School of Electronics and Computer Science
University of Southampton

COMP6247(2018/19): Reinforcement and Online Learning Lab 1 $\boxed{10\%}$

Issue	7 February 2019
Due	15 February 2019

The lab work may be done in any programming language, but Python is recommended. Snippets of code are given to help you get started; these are best viewed as pseudo-code and not as full working programmes.

Labs form part of the assessment for this module. This is worth 10% of the total assessment.

Lab One has two broad aims:

- To demonstrate an understanding of dynamic programming, its use in calculating shortest path through a network and using the shortest path calculations to understand network connectivities.
- To use the `OpenAi Gym` environment to show an understanding of a reinforcement learning algorithm.

The work set in Lab One is for 10% of the grade. But ideas and tools developed here will be useful in the final project set, later in the Semester.

1. Study the code on computing Fibonacci numbers; show by comparing the execution time how dynamic programming in which pre-computed partial solutions are re-used makes the algorithm efficient.

2. Study the pseudo code on computing all shortest paths on the graph given. Construct a network of 100 nodes with a connection density of 30%, the probability of connections being uniform and the cost of any link is constant set at 1.0. Compute the pairwise shortest distances and plot a histogram of these. **Optional:** Repeat the above for a preferential attachment network of similar size and connectivity.

3. Study the Mountain Car problem (Sutton and Barto, *Reinforcement Learning*, 2nd Edition; Example 10.1, page 244) and the implementation of learning a control policy using Q-Learning (Appendix C). Identify the steps where explortion and exploitation are implemented and how the continuous state and action spaces are discretized.

   For three enries in the Q-table, extract the action values as learning takes place and plot their evolution. Show how convergence changes when the exploration probability is increased / decreased from the default value of 0.05. Discuss any variation you note.

4. Write a report on your work. Your report should be *no more than* *four pages* in length. Typesetting your report in LaTeXstrongly recommended.

Mahesan Niranjan February 2019

Appendix A: Computing Fibonacci Numbers by Dynamic Programming

```
past_fib = {}
def fibonacci(n):
    if n in past_fib:
        return past_fib[n]
    if n == 0 or n == 1:
        past_fib[n] = 1
        return 1
    total = fibonacci(n-1) + fibonacci(n-2)
    past_fib[n] = total
    return total
print(fibonacci(1))
print(fibonacci(30))
```

Appendix B: Computing Shortest Path in a Network

```
graph = {0: {1:2, 4:4},
         1: {2,3},
         2: {3:5, 4:1},
         3: {0:8},
         4: {3:3}}

def allPairsShortestPath(g):
   dist = {}
   pred = {}
   for u in g:
       dist[u] = {}
       pred[u] = {}
       for v in g:
           dist[u][v] = sys.maxsize
           pred[u][v] = None

       dist[u][u] = 0
       pred[u][u] = None
       for v in g[u]:
           dist[u][v] = g[u][v]
           pred[u][v] = u

   for mid in g:
        for u in g:
            for v in g:
                newlen = dist[u][mid] + dist[mid][v]
                if newlen < dist[u][v]:
                    dist[u][v] = newlen
                    pred[u][v] = pred[mid][v]

   return(dist, pred)

def constructShortestPath(s, t, pred):
    path = [t]
    while t != s:
        t = pred[s][t]
        if t is None:
            return None
        path.insert(0,t)

    return path
```

```
dist, pred = allPairsShortestPath(graph)
```

Appendix C: Mountain Car

```
import gym
import numpy as np

env_name = "MountainCar-v0"
env = gym.make(env_name)

obs = env.reset()
env.render()

n_states   = 40
episodes   = 10
initial_lr = 1.0
min_lr     = 0.005
gamma      = 0.99
max_stps   = 300
epsilon    = 0.05

env = env.unwrapped
env.seed()
np.random.seed(0)

def discretization(env, obs):
    env_low    = env.observation_space.low
    env_high   = env.observation_space.high
    env_den    = (env_high - env_low) / n_states
    pos_den    = env_den[0]
    vel_den    = env_den[1]
    pos_high   = env_high[0]
    pos_low    = env_low[0]
    vel_high   = env_high[1]
    vel_low    = env_low[1]
    pos_scaled = int((obs[0] - pos_low) / pos_den)
    vel_scaled = int((obs[1] - vel_low) / vel_den)

    return pos_scaled, vel_scaled

q_table = np.zeros((n_states, n_states, env.action_space.n))
total_steps = 0
for episode in range(episodes):
    print("Episode:", episode)
    obs = env.reset()
    total_reward = 0
    alpha = max(min_lr, initial_lr*(gamma**(episode//100)))
    steps = 0
    while True:
        env.render()
        pos, vel = discretization(env, obs)

        if np.random.uniform(low=0, high=1) < epsilon:
            a = np.random.choice(env.action_space.n)
        else:
            a = np.argmax(q_table[pos][vel])

        obs, reward, terminate,_ = env.step(a)
        total_reward += abs(obs[0]+0.5)
        pos_, vel_ = discretization(env, obs)

        q_table[pos][vel][a] = (1-alpha)*q_table[pos][vel][a]
        + alpha*(reward+gamma*np.max(q_table[pos_][vel_]))

        steps += 1
        if terminate:
            break

while True:
    env.render()
```

Appendix D: Animation on google colab notebook

Unfortunately you may be prevented from installing packages (e.g. `gym`) omn the university system. One way around this is to use `google colab`, a collaborative working environment running on the cloud. By using the example below and running the Mountain Car programme in a notebook on colab, you may be able to circumvent this in the laboratory. This is inefficient because the packages are installed every time you run the programme, hence where possible, set up the environment in your own machine.

```
!apt-get install -y xvfb python-opengl > /dev/null 2>&1
!pip install gym pyvirtualdisplay > /dev/null 2>&1

import gym
import numpy as np
import matplotlib.pyplot as plt
from IPython import display as ipythondisplay

from pyvirtualdisplay import Display
display = Display(visible=0, size=(400, 300))
display.start()

env = gym.make("CartPole-v0")
env.reset()
prev_screen = env.render(mode='rgb_array')
plt.imshow(prev_screen)

for i in range(500):
  action = env.action_space.sample()
  obs, reward, done, info = env.step(action)
  screen = env.render(mode='rgb_array')

  plt.imshow(screen)
  ipythondisplay.clear_output(wait=True)
  ipythondisplay.display(plt.gcf())

  if done:
    break

ipythondisplay.clear_output(wait=True)
env.close()
```