SGN-13006 Introduction to Pattern Recognition and Machine Learning
TAU Computing Sciences
Exercise 5
Reinforcement learning (OpenAI Gym))

Be prepared for the exercise sessions (watch the demo lecture). You may ask TAs to help if you cannot make your program to work, but don't expect them to show you how to start from the scratch.

1. **OpenAI Gym – Taxi (V3) Environment** (50 points)

   In this exercise we will use the OpenAI Gym environment (https://www.openai.com/). It should be pre-installed in your computer, but if not then follow these instructions: https://github.com/openai/gym. Launch Python and type the following commands:

   ```
     $> python3
     >>> import gym
     >>> env = gym.make("Taxi-v3")
     >>> env.reset()
     >>> env.render()
   ```

   You should see a map with four locations. Read the description of the map from the source: https://github.com/openai/gym/blob/master/gym/envs/toy_text/taxi.py Use the commands 0-5 and solve the problem manually (render after each step):

   ```
     >>> state, reward, done, info = env.step(1)
     >>> env.render()
   ```

   Your task is to implement Q-learning to solve the Taxi problem with optimal policy. For this you need to fill in the missing parts in the following script:

   ```
   # Load OpenAI Gym and other necessary packages
   import gym
   import random
   import numpy
   import time

   # Environment
   env = gym.make("Taxi-v3")

   # Training parameters for Q learning
   alpha = 0.9 # Learning rate
   gamma = 0.9 # Future reward discount factor
   num_of_episodes = 1000
   num_of_steps = 500 # per each episode

   # Q tables for rewards
   Q_reward =  -100000*numpy.ones((500,6))

   # Training w/ random sampling of actions
   # YOU WRITE YOUR CODE HERE

   state = env.reset()
   tot_reward = 0
   for t in range(50):
       action = numpy.argmax(Q_reward[state,:])
       state, reward, done, info = env.step(action)
       tot_reward += reward
       env.render()
       time.sleep(1)
       if done:
   ```

```
        print("Total reward %d" %tot_reward)
        break
```

When you have implemented the training part, then run your method ten times and compute the average *total reward* and the average *number of actions*.