1. Introduction

In this project, we address the task of learning control policies for text-based games using reinforcement learning. In these games, all interactions between players and the virtual world are through text. The current world state is described by elaborate text, and the underlying state is not directly observable. Players read descriptions of the state and respond with natural language **commands** to take actions.

For this project you will conduct experiments on a small **Home World**, which mimic the environment of a typical house.The world consists of a few rooms, and each room contains a representative object that the player can interact with. For instance, the kitchen has an **apple** that the player can **eat.** The goal of the player is to finish some quest. An example of a quest given to the player in text is **You are hungry now** . To complete this quest, the player has to navigate through the house to reach the kitchen and eat the apple. In this game, the room is **hidden** from the player, who only receives a description of the underlying room. At each step, the player read the text describing the current room and the quest, and respond with some command (e.g., **eat apple**). The player then receives some reward that depends on the state and his/her command.

In order to design an autonomous game player, we will employ a reinforcement learning framework to learn command policies using game rewards as feedback. Since the state observable to the player is described in text, we have to choose a mechanism that maps text descriptions into vector representations. A naive approach is to create a map that assigns a unique index for each text description. However, such approach becomes difficult to implement when the number of textual state descriptions are huge. An alternative method is to use a bag-of-words representation derived from the text description. This project requires you to complete the following tasks:

1. Implement the tabular Q-learning algorithm for a simple setting where each text description is associated with a unique index.

2. Implement the Q-learning algorithm with linear approximation architecture, using bag-of-words representation for textual state description.

3. Implement a deep Q-network.

4. Use your Q-learning algorithms on the **Home World** game.

Setup:

As with the previous projects, please use Python's NumPy numerical library for handling arrays and array operations; use matplotlib for producing figures and plots.

1. Note on software: For the all the projects, we will use python 3.6 augmented with the NumPy numerical toolbox, the matplotlib plotting toolbox. For THIS project, you will also be using **PyTorch** for implementing Neural Nets

2. Download rl.tar.gz and untar it in to a working directory. The archive contains various data files, along with the following python files:

- `agent_tabular.py` where you will implement an agent using tabular Q-learning
- `agent_linear.py` where you will implement an agent using Q-learning with linear approximation
- `agent_dqn.py` where you will implement an agent using a deep Q-network

Tip: Throughout the whole online grading system, you can assume the NumPy python library is already imported as `np`. In some problems you will also have access to python's `random` library, and other functions you've already implemented. Look out for the "Available Functions" Tip before the codebox, as you did in the last project.

This project will unfold both on MITx and on your local machine. However, we encourage you to first implement the functions locally and run the test scripts to validate basic functionality. Think of the online graders as a submission box to submit your code when it is ready. You should not have to use the online graders to debug your code. A good strategy for this project is to first implement all the functions from tab 3 and 4 to check for acceptable performance before submitting your code online.

Topic: Unit 5 Reinforcement Learning (2 weeks) :Project 5: Text-Based Game / 1. Introduction