Documentation

Reinforcement Learning Setup:

State: Day and time based on their location and last notification sent. I split time in 4 parts, particularly morning, afternoon, evening and sleeping.

Action: either send notification or not

Reward: If notification not send 0 reward and, if send and user interact reward +3, if it ignores than rewards -1.

DESCRIPTION OF EACH FILE:

1) File name: Constant:

=> This file has assign value to all the features in state. If you want to change anything in state, you have to modify the values here.

2) Folder Name: human_modelling_utils , File name: utils.py

i) Function: getTimeState(hour, miniute):

=> This function will decide how to break time (state) in hour. It will take variable hour and minute, and assign that hour, minute into a particular state with the help of constant file. You can modify state time as per your choice in this function.

ii) Function: getDayState(day):

=> This function returns day feature of a particular state.

iii) Function: getLastNotificationState(last_notification_time):

=> This function takes last_notification_time and separate it into two parts, i.e. notification sends within 1 hour or not. You can modify it as per your choice.

iv) Function: getDeltaMinutes(day1, hour1, minute1, day2, hour2, minute2):

=> This function returns difference between 2 times.

v) Function: allTimeStates():

=> This function returns total time feature of state that has been made. If you modified the getTimeState function than you should modify here also.

vi) Function: allDayStates():

=> Similar to allTimeStates function shown above.

vii) Function: allLastNotificationStates():

=> Similar to allTimeStates.

viii) Function: normalize(*args):

=> This function normalize the value.

ix) Function: argmaxDict(d):

=> This function help to find the maximum value of an action in any given state. i.e. help to update the q-agent value function.

3) Folder name: environment:

i) File name: __init__:

⇨ This file initialize all other file (function), in the environment folder.

ii) File name: base_environment:

⇨ This file constructs a class BaseEnvironment, which is the main class of all other file in this folder.

iii) File name: morning_baised_user:

⇨ This file contains class MorningBaisedUser. This is an inheritance class of BaseEnvironment. This class contain 2 function one is __init__ and other is the getResponseDistribution. Here, I have initial the behaviour of user that 90% time the user click the notification in the morning and 10% time he will click in other state.

Now the getResponseDistribution funtion will take state feature and give probability of answering and not answering the notification.

4) **Folder name: Openai_gym, File name: basic_engagement_gym_base.py :**

=> This is the main file for custom environment, I have used gym library to build custom environment. The main class is BaseEngagementGymBase.

The main functions in this class are as follows:

i) Function: __init__(self, config = None):

⇨ This function initialize the attributes of this class and the main attributes are rewardCriteria, environment, episodeLengthDay, stepSizeMinute.

ii) Function: get_observation_space():

⇨ This function is used to define the number of variable in observation space , in the init function above. I used Box tool from gym.spaces library to define it.

iii) Function: reset():

⇨ This function gives the initial state to agent and reset is used after each episode to initialize state.

iv) Function: step(self, action):

⇨ This function takes input as action, that our agent takes and provide next_state, reward and done parameter. Done variable is used to check that the episode is end or not.

v) Function: _generate_state():

⇨ This function gives the next state.

vi) Function: _generate_reward(action):

⇨ This function gives reward on the basis of users probability distribution.

vii) Function: _printResults():

⇨ This functions print the result of our experiment.

5) Folder name: Agent:

i) File name: __init__:

=> This file is to initialize all other function in this folder.

ii) File name: base_agent.py:

=> This file contain class name BaseAgent, which is the main class for all other file.

iii) File name: Q_learning_agent2:

=> This file contains inheritance class QLearningAgent2, this class contains functions : getAction, feedReward, feedBatchRewards, generateInitialModel, _updateQTable, printQTable.

a) **getAction(self, stateTime, stateDay, stateLastNotification):**

=> This function take state features and take actions for agent, while using
Ɛ- Greedy mechanism for Q-learning.

b) feedReward:

=> This function feedReward for the action taken by agent.

c) feedBatchReward:

=> This is for offline data and it feed reward in batches.

d) generateInitialModel:

=> This function is to initialize the q function.

e) updateQTable:

=> This function updates q-function for every previous state, action pair.

f) printQTable:

=> This function print Q function value for each state-action pair.

6) Folder Name: human_modelling_utils , File name: chronometer.py:

=> This file contains class Chronometer, and every functions is explain in the file itself.