Reinforcement_learning

June 28, 2021

1 Introduction to Reinforcement Learning

Author: Jacob Koehler, Guilherme Freitas and Dhavide Aruliah
Reviewer: Jessica Cervi
Expected time = 2.5 hours
Total points = 65 points

1.1 Assignment Overview

The focus of this assignment is **Reinforcement Learning**, i.e., the branch of machine learning concerned with choosing actions in an environment where labeled input/output pairs may not be available and sub-optimal actions may not be explicitly corrected. The principal idea is to balance the exploration of uncharted territory with exploitation of current knowledge to maximize some kind of a cumulative reward. In this assignment, we aim to encapsulate the ideas from the lectures in code. In the first part, we'll use some basic pre-made environments from OpenAI's `gym` package to explore the effect of different policies on agents. In the second part, we'll experiment with a *multi-armed bandit*.

This assignment is designed to build your familiarity and comfort coding in Python while also helping you review key topics from each module. As you progress through the assignment, answers will get increasingly complex. It is important that you adopt a data scientist's mindset when completing this assignment. **Remember to run your code from each cell before submitting your assignment.** Running your code beforehand will notify you of errors and give you a chance to fix your errors before submitting. You should view your Vocareum submission as if you are delivering a final project to your manager or client.

Vocareum Tips - Do not add arguments or options to functions unless you are specifically asked to. This will cause an error in Vocareum. - Do not use a library unless you are expicitly asked to in the question. - You can download the Grading Report after submitting the assignment. This will include feedback and hints on incorrect questions.

1.1.1 Learning Objectives

- Differentiate between Reinforcerment Learning and Machine Learning
- Implement various policies for different games
- Represent Markov decision processes
- Define transition probability and reward matrices
- Implement a Q-Value iteration and extract the optimal policy
- Implement multi-armed bandit problems in Python