**[FY22 AI School] Project Report**

*[Note: Please make sure to summarize the critical information only.* ***No more than 5 pages*** *in this document. No more than 3 pages are preferred] [Replace the lines in square brackets with your content in below chapters]*

**General Information (This is a mandatory chapter)**

* Project Name: Frozen Lake
* Area: AI + Game
* Group ID: G7
* Group members:

|  |  |  |
| --- | --- | --- |
| Alias | Microsoft email address | Role in the project |
| joshuachen | joshuachen@microsoft.com | Design and implement algorithm |
| qiso | qiso@microsoft.com | Documentation |
| xingfan | xingfan@microsoft.com | Algorithms & document |
| xinglinyu | xinglinyu@microsoft.com | Design and implement algorithm |

**Overview**

[Frozen lake](https://www.gymlibrary.ml/environments/toy_text/frozen_lake/) is an environment of AI Gyms. involves crossing a frozen lake from Start(S) to Goal(G) without falling into any Holes(H) by walking over the Frozen(F) lake. The agent may not always move in the intended direction due to the slippery nature of the frozen lake. That is, if you go in a certain direction, there is only 0.333% chance that the agent will really go in that direction, which means the agent is uncertain and only partially depends on the chosen direction.

We tried in total four methods on this game. They are policy iteration, value iteration, Q-learning, Double SARSA and Double Q-learning. We took the final policy of each algorithm and tested them in 10000 times tries separately. Finally, we compared the success rate (the higher the better) as well as the training episodes needed (the lower the better) to convergence of them.

The best success rate we achieve is **56.8%** using the **double SARSA**. It took about 6500 episodes to converge.

The final output was included in a Jupyter notebook, named as

1. AICourseFY22-FrozenLake-Joshua-Double-Sarsa
2. AICourseFY22-FrozenLake-Joshua-Double-QLearning
3. AICourseFY22-FrozenLake-Xinglin-Value\_Policy\_Iteration.ipynb

**Motivation**(This is an optional chapter)

Among the two games given, we chose Frozen Lake as our final project. Compared with Blackjack game, we think that Frozen Lake leaves more space to try different settings and practice reinforcement learning algorithms.

We were all almost new to AI Gym and reinforcement learning. We took this project as good practice, so we focused on comparing and benchmarking RL algorithms instead of fine tuning a specific method.

**Project Details**

1. **Prerequisites**

Gym version should be 0.25.0, as older version (like 0.18.0) only includes FrozenLake\_v0.

1. **Analysis**

About the Frozen Lake Env, let’s first do an env analysis.

A picture containing graphical user interface

Description automatically generated

Fig.1 Frozen Lake Env Analysis

This Fig.1(a) shows 2 paths. Let’s see which one should be the optimized path.

Due to the slippery property, the agent will move in the intended direction with probability 1/3 or move in either perpendicular direction with equal probability 1/3 in both directions.

- Like at state `S0`, agent takes down action (a=1), it may arrive at `S4` with probability 1/3, or arrive at `S1` with probability 1/3, or stay at `S0` with probability 1/3.

- Like at state `S6`, agent takes down action (a=1), it may arrive at `S10` with probability 1/3, or fall into Holes (`S5` or `S7`) with probability 2/3.

Actually, if at the state `S6`, no matter what action the agent takes, it will fall into Holes with probability at least 1/3. If at state `S10`, if any of up/right/down action is taken, the agent will fall into Holes with probability 1/3.

These two states are risk states, the agent should avoid them.

So, comparing Path2 with 2 risk states `S6` and `S10`, Path1 would be more optimal. Path1 has state sequences [S0, S4, S8, S9, S13, S14, S15].

- At `S0`, we expect the agent arrive at next state `S4` without `S1`, so a left action (a=0) is suitable.

- At `S4`, we expect the agent arrive at next state `S8` without `S5`, so a left action (a=0) is suitable.

- At `S8`, we expect the agent arrive at next state `S9` without `S12`, so a up action (a=3) is suitable.

- At `S9`, we expect the agent arrive at next state `S13` without `S5`, so a down action (a=1) is suitable.

- At `S13`, we expect the agent arrive at next state `S14` without `S12`, so a right action (a=2) is suitable.

- At `S14`, we expect the agent arrive at next state `S15` without `S10`, so a down action (a=1) is suitable.

- At `S15`, we expect the agent arrive at next state `S15` without `S11`, so a down action (a=1) is suitable.

So according to the slippery property, we infer to the optimized policy with Path1: [0, 0, 3, 1, 2, 1, 1].

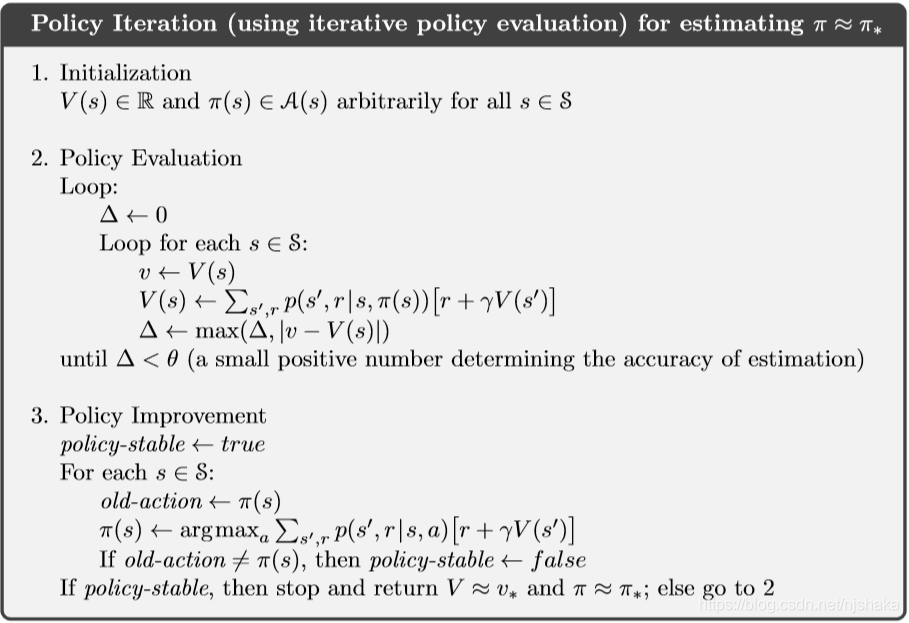
Moreover, if we define a policy function Pi(s) which means an action the agent should take in state s, the optimized policy should be

`Pi(S)=[Pi(S0)=0, Pi(S1)=3, Pi(S2)=3, Pi(S3)=3, Pi(S4)=0, Pi(S5)=?, Pi(S6)=0, Pi(S7)=?, Pi(S8)=3, Pi(S9)=1, Pi(S10)=0, Pi(S11)=?, Pi(S12)=?, Pi(S13)=2, Pi(S14)=1, Pi(S15)=0]`, like Fig.2 (c), where `?` means no action available because the agent has fallen into holes.

So, our work is to find this optimized policy by interaction with frozen lake environment.

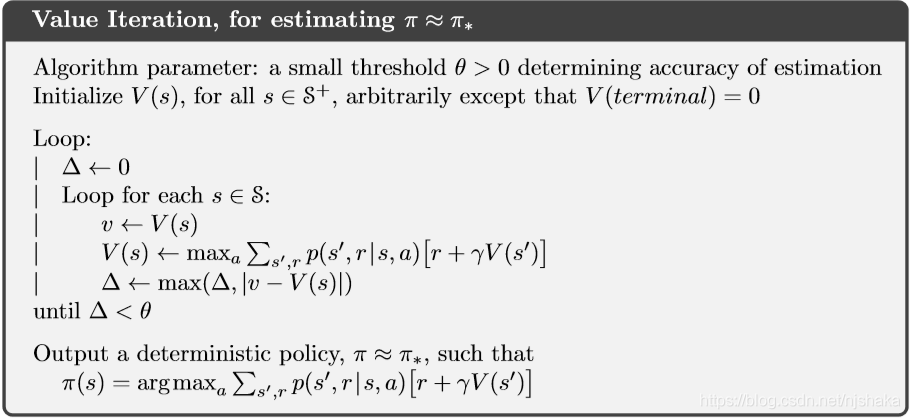
1. **Solutions**

* Policy Iteration



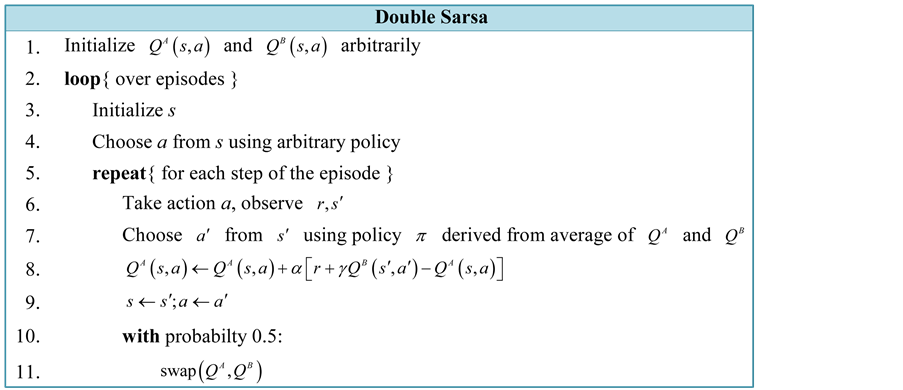
With policy iteration, we get an initial policy (like randomly take an action in action space). Then we calculate the value for every state and the expected state-action value. And later we check if it has been converged, or we will update the policy.

* Value Iteration



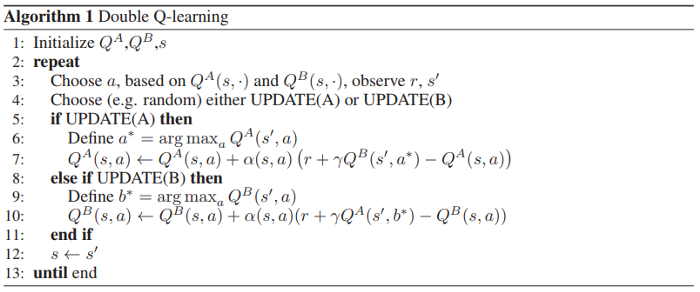
For a particular state, we first calculate the state-action values for all the possible actions from that state, and then update the value function of that state with the greatest state-action value. The Value Iteration terminates when the difference between all the new State values and the old State Values is a negligibly small value (convergence).

* Double – SARSA



Different from general SARSA, we try double-sarsa. Like the general SARSA, we also update the Q-value based on the action/learning rate/policy. But we have two Q-tables (in the code they are Q1 and Q2). First, make a prediction. Then calculate the target based on the reward/prediction/Q2 (with next obs and next action)/gamma. And then update Q1 based on the previous result and learning rate. By decoupling the two tables, we can have a more stable result. In our implementation, the learning rate is set to 0.1. Gamma is set to 0.9. e\_greed is set to 0.15.

* + Double Q-Learning



As for the double SARSA, we just follow the general practice. The main difference between this one and the double SARSA is the way we calculate the target and the way to update the table. In the code, we also have two tables named Q1 and Q2. We calculate the target using gamma times

max(Q2[next\_obs, :]). Then update Q1 based on the previous value and the learning rate. In our implementation, the learning rate is set to 0.1. Gamma is set to 0.99. e\_greed is also 0.1.

1. **Results/Conclusions**

**8x8 Frozen Lake**

|  |  |  |  |
| --- | --- | --- | --- |
| Algorithm Name | Train Convergence Episode | Test Episodes | Test Success Rate |
| Policy Iteration | 13 | 100000 | 51.2% |
| Value Iteration | 2357 | 100000 | 47.8% |
| Double SARSA | 6500 | 100000 | 56.8% |
| Double Q-Learning | Can’t converge | 100000 | 7.5% |

**Future works**

We didn’t fine tune our algorithms as the game itself was simple. However, we all think that should be a part that deserves more attention.

**References**

1. [Solving the FrozenLake environment from OpenAI gym using Value Iteration | by Diganta Kalita | Analytics Vidhya | Medium](https://medium.com/analytics-vidhya/solving-the-frozenlake-environment-from-openai-gym-using-value-iteration-5a078dffe438)
2. [强化学习实战（一）：用值迭代和策略迭代解决Frozen Lake问题\_MADRLer的博客-CSDN博客](https://blog.csdn.net/njshaka/article/details/89237941)
3. [Double Sarsa and Double Expected Sarsa with Shallow and Deep Learning (scirp.org)](https://www.scirp.org/Journal/PaperInformation.aspx?PaperID=71237)
4. [Double Q-Learning, the Easy Way. Q-learning (Watkins, 1989) is… | by Ziad SALLOUM | Towards Data Science](https://towardsdatascience.com/double-q-learning-the-easy-way-a924c4085ec3)