CMPT 419/983 Assignment 3

- Due date: Dec. 2
- Submit zip file to CourSys

1) This question guides you through implementing the policy gradient algorithm with average reward baseline.

   Preparation:

   - Install gym and TensorFlow for Python. Documentation can be found at https://gym.openai.com/ and https://www.tensorflow.org/install.
   - Replace "`cartpole.py`" in gym with the version provided. The included file "`cartpole_stabilize.py`" contains the skeleton code for training a cartpole to achieve its goal of keeping its position centred and pole upright.

   The cartpole environment consists of a rotatable pole mounted on top of a cart. The states of the system are the position and velocity (x, v) of the cart, and the angular position and velocity (θ, ω) of the pole. The two possible actions are to push the cart left or right with a constant force.

   Our goal in this problem is to keep the cart's position near zero and the pole near upright for as long as possible. To encourage this, in the custom environment defined in the provided "cartpole.py" file, the cartpole system receives a reward of 1 for every time step in which its state satisfies $|x| \leq 0.5$ and $|\theta| \leq 4 \times \frac{\pi}{180}$. Training episodes terminate when the system state violates $|x| \leq 1.5$ or $|\theta| \leq 12 \times \frac{\pi}{180}$.

   a) In the __init__ method of the agent class, define a policy network that takes as input the state, has two fully hidden layers of the desired number of neurons with ReLU activation, and outputs the probability distribution of applying the two possible actions.

   b) In the __init__ method of the agent class, compute the probability of applying the actions in the input data.

   c) In the __init__ method of the agent class, define the loss function such that its gradient is $\nabla_\theta J(\theta)$.

d) Complete the `compute_advantage` function, which should compute a list of advantage values $A_t \coloneqq \sum_{t' \geq t} \gamma^{t'-t} r(s_t, a_t) - b$ for every time step across a batch of episodes, where $b = \mathbb{E}_{\tau \sim p(\tau;\theta)} \sum_{t \geq 0} \gamma^t r(s_t, a_t)$ is the average reward across the batch of episodes. Note that the batch size is specified by the "update_frequency" variable.

e) Complete the main part of the script (fill in the unmodified "`cartpole_stabilize.py`" at lines 73-78, 104-107, 122-124).

f) Produce several plots showing the state of cart-pole system at different snapshots in time for a well-performing episode.

g) Produce a plot showing sum of discounted reward in each episode vs. episode number.

2) EKF SLAM. Consider the Dubins Car, given by the dynamics

$$\dot{x} = v\cos\theta$$
$$\dot{y} = v\sin\theta$$
$$\dot{\theta} = \omega$$

a) Suppose there are 10 landmarks, with fixed x- and y-positions. Derive a discrete time model for the augmented states $(x, y, \theta, m_1, \ldots, m_{10})$ by using forward Euler.

b) Derive the Jacobian of the above transition model.

c) Starting from the provided MATLAB code, implement the one-step EKF algorithm to estimate the augmented state. You do not need to modify any code in the sub-folders. For your information, the `vehicle_dynamics` folder contains a simple vehicle dynamics simulator, and the `vehicle_sensors` folder contains a vehicle sensor simulator that noisily measures the range and bearing of landmarks up to some maximum distance.