Back

Congrats Daniel! This project has been marked as completed.

Project Rating

★★★★☆

Teacher's Comment

"GOOD WORK"

-

Was this helpful?

Community Link

Publish to Community

Edit Your Project

Last Submitted

Previous Submissions

3rd Mar 2024 Open Link

Start Project

PRO-C126: ARCHERY PLAYER 3 Completed

In Class 126, We Learned How To Train Agent Using The Rl Model. In This Project, We Are Going To Train The Agent To Play Archery And Hit The Center Of The Target.

Goal of the Project:

In class 126, we learned how to train agent using the RL model. In this project, we are going to train the agent to play archery and hit the center of the target.

Story:

At the Chicago School of Artificial Intelligence, an archery competition is held for Robots. **Archery-Target** is a game in which the players shoot sharp-pointed arrows at a round target having 10 rings. Joseph has to train his robot to hit the Bull's eye in the very first attempt. He needs your help so that he can use the RL method to train the robot to play archery.

Can you help him to do so?

Project Template Output

RL Problem to Solve
Hit the center of the target with maximum reward
Number of **State**: ?
Number of **Actions**: ?

Expected Output

```
Total Episode Reward :  5489.999999999999
Ending Game Episode:  999
Average Total Reward : 5.489999999999999
```

***This is just for your reference. We expect you to apply your own creativity to the project.**

Note: This project is the continuation of **Project 125**. You can continue with the same project code. Previously we created the **Reward matrix** and defined the **take_action()** function to choose random actions.

Submit Your Project

Learn how to submit your project

Paste your project URL

Submit Project

Class Summary

This project is based on your last class PRO-C126

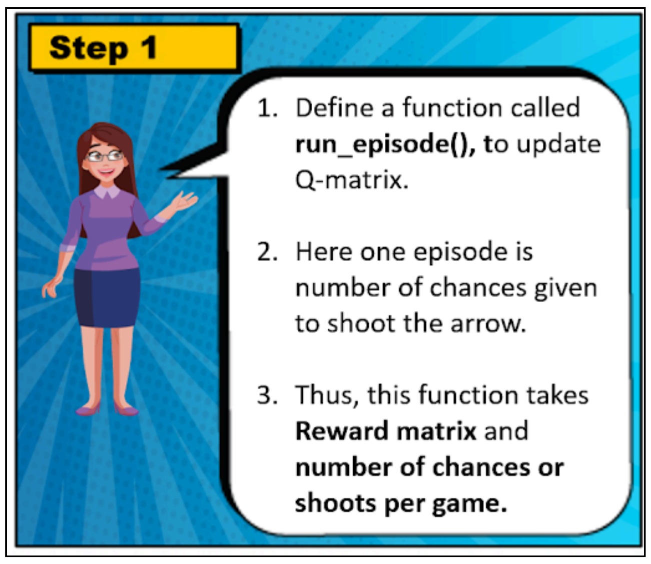
View Class Summary

Ask a doubt to your tea HELP

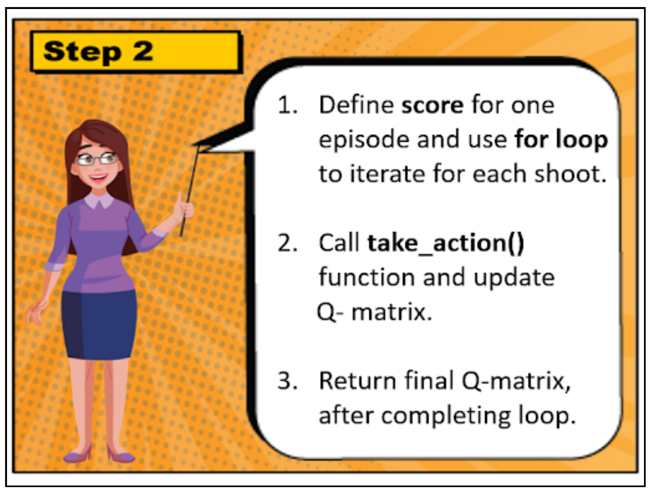
Getting Started:

1. Open the boilerplate link or you can continue with the previous project's Google Colab file.

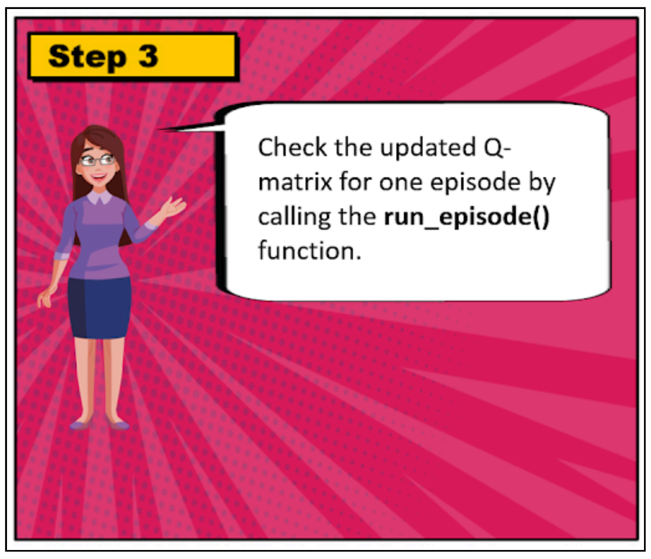
Specific Tasks to complete the Project:


Step 1
1. Define a function called **run_episode(),** to update Q-matrix.
2. Here one episode is number of chances given to shoot the arrow.
3. Thus, this function takes **Reward matrix** and **number of chances or shoots per game.**

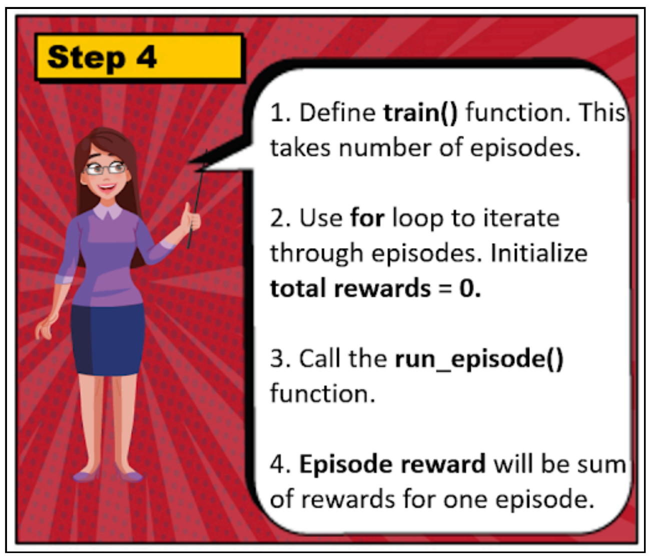
```
#Define run_episode() method

def run_episode(reward_matrix, shoot_per_game=5):
```


Step 2
1. Define **score** for one episode and use **for loop** to iterate for each shoot.
2. Call **take_action()** function and update Q- matrix.
3. Return final Q-matrix, after completing loop.

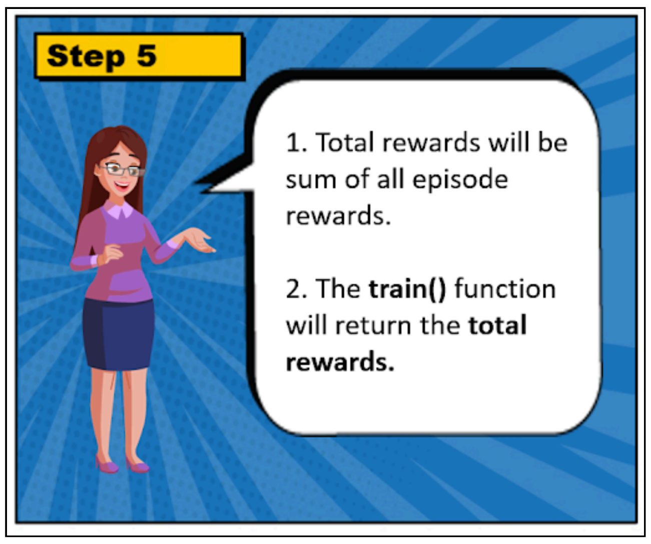
```
def run_episode(reward_matrix, shoot_per_game=5):

  score = 0

  #use for loop to iterate for number of chances

        #print shoot number

        #call take_action method to get the action

        #increase the score

        # print shoot number ends

  #Update Q-matrix

  #return updated Q-matrix
```


Step 3
Check the updated Q-matrix for one episode by calling the **run_episode()** function.

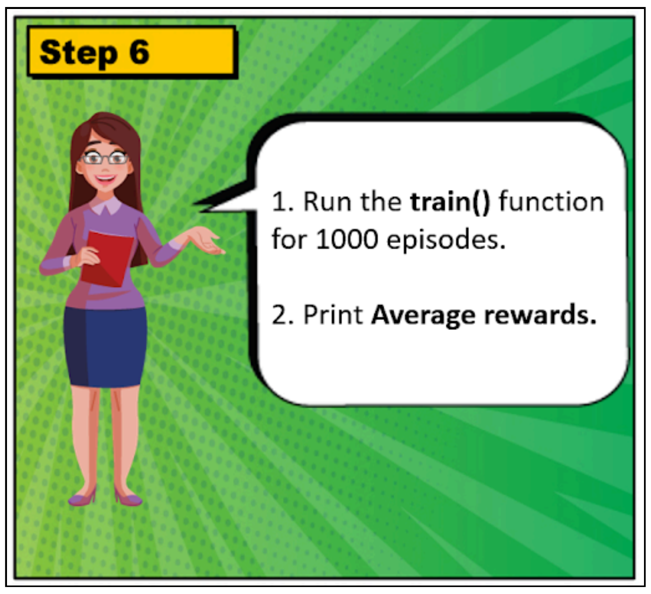
```
#Call run_episode method to check the final Q-matrix for one episode
```


Step 4
1. Define **train()** function. This takes number of episodes.
2. Use **for** loop to iterate through episodes. Initialize **total rewards = 0.**
3. Call the **run_episode()** function.
4. **Episode reward** will be sum of rewards for one episode.

```
#Define train() function

def train(episodes):
  #Use for loop to iterate through episodes

    #Initialize total_reward variable

    #Print 'episode start' with episode number

    #Call run_episode() method to get the q-matrix for one episode

    #Episode reward will be sum of all the rewards for one episode

    #print episode reward
```


Step 5
1. Total rewards will be sum of all episode rewards.
2. The **train()** function will return the **total rewards.**

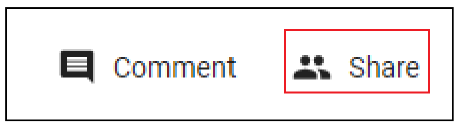
```
#Define train() function

def train(episodes):
  #Use for loop to iterate through episodes

    #Initialize total_reward variable

    #Print 'episode start' with episode number

    #Call run_episode() method to get the q-matrix for one episode

    #Episode reward will be sum of all the rewards for one episode

    #print episode reward

    #Total reward will be sum of all the episode reward


  #return total_reward
```


Step 6
1. Run the **train()** function for 1000 episodes.
2. Print **Average rewards.**

```
#Run the train() function for 1000 episodes


#Print Average reward
```

Submitting the Project:

1. **SAVE** all the changes made to the project.
2. Click on **"Run"** once to check if it is working.
3. Open GitHub and create a repository named **Project126**.
4. Click **Share**.



5. Click Change and choose the **'anyone with the link'** option.

Hints:

1. In step 4, take chances or **shoot per game = 5** and run the **run_episode()** function.
2. In step 6, the average reward will be the total **reward/number** of episodes.

Conclusion:

In this project, we checked the general performance of the simplest Reinforcement Learning problem. It is one state multiple actions problem, also known as the **"K-Armed Bandit"** problem.

One of the major use cases of this type of problem can be seen in selecting the right advertisement out of many to be displayed on the web page. The machine can be taught to pick the **best advertisement with the most user clicks!!**