MITx 6.86x

Machine Learning with Python-From Linear Models to Deep Learning

4. Tabular Q-learning for Home World game

Bookmark this page

Project due May 10, 2023 08:59 -03 Completed

In this section you will evaluate the tabular Q-learning algorithms for the *Home world* g
state observable to the player is described in text. Therefore we have to choose a mec
descriptions into vector representations.

In this section you will consider a simple approach that assigns a unique index for each
particular, we will build two dictionaries:

- `dict_room_desc` that takes the room description text as the key and returns a uniqu
- `dict_quest_desc` that takes the quest description text as the key and returns a uni

For instance, consider an observable state $s = (s_r, s_q)$, where s_r and s_q are the text
current room and the current request, respectively. Then $i_r =$`dict_room_desc[`s_r`]` g
s_r and $i_q =$`dict_quest_desc[`s_q`]` gives the scalar index for s_q. That is, the textual s
mapped to a tuple $I = (i_r, i_q)$.

Normally, we would build these dictionaries as we train our agent, collecting descriptio
the list of known descriptions. For the purpose of this project, these dictionaries will be

Evaluating Tabular Q-learning on Home World

1.0/1 point (graded)

The following python files are provided:

- `framework.py` contains various functions for the text-based game environment th
  implemented for you. Some functions that you can call to train and testing your rein
  algorithms:
    - `newGame()`
        - Args: None
        - Return: A tuple where the first element is a description of the initial room, th
          description of the quest for this new game episode, and the last element is
          value *False* implying that the game is not over.
    - `step_game()`
        - Args:
            - `current_room_desc` : An description of the current room
            - `current_quest_desc` : A description of the current quest state

In this section, you will evaluate your learning algorithm for the Home World game. The
measure an agent's performance is the cumulative discounted reward obtained per epi
episodes.

The evaluation procedure is as follows. Each experiment (or run) consists of multiple ep
epochs is `NUM_EPOCHS`). In each epoch:

1. You first train the agent on `NUM_EPIS_TRAIN` episodes, following an -greedy p
   `TRAINING_EP` and updating the values.

2. Then, you have a testing phase of running `NUM_EPIS_TEST` episodes of the gam
   -greedy policy with `TESTING_EP`, which makes the agent choose the best a
   current Q-values of the time. At the testing phase of each epoch, you will c
   discounted reward for each episode and then obtain the average reward over th
   episodes.

Finally, at the end of the experiment, you will get a sequence of data (of size `NUM_EPOC`
testing performance at each epoch.

Note that there is randomness in both the training and testing phase. You will run the e
times and then compute the averaged reward performance over `NUM_RUNS` experimen

Most of these operations are handled by the boilerplate code provided in the `agent_t`
functions `run`, `run_epoch` and `main`, but you will need to complete the `run_episo`

Write a `run_episode` function that takes a boolean argument (whether the epsiode is
not) and runs one episode.

Reminder: You should implement this function locally first. Make sure you can achieve
on the Home World game before submitting your code

Available Functions: You have access to the NumPy python library as `np`, framework
`framework.newGame()` and `framework.step_game()`, constants `TRAINING_EP` and
dictionaries `dict_room_desc` and `dict_quest_desc` and previously implemented fun
`epsilon_greedy` and `tabular_q_learning`

```
def run_episode(for_training):
    """ Runs one episode
    If for training, update Q function
    If for testing, computes and return cumulative discounted reward

    Args:
        for_training (bool): True if for training

    Returns:
        None
    """
    epsilon = TRAINING_EP if for_training else TESTING_EP
    gamma_step = 1
```

Report performance

2/2 points (graded)

In your Q-learning algorithm, initialize at zero. Set `NUM_RUNS` , `NUM_EPIS_TRAI` `NUM_EPIS_TEST` , , `TRAINING_EP` , `TESTING_EP` and the le

Please enter the number of epochs when the learning algorithm converges. That is, the become stable.

15 ✓

Please enter the *average episodic rewards* of your Q-learning algorithm when it conver

< Previous Next >

0.52 ✓