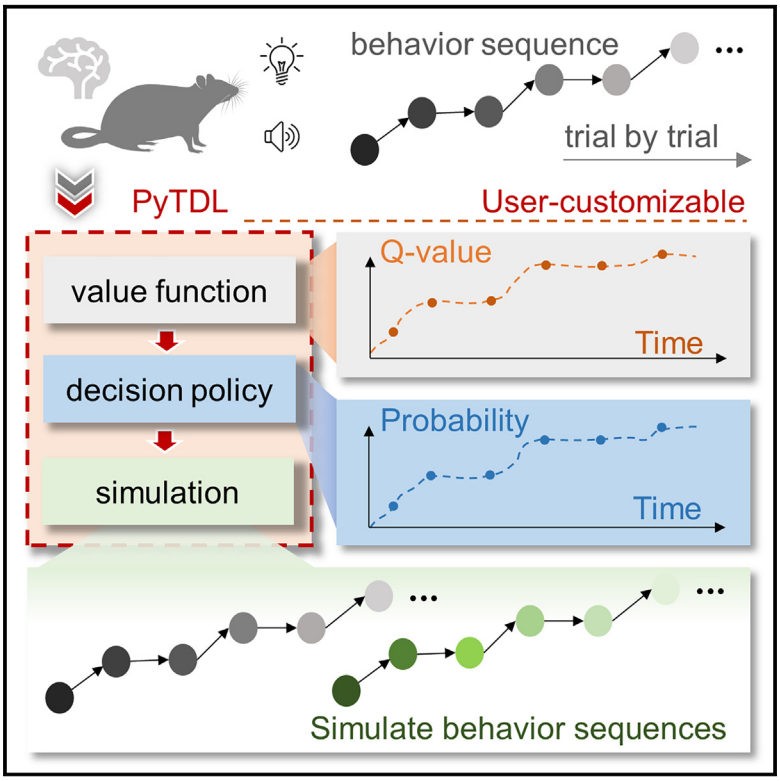
iScience

Article

"PyTDL": A versatile temporal difference learning algorithm to simulate behavior process of decision making and cognitive learning

Graphical abstract



Authors

Qiyun Wu, Xiaodan Yang, Kaishu Wang, Min Zhu, Jiejunyi Liang, Yunyun Han

Correspondence

mzhu@tjh.tjmu.edu.cn (M.Z.), jjy_liang@hust.edu.cn (J.L.), yhan@hust.edu.cn (Y.H.)

In brief

Health sciences; Natural sciences; Applied sciences

Highlights

- PyTDL is a TD learning frame to accommodate user-defined value and decision functions
- Decision switch processes can be captured by PyTDL with dynamic learning rate
- Exploring reward probability can be simulated with PyTDL with dynamic exploration

Wu et al., 2025, iScience *28*, 111600
January 17, 2025
https://doi.org/10.1016/j.isci.2024.111600

CellPress

iScience

CellPress
OPEN ACCESS

Article

"PyTDL": A versatile temporal difference learning algorithm to simulate behavior process of decision making and cognitive learning

Qiyun Wu,[1,5] Xiaodan Yang,[2,5] Kaishu Wang,[3] Min Zhu,[4,*] Jiejunyi Liang,[1,*] and Yunyun Han[2,6,*]

[1]State Key Laboratory of Intelligent Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan 430074, China

[2]Department of Neurobiology, School of Basic Medicine, Tongji Medical College, Huazhong University of Science and Technology, Wuhan 430030, China

[3]School of Civil and Hydraulic Engineering, Huazhong University of Science and Technology, Wuhan 430074, China

[4]Department of Thoracic Surgery, Tongji Hospital, Tongji Medical College, Huazhong University of Science and Technology, Wuhan 430030, China

[5]These authors contribute equally

[6]Lead Contact

*Correspondence: mzhu@tjh.tjmu.edu.cn (M.Z.), jjy_liang@hust.edu.cn (J.L.), yhan@hust.edu.cn (Y.H.)

https://doi.org/10.1016/j.isci.2024.111600

SUMMARY

Humans and animals excel at learning complex tasks through reward-based feedback, dynamically adjusting value expectations and choices based on past experiences to optimize outcomes. However, understanding the hidden cognitive components driving these behaviors remains challenging. Neuroscientists use the Temporal Difference (TD) learning model to estimate cognitive elements like value representation and prediction error during learning and decision-making processes. However, traditional TD algorithms fall short in diverse and dynamic tasks due to their fixed patterns. We present PyTDL, a Python-based modular framework that enables customizable value updating functions and decision policies, effectively simulating dynamic, nonlinear cognitive processes. PyTDL's utility was demonstrated by modeling the decision-making processes of animals in two cognitive tasks under uncertain conditions. As open-source software, PyTDL offers a user-friendly GUI and APIs, empowering researchers to tailor models for specific tasks, align computational models with empirical data, and advance the understanding of brain learning and decision-making in complex environments.

INTRODUCTION

Humans and animals are exceedingly adept at learning to perform complex tasks when the only feedback provided is a reward for correct actions. The capacity for such learning hinges on the flexibility of behavior and cognition, which critically depends on the ability to adjust the value expectations and choices based on past experiences to obtain the best outcomes.[1] The process is both complex and dynamic, involving the evaluation of options, anticipation of outcomes, and optimization of choices based on prior experiences. However, these fundamental cognitive components are challenging to measure directly, as they are often obscured behind the behavioral outputs.

Neuroscientists have employed the temporal difference (TD) learning[2] model to ascertain the cognitive components that underpin behavioral choices and adaptive behaviors. The TD algorithm, a reinforcement learning (RL) framework, is based on the principle of learning from experience and updating predictions of future outcomes based on previous experiences.[3–6] The state-action value learning sequence bears resemblance to the adaptive learning process, wherein an agent learns the values (i.e., the expected reward) associated with specific actions in response to sensory inputs (the state), selects actions based on these values, and subsequently updates the values according to feedback error.[7] Therefore, the trial-by-trial incentive value and prediction error can be estimated for decision sequences. Moreover, neural signals corresponding to TD components, such as the value representation[8–12] and prediction error,[13] have been observed in various brain regions, including dopaminergic neurons in the ventral tegmental area and prefrontal cortical areas.[14–19]

While a generic TD algorithm may provide a foundational framework for modeling the decision-making process of cognitive tasks, the classic TD model is inadequate for addressing the complexities of more dynamic cognitive tasks, which often involve nonlinear and context-dependent learning. The inflexibility of traditional TD models primarily lies in their reliance on fixed learning rates, scalar prediction errors, and a static exploration-exploitation trade-off decision policy, which limits their adaptability to dynamic environments. Consequently, to

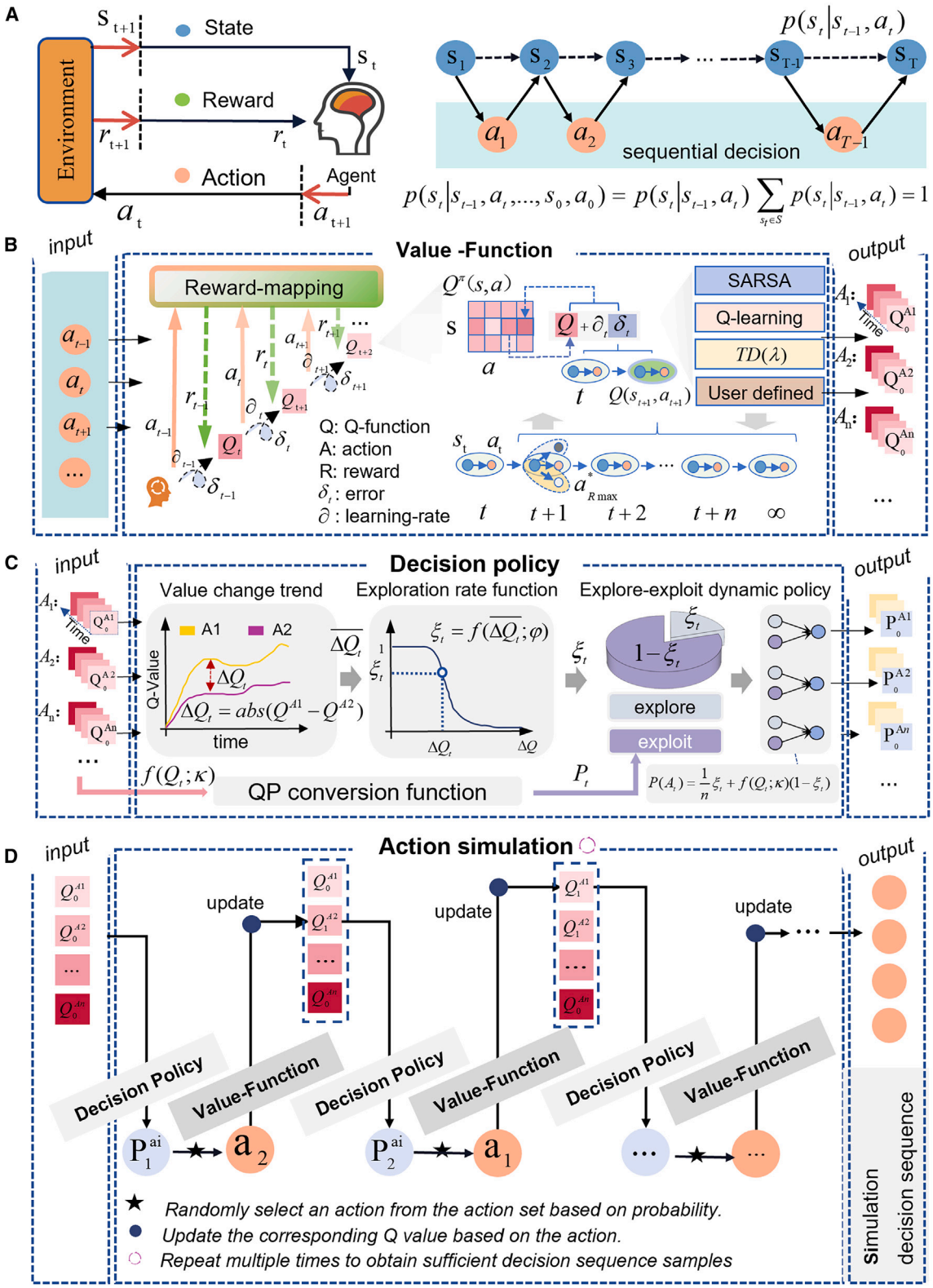


Figure 1. Overview of key functional modules of PyTDL

(A) Reinforcement Learning and Markov Decision Process. It Illustrates the foundational concepts of reinforcement learning integrated with Markov Decision Process for decision-making tasks.

(legend continued on next page)

effectively investigate and elucidate the cognitive underlying decision making in more complex and varied settings, it is essential to enhance the versatility of TD models.

Here, we have developed a Python-based open-source software toolkit, PyTDL, designed for applying the TD learning model in cognitive contexts. PyTDL offers a flexible framework for the TD algorithm, incorporating adaptive features such as custom-defined flexible non-linear learning, a value updating function, and a dynamic adjusting policy for exploration-exploitation. This toolkit is tailored to adapt the TD learning model for specific cognitive tasks, providing researchers with a valuable tool for constructing computational models that align closely with empirical data. Such alignment facilitates the validation and refinement of theories regarding learning and decision-making processes.

RESULTS

PyTDL provides key functional modules to simulate the process of decision making and associate learning

RL is a process of optimizing decision strategies through dynamic interactions between agents and their environment. This paradigm involves agents acting on the environment, receiving feedback, and continually updating their strategies to maximize long-term rewards (Figure 1A). This approach parallels the adaptive behavior observed in biological entities exploring environments. In such biological contexts, organisms adjust their behavior strategies (decision policies) based on current information and historical experiences (value functions) to respond to positive (rewards) or negative (punishments) outcomes, thereby making optimal decisions in varying contexts. PyTDL integrates the principles of biological cognitive decision-making processes with the TD algorithm, offering a suite of adaptable modules that capture the trial-by-trial evolution of decision-making processes over time. It includes four key modules: value function, decision policy, action simulation and evaluation.

The value function module of PyTDL conceptualizes biological decision sequences as Markov Decision Processes (MDP), where state transitions adhere to the Markov property: they depend solely on the current state and the action taken, without reliance on prior states (Figure 1A). In biological cognitive decision-making processes, comprehensive prior knowledge of environmental dynamics, such as state transition probabilities, is often unavailable. PyTDL addresses this limitation by employing the TD algorithm in conjunction with dynamic programming and Monte Carlo methods. The estimation is progressively refined based on partial prior experience. Updates to the value function are driven by two distinct types of information: post-action rewards/punishments and adjustments aimed at minimizing reward prediction errors (Figure 1B). PyTDL's value function module supports various estimation algorithms (e.g., Q-Learning, SARSA, TD(λ)) as well as diverse custom-defined value algorithms to model more dynamic environments and task regimes. This flexible value update mechanism enhances adaptability, enabling more effective and realistic modeling of the learning process from each encounter.

The decision policy module of PyTDL transforms estimated action values into probabilities for choosing them. Decision prioritization is based on historical experiences through value-based action selection. Notably, in biological decision-making scenarios, uncertainty often leads to a preference for novel options over immediate rewards.[20] Understanding how uncertainty and novelty jointly regulate the exploration-exploitation balance remains a challenge.[21,22] PyTDL addresses this by dynamically balancing exploration and exploitation through a combination of random and value-based approaches (Figure 1C). The module includes several standard mapping functions that link value and exploration rate, such as complementary Sigmoid, polynomial decay, and exponential decay functions, and also allows for the design of custom transformation functions tailored to specific experimental paradigms or decision tasks. This flexibility facilitates exploration into how behavioral valuation intricately influences the evolution of decision making over time.

The action simulation module of PyTDL operates as a closed-loop framework integrating value function and decision policy modules to generate simulated decision sequences using the Markov Chain Monte Carlo (MCMC) method. In biological decision making, decision transitions follow a pattern that mirrors the Markov decision process, creating a closed-loop system that encompasses the entire decision-making process: value assessment, evaluation of decision transition probabilities, decision execution, and subsequent reward-guided value adjustment and decision evaluation (Figure 1D). This iterative process produces sequences of decisions and outcomes, enabling comparison between the distribution of decision probabilities at a given state and the empirical action sequences. The simulation module is a powerful tool for researchers to explore how external information is transformed into internal representations for decision making and to efficiently simulate decision-making processes based on these insights. It is particularly useful for exploratory research in cognitive decision-making and learning behaviors.

The evaluation module of PyTDL uses consistency between experimental and simulated behavioral data to assess model performance, with specific metrics including consistency of decision-making tendencies and statistical consistency. It also supports user customization.

Estimating the trial-by-trial incentive value and prediction error with versatile learning and value functions

In goal-directed tasks, an agent seeks to maximize rewards, making decisions based on the environment state and the expected

(B) Flexible Value Function module incorporates various algorithms including SARSA, Q-learning, TD(λ) algorithms, and supports the implementation of user-defined value functions to estimate action values.

(C) Flexible Decision Policy module features traditional softmax and epsilon-greedy algorithms to convert Q-value into decision probabilities and supports customized decision functions such as dynamic exploration-exploitation policy.

(D) Action Simulation module integrates the Value Function and Decision Policy modules using the Markov Chain Monte Carlo (MCMC) method to generate simulated sequences of actions, enabling the analysis of decision-making processes.

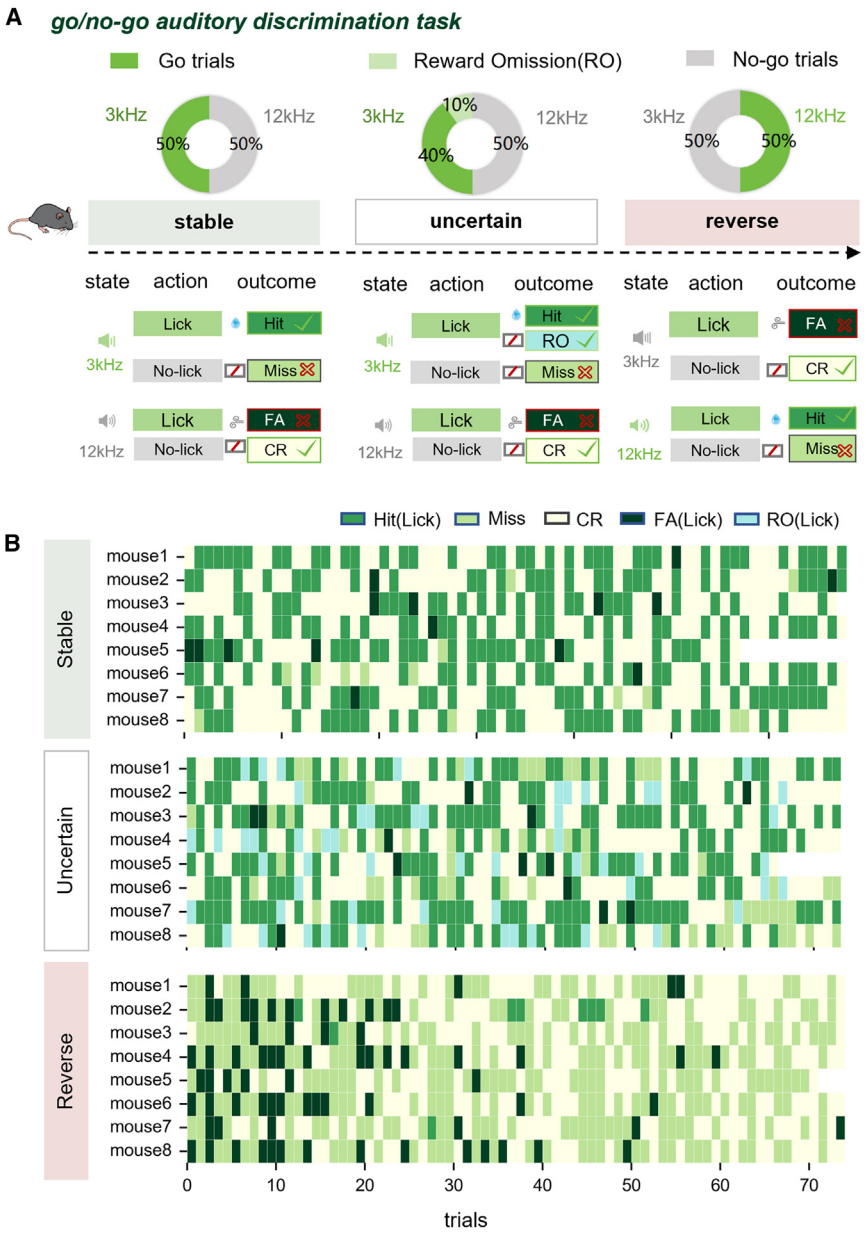


Figure 2. Performance of the go/no-go auditory discrimination task

(A) Schematic representation of the three sessions of the go/no-go auditory discrimination task, illustrating the modified outcome rules used to modulate error feedbacks: stable session (consistent rewards and punishments), uncertain session (random reward omissions), and reverse session (reversed reward-punishment contingencies).

(B) Action sequences of eight animals across the stable, uncertain, and reverse sessions, showing how their performance adapts to the different outcome rules.

rewards for available actions. These expected rewards arise from the cumulative consequences of specific actions, and decision adaptation manifests as a choice probability over multiple trials, making direct measurement of trial-by-trial decision adjustment challenging. However, this process is typically influenced by sequential state-action events and reward prediction errors, which can be modeled using RL methods. For instance, the TD algorithm values state-action pairs sequentially based on the reward obtained in each trial. A widely used value updating rule is the classical Q-learning model, which applies a fixed learning rate to minimize the reward prediction error. However, in cognitive tasks with non-stationary environments, the rigidity can hinder the model's ability to capture the subtle adjustments and nuanced decision dynamics inherent in complex decision scenarios.

For example, in the go/no-go auditory discrimination task (Figure 2A), animals were trained to decide whether to "lick" or "not lick" in response to two distinct tones. Each trial's state—whether it was a go cue or a no-go cue—was indicated by one of the two tones (3 kHz for go and 12 kHz for no-go), presented randomly in a 50/50 ratio. Animals were trained to lick in response to the go cue (3 kHz) to receive a sucrose reward, and to refrain from licking in response to the no-go cue (12 kHz) to avoid an air puff punishment. Reward and action codes are detailed in Table 1. In the stable session, the animals experienced a reliable association between licking the go cue (3 kHz) and receiving a sucrose reward (Figure 2B, upper panel). In the subsequent sessions, the animals were presented with unexpected outcomes, prompting a shift in their decision-making strategy in response to the feedback. In the uncertain session, the reward was omitted in 10% of go trials upon correct licking. This led to an increase in the number of miss trials in the later phase of the session (Figure 2B, middle panel). In the reverse session, the stimulus-reward contingency was completely reversed. Licking upon 3 kHz resulted in an air puff punishment rather than a reward. This led to a rapid cessation of licking after only a few trials (Figure 2B, bottom panel), demonstrating a rapid shift in decision-making strategy. Therefore, in the uncertain and reverse sessions, the animals exhibited a decreased likelihood of selecting the "lick" action upon 3 kHz. However, the shift away from "lick" was notably more rapid in the reverse session due to the consistent punitive feedback associated with the 3 kHz tone.

Throughout the learning and adapting process, the animals appear to utilize a nonlinear learning approach, wherein the state-action values are dynamically updated based on external feedback associated with specific signals. The adaptive mechanism enabled them to adapt to both mild (uncertain) and drastic

Table 1. Coding of actions and rewards in the go/no-go auditory discrimination task

state	Start cue	choice	Action	Action Code	reward	Reward Code
Stable / Uncertain	Go cue (3kHz)	Lick	Hit	1	sucrose	1.0
			RO	5	/	0.1
		No -lick	Miss	2	/	0.1
	No-go cue (12kHz)	Lick	FA	4	Air-Puff	−1.0
		No -lick	CR	3	/	0.1
Reverse	Go cue (12kHz)	Lick	FA	4	Air-Puff	−1.0
		No -lick	CR	3	/	0.1
	No-go cue (3kHz)	Lick	Hit	1	sucrose	1.0
		No -lick	Miss	2	/	0.1

(reversal) changes in task rules. PyTDL incorporates the stimuli, decision actions, and corresponding rewards experienced by mice into its decision-making modeling process (Figure 3A). Throughout the task, the go cue (3 kHz tone) plays a central role in guiding the animals' learning process. Any perturbance in the stimulus-reward contingency during the uncertain and reverse sessions—such as reward omission or punishment for the lick action in response to the 3 kHz tone—resulted in unanticipated outcomes, prompting a shift in decision-making strategies.

Here, we proposed a non-linear learning rate function in which the learning rate (∂_t) is dynamically adjusted based on the magnitude of the error feedback ($\partial_t = \theta'\partial_{t-1} + \theta|\delta_{t-1}|$), to mimic the dynamic process of the task (Figure 3A). Unlike the conventional Q-learning approach, which employs a fixed learning rate, our function allows for an increase in the learning rate when the feedback error is substantial and a decrease when the feedback is in line with expectations. This adjustment facilitates a balance between robustness and flexibility, accommodating dynamic environments more effectively. This custom-defined function has been seamlessly integrated into the value estimation module of the PyTDL framework, enabling simulation of results from both the conventional and dynamic models using the PyTDL package.

The trial-by-trial incentive values of lick action upon 3 kHz (Q^{lick}_{3kHz}) for each animal across the three sessions were estimated using either the conventional Q-learning algorithm or a custom-defined dynamic learning function within the PyTDL framework (Figure 3B). Utilizing the decision policy module of PyTDL, the action probabilities based on these estimated values were calculated using a softmax function (Figure 3C). The mice demonstrated a sustained high estimated value (Q^{lick}_{3kHz}) and decision probability for licking at 3 kHz during the stable phase, indicating a consistent and robust "lick" preference. In the uncertain session, where mice experienced some reward omission (RO) trials, both value estimates and lick probabilities exhibited fluctuations, although the mice maintained a high decision tendency to lick. In the subsequent reverse phase, where the feedback for the same licking behavior at 3 kHz shifted from reward to punishment, the previously high estimated value and selection probabilities for licking rapidly declined, indicating a low decision preference for licking at 3 kHz. The incentive values and lick probabilities estimated by both models showed a clear decline in the uncertain and reverse sessions due to the introduction of reward omission and the switch from reward to punishment. However, the results from the two models differed, and it was unclear which model more accurately represented the behavioral performance.

To evaluate the consistency of the model-estimated action sequence and the experiment data, the decision sequence can be simulated by the Markov Chain Monte Carlo method using the action simulation module of PyTDL. We simulated 5,000 repetitions of action sequences based on the cue sequence for each tested animal using both models with the parameters shown in Table 2. The simulated action sequences (average of 5,000 repetitions) and the empirical experimental data through the stable, uncertain, and reverse sessions were compared as the cumulative lick number upon the 3 kHz tone (Figure 4A), where the competing options of 'lick' and 'no-lick' were assigned values of 1 and 0, respectively. During the stable session, mice exhibited a pronounced inclination to 'lick', as shown by the steep slopes of the trend. In the uncertain session, the trend of choosing 'lick' fluctuated and slowed due to random reward omission in some trials. However, in the reverse session, after several punishments for licking upon the previously reward-associated tone, mice quickly adjusted their decision tendencies, leading to a rapid decline in the 'lick' tendency. The Q-learning model with a fixed learning rate could mimic the high lick probability in the stable and uncertain sessions but failed to capture the fast decision switching in the reverse session. In contrast, the dynamic learning function faithfully recaptured both the small decline in lick tendency in the uncertain session and the rapid switch-off of licking in the reverse session. The empirical lick probability of each animal through the stable, uncertain, and reverse sessions, calculated by a sliding window of 20–30 trials, fell within the 95% Limits of Agreement (LoA) in the sequences simulated with the dynamic learning function but not with the Q-learning function (Figure 4B).

The performance of all eight mice in our experimental dataset during the stable, uncertain, and reverse sessions was summarized (Figure 4C). The performance of the eight animals within the same session did not significantly differ between individuals (Friedman test, n = 8 mice). However, the lick probability upon the 3 kHz tone was significantly lower in the uncertain and reverse sessions compared to the stable session (Paired sample t-tests, n = 8 mice). The lick probability of each session simulated with the dynamic learning function exhibited a similar trend to the empirical data (Figure 4D).

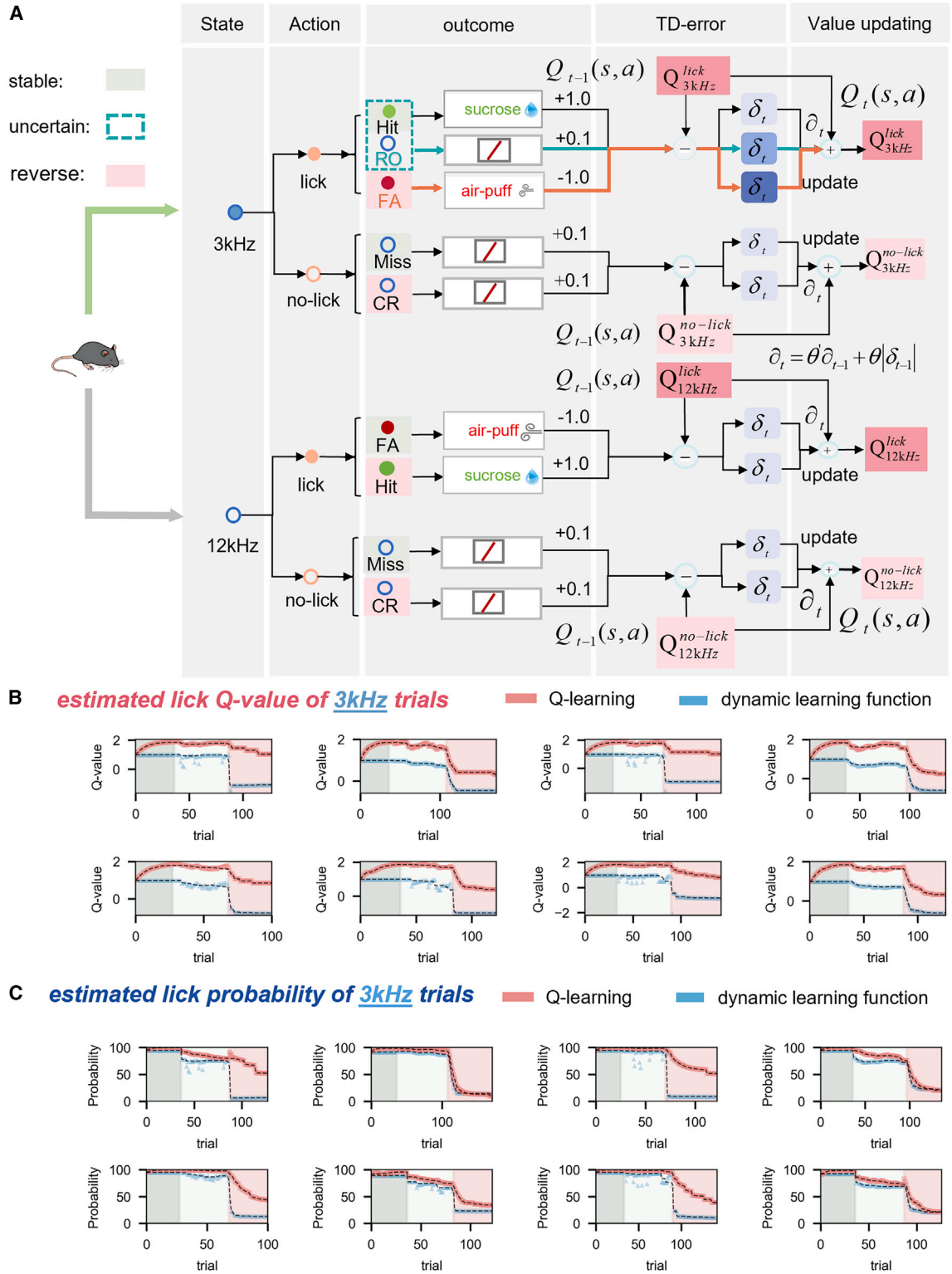


Figure 3. Estimated value and decision probability of 3kHz during the go/no-go auditory discrimination task

(A) The components of the TD learning model used to estimate trial-by-trial incentive state-action values across the three sessions of the task. Note the "lick" action in response to the 3 kHz tone, with reward omission (RO) in the uncertain session and air puff punishment (FA) in the reverse session, resulting in significant prediction errors and changes in value (ΔQ) for subsequent trials.

(legend continued on next page)

Table 2. Model-related parameters of 3kHz go/no-go auditory discrimination task Parameters

	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6	Mouse 7	Mouse 8
Q_0^{lick}	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
$Q_0^{no-lick}$	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02
∂_{init}	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
γ	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
θ'	0.60	0.60	0.60	0.60	0.60	0.87	0.60	0.60
θ	0.60	0.63	0.60	0.60	0.20	0.017	0.06	0.50
∂_{QL}	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
γ_{QL}	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
β_{stable}	3.0	2.6	3.0	3.0	3.0	2.5	3.0	2.2
$\beta_{uncertain}$	1.3	2.2	2.8	2.8	2.9	2.7	1.6	1.3
$\beta_{reverse}$	2.2	1.48	2.2	2.2	2.2	3.3	1.8	1.1

This experiment underscores the challenge that conventional TD models with rigid learning rates, such as Q-learning, face in capturing both the robustness of decision-making in uncertain situations and the flexibility required for rapid adaptation in reversed environments. To simulate the learning and decision-making processes of organisms in the real world, flexible value functions prove more adept at generalizing to new and complex situations, allowing the model to remain effective in previously unencountered scenarios. Therefore, PyTDL's robust modeling capability, particularly in depicting the learning behavior process, is due to the implementation of a more flexible learning and value-updating algorithm in the TD learning model. This allows neuroscientists to construct computational models with decision components derived from empirical behavior data of various dynamic cognitive tasks.

Predicting the action sequence with a flexible customer-defined decision policy to capture the adaptive behavior in complex cognitive tasks

The model's mapping functions between values and decision probabilities primarily reflect decisions based on past experiences. This is achieved through the incorporation of transformation functions, such as softmax and sigmoid, within the decision policy module. However, in actual biological decision-making processes, individuals not only choose based on the estimated values but also engage in a certain degree of exploration. When encountering decision conflicts in uncertain environments, animals often exhibit a preference for novel options, even if they are likely associated with lower rewards. In cognitive learning experiments, the variability and complexity of task environments challenge the classical, rigid decision strategies typical of RL, complicating the adaptation to complex biological cognitive decision processes. Therefore, to accurately capture the dynamic interplay between exploration and exploitation, it is imperative to develop a more flexible decision policy function that can account for the nuanced and diverse behaviors observed in cognitive decision-making.

To investigate the dynamic regulation of decision policies in the exploration-exploitation balance under complex decision tasks, we designed a reward probability-based two free-choice task. Mice were given the freedom to lick one of two ports, indicated by purple and green LEDs, and were rewarded with a sucrose solution at a predetermined probability. At the start of each session, a high (90%) and low (30%) reward probability was randomly assigned to each side, and this assignment was unknown to the animals. The mice had to explore the outcomes of different choices to determine which side was associated with the high reward probability (Figure 5A). Initially, the animals exhibited random-like choices (~first 40 trials), but subsequently, they suddenly developed a strong bias toward the side with the high reward probability (Figures 5B and 5C).

We incorporated the stimuli, decision actions, and corresponding rewards experienced by mice into the decision value iteration process using the SARSA algorithm (Figure 6A). In this task, the animal had the freedom to make either choice at the beginning of each trial, meaning the state of each trial was always the same, and the animal started with the same incentive value to choose either the green or the purple side. Either action could receive a sucrose reward or no reward, depending on the reward probability. As the animal accumulated experience of receiving or not receiving rewards from both sides, the incentive values of both sides differentiated. The trial-by-trial value to choose the high-reward side was estimated by the SARSA model according to the action sequence. The estimated value of licking the high-reward side rapidly increased in the early phase and remained high in the late phase (Figure 6B). In contrast, the action choice exhibited a sustained random-lick decision pattern in the early phase and a sudden increase in

(B) Trial-by-trial Q-values for the "lick" action in response to the 3 kHz tone (Q_{3kHz}^{lick}), estimated using the conventional Q-learning algorithm (pink dashed lines) and the customized dynamic learning function (blue dashed lines) across all three sessions. Each subplot represents data from one animal, with background colors indicating different task sessions.

(C) Changes in lick probability for the 3 kHz tone, estimated using the softmax function based on the Q-values from (B). Note that 3 kHz and 12 kHz tones are treated as independent states; thus, the "lick" action value for the 3 kHz signal is not updated when a 12 kHz trial occurs (see Figure S1). Panels (B) and (C) display all 3 kHz trials through all three sessions.

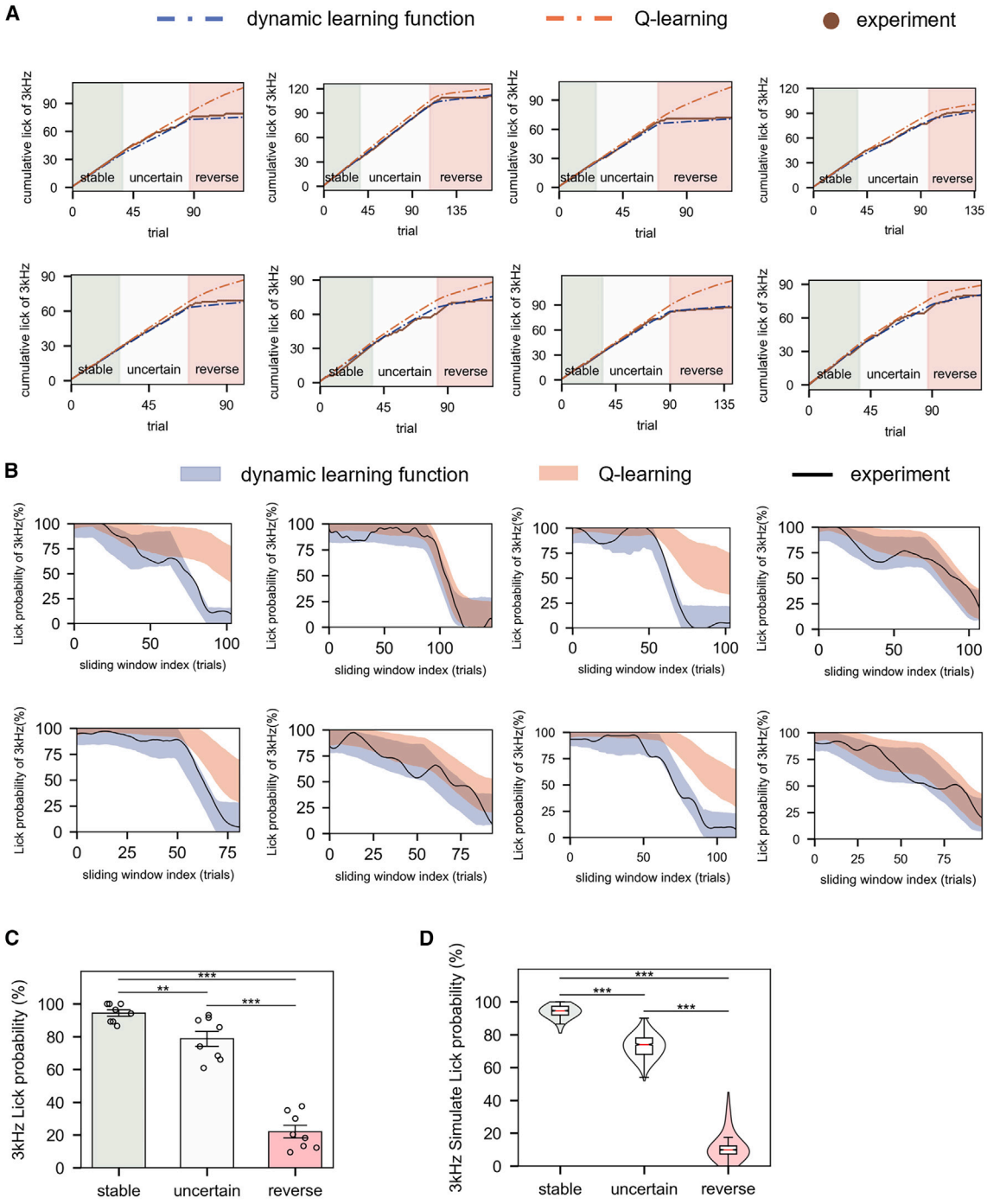


Figure 4. Evaluation of consistency between simulated action and empirical action sequences in the go/no-go auditory discrimination task

(A) Trends in cumulative licks for the 3 kHz tone from 8 mice compared to simulated cumulative licks, averaged over 5000 repetitions of action simulations using customized dynamic learning and Q-learning models. Each subplot represents data from one animal, with background colors indicating different task sessions.

(B) Experimental lick probabilities for the 3 kHz tone compared to the 95% LoA intervals of sequences simulated with customized dynamic learning (blue shaded area) and Q-learning (orange shaded area) models. Experimental lick probabilities were calculated using a sliding window technique with varying window sizes (25, 30, 20, 30, 20, 25, 20, 30) and a step size of 1.

(C) Average licking probabilities for the 3 kHz tone in the stable, uncertain, and reverse sessions. Each circle represents an individual animal. Paired sample t-tests indicate significant differences between sessions (n = 8 mice). The Friedman Test showed no significant differences in licking probability within the same session, indicating consistent behavior among different mice in the same session.

(legend continued on next page)

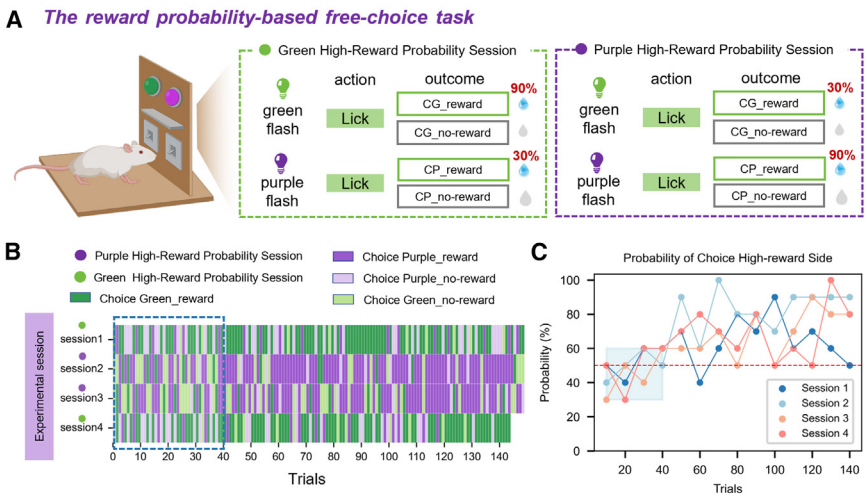


Figure 5. Performance of the reward probability-based two free-choice task

(A) The task scheme of the reward probability-base two free-choice task. The green and purple LEDs were associated with high reward probability (90%) in the Green High-Reward-Probability (Green HRP) and Purple High-Reward-Probability (Purple HRP) sessions, respectively.

(B) Action sequence of the one example experiment animal performing 4 randomly assigned Green HRP or Purple HRP sessions.

(C) Probability of selecting the high-reward side, shown in bins of 10 trials. The initial stage of each session (first 40 trials), when the animal exhibited random-like exploration, was highlighted by the blue dashed box and blue shade in (B) and (C).

preference to the high-reward side afterward (Figure 5C), suggesting that the animals initially employ a more exploratory strategy and later a more value-based strategy.

To capture the two-phase preference in decision strategy, we implemented a dynamic exploration rate function in the decision policy module of PyTDL. In this function, the exploration was determined by the difference in incentive values between two competing options (ΔQ) according to a transformation sigmoid function ($\xi_t = 1 - (1 + \exp(-\omega(\overline{\Delta Q_t} - B)))^{-1}$). When observed ΔQ remained small, random exploration strategies dominated. As ΔQ increased with trials, it resulted in an accelerating preference for high-value actions (Figure 6A). The probability of choosing the high-reward side estimated by the custom-defined dynamic exploration function was right-shifted compared to that estimated by the value-based (softmax) function (Figure 6C), suggesting that the former better captured the exploratory behavior in the early phase.

To compare the process of establishing a choice preference between the empirical behavioral data and the simulated action sequences derived from the TD learning models, the motivational values were set to 1 and -1 for the competing options associated with the high (90%) and low (30%) reward probability, respectively. The cumulative motivational value reflected the net choice preference established through trials conducted during the early phase of each session (Figure 7A). Notably, when mice initially encountered competing options with unknown reward probability, they tended to randomly explore each choice for several trials, exhibiting no clear preference in their decision-making at the beginning of the session, as shown by the horizontal line representing the cumulative motivational values. This type of exploratory behavior is typical in the process of cognitive learning. The action sequence of each model was simulated using the MCMC method for 5000 repetitions. The averaged simulation sequences were generated using the classic softmax value mapping function, the epsilon-greedy function, and the dynamic exploration function (see Table 3 for parameter settings). Both the softmax and epsilon-greedy strategies rapidly developed a preference for the high-reward option, failing to capture the exploratory effort to discern the reward probabilities and identify the side associated with higher reward probability (Figure 7A, orange and pink dash-dotted lines). In contrast, the action sequences simulated by the dynamic exploration model (Figure 7A, blue dashed lines) aligned more closely with the experimental data. To better compare the simulated sequences to the actual behavioral data, a sliding window of 35 trials was applied to the empirical action sequence to obtain the decision probability. The experimental data fell within the 95% LoA of the decision probability sequences simulated by the dynamic exploration model, but not within the 95% LoA of the softmax and epsilon-greedy models (Figure 7B).

In summary, we compared the probability of choosing the high-reward side in the early phase (the first 30 trials) and in the late phase (the 100th to 129th trials) of each session. The choice bias toward the high-reward side was much stronger in the late phase (Figure 7C, paired samples t-test, 10 sessions from 4 animals). The Friedman Test showed no significant differences within the same phase across sessions and animals, confirming stable and reliable performance across different sessions in the same phase. The choice probabilities simulated by the dynamic exploration function exhibited a similar differential preference in the early and late phases (Figure 7D).

The dynamic exploration rate within the customizable decision functions indicates the probability of biological entities making random decisions. This rate quantifies the contributions from

(D) The violin plot shows licking probabilities for the 3 kHz tone from simulations using the dynamic learning TD model. Each response to the tested cue sequence from 8 experimental animals was simulated 5000 times. Simulation results show significant differences between sessions, suggesting task rule changes significantly influence decision-making behavior (Paired sample t-tests). *$p < 0.05$, **$p < 0.01$, ***$p < 0.001$. Data are presented as (C) mean ± s.e.m. or (D) violin plots (center line represents the median; the box limits represent the upper and lower quartiles; whiskers indicate 1.5 × interquartile range; the body of the violin displays the data's density distribution). Detailed statistics are presented in Table S1.

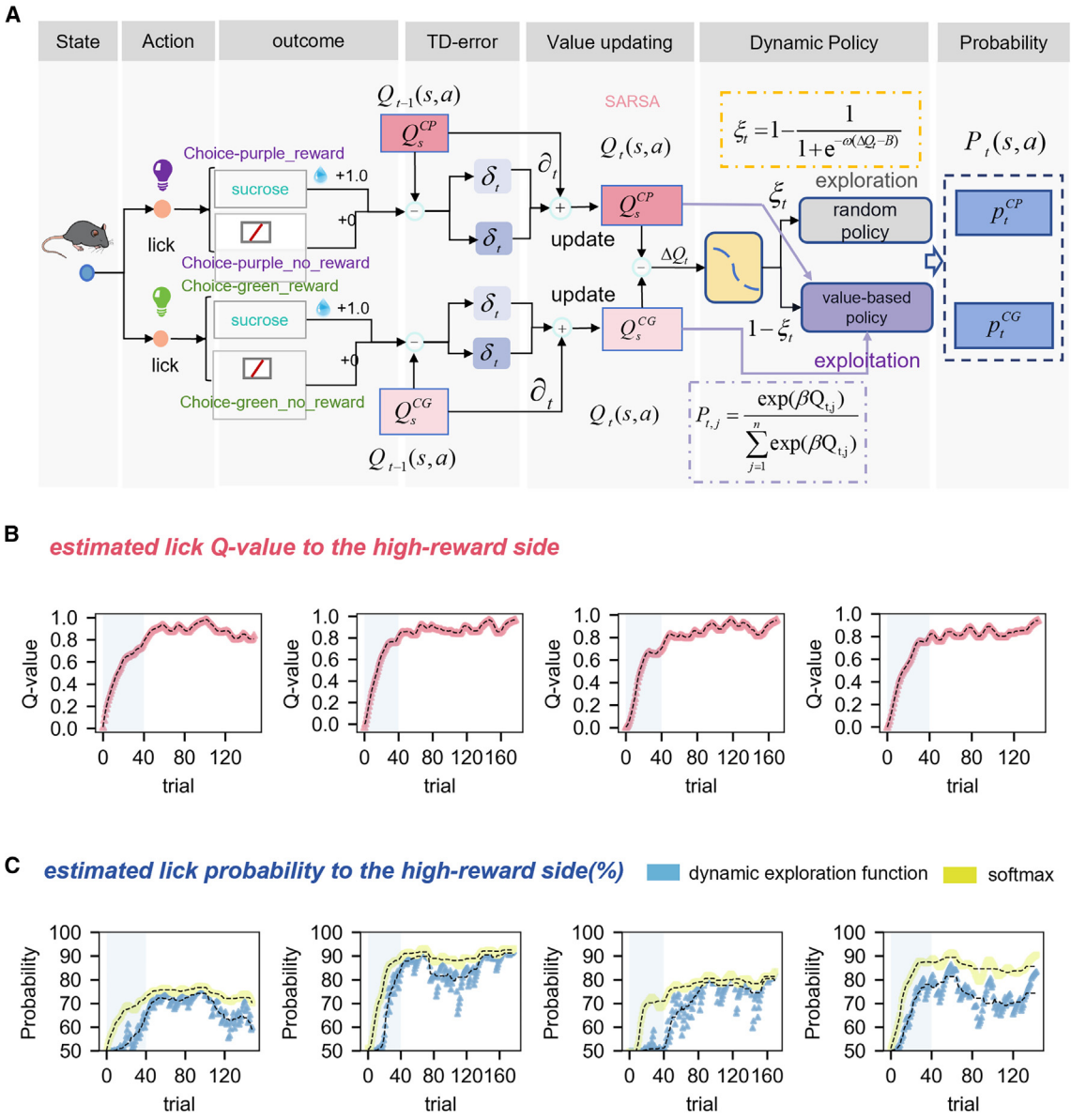


Figure 6. Estimated value and decision probability during the reward probability-based two free-choice task

(A) The components of the TD learning model estimate trial-by-trial incentive state-action values and decision probabilities in the free-choice task. The flexible custom decision policy dynamically adjusts the exploration rate based on the value difference between competing options, effectively characterizing the behavioral process of gradually forming decision preferences.

(B) Trial-by-trial Q-value estimates for choosing to lick the high-reward-probability side, calculated using the SARSA algorithm based on the empirical behavior sequence.

(C) Changes in the probability of licking the high-reward-probability side, estimated by the softmax function (yellow dashed lines) and the customized dynamic exploration function (blue dashed lines) based on the Q-values. Each subplot represents one recorded session.

both value-based decision strategies and random exploration tactics, allowing for a flexible shift between the two under appropriate conditions. Researchers can also design various transformation functions based on specific experimental paradigms or decision tasks to explore how past rewards and information dynamically guide current decision making in value-based decision strategies.

PyTDL can be used with GUI and API

PyTDL, a versatile Python framework for modeling and simulating biological cognitive decision-making processes, integrates the Temporal-Difference algorithm, showcasing its flexibility and applicability in handling complex, unknown, or continuous biological decision scenarios. The framework is designed to assist users in constructing their models efficiently,

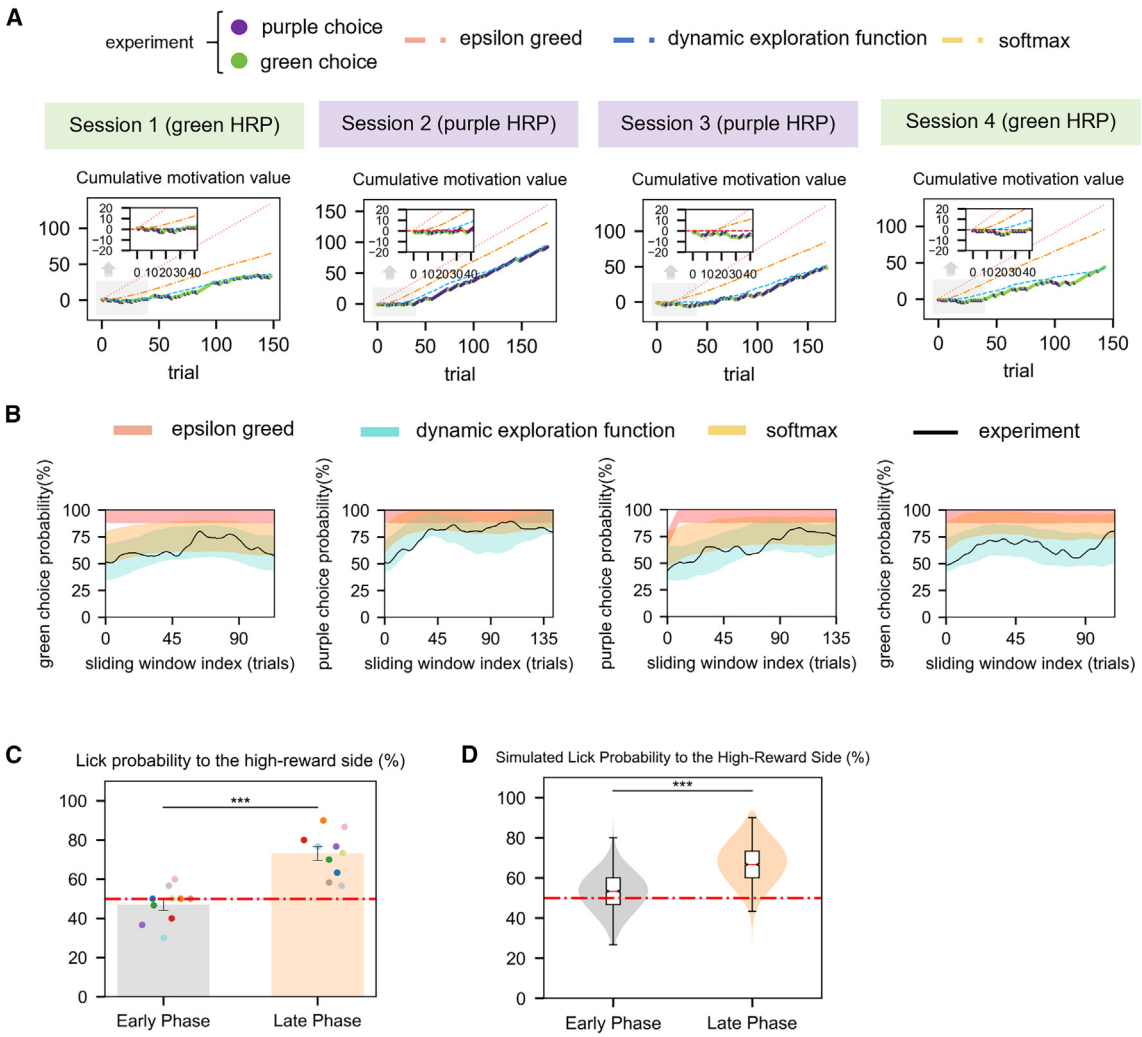


Figure 7. Evaluation of the consistency of simulated and behavioral action sequences in the reward probability-based two free-choice task

(A) The cumulative motivational value to lick to the high-reward-probability side of each session. Lick to the high-reward-probability side and the low-reward-probability side were assigned to 1 and -1, respectively. The simulated cumulative motivational value was derived by averaging the results of 5000 iterations of action simulations. The initial stage of first 40 trials were highlighted by the gray boxes and enlarged in the insets.

(B) The probability to choose the high-reward-probability side of each session of experiment was compared to the 95% LoA interval for the sequences simulated with the customized dynamic exploration, epsilon-greedy and softmax algorithms, which were showed as blue, pink and orange shades, respectively. The lick probability of experiment data was obtained by sliding window technique, with a window size of 35, and a step size of 1.

(C) The average probabilities to choose the high-reward side in the early decision phase (the first 30 trials) and the late phase (the 100th to 129th trials) of each session. (Paired sample t-tests, $n = 10$ sessions from 4 mice). The Friedman Test resulted in no significant differences in the probabilities of lick to the high-reward side within the same phase, indicating stable and consistent behavior among different session in the same phase ($n = 10$ sessions from 4 mice).

(D) The violin plot shows the probability to choose the high-reward likelihood side in the early and late phase from the action sequence simulated by the dynamic exploration model ($n = 5{,}000$ repetitions). $^{*}p < 0.05$, $^{**}p < 0.01$, $^{***}p < 0.001$. Data are presented as (C) mean ± s.e.m. or (D) violin plots (center line represents the median; the box limits represent the upper and lower quartiles; whiskers indicate 1.5 × interquartile range; the body of the violin displays the data's density distribution). Statistical details are presented in Table S1.

supported by a user-friendly software package. The toolkit's structure reflects the framework's workflow and emphasizes modularity, allowing easy application of model components across various contexts. (Figure 8A).

PyTDL offers a user interface that comprises three main functionalities—Build model, Simulation, and Evaluation— effectively simplifying the model adjustment process (Figure 8B). Enhancing flexibility and customization, PyTDL includes Qt Designer interface design files and modifiable main functions, allowing users to tailor the toolkit to their specific research needs. The framework aims to provide researchers with a tool for quick assembly through key functional modules. Users can dynamically adjust the parameters and functions of each module in real-time based on model results, thereby constructing a representational model that closely aligns with actual empirical data.

Table 3. Model-related parameters of the reward probability-based two free-choice task

	Session 1	Session 2	Session 3	Session 4
Q_0^{CP}	0.0	0.0	0.0	0.0
Q_0^{CG}	0.0	0.0	0.0	0.0
∂	0.1	0.1	0.1	0.1
γ	0.0	0.0	0.0	0.0
ω	15	15	25	7
B	0.50	0.45	0.50	0.40
β	1.8	3.7	2.3	3.6
ε_{greed}	0.1	0.1	0.1	0.1

Released under the MIT license, PyTDL supports both open-source and commercial uses and requires Python version 3.8 or higher. It depends on libraries including NumPy, Pandas, Matplotlib, PyQt5, SciPy, Seaborn and scikit-learn. For cognitive decision-making data modeling with PyTDL, it's crucial to standardize raw data into a CSV format or generate a Pandas DataFrame using Python functions. The data file should include labels (outcomes) related to behavioral decisions to ensure compatibility with the toolkit's functionalities. The source code is available on GitHub at https://github.com/hus001/PyTDL, where users can contribute and seek assistance. The repository includes the full framework source code, the demo Jupyter Notebook scripts to execute each module with customized functions, GUI source code, behavioral dataset of cognitive tasks, and comprehensive API documentation.

DISCUSSION

We introduce a versatile toolkit tailored for modeling cognitive processes of decision making with the TD algorithm from model-free RL, offering customizable estimation functions to fit specific research needs. The toolkit accommodates customizable value estimation and decision probability conversion functions, enabling researchers to precisely define the dynamics of the learning process and the correlation between values and selection probabilities under various conditions. This enhances the toolkit's adaptability to diverse biological cognitive decision-making scenarios. This flexibility extends to incorporating exploration-based decision-making strategies, allowing for the simulation of organisms' exploration and utilization behaviors during learning with dynamic adjustments. The framework consists of modules designed to probabilistically characterize the decision-making process in alignment with the Markov decision-making process.

To illustrate the toolkit's application, we employed it to model decision processes in two distinct cognitive tasks, incorporating customized value estimation and decision policy functions. In the go/no-go auditory discrimination task, conventional Q-learning struggles to adapt to varying learning rates in response to different degrees of unexpected outcomes. Implementing a dynamic learning rate function within the value estimation module is crucial to emulate decision adjustments across all three sessions. In the reward probability-based free-choice task involving unknown reward probabilities, subjects exhibit random-like exploration of reward probability at the session outset. Traditional decision policies, such as value-based softmax or epsilon-greedy functions, prove inadequate in capturing the cumulative trials needed to assess reward probability. Consequently, a dynamic value-based exploration-exploitation function must be integrated into the decision policy module to replicate this process. The toolkit also includes an evaluation module specifically designed to assess the agreement between simulated decision sequences and real-world behavior sequences through both propensity and statistical analysis. In the propensity analysis, decision categories are assigned positive or negative values, with their cumulative sums calculated to illustrate decision-makers' tendencies. This method assesses the consistency of decision propensities by juxtaposing the cumulative variations in experimental data against the average variations found in simulated sequences, which are generated using the MCMC method. In parallel, the statistical analysis employs a sliding window to divide decision sequences into segments, facilitating the estimation of local decision probabilities. The 95% LoA are then utilized to verify the statistical consistency between the sequences.

The introduction of an open-source toolkit with versatile modeling framework serves the purpose of facilitating the streamlined development of models for the quantitative characterization of cognitive learning and decision processes. This toolkit stands as a valuable resource, addressing inherent challenges in estimation of biological process of cognitive decision-making with computational algorithm. The flexible framework helps neuroscientists to adapt it for more complex and fine-tuned tasks which are designed to address specific cognitive functions. The model parameter and results reveal the psychological process underlying the behavioral decisions, and the correlation between them and the functional properties of neural populations can help us understand the neural mechanism for higher cognitive functions.

Limitations of the study

The TD learning algorithm excels in elucidating the direct correlation between behavioral actions and ensuing rewards, which makes it particularly valuable for modeling trial-by-trial cognitive learning and decision-making processes. However, this algorithm oversimplifies the complex nature of real-world decision making by not accounting for various internal and external factors such as emotions, motivation, and social interactions. Consequently, PyTDL provides a framework based on TD learning to investigate decision-making processes under relatively stable internal conditions. The toolkit focuses on revealing the learning process rooted in trial-and-error experiences and exploring how the brain utilizes reward prediction errors. Nevertheless, the capacity to address the multifaceted nature of decision-making in more dynamic and contextually varied scenarios is ultimately limited by the design of the cognitive models. PyTDL provides researchers with a toolkit for easily implementing their models within the TD learning framework.

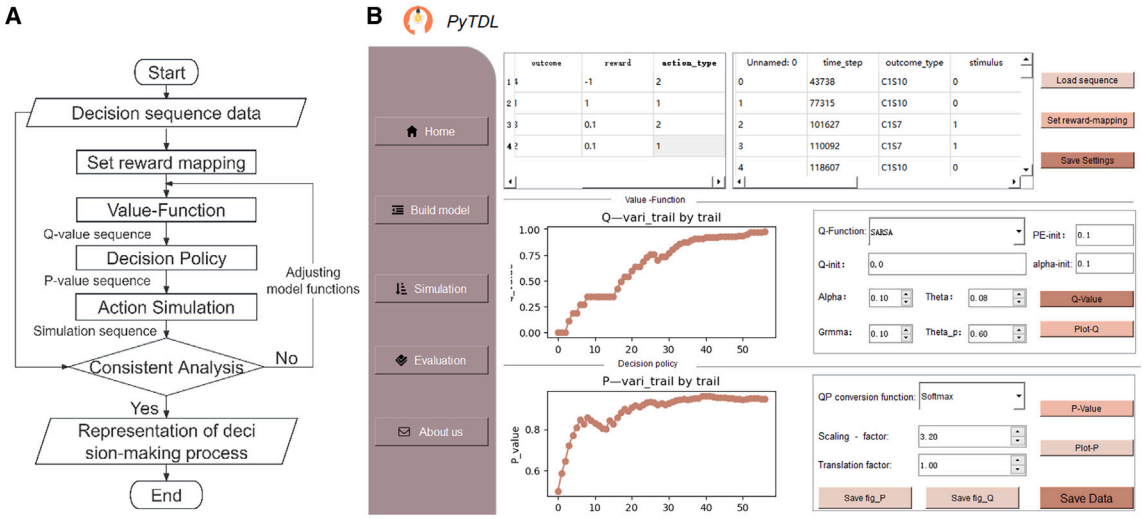


Figure 8. The working flow and GUI of PyTDL

(A) Schematic representation of the working flow in PyTDL for modeling decision sequences. The diagram illustrates the sequential process from inputting experimental data through the flexible value function and decision policy modules to simulating and analyzing decision-making sequences.

(B) Snapshot of the GUI of PyTDL. The image shows the main interface where users can interact with the various modules, configure parameters, and visualize results.

RESOURCE AVAILABILITY

Lead contact

Further information and requests for resources should be directed to and will be fulfilled by the lead contact, Yunyun Han (yhan@hust.edu.cn).

Materials availability

This study did not generate new unique reagents.

Data and code availability

- Data: The data supporting this study are available on GitHub at https://github.com/hus001/PyTDL/tree/main/experimental%20data.
- Code: The code used for data analysis and PyTDL in this study is also available on GitHub at https://github.com/hus001/PyTDL.
- Other Items: Any additional information required to reanalyze the data reported in this paper is available from the lead contact upon request.

ACKNOWLEDGMENTS

This work was supported by Interdisciplinary Research Program of HUST (2024JCYJ007), the National Natural Science Foundation of China (52475017, 32271082 and 52005191), Hubei Provincial Natural Science Foundation (2022CFA067, 2023AFB776), Major Scientific Research Facility Project of Jiangsu Province (BM2022010) and State Key Laboratory of Intelligent Manufacturing Equipment and Technology (IMETZZ2024008).

AUTHOR CONTRIBUTIONS

Y.H. and J.L. designed the project; Q.W. wrote the software and analyzed the data; X.Y. and M.Z. carried out the animal experiments; K.W. wrote the GUI; and Y.H., J.L., Q.W., and M.Z. wrote the manuscript.

DECLARATION OF INTERESTS

The authors declare no competing interests.

STAR★METHODS

Detailed methods are provided in the online version of this paper and include the following:

- KEY RESOURCES TABLE
- EXPERIMENTAL MODEL AND STUDY PARTICIPANT DETAILS
  - Training and testing of the go/no-go auditory discrimination task
  - Training and testing of the reward probability-based two free-choice task
- METHOD DETAILS
  - Value function
  - Decision policy
  - Simulation based on MCMC method and evaluation
  - Consistency of decision-making tendencies
  - Statistical consistency
- QUANTIFICATION AND STATISTICAL ANALYSIS

SUPPLEMENTAL INFORMATION

Supplemental information can be found online at https://doi.org/10.1016/j.isci.2024.111600.

Received: February 25, 2024
Revised: August 13, 2024
Accepted: December 11, 2024
Published: December 14, 2024

REFERENCES

1. Tai, L.-H., Lee, A.M., Benavidez, N., Bonci, A., and Wilbrecht, L. (2012). Transient stimulation of distinct subpopulations of striatal neurons mimics changes in action value. Nat. Neurosci. *15*, 1281–1289.
2. Sutton, R.S. (1988). Learning to predict by the methods of temporal differences. Mach. Learn. *3*, 9–44.
3. Tobia, M.J., Guo, R., Gläscher, J., Schwarze, U., Brassen, S., Büchel, C., Obermayer, K., and Sommer, T. (2016). Altered behavioral and neural

responsiveness to counterfactual gains in the elderly. Cognit. Affect Behav. Neurosci. *16*, 457–472.

4. Mao, H., Negi, P., Narayan, A., Wang, H., Yang, J., Wang, H., Marcus, R., Khani Shirkoohi, M., He, S., and Nathan, V. (2019). Park: An open platform for learning-augmented computer systems. Adv. Neural Inf. Process. Syst. *32*.
5. Stetsenko, A., and Koos, T. (2023). Neuronal implementation of the temporal difference learning algorithm in the midbrain dopaminergic system. Proc. Natl. Acad. Sci. USA *120*, e2309015120.
6. Lee, D., Seo, H., and Jung, M.W. (2012). Neural basis of reinforcement learning and decision making. Annu. Rev. Neurosci. *35*, 287–308.
7. Sutton, R.S., and Barto, A.G. (2018). Reinforcement Learning: An Introduction (MIT press).
8. Platt, M.L., and Glimcher, P.W. (1999). Neural correlates of decision variables in parietal cortex. Nature *400*, 233–238.
9. Barraclough, D.J., Conroy, M.L., and Lee, D. (2004). Prefrontal cortex and decision making in a mixed-strategy game. Nat. Neurosci. *7*, 404–410.
10. Pastor-Bernier, A., and Cisek, P. (2011). Neural correlates of biased competition in premotor cortex. J. Neurosci. *31*, 7083–7088.
11. Sul, J.H., Kim, H., Huh, N., Lee, D., and Jung, M.W. (2010). Distinct roles of rodent orbitofrontal and medial prefrontal cortex in decision making. Neuron *66*, 449–460.
12. Kim, H., Sul, J.H., Huh, N., Lee, D., and Jung, M.W. (2009). Role of striatum in updating values of chosen actions. J. Neurosci. *29*, 14701–14712.
13. Amo, R., Matias, S., Yamanaka, A., Tanaka, K.F., Uchida, N., and Watabe-Uchida, M. (2022). A gradual temporal shift of dopamine responses mirrors the progression of temporal difference error in machine learning. Nat. Neurosci. *25*, 1082–1092.
14. Schultz, W., Dayan, P., and Montague, P.R. (1997). A neural substrate of prediction and reward. Science *275*, 1593–1599.
15. Montague, P.R., Dayan, P., and Sejnowski, T.J. (1996). A framework for mesencephalic dopamine systems based on predictive Hebbian learning. J. Neurosci. *16*, 1936–1947.
16. Montague, P., Dayan, P., Nowlan, S.J., Pouget, A., and Sejnowski, T. (1992). Using aperiodic reinforcement for directed self-organization during development. Adv. Neural Inf. Process. Syst. *5*.
17. Montague, P.R., and Sejnowski, T.J. (1994). The predictive brain: temporal coincidence and temporal order in synaptic learning mechanisms. Learn. Mem. *1*, 1–33.
18. Sejnowski, T.J., Dayan, P., and Montague, P.R. (1995). Predictive Hebbian Learning. In Proceedings of the eighth annual conference on Computational learning theory, pp. 15–18.
19. Schultz, W. (1998). Predictive reward signal of dopamine neurons. J. Neurophysiol. *80*, 1–27.
20. Hogeveen, J., Mullins, T.S., Romero, J.D., Eversole, E., Rogge-Obando, K., Mayer, A.R., and Costa, V.D. (2022). The neurocomputational bases of explore-exploit decision-making. Neuron *110*, 1869–1879.e5.
21. Cockburn, J., Man, V., Cunningham, W., and O'Doherty, J.P. (2022). Novelty and uncertainty interact to regulate the balance between exploration and exploitation in the human brain. Neuron *110*, 2691–2702.e8. https://doi.org/10.1016/j.neuron.2022.05.025.
22. Gershman, S.J. (2019). Uncertainty and exploration. Decision *6*, 277–286.
23. Chen, X., and Li, H. (2017). ArControl: an arduino-based comprehensive behavioral platform with real-time performance. Front. Behav. Neurosci. *11*, 244.
24. Chen, W., Liang, J., Wu, Q., and Han, Y. (2024). Anterior cingulate cortex provides the neural substrates for feedback-driven iteration of decision and value representation. Nat. Commun. *15*, 6020.
25. Ito, M., and Doya, K. (2009). Validation of decision-making models and analysis of decision variables in the rat basal ganglia. J. Neurosci. *29*, 9861–9874.

STAR★METHODS

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Deposited data		
Behavioral performance data for two cognitive decision-making tasks	This paper	https://github.com/hus001/PyTDL/tree/main/experimental%20data
Software and algorithms		
Python version 3.8	Python Software	https://www.python.org
PyTDL toolkit	This study	https://github.com/hus001/PyTDL

EXPERIMENTAL MODEL AND STUDY PARTICIPANT DETAILS

C57BL male mice (2–4 months old, 20–30 g, Vital River) were used in this study. All procedures were performed following protocols approved by the Animal Care and Use Committee at Huazhong University of Science and Technology. The animals were kept under a 12-h light/dark cycle and all experiments were performed during the light cycle. The animals were under food restriction for 1 week before training for the tasks, and their body weight was controlled above 80% of the initial body weight through the entire experiments. The behavior apparatus was controlled by a Arduino-based system controlled by ArControl software.[23] In the go/no-go experiment, the animals were trained to perform the task with a head-fixed position for future two-photon microscopy recordings. The surgery to implant the head bars was performed under full anesthesia as described before[24] and the mice were singly housed after surgery to avoid fighting or damaging the head-plate. Their health conditions were examined daily for at least one week after surgery.

Training and testing of the go/no-go auditory discrimination task

After a minimum of one week of post-surgery recovery, mice were handled daily for 10 min to 1 h for at least 3 days before the initial training. Mice were subjected to food restriction to achieve a body weight of 80%–90% of their free-feeding body weight and subsequently trained on the auditory go/no-go discrimination task. Behavioral experiments were controlled and recorded using an Arduino-based platform and ArControl software.[23] The training process consisted of three phases: habituation, association, and discrimination. In the habituation phase, mice were trained to reliably trigger a reward of 5μl 10% sucrose water (Thermo Fisher Scientific) by licking the spout. This phase familiarized the mice with the task apparatus and the reward process. In the subsequent association phase, only the go cue (3 kHz, 70 dB) was presented for 1 s, followed by a 2 s response window in which the mice could be rewarded with sucrose solution by licking the spout. To prevent the continuous liking, the stimulus began only after the mice had ceased licking for at least 4–6 s. The association phase took about 3–5 days, during which mice learned to suppress impulsive licking outside the stimulus and response windows. Then in the subsequent discrimination phase, mice were trained to discriminate between two different tones: the go cue (3 kHz, 70 dB) and the no-go cue (12 kHz, 70 dB). The go and no-go cues were presented in a random order with a 4–6 s inter-trial interval. The reward was delivered only for licks in the response window following the go cue (Hit). Incorrect lickings in response to the no-go cue (False Alarms, FA) were punished with a 0.2 s air puff (20 psi) delivered through a flattened 25 g needle directed at the whiskers. Mice typically learned to perform the task within 3–10 days, completing 150–200 trials per day. The training was considered completed when the animal achieved consistent and high performance (d' (d-prime) $\geq$1.5) for at least three consecutive days. Only animals that completed training within 3 weeks were eligible for recording. Each of the eight animals was tested with the stable-uncertain-reversal session once.

Training and testing of the reward probability-based two free-choice task

Mice were first guided to lick at one of the valves, receiving a reward of 5μl sucrose solution every time regardless of the side chosen. This initial phase familiarized the animals with the task and reward system. The following training with differential reward probability was conducted in two stages, with one session per day and 200–300 trials per session. On each day of Stage 1, one side (e.g., indicated by the purple LED) was associated with a high-reward probability. Mice were trained to demonstrate a clear preference for this side. Once a clear preference was established, the high-reward side was switched to the other cue (e.g., a

different LED color). This process continued until mice learned the new high-reward side. Subsequent sessions involved switching the high-reward side to the opposite side whenever the mice showed a clear preference for the high-reward side at the end of the previous day's session. Animals advanced to Stage 2 if they could consistently learn the new high-reward side within one day for three consecutive days. In stage 2, the high-reward side was randomly assigned to one of the two sides each day. If animals exhibited signs of learning the "switching" rule, such as a clear preference for the low-reward likelihood side of the previous day at the beginning of the session, additional training sessions were conducted with randomly assigned high-reward side. The training was considered complete when animals no longer exhibited a preference associated with the condition of the previous day at the beginning of the session for at least three consecutive days. Only those animals that met this criterion were included in the experiment phase.

METHOD DETAILS

Value function

A sequential decision problem with observable states and random transformations can be described as a Markov decision process (MDP), which is characterized by the following parameters:

S: the set of states.

A: the set of actions.

P: the probability to transition from state s_t to s_{t+1} when taking action a_t.

R: the immediate reward after the transition from state s_t to s_{t+1} when taking action a_t.

γ: a discount factor in [0, 1] that adjusts future rewards to present value.

The MDP provides a structured framework for formulating decision-making problems in environments characterized by stochastic dynamics and uncertainty. It explicitly models the environment through state transitions and rewards, facilitating efficient decision-making by an agent through the use of a value function. The action-value function $Q_\pi(s_t, a_t)$ assesses the expected reward for selecting a specific action a_t in various states s_t . The policy $\pi(a|s)$ determines the probability of executing each action in a given state. This methodology encourages the agent to make decisions based on the current state, independent of past state histories, thereby streamlining model construction and computational procedures.

In the TD learning algorithm, the agent continuously interacts with the environment, observing states, taking actions, and receiving rewards at each timestep. It uses the experiences from these interactions to incrementally learn and update its value functions. Specifically, the TD algorithm updates the estimation of the value function by calculating the TD error.

$$Q_{t+1}(s,a) \leftarrow Q_t(s,a) + \partial\delta_t \quad \text{(Equation 1)}$$

The parameter δ_t is defined as the reward prediction error. This part of the error reflects the inherent uncertainty of the environment. The prediction error is a key component in the TD algorithm. It represents the difference between the predicted value of a lead or action and the actual outcome or reward benefit. This mechanism allows for incremental updates, making the function more adaptable and enabling the algorithm to efficiently learn from the experience of each individual step. Here, are a few commonly used algorithms for estimating value functions.

A representative of the TD algorithm is the Q-learning algorithm, which is used to estimate the action value function $Q(s,a)$. The Q-learning algorithm works by constantly performing actions, observing the reward and the next state, and using the above update rules to update the estimation of the value function. The updated rules for Q-learning are:

$$Q_{t+1}(s,a) \leftarrow Q_t(s,a) + \partial[r + \gamma\, max(Q_t(s',a')) - Q_t(s,a)] \quad \text{(Equation 2)}$$

where $Q(s,a)$ is a functional estimate of the value of the state action pair (s, a) and ∂ is the learning rate, r is an instant reward for performing action a in the current state. s' is the next state to enter after action is executed. γ is the discount factor, which is used to balance the importance of the value function estimate of the immediate reward and subsequent state. $Q(s',a')_{max}$ represents the Q-value of the action with the highest expected return among all possible actions in the next state s'.

In the TD learning framework, the SARSA algorithm updates the estimates of Q-values using the reward prediction error δ_t on a trial-by-trial basis. SARSA's updated rules are:

$$Q_{t+1}(s_t,a_t) \leftarrow Q_t(s_t,a_t) + \partial[r + \gamma Q_t(s_{t+1},a_{t+1}) - Q_t(s_t,a_t)] \quad \text{(Equation 3)}$$

where $Q(s',a')$ is the estimate of the action value for taking action in the next state and other parameters are the same as Q-learning.

Classical TD algorithms often use a fixed learning rate, which can hinder their ability to adapt to uncertain or novel situations in simulating biological decision-making. To overcome this, PyTDL has introduced a dynamic learning rate mechanism, enhancing the model's adaptability to environmental changes. This approach allows for flexible parameter updates, enabling the model to learn efficiently in stable conditions with a lower learning rate, and quickly adjust to new or uncertain situations by increasing the learning rate. This method has been successfully applied to the go/no-go auditory discrimination task study

discussed in this paper, demonstrating its effectiveness. The specific representation of the dynamic learning function is as follows:

$$\begin{cases} Q_{t+1}(s_t, a_t) \leftarrow Q_t(s_t, a_t) + \partial_t(r + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t)) \\ \partial_t = \theta' \partial_{t-1} + \theta |\delta_{t-1}| \end{cases} \quad \text{(Equation 4)}$$

where θ' is a discount factor for the learning rate and θ is the error conversion coefficient, which is used to dynamically integrate the error size into the learning rate change process; and ∂_{t-1} is the last learning rate.

The valuation functions evolved from the TD algorithm are not elaborated here in detail. Our intention is to provide several classical valuation algorithms and descriptions of custom valuation functions to explain the computational principles of the value function module. We encourage users to quickly establish customized valuation functions that meet their research needs, based on the characteristics of empirical data and the specific cognitive decision-making tasks at hand.

Decision policy

Within the domain of biological cognitive decision-making modeling, the process of converting the valuation of decision sequences (Q-values) into probability values (p-values) acts as a bridge that connects mathematical models with the genuine complexity of biological behavior. This conversion is essential to endow the model with the flexibility and randomness of decision-making, highlighting those organisms are not always rational calculators in the face of multiple choices. Instead, they are dynamic learners who embrace the exploration of the unknown and the introduction of randomness into the decision-making process.

In PyTDL, the decision policy module employs this probabilistic approach, enabling the model to simulate how organisms balance exploration and exploitation in uncertain environments. Specifically, the dynamic adjustment of the decision-making policy aims to dynamically adjust the exploration rate in combination with changes in the Q-value series. This approach consists of a stochastic strategy and a value strategy. The formula is as follows:

$$\pi(a|s) \Leftarrow \begin{cases} P(a = a_i) = \frac{1}{n} & P_\pi = \xi_t \\ P(a = a_i) = f(Q_t; \kappa) & P_\pi = 1 - \xi_t \end{cases} \quad \text{(Equation 5)}$$

where a_i is the element in the action collection, n is the length of the action set. P_π is the probability of the decision strategy, which controls the probability of different strategies used when the current decision action is generated. $f(Q_t; \kappa)$ is the valuation-probability conversion function (QP conversion function) and κ is the function parameter.

The core of the dynamic exploration-exploitation balance decision policy lies in the meticulous modulation of the contributions of value-driven and stochastic exploration activities to the decision-making process through dynamic adjustment of the exploration rate. This strategy further quantifies the specific impacts of these two approaches on decision behavior, thus providing a probabilistic estimate for current decision actions. The formula is as follows:

$$P(a_t) = \frac{1}{n}\xi_t + f(Q_t; \kappa)(1 - \xi_t) \quad \text{(Equation 6)}$$

where $P(a_t)$ is the probabilistic estimate for the current decision-making action.

A key aspect of this policy is the creation of a flexible functional relationship that closely links the variation in value differences among options (ΔQ_t) with the dynamic adjustment of the exploration rate (ξ_t), as demonstrated in Equation 7. This method is deeply inspired by the learning processes of biological entities, particularly the phenomenon of the exploration rate decreasing over time. A variety of mathematical functions, such as the complementary sigmoid function, polynomial decay function, and exponential decay function, can be utilized to precisely depict this relationship. The diversity of function choices offers the flexibility to adjust the strategy as needed, thereby effectively simulating the exploration behavior in cognitive decision-making processes across different organisms.

$$\xi_t = f(\overline{\Delta Q_t}, \phi) \quad \text{(Equation 7)}$$

where $f(\overline{\Delta Q_t}; \phi)$ is the function of the change in the exploration rate, which is the ϕ parameter of the function, $\overline{\Delta Q_t}$ is the average of the valuation differences selected for each selection at the current moment, ξ_t is the exploration rate of the current moment,

Here, we use the complementary sigmoid function applied in the reward probability-based free-choice task as an example to illustrate how the exploration-exploitation dynamically adjusted decision policy works. The formula is as follows:

$$\xi_t = 1 - \frac{1}{1 + e^{-\omega(\overline{\Delta Q_t} - B)}} \quad \text{(Equation 8)}$$

where ω is the attenuation coefficient, and the higher the value, the faster the attenuation rate, B is the symmetrical center of the attenuation curve, and the left and right translations of the curve can be performed.

Another key aspect of this policy is the use of past experience to make decisions, which is mainly reflected in the model on the basis of the value function sequence calculated by the model, the establishment of QP conversion function. The QP conversion function

can be customized to further characterize the dynamic process of decision-making. The following describes how to create a QP conversion function $f(Q_t;\kappa)$:

According to the TD algorithm (SARSA algorithm is used as an example), we can establish the following iterative process based on the behavioral data of real biological decisions (More specifically, the value function for the action taken or state visited by the animal updates based on reward prediction error, while the value functions of all other actions remain unchanged[25] c:

The value valuation sequence of specific repetitive actions in each specific state can be quantified, and finally, the Q value sequence corresponding to the decision can be obtained $\{Q_n^{a_1}\}$, $\{Q_n^{a_2}\}\ldots\{Q_n^{a_n}\}$

$$\begin{cases} \{Q_n^{a_1}\} \Leftarrow \begin{cases} Q_0^{a_1} = Q_{a_1-init} \\ Q_n^{a_1} = Q_{n-1}^{a_1} + \partial(r_n + \gamma r_{n+1}) - Q_{n-1}^{a_1} \end{cases} \\ \{Q_n^{a_2}\} \Leftarrow \begin{cases} Q_0^{a_2} = Q_{a_2-init} \\ Q_n^{a_2} = Q_{n-1}^{a_2} + \partial(r_n + \gamma r_{n+1}) - Q_{n-1}^{a_2} \end{cases} \\ \cdots \\ \{Q_n^{a_n}\} \Leftarrow \begin{cases} Q_0^{a_n} = Q_{a_n-init} \\ Q_n^{a_n} = Q_{n-1}^{a_n} + \partial(r_n + \gamma r_{n+1}) - Q_{n-1}^{a_n} \end{cases} \end{cases} \quad \text{(Equation 9)}$$

where Q_{a-init} is the initial value of the corresponding decision action a, $\{Q_n^a\}$ is the Q value was updated for different decision sequences, n is the length of the decision-making sequence.

Write the Q value sequence in matrix form and denote it with Q:

$$\boldsymbol{Q} = \left[\{Q_n^{a_1}\}, \{Q_n^{a_2}\}\ldots\{Q_n^{a_n}\}\right] \quad \text{(Equation 10)}$$

where: $Q_{ij} \in \boldsymbol{Q}$, i is the time steps index and j is the decision type number.

We can achieve probability with a linear model:

$$P_{ij} = f(Q_{ij},k,b) = \frac{kQ_{ij}+b}{\sum_{j=1}^{n} kQ_{ij}+b} \quad \text{(Equation 11)}$$

where k is the linear scaling factor and b is the translation factor.

According to the correlation between the Q value and the p value of the experimental data, the values of k and b were determined, and the positive and negative maps of QP were realized by the positive and negative values of k. There is uncertainty and randomness in the representation of the display probability of the biological decision-making process, and sometimes the linear Decision function cannot describe the decision-making process well, so we need to consider the nonlinear model to realize the QP transformation.

Consider the nonlinear softmax function as an example:

$$P_{ij} = f(Q_{ij},\beta) = \frac{e^{\beta Q_{ij}}}{\sum_{j=1}^{n} e^{\beta Q_{ij}}} \quad \text{(Equation 12)}$$

where β is the sensitivity coefficient, which is used to control the sensitivity to the Q value during the decision process.

The softmax function uses an exponential function to convert the Q value, and there are similar nonlinear methods to this method, such as the conversion of the Q value through the sigmoid function, and the normalization algorithm is used to make the sum of the probabilities of each decision action. The specific formula is as follows:

$$\begin{cases} sigmoid(x) = \frac{1}{1+e^{-x}} \\ P_{ij} = f(Q_{ij},\omega) = \frac{sigmoid(\omega Q_{ij})}{\sum_{j=1}^{n} sigmoid(\omega Q_{ij})} \end{cases} \quad \text{(Equation 13)}$$

where ω is the sensitivity coefficient, which is used to control the sensitivity to the Q value during the decision process.

We provide several QP conversion functions and dynamic exploration rate functions for reference, encouraging researchers to develop customized decision policies tailored to specific decision tasks or experimental paradigms. These decision policies establish a mapping between decision values and probabilities, turning the previously unknown probabilities of decision transitions in the biological cognitive decision-making process into known quantities. By revealing these unknown transition probabilities, we facilitate a shift that aligns with the Markov Decision Process (MDP). Adhering to the MDP framework allows for the closed-loop conversion of

Action-Valuation-Probability-Action (A-Q-P-A), offering a solution for simulating the closed-loop transformation of decision sequences.

Simulation based on MCMC method and evaluation

Markov Chain Monte Carlo (MCMC) utilizes the concept of Markov chain to generate samples of the target distribution. The basic idea is to construct a Markov chain so that its steady-state distribution matches the desired target distribution. Once the chain reaches equilibrium, the sample it generates can be treated as a sample from the target distribution.

As mentioned above. The p value generated by the action valuation can be obtained through the decision policy, and the probability of decision transfer can be described as $P_{aa'}$:

$$P_{aa'} = P((s,a')|(s,a)) \quad \text{(Equation 14)}$$

We can generate simulated action sequences using decision transition probability chains derived from a model built on empirical data. It's also feasible to provide the initial iteration Q-value, and then convert this Q-value into probabilities through the decision policy, selecting actions based on these probabilities $P_{aa'}$. Subsequently, the Q-value is updated again via the value function. This iterative process not only produces a sequence of simulated actions but also generates sequences of values and probabilities for the decision-making process. A distinctive aspect of the Markov Decision Process (MDP) is its inherent randomness. Within an MDP framework, each decision can lead the agent to various states and result in different immediate rewards. This variability stems from both the current state and actions, and the unpredictability of the environment. By repeatedly applying the MDP decision-making process thousands of times via Markov Chain Monte Carlo (MCMC) methods, the actions selected by the agent in any given state achieve consistency over time. To capture the evolution of Q-values and probabilities (P-values) throughout the decision-making process, we recommend employing the mean series obtained from extensive samples of repeated decision sequences.

The observed coherence between the biological decision sequence and its simulated counterpart intuitively demonstrates the reliability and effectiveness of the Q-value and P-value sequences, which serve as crucial intermediaries linking these two sequences. Consequently, we introduce the following two indicators to assess the consistency between the empirical and simulated decision sequences.

Consistency of decision-making tendencies

The goal of decision propensity consistency is to utilize the current moment's decision propensity by encoding decision (action) categories with positive e_i^+ and negative e_i^- values, and then calculating their cumulative sums C_t to reflect tendencies of decision-makers in the decision-making process. The higher the value of C_t, the stronger the tendencies for positive choices, and the change in slope reflects the increase or decrease in propensity. By comparing the cumulative sums of decision behavior data collected from experiments C_t with the cumulative sums of mean sequences $\overline{C_t}$ from numerous simulated decision sequences obtained through the MCMC method, we can intuitively reflect the consistency of decision-making tendencies.

$$C_t = \sum_{i=1}^{t} e_i \quad \text{(Equation 15)}$$

Statistical consistency

Statistical consistency verification is conducted by segmenting both real and simulated decision sequences, which are then divided into multiple subsets through a fixed or variable-sized sliding window approach. Within each window, the local probability of a decision, corresponding to that window size, is estimated by statistically analyzing the frequency of occurrences. By continuously shifting the sliding window, we track the temporal evolution of the decision sequence's probability with the Savitzky-Golay Filter. To evaluate the statistical concordance of these sequences, we employ the 95% LoA. This measure represents the statistical bounds within which the probability trajectory of the simulated decision sequence occurs, encompassing 95% of the real decision sequence's probability trajectory. If the probability trajectory of the actual decision sequence falls within the 95% LoA, it is inferred that the simulated sequence is statistically consistent with the real experimental sequence.

$$LoA_{95\%} = \overline{d} \pm 1.96 \times SD_d \quad \text{(Equation 16)}$$

QUANTIFICATION AND STATISTICAL ANALYSIS

Quantitative data are presented as mean ± SEM or as violin plots (with the median, upper and lower quartiles, 1.5 × interquartile range, and the density distribution of the data). Statistical significance was assessed using paired t-tests for between-group comparisons or the Friedman test for within-group comparisons. All plots and statistical analyses were performed using Python software. Differences with p-values < 0.05 were considered statistically significant. The statistical methods and exact sample size (n) used in each quantified figure are reported in the figure legends (Figures 4C, 4D, 7C, and 7D). Detailed statistical information and data points are provided in Table S1.