Data-driven Intelligent Systems: Practical Assignment #6

Due on Fri, June 17, 2020

Task 1

Markov Decision Process (MDP), 20 points.
Explain in your own words what a Markov Decision Process process is and how an agent would interact with it. Give one example of an MDP and one example of a partially observable MDP (POMDP) in real life.

Task 2

Exploration vs. exploitation, 20 points.
Describe what exploration and exploitation means in the context of reinforcement learning. Name and explain a method for achieving exploration. What would happen if an agent explored too little or too much?

Task 3

SARSA and Q-Learning, 30 points.
Give answers to the following questions:

1. What is temporal difference (TD) learning?
2. How does the SARSA algorithm work?
3. How does Q-learning work, and how is it different from SARSA?

Optional: What is the problem with tabular Q-learning? Can it be used in environments with a continuous state space? Explain how neural networks could be used to improve Q-learning.

Task 4

Implementing Q-Learning, 30 points.
Now for the fun part. You will implement the Q-Learning algorithm and get an agent to solve an example environment! For this exercise we will be using *OpenAI Gym*, a library with which you can test, benchmark and compare reinforcement learning agents on a variety of different tasks, ranging from simple toy problems to Atari games and even simulations of robots. You can install OpenAI Gym with `pip install gym`.
We will be focusing on an environment called `Taxi-v3`. Your agent, the taxi driver (green) must pick up the passenger (blue) and then drop them off at this desired destination (magenta). To do so, it must navigate around a grid. Available actions are: Moving in each direction, picking the passenger up, and dropping the passenger off. At each timestep, the agent receives a reward of -1 (for taking so long). Furthermore, dropping the passenger off at their destination successfully awards the agent a reward of $+20$. However, using an illegal action, such as picking the passenger up when they are already in the vehicle, will produce a reward of -10.
In your material folder you will find `q_learning.ipynb`. Complete the missing code segments marked with `TODO`. If done correctly, you will notice your agent getting faster at completing the task over time! Explain the meaning of the values in the Q-table after training.

Optional: Let your agent balance a pole in the CartPole-v1 environment. The observations are continuous and not discrete, so you must first discretize them!

Hint: You can access the limits of the observation space with `env.observation_space.high` and `env.observation_space.low`. Bin the continuous action space per dimension and enumerate all possible combinations of each dimension (note the exponential growth so choose a small number of bins).

Further studies

If you want to go further with reinforcement learning and even use some neural networks instead of discretizing the state space, feel free to try it and ask us questions. It is rather straightforward to do so with the cartpole environment, but, unfortunately, Q-learning with neural networks is not very stable and requires some modifications to make it work properly. For a tutorial with PyTorch to learn cartpole based off of pixels have a look at `https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html`. A general and comperehensive introduction to deep reinforcement learning can be found at `https://spinningup.openai.com/en/latest/`.