Guiding Agents under Uncertainty: A Machine Learning Strategy for FrozenLake-v1

S 19 323
A. M. H. S. T. Bandara
DSC4173/CSC4173 - Machine Learning
Department of Statistics and Computer Science
Faculty of Science
University of Peradeniya

Table Of Contents

1.INTRODUCTION

Reinforcement learning is an area of machine learning that focuses on how agents can make a sequence of decisions to achieve a goal within an uncertain environment. The FrozenLake-v1 environment, available through the Gymnasium Python library, is a classic example for this purpose.

Figure 1. FrozenLake-v1 4x4 grid

It consists of a grid-based map that represents a frozen lake with holes, where the agent must navigate from a starting state to a goal state. The challenge lies in the slippery surface, which makes actions stochastic — the agent may not always move in the intended direction. This project aims to explore how data-driven strategies can guide the agent to achieve its goal more effectively by collecting experience data, training a predictive model, and designing a decision-making strategy based on that model by balancing exploration and exploitation.

2.PROBLEM STATEMENT

The primary goal of this project was to develop, test, and refine a strategy to enhance an agent's capability to navigate successfully within the FrozenLake-v1 environment. To achieve this, the project focused on estimating the state-action value function Q(s,a) through the application of Q-learning, a reinforcement learning technique that facilitates an agent's learning through its interactions with the environment. By utilizing the derived Q-table, the agent is directed to make decisions that optimize its chances of reaching the designated goal while avoiding pitfalls. The effectiveness of this guided agent is subsequently assessed and compared to that of a random agent, providing a measure of the improvements obtained through the learning process. Furthermore, the project examines the potential benefits of balancing exploration and exploitation to further enhance the agent's ability to identify an optimal path amid the uncertainties presented by the slippery surface.

3.METHODOLOGY

3.1 Environment Setup

The Gymnasium FrozenLake-v1 environment with a 4×4 grid serves as the testbed. The agent starts in one corner and must reach the goal in the opposite corner while avoiding holes. Due to the slippery surface, intended actions may not always result in expected state transitions.

3.2 Data Collection

A total of 10,000 episodes were run using a random agent to collect data.

For each step, we recorded:

- State: The position before the action.
- Action: The chosen action.
- Reward: Immediate reward (1 for goal, 0 otherwise).
- Goal Proximity: How far the agent is from the goal.
- Total Reward: Whether the episode reached the goal.

This dataset provides a baseline for performance comparison.

3.3 Goal Proximity Calculation

At each step, the Manhattan distance to the goal was calculated to serve as a simple measure of proximity, helping to analyze how state location affects the likelihood of reaching the goal.For a given state sss, the row and column positions were determined, and the Manhattan distance was computed as:

$$D(s) = |xgoal - xs| + |ygoal - ys|$$

where (xs, ys) represents the agent's current position and $(xgoal, ygoal)$ is the goal state's position on the grid. In the FrozenLake 4×4 grid, states are numbered 0 to 15. So the position (xs, ys) is computed as:

$$xs = state \;//\; 4 \quad , \quad ys = state\%4$$

with the goal position being (3,3)

3.4 Data Storage and Analysis

The collected data was stored in a Pandas DataFrame. This included columns for State, Action, Reward, Goal Proximity, Total Reward.

3.5 Success Rate of Random Agent

By counting episodes where the agent reached the goal, we calculated the random agent's baseline success rate. For Frozen Lake, this rate is typically low due to stochastic moves.

4.PREDICTIVE MODEL DEVELOPMENT

4.1 Importance Metric Definition

The Importance of an action was defined as the total reward obtained in its episode, indicating whether that action contributed to ultimately achieving the goal.

4.2 Machine Learning Model Training

A Random Forest Regressor was trained to predict the Importance value for any (state, action) pair, given the current goal proximity as an input feature. This model learns to approximate the expected value of an action in a specific state context, effectively serving as a simple predictive policy. The model was trained using scikit-learn's implementation, leveraging multiple cores for faster training.

The training data was prepared by combining three features:

- The **state** the agent is in.
- The **action** the agent takes.
- The **goal proximity**, calculated using the Manhattan distance from the agent's position to the goal.

The target label for training was the Importance metric, which was defined using the episode's Total_Reward in the dataset collected from random actions. While this is a coarse label, it serves to provide a basic mapping that the model can learn from.

A Random Forest was chosen for its robustness, ability to capture non-linear relationships, and its ease of interpretation for this tabular problem.

5.AGENT DESIGN AND IMPLEMENTATION

5.1 Guided Agent Strategy

Once the Random Forest model was trained, it was used to guide the agent's decision-making process. The key idea is that, at each step, the agent must choose the action that maximizes the predicted Importance value.

For every possible action in the action space (in FrozenLake, actions are {0: left, 1: down, 2: right, 3: up}), the agent:

- Computes the goal proximity for the hypothetical next state.
- Uses the trained Random Forest to predict the expected Importance for that combination.
- Selects the action with the highest predicted value.

To reduce redundant predictions, a simple caching mechanism was implemented: predicted values for combinations are stored and reused whenever the same combination occurs again. This speeds up policy execution during large-scale evaluation.

The resulting strategy is purely **greedy** — it always selects the action with the maximum predicted value without introducing any randomness.Here, the agent fully **exploits** the knowledge captured by the model.

5.2 Evaluation of Guided Agent

To evaluate how well the guided agent performs, the policy was run for 1,000 episodes in the FrozenLake environment. At each step, the agent used the greedy selection strategy to pick its next move. The environment's stochastic nature means that even with an optimal prediction, the agent might slip into a hole or miss the goal due to unintended moves.

The agent's performance was measured by recording the total reward for each episode (1 if the goal was reached, 0 otherwise). The proportion of successful episodes provided a quantitative measure of the improvement over the random baseline. The notebook output confirmed that the guided agent's success rate was consistently higher than that of the purely random agent, demonstrating the benefit of using a learned predictive model.

5.3 Improved Agent with Exploration–Exploitation

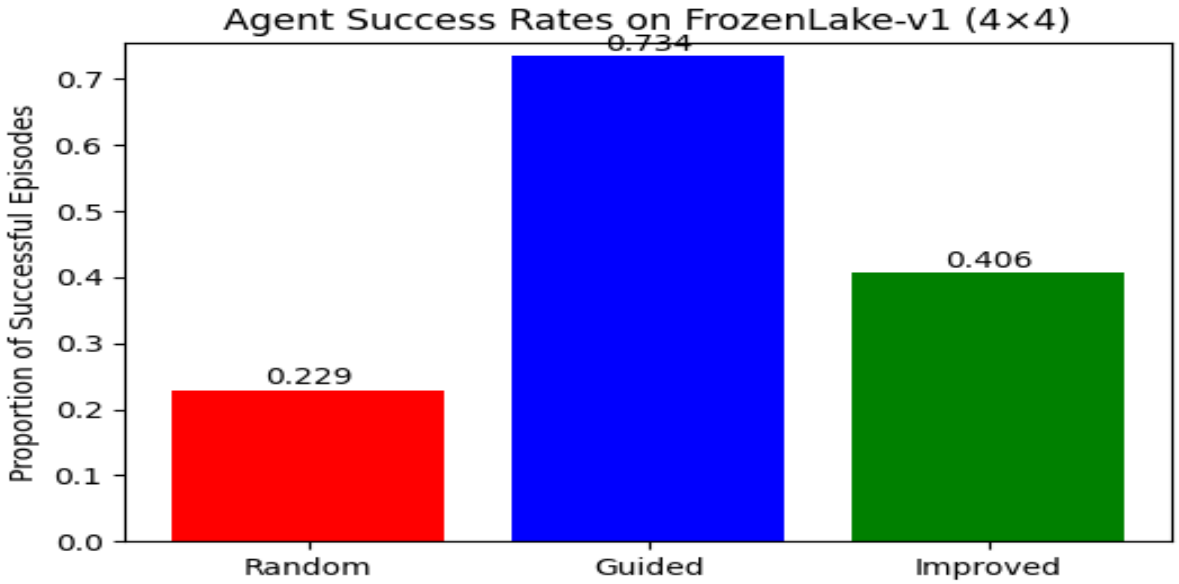
While a purely greedy agent can perform well in deterministic environments, the FrozenLake grid's slippery nature makes it easy for the agent to get stuck in local patterns or suboptimal paths. To address this, an **ε-greedy strategy** was introduced as an improvement.

The ε-greedy agent modifies the greedy policy by occasionally selecting a random action with probability ε (epsilon), and otherwise using the guided policy. ε was set to 0.1, meaning there is a 10% chance of exploring a random action and a 90% chance of exploiting the best-known action according to the predictive model.The improved agent was evaluated under the same conditions as the guided agent: 1,000 episodes with success rates calculated for direct comparison.

Next, rather than using a fixed exploration rate, a dynamic ε decay schedule was applied: the agent begins with a high exploration probability, encouraging it to sample different actions in the early episodes when its knowledge is limited. As training progresses, ε gradually decays toward a lower bound, shifting the agent's behavior toward exploiting the best-known actions according to the predictive model. This approach helps the agent discover potentially better paths early on, while reducing unnecessary random moves once a stable policy has emerged. The improved agent using this dynamic ε-greedy method was evaluated under the same conditions as the guided agent—over 1,000 episodes with a maximum step limit per episode—to compare success rates and examine how controlled exploration affects overall performance in a stochastic environment.

6.RESULTS

The bar plot comparing agent success rates showed that the Guided Agent achieved the highest success rate of approximately 0.73, demonstrating the benefit of always exploiting the best-known action as predicted by the model. The Improved Agent, which used an ε-greedy approach with 10% random exploration, achieved a lower success rate of about 0.41 (The success rate of the improved agent using the dynamic ε-greedy method was 0.3213, hence considered the improved model with higher success rates among these two models) . The Random Agent remained the baseline with a much lower success rate of about 0.23. This pattern indicates that the learned model significantly improves navigation compared to random actions, while additional exploration can reduce performance in this simple environment.



7. DISCUSSION

The results highlight that for a small, well-defined state space like FrozenLake-v1, a purely greedy policy based on a learned predictive model can outperform strategies that include additional exploration. Although the ε-greedy approach is theoretically beneficial in larger or more complex environments, in this case, the environment's stochastic transitions already introduce enough randomness. As a result, forcing extra exploration often leads the agent into suboptimal paths or terminal holes, lowering the overall success rate. This demonstrates that the effectiveness of exploration–exploitation strategies depends on the nature of the environment and the robustness of the learned policy.

8.CONCLUSION

This mini project successfully demonstrated a simple yet effective method for guiding an agent in the FrozenLake-v1 environment using a data-driven strategy. By collecting interaction data and training a Random Forest model to approximate the importance of state-action pairs, the agent learned to make better-informed decisions compared to taking random actions. The guided agent, which always selected the predicted best action, achieved the highest success rate, clearly outperforming the random agent baseline. Interestingly, the ε-greedy enhancement, which added deliberate exploration, did not improve performance in this case. This outcome highlights that in small, highly stochastic environments, adding exploration can lead to unnecessary risk of falling into holes or deviating from the optimal path. Overall, the project confirms that simple predictive models can significantly boost success rates in well-defined reinforcement learning tasks when appropriate exploitation strategies are used.

9. FUTURE WORK

Future extensions include:

- Testing the learned strategy on larger or more complex grid maps with different hole configurations to evaluate generalization performance.
- Replacing the Random Forest model with a tabular Q-learning approach or a Deep Q-Network (DQN) to handle larger state spaces.
- Benchmarking alternative reinforcement learning algorithms, such as SARSA, Monte Carlo control, or Policy Gradient methods, to compare learning stability and performance in similar stochastic settings.

10. REFERENCES

Johnny Code. (2023, June 25). *Q-Learning Tutorial 1: Train Gymnasium FrozenLake-v1 with Python Reinforcement Learning* [Video]. YouTube. https://www.youtube.com/watch?v=ZhoIgo3qqLU1

R&D World. (2023, August 4). *Slip and slide into reinforcement learning with the Frozen Lake challenge*. https://www.rdworldonline.com/slip-and-slide-into-reinforcement-learning-with-the-frozen-lake-challenge/2

Farama Foundation. (n.d.). *Gymnasium documentation*. https://gymnasium.farama.org/

FareedKhan-dev. (2023). Reinforcement-Learning-on-Frozen-Lake-v1-openAI-gym [GitHub repository]. GitHub. https://github.com/FareedKhan-dev/Reinforcement-Learning-on-Frozen-Lake-v1-openAI-gym1

moripiri. (2023). Reinforcement-Learning-on-FrozenLake [GitHub repository]. GitHub. https://github.com/moripiri/Reinforcement-Learning-on-FrozenLake

DataCamp. (2022, October 27). An introduction to Q-learning: A tutorial for beginners. https://www.datacamp.com/tutorial/introduction-q-learning-beginner-tutorial