Evaluating Multiple Reinforcement Learning Agents in the Frozen Lake Environment in Scalability and Stochasticity

Julie Jeong
(sj598)

Group 3

Sophia Pham
(tpp38)

1 Introduction

The Frozen Lake environment (Gymnasium Frozen Lake) is a classic RL testbed where agents navigate a grid to reach a goal while avoiding traps [1]. Adding to the challenge, the environment can be "slippery" (`is_slippery=True`), meaning the agent's movements are subject to randomness: there is only a 1/3 chance of moving in the intended direction, with equal probabilities of moving perpendicularly. This stochastic behavior requires algorithms to be both adaptable and robust.

In this project, we hypothesize that (1) as the grid size increases (from 4x4 to 5x5), function approximation methods and policy gradient algorithms (REINFORCE, Actor-Critic) will perform better than tabular Q-learning due to their ability to handle larger state-action spaces, and (2) when the environment is stochastic (`is_slippery=True`), policy gradient methods will show greater adaptability compared to tabular Q-learning and function approximation. These hypotheses reflect our interest in exploring how different RL methods handle increasing complexity and randomness.

To test these hypotheses, we implemented tabular Q-learning, function approximation with a linear model, and policy gradient methods in the Frozen Lake environment. By comparing their performance across different grid sizes and levels of stochasticity, we aim to better understand the trade-offs between scalability, adaptability, and robustness for these algorithms in uncertain environments.

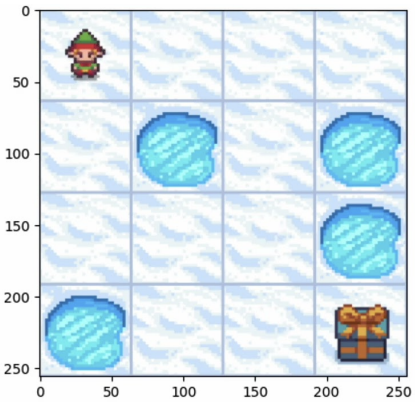

Figure 1: Visualization of the Frozen Lake 4x4 environment. The agent starts in the top-left corner and must navigate to the goal in the bottom-right corner, avoiding holes.

2 Problem

The Frozen Lake environment is represented as a grid where the agent starts at [0,0] and must navigate to a goal located at far extent of the world (e.g. [3,3] for the 4x4 environment) while avoiding holes. Each tile in the grid can either be safe, a hole, or the goal. The agent's objective is to maximize its cumulative reward by reaching the goal and avoiding falling into holes, which terminate the episode.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

2.1 Action Space

The action space in Frozen Lake is discrete and consists of four possible actions: **Move Left (0), Move Down (1), Move Right (2), Move Up (3)**. These actions represent the agent's movement choices at each time step. The agent's success rate in moving in the intended direction depends on whether `is_slippery=True`, which introduces stochasticity into the environment.

2.2 Observation Space

The observation space is also discrete and is defined by the total number of states, which depends on the size of the grid: **4x4 grid** has 16 states, each corresponding to a tile, and **5x5 grid** has 25 states. Each state represents the agent's current position on the grid. The state is encoded as an integer that maps to a specific tile in the grid.

2.3 Reward Structure

The reward structure in Frozen Lake is simple in guiding the agent's behavior toward the goal:

- **+1** for reaching the goal tile.
- **0** for stepping onto any other tile, including safe tiles and holes.

There are no explicit penalties for falling into holes or stepping onto non-goal tiles, but falling into a hole ends the episode, creating an implicit penalty by preventing further rewards. This sparse reward system encourages the agent to explore the grid to discover the optimal path to the goal.

For future implementation, these reward structures can be modified such that there is a penalty for stepping on the hole. This would incentivize the agents to avoid the holes further, thus resulting in more optimized results for reaching the goals.

2.4 Agents

To address these challenges, we evaluated five types of agents:

1. **Random Bellman Agent:** A baseline agent that applies random Bellman updates to its policy. It serves as a naive benchmark to highlight the effectiveness of more structured approaches.
2. **Tabular Q-Learning:** A traditional value-based algorithm that explicitly maintains a Q-value table for all state-action pairs, suitable for small state spaces.
3. **Function Approximation:** A method that uses a linear model to estimate Q-values, enabling scalability to larger grids by generalizing across states.
4. **REINFORCE:** A policy gradient method that directly optimizes a stochastic policy, allowing the agent to learn by maximizing cumulative rewards. This approach focuses entirely on the policy and does not rely on a value function.
5. **Actor-Critic:** A hybrid method that combines policy optimization with a value function for variance reduction, offering the strengths of both value-based and policy-based approaches.

2.5 Research Objective

Our primary goal is to understand how these agents perform under varying environmental conditions, including different grid sizes (4x4 and 5x5) and levels of stochasticity (`is_slippery=True/False`). We aim to evaluate their scalability, robustness, and ability to learn effective policies.

3 Approach

To evaluate the hypotheses presented in this study, we implemented and compared the five agents—Random Bellman, Tabular Q-Learning, Function Approximation, REINFORCE, and Actor-Critic—under varying configurations of the Frozen Lake environment. Specifically, the following actions were taken:

- **Grid Sizes**: We tested the agents on two grid sizes, 4×4 and 5×5, to assess scalability. This directly addresses Hypothesis 1, that function approximation and policy gradient methods will outperform tabular Q-learning as the environment size increases.

- **Stochasticity** (`is_slippery`): We evaluated each agent in deterministic (`is_slippery=False`) and stochastic (`is_slippery=True`) conditions to test Hypothesis 2, that policy gradient methods will be more robust to stochasticity than tabular and function approximation methods.

Later in this section, we describe the implementation of the agents used in our analysis. Each agent was developed to deal with the challenges created by the Frozen Lake environment, including the stochasticity introduced by slippery tiles (`is_slippery=True`) and varying grid sizes.

Detailed implementation of all agents can be found at https://github.com/juliejeong/robot-learning-final-project/tree/main.

3.1 Random Bellman Agent

In the Frozen Lake environment, this agent selects actions randomly from the action space, without attempting to optimize its behavior, then updates the value of states using the Bellman equation:

$$V(s) \leftarrow V(s) + \alpha \left[r + \gamma V(s') - V(s) \right]$$

where:

- s: Current state.
- s': Next state after taking an action.
- r: Reward received upon transitioning to s'.
- α: Learning rate, determining the size of updates.
- γ: Discount factor, balancing the importance of immediate and future rewards.
- $V(s)$: Estimated value of the current state.

Algorithm 1 Random Bellman Agent

```
Initialize state values V(s) = 0 for all states
for each episode do
    Initialize starting state s
    while s is not terminal do
        Randomly select an action a from the action space
        Perform action a and observe next state s' and reward r
        Update V(s) ← V(s) + α [r + γV(s') − V(s)]
        Set s ← s'
    end while
end for
```

3.2 Tabular Q-Learning Agent

This agent learns an action-value function $Q(s, a)$ to estimate the expected cumulative reward for each state-action pair [2]. This helps the agent to optimize its behavior in the Frozen Lake environment, even under stochastic conditions. The Q-value is updated as:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right]$$

where:

- s, a: Current state and action.
- s': Next state after taking action a.
- a': Next action chosen to maximize $Q(s', a')$.
- r, α, γ: As defined above.
- $Q(s, a)$: Estimated Q-value of the current state-action pair.

The agent uses an ϵ-greedy policy to balance exploration and exploitation:

$$a = \begin{cases} \text{random action} & \text{with probability } \epsilon \\ \arg\max_a Q(s,a) & \text{otherwise.} \end{cases}$$

Algorithm 2 Tabular Q-Learning Agent

Initialize $Q(s,a) = 0$ for all state-action pairs
for each episode **do**
    Initialize starting state s
    while s is not terminal **do**
        Select action a using ϵ-greedy policy
        Perform action a and observe next state s' and reward r
        Update $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$
        Set $s \leftarrow s'$
    end while
end for

3.3 Function Approximation Agent (Linear Regression)

This agent uses linear regression to generalize Q-values across larger state-action spaces. The Q-value is predicted as:

$$Q(s,a) = \mathbf{w}^\top \phi(s,a)$$

where:

- $\mathbf{w}$: Weight vector, learned through training.
- $\phi(s,a)$: Feature vector representation of the state-action pair.

The weights are updated using the rule:

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right] \phi(s,a)$$

Algorithm 3 Function Approximation Agent

Initialize weights $\mathbf{w} = 0$
for each episode **do**
    Initialize starting state s
    while s is not terminal **do**
        Select action a using ϵ-greedy policy
        Compute features $\phi(s,a)$
        Predict $Q(s,a) = \mathbf{w}^\top \phi(s,a)$
        Perform action a and observe s', r
        Compute target $y = r + \gamma \max_{a'} Q(s',a')$
        Update $\mathbf{w} \leftarrow \mathbf{w} + \alpha(y - Q(s,a))\phi(s,a)$
        Set $s \leftarrow s'$
    end while
end for

3.4 REINFORCE Agent

This agent is a policy gradient method that directly optimizes a stochastic policy $\pi_\theta(a|s)$. The objective is to maximize the expected return:

$$J(\theta) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t\right]$$

The policy is updated using:

$$\theta \leftarrow \theta + \alpha \sum_t \nabla_\theta \log \pi_\theta(a_t|s_t) G_t$$

where G_t is the cumulative reward from time step t.

Algorithm 4 REINFORCE Agent

for each episode **do**
  Initialize empty trajectory τ
  while s is not terminal **do**
    Sample action $a \sim \pi_\theta(a|s)$
    Perform action a, observe r and s'
    Store (s, a, r) in τ
    Set $s \leftarrow s'$
  end while
  Compute G_t for all time steps in τ
  Update policy parameters:

$$\theta \leftarrow \theta + \alpha \sum_t \nabla_\theta \log \pi_\theta(a_t|s_t) G_t$$

end for

3.5 Actor-Critic Agent

This agent combines policy gradients with value-based methods to reduce variance. It uses a policy network $\pi_\theta(a|s)$ and a value network $V_\phi(s)$. The advantage function is computed as:

$$A(s, a) = G_t - V_\phi(s)$$

The policy loss and value loss are:

$$L_{\text{policy}} = -\mathbb{E}\left[\log \pi_\theta(a|s) A(s, a)\right], \quad L_{\text{value}} = \mathbb{E}\left[(G_t - V_\phi(s))^2\right]$$

Algorithm 5 Actor-Critic Agent

for each episode **do**
  while s is not terminal **do**
    Sample action $a \sim \pi_\theta(a|s)$
    Perform action a, observe r and s'
    Compute advantage $A(s, a) = r + \gamma V_\phi(s') - V_\phi(s)$
    Update policy parameters θ using $A(s, a)$
    Update value network parameters ϕ
    Set $s \leftarrow s'$
  end while
end for

4 Results

4.1 Training Reward Analysis

To evaluate the performance of different agents, we analyzed their average rewards over 5000 episodes under four distinct settings of the Frozen Lake environment. These settings vary based on grid size (4×4 or 5×5) and stochasticity (`is_slippery=True` or `is_slippery=False`). Below, we present the results for each setting.

4.1.1 Grid Size = 4×4, `is_slippery=False`

In this configuration, where the environment has deterministic transitions, the following observations were made:

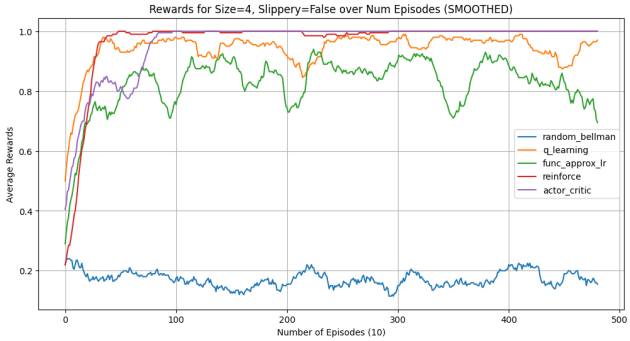

Figure 2: Rewards for Grid Size = 4×4, `is_slippery=False`.

- **Actor-Critic and REINFORCE:** Both agents converged the fastest, achieving an average reward of approximately 1.0 early in training and maintaining this performance consistently.
- **Q-Learning:** This agent performed slightly better than the Function Approximation agent, achieving an average reward of 0.9 compared to 0.8.
- **Function Approximation:** Although slower than Q-Learning, this agent demonstrated stable performance, converging to 0.8 by the end of training.
- **Random Bellman:** As expected, this baseline agent performed the worst, maintaining an average reward of 0.2.

4.1.2 Grid Size = 4×4, `is_slippery=True`

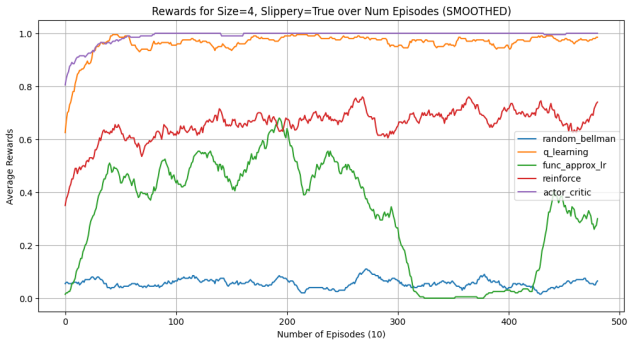

Figure 3: Rewards for Grid Size = 4×4, `is_slippery=True`.

Under stochastic conditions, where the agent's movement is no longer deterministic, the following trends were observed:

- **Actor-Critic and Q-Learning:** Both agents maintained strong performance, achieving average rewards close to 1.0. Actor-Critic performed slightly better overall.
- **REINFORCE:** While this agent initially performed well, its average reward dropped to 0.7 under stochastic conditions. It still outperformed Function Approximation.
- **Function Approximation:** This agent exhibited an unusual performance pattern, with rewards dropping from 0.5 to nearly 0 and later recovering to 0.4 toward the end of training.
- **Random Bellman:** The agent performed poorly, maintaining rewards around 0.1, slightly worse than in the deterministic case.

4.1.3 Grid Size = 5×5, `is_slippery=False`

In a larger, deterministic environment, we observed the following:

- **Actor-Critic:** This agent achieved the best performance, converging quickly to an average reward of 1.0 and maintaining this throughout training.
- **Q-Learning, REINFORCE, and Function Approximation:** These three agents performed comparably, achieving average rewards around 0.65 for Q-Learning and 0.5 for both REIN-

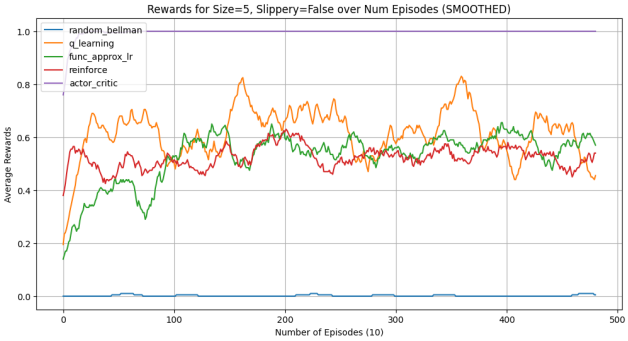

Figure 4: Rewards for Grid Size = 5×5, `is_slippery=False`.

FORCE and Function Approximation. However, their training curves exhibited significant fluctuations, especially Q-Learning.

- **Random Bellman:** The performance of this agent dropped significantly in the larger grid, with rewards close to 0.

4.1.4 Grid Size = 5×5, `is_slippery=True`

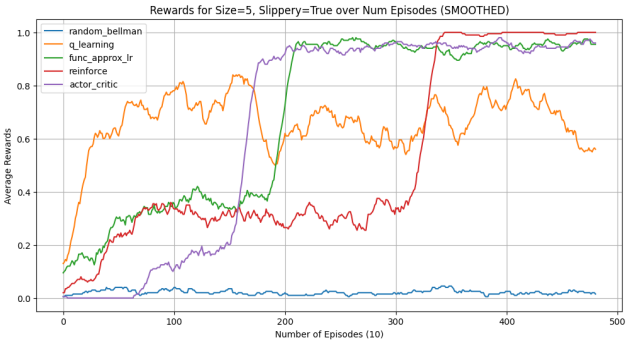

Figure 5: Rewards for Grid Size = 5×5, `is_slippery=True`.

In the most complex scenario, involving both a larger grid and stochastic transitions, the following outcomes were observed:

- **Actor-Critic, REINFORCE, and Function Approximation:** All three agents converged to near-optimal performance (1.0 for REINFORCE and 0.95 for Actor-Critic and Function Approximation). REINFORCE converged slightly later than the others but achieved the highest reward.
- **Q-Learning:** This agent exhibited fluctuating rewards between 0.6 and 0.8 throughout training, struggling to stabilize.
- **Random Bellman:** As with other settings, this agent performed poorly but showed slight improvements compared to 4.1.3.

4.2 Evaluation Reward Analysis

In addition to the training rewards, we analyzed the evaluation rewards, which was averaged over 100 iterations of the trained model. In Table 1, Q-Learning demonstrated exceptional performance on the smaller grid size (4×4) across both slippery and non-slippery settings, consistently achieving optimal rewards of 1. The Approximate Function LR agent showed strong adaptability to stochastic environments (`is_slippery=True`), performing relatively better than other agents in such conditions. Among all the models, Actor-Critic was the most stable and reliable performer, maintaining relatively high rewards across all four configurations, regardless of grid size or stochasticity.

Size	Slippery	Random Bellman	Q-Learning	Approx. Function LR	Reinforce	Actor-Critic
4	False	0.14	1.0	0.89	1.0	0.99
4	True	0.08	1.0	0.92	0.64	0.92
5	False	0.00	0.71	0.54	0.5	0.97
5	True	0.01	0.53	0.97	1.0	0.71

Table 1: Evaluation Rewards Averaged over 100 Iterations.

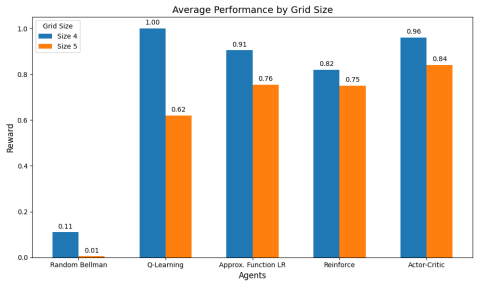

Figure 6: Average Performance by Grid Size

Figure 6 shows the average performance by grid size. When the grid size increases, all models showed a decline in performance. However, Reinforce and Actor-Critic exhibited the smallest performance decreases (12-13%), while Q-Learning experienced the largest drop (38%), followed by Approximate Function LR with a 15% decline.

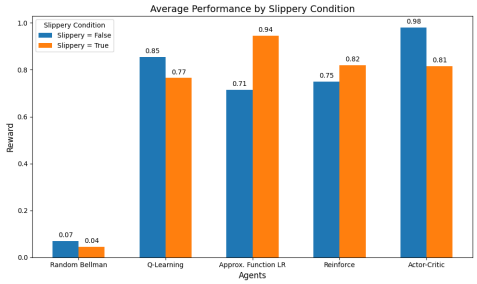

Figure 7: Average Performance by Slippery Condition

Figure 7 shows the average performance under slippery conditions. Q-Learning and Actor-Critic showed a decline in performance when the slippery condition was set to true. Interestingly, contrary to expectations that rewards would decrease for all agents, the other two models showed an overall improvement. Specifically, Actor-Critic exhibited a significant decrease in performance (17%), while Approximate Function LR demonstrated a noticeable increase (23%).

5 Conclusion

In this paper, we analyzed the performance of various reinforcement learning agents in the Frozen Lake environment under increasing complexity and randomness.

For Hypothesis 1, which proposed that function approximation methods and policy gradient algorithms would perform better than tabular Q-learning as grid size increases, the results were mostly consistent with expectations. Policy gradient methods, especially Actor-Critic, showed strong scalability in larger grid environments. In contrast, Q-Learning struggled to generalize, with its performance declining significantly, highlighting the limitations of tabular methods in handling larger state-action spaces.

For Hypothesis 2, which suggested that policy gradient methods would be more robust to stochastic environments (`is_slippery=True`), the results were mixed. While Actor-Critic and REINFORCE demonstrated adaptability, the function approximation method surprisingly excelled in the stochastic setting, outperforming the other agents. This suggests that generalization capabilities in function approximation can play a crucial role in handling randomness. Meanwhile, Q-Learning's performance declined under stochastic conditions, reinforcing its limited adaptability compared to the other methods.

In conclusion, the findings support the hypotheses that function approximation and policy gradient methods outperform tabular Q-learning in larger and more complex environments. However, the unexpected strength of function approximation in stochastic conditions highlights opportunities for further research into the interplay between generalization and robustness in reinforcement learning.

References

[1] Farama Foundation. (2023) Frozen Lake Environment. *Gymnasium Documentation: Frozen Lake*. Available at: `https://gymnasium.farama.org/environments/toy_text/frozen_lake/`. Accessed: 2024-12-13.

[2] Farama Foundation. (2023) Train an Agent. *Gymnasium Documentation: Train Agent*. Available at: `https://gymnasium.farama.org/introduction/train_agent/`. Accessed: 2024-12-13.

Appendix

Appendix A: Data for Average Performance of Agents

Size	Random Bellman	Q-Learning	Approx. Function LR	Reinforce	Actor-Critic
4	0.11	1.0	0.905	0.82	0.96
5	0.005	0.62	0.755	0.75	0.84

Table 2: Average Performance by Size

Slippery	Random Bellman	Q-Learning	Approx. Function LR	Reinforce	Actor-Critic
False	0.07	0.855	0.715	0.75	0.98
True	0.045	0.765	0.945	0.82	0.815

Table 3: Average Performance by Slippery Condition

Contribution

In our group, Sophia worked on the Random Bellman, Q-Learning, and Function Approximation agents, while Julie focused on REINFORCE and Actor-Critic. We collaborated on the introduction, problem, and conclusion sections, dividing the approach and results based on our assigned agents. Together, we brainstormed ideas, conducted research, and prepared the slides and video to ensure a cohesive final project.