IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 28, 2021, accepted September 30, 2021, date of publication October 4, 2021, date of current version October 15, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3117855

Evaluation of Instance-Based Learning and Q-Learning Algorithms in Dynamic Environments

ANMOL GUPTA[1], PARTHA PRATIM ROY[1], (Senior Member, IEEE), AND VARUN DUTT[2], (Senior Member, IEEE)

[1]PARIMAL, Indian Institute of Technology Roorkee, Haridwar, Roorkee 247667, India
[2]Applied Cognitive Science Laboratory, Indian Institute of Technology Mandi, Kamand, Himachal Pradesh 175005, India

Corresponding author: Anmol Gupta (agupta@cs.iitr.ac.in)

This work was supported in part by the Centre for Artificial Intelligence and Robotics, Defense Research and Development Organization (DRDO), under Grant IITM/DRDO/VD/324, and in part by the Life Sciences Research Board DRDO, India, under Grant IITM/DRDO-LSRB/VD/301.

ABSTRACT Reinforcement learning is an unsupervised learning algorithm, where learning is based upon feedback from the environment. Prior research has proposed cognitive (e.g., Instance-based Learning or IBL) and statistical (Q-learning) reinforcement learning algorithms. However, an evaluation of these algorithms in a single dynamic environment has not been explored. In this paper, a comparison between the statistical Q-learning algorithm and the cognitive IBL algorithm is presented. A well-known environment, ''Frozen Lake,'' is used to train, generalize, and scale Q-learning and IBL algorithms. For generalizing, the Q-learning and IBL agents were trained on one version of the Frozen Lake and tested on a permuted version of the same environment. For scaling, the two algorithms were tested on a larger version of the Frozen Lake environment. Results revealed that the IBL algorithm used less training time and generalized better to different environment variants. The IBL algorithm was also able to show scalability by retaining its superior performance in the larger environment. These results indicate that the IBL algorithm could be proposed as an alternative to the standard reinforcement learning algorithms based on dynamic programming such as Q-learning. The inclusion of human factors (such as memory) in the IBL algorithm makes it suitable for robust learning in complex and dynamic environments.

INDEX TERMS Reinforcement learning, Q-learning, instance-based learning, openAI, cognitive modeling, frozen lake, dynamic environment.

I. INTRODUCTION

Learning is acquiring new knowledge, behaviors, skills, preferences, or modifying existing ones [1]. Children learn by seeking environmental stimulation, which develops objects, space, time, and causality [2]. Adults, on the other hand, use their prior knowledge while interacting with the environment. For instance, an adult and a child are likely to respond differently while engaging with an environment that contains fire. The adult may choose not to interact with the fire due to her prior experiences. Whereas the child, who has not interacted with fire before, may decide otherwise. Only after this interaction, the child is likely to infer that fire is dangerous. The above trial-and-error learning process, termed reinforcement learning, may help adults and children learn over time [3]. Learning via reinforcements (i.e., rewards or punishments) in the real world is not only common to humans, but it is an essential skill to possess even for robots or agents that are built to work in real or virtual environments [4]–[6]. In the absence of real environments, virtual environments (also called microworlds or games) allow researchers to compress time and space and to create different kinds of dynamic complexities for intelligent algorithms [7]–[10]. Thus, these environments are ideal platforms to investigate the potential of reinforcement learning algorithms.

A popular example of a virtual environment is Frozen Lake [11], a 4 × 4 (16 tiles) grid world. An agent must begin at the start tile in Frozen Lake and reach the goal tile [11], [12]. Some tiles are ''frozen,'' and the agent can walk over them; other tiles have ''holes.'' If the agent steps on a hole tile, it falls and dies. Furthermore, there is inherent randomness in Frozen Lake. The agent can even slip on a frozen tile with some probability. Thus, the agent's actual movement direction may differ from the agent's intended movement

The associate editor coordinating the review of this manuscript and approving it for publication was Chao Shen.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

direction due to slipping on the frozen tiles [11], [12]. For example, the agent may move left even when it chooses to move right. Frozen Lake has a default reward system associated with it: The agent gets a reward of +1 when it reaches the goal tile, and, at all other times, there is a 0 reward. Overall, with its dynamic features, Frozen Lake provides an ideal environment for evaluating reinforcement learning algorithms.

There are at least two learning paradigms, namely, supervised and unsupervised [14], through which the agent can learn the mechanics of an environment like Frozen Lake [15]. In supervised learning, an agent learns to navigate the environment by comparing its decisions against the optimal decisions (resulting in maximum reward) [14]. However, this supervised approach may not work when the calculation of optimal solution is complex or when the optimal solution may change between the training and test conditions. In such a case, the agent may rely upon unsupervised learning to learn the environment's mechanics. The agent does not know optimal decisions in unsupervised learning and instead learns from different outcomes [14]. Reinforcement learning (RL) is an example of unsupervised learning algorithms, where agents/people learn via making decisions and getting feedback on their decisions [3], [16]–[18].

Prior research has proposed several cognitive RL algorithms in single-player and multi-player canonical games with known optimal solutions [19]–[21]. These cognitive RL algorithms possess free parameters that are motivated by human cognition. For example, Erev and Roth [20] investigated how specific cognitive RL algorithms could predict human data in simple single-player and multi-player games with unique and mixed strategy equilibria (some of these algorithms possessed recency and exploration parameters). According to Erev and Roth [20], the cognitive RL algorithm fitted human decisions better than the optimal predictions, even with a single parameter. Similarly, Beggs [19] further examined the convergence of payoffs and strategies in certain cognitive RL algorithms given in [20] and [22]. Results revealed that when all players used the convergence rule, it iteratively eliminated the dominant strategies. Laslier *et al.* [21] evaluated a cumulative proportional reinforcement (CPR) rule in the same context. An agent played an action with a probability proportional to the cumulative utility of the action. The performance of the CPR rule was compared with other reinforcement rules and replicator dynamics. Dutt [23] proposed a cognitive RL algorithm for capturing human decisions in a dynamic decision-making task involving stock control. The proposed RL algorithm possessed cognitive parameters such as attention and time for making corrections to the stock.

Some researchers have compared the performance of several statistical RL algorithms [24]–[27]. For example, Singal *et al.* [25] compared several statistical RL algorithms to capture human performance in a platform jumper game. These authors compared imitation learning [28], Q-learning [18], and DeepQ Learning [29] in their ability to account for human decisions. Results revealed that imitation learning and Q-learning performed similarly to human decisions in a slow version of the game; however, the DeepQ algorithm performed similarly to human decisions in the game's fast version. Mnih *et al.* [29] compared the DeepQ algorithm's performance with professional human game testers on the classic 49 Atari 2600 games [30]. They showed that the algorithm received only the pixel information and game scores as input and achieved a level comparable to human testers.

Although several cognitive and statistical RL algorithms have been proposed, a comparison of these algorithms on a single dynamic task is lacking in the literature. The primary objective of this research is to overcome this literature gap. Specifically, a statistical RL algorithm (Q-learning [18]) is compared with a cognitive RL algorithm (Instance-based learning (IBL) [14], [15]) across several variations in the Frozen Lake environment. The proposed comparison allows researchers working in cognitive science and artificial intelligence areas to appropriately choose the approach that produces faster learning in lesser time.

Q-learning is a statistical RL algorithm proposed by Watkins and Dayan [18]. In Q-learning, there are states, actions, and rewards. Although the Q-learning approach is attractive and popular, it is statistical, and it works based on exploring new actions and exploiting the tried actions in different states [18].

Beyond Q-learning, the cognitive IBL algorithm has been proposed in the literature [14], [15]. In IBL, an agent may decide on a situation and then experience an outcome (reward/punishment) as the feedback [17]. These experiences are stored in memory in the form of instances, which have three parts: situation (S; a set of attributes describing the environment), decision (D; responses to the situation by the agent), and utility (U; how good the decision was, as determined by the feedback from the task) [14], [15]. When a decision-maker finds itself in a situation (a set of cues), the most activated and similar instances to this situation are matched and retrieved from the memory. Prior research has built many IBL models for several environmental tasks, including the water purification plant, multi-arm bandit problems, cybersecurity, and 2×2 games [31]–[37].

This work's novelty is a comprehensive comparison of the statistical Q-learning and the cognitive IBL RL algorithms across three different experiments. In the first experiment, the Q-learning and IBL algorithms were compared using their default parameters in different outcome conditions in the Frozen Lake environment. In the second experiment, the algorithms' parameters were re-calibrated, and the algorithms were again compared to different outcome conditions in the Frozen Lake environment. In the third experiment, the original Frozen Lake was scaled to a larger environment, and the calibrated IBL and Q-learning algorithms were compared in the larger environment. Another novel contribution of the work is the definition of a new testing framework for the RL algorithms. The algorithms were trained in the

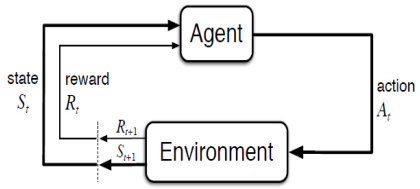


FIGURE 1. The agent-environment interaction in the Q-learning algorithm. Source: [38].

default environment. However, the environment was changed for testing by randomly permuting it while maintaining the same difficulty level. Also, the environment was scaled up to an 8×8 (64 tiles) grid world, and different reward functions were varied to test the algorithms' generalization.

Furthermore, two new metrics, success ratio and optimal actions, which captured how and what the algorithms were learning, were developed and evaluated for both IBL and Q-learning algorithms. Overall, these measures allowed us to compare the convergence speed of the two algorithms. Specifically, the calculation of optimal actions allowed us to test to what extent the two algorithms were following the optimal paths in the environment.

It is believed that this combination of different training and testing environments and the custom-made matrices (to assess whether the algorithms were following the optimal paths) provided superior ways of quantifying algorithms' generalization capabilities. These features also forced the algorithms to learn the underlying subtleties in the task and facilitated learning from one environment to another.

In what follows, first, the mechanics of the Q-learning and IBL algorithms are explained. Second, the methodology of different Frozen Lake environments used across the three experiments is provided. Next, the results across the three experiments are discussed. Finally, the implications of developing the Q-learning and IBL algorithms in modeling agents in synthetic environments are discussed.

II. REINFORCEMENT LEARNING ALGORITHMS

In this section, the Q-learning and IBL algorithms are discussed. These algorithms are later evaluated on the Frozen Lake environment (more details ahead in the manuscript).

A. Q-LEARNING

Q-learning is a form of reinforcement learning that uses Q-values (also called action values) to iteratively improve a learning agent's decision actions across different states in the environment [18]. The environment is modeled with a finite Markov Decision Process [38], where the environment has states that change due to the agent's action. Fig. 1 shows the working of the Q-learning algorithm. As shown in Fig. 1, at time step t, the agent receives the environment's state s_t and, based on s_t, the agent selects an action a_t. In the next time step, as a result of the action, the agent receives a numerical reward of r_{t+1} and finds itself in a new state, s_{t+1}. A Q-value is associated with each state and action pair (the starting

TABLE 1. Q-learning for frozen lake.

Line No.	Activity
1	For 1 to n agents, do
2	Define a new agent 'a'
3	Set parameters for a: learning rate, gamma, and epsilon
4	Initialize Q Table of n states x actions
5	For episode 1 to n do
6	While t < max steps do
7	Reset the environment
8	If random(0,1) < epsilon then
9	Action= sample action from train env
10	Else
11	Action = argmax(Q[state,:])
12	End if
13	Take the chosen action and collect the reward
14	Calculate the new Q-value
15	If a new state is a goal state, then
16	Increase the number of successful episodes
17	Break
18	If a new state is a hole state, then
19	Break
20	End while
21	Check if action is optimal
22	Update success ratio and optimal actions
23	End for
24	End for

Q-value may be zero or randomly assigned). A change in the Q-value is governed by the Bellman equation [39], and it is calculated by the following:

$$Q_{t+1}(s_t, a_t) = (1-\alpha)\, Q_{t(s_t,a_t)} + \alpha\,(r_t + \gamma max\,[Q(s_{t+1}, a_t)]) \quad (1)$$

where, $Q_{t+1}(s_t, a_t)$ and $Q_t(s_t, a_t)$ are newer and older Q-values corresponding to the state s_t and action a_t. The r_{t+1} is the reward obtained by moving from the state s_t to the state s_{t+1} by taking action a_t. The α is the learning rate that controls how much of the difference between older and newer Q-values is considered to update the Q-values. The γ is the discount factor that controls the discounting of the Q-values for future states.

Table 1 presents the Q-learning algorithm in the context of the Frozen Lake environment. In lines 1-4 (Table 1), the agent is initialized, the algorithm's parameters are set, and the Q-table is initialized (for each state and action pair). The Q-table is used for selecting the actions of the agent in a state. Line 5 defines the loop to make the agent play the game for n episodes. In an episode, there are three stopping conditions for the agent: The agent falls into a hole, dies, and the game stops; the agent cannot reach the goal after a maximum number of steps (indicated by max steps); or, the agent successfully reaches the goal. These stopping criteria are checked in lines 6 and 15-19. In line 7, the environment is reset so that each episode starts with a new environment. In lines 8-12, the agent selects an action based on exploration or exploitation. Once the action is decided, the agent takes action and collects the resulting reward in line 13. With the received reward, the state and action pair's Q-value is updated

TABLE 2. Hyper-parameters and their default values for Q-learning algorithm.

Hyperparameter	Default Value
Learning Rate	0.90
Gamma	0.81
Epsilon	0.70

in line 14. Line 22 updates the performance metrics, which are discussed ahead in this article.

In Q-learning, the learning rate, discount factor (gamma), and epsilon are hyper-parameters. These parameters are set empirically to improve the algorithm's performance. The learning rate is defined in the range [0, 1], where 0 means that the Q-values are never updated, and nothing is learned. In contrast, a high learning rate value (e.g., equal to 0.9) means that agent's learning of the environment occurs quickly. The discount factor γ, also in the range [0, 1], models the discounting of future rewards compared to the immediate rewards. Mathematically, the discount factor needs to be set to less than 1 for the algorithm to converge [39]. Finally, epsilon works as a threshold for the algorithm to randomly choose the action or maximize Q-value. The epsilon parameter is responsible for the exploration-exploitation tradeoff in Q-learning [39]. The default values for these hyper-parameters are mentioned in Table 2.

During training, the default parameters are set, and an agent starts interacting with the environment. The agent decides to take a particular action that maximizes the Q-value in the state. After the agent takes action, the state of the environment changes, and the environment responds with a reward. Based on the reward, the agent updates the Q-value of the state and the associated action. An episode is defined as a sequence of states, actions, and rewards resulting in the agent reaching the goal tile or dying. The Q-learning algorithm updates the parameters during the episode and carries forward the learned or best parameters from one episode to another. Hence, an agent is trained over multiple episodes, and it learns the subtleties of the environment.

B. INSTANCE-BASED LEARNING

Instance-based Learning (IBL) proposes that humans make dynamic decisions by accumulating and refining instances. These instances comprise situation, action, and utility [31]. When an IBL agent interacts with the dynamic environment, it recognizes the present situation by comparing it to the past instances in its memory. In the absence of similar instances, the IBL agent uses a heuristic rule. However, with the accumulation of instances, the agent makes decisions based on retrieving the most similar and activated instances from memory. These instances are then blended (one per decision action), and the action that has the highest blended value is the one that gets executed. The IBL algorithm uses the formulations of activation, probability of retrieval, and blending from the ACT-R architecture to facilitate the encoding,

TABLE 3. IBL in frozen lake.

Line No.	Activity
1	For 1 to n agents, do
2	Define a new Agent a
3	Set parameters for a: cognitive noise, decay, and default utility
4	Define *situations* for IBL
5	For episode 1 to n do
6	While t < max steps do
7	Reset the environment
8	Choose an action from the SDU triplets
9	Take the chosen action and collect the reward
10	Update the utility for the situation, decision pair
11	If the new situation is a goal, then
12	Increase the number of successful episodes
13	Break
14	If a new state is a hole, then
15	Break
16	End while
17	Check if action is optimal
18	Update success ratio and optimal actions
19	End for
20	End for

storage, and retrieval processes [40]. ACT-R is a cognitive modeling architecture built on principles of human cognition. It is supported by many empirical studies in memory, learning, and problem-solving [40]. In every situation, the IBL agent executes an action with the highest blended value. The blended value V_j is updated as per the following equation:

$$V_j = \sum_{i=1}^{n} p_i x_i \tag{2}$$

where, x_i is the observed outcome and p_i is the retrieval probability of the instance i containing the outcome. The p_i is given by the following equation:

$$p_{i,t} = \frac{e^{A_{i,t}/\tau}}{\sum_j e^{A_{j,t}/\tau}} \tag{3}$$

where, τ is the random noise and $A_{i,t}$ is the activation of the instance i. $A_{i,t}$ is given by:

$$A_{i,t} = \sigma \ln\left(\frac{1-\gamma_{i,t}}{\gamma_{i,t}}\right) + \ln \sum_{t_p \in 1,\ldots t-1} \left(t - t_p\right)^{-d} \tag{4}$$

where, d is the decay parameter, which determines the memory decay. The σ is the cognitive noise parameter, which determines the agent-to-agent variability in activations. The γ is a random draw from a uniform probability distribution, which controls the cognitive noise, and t_p is each previous trial index in which the instance i was observed. The IBL algorithm was modified to work with Frozen Lake, and the resulting pseudo-code of the algorithm is given in Table 3.

In lines 1-4 (Table 3), the agent is initialized, the model parameters are set, and the situation is initialized (for each state and action pair) in IBL. Line 5 defines the loop to make the agent play the game for n episodes. During an episode, there are three stopping conditions for the agent: The agent falls into a hole, dies, and the game stops; the agent does not reach the goal state after a maximum number of steps

TABLE 4. Hyper-parameters and their values for the IBL algorithm.

Hyperparameter	Default Value
Cognitive Noise	0.25
Decay	0.50
Default Utility	10,000

indicated by max steps; or, the agent successfully reaches the goal state. These conditions are checked on lines 6 and 11-15. In line 7, the environment is reset so that each episode starts with a new environment. In lines 8-9, the agent selects an action based upon the blended value and collects the reward. With the received feedback, the utility values of the situation and decision pair are updated on line 10. Lines 17-18 update the performance metrics, which are discussed later in this article.

In IBL, there are three hyper-parameters, i.e., cognitive noise, decay, and default utility [31]. The default utility provides the goodness of action in the absence of an outcome experienced by the agent. The decay hyper-parameter controls the rate at which activation for the previously experienced instances decays with time. Cognitive noise is added to capture the variability in decisions from one agent to the other in the same situation. The default values of the hyperparameters are shown in Table 4.

During the training, an IBL agent interacts with the environment and stores instances in its memory from this interaction. The agent decides for the action that has the highest blended value among all actions. After the agent takes action, the agent updates the outcome in the instance corresponding to the action. The outcome is based upon the reward received by the agent as feedback in the task.

C. INSTANCE-BASED LEARNING VERSUS Q-LEARNING

IBL, being a cognitive algorithm, has a decision-making process that is similar to how people make decisions. The learning in IBL is distinct from Q-learning, and it is much more than simply accumulating a single value over episodes (like the Q-value in Q-learning). Learning in IBL is a compilation of learning mechanisms proposed by Langley and Simon [41] in 1981. It not only comprises of accumulation of instances (situation-decision-utility or SDU triplets), but it consists of five distinct mechanisms:

- *Instance-based knowledge:* The accumulation of knowledge in the form of instances containing the SDU.
- *Recognition-based retrieval:* Memory retrieval of SDUs according to the similarity between the evaluated situation and instances stored in memory.
- *Adaptive strategies:* Adaptation from heuristic-based to instance-based decisions according to interactive practice in the dynamic task.
- *Necessity:* Method to control the continuation of the alternative search.
- *Feedback update:* Method to update the utility of SDUs and maintain the causal attribution of results to actions.

These IBL mechanisms are different from the mechanisms in the Q-learning algorithm. For example, there is no recognition-based retrieval in Q-learning because there is no concept of memory. However, memory is an integral part of learning in IBL. Another difference between the mechanisms of IBL and Q-learning is the presence of feedback updates in IBL and its absence in Q-learning. On receiving the reward r_t, IBL updates the utility of all instances in memory that led to the decision action. In Q-learning, one only updates the Q-value of the current state and action pair. Similarly, there is no concept of reward discounting in IBL. However, in Q-learning, the temporal discounting of rewards is implemented through a linear decline of r_t with time.

Next, different experiments to compare IBL and Q-learning are discussed.

III. EXPERIMENTS

Three separate experiments were performed to compare IBL and Q-learning algorithms based on certain performance metrics. In each experiment, the performance and generalization capabilities of the IBL and Q-learning algorithms were compared. First, both the algorithms were trained on the default Frozen Lake environment with the default reward function (+1 for reaching the goal state and 0 otherwise) in all three experiments. After training, the algorithms were tested in a permuted environment with different reward functions. Along with the permuted environment and new reward functions, new metrics were made to judge the algorithms' performance. It was hypothesized that the training and testing steps would help compare the performance of the Q-learning and IBL algorithms.

A. EXPERIMENT 1: IBL VERSUS Q-LEARNING WITH DEFAULT PARAMETERS IN THE 4 × 4 FROZEN LAKE ENVIRONMENT

The experiment's objective was to compare the performance of IBL and Q-learning algorithms on the default and permuted versions of a 4 × 4 Frozen Lake environment.

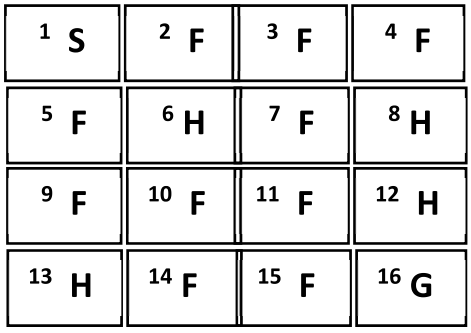


FIGURE 2. Default 4 × 4 Frozen Lake with S: Starting Cell, F: Frozen Surface Cell, H: Hole Cell, and G: Goal Cell.

1) METHODS

a: DEFAULT 4 × 4 FROZEN LAKE ENVIRONMENT

First, the default Frozen Lake was considered the Q and IBL agents' training environment (see Fig. 2). The default

environment consisted of a specific combination of frozen and hole cells. The frozen cells represented a solid surface, where the agent could walk safely without dying. The hole represented the lake area, where the agent would fall into the lake and die. The agent started at the first (start) cell (marked as S) and had to reach the last (goal) cell (marked as G). Each algorithm aimed to make the agent reach the G cell starting from the S cell.

b: PERMUTED 4 × 4 FROZEN LAKE ENVIRONMENT

The permuted 4 × 4 Frozen Lake environment was generated by swapping the second and third row of the default environment (see Fig. 3). This swap ensured that the difficulty levels of the default and permuted environment were similar. The difficulty level of the environment could be judged by the number of holes and the optimal paths leading to the Goal cell. The suggested permutation kept the number of holes and the optimal paths, the same for both environments. Again, each algorithm aimed to make the agent reach the goal cell from the start cell. The permuted environment was used to test the agents' learned ability (via training on the default 4 × 4 Frozen Lake).

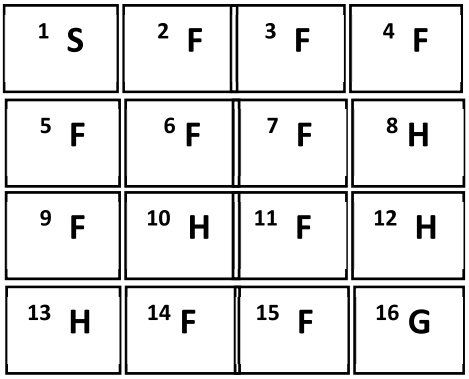


FIGURE 3. Permuted 4 × 4 Frozen Lake with S: Starting Cell, F: Frozen Surface Cell, H: Hole Cell, and G: Goal Cell.

c: REWARD FUNCTIONS

Table 5 shows the different reward functions investigated in this research. The first reward function (referred to as default rewards) gave the agent feedback of +1 on reaching the goal and feedback of 0 otherwise. The second function (referred to as negative rewards) punished the agent with feedback of −1 when it fell in a hole and rewarded it with feedback of +1 when it reached the goal. The third function (referred to as negative-and-positive rewards) punished the agent with a reward of −1 for falling in a hole; however, it also rewarded the agent with a small positive reward (+0.01/cell) when the agent walked on the frozen cells (except the goal cell). Like the previous two reward functions, the agent got a reward of +1 when it reached the goal. In the default Frozen Lake environment, the reward function used was default rewards. However, in the permuted Frozen Lake environment, the reward functions used were negative rewards and negative-and-positive rewards.

TABLE 5. Different reward functions to test the robustness of the Q-learning and IBL algorithms.

Reward Functions	Reaching the Goal cell	Falling in the hole	Reaching a Frozen cell (except Goal and Hole)
1. default rewards	+1	0	0
2. negative rewards	+1	-1	0
3. negative-and-positive rewards	+1	-1	+0.01

d: PERFORMANCE MEASURES

SUCCESS RATIO AND EPISODE BINS

Success ratio was defined as the number of times an agent successfully reached the goal cell divided by the number of times the agent tried to reach the goal cell. For example, if the agent succeeded in reaching the goal cell ten times out of a total of 50 tries, then the success ratio was 0.2 (=10/50). An episode in the Frozen Lake environment was defined as a successful movement of an agent from the S cell to the G cell or the death of an agent mid-way while moving from the S cell to the G cell. The success ratio was averaged for each set of 50 contiguous episodes into an episode bin to visualize the results. For instance, if an agent in an algorithm attempted 1500 episodes, then the number of episode bins was 30 (=1500 episodes/50 episodes per bin).

OPTIMAL PATHS AND ACTIONS

An optimal path was defined as a path that started in the S cell and helped the agent move towards the G cell in the shortest possible manner. Only the forward movements toward the G cell were considered in the optimal path. The number of optimal paths was kept the same across both the default (training) and the permuted (test) environments for keeping the comparisons fair. As shown in Fig. 4a and 4b, there were three optimal paths shown by arrows in both environments.

For example, in the default 4 × 4 Frozen Lake environment given in Fig. 4a, if the agent was in cell 3 and it took a down action and reached cell 7, this move was considered an optimal action. Similarly, the agent could take two optimal actions, right and down, from cell 1. The percentage of times the agent took an optimal action from each cell was calculated to evaluate the performance. The percentage of optimal actions in a Frozen Lake cell was defined as the number of times the agent chose an optimal action in the cell divided by the agent's total actions from that cell. The number of optimal actions taken in a cell is a measure that may explain the difference in the performance of the two algorithms in terms of success ratio.

e: MODEL PARAMETERS AND MODEL EXECUTION

In the first experiment, IBL was compared with Q-learning by keeping the hyper-parameters in both algorithms to their

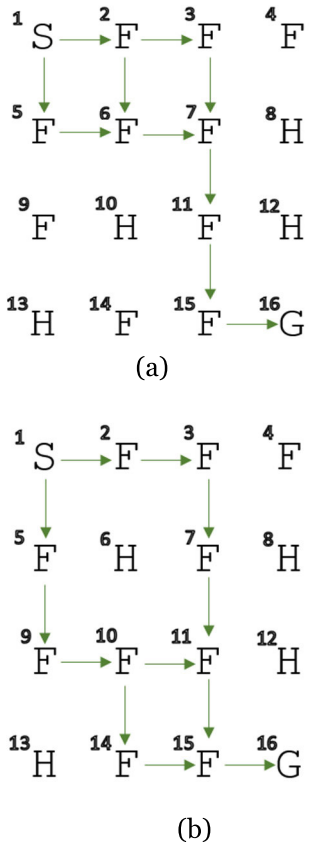


FIGURE 4. Optimal paths in (a) the default 4 × 4 Frozen Lake environment and (b) the permuted 4 × 4 Frozen Lake environment.

default values. Table 1 and Table 2 detail the default values of parameters in each algorithm.

In IBL, the default value of decay meant weak recency, and the default value of cognitive noise referred to a small amount of noise to capture variability. The default value of utility referred to a reasonable value to drive exploration in the IBL model. In Q-learning, the default value of the learning rate meant quicker learning. The default value of gamma meant moderate discounting of future rewards. Finally, the default value of epsilon meant moderate exploration.

Both IBL and Q-learning algorithms were run for 50 agents each. These number of agents accounted for the agent-to-agent variability in performance. Each agent ran for 1500 episodes on the default 4 × 4 Frozen Lake environment (see Fig. 2). The agents' memory (in IBL) or the Q-matrix (in Q-learning) were not reset between episodes. After completing these 1500 episodes, the algorithm's generalizability was tested by running the same agent for 1500 episodes on the permuted 4 × 4 Frozen Lake environment (see Fig. 3). Again, the memory (in IBL) or the Q-matrix (in Q-learning) were not reset during the switch from the default environment to the permuted environment. Overall, 50 agents were first trained on the default environment and then tested on a permuted environment. This training and test framework helped investigate how much learning was transferred from the default 4 × 4 Frozen Lake environment to the permuted 4 × 4 Frozen Lake environment [42].

f: DATA ANALYSES

One-way ANOVAs were performed to evaluate the main effect of the models (IBL and Q-learning) on the success ratios, keeping the alpha level at 0.05 and the power level at 0.80. Also, mixed factorial ANOVAs were performed to evaluate the interaction effects of the success ratio variations with the episodes for both the models (IBL or Q-learning). For mixed factorial ANOVAs, the alpha level was kept at 0.05 and the power levels at 0.80. Based on the Q-Q plots (between expected quantiles and normal quantiles) and the K-S test, the dependent variable (success ratio) was found to be normally distributed. Mauchly's test of sphericity was also reported.

2) RESULTS

Results are presented as the average success ratio of IBL, and Q-learning algorithms averaged over all episodes and agents for the training and testing environments. The default 4 × 4 Frozen Lake environment (see Fig. 2) and the default reward setting (see Table 5) were used for training. For testing, the permuted 4 × 4 Frozen Lake environment (see Fig. 3) was used in different reward settings, namely default rewards, negative rewards, and negative-and-positive rewards (see Table 5). Besides the average success ratio, the success ratio (averaged over 50 agents) in each episode bin was also analyzed. Lastly, the percentage of optimal actions took by the agent in each state of the training and testing environments was analyzed.

a: TRAINING IBL AND Q-LEARNING ON THE DEFAULT 4 × 4 FROZEN LAKE ENVIRONMENT WITH DEFAULT REWARDS

Fig. 5(a) shows the average success ratio of IBL and Q-learning algorithms averaged over all episodes and participants in the default 4 × 4 Frozen Lake environment with default rewards. As seen in the figure, IBL achieved a better average success ratio compared to Q-learning (IBL: 0.023 > Q-learning: 0.015; F (1, 97) = 49.289, $p < 0.001$, $\eta^2 = 0.337$). Fig. 6(a) shows the success ratio of IBL and Q-learning algorithms in each episode bin averaged over all participants. There was a significant interaction between the algorithms and episode bins (F (1, 253.84) = 2.663, $p < 0.001$, $\eta^2 = 0.027$). As shown in Fig. 6(a), although the success ratio increased rapidly in the first 3-episodes in both IBL and Q-learning algorithms, this increase and subsequent stabilization were much higher in the IBL algorithm than the Q-learning algorithm.

b: TESTING OF IBL AND Q-LEARNING ON THE PERMUTED 4 × 4 FROZEN LAKE ENVIRONMENT WITH DEFAULT REWARDS

Fig. 5(b) shows the success ratio of IBL and Q-learning algorithms averaged over all episodes and participants in the permuted 4 × 4 Frozen Lake environment with default rewards. As seen in the figure, IBL achieved a better average success ratio compared to Q-learning (IBL: 0.024 > Q-learning:

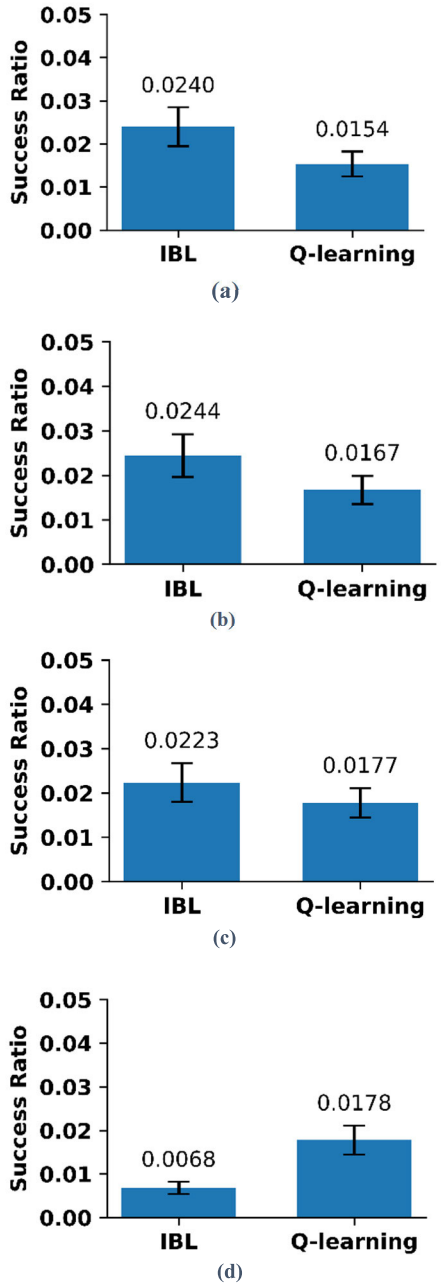


FIGURE 5. Average success ratio of IBL and Q-learning during (a) training on the default 4 × 4 Frozen Lake environment with default rewards; (b) testing on the permuted 4 × 4 environment with default rewards; (c) testing on the permuted 4 × 4 environment with negative rewards; and, (d) testing on the permuted 4 × 4 environment with negative-and-positive rewards. The error bars show 95% CI around the point estimate.

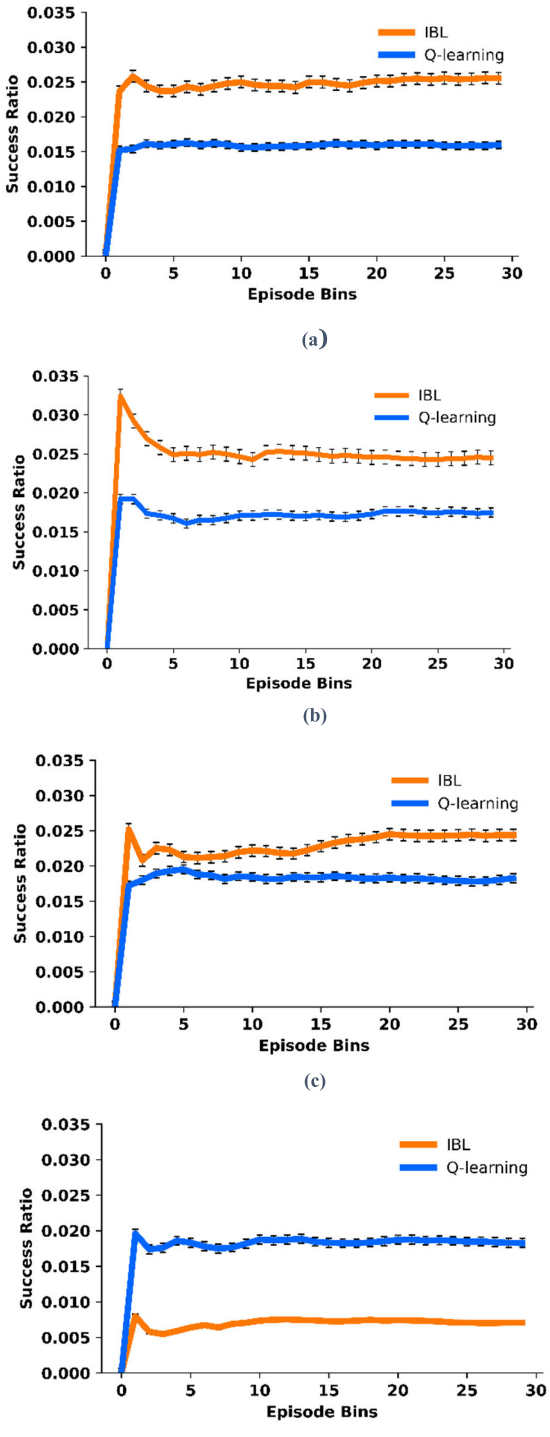


FIGURE 6. Average success ratio in each episode bin across both IBL and Q-learning during (a) training on the default 4 × 4 Frozen Lake environment with default rewards; (b) testing on the permuted 4 × 4 environment with default rewards; (c) testing on the permuted 4 × 4 environment with negative rewards; and, (d) testing on the permuted 4 × 4 environment with negative-and-positive rewards. The error bars show 95% CI around the point estimate.

0.016; $F\ (1, 97) = 43.61$, $p < 0.001$, $\eta^2 = 0.310$. Fig. 6(b) shows the success ratio of IBL and Q-learning algorithms in each episode bin averaged over all participants. There was a significant interaction between the algorithms and episode bins ($F\ (1, 264.53) = 3.62$, $p < 0.001$, $\eta^2 = 0.036$). The changes in success ratios of IBL and Q-learning algorithms over episode bins followed the same pattern (a subsequent stabilization followed the initial rapid increase) as shown in the default 4 × 4 Frozen Lake environment with default rewards (see Fig. 6(b)). The stabilization was much higher in the IBL algorithm compared to the Q-learning algorithm.

c: TESTING OF IBL AND Q-LEARNING ON THE PERMUTED 4 × 4 FROZEN LAKE ENVIRONMENT WITH NEGATIVE REWARDS

Fig. 5(c) shows the average success ratio of IBL and Q-learning algorithms averaged over all episodes and participants for the permuted 4 × 4 Frozen Lake environment with negative rewards. As seen in the figure, IBL achieved a better average success ratio compared to Q-learning (IBL: 0.022 > Q-learning: 0.017; $F\ (1,97) = 75.6, p < 0.001, \eta^2 = 0.438$). Fig. 6(c) shows the success ratio of IBL and Q-learning algorithms in each episode bin averaged over all participants. There was a significant interaction between the algorithms and episode bins ($F\ (1,\ 255.64) = 2.40,\ p < 0.001,\ \eta^2 = 0.024$). As seen in Fig. 6(c), the success ratio of the IBL algorithm was higher than that of the Q-learning algorithm, and it took some number of episode bins to stabilize the success ratio at a value. However, at no point in time, the IBL algorithm lagged behind the Q-learning algorithm.

d: TESTING OF IBL AND Q-LEARNING ON THE PERMUTED 4 × 4 FROZEN LAKE ENVIRONMENT WITH NEGATIVE-AND-POSITIVE REWARDS

Fig. 5(d) shows the average success ratio of IBL and Q-learning algorithms averaged over all episodes and participants for the permuted 4 × 4 Frozen Lake environment with negative-and-positive rewards. As seen in the figure, the Q-learning algorithm achieved a better average success ratio compared to IBL (Q-learning: 0.017 > IBL: 0.006; $F\ (1,\ 97) = 68.8,\ p < 0.001,\ \eta^2 = 0.415$). Fig. 6(d) shows the success ratio of IBL and Q-learning algorithms in each episode bin averaged over all participants. There was a significant interaction between the algorithms and episode bins ($F\ (1,\ 265.54) = 2.687,\ p < 0.001,\ \eta^2 = 0.027$). Unlike the other reward functions, Q-learning performed better than IBL. As shown in Fig. 6(d), although the success ratio increased rapidly in the first 3-episodes in both algorithms, this increase and subsequent stabilization were much higher in Q-learning than in IBL.

e: OPTIMAL ACTIONS BY IBL AND Q-LEARNING

Next, the percentage of optimal actions taken by both the IBL and Q-learning algorithms during training (in the default environment) and testing (in the permuted environments) was investigated. Fig. 7(a) shows the percentage of optimal actions taken from each cell of the default 4 × 4 Frozen Lake environment with default rewards. IBL achieved a higher percentage of optimal actions than Q-learning in 100% of the cells (i.e., ten out of the ten cells). Fig. 7(b) shows the percentage of optimal actions taken from each cell of the permuted 4 × 4 Frozen Lake environment with default rewards. IBL achieved a higher percentage of optimal actions than the Q-learning algorithm in 100% of the cells (i.e., 8 out of the 8 cells). Similarly, Fig. 7(c) shows the percentage of optimal actions taken from each cell of the permuted 4 × 4 Frozen Lake environment with negative rewards. IBL again achieved a higher percentage of optimal actions than Q-learning in 100% of the cells (i.e., 8 out of the 8 cells).

S IBL: 67, QL: 55	F IBL: 17, QL: 5	F IBL: 10, QL: 7	F
F IBL: 17, QL: 16	F IBL: 10, QL: 7	F IBL: 3.9, QL: 3.1	H
F	H	F IBL: 3.1, QL: 2.8	H
H	F	F IBL: 1.8, QL: 0.9	G

(a)

S	F IBL: 42, QL: 38	F IBL: 15, QL: 11	F
F IBL: 24, QL: 20	F IBL: 19, QL: 18	F IBL: 13, QL: 10	H
F	H	F IBL: 5, QL: 3	H
H	F	F IBL: 1.8, QL: 1.0	G

(b)

S IBL: 11, QL: 55	F IBL: 4, QL: 39	F IBL: 1.6, QL: 12	F
F IBL: 1.9, QL: 20	F IBL: 1.8, QL: 16	F IBL: 3.7, QL: 10	H
F	H	F IBL: 1.4, QL: 3.6	H
H	F	F IBL: 0.5, QL: 1	G

(c)

S IBL: 73, QL: 59	F IBL: 20, QL: 7	F IBL: 12, QL: 7	F
F IBL: 19, QL: 15	H	F IBL: 9, QL: 6	H
F IBL: 13, QL: 10	F IBL: 10, QL: 6	F IBL: 6, QL: 3.2	H
H	F IBL: 5.8, QL: 3	F IBL: 3.5, QL: 2	G

(d)

FIGURE 7. Optimal action percentages by IBL and Q-learning algorithms during (a) training with default rewards; (b) testing with default rewards; (c) testing with negative rewards; and (d) testing with negative-and-positive rewards. Each cell represents the percentage of optimal actions taken by IBL, followed by Q-learning.

Finally, Fig. 7(d) shows the percentage of optimal actions taken from each cell of the permuted 4 × 4 Frozen Lake environment with negative-and-positive rewards. In this environment, the Q-learning algorithm achieved a higher percentage of optimal actions than the IBL algorithm in 100% of the cells (i.e., 8 cells out the 8 cells).

3) DISCUSSION

In the first experiment, the IBL and Q-learning algorithms were trained on the default Frozen Lake environment with the default reward function. After training, the two algorithms were tested on the permuted environment with three different reward functions. Results showed that IBL outperformed Q-learning during training (on the default environment with default rewards). Furthermore, IBL performed better than Q-learning for the default and negative reward functions during testing on the permuted environment. Wherever IBL performed better, it achieved a better average success ratio, taking more optimal actions than Q-learning. The difference between the IBL and Q-learning algorithms' success ratios could be attributed to the percentage of optimal actions.

During testing with the negative-and-positive reward function, results showed that Q-learning performed better than IBL. A likely reason behind the reduced success ratio of IBL for negative-and-positive rewards may be the significant change from the way rewards were given in the default reward function. However, another reason could be the values of the default value of the hyperparameters in the IBL model. For example, the default value of the decay parameter (=0.5) [43], [44] meant that the algorithm's agent relied heavily on distant memories. However, these distant memories may not have worked well when the IBL agent was tested in the negative-and-positive reward function, which differed from the training environment. In contrast, Q-learning does not support any form of memory besides the initialization of the new environment's Q-values. Perhaps, this absence of memory helped the Q-learning agent explore the test environment with the negative-and-positive reward function.

There is literature that shows that calibrating memory-based ACT-R model parameters may help the model [45]. For example, Reitter and Lebiere [45] proposed a foraging game where the agents were supposed to communicate with each other and an agent's performance depended on the communicating facts stored in memory. Results revealed that increasing the individual decay rate increased the agent's average task performance. Thus, forgetting was helpful, and it allows the agent to discard outdated information. In a different study, Lebiere [46] showed that first-grade students exhibited a decay value of 0.75 for making additions on large numbers, and proposed that the reliance on calculation (instead of memory) is higher for first-grade school children. Gonzalez *et al.* [47] built IBL models for market entry tasks, which involved four persons and two alternatives (enter or stay out) games. The winning model used a 1.97 value of decay, which made it rely on recent experiences and lowered the chances of repeating past choices over trials in the task. Together, these results suggest that modelers may calibrate the decay hyperparameter's value to alter the model's reliance on memory. This calibration and more careful testing is the goal of the next experiment.

B. EXPERIMENT 2: IBL VERSUS Q-LEARNING WITH NEGATIVE-AND-POSITIVE REWARD FUNCTION IN THE 4 × 4 PERMUTED FROZEN LAKE ENVIRONMENT

This experiment's objective was to compare the performance of IBL and Q-learning on the permuted 4 × 4 Frozen Lake environment with the negative-and-positive reward function after tuning hyper-parameters. The genetic algorithm (GA) [48] was used for hyper-parameter tuning.

1) METHODS

a: ENVIRONMENT, REWARD FUNCTION, PERFORMANCE METRICS, MODEL PARAMETERS, AND DATA ANALYSES

GA [48] is an optimization algorithm inspired by Charles Darwin's natural evolution theory and natural selection [48]. Effectively, the fittest individuals are selected for reproduction to produce the offspring of the next generation. The success ratio (as described in experiment 1) was used as the fitness function in this context. A population size of 20 was chosen, and 100 generations were run with a 20% mutation rate and 80% crossover rate [49]. An average of 5 agents per algorithm was taken (for each hyper-parameter combination) instead of a single agent to reduce the effect of randomness. If the change in the fitness value was less than 0.05 in the last ten generations, the calibration process was stopped.

The hyper-parameter values found by GA for IBL were 0.76, 1.21, and 5,764 for the cognitive noise, decay, and default utility, respectively. The hyper-parameter values found for Q-learning were 0.81, 0.96, and 0.90 for the learning rate, gamma, and epsilon, respectively. During training, the default 4 × 4 Frozen Lake environment with the default reward function (Table 5) was used. The negative-and-positive reward function (Table 5) was used during testing in the permuted 4 × 4 Frozen Lake environment. The success ratio and optimal actions were used for the performance metrics (as described in experiment 1). IBL and Q-learning algorithms were run with 50 agents and for 1500 episodes (with the newly found hyperparameters) to generate the dependent measures during testing. The same data analyses as experiment 1 were performed.

2) RESULT

a: TESTING OF IBL AND Q-LEARNING ON THE PERMUTED 4 × 4 FROZEN LAKE ENVIRONMENT WITH NEGATIVE-AND-POSITIVE REWARDS

Fig. 8(a) shows the average success ratio of IBL and Q-learning algorithms averaged over all episodes and participants for the permuted 4 × 4 Frozen Lake test environment with negative-and-positive rewards (after hyper-parameter calibration). As seen in Fig. 8(a), IBL achieved a similar success ratio as Q-learning for the

TABLE 6. Results for training and testing of IBL and Q-learning algorithms.

Condition	IBL	Q - Learning	ANOVA results	Interaction between episodes and algorithms	Mauchly's Test
Success ratio when trained on default rewards	0.023	0.015	F (1, 97) = 49.29, $p < 0.001$, $\eta^2 = 0.337$	F (1, 253.84) = 2.663, $p < 0.001$, $\eta^2 = 0.027$	Violated* $\chi^2(434) = 6620.34$, $p < 0.05$
Success ratio when tested on default rewards	0.024	0.016	F (1, 97) = 43.61, $p < 0.001$, $\eta^2 = 0.310$	F (1, 264.53) = 3.62, $p < 0.001$, $\eta^2 = 0.036$).	Violated* $\chi^2(434) = 6620.34$, $p < 0.05$
Success ratio when tested on negative rewards	0.022	0.017	F (1,97) = 75.60, $p < 0.001$, $\eta^2 = 0.438$	F (1, 255.64) = 2.40, $p < 0.001$, $\eta^2 = 0.024$)	Violated* $\chi^2(434) = 6620.34$, $p < 0.05$
Success ratio when tested on negative-and-positive rewards	0.006	0017	F (1,97) = 68.80, $p < 0.001$, $\eta^2 = 0.415$	F (1, 265.54) = 2.687, $p < 0.001$, $\eta^2 = 0.027$	Violated* $\chi^2(434) = 6620.34$, $p < 0.05$

* Greenhouse-Geisser correction ($\varepsilon = 0.094$) was used when Mauchly's test was violated

negative-and-positive reward function in the permuted 4 × 4 Frozen Lake (IBL: 0.017 ∼ Q-learning: 0.018; F (1, 97) = 1.072, $p = 0.31$, $\eta^2 = 0.011$). Fig. 8(b) shows the success ratio of IBL and Q-learning algorithms in each episode bin averaged over all participants. Mauchly's test for the success ratio indicated that the assumption of sphericity was violated (χ^2 (434) = 6525.87, $p < 0.05$). Therefore, degrees of freedom were corrected using Greenhouse-Geisser correction ($\varepsilon = 0.089$). The interaction between algorithm and episodes (F (1,255.98) = 0.59, $p = 0.10$, $\eta^2 = 0.005$) was also non-significant.

b: OPTIMAL ACTIONS BY IBL AND Q-LEARNING

Table 7 shows that the IBL algorithm achieved a higher or equal percentage of optimal actions than the Q-learning algorithm in 100% of the cells (i.e., 8 out of the 8 cells) in the permuted 4 × 4 Frozen Lake test environment with negative-and-positive-reward.

3) DISCUSSION

In this experiment, the hyper-parameters of both algorithms were tuned using GA. Both algorithms were run on the permuted 4 × 4 Frozen Lake environment (with positive-and-negative reward). This environment was where IBL (with default hyperparameters) had performed poorly against Q-learning (with default hyperparameters) in experiment 1. Results showed that IBL matched Q-learning's performance after the hyper-parameter tuning, and the difference in the success ratio was not significant. Also, the number of optimal actions was equal or higher in IBL compared to Q-learning. IBL always took equal or more optimal actions than Q-learning.

Thus, the increase in the success ratio of IBL can be attributed to the change in the hyper-parameter values. The new values of cognitive noise, decay, and default utility were 0.76, 1.21, and 5,764, different from the default values 0.25, 0.50, and 10000 used in experiment 1. The increased decay reflects the reduced reliance on memory, while the increase in cognitive noise makes the IBL agent more robust to the change in environment. The decrease in the default utility meant that the agent would be forced to explore less. Together, it meant that the agent could adjust better to the changes in the environment during testing.

The results of the first and second experiments showed that IBL could perform and generalize better than Q-learning. In the next experiment, the scalability of the two algorithms was compared by scaling the Frozen Lake environment to an 8 × 8 matrix instead of the default 4 × 4.

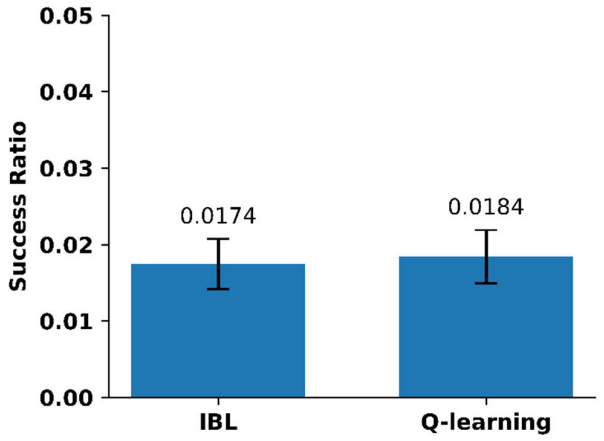
C. EXPERIMENT 3: IBL VERSUS Q-LEARNING WITH DEFAULT PARAMETERS IN THE 8 × 8 FROZEN LAKE ENVIRONMENT

This experiment's objective was to compare IBL and Q-learning algorithms' performance on a scaled 8 × 8 Frozen Lake environment.

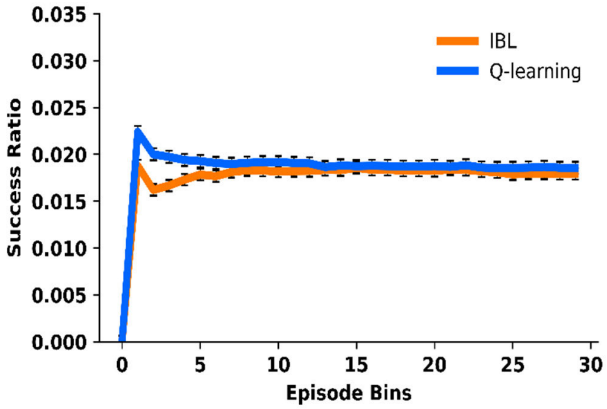
1) METHODS

a: ENVIRONMENT, REWARD FUNCTION, AND PERFORMANCE METRICS, MODEL PARAMETERS, AND DATA ANALYSES

In the third and last experiment, the original Frozen Lake was scaled to an 8 × 8 matrix, and the experiments were repeated with default reward to compare IBL and Q-learning's scalability. Fig. 9 shows the new 8 × 8 environment that was used for training. For testing, the 8 × 8 Frozen Lake environment was randomly permuted (see Fig. 10). Default reward was used for both the new and permuted 8 × 8 Frozen Lake environments (as described in Table 5). Success ratio and optimal actions (as described in experiment 1) were used to compare the two algorithms. The only change in this experiment's parameters was increasing the number of episodes to 5000 due to the increase in the number of cells (sixty-four compared to sixteen in the previous experiments). Like experiment 1, for visualizing the results, the success ratio



(a)



(b)

FIGURE 8. Performance of the algorithms when tested with negative-and-positive rewards on the permuted 4 × 4 Frozen Lake test environment. (a) The average success ratio for IBL and Q-learning; (b) The average success ratio in each episode bin across both IBL and Q-learning algorithms. The error bars show 95% CI around the point estimate.

was averaged for each set of 50 contiguous episodes into an episode bin. Here, the number of episode bins was 100 (=5000 episodes/50 episodes per bin). A similar analysis of the number of optimal steps taken at each cell was performed to ensure that the agents were learning. The other parameters remained the same as described in experiment 1. The same data analyses, as experiment 1, were performed.

2) RESULTS

a: TRAINING OF IBL AND Q-LEARNING ON THE DEFAULT 8 × 8 FROZEN LAKE ENVIRONMENT WITH DEFAULT REWARDS

Fig. 11(a) shows the average success ratio of IBL and Q-learning algorithms averaged over all episodes and participants in the default 8 × 8 Frozen Lake training environment with default rewards. As seen in Fig. 11(a), IBL in the training environment achieved a better average success

1 S	2 F	3 F	4 F	5 H	6 F	7 F	8 H
9 F	10 F	11 F	12 F	13 H	14 F	15 H	16 F
17 F	18 F	19 F	20 F	21 H	22 F	23 H	24 F
25 H	26 F	27 F	28 H	29 F	30 F	31 F	32 F
33 H	34 F	35 F	36 F	37 F	38 F	39 F	40 F
41 F	42 F	43 F	44 F	45 F	46 F	47 F	48 F
49 F	50 F	51 F	52 F	53 F	54 F	55 F	56 H
57 F	58 F	59 F	60 F	61 F	62 H	63 F	64 G

FIGURE 9. Default 8 × 8 Frozen Lake training environment with S: Starting Cell, F: Frozen Surface, H: Hole, and G: Goal.

1 S	2 F	3 F	4 F	5 F	6 F	7 F	8 F
9 F	10 F	11 F	12 F	13 F	14 F	15 F	16 F
17 F	18 F	19 F	20 H	21 F	22 F	23 F	24 F
25 F	26 F	27 F	28 F	29 F	30 H	31 F	32 F
33 F	34 F	35 F	36 H	37 F	38 F	39 F	40 F
41 F	42 H	43 H	44 F	45 F	46 F	47 H	48 F
49 F	50 H	51 F	52 F	53 H	54 F	55 H	56 F
57 F	58 F	59 F	60 H	61 F	62 F	63 F	64 G

FIGURE 10. Permuted 8 × 8 Frozen Lake test environment with S: Starting Cell, F: Frozen Surface, H: Hole, and G: Goal.

ratio than Q-learning (IBL: 0.008 > Q-learning: 0.002; F (1, 97) = 468.01, $p < 0.001$, $\eta^2 = 0.828$). Fig. 11(b) shows the success ratio of IBL and Q-learning algorithms in each episode bin averaged over all participants in the training environment. Mauchly's test showed that the success ratio's sphericity was not violated ($p = .188$, χ^2 (302) = 4892). As seen in the previous experiments, there was an interaction effect of the algorithm and episodes (F (1, 97) = 2.781, $p < 0.001$, $\eta^2 = 0.028$). The success ratio increased rapidly for the first few episodes, and the increase and subsequent stabilization were much higher in the IBL algorithm than in the Q-learning algorithm. The difference in the average success ratio between the two curves was much higher in the scaled version. It is evident that even for large environments, IBL performed better than Q-learning.

b: TESTING OF IBL AND Q-LEARNING ON THE PERMUTED 8 × 8 FROZEN LAKE ENVIRONMENT WITH DEFAULT REWARDS

Fig. 12(a) shows the average success ratio of IBL and Q-learning algorithms averaged over all episodes and participants in the permuted 8 × 8 Frozen Lake test environment with default rewards. As seen in Fig. 12(a), IBL achieved a better average success ratio than Q-learning during testing as well (IBL: 0.025 > Q-learning: 0.006; F (1, 97) = 43.678,

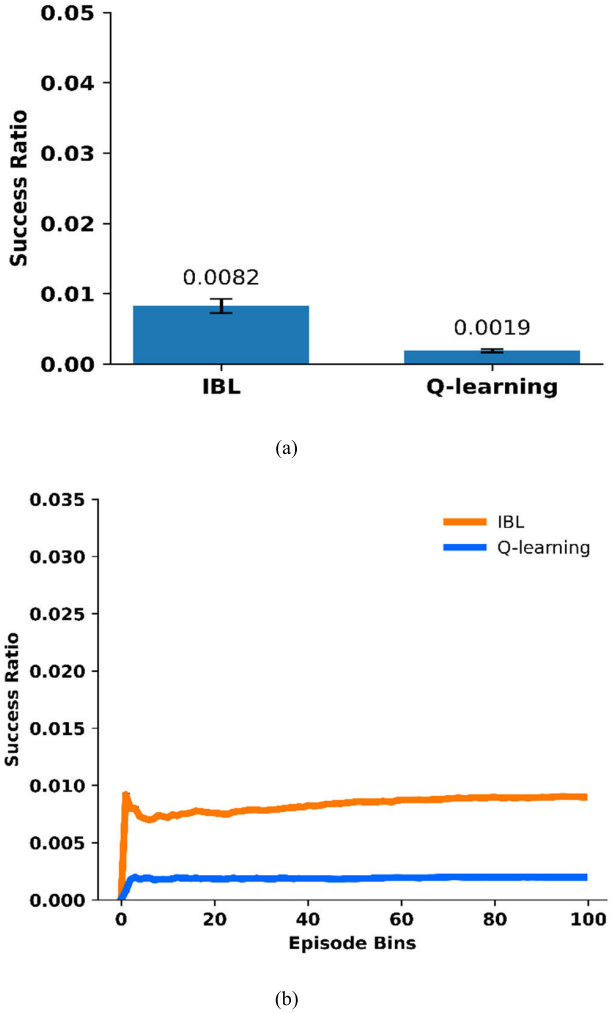


FIGURE 11. Performance of the algorithms when run in default 8 × 8 Frozen Lake environment with default rewards. (a) The average success ratio for IBL and Q-learning and (b) The average success ratio in each episode bin across both IBL and Q-learning algorithms. The error bars show 95% CI around the point estimate.

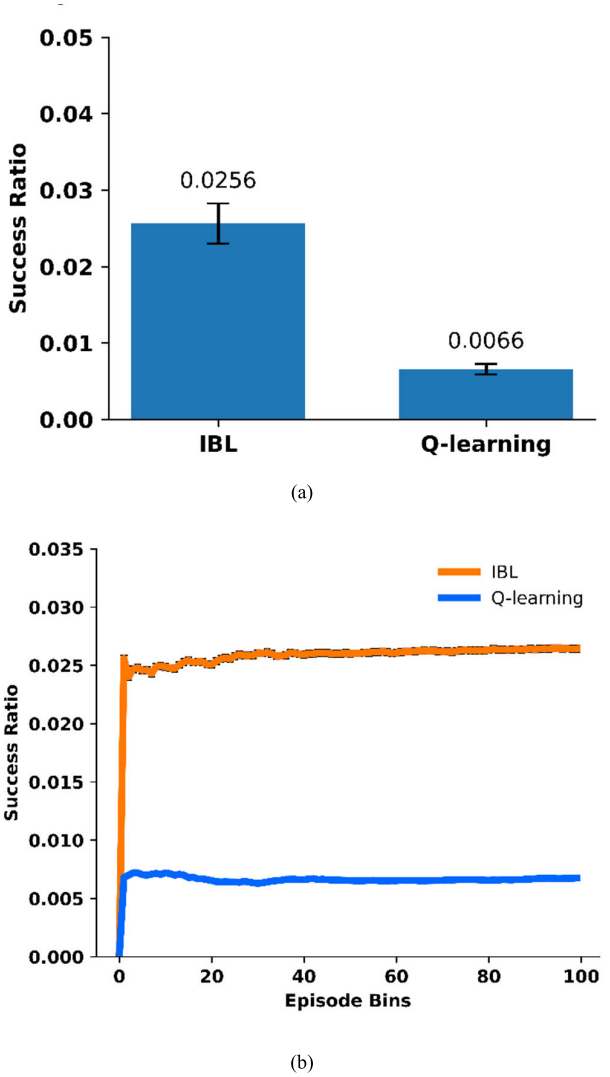


FIGURE 12. Performance of the algorithms when run in permuted 8 × 8 Frozen Lake environment with default rewards. (a) The average success ratio for IBL and Q-learning (b) The average success ratio in each episode bin across both IBL and Q-learning algorithms. The error bars show 95% CI around the point estimate.

$p < 0.001$, $\eta^2 = 0.310$). Fig. 12(b) shows the success ratio of IBL and Q-learning algorithms in each episode averaged over all participants in the test environment. Mauchly's test showed that the success ratio's sphericity was not violated ($p = .152$, $\chi^2\,(302) = 4850$). There was an interaction effect of the algorithm and episodes ($F\,(1, 97) = 3.628$, $p < 0.001$, $\eta^2 = 0.036$). A similar curve was observed during testing, with IBL significantly outperforming Q-learning.

c: OPTIMAL ACTIONS BY IBL AND Q-LEARNING

Fig. 13(a) shows that for the default 8 × 8 Frozen Lake training environment with default rewards. The IBL algorithm achieved a higher percentage of optimal actions than the Q-learning algorithm in 100% of the cells (i.e., 51 cells out of the 51 cells). Fig. 13(b) shows that for the permuted 8 × 8 Frozen Lake test environment with default rewards. The IBL algorithm again achieved a higher percentage of optimal actions than the Q-learning algorithm in 100% of the cells (i.e., 41 out of the 41 cells). The hyper-parameter values for both IBL and Q-learning were the same as in experiment 1.

TABLE 7. Optimal actions taken in negative-and-positive reward after hyper-parameter calibration.

S IBL: 60, QL: 55	F IBL: 40, QL: 38	F IBL: 13, QL: 12	F
F IBL: 22, QL: 20	F IBL: 17, QL: 17	F IBL: 11, QL: 10	H
F	H	F IBL: 4.2, QL: 3.6	H
H	F	F IBL: 1.2, QL: 1.1	G

3) DISCUSSION

In this experiment, the algorithms were run on a larger 8 × 8 version of Frozen Lake to test the scalability. Results showed

S I: 68.12 Q: 56.11	F I: 46.72 Q: 43.44	F I: 34.53 Q: 31.46	F I: 23.44 Q: 20.59	F I: 15.01 Q: 12.80	F I: 10.07 Q: 7.8	F I: 7.20 Q: 5.04	F I: 2.85 Q: 1.71
F I: 46.65 Q: 43.47	F I: 41.94 Q: 41.65	F I: 34.60 Q: 33.39	F I: 12.79 Q: 10.48	F I: 15.59 Q: 13.24	F I: 11.01 Q: 8.39	F I: 8.15 Q: 5.52	F I: 3.21 Q: 1.98
F I: 34.47 Q: 31.63	F I: 34.21 Q: 33.24	F I: 15.35 Q: 12.94	H	F I: 9.07 Q: 7.33	F I: 4.30 Q: 3.05	F I: 7.41 Q: 4.62	F I: 3.39 Q: 1.63
F I: 13.62 Q: 11.19	F I: 12.76 Q: 11.90	F I: 10.33 Q: 9.2	F I: 4.64 Q: 3.53	F I: 3.30 Q: 2.11	H	F I: 4.80 Q: 2.72	F I: 2.81 Q: 1.12
F	F	F	H	F I: 2.61 Q: 1.32	F I: 1.71 Q: 0.88	F I: 1.19 Q: 0.72	F I: 2.12 Q: 0.62
F	H	H	F	F I: 0.48 Q: 0.23	F I: 0.67 Q: 0.19	H	F I: 1.46 Q: 0.33
F	H	F	F	H	F I: 0.26 Q: 0.05	H	F I: 0.79 Q: 0.16
F	F	F	H	F	F I: 0.05 Q: 0.01	F I: 0.03 Q: 0.005	G

(a)

S I: 68.35 Q: 56.43	F I: 46.68 Q: 43.42	F I: 34.29 Q: 31.42	F I: 23.06 Q: 20.65	F I: 7.38 Q: 6.32	H	F	H
F I: 46.40 Q: 43.13	F I: 41.78 Q: 41.36	F I: 34.06 Q: 33.08	F I: 12.33 Q: 10.16	F I: 7.32 Q: 5.65	F I: 5.46 Q: 3.55	F I: 2.31 Q: 1.28	F I: 0.47 Q: 0.25
F I: 34.44 Q: 31.21	F I: 33.81 Q: 32.67	F I: 14.94 Q: 12.89	H	H	F I: 1.30 Q: 0.69	F I: 1.88 Q; 0.85	F I: 0.61 Q: 0.24
F I: 24.08 Q: 21.28	F I: 24.87 Q: 23.23	F I: 19.89 Q: 18.04	F I: 4.23 Q: 4.29	H	F I: 1.7 Q: 0.83	F I: 0.80 Q: 0.29	F I: 0.61 Q: 0.19
F I: 16.28 Q: 13.40	F I: 9.09 Q: 7.47	F I: 14.11 Q: 11.03	F I: 9.11 Q: 6.0	F I: 4.6 Q: 2.67	F I: 0.99 Q: 0.63	H	F I: 0.43 Q: 0.13
F I: 4.98 Q: 2.94	H	F I: 7.02 Q: 4.91	F I: 6.43 Q: 3.67	F I: 2.42 Q: 1.05	F I: 3.05 Q: 1.21	F I: 1.81 Q: 0.54	F I: 0.47 Q: 0.14
F I: 2.21 Q: 1.08	F I: 1.75 Q: 1.20	F I: 2.19 Q: 1.32	F I: 2.20 Q: 1.06	F I: 3.62 Q: 1.36	F I: 2.94 Q: 0.98	F I: 2.23 Q: 0.56	F I: 0.59 Q: 0.15
F	F	F	H	F I: 0.69 Q: 0.30	F I: 0.97 Q: 0.27	F I: 1.0 Q: 0.20	G

(b)

FIGURE 13. Optimal action percentages by IBL and Q-learning on (a) 8 × 8 Frozen Lake training environment with default rewards; (b) permuted 8 × 8 Frozen Lake test environment with default rewards. Note: I represent IBL, and Q represents Q-learning.

that IBL outperformed Q-learning and the difference was higher than the 4 × 4 versions in experiment 2. The success ratio and optimal actions reinforced that the IBL algorithm was better than the Q-learning algorithm in terms of success ratios and optimal actions.

IV. GENERAL DISCUSSION

In this paper, two approaches to reinforcement learning were compared when the algorithms were articulated to maximize the expected reward in a Markov decision process [3]. While Q-learning used statistical algorithms for learning, IBL relied on cognitive mechanisms, which humans appear to learn [35].

In the first experiment, it was observed that IBL outperformed Q-learning in two of the three reward functions during testing. It was hypothesized that the underperformance of IBL was due to the significant change in the reward functions during training and testing. The underlying reason was the *decay* hyper-parameter in IBL, which controlled how much the algorithm relied on the memory [17]. A value of 0.5 meant that the algorithm was relying heavily on a distant memory. Perhaps, the IBL algorithm could not explore enough in a different environment.

For this reason, it was decided to undertake the second experiment, where the hyper-parameters of both algorithms were calibrated using a genetic algorithm. The new values of hyper-parameters indicated that the amount of randomness in IBL might be increased, which confirmed our hypothesis. The second experiment's results were as expected, and IBL outperformed Q-learning for the positive-and-negative reward function as well. In the third experiment, the environment was scaled to an 8 × 8 Frozen Lake. It was found that IBL could retain its lead even in the scaled environment, and the margin of difference was even higher this time.

V. CONCLUSION

To conclude, the results showed that IBL ran faster and learned the optimal paths better than Q-learning. The role of memory was also investigated when the agent was tested in a significantly different environment from the training environment. Having taken memory into account, the IBL agent trained faster, scaled, and generalized better. Our testing framework also allowed us to infer the two algorithms' learning mechanisms correctly. The hyper-parameters were set via GA to find better and more robust learning parameters in IBL.

The current investigation on using IBL (as a possible alternative to Q-learning) has revealed promising results. It is expected that the reported results would generalize to other similar environments given their testing in several reward settings and on the scaled 8 × 8 matrices. One theoretical implication of the results is developing better machine learning algorithms for complex dynamic environments that incorporate the different cognitive mechanisms proposed in the IBL algorithm. These cognitive mechanisms in the IBL algorithm have been developed by relying on humans' behavioral data across several studies [14], [15]. One practical implication of cognitive techniques inclusion may enable machine learning algorithms to make faster decisions as people do in different decision tasks. Also, including cognitive techniques in machine learning algorithms may make them learn faster while training and transfer faster to novel environments. In IBL, at any point in time, the decision-making

process can be investigated by enumerating the memory contents, what could be retrieved, and why something gets retrieved. This interpretability allows us to explain the IBL algorithm's decisions, which is not necessarily the case with deep learning algorithms. Thus, the IBL algorithm promises a solution to two of the most significant drawbacks of machine learning or deep learning algorithms: generalizability and interpretability.

The IBL algorithm promises to improve the machine learning algorithms and could also be used as a decision aid to improve learning among people. Cognitive algorithms like IBL have already been used to improve learning among people in several practical applications like teaching LISP [50] and Algebra [51] to kids. One could also use IBL-like algorithms to improve learning abilities among children [52]. Overall, there are several practical applications of this research to both machine and human learning.

However, there are certain limitations in the current study. For example, some learning rate decay techniques were not used in the Q-learning algorithm, which could have sped up the convergence. However, even after using learning rate decay, one may not expect differences in the IBL and Q-learning algorithms' reported results. The IBL algorithm would have also converged faster if the learning decay had been used in it. Another limitation of our approach is that the Q-learning algorithm's deep learning variants were not considered in this paper. One reason for not doing so was that there was no deep version available for the IBL algorithm. The comparisons of deeper variants of IBL and Q-learning algorithms are planned as a future endeavor.

There are several research questions to pursue as part of future research. For example, one could explore other OpenAI gym environments, including gaming environments, and compare different IBL and Q-learning algorithms. Second, it may be worthwhile to investigate different sampling strategies while exploring IBL and Q-learning algorithms. These sampling strategies may be motivated by how people search between different options. Some of these ideas form the immediate next steps in our program of model comparison in learning environments.

REFERENCES

[1] R. Gross, *Psychology: The Science of Mind and Behavior*, 7th ed. London, U.K.: Hodder, 2015.

[2] J. Piaget, "Part I: Cognitive development in children: Piaget development and learning," *J. Res. Sci. Teaching*, vol. 2, no. 3, pp. 176–186, Sep. 1964.

[3] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[4] S. E. Barbosa and M. D. Petty, "Reinforcement learning in an environment synthetically augmented with digital pheromones," *Adv. Artif. Intell.*, vol. 2014, pp. 1–23, Mar. 2014.

[5] A. L. Thomaz and C. Breazeal, "Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance," in *Proc. AAAI*, vol. 6, 2006, pp. 1000–1005.

[6] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2017, pp. 3389–3396.

[7] E. Care, P. Griffin, C. Scoular, N. Awwal, and N. Zoanetti, "Collaborative problem-solving tasks," in *Assessment and Teaching of 21st Century Skills*. Springer, 2015, pp. 85–104.

[8] J. Funke, "Complex problem solving: A case for complex cognition?" *Cognit. Process.*, vol. 11, no. 2, pp. 133–142, May 2010.

[9] C. Gonzalez, P. Vanyukov, and M. K. Martin, "The use of microworlds to study dynamic decision making," *Comput. Hum. Behav.*, vol. 21, no. 2, pp. 273–286, Mar. 2005.

[10] J. D. Sterman, "Learning in and about complex systems," *Syst. Dyn. Rev.*, vol. 10, nos. 2–3, pp. 291–330, 1994.

[11] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI gym," 2016, *arXiv:1606.01540*. [Online]. Available: https://arxiv.org/abs/1606.01540

[12] T. Beysolow II, *Applied Reinforcement Learning With Python: With OpenAI Gym*. New York, NY, USA: Springer, 2019.

[13] S. Ravichandiran, *Hands-On Reinforcement Learning With Python: Master Reinforcement and Deep Reinforcement Learning Using OpenAI Gym and TensorFlow*. Birmingham, U.K.: Packt Publishing Ltd, 2018.

[14] M. Alloghani, D. Al-Jumeily, J. Mustafina, A. Hussain, and A. J. Aljaaf, "A systematic review on supervised and unsupervised machine learning algorithms for data science," in *Supervised and Unsupervised Learning for Data Science*, M. W. Berry, A. Mohamed and B. W. Yap, Eds. Cham, Switzerland: Springer, 2020, pp. 3–21.

[15] T. Mitchell, B. Buchanan, G. de Jong, T. Dietterich, and P. Rosenbloon, "Machine learning annual review of computer science," *J. Comput. Sci.*, vol. 4, pp. 417–433, 1990.

[16] C. Gonzalez, J. F. Lerch, and C. Lebiere, "Instance-based learning in dynamic decision making," *Cognit. Sci.*, vol. 27, pp. 591–635, Jul./Aug. 2003.

[17] C. Gonzalez and V. Dutt, "Instance-based learning: Integrating sampling and repeated decisions from experience," *Psychol. Rev.*, vol. 118, no. 4, p. 523, 2011.

[18] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.

[19] A. W. Beggs, "On the convergence of reinforcement learning," *J. Econ. Theory*, vol. 122, no. 1, pp. 1–36, May 2005.

[20] I. Erev and A. E. Roth, "Predicting how people play games: Reinforcement learning in experimental games with unique, mixed strategy equilibria," *Amer. Econ. Rev.*, vol. 88, pp. 848–881, Sep. 1998.

[21] J.-F. Laslier, R. Topol, and B. Walliser, "A behavioral learning process in games," *Games Econ. Behav.*, vol. 37, no. 2, pp. 340–366, Nov. 2001.

[22] A. E. Roth and I. Erev, "Learning in extensive-form games: Experimental data and simple dynamic models in the intermediate term," *Games Econ. Behav.*, vol. 8, no. 1, pp. 164–212, 1995.

[23] V. Dutt, "Explaining human behavior in dynamic tasks through reinforcement learning," *J. Adv. Inf. Technol.*, vol. 2, no. 3, pp. 177–188, Aug. 2011, doi: 10.4304/jait.2.3.177-188.

[24] J. Fan, Z. Wang, Y. Xie, and Z. Yang, "A theoretical analysis of deep Q-learning," 2019, *arXiv:1901.00137*. [Online]. Available: https://arxiv.org/abs/1901.00137

[25] H. Singal, P. Aggarwal, and V. Dutt, "Modeling decisions in games using reinforcement learning," in *Proc. Int. Conf. Mach. Learn. Data Sci. (MLDS)*, Dec. 2017, pp. 98–105, doi: 10.1109/MLDS.2017.13.

[26] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017, doi: 10.1109/MSP.2017.2743240.

[27] S. S. Mousavi, M. Schukat, and E. Howley, "Deep reinforcement learning: An overview," *Proceedings of SAI Intelligent Systems Conference*, vol. 16. Cham, Switzerland: Springer, 2018, pp. 426–440, doi: 10.1007/978-3-319-56991-8_32.

[28] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Comput. Surv.*, vol. 50, no. 2, p. 21, Apr. 2017, doi: 10.1145/3054912.

[29] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015, doi: 10.1038/nature14236.

[30] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *J. Artif. Intell. Res.*, vol. 47, pp. 253–279, Jun. 2013, doi: 10.1613/jair.3912.

[31] C. Gonzalez and C. Lebiere, "Instance-based cognitive models of decision making," in *Transfer of Knowledge in Economic Decision-Making*, D. Zizzo and A. Courakis, Eds. New York, NY, USA: Palgrave Macmillan, 2005, pp. 148–165.

[32] C. Lebiere, C. Gonzalez, and M. Martin, "Instance-based decision-making model of repeated binary choice," in *Proc. 8th Int. Conf. Cogn. Modeling*, Ann Arbor, MI, USA, 2007.
[33] C. A. Stevens, N. A. Taatgen, and F. Cnossen, "Instance-based models of metacognition in the Prisoner's dilemma," *Topics Cognit. Sci.*, vol. 8, no. 1, pp. 322–334, Jan. 2016.
[34] V. Dutt and C. Gonzalez, "The role of inertia in modeling decisions from experience with instance-based learning," *Frontiers Psychol.*, vol. 3, p. 177, Jun. 2012.
[35] T. Lejarraga, V. Dutt, and C. Gonzalez, "Instance-based learning: A general model of repeated binary choice," *J. Behav. Decis. Making*, vol. 25, no. 2, pp. 143–153, Apr. 2012.
[36] V. Dutt, Y.-S. Ahn, and C. Gonzalez, "Cyber situation awareness: Modeling the security analyst in a cyber-attack scenario through instance-based learning," in *Proc. IFIP Annu. Conf. Data Appl. Secur. Privacy*, 2011, pp. 280–292.
[37] V. Dutt, Y.-S. Ahn, and C. Gonzalez, "Cyber situation awareness: Modeling detection of cyber attacks with instance-based learning theory," *Hum. Factors, J. Hum. Factors Ergonom. Soc.*, vol. 55, no. 3, pp. 605–618, Jun. 2013.
[38] R. A. Howard, *Dynamic Programming and Markov Processes*. Washington, DC, USA: Wiley, 1960.
[39] F. S. Melo, "Convergence of Q-learning: A simple proof," Inst. Syst. Robot., Tech. Rep., Sep. 2001, pp. 1–4.
[40] J. R. Anderson, *The Adaptive Character of Thought*. Hove, U.K.: Psychology Press, 2013.
[41] P. Langley and H. A. Simon, "The central role of learning in cognition," in *Cognitive Skills and Their Acquisition*. Bowling Green, OH, USA: JSTOR, 1981, pp. 361–380.
[42] J. R. Busemeyer and Y.-M. Wang, "Model comparisons and model selections based on generalization criterion methodology," *J. Math. Psychol.*, vol. 44, no. 1, pp. 171–189, Mar. 2000, doi: 10.1006/jmps.1999.1282.
[43] R. Thomson, C. Lebiere, J. R. Anderson, and J. Staszewski, "A general instance-based learning framework for studying intuitive decision-making in a cognitive architecture," *J. Appl. Res. Memory Cognition*, vol. 4, no. 3, pp. 180–190, Sep. 2015, doi: 10.1016/j.jarmac.2014.06.002.
[44] J. R. Anderson and C. Schunn, "Implications of the ACT-R learning theory: No magic bullets," in *Advances in Instructional Psychology, Educational Design and Cognitive Science*. Mahwah, NJ, USA: Lawrence Erlbaum Associates, 2000, pp. 1–33.
[45] D. Reitter and C. Lebiere, "Social cognition: Memory decay and adaptive information filtering for robust information maintenance," in *Proc. Nat. Conf. Artif. Intell.*, vol. 1, Jan. 2012, pp. 242–248.
[46] C. Lebiere, "The dynamics of cognition: An ACT-R model of cognitive arithmetic," *Kognitionswissenschaft*, vol. 8, no. 1, pp. 5–19, Jun. 1999, doi: 10.1007/BF03354932.
[47] C. Gonzalez, V. Dutt, and T. Lejarraga, "A loser can be a winner: Comparison of two instance-based learning models in a market entry competition," *Games*, vol. 2, no. 1, pp. 136–162, Mar. 2011, doi: 10.3390/g2010136.
[48] D. Whitley, "A genetic algorithm tutorial," *Statist. Comput.*, vol. 4, no. 2, pp. 65–85, Jun. 1994.
[49] V. Dutt and C. Gonzalez, "Making instance-based learning theory usable and understandable: The instance-based learning tool," *Comput. Hum. Behav.*, vol. 28, no. 4, pp. 1227–1240, Jul. 2012, doi: 10.1016/j.chb.2012.02.006.
[50] A. T. Corbett, K. R. Koedinger, and W. H. Hadley, "Cognitive tutors: From the research classroom to all classrooms," in *Technology Enhanced Learning: Opportunities for Change*. Abingdon, U.K.: Routledge, 2001, pp. 235–263.
[51] S. Ritter, J. R. Anderson, K. R. Koedinger, and A. Corbett, "Cognitive tutor: Applied research in mathematics education," *Psychonomic Bull. Rev.*, vol. 14, no. 2, pp. 249–255, Apr. 2007.
[52] K. R. Butcher and V. Aleven, "Integrating visual and verbal knowledge during classroom learning with computer tutors," in *Proc. Annu. Meeting Cognit. Sci. Soc.*, vol. 29, 2007, pp. 137–142.

ANMOL GUPTA is currently pursuing the Ph.D. degree in computer science with the Indian Institute of Technology Roorkee, India. In his current research, he identifies cognitive signatures of depression using cognitive modeling, computational neuroscience, and machine learning. His research interests include machine learning, deep learning, and reinforcement learning.

PARTHA PRATIM ROY (Senior Member, IEEE) received the M.S. and Ph.D. degrees from Universitat Autonoma de Barcelona, Spain. He worked with TATA Consultancy Services from 2003 to 2005. From 2011 to 2012, he was a Postdoctoral Research Fellow with the RFAI Laboratory, France. He was with the Synchromedia Laboratory, Canada, in 2013. He worked with the Advanced Technology Group, Samsung Research Institute Noida, India, from 2013 to 2014. He is currently an Associate Professor with the Department of Computer Science and Engineering, IIT Roorkee, India. He has published more than 225 articles in international journals and conferences. His research interests include pattern recognition, human–computer interaction, bio-signal analysis, and multilingual text recognition. He is an Associate Editor of *IET Image Processing*, *IET Biometrics*, *IEICE Transactions on Information and Systems*, and *Computer Science* (Springer Nature). He is a Regional Editor of the *Journal of Multimedia and Information System* and a Guest Editor of *International Journal of Distributed Sensor Networks*.

VARUN DUTT (Senior Member, IEEE) received the Ph.D. degree in engineering and public policy from Carnegie Mellon University, in 2011. He currently works as an Associate Professor with the School of Computing and Electrical Engineering and the School of Humanities and Social Sciences, Indian Institute of Technology, Mandi. He is serving as an Associate Editor for *Frontiers in Cognitive Science* journal, a Review Editor for *Frontiers in Decision Neuroscience* journal, and a member for the Editorial Board of the *International Journal on Cyber Situational Awareness*.