Exercise 1

Reinforcement Learning
Lecturer: Alon Peled-Cohen
TA: Yarin Udi

Due Date: June, ..., 2024

1 Introduction

In this exercise you will:

1. Practice basic dynamic programing definitions.
2. Get familiar with PyTorch by training a simple neural network on a small dataset.
3. Get familiar with OpenAI gym environment by training an agent using random search.

2 Setup

For the programing section, you will need to setup a Python environment containing the following packages: PyTorch and OpenAI Gym.

- Install the relevant packages in a virtual environment (e.g. venv or Anaconda)
- PyTorch installation, follow the instruction here
- OpenAI Gym installation, follow the instruction here

3 Theory

3.1 Question 1: Longest increasing subsequence [1]

Given a sequence of n real numbers $a_1, a_2, ..., a_n$, find the longest strictly increasing subsequence (not necessarily contiguous). For example, for the sequence (3, 1, 5, 3, 4), the solution is (1, 3, 4). Remark: the number of subsequences is 2^n, therefore an exhaustive search is inefficient.

[1] Question taken from "046194 - Learning and Planning in Dynamical Systems" by Shie Manor

3.2 Question 2: Moses The Mouse [2]

Moses the mouse starts his journey in the southwest room in a $M*N$ rectangular apartment with $M*N$ rooms of size $1*1$, some of which contain cheese. After his rare head injury in the mid-scroll war, Moses can only travel north or east. An illustration of Moses's life for $M=5, N=8$ is given in Figure 1

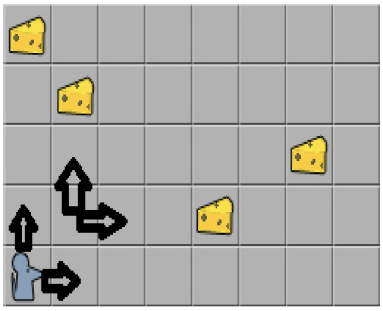

Figure 1: Moses The Mouse

Being a mouse and all, Moses wants to gather as much cheese as possible until he reaches the north-east room of the apartment.

1. Formulate the problem as a finite horizon decision problem: Define the state space, the action space and the cumulative cost function.
2. What is the horizon of the problem?
3. How many possible trajectories are there? How does the number of trajectories behaves as a function of N when M = 2? How does it behave as a function of N when M = N? Please note that you don't need to calculate the exact number of states, you can give the order number (this also apply to the rest of this question).
4. Aharon, Moses's long lost war-buddy woke up confused next to Moses and decided to join him in his quest (needless to say, both mice suffer the same rare head injury).
   (a) Explain what will happen if both mice ignore each other's existence and act 'optimal' with respect to the original problem.
   (b) Assume both mice decided to coordinate their efforts and split the loot. How many states and actions are there now?
   (c) Now their entire rarely-head-injured division has joined the journey. Assume there's a total of K mice, how many states and actions are there now?

[2] Question taken from "046194 - Learning and Planning in Dynamical Systems" by Shie Manor

3.3 Question 3: Language Model [3]

In the city of Bokoboko the locals use a language with only 3 letters ('B','K','O'). After careful inspection of this language, researchers have reached two conclusions:

1. Every word starts with the letter 'B'.
2. Every consecutive letter is distributed only according to the previous letter as follows:

$$P\left(l_{t+1}|l_t\right) = \begin{matrix} B \\ K \\ O \\ - \end{matrix} \begin{bmatrix} 0.1 & 0.325 & 0.25 & 0.325 \\ 0.4 & 0 & 0.4 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.4 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Where '-' represents the end of a word. For example, the probability of the word 'bko' is given by 0.325 * 0.4 * 0.4 = 0.052.

1. Find the probability of the following words: 'Bob', 'Ok', 'B', 'Book', 'Boooo'
2. We wish to find the most probable word in the language of length K.
   (a) Formulate the problem as a finite horizon decision problem: Define the state space, the action space and the multiplicative cost function.
   (b) Bound the complexity of finding the best combination.
   (c) Find a reduction from the given problem to an analogous problem with additive cost function instead.
   (d) Explain when each approach (multiplicative vs. additive) is preferable. Hint: Think of the consequence of applying the reduction on the memory representation of a number in a standard operating system.
   (e) Write a code in Python which finds the most probable word of a given size using dynamic programming. What is the most probable word of size 5?

4 Programming

4.1 Question 1: MNIST

Given the MNIST dataset which consists of $28*28$ image showing a handwritten digit from 0 to 9, and corresponding labels stating what digit the image shows,

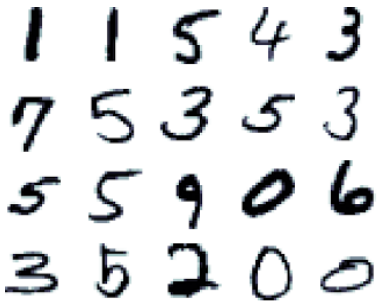

Figure 2: Exapmle of MNIST Images

the goal is to learn a classifier that predicts the class (label) of several test-set inputs

You are provided with the training script "mnist.py". The script trains a simple logistic regression model to solve the task at hand. For it to work, you'll need to implement the missing functionality.

1. Implement the training and evaluation loop currently marked as "TODO" comments in the code. Use the already-defined model and dataset. Make sure you report the loss on the training set each batch and the accuracy on the evaluation set at the end of training.

2. Find a better optimization configuration that works well (learning rates, mini-batch size, no. of epochs, and so on). You are provided with some starter configuration options for SGD. However, more options are possible and the values provided are not necessarily optimal. With a better choice of parameters, you will converge much faster and achieve better accuracy.

   The new configuration should result **faster** (in terms of #epochs) convergence. **Optional:** You are not limited to SGD and are encouraged to try other optimization methods, see PyTorch optim package.

3. Train a deeper model by introducing a ReLU non-linearity and another linear layer. The size of the hidden layer should be 500.

Submit your modified code (for each part), plots of the training loss curve comparing all the parts together, the accuracy you achieved on the test set, and a path to the model you trained. **NOTE:** Evaluate the different configurations for at least 100 epochs.

4.2 Question 2: OpenAI Gym

In this question you will learn how to use OpenAI Gym by training an agent using random search on the "CartPole-v1" environment. The problem consists

[3]Question taken from "046194 - Learning and Planning in Dynamical Systems" by Shie Manor

of balancing a pole connected with one joint on top of a moving cart. The only actions are to add a force of -1 or +1 to the cart, pushing it left or right.

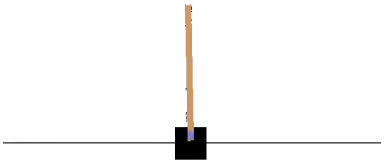

Figure 3: CartPole Environment

1. Familiarize yourself with OpenAI Gym by going over the short introduction available here: https://www.gymlibrary.dev/. You should be able to understand how to load an environment, perform an action and the observations returned.

2. In CartPole's environment, each step returns a 4-dimensional observation vector representing information such as the angle of the pole, cart position, and so on. Given that observation, your agent will need to act: move left or right.

   Implement the following agent: Given a 4-dimensional observations, $o_t \in R^4$, define the following agent:

   $$a_t = \begin{cases} 1 & \text{if } o_t * w \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

   Where $w \in R^4$ is a vector of weight, each weight corresponding to one of the observations. Initialize the weights randomly between [-1, 1].

3. Evaluate your agent by running an **episode** of the environment. An episode ends when the pole drops or 200 steps have passed. The score of your agent, should be the accumulated reward for all steps in the episode.

4. Train your agent using **random search**. This is done by sampling different weights multiple times and greedily choosing the weights that scored the highest reward. You should sample at least 10000 times.

5. Evaluate the suggested random search scheme. Run the search for 1000 times and report for each search the number of episodes required until the agent reached a score of 200. Plot a histogram of number of episodes required until score 200 and report the average number of episodes.

Submit your full training code(agent, episode and random search), the histogram plot and the average number of episodes.