Contents

1 Introduction

In the provided exercise we were asked to create different strategy for a Tic-tac-toe environment, and two example were provided one a medium blog another a Kaggle notebook, and a sample code working with OpenAI gym environment. With the flexibility option of creating new strategy and exploring different option. My solution choice was to implement Thompson Sampling algorithm, which is a popular probabilistic algorithm used in decision making under uncertainty, particularly in the context of multi-armed bandit problem (Karn, 2023).

Thompson Sampling tackles the multi-armed problem by applying a Bayesian approach. It maintains a probability distribution for each machines expected reward representing our uncertainty about the true reward distribution. At each round the algorithm sample from these distributions and selects the machine with highest sample as the action to take (Bismi, 2023).

2 Structure of the program

- Exercise 1
  - Environment
    - Board.py
    - Player.py
  - Main.py

The project folder contains Board.py, Player.py and main.py where board.py has the Tic-tac-toe environment as show on the medium post and the Player.py file has player class and all the player behavior defined in one file and implementation on main.py file.

The initial idea behind the project was to build a strategy based on the provided code sample, but reading OpenAI gym documentation I really wanted to explore the Tic-tac-toe environment as well, so I started with a basic 3x3 board with help of a python library numpy, which really great for multi-dimensional arrays and matrices along with a large collection of mathematical function. With all the variable calculated in my mind all of the possible features and function were implemented such as winner checker function, function to check for available position, function to render the board and almost everything that are required for a Tic-Tac-Toe game.

As for a player class where we have implemented the Thompson sampling technique with binary rewards, first we initialize two array alpha and beta to accommodate failure and success. The reward for each action follows beta distribution (Morales, 2022). The algorithm sample from a

beta distribution for each action, chooses the action with the highest sample, observes the reward and updates the corresponding beta distribution, a function for saving model and loading the model is implemented with help of the pickle library. A Human Agent class is also implemented which inherits from Player class, for a human to fight

2.1 Breakdown of the algorithm

Thompson Sampling algorithm initializes a prior distribution for each machines reward. In each round algorithm samples a reward value from each machines distribution. The machine with the highest sampled value is selected, after the action is selected, algorithm observe the actual reward obtained from that machine. Using the observed result algorithm updates the probability distribution for selected machine. The updated distribution becomes the prior distribution for next round, and the process repeat until the algorithm balance exploration and exploitation.

Reason behind choosing the algorithm, an implemented library in python, wide range of internet article supporting and giving tutorial about the implementation.

2.1.1 Findings

While implementation the agent (Player 1) keeps exploiting the same move to win and there is less sign of exploration and reward as seen in the graph, there is couple of different wins in Column 2 but maximum number of winning are from exploiting Column one again and again, as for the win in Rows there are Four winning move in Row one and One Wining move in Row Two and this output is from 50000 number of training. As for winning agent Player one has maximum number of Wins.

```
fig, ax = plt.subplots()
label = ["Col 1", "Col 2", "Col 3"]
color = ['tab:red', 'tab:blue' , 'tab:orange']

bar_labels = ['red', 'blue', 'orange']
ax.bar(label, winsCol, label=bar_labels, color=color)
ax.set_ylabel('Number of Traning Cycle')
ax.set_title('Tic Tac Toe')
ax.legend(title='Game')

plt.show()
```

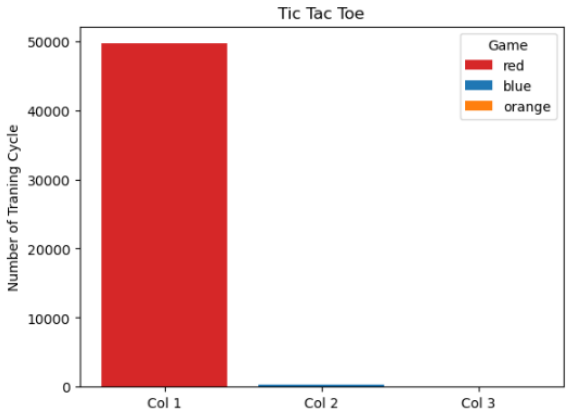


Figure 1 Numbers of Wins - Columns

```
fig, ax = plt.subplots()
label = ["Row 1", "Row 2", "Row 3"]
color = ['tab:red', 'tab:blue' , 'tab:orange']

bar_labels = ['red', 'blue', 'orange']
ax.bar(label, winsRow, label=bar_labels, color=color)
ax.set_ylabel('Number of Training Cycle')
ax.set_title('Tic Tac Toe')
ax.legend(title='Game')

plt.show()
```

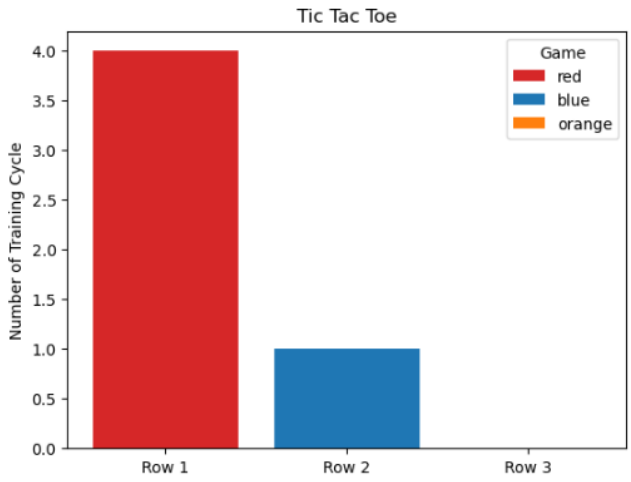


Figure 2 Numbers of Win - Rows

```
plt.figure(figsize=(10, 6))
plt.plot(p1_wins, label="Player 1 Wins", color="blue")
plt.plot(p2_wins, label="Player 2 Wins", color="red")
plt.plot(draws, label="Draws", color="green")
plt.xlabel("Number of Games")
plt.ylabel("Cumulative Wins/Draws")
plt.title("Thompson Sampling: Cumulative Wins/Draws Over Time")
plt.legend()
plt.show()
```

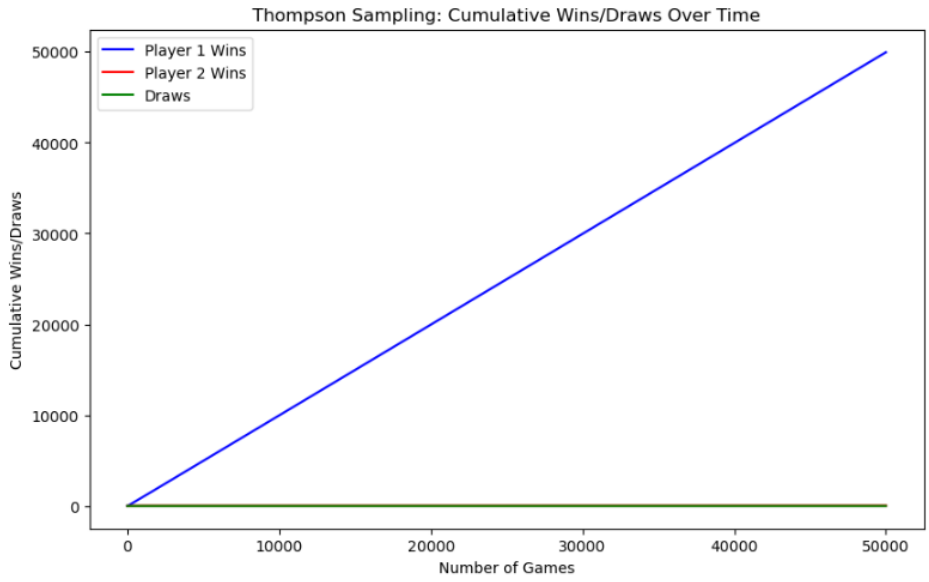


Figure 3 Calculating Player Wins

3 Access the Program

The main file has all the implementation of the classes where we can pass the argument.

For the training there are two argument that are taken in consideration.

--training 1, argument training 1 initialize turn on with 1, which start the training process.\

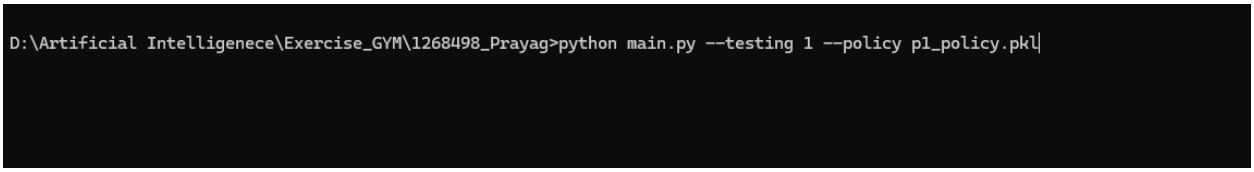
--epoch 10000, argument epoch symbolizes the number of training loops

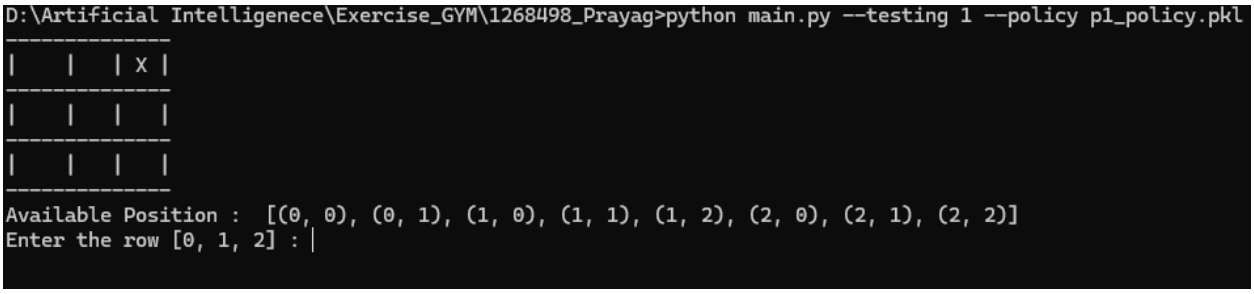
```
D:\Artificial Intelligenece\Exercise_GYM\1268498_Prayag>python main.py --training 1 --epoch 100000
```

For the testing there are two argument which are taken in consideration.

---testing 1, argument testing 1 initialize turn on with 1, which start the testing process.

--policy name of the file, argument takes the file path where we save the model and it loads up the human agent who can play with trained agent.





4 References

Bismi, I. (2023, 07 26). *Thompson Sampling: A Powerful Algorithm for Multi-Armed Bandit Problems*. Retrieved from Medium: https://medium.com/@iqra.bismi/thompson-sampling-a-powerful-algorithm-for-multi-armed-bandit-problems-95c15f63a180

Karn, A. R. (2023, 07 20). *Thompson Sampling - Python Implementation*. Retrieved from Medium: https://medium.com/@ark.iitkgp/thompson-sampling-python-implementation-cb35a749b7aa

Morales, R. (2022, 08 13). *The beta-binomial model: an introduction to Bayesian statistics*. Retrieved from Medium: https://medium.com/@romoflow/the-beta-binomial-model-an-introduction-to-bayesian-statistics-154395875f93