Data Science Applications Reinforcement Learning Assignment Project

A simple and well designed structure is essential for any machine learning project, project template that combines **simplicity, best practice for CODE structure** and **good CODE design**. The main idea is that there's much same stuff you do every time when you start our machine learning project, so wrapping all this shared stuff will help you to change just the core idea every time you start our machine learning project.

So, here’s a simple readme template that help you get into our project faster and just focus on your notice and explainations, etc)

In order to decrease repeated code shunks, increase the time that can read the code in, flexibility an reusability we used a functional programming structure that focused on split all problems in our project in functions and use that functions many times in many places in the code without repeating the code.

Requirements

- numpy (The fundamental package for scientific computing with Python)
- scipy (Fundamental algorithms for scientific computing in Python.)
- gymlibrary (Gym is a standard API for reinforcement learning, and a diverse collection of reference environments.)
- ipython (IPython provides a rich architecture for interactive computing like parallel computing, GUI toolkits)

Functions

**** First Part ****

**** Turn this code into a module of functions that can use multiple environments **

1) Set_Environment

"""Function Change_Environment choose environment and render it Input : environment_name Output : change the environment name and render it as a test and return the environment """

2) Brute_force_approach

""" Function Brute force approach Output : return frames to print them, print Timesteps and Penalties """

3) print_frames

""" Function : print_frames Input : frames Output : print the frames """

4) TrainingAgentFixedHyperParametersQLearning

"""Function Training Agent with Fixed HyperParameters Input : q_table, env, alpha, gamma, epsilon and trianingTimes Output : print Episode numbers that already trained

and return env and qtable after training """

5) EvaluationWithQLearning

"""Function : Evaluate agent's performance after Q-learning Input : env, episodes, q_table output : print episodes, timesteps and penalities and return episodes,timesteps,penalties """

- **Finally : I changed My enviroment form [Taxi-v3] to [FrozenLake-v0] without any changes or conflicts**

**** Second Part ****

**** Tune alpha, gamma, and/or epsilon using a decay over episodes **

6) TrainingAgentDescreasingHyperParametersQLearning

"""Function Training Agent with Descreasing HyperParameters to Tune alpha, gamma, and/or epsilon using a decay over episodes

Input : q_table, epsilon and trianingTimes Output : print Episode numbers that already trained and return env and qtable after training """

- **Finally : In comparison sion with our function in this part and the other function withous any decreasing hyperparameters, our function has very little change timestpes in comparison the previous function that work with fixed hyper parameters**

**** Third Part ****

**** Implement a grid search to discover the best hyperparameters **

7) TrainingAgentFixedHyperParametersQLearningWithoutLog

"""Function Training Agent with Fixed HyperParameters Without screen Log and use it in Implementation of a grid search to discover the best hyperparameters Input : q_table, env, alpha, gamma, epsilon and trianingTimes Output : print Episode numbers that already trained and return env and qtable after training """

- **Finally : I used combinations of hyperparameters and I selected the best model based on the smallest timestamp and smallest number of penalties**

Run the Code

- Upload the ipynb code file into "Google Colab" or Anaconda "Jupyter Notebook"

- Press "Run All" in the control panel or "Restart Kernel and Run All" to run all code
- In case of run each code cell alone, press the run button that appear at each code cell tents

Contributing

Any kind of enhancement or contribution is welcomed.