fisherman agent

The fisherman agent selects and act as one agent based on the multi-armed bandit where the successive agents are “arms”. In this way, we aimed to be able to play with the agent who seems to have the strongest winning rate in the game of 100 times.

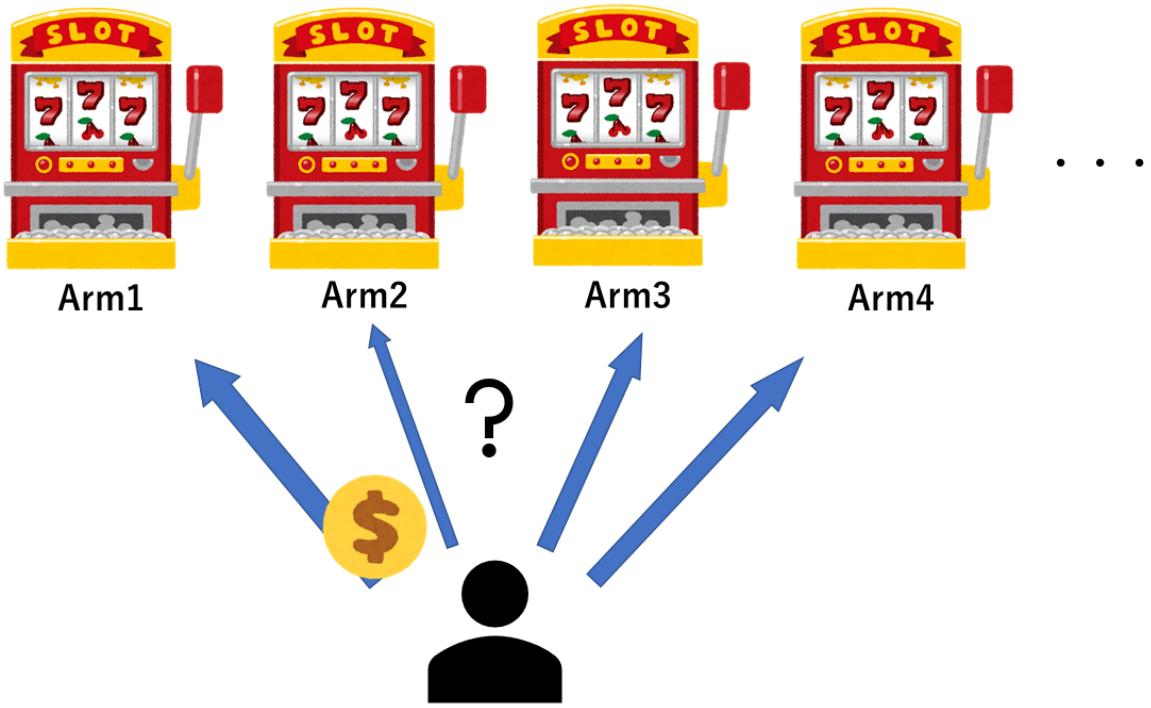
We utilized 2017/cndl team and 2017/kasuka team in this tournament as arms.

Idea

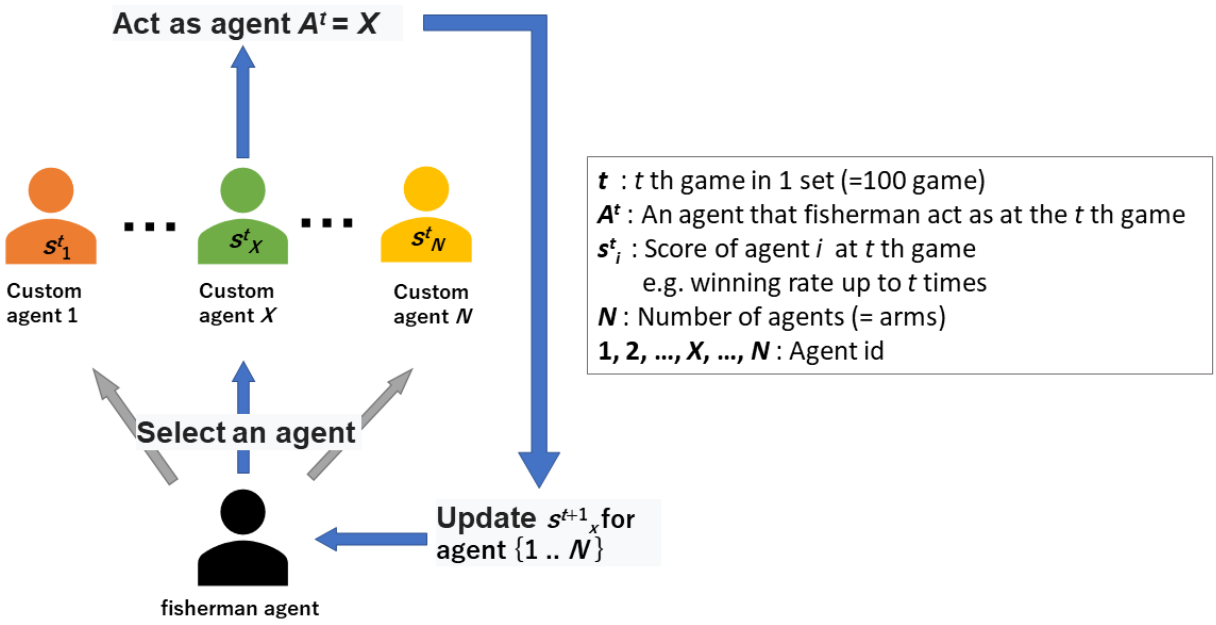
Multi-armed Bandit (MAB) problem

> In the problem, each machine provides a random reward from a probability distribution specific to that machine. The objective of the gambler is to maximize the sum of rewards earned through a sequence of lever pulls. The crucial tradeoff the gambler faces at each trial is between "exploitation" of the machine that has the highest expected payoff and "exploration" to get more information about the expected payoffs of the other machines. The trade-off between exploration and exploitation is also faced in machine learning. In practice, multi-armed bandits have been used to model problems such as managing research projects in a large organization like a science foundation or a pharmaceutical company. In early versions of the problem, the gambler begins with no initial knowledge about the machines.

(Wikipedia)



Applying to AIWolf agent



How to use

Class file path

`com.gmail.fishing.village.FishermanPlayer`

Environment

- Java8
- AIWolf platform ver0.5.6
- Agent Jar file(s) built with Java8 (e.g. 2018/cndl, 2018/Rsaito, ...)

Configuration

1. Building Jar file that you want to use

Using fisherman, agent jar files are required. To setup agents, plese refer to each document.

2. Embedding agents

In `FishermanPlayer.java#setModelList`, set agent instances and add them to array named `modelList`.

3. Configuration of bandit algorithm

To select agent, Fisherman has 3 bandit algorithms: ε-Greedy, A/B test, UCB1. Fisherman requires one selector to play. Selctor methods are defined in `ModelSelector.java`.