ID: 204693410 and 034662072

Assignment 1 Project Details

Taxi.py - The file that holds the Taxi class with all the relevant data structures and function for learning the environment with an agent as well as performing preforming policy iteration.

main.py - The main file for code execution.

3-simulations - 3 simulation run example. Also in the next link as a rendered file – 3-simulations.

Improvement_rate – A graph presenting the improvement rate of the algorithm. Y axis represents the mean expected return of all n starting points and the X axis represents the iterations in the policy iteration algorithm.

Methods:

In Taxi.py we implement an environments learner signed as env_learner(). The initialization of the Taxi class requires the user to insert the gama and the number of different environments to learn (n_env) as arguments. The user will then call env_learner() and learn all n_env environments. Note that although the gym library designs the environment to end after 200 steps, we bypass that limitation for the purpose of learning as much as we can from the environments. The environment terminates only after the terminating state is achieved. Therefore, a single environment learning process may take 1000 steps or even 10,000 steps with the latter being rare.

The details about the environments such as the transition from state to state and the rewards are learnt during the process and saved in different arrays, for the purpose of implementing policy iteration after the environment learning process is complete. The following are the main data structures used from class Taxi in the project:

transition_function - a 2D array that the index represents the state, thus 500 elements. And each element consists of at most 6 possible successive states. All states are initialized as -1, -1 for no successive state.

Reward_function – array that the index represents the state, thus 500 elements. And each element consists of 6 possible rewards. All rewards initialized as -10.

value_function – 1D array for the value of each state, updated in policy evaluation. Initialized as 0.

policy_function – 1D array for the best policy of each state. Policies are initialized as random policies.

After the environments are learnt, we run a loop which iterates over policy evaluation and policy improvement and ends until no policy has improved. Since we treat n different environments, we therefore plot the improvement rate of the average expected return of all starting states. Note that in our implementation we use asynchronous policy evaluation.

In the main.py file we have the simulation() function which runs a simulation with a random starting point of a random generated environment and plots the steps taken based on the optimal policy as well as prints the details of each step. If the simulation runs into an unknown state, a random action takes place. In this file the gama and number of environments (n_env) can be changed.

check_states is an array that hold the values of each state according to its state number as requested in the project's description.

The function for returning the value function is a Taxi-class variable which is simply copied to check_states which is why they both perform similarly. Thus, given a state as index returns the value generated with the policy iteration algorithm.

After running main.py, learning the environment followed by the policy iteration will take place. Subsequently, a simulation will be shown, once it completes each state and its generated value will be printed. If a state was never observed the value is defaulted to 0.