

IN4010 Practical Assignment: Reinforcement Learning 1

December 1, 2019

Contents

1	Intro	2
2	Set up	2
3	Environments	2
4	Q-Learning	3

1 Intro

In this practical assignment, we will get more acquainted with reinforcement learning through implementing the Q-learning algorithm and running some experiments.

In the zip with code you can find the following files:

1. `simple_grid.py`, the environment(s) we will be using, see Section 3 for a description.
2. `q_learning_main.py`, contains a main loop you can use to run your experiments.
3. `q_learning_skeleton.py`, file for the Q-learning agent, to be coded up.

Deliverable

For this assignment you are required to upload a zip-file with your code containing solutions to the coding exercises and a pdf with answers to the questions.

2 Set up

You will need to get a working python3 installation and install open AI gym.

Set up your python environment

For managing python environments, we highly recommend `virtualenv` and `virtualenvwrapper`.

See: <https://virtualenvwrapper.readthedocs.io/en/latest/>

These will allow you to create project-specific python environments. For instance, after installing `virtualenvwrapper`, setting up a project workspace is as simple as

```
mkvirtualenv -p /usr/bin/python3 AIT
workon AIT
pip install ipython #optional if you want ipython
```

Install open AI gym

Now we can install gym:

```
pip install gym[box2d]
```

(For some reason, the box2d environments are not installed by default and hence need to be specified explicitly: <https://github.com/openai/gym/issues/1603>)

3 Environments

We will experiment with drunken walk environments that are located in `simple_grid.py`.

A simple grid environment, completely based on the code of 'FrozenLake', credits to the original authors.

You're finding your way home (G) after a great party which was happening at (S). Unfortunately, due to recreational intoxication you find yourself only moving into the intended direction 80% of the time, and perpendicular to that the other 20%.

To make matters worse, the local community has been cutting the budgets for pavement (.) maintenance, which means that the way to home is full of potholes (H), which are very likely to make you trip. If you fall, you are obviously magically transported back to the party, without getting some of that hard-earned sleep.

There are different maps available, an example is the "walkInThePark" which has the following shape:

```
"walkInThePark": [
  "S.....",
  ".....H..",
  ".....",
  ".....H..",
  ".....",
  "...H...G"
]
```

Another example is the "theAlley" map:

```
"theAlley": [
  "S...H...H...G"
]
```

An episode ends when you either trip or reach the goal. Reaching the goal gives a reward of +10. When stepping in a pothole there is a 20% chance to trip, tripping results in a broken leg penalty.

4 Q-Learning

The Q-Learning algorithm allows us to estimate the optimal Q function using only trajectories from the MDP obtained by following some policy.

Q-learning with ϵ -greedy exploration acts in the following way at a timestep t :

1. In the current state, s , take action a such that a is random with probability ϵ and the greedy action ($a = \max_{a \in A} Q(s, a)$) with probability $1 - \epsilon$;
2. Observe the reward and the next state, r and s' .
3. Update the Q-value as follows:

$$Q^{\text{new}}(s, a) = (1 - \alpha)Q^{\text{old}}(s, a) + \alpha[r + \gamma \max_{a' \in A} Q^{\text{old}}(s', a')]$$

Note that when the episode terminates in s' , the update is as follows:

$$Q^{\text{new}}(s, a) = (1 - \alpha)Q^{\text{old}}(s, a) + \alpha r$$

Coding Exercise 1. Implement Q-learning with ϵ -greedy action selection, complete the class given in `q_learning_skeleton.py`.

Question 1. Which environment, "walkInThePark" or "theAlley", is more difficult to learn in? Why?

Walk in the park

We'll start using the "walkInThePark" map.

Question 2. For the "walkInThePark" map, run some experiments for 1000 episodes with the following settings: $\epsilon = 0.05$, $\gamma = 0.9$, $\alpha = 0.1$. Does the agent learn an optimal policy? Why (not)? Report the (greedy) policy that the agent learned.

The alley

Now we will use the "theAlley" map.

Question 3. Calculate (or compute) Q^* , the optimal Q-values, for the "theAlley" map with $\gamma = 0.9$, $\text{BROKEN_LEG_PENALTY} = -10$.

Question 4. Run some experiments for 1000 episodes with the following settings: $\epsilon = 0.05$, $\gamma = 0.9$, $\alpha = 0.1$, $\text{BROKEN_LEG_PENALTY} = -10$. Does the agent learn an optimal policy? Why (not)?

Question 5. Now calculate (or compute) Q^* , the optimal Q-values, for the "theAlley" map with $\gamma = 0.9$, $\text{BROKEN_LEG_PENALTY} = -5$.

Question 6. Run some experiments for 1000 episodes with the following settings: $\epsilon = 0.05$, $\gamma = 0.9$, $\alpha = 0.1$, $\text{BROKEN_LEG_PENALTY} = -5$. Does the agent always learn an optimal policy? Why (not)?

Coding Exercise 2. Try to change the exploration strategy of the agent in a way that allows it to find the optimal solution more often (and quicker).

Question 7. Describe your new exploration strategy. Does it help the agent in learn the optimal policy more often/quicker?