_j^{(i)}(1) + \alpha_j^{(i)}\beta_j^{(i)}u_j^{(i)}(2) \\
z_j^{(i)}(3) &= (1-\alpha_j^{(i)})z_j^{(i)}(2) + \alpha_j^{(i)}\beta_j^{(i)}u_j^{(i)}(3) \\
&= \left(1-\alpha_j^{(i)}\right)^2\alpha_j^{(i)}\beta_j^{(i)}u_j^{(i)}(1) + (1-\alpha_j^{(i)})\alpha_j^{(i)}\beta_j^{(i)}u_j^{(i)}(2) + \alpha_j^{(i)}\beta_j^{(i)}u_j^{(i)}(3) \\
&\vdots \\
z_j^{(i)}(n) &= (1-\alpha_j^{(i)})z_j^{(i)}(n-1) + \alpha_j^{(i)}\beta_j^{(i)}u_j^{(i)}(n) = \sum_{t=1}^{n}\left(1-\alpha_j^{(i)}\right)^{n-t}\alpha_j^{(i)}\beta_j^{(i)}u_j^{(i)}(t),
\end{aligned}
$$

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

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

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

$$
\tilde{U}^{(i)}(t) = \begin{bmatrix} U^{(i)}(t) \\ 0 \end{bmatrix} \in \mathbf{R}^{n\times m}, \quad U^{(i)}(t) = \begin{bmatrix} u^{(i)}(t)^T \\ \vdots \\ u^{(i)}(1)^T \end{bmatrix} \in \mathbf{R}^{t\times m}. \tag{3.1}
$$

The matrices $\tilde{U}^{(i)}(t)$ given by (3.1) can be interpreted as follows: For each $t = 1, \dots, n$, the matrix $\tilde{U}^{(i)}(t)$ is formed by padding an $(n-t)\times m$ matrix with all entries zero to the end of the corresponding subreward signal matrix $U^{(i)}(t)$, such that $\tilde{U}^{(i)}(t) \in \mathbf{R}^{n\times m}$. Define the transformation $F\colon \mathbf{R}^m \times \mathbf{R}^m \to \mathbf{R}^{m\times n}$, given by

$$
F\colon (a,b) \mapsto \begin{bmatrix} a_1b_1 & (1-a_1)^1a_1b_1 & \cdots & (1-a_1)^{n-1}a_1b_1 \\ \vdots & \vdots & \ddots & \vdots \\ a_mb_m & (1-a_m)^1a_mb_m & \cdots & (1-a_m)^{n-1}a_mb_m \end{bmatrix}, \quad a,b \in \mathbf{R}^m, \tag{3.2}
$$

the problem (2.6) can be written as

$$
\begin{array}{ll}
\text{minimize} & -\sum_{t=1}^n \ell(x(t), y(t)) \\
\text{subject to} & x(t) = \begin{bmatrix} z^{(1)}(t) & \cdots & z^{(k)}(t) \end{bmatrix} w \\
& z^{(i)}(t) = \mathbf{diag}(F(\alpha^{(i)}, \beta^{(i)})\tilde{U}^{(i)}(t)) \\
& z^{(i)}(0) = 0, \quad 0 \preceq \alpha^{(i)} \preceq 1, \quad \beta^{(i)} \succeq 0 \\
& i = 1, \dots, k, \quad t = 1, \dots, n,
\end{array} \tag{3.3}
$$

where the variables are $\alpha^{(i)}, \beta^{(i)} \in \mathbf{R}^m$, $z^{(i)}(1), \dots, z^{(i)}(n) \in \mathbf{R}^m$ for all $i = 1, \dots, k$, and $x(1), \dots, x(n) \in \mathbf{R}^m$; the problem data are $w \in \mathbf{R}^k$, $y(1), \dots, y(n) \in \mathbf{R}^m$, and $\tilde{U}^{(i)}(1), \dots, \tilde{U}^{(i)}(n) \in \mathbf{R}^{m\times n}$ for all $i = 1, \dots, k$ given by (3.1).

We can now easily check the convexity of (2.6) via the equivalent form (3.3). Note that the objective function in (3.3) can be written as

$$-\sum_{t=1}^{n} \ell(x(t), y(t)) = -\sum_{t=1}^{n} \log\left(\frac{y(t)^T \exp x(t)}{\sum_{i=1}^{m} \exp x_i(t)}\right) \\ = -\sum_{t=1}^{n}\left(y(t)^T x(t) - \log\sum_{i=1}^{m} \exp x_i(t)\right), \tag{3.4}$$

where the second equality is from the fact that, by (2.1), the vector $y(t) \in \{e_1, \ldots, e_m\} \subseteq \mathbf{R}^m$ is a standard basis vector for all $t = 1, \ldots, n$. Since in the last expression of (3.4), the first term $y(t)^T x(t)$ is affine and the second log-sum-exp term is convex [BV04, §3.1], by basic convex analysis [Roc70, BV04], we conclude that the objective of (3.3) is convex in the variables $x(t)$. It follows then immediately that the problem (3.3) 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 (3.2) is *not* affine. Hence, we conclude that the RL model fitting problem (3.3) is *not* convex. As a result, even if the time complexity of evaluating the objective and constraints of (2.6) is only $O(mnk)$, the complexity of *solving* it to global optimality in the worse case can be exponential in mnk.

4 Solution method

4.1 The convex surrogate

Analysis in §3 indicates that by relaxing the transformation F given by (3.2) to be affine, the RL model fitting problem can then be convexified. To make such relaxation more explicit, we consider an equivalent formulation of (3.3) 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 (3.3) is then equivalent to

$$\begin{array}{ll} \text{minimize} & -\sum_{t=1}^{n} \ell(x(t), y(t)) \\ \text{subject to} & x(t) = \begin{bmatrix} z^{(1)}(t) & \cdots & z^{(k)}(t) \end{bmatrix} w \\ & z^{(i)}(t) = \mathbf{diag}(G^{(i)}\tilde{U}^{(i)}(t)), \quad z^{(i)}(0) = 0 \\ & g_{j+1}^{(i)} = \mathbf{diag}(\eta^{(i)}) g_j^{(i)}, \quad 0 \preceq \eta^{(i)} \preceq 1, \quad g_n^{(i)} \succeq 0 \\ & i = 1, \ldots, k, \quad j = 1, \ldots, n-1, \quad t = 1, \ldots, n, \end{array} \tag{4.1}$$

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 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 (3.1). Now it can be easily seen that the nonconvexity of (4.1) is from the constraints

$$g_{j+1}^{(i)} = \mathbf{diag}(\eta^{(i)})g_j^{(i)}, \quad 0 \preceq \eta^{(i)} \preceq 1, \quad i = 1, \ldots, k, \quad j = 1, \ldots, n-1, \tag{4.2}$$

(or really, the first equality constraint in (4.2)), which require the entries of the matrices $G^{(i)}$ to decay *geometrically* along each respective row. To convexify (4.1), we relax the constraints in (4.2) to

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

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

$$\begin{array}{ll} \text{minimize} & -\sum_{t=1}^{n} \ell(x(t), y(t)) \\ \text{subject to} & x(t) = \left[\begin{array}{ccc} z^{(1)}(t) & \cdots & z^{(k)}(t) \end{array}\right] w \\ & z^{(i)}(t) = \mathbf{diag}(G^{(i)}\tilde{U}^{(i)}(t)) \\ & z^{(i)}(0) = 0, \quad g_1^{(i)} \succeq \cdots \succeq g_n^{(i)}, \quad g_n^{(i)} \succeq 0 \\ & i = 1, \ldots, k, \quad t = 1, \ldots, n \end{array} \tag{4.3}$$

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 (4.3) efficiently in many ways, *e.g.*, via interior-point methods [NN94, NW99, BV04]. We should note that the time complexity of *evaluating* (4.3) is $O(mn^2k)$, which is higher than that of evaluating (2.6) by a factor of n. However, since (4.3) is convex, it can be solved to global optimality in polynomial time. Specifically, the time complexity of *solving* (4.3) 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 (4.3) is expected to be less than that of solving the nonconvex problem (2.6), which can be exponential in mnk in the worse case. See §6 for some numerical results on the solution time of (2.6) and (4.3) in applications.

There are several properties by solving the relaxed problem (4.3) as a convex surrogate to the RL model fitting problem (3.3) (or equivalently to (2.6)). Let $\alpha^{(i)}$ and $\beta^{(i)}$, $i = 1, \ldots, k$, be in the feasible set of (3.3), since (4.3) is a relaxation of (3.3), then there must exist feasible points $G^{(i)}$ for all $i = 1, \ldots, k$ to (4.3), such that $G^{(i)} = F(\alpha^{(i)}, \beta^{(i)})$. Hence, the optimal value of the relaxed problem (4.3) gives a lower bound on the optimal value of the RL model fitting problem (3.3). In particular, suppose the problem (4.3) achieves optimal 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 $i = 1, \ldots, k$, $j = 1, \ldots, n-1$, *i.e.*, the constraints (4.2) are satisfied, then such a lower bound to (3.3) obtained from solving (4.3) is tight. In general, of course, this does not happen — at least some rows of $G^{(i)\star}$ do not decrease geometrically, and in 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 (4.3) has very similar performance to via (3.3), 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, *i.e.*, the noise levels in the data are high, no fitting method (and, in particular, neither (3.3) nor (4.3)) 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, *i.e.*, the noise levels are low, even simple estimation methods (including via (4.3)) can do a good job at estimating the (sub)value functions and the RL model parameters. So it is only problems in between 'hard' and 'easy' where we could possibly see a significant difference in fitting performance between (3.3) and (4.3), whereas in this region, we observe from numerical experiments that they achieve very similar performance.

4.2 Recovering RL model parameters

Let $G^{(i)\star}$, $i = 1, \dots, k$, be the optimal point of the relaxed RL model fitting problem (4.3), it is then sufficient for most applications to compute the corresponding (sub)value functions, $x^\star(t)$ and $z^{(i)\star}(t)$, which are the most interested variables for 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 $i = 1, \dots, k$ 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 $i = 1, \dots, k$, we can conclude that $\alpha^{(i)\star}$ and $\beta^{(i)\star}$ are the (globally) optimal point to (3.3), 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 $G^{(i)\star} \in \mathbf{R}^{m\times n}$, $i = 1, \dots, k$, then the jth entry to the vectors $\alpha^{(i)\star}, \beta^{(i)\star} \in \mathbf{R}^m$ can be recovered by solving the problem

$$\begin{array}{ll} \text{minimize} & \left\| f(\alpha_j^{(i)}, \beta_j^{(i)}) - \bar{g}_j^{(i)\star} \right\|_2^2 \\ \text{subject to} & 0 \le \alpha_j^{(i)} \le 1, \quad \beta_j^{(i)} \ge 0 \end{array} \tag{4.4}$$

individually for all $i = 1, \dots, k$, $j = 1, \dots, m$, with optimization 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 §3, we may see that the problem (4.4) is *not* convex, and hence, we consider finding a solution to (4.4) via local minimization with repeated initialization, *i.e.*, finding several local minima of (4.4) 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 (4.4), the problem of recovering the RL model parameters can be formulated as a convex program. We leave the corresponding discussion for this approach in §A, and will not consider it further in this paper for the reasons listed there.

4.3 Truncation of the horizon

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

$$z_j^{(i)}(t) = \sum_{\tau=1}^{t} (1-\alpha_j^{(i)})^{t-\tau} \alpha_j^{(i)} \beta_j^{(i)} u_j^{(i)}(\tau) \approx \sum_{\tau=t-p+1}^{t} (1-\alpha_j^{(i)})^{t-\tau} \alpha_j^{(i)} \beta_j^{(i)} u_j^{(i)}(\tau), \tag{4.5}$$

for all $i = 1, \dots, k$, $j = 1, \dots, m$, since the term $(1 - \alpha_j^{(i)})^{t-\tau}$ can be very close to zero for small τ. The approximation (4.5) can be interpreted as truncating the horizon to the last p steps when accumulating the subreward signals, instead of using the full history until the start of the episode, *i.e.*, the current subvalue function $z^{(i)}(t)$ is only dependent on the last p subreward signals $u^{(i)}(t-p+1), \dots, u^{(i)}(t)$ (zero padding when $\tau \le 0$). As two extreme examples, if $p = n$, no truncation is applied; if $p = 1$, the subvalue function $z^{(i)}(t)$ can be determined only from $u^{(i)}(t)$, *i.e.*, there is no "memory" in the decision process.

The approximation (4.5) can be easily integrated into the RL model fitting problem (3.3) by replacing the transformation F defined by (3.2) with $F_p \colon \mathbf{R}^m \times \mathbf{R}^m \to \mathbf{R}^{m \times p}$, given by

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

and replacing the problem data $\tilde{U}^{(i)}(t)$ defined by (3.1) with the submatrices $\tilde{U}_p^{(i)}(t) \in \mathbf{R}^{m \times p}$ consisting of the first p rows of $\tilde{U}^{(i)}(t)$ for all $t = 1, \dots, n$. Correspondingly, the relaxed problem (4.1) and the problem (4.4) for recovering RL model parameters can be easily adapted.

In practice, the horizon length p is a hyperparameter chosen by the user in prior. When n is large and $p \ll n$, introducing the approximation (4.5) can significantly decrease the solving time since the number of (scalar) variables in (4.1) is reduced from kmn to kmp, which corresponds to a reduction of the time complexity of evaluating (4.3) from $O(mn^2k)$ to $O(mp^2k)$.

5 Implementation

In this section we describe our implementation of the ideas described in §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, which, at initialization, takes an integer and a boolean to specify the horizon length p (*cf.* §4.3) and whether the model parameters are shared across bandits (as the basic forgetting Q-learning model described in §2.1), respectively. To fit the RL model to some data via solving the relaxed problem (4.3), the user calls the `fit` method, which implements a solver for (4.3) based on the domain specific language `CVXPY` [DB16, AVDB18] for convex optimization problems. The `fit` method 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 $u^{(i)}(t) \in \mathbf{R}^m$, $i = 1, \dots, k$, $t = 1, \dots, n$, in (2.6).
- `actions`: A `numpy` array with shape `(n, m)`, corresponding to the problem data $y(t) \in \{e_1, \dots, e_m\} \subseteq \mathbf{R}^m$, $t = 1, \dots, n$, in (2.6).
- `w`: A number or a `numpy` array with shape `(k,)`, corresponding to the data $w \in \mathbf{R}^k$ in (2.6). 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, which finds a solution to the problem (4.4) via repeated local minimization using `SciPy` [VGO+20]. Note that the argument `concurrent` for this method is set to `True` by default, which allows the distribution of the solution finding process of the problem (4.4) with different data, *i.e.*, individual rows of $G^{(i)\star}$, to multiple processes, such that all entries of the parameter vectors $\alpha^{(i)\star}$ and $\beta^{(i)\star}$ can be recovered in parallel.

Once the RL model is fit, it can be used via either the `predict` or `score` method. The `predict` method takes the data `rewards` and `w` as in the `fit` method, and returns the predicted probability of selecting individual actions for all time steps, according to (1.3), 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 of the objective of the problem (2.6). Note that calling the `fit_param` method during model fitting is not mandatory to use the `predict` and `score` method. 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 (4.3); 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 (4.4).

6 Empirical experiments

In this section, we evaluate our method proposed in §4 for fitting RL models under several popular multi-armed bandit environments, in comparison 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 (1.1) to (1.3). The model fitting problem corresponds to (2.3).
- IND: Extend the BSC setup by incorporating different learning rate α and reward sensitivity β for individual choices. The model fitting problem corresponds to (2.4).
- SUB: Extend the IND setup by further incorporating two subreward signals and subvalue functions. The first subreward signal is the same as the reward signal used for BSC and IND, given by (1.1), whereas the second subreward signal is equal to $y(t)$ given by (2.1) for all $t = 1, \ldots, n$ (such a setup is sometimes considered to model the subject's behavior of repeating the last action under bandit tasks [BNLS22]). 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 (2.5) 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 [HDB+19, HTAW22], while the larger version aims at those tasks designed for human, *e.g.*, in [KPZ+25]. 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 1000 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.

	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

Direct local minimization with repeated initiation. As shown in §3, fitting an RL model to behavioral data consists in solving some instance of a nonconvex optimization problem (2.6). One of the most direct approaches for solving nonconvex optimization problems that appears in application [BNLS22] is to just apply local minimization methods repeatedly from different initial points, and return the local optimal point with the best performance, *i.e.*, for (2.6), with the least cumulative negative log-likelihood value. Albeit the existence of a huge range of local minimization algorithms, one should note that directly minimizing (2.6) needs to include bound constraints on the model parameters α and β, in which case the following solvers are widely considered and easily accessible via the Python library `SciPy` [VGO+20]: Nelder-Mead [NM65, GH12], L-BFGS-B [ZBLN97], TNC [Nas84], SLSQP [Kra88], Powell [Pow64], trust region with constraints (Trust-Region) [BCL99, CGT00], COBYLA [Pow94], and COBYQA [Pow02, Rag22, RZ24]. 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 simplex based on the bounds. All the aforementioned solvers are evaluated in our empirical experiments (see §6.5 and §B for numerical results).

Probabilistic inference via Monte Carlo. Instead of trying to find a (locally) optimal point of (2.6) as the direct local minimization method, Monte Carlo methods aim at estimating the posterior distribution of the model parameters given the observed subject actions $y(t)$

for all $t = 1, \dots, n$. With slight abuse of notation, let $\alpha = (\alpha^{(1)}, \dots, \alpha^{(k)}) \in \mathbf{R}^{mk}$, $\beta = (\beta^{(1)}, \dots, \beta^{(k)}) \in \mathbf{R}^{mk}$, the target distribution of the inference process is written as

$$p(\alpha, \beta \mid y(1), \dots, y(n)) \propto p(y(1), \dots, y(n) \mid \alpha, \beta) p(\alpha, \beta).$$

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. Note that here the user needs to specify a slightly tighter constraint $\beta \in [0, \beta_{\max}]$ rather than $\beta \in [0, \infty)$ as in (2.6). Although one may consider some distribution with support $[0, \infty)$ to make the prior $p(\beta)$ consistent with the corresponding constraints in (2.6), it is then difficult to come up with reasonable parameters that control the shape of the prior distribution, and it may take more effort to obtain enough samples for an accurate estimation of the target posterior. After obtaining the posterior $p(\alpha, \beta \mid y(1), \dots, y(n))$, one may choose the parameters $\alpha^\star$ and $\beta^\star$ corresponding to the highest density as a solution to (2.6). In practice, Monte Carlo methods for obtaining a solution to the fitting problem of RL models are less used [HTAW22] than direct local minimization methods due to its difficulties in implementation and debugging, even though the Python library `PyMC` [APAC+23] has significantly simplified this procedure. On the other hand, Monte Carlo methods can in theory provide the most accurate solution to (2.6), at the price of computing time.

6.3 Solver configurations

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

- MC: Probabilistic inference method 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 2000 burn-in samples and 5000 estimation samples.

- D-LOC: Directly solve the model fitting problem (2.6) 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, with the number of repeated initializations set to 5.

- CVX: Solve the corresponding convex relaxation problem (4.3). Note that this approach does not recover the RL model parameters $\alpha^{(i)\star}$ and $\beta^{(i)\star}$, $i = 1, \dots, k$, and hence the evaluation was only performed based on the estimated value functions $x^\star(t)$, $t = 1, \dots, n$.

- CVX-T: The same as CVX, but with a truncated horizon (4.5), where $p = 5$.

- CVX-LOC: First perform CVX and then recover the model parameters by finding a solution to (4.4) 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, with the number of repeated initializations set to 5. Note that the concurrent computing feature for the parameter recovery problem (4.4) with different data as introduced in §5 is enabled.
- CVX-LOC-T: The same as CVX-LOC, but with the truncated horizon approximation (4.5), 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 (2.6) from $\beta^{(i)} \succeq 0$ to $\beta^{(i)}_{\min} \preceq \beta^{(i)} \preceq \beta^{(i)}_{\max}$ for all $i = 1, \ldots, k$, where $\beta^{(i)}_{\min}$ 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 §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 $\pi^{\text{gt}}(t)$ and the estimated $\pi^\star(t)$ (obtained using the ground truth $x^{\text{gt}}(t)$ and the recovered $x^\star(t)$ respectively), across $t = 1, \ldots, n$, *i.e.*,

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

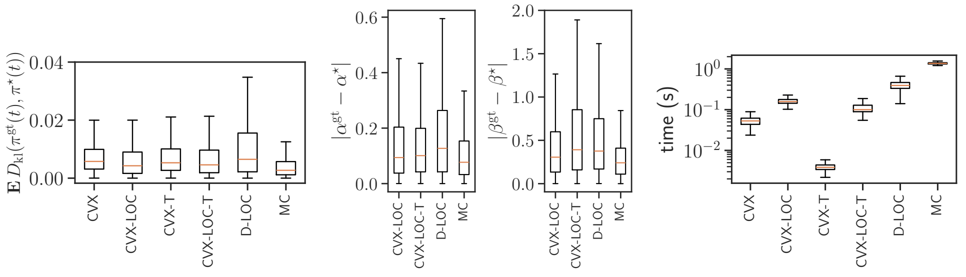
Secondly, to measure the error of the recovered model parameters, we calculate the ℓ_2-norm of the difference between the ground truth α^{gt}, β^{gt}, and the corresponding $\alpha^\star$, $\beta^\star$, respectively, *i.e.*, $\|\alpha^{\text{gt}} - \alpha^\star\|_2$ and $\|\beta^{\text{gt}} - \beta^\star\|_2$. 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 these notation can be determined from the context (or the text). For the BSC setup, such metric is simply the absolute value of the difference.

6.5 Numerical results

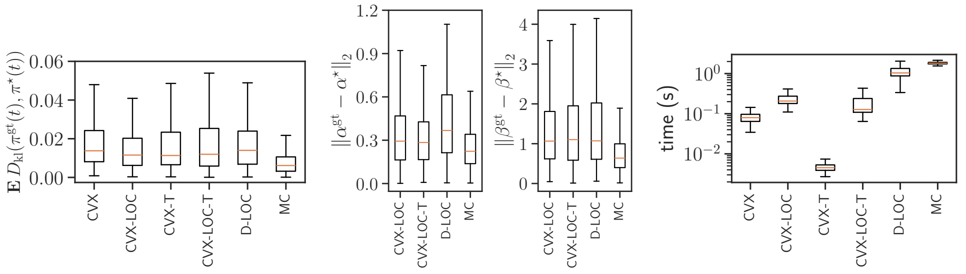
Note that when reporting the numerical results, since the major difference between different local minimization solvers as listed in the first paragraph of §6.2 only appears in computing time (as shown in tables 2–7), we select the Trust-Region algorithm as the default solver in the following discussion (figures 1 and 2) because of its robustness, numerical stability, and broad applicability across different problem scales. Readers may refer to the tables 2–7 of §B for the detailed numerical values corresponding to the figures 1 and 2, as well as those results from the other local minimization solvers that are not included in the figures.

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, while all other methods had similar performance but slightly below MC (figure 1, columns 1–3). Surprisingly, the additional truncated horizon approximation (4.5) 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, and 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 for solution finding of (4.4) with different data).

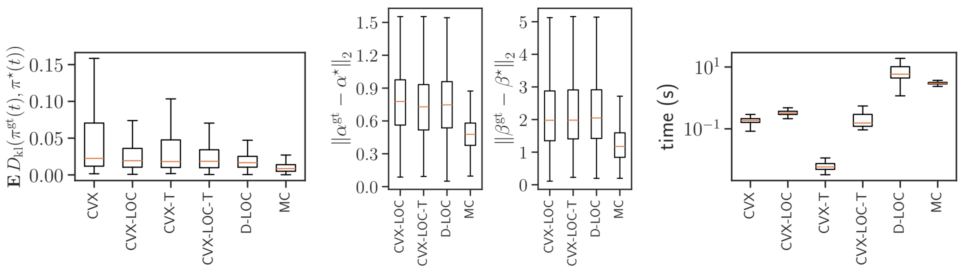
The 10AB environment. The performance of different solution methods regarding the fitting accuracy under the 10AB environment (figure 2) is more or less similar to those shown for the 2AB environment in figure 1, with slight differences only appearing under the IND and SUB setups (figures 2b and 2c). Specifically, under these two setups, we may notice that the accuracy of recovering the subjects' action selection probability from 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, although such a minor decrease regarding the fitting accuracy of the value functions does not have much influence in practice, especially that the real world environments where the subjects can handle are not quite likely to become so complicated. In addition, one may also 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 about the subject's action selection probability does not vary much across different solution methods, especially between our methods and D-LOC, it is reasonably expected that there exist multiple groups of RL model parameters that could lead to the same observed behavior under such a complex environment setup. 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 $\mathbf{R}^{20}$, we may conclude that the mean componentwise error, *i.e.*, for each number of the parameters, is still below 1, which does not really influence the real world applications either. The results for the computing efficiency of different solution methods under the 10AB environment are almost exactly the same to those observed under the 2AB environment, except that the solving time increased significantly for all methods compared to the 2AB case (figure 2, last column). Notably, with setups IND and SUB, the D-LOC method may take much longer than the MC method, but result in even worse performance in recovering the value functions and model parameters.



(a) The BSC setup.

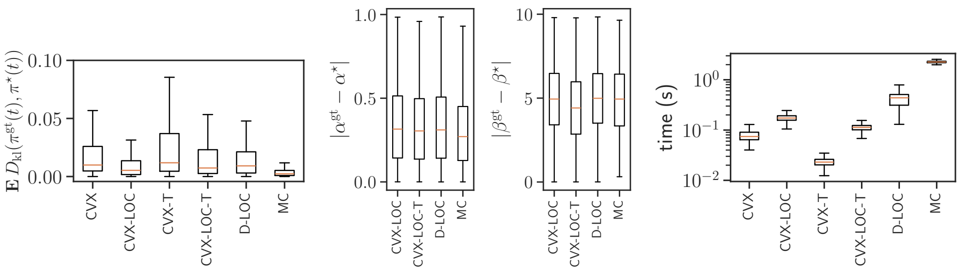


(b) The IND setup.

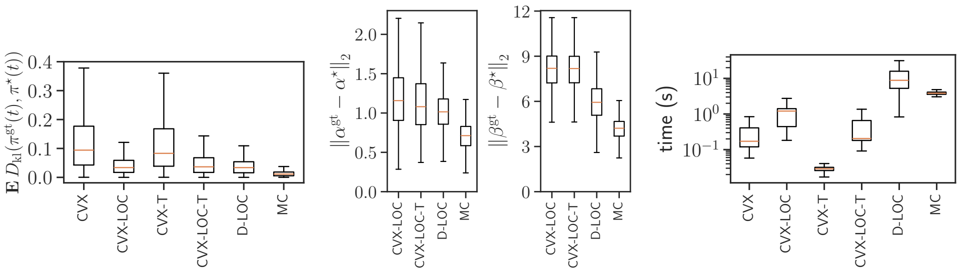


(c) The SUB setup.

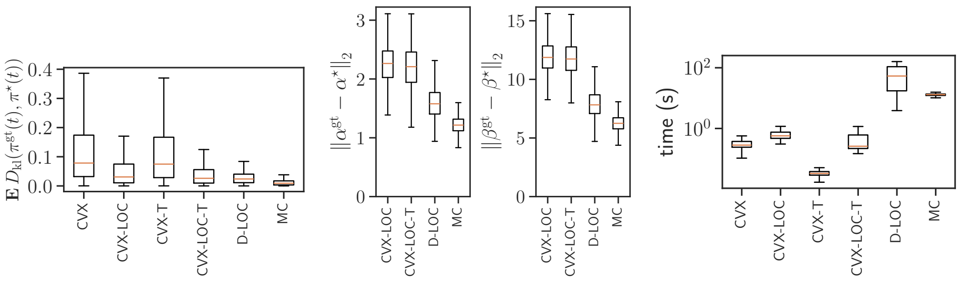
Figure 1 Performance of different solution methods under the 2AB environment.



(a) The BSC setup.



(b) The IND setup.



(c) The SUB setup.

Figure 2 Performance of different solution methods under the 10AB environment.

7 Conclusion and discussion

7.1 Summary of numerical results

Numerical results as listed in §6.5 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 method of probabilistic inference via Monte Carlo could lead to the best performance in the fitting accuracy, its sampling process may last quite long, which 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 approach of directly solving the RL model fitting problem (2.6) through repeated local minimization may not be ideal, as it achieves only moderate solution accuracy while requiring comparable or even greater computational time than probabilistic inference methods based on Monte Carlo sampling. To the best of our knowledge, the preference for the direct solution method may be due to the relative simplicity of implementing and debugging direct local minimization solvers, compared to Monte Carlo sampling procedures, which are well known for being difficult to debug. 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.

7.2 Judging a heuristic fitting

Recall that our method, by solving the convex surrogate (4.3), computes a lower bound for the original problem (2.6). 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 suboptimality of any other heuristic fitting results (at least semi-quantitatively).

Let J be a heuristic objective value of (2.6) (obtained via any solution method), and let $J^\star$ be the global minimum. Suppose that by solving the convex program (4.3), we obtain a lower bound J^{lb} to (2.6), 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).

7.3 Previous and related work

Readers that 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.* [BNLS22], 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 paper offers a deeper theoretical insight, showing that the generalized logistic regression model is, in fact, a convex relaxation of the original RL model.

Acknowledgments

This work has been supported by 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.

Data availability

The source code to generate the data and reproduce our numerical results provided in §6.5 can be found at `https://github.com/nrgrp/fit_rl_mab`.

A A convex formulation for recovering RL model parameters

In this section, we discuss an option of formulating the optimization problem of recovering the RL model parameters from the optimal point $G^{(i)\star}$ of the problem (4.3), $i = 1, \dots, k$, as a convex program.

Instead of directly penalizing the difference between $f(\alpha_j^{(i)}, \beta_j^{(i)})$ and $\bar{g}_j^{(i)\star}$ using the ℓ_2-squared penalty function as in (4.4), we consider measuring the difference between these two terms in the log space, *i.e.*, for all $i = 1, \dots, k$, $j = 1, \dots, m$, we solve the following problem:

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

with optimization variables $\alpha_j^{(i)}, \beta_j^{(i)} \in \mathbf{R}$ and data $\bar{g}^{(i)\star} \in \mathbf{R}^n$ from the jth row of $G^{(i)\star}$, and the transformation $f \colon \mathbf{R} \times \mathbf{R} \to \mathbf{R}^n$ is given by

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

The objective of the problem (A.1) can be explicitly written as

$$\begin{aligned} \left\| \log f(\alpha_j^{(i)}, \beta_j^{(i)}) - \log \bar{g}_j^{(i)\star} \right\|_2^2 &= \left\| \begin{bmatrix} \log(\alpha_j^{(i)} \beta_j^{(i)}) \\ \log((1-\alpha_j^{(i)})^1 \alpha_j^{(i)} \beta_j^{(i)}) \\ \vdots \\ \log((1-\alpha_j^{(i)})^{n-1} \alpha_j^{(i)} \beta_j^{(i)}) \end{bmatrix} - \log \bar{g}_j^{(i)\star} \right\|_2^2 \\ &= \left\| \begin{bmatrix} 0 \times \log(1-\alpha_j^{(i)}) + \log(\alpha_j^{(i)} \beta_j^{(i)}) \\ 1 \times \log(1-\alpha_j^{(i)}) + \log(\alpha_j^{(i)} \beta_j^{(i)}) \\ \vdots \\ (n-1) \times \log(1-\alpha_j^{(i)}) + \log(\alpha_j^{(i)} \beta_j^{(i)}) \end{bmatrix} - \log \bar{g}_j^{(i)\star} \right\|_2^2 . \end{aligned}$$

Introducing the variable transformations

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ \vdots & \vdots \\ n-1 & 1 \end{bmatrix}, \quad v_j^{(i)} = \begin{bmatrix} \log(1-\alpha_j^{(i)}) \\ \log(\alpha_j^{(i)} \beta_j^{(i)}) \end{bmatrix}, \quad s_j^{(i)} = \log \bar{g}_j^{(i)\star},$$

we transform the problem (A.1) into an equivalent constrained least squares problem

$$\begin{array}{ll} \text{minimize} & \left\| A v_j^{(i)} - s_j^{(i)} \right\|_2^2 \\ \text{subject to} & v_{j,1}^{(i)} < 0 \end{array} \tag{A.2}$$

with variables $v_j^{(i)} \in \mathbf{R}^2$ and data $A \in \mathbf{R}^{n \times 2}$, $s_j^{(i)} \in \mathbf{R}^n$. (The notation $v_{j,1}^{(i)}$ denotes the first entry of the vector $v_j^{(i)}$.)

Formulating the problem of recovering the RL model parameters as (A.1) benefits from the property that it can be solved robustly and efficiently via the convex constrained least squares problem (A.2). However, such a formulation may suffer from severe numerical issues. First, notice that the inequality constraint $v_{j,1}^{(i)} < 0$ of (A.2) has to be strict, which is not supported by most conic solvers for convex optimization problems. If we just solve with the relaxed constraint $v_{j,1}^{(i)} \leq 0$, it is likely that we obtain some optimal point $v_{j,1}^{(i)\star}$ with the first entry being 0, in which case the corresponding optimal point $\alpha_j^{(i)\star} = 0$, where the second entry $\log(\alpha_j^{(i)\star}\beta_j^{(i)\star})$ does not exist for any $\beta_j^{(i)\star} \geq 0$. Besides, the objective of (A.1) tends to add a very large penalty to those entries of $\bar{g}_j^{(i)\star}$ that are close to zero compared to the large entries, even when they have the same residual. Such behavior is due to the characteristic of the logarithmic function $x \mapsto \log x$, whose slope approaches infinity as $x \to 0$. Recall that in (4.3), we expect that each row of the optimal point $G^{(i)\star}$ to decay (ideally) geometrically. Combining these two effects together, the solution of (A.1) with data $\bar{g}_j^{(i)}$ tends to have a very good fit to the tail of the vector $\bar{g}_j^{(i)}$ where the entries are nearly zero, while the residual at the beginning entries can be very large. If n is large, *i.e.*, there might exist many nearly zero entries in $\bar{g}_j^{(i)}$, the results can be problematic since they tend to provide a fitting that matches the noninformative small tails of $\bar{g}_j^{(i)}$ which are largely influenced by numerical roundoff error, instead of the informative entries at the beginning. For these reasons, although we demonstrate this option of formulating the problem of recovering the RL model parameters as the convex program (A.1), we will not consider this approach in practice.

B Additional tables

In tables 2 to 7, we list the detailed numerical results of all experiments in §6, corresponding to figures 1 and 2. As introduced in §6.4, the expectation term appearing in the first row of each table is over $t = 1, \dots, n$.

Table 2 Numerical results correspond to figure 1a.

		Nelder-Mead	L-BFGS-B	TNC	SLSQP	Powell	Trust-Region	COBYLA	COBYQA	–
$\mathbf{E}\, D_{\mathrm{kl}}(\pi^{\mathrm{gt}}(t), \pi^\star(t))$	CVX	–	–	–	–	–	–	–	–	0.006 (0.003–0.010)
	CVX-LOC	0.004 (0.002–0.009)	0.004 (0.002–0.009)	0.004 (0.002–0.009)	0.004 (0.002–0.009)	0.004 (0.002–0.009)	0.004 (0.002–0.009)	0.004 (0.002–0.009)	0.004 (0.002–0.009)	–
	CVX-T	–	–	–	–	–	–	–	–	0.005 (0.003–0.010)
	CVX-LOC-T	0.005 (0.002–0.010)	0.005 (0.002–0.010)	0.005 (0.002–0.010)	0.005 (0.002–0.010)	0.005 (0.002–0.010)	0.005 (0.002–0.010)	0.005 (0.002–0.010)	0.005 (0.002–0.010)	–
	D-LOC	0.006 (0.002–0.016)	0.007 (0.002–0.017)	0.007 (0.002–0.016)	0.007 (0.002–0.017)	0.007 (0.002–0.016)	0.006 (0.002–0.016)	0.006 (0.002–0.016)	0.007 (0.002–0.016)	–
	MC	–	–	–	–	–	–	–	–	0.003 (0.001–0.006)
$\lvert\alpha^{\mathrm{gt}} - \alpha^\star\rvert$	CVX-LOC	0.10 (0.04–0.21)	0.09 (0.04–0.20)	0.09 (0.04–0.20)	0.09 (0.04–0.20)	0.09 (0.04–0.20)	0.09 (0.04–0.20)	0.09 (0.04–0.20)	0.09 (0.04–0.20)	–
	CVX-LOC-T	0.10 (0.04–0.20)	0.10 (0.04–0.20)	0.10 (0.04–0.20)	0.10 (0.04–0.20)	0.10 (0.04–0.20)	0.10 (0.04–0.20)	0.10 (0.04–0.20)	0.10 (0.04–0.20)	–
	D-LOC	0.13 (0.04–0.28)	0.13 (0.04–0.27)	0.13 (0.04–0.28)	0.13 (0.04–0.28)	0.13 (0.04–0.28)	0.13 (0.04–0.26)	0.13 (0.04–0.28)	0.13 (0.04–0.28)	–
	MC	–	–	–	–	–	–	–	–	0.08 (0.03–0.15)
$\lvert\beta^{\mathrm{gt}} - \beta^\star\rvert$	CVX-LOC	0.30 (0.13–0.58)	0.31 (0.14–0.60)	0.30 (0.13–0.59)	0.31 (0.13–0.60)	0.30 (0.13–0.59)	0.31 (0.13–0.60)	0.31 (0.14–0.59)	0.31 (0.13–0.60)	–
	CVX-LOC-T	0.38 (0.16–0.82)	0.38 (0.16–0.82)	0.38 (0.16–0.83)	0.39 (0.16–0.83)	0.38 (0.15–0.82)	0.39 (0.16–0.85)	0.40 (0.16–0.81)	0.39 (0.16–0.85)	–
	D-LOC	0.35 (0.17–0.72)	0.37 (0.17–0.73)	0.37 (0.17–0.73)	0.37 (0.17–0.73)	0.38 (0.17–0.75)	0.38 (0.17–0.75)	0.41 (0.18–0.79)	0.41 (0.18–0.79)	–
	MC	–	–	–	–	–	–	–	–	0.24 (0.11–0.41)
time (ms)	CVX	–	–	–	–	–	–	–	–	53 (44–62)
	CVX-LOC	77 (67–86)	66 (56–75)	83 (72–92)	61 (52–70)	104 (85–177)	158 (144–178)	136 (98–214)	214 (197–261)	–
	CVX-T	–	–	–	–	–	–	–	–	3.8 (3.4–4.4)
	CVX-LOC-T	7.8 (7.2–8.4)	8.3 (7.4–9.2)	12 (10–13)	6.9 (6.3–7.6)	18 (12–31)	100 (89–129)	20 (14–33)	153 (138–196)	–
	D-LOC	121 (112–131)	48 (30–54)	257 (225–296)	40 (30–50)	125 (112–157)	393 (333–465)	94 (66–232)	226 (193–331)	–
	MC	–	–	–	–	–	–	–	–	1366 (1320–1419)

Table 3 Numerical results correspond to figure 1b.

		Nelder-Mead	L-BFGS-B	TNC	SLSQP	Powell	Trust-Region	COBYLA	COBYQA	–
$\mathbf{E}\, D_{\mathrm{kl}}(\pi^{\mathrm{gt}}(t), \pi^\star(t))$	CVX	–	–	–	–	–	–	–	–	0.014 (0.008–0.024)
	CVX-LOC	0.012 (0.006–0.021)	0.012 (0.006–0.020)	0.012 (0.006–0.020)	0.012 (0.006–0.020)	0.012 (0.006–0.020)	0.012 (0.006–0.020)	0.011 (0.006–0.020)	0.012 (0.006–0.020)	–
	CVX-T	–	–	–	–	–	–	–	–	0.011 (0.007–0.023)
	CVX-LOC-T	0.012 (0.006–0.025)	0.012 (0.006–0.025)	0.012 (0.006–0.025)	0.012 (0.006–0.025)	0.012 (0.006–0.025)	0.012 (0.006–0.025)	0.013 (0.006–0.026)	0.012 (0.006–0.025)	–
	D-LOC	0.014 (0.007–0.024)	0.014 (0.007–0.025)	0.014 (0.007–0.024)	0.014 (0.007–0.025)	0.014 (0.007–0.024)	0.014 (0.007–0.024)	0.014 (0.007–0.024)	0.014 (0.007–0.024)	–
	MC	–	–	–	–	–	–	–	–	0.006 (0.003–0.011)
$\Vert\alpha^{\mathrm{gt}} - \alpha^\star\Vert_2$	CVX-LOC	0.30 (0.17–0.48)	0.29 (0.16–0.47)	0.29 (0.16–0.46)	0.29 (0.16–0.47)	0.29 (0.16–0.47)	0.29 (0.16–0.47)	0.29 (0.16–0.47)	0.29 (0.16–0.47)	–
	CVX-LOC-T	0.28 (0.17–0.43)	0.28 (0.16–0.43)	0.28 (0.16–0.43)	0.28 (0.16–0.43)	0.28 (0.16–0.43)	0.28 (0.17–0.43)	0.29 (0.17–0.45)	0.28 (0.17–0.43)	–
	D-LOC	0.37 (0.22–0.61)	0.38 (0.22–0.62)	0.37 (0.21–0.61)	0.38 (0.22–0.62)	0.39 (0.22–0.64)	0.37 (0.21–0.61)	0.38 (0.22–0.63)	0.38 (0.22–0.63)	–
	MC	–	–	–	–	–	–	–	–	0.22 (0.14–0.34)
$\Vert\beta^{\mathrm{gt}} - \beta^\star\Vert_2$	CVX-LOC	1.05 (0.60–1.79)	1.06 (0.60–1.80)	1.06 (0.60–1.80)	1.06 (0.60–1.80)	1.06 (0.60–1.81)	1.06 (0.62–1.81)	1.06 (0.60–1.81)	1.07 (0.62–1.84)	–
	CVX-LOC-T	1.12 (0.59–2.24)	1.02 (0.55–1.87)	1.07 (0.58–1.89)	1.04 (0.56–1.88)	1.02 (0.55–1.86)	1.10 (0.58–1.95)	1.14 (0.64–2.09)	1.12 (0.59–2.23)	–
	D-LOC	0.98 (0.58–1.74)	1.05 (0.62–1.81)	1.04 (0.59–1.75)	1.05 (0.61–1.85)	1.07 (0.60–1.91)	1.07 (0.61–2.02)	1.08 (0.62–1.83)	1.11 (0.63–2.14)	–
	MC	–	–	–	–	–	–	–	–	0.64 (0.40–1.00)
time (ms)	CVX	–	–	–	–	–	–	–	–	80 (64–97)
	CVX-LOC	108 (90–125)	157 (138–176)	117 (101–135)	93 (77–110)	184 (140–248)	207 (180–274)	218 (161–301)	283 (247–350)	–
	CVX-T	–	–	–	–	–	–	–	–	4.5 (3.9–5.3)
	CVX-LOC-T	10 (9.4–11)	74 (71–77)	15 (14–17)	10 (9–11)	21 (17–33)	127 (109–239)	30 (20–39)	184 (154–252)	–
	D-LOC	481 (410–559)	142 (118–179)	927 (855–1154)	120 (98–163)	374 (299–462)	1038 (875–1342)	521 (252–995)	1040 (575–3092)	–
	MC	–	–	–	–	–	–	–	–	1811 (1741–1901)

Table 4 Numerical results correspond to figure 1c.

		Nelder-Mead	L-BFGS-B	TNC	SLSQP	Powell	Trust-Region	COBYLA	COBYQA	–
$\mathbf{E}\, D_{\text{kl}}(\pi^{\text{gt}}(t), \pi^\star(t))$	CVX	–	–	–	–	–	–	–	–	0.022 (0.012–0.071)
	CVX-LOC	0.019 (0.010–0.036)	0.019 (0.010–0.036)	0.019 (0.010–0.036)	0.019 (0.010–0.036)	0.019 (0.010–0.036)	0.019 (0.010–0.036)	0.020 (0.010–0.036)	0.019 (0.010–0.036)	–
	CVX-T	–	–	–	–	–	–	–	–	0.018 (0.010–0.048)
	CVX-LOC-T	0.018 (0.010–0.034)	0.018 (0.010–0.034)	0.018 (0.010–0.034)	0.018 (0.010–0.034)	0.018 (0.010–0.034)	0.018 (0.010–0.034)	0.018 (0.010–0.034)	0.018 (0.010–0.034)	–
	D-LOC	0.017 (0.011–0.025)	0.017 (0.011–0.026)	0.017 (0.011–0.026)	0.017 (0.010–0.025)	0.017 (0.011–0.025)	0.017 (0.010–0.025)	0.017 (0.010–0.025)	0.017 (0.010–0.025)	–
	MC	–	–	–	–	–	–	–	–	0.009 (0.005–0.014)
$\lVert\alpha^{\text{gt}} - \alpha^\star\rVert_2$	CVX-LOC	0.83 (0.61–1.02)	0.73 (0.50–0.91)	0.74 (0.53–0.94)	0.74 (0.50–0.93)	0.78 (0.56–0.98)	0.78 (0.56–0.98)	0.77 (0.55–0.97)	0.78 (0.56–0.98)	–
	CVX-LOC-T	0.70 (0.50–0.91)	0.73 (0.52–0.94)	0.71 (0.50–0.92)	0.73 (0.52–0.94)	0.73 (0.51–0.93)	0.73 (0.52–0.93)	0.73 (0.52–0.93)	0.73 (0.52–0.93)	–
	D-LOC	0.81 (0.60–1.01)	0.76 (0.56–0.97)	0.73 (0.53–0.94)	0.77 (0.57–1.00)	0.78 (0.57–0.99)	0.75 (0.54–0.96)	0.76 (0.56–0.96)	0.81 (0.59–1.01)	–
	MC	–	–	–	–	–	–	–	–	0.477 (0.377–0.581)
$\lVert\beta^{\text{gt}} - \beta^\star\rVert_2$	CVX-LOC	2.00 (1.41–2.80)	2.00 (1.40–2.84)	2.01 (1.40–2.83)	2.00 (1.41–2.80)	2.01 (1.42–2.84)	1.98 (1.35–2.88)	1.99 (1.41–2.77)	1.93 (1.32–2.74)	–
	CVX-LOC-T	2.02 (1.32–3.00)	2.11 (1.43–2.99)	2.13 (1.44–3.03)	2.06 (1.41–2.99)	2.01 (1.32–2.96)	1.98 (1.41–2.91)	1.99 (1.36–2.87)	2.13 (1.45–3.05)	–
	D-LOC	2.02 (1.44–2.73)	2.01 (1.40–2.75)	1.84 (1.31–2.58)	2.05 (1.45–2.76)	2.02 (1.37–2.79)	2.05 (1.42–2.91)	1.87 (1.31–2.52)	2.14 (1.51–3.00)	–
	MC	–	–	–	–	–	–	–	–	1.17 (0.84–1.59)
time (ms)	CVX	–	–	–	–	–	–	–	–	186 (161–214)
	CVX-LOC	215 (189–242)	301 (266–336)	227 (200–255)	205 (178–233)	350 (283–431)	325 (292–370)	357 (273–435)	415 (355–488)	–
	CVX-T	–	–	–	–	–	–	–	–	5.9 (4.9–7.5)
	CVX-LOC-T	12 (11–13)	87 (82–105)	17 (16–19)	12 (11–13)	33 (25–42)	154 (120–293)	33 (23–42)	194 (154–292)	–
	D-LOC	4532 (4060–4928)	1040 (765–1409)	3025 (2939–3043)	655 (515–842)	2255 (1966–2611)	5848 (4402–10247)	2741 (2080–3274)	8124 (3083–16620)	–
	MC	–	–	–	–	–	–	–	–	2967 (2829–3177)

Table 5 Numerical results correspond to figure 2a.

		Nelder-Mead	L-BFGS-B	TNC	SLSQP	Powell	Trust-Region	COBYLA	COBYQA	–
$\mathbf{E}\, D_{\mathrm{kl}}(\pi^{\mathrm{gt}}(t), \pi^{\star}(t))$	CVX	–	–	–	–	–	–	–	–	0.009 (0.005–0.026)
	CVX-LOC	0.005 (0.002–0.014)	0.005 (0.002–0.014)	0.005 (0.002–0.014)	0.005 (0.002–0.014)	0.005 (0.002–0.014)	0.005 (0.002–0.014)	0.005 (0.002–0.014)	0.005 (0.002–0.014)	–
	CVX-T	–	–	–	–	–	–	–	–	0.012 (0.005–0.037)
	CVX-LOC-T	0.007 (0.003–0.023)	0.007 (0.003–0.023)	0.007 (0.003–0.023)	0.007 (0.003–0.023)	0.007 (0.003–0.023)	0.007 (0.003–0.023)	0.007 (0.003–0.023)	0.007 (0.003–0.023)	–
	D-LOC	0.009 (0.003–0.021)	0.009 (0.003–0.021)	0.009 (0.003–0.021)	0.009 (0.003–0.021)	0.009 (0.003–0.021)	0.009 (0.003–0.021)	0.009 (0.003–0.021)	0.009 (0.003–0.021)	–
	MC	–	–	–	–	–	–	–	–	0.002 (0.001–0.005)
$\lvert\alpha^{\mathrm{gt}} - \alpha^{\star}\rvert$	CVX-LOC	0.07 (0.02–0.15)	0.07 (0.02–0.15)	0.07 (0.02–0.15)	0.07 (0.02–0.15)	0.07 (0.02–0.15)	0.07 (0.02–0.15)	0.07 (0.02–0.15)	0.07 (0.02–0.15)	–
	CVX-LOC-T	0.07 (0.03–0.15)	0.07 (0.03–0.15)	0.07 (0.03–0.15)	0.07 (0.03–0.15)	0.07 (0.03–0.15)	0.07 (0.03–0.15)	0.07 (0.03–0.15)	0.07 (0.03–0.15)	–
	D-LOC	0.08 (0.03–0.20)	0.08 (0.03–0.20)	0.08 (0.03–0.20)	0.08 (0.03–0.20)	0.08 (0.03–0.20)	0.08 (0.03–0.20)	0.08 (0.03–0.20)	0.08 (0.03–0.20)	–
	MC	–	–	–	–	–	–	–	–	0.05 (0.02–0.09)
$\lvert\beta^{\mathrm{gt}} - \beta^{\star}\rvert$	CVX-LOC	0.72 (0.30–1.38)	0.72 (0.30–1.34)	0.71 (0.30–1.34)	0.71 (0.30–1.34)	0.71 (0.30–1.34)	0.71 (0.30–1.34)	0.69 (0.29–1.32)	0.71 (0.30–1.34)	–
	CVX-LOC-T	0.90 (0.37–1.81)	0.90 (0.37–1.81)	0.90 (0.37–1.81)	0.90 (0.37–1.80)	0.90 (0.37–1.80)	0.90 (0.38–1.81)	0.85 (0.37–1.72)	0.90 (0.37–1.81)	–
	D-LOC	0.96 (0.42–1.67)	0.96 (0.42–1.67)	0.96 (0.42–1.67)	0.96 (0.42–1.67)	0.96 (0.42–1.67)	0.96 (0.42–1.67)	0.92 (0.42–1.65)	0.96 (0.42–1.67)	–
	MC	–	–	–	–	–	–	–	–	0.47 (0.20–0.87)
time (ms)	CVX	–	–	–	–	–	–	–	–	74 (64–91)
	CVX-LOC	94 (84–110)	90 (80–105)	100 (90–115)	81 (71–95)	116 (95–147)	173 (157–192)	303 (140–351)	231 (207–262)	–
	CVX-T	–	–	–	–	–	–	–	–	23 (20–26)
	CVX-LOC-T	26 (24–29)	28 (25–30)	29 (27–32)	26 (24–29)	33 (29–38)	113 (101–123)	60 (37–68)	168 (134–185)	–
	D-LOC	111 (86–124)	61 (53–73)	248 (181–289)	45 (39–54)	140 (111–235)	438 (321–509)	356 (136–921)	240 (209–288)	–
	MC	–	–	–	–	–	–	–	–	2259 (2187–2334)

Table 6 Numerical results correspond to figure 2b.

		Nelder-Mead	L-BFGS-B	TNC	SLSQP	Powell	Trust-Region	COBYLA	COBYQA	–
$\mathbf{E}\, D_{\mathrm{kl}}(\pi^{\mathrm{gt}}(t), \pi^{\star}(t))$	CVX	–	–	–	–	–	–	–	–	0.09 (0.04–0.18)
	CVX-LOC	0.034 (0.017–0.059)	0.034 (0.017–0.059)	0.034 (0.017–0.059)	0.034 (0.017–0.059)	0.034 (0.017–0.059)	0.034 (0.017–0.059)	0.034 (0.017–0.059)	0.034 (0.017–0.059)	–
	CVX-T	–	–	–	–	–	–	–	–	0.08 (0.04–0.17)
	CVX-LOC-T	0.037 (0.017–0.068)	0.036 (0.017–0.068)	0.036 (0.017–0.068)	0.036 (0.017–0.068)	0.037 (0.017–0.068)	0.036 (0.017–0.068)	0.037 (0.017–0.068)	0.036 (0.017–0.068)	–
	D-LOC	0.033 (0.016–0.054)	0.034 (0.016–0.054)	0.033 (0.016–0.055)	0.034 (0.016–0.054)	0.033 (0.016–0.053)	0.034 (0.016–0.054)	0.034 (0.016–0.056)	0.034 (0.016–0.056)	–
	MC	–	–	–	–	–	–	–	–	0.010 (0.006–0.018)
$\Vert\alpha^{\mathrm{gt}} - \alpha^{\star}\Vert_2$	CVX-LOC	1.16 (0.91–1.45)	1.16 (0.91–1.45)	1.16 (0.91–1.45)	1.16 (0.91–1.45)	1.16 (0.91–1.45)	1.16 (0.91–1.45)	1.16 (0.91–1.45)	1.16 (0.91–1.45)	–
	CVX-LOC-T	1.08 (0.85–1.37)	1.08 (0.85–1.37)	1.08 (0.85–1.37)	1.08 (0.85–1.37)	1.08 (0.85–1.38)	1.08 (0.85–1.37)	1.08 (0.85–1.37)	1.08 (0.85–1.37)	–
	D-LOC	1.34 (1.13–1.53)	1.19 (1.02–1.37)	1.21 (1.02–1.37)	1.18 (1.01–1.37)	1.46 (1.24–1.67)	1.02 (0.86–1.18)	1.20 (1.03–1.39)	1.31 (1.11–1.51)	–
	MC	–	–	–	–	–	–	–	–	0.71 (0.58–0.83)
$\Vert\beta^{\mathrm{gt}} - \beta^{\star}\Vert_2$	CVX-LOC	8.17 (7.18–9.00)	8.18 (7.18–9.02)	8.15 (7.13–8.98)	8.17 (7.18–9.02)	8.17 (7.18–9.02)	8.19 (7.23–9.01)	8.00 (7.08–8.92)	8.01 (7.02–8.93)	–
	CVX-LOC-T	7.92 (7.00–8.88)	8.08 (7.14–8.98)	7.96 (7.00–8.85)	8.08 (7.13–8.96)	8.02 (7.10–8.94)	8.18 (7.24–8.99)	8.01 (7.01–8.87)	8.25 (7.27–9.08)	–
	D-LOC	7.66 (6.68–8.52)	6.65 (5.69–7.55)	6.42 (5.56–7.38)	6.64 (5.78–7.50)	8.22 (7.18–9.19)	5.94 (5.08–6.84)	6.17 (5.40–7.04)	6.77 (5.86–7.68)	–
	MC	–	–	–	–	–	–	–	–	4.21 (3.69–4.67)
time (ms)	CVX	–	–	–	–	–	–	–	–	171 (120–408)
	CVX-LOC	201 (142–450)	343 (224–634)	213 (152–459)	190 (134–443)	262 (189–514)	1209 (443–1412)	442 (348–716)	444 (354–738)	–
	CVX-T	–	–	–	–	–	–	–	–	29 (26–32)
	CVX-LOC-T	35 (32–39)	237 (147–329)	42 (38–45)	34 (32–38)	46 (42–53)	205 (181–656)	69 (57–76)	222 (187–257)	–
	D-LOC	7595 (7194–7646)	1425 (762–2284)	7626 (5206–7871)	814 (493–1104)	2394 (2101–2794)	8810 (5248–15874)	1927 (1859–1956)	18286 (5331–41791)	–
	MC	–	–	–	–	–	–	–	–	3828 (3573–4111)

Table 7 Numerical results correspond to figure 2c.

		Nelder-Mead	L-BFGS-B	TNC	SLSQP	Powell	Trust-Region	COBYLA	COBYQA*	–
$\mathbf{E}\, D_{\text{kl}}(\pi^{\text{gt}}(t), \pi^\star(t))$	CVX	–	–	–	–	–	–	–	–	0.078 (0.032–0.174)
	CVX-LOC	0.030 (0.010–0.075)	0.031 (0.011–0.075)	0.031 (0.011–0.075)	0.031 (0.011–0.075)	0.031 (0.011–0.075)	0.031 (0.011–0.075)	0.031 (0.010–0.075)	0.031 (0.011–0.075)	–
	CVX-T	–	–	–	–	–	–	–	–	0.075 (0.028–0.167)
	CVX-LOC-T	0.026 (0.010–0.056)	0.026 (0.010–0.056)	0.026 (0.010–0.056)	0.026 (0.010–0.056)	0.026 (0.010–0.056)	0.026 (0.010–0.056)	0.026 (0.010–0.057)	0.026 (0.010–0.056)	–
	D-LOC	0.023 (0.011–0.039)	0.024 (0.011–0.041)	0.023 (0.011–0.039)	0.024 (0.011–0.041)	0.023 (0.011–0.039)	0.024 (0.011–0.040)	0.022 (0.010–0.037)	–	–
	MC	–	–	–	–	–	–	–	–	0.009 (0.003–0.017)
$\|\alpha^{\text{gt}} - \alpha^\star\|_2$	CVX-LOC	2.31 (2.05–2.52)	2.26 (2.02–2.48)	2.24 (2.00–2.47)	2.26 (2.02–2.48)	2.27 (2.03–2.48)	2.26 (2.03–2.48)	2.26 (2.02–2.48)	2.26 (2.02–2.48)	–
	CVX-LOC-T	2.19 (1.90–2.44)	2.21 (1.94–2.46)	2.15 (1.86–2.42)	2.21 (1.94–2.46)	2.21 (1.94–2.46)	2.21 (1.94–2.46)	2.18 (1.92–2.45)	2.21 (1.94–2.46)	–
	D-LOC	2.17 (1.93–2.37)	1.89 (1.71–2.06)	1.85 (1.68–2.04)	1.88 (1.70–2.06)	2.34 (2.16–2.54)	1.58 (1.40–1.77)	1.90 (1.70–2.07)	–	–
	MC	–	–	–	–	–	–	–	–	1.22 (1.12–1.32)
$\|\beta^{\text{gt}} - \beta^\star\|_2$	CVX-LOC	12.2 (11.2–13.1)	12.1 (11.1–13.0)	11.9 (11.0–12.9)	12.0 (11.0–12.9)	12.2 (11.3–13.1)	11.9 (11.0–12.9)	12.1 (11.2–13.0)	11.8 (10.8–12.8)	–
	CVX-LOC-T	12.2 (11.1–13.2)	11.7 (10.7–12.7)	12.0 (11.0–13.0)	11.9 (11.0–13.0)	12.1 (11.1–13.2)	11.7 (10.8–12.8)	12.0 (11.0–13.0)	11.8 (10.8–12.8)	–
	D-LOC	10.9 (9.8–12.0)	9.2 (8.3–10.2)	9.2 (8.4–10.1)	9.3 (8.4–10.1)	12.0 (11.0–13.0)	7.8 (7.1–8.7)	9.0 (8.2–9.9)	–	–
	MC	–	–	–	–	–	–	–	–	6.25 (5.78–6.72)
time (ms)	CVX	–	–	–	–	–	–	–	–	283 (241–374)
	CVX-LOC	312 (271–410)	456 (353–646)	325 (284–421)	301 (259–397)	643 (542–747)	572 (473–755)	469 (341–614)	549 (470–728)	–
	CVX-T	–	–	–	–	–	–	–	–	33 (29–38)
	CVX-LOC-T	41 (37–46)	198 (142–348)	51 (47–56)	41 (37–46)	62 (54–71)	260 (219–610)	69 (49–78)	258 (229–313)	–
	D-LOC	28473 (28010–28687)	5942 (2795–11515)	49086 (20681–56594)	3669 (2488–5101)	12386 (8115–15662)	53444 (17392–108439)	3854 (3720–3885)	–	–
	MC	–	–	–	–	–	–	–	–	13044 (12232–13640)

[*] The COBYQA solver could not return a solution in acceptable time when applied in the D-LOC method, and hence the corresponding numerical results are omitted.

References

[APAC+23] O. Abril-Pla, V. Andreani, C. Carroll, L. Dong, J. J. Fonnesbeck, M. Kochurov, R. Kumar, J. Lao, C. C. Luhmann, O. A. Martin, M. Osthege, R. Vieira, T. Wiecki, and R. Zinkov. PyMC: A modern, and comprehensive probabilistic programming framework in Python. *PeerJ Computer Science*, 9:e1516, 2023.

[AVDB18] A. Agrawal, R. Verschueren, S. Diamond, and S. Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60, 2018.

[BCL99] M. Am Branch, T. F. Coleman, and Y. Li. A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems. *SIAM Journal on Scientific Computing*, 21(1):1–23, 1999.

[BGL+19] B. A. Bari, C. D. Grossman, E. E. Lubin, A. E. Rajagopalan, J. I. Cressy, and J. Y. Cohen. Stable representations of decision variables for flexible behavior. *Neuron*, 103(5):922–933, 2019.

[BNLS22] C. C. Beron, S. Q. Neufeld, S. W. Linderman, and B. L. Sabatini. Mice exhibit stochastic and efficient action switching during probabilistic decision making. *Proceedings of the National Academy of Sciences*, 119(15):e2113961119, 2022.

[BV04] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[CGT00] A. R. Conn, N. I. M. Gould, and P. L. Toint. *Trust Region Methods*. SIAM, 2000.

[CMA19] V. D. Costa, A. R. Mitz, and B. B. Averbeck. Subcortical substrates of explore-exploit decisions in primates. *Neuron*, 103(3):533–545, 2019.

[DB16] S. Diamond and S. Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[DLCSS+23] B. De La Crompe, M. Schneck, F. Steenbergen, A. Schneider, and I. Diester. FreiBox: A versatile open-source behavioral setup for investigating the neuronal correlates of behavioral flexibility via 1-photon imaging in freely moving mice. *Eneuro*, 10(4), 2023.

[DLK18] C. H. Donahue, M. Liu, and A. C. Kreitzer. Distinct value encoding in striatal direct and indirect pathways during adaptive learning. *bioRxiv*, page 277855, 2018.

[DOD+06] N. D. Daw, J. P. O'doherty, P. Dayan, B. Seymour, and R. J. Dolan. Cortical substrates for exploratory decisions in humans. *Nature*, 441(7095):876–879, 2006.

[EAM18] R. B. Ebitz, E. Albarran, and T. Moore. Exploration disrupts choice-predictive signals and alters dynamics in prefrontal cortex. *Neuron*, 97(2):450–461, 2018.

[GH12] F. Gao and L. Han. Implementing the Nelder-Mead simplex algorithm with adaptive parameters. *Computational Optimization and Applications*, 51(1):259–277, 2012.

[HDB$^+$19] R. Hattori, B. Danskin, Z. Babic, N. Mlynaryk, and T. Komiyama. Area-specificity and plasticity of history-dependent value coding during learning. *Cell*, 177(7):1858–1872, 2019.

[HHJ$^+$23] R. Hattori, N. G. Hedrick, A. Jain, S. Chen, H. You, M. Hattori, J.-H. Choi, B. K. Lim, R. Yasuda, and T. Komiyama. Meta-reinforcement learning via orbitofrontal cortex. *Nature Neuroscience*, 26(12):2182–2191, 2023.

[HTAW22] K. Hamaguchi, H. Takahashi-Aoki, and D. Watanabe. Prospective and retrospective values integrated in frontal cortex drive predictive choice. *Proceedings of the National Academy of Sciences*, 119(48):e2206067119, 2022.

[ID09] M. Ito and K. Doya. Validation of decision-making models and analysis of decision variables in the rat basal ganglia. *Journal of Neuroscience*, 29(31):9861–9874, 2009.

[KPZ$^+$25] K. Kleespies, P. C. Paulus, H. Zhu, F. Pargent, M. Jakob, J. Werle, M. Czisch, J. Boedecker, S. Gais, and M. Schonauer. Sleep resolves competition between explicit and implicit memory systems. *bioRxiv*, pages 2025–02, 2025.

[Kra88] D. Kraft. A software package for sequential quadratic programming. Technical Report DFVLR-FB 88-28, DLR German Aerospace Center-Institute for Flight Mechanics, 1988.

[MBB18] K. J. Miller, M. M. Botvinick, and C. D. Brody. From predictive models to cognitive models: Separable behavioral processes underlying reward learning in the rat. *bioRxiv*, 2018.

[Nas84] S. G. Nash. Newton-type minimization via the Lanczos method. *SIAM Journal on Numerical Analysis*, 21(4):770–788, 1984.

[NM65] J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965.

[NN94] Y. Nesterov and A. Nemirovskii. *Interior-point Polynomial Algorithms in Convex Programming*. SIAM, 1994.

[NW99] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 1999.

[PCT$^+$16] N. F. Parker, C. M. Cameron, J. P. Taliaferro, J. Lee, J. Y. Choi, T. J. Davidson, N. D. Daw, and I. B. Witten. Reward and choice encoding in terminals of midbrain dopamine neurons depends on striatal target. *Nature Neuroscience*, 19(6):845–854, 2016.

[Pow64] M. J. D. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, 7(2):155–162, 1964.

[Pow94] M. J. D. Powell. A direct search optimization method that models the objective and constraint functions by linear interpolation. In S. Gomez and J.-P. Hennart, editors, *Advances in Optimization and Numerical Analysis*, pages 51–67. Springer, 1994.

[Pow02] M. J. D. Powell. UOBYQA: Unconstrained optimization by quadratic approximation. *Mathematical Programming*, 92(3):555–582, 2002.

[Rag22] T. M. Ragonneau. *Model-Based Derivative-Free Optimization Methods and Software*. PhD thesis, Department of Applied Mathematics, The Hong Kong Polytechnic University, Hong Kong, China, 2022.

[Roc70] R. T. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

[RZ24] T. M. Ragonneau and Z. Zhang. COBYQA Version 1.1.2, 2024.

[SBD$^+$16] B. Seymour, M. Barbe, P. Dayan, T. Shiner, R. Dolan, and G. R. Fink. Deep brain stimulation of the subthalamic nucleus modulates sensitivity to decision outcome value in Parkinson's disease. *Scientific Reports*, 6(1):32509, 2016.

[SUDK05] K. Samejima, Y. Ueda, K. Doya, and M. Kimura. Representation of action-specific reward values in the striatum. *Science*, 310(5752):1337–1340, 2005.

[TLB$^+$12] L.-H. Tai, A. M. Lee, N. Benavidez, A. Bonci, and L. Wilbrecht. Transient stimulation of distinct subpopulations of striatal neurons mimics changes in action value. *Nature Neuroscience*, 15(9):1281–1289, 2012.

[VGO$^+$20] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 17:261–272, 2020.

[VLS$^+$20] P. Vertechi, E. Lottem, D. Sarra, B. Godinho, I. Treves, T. Quendera, M. N. O. Lohuis, and Z. F. Mainen. Inference-based decisions in a hidden state foraging task: Differential contributions of prefrontal cortical areas. *Neuron*, 106(1):166–176, 2020.

[ZBLN97] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software*, 23(4):550–560, 1997.

[ZCK$^+$24] H. Zhu, B. De La Crompe, G. Kalweit, A. Schneider, M. Kalweit, I. Diester, and J. Boedecker. Multi-intention inverse Q-learning for interpretable behavior representation. *Transactions on Machine Learning Research*, 2024.