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

Be prepared for the exercise sessions. You may ask TA questions regarding your solutions, but don't expect them to show you how to start from the scratch. Before the end of the session, demonstrate your solution to TA to receive exercise points.

1. **OpenAI Gym – Taxi-v2 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-v2")
   >>> 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:

   ```
   import gym
   import random
   import numpy
   import time

   env = gym.make("Taxi-v2")
   next_state = -1000*numpy.ones((501,6))
   next_reward =  -1000*numpy.ones((501,6))

   # Training
   # THIS YOU NEED TO IMPLEMENT

   # Testing
   test_tot_reward = 0
   test_tot_actions = 0
   past_observation = -1
   observation = env.reset();
   for t in range(50):
       test_tot_actions = test_tot_actions+1
       action = numpy.argmax(next_reward[observation])
       if (observation == past_observation):
           # This is done only if gets stuck
           action = random.sample(range(0,6),1)
           action = action[0]
       past_observation = observation
       observation, reward, done, info = env.step(action)
       test_tot_reward = test_tot_reward+reward
       env.render()
       time.sleep(1)
   ```

```
    if done:
        break
print("Total reward: ")
print(test_tot_reward)
print("Total actions: ")
print(test_tot_actions)
```

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