Lecture: Artificial Intelligence
Project 2

Summer Semester 2022

Prof. Dr. Eirini Ntoutsi
M.Sc. Arjun Roy[1]
AG Künstliche Intelligenz und Maschinelles Lernen

Announced: 28.06.2022

---

1 Organization

- In order to pass you need to complete the task given in Sec. 4.
- The task throws coding challenges which you need to complete in a given notebook (.ipynb file) named `pacman.ipynb` which you can directly download from the white-board.
- You may participate in this project as a group of 2 persons.
- Submit your solution no later than 29th July 23:59.
- You can make re-submissions until the deadline, but the latest received submission will be graded only.
- Submit your solution as a zip file with the following name format: **"[YourNames seperated by commas]_AI22Project2"** containing your files: 'pacman.ipynb' (your codes), and 'explain.pdf' (your justification/explanation for each function/class defined by you and your training observations) files.

The problem and background concepts are described in Section 2; the environment, library dependencies and technical requirements are described in Section 3. If you encounter any bugs or have any further question, please write to: arjun.roy@fu-berlin.de

2 Problem definition and basic concepts

Reinforcement learning is an area of machine learning concerned with how intelligent agents ought to take actions in an environment in order to maximize the notion of cumulative reward[2]. In this project, we will focus on the game of Pacman[3].

[1]We acknowledge the assistance of MSc. Emmanouil Panagiotou in developing the project.
[2]https://en.wikipedia.org/wiki/Reinforcement_learning
[3]https://en.wikipedia.org/wiki/Pac-Man

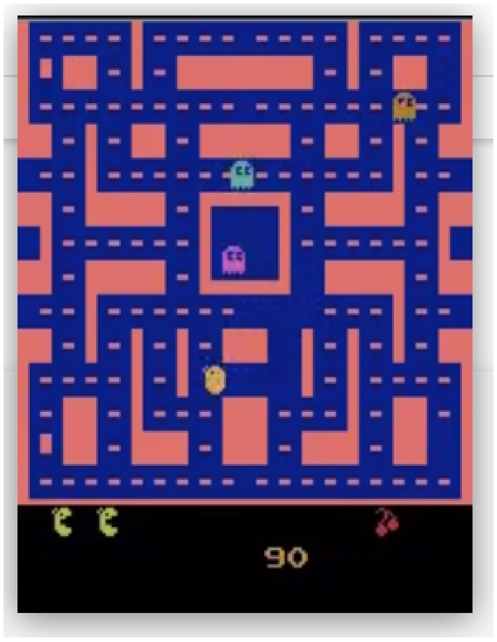

Figure 1: Example: Pacman game environment

Rules: Figure 1 demonstrates an example of a Pacman game environment you will require to solve. The rules of the game are as follows:

- There is **1** Pacman and **many** ghosts in the maze at a time.
- In total Pacman has **3** lives.
- Pacman can go any direction and turn whenever it want as long as it stay within the lines of the maze.
- The pacman collects scores on eating each food pellet (small dots) in the maze.
- The goal of Pacman is to eat all the food pellets in the maze without being eaten by the malicious patrolling ghosts.
- Upon being caught/eaten by a ghost, the Pacman loses a life and starts with another life if any life is left. However, the score of the previous life gets carried forward.
- The game terminates if either Pacman eats up all the food pallets or if Pacman has no more life left.

The goal of the project is to teach the Pacman agent to play optimally using RL and in particular, deep Q-learning.

Problem definition The problem formulation is as follows:

- **States:** Game frames (image array)
- **Actions:** 9 actions ['NOOP', 'UP', 'RIGHT', 'LEFT', 'DOWN', 'UPRIGHT', 'UPLEFT', 'DOWNRIGHT', 'DOWNLEFT']
- **Rewards:** A real valued scalar in the range: $(-\infty, \infty)$
- **Termination (IS_DONE):** A boolean value pointing the game is terminated or not
- **Lives:** A positive integer indicating the number of lives available to the agent through the game.

Deep Q-learning Deep Q-Learning replaces the regular Q-table with a neural network, called Deep Q-Network (DQN) - lets denote the DQN parameters by θ. Rather than mapping a state-action pair (s, a) to a q-value, a neural network maps input states s to (action a, q-value) pairs.
DQN: In our case, the DQN will take as input a state (image array) and will return the q-values of the different actions. Designing the DQL architecture is part of your work. In your report, please describe the architecture and justify your selection.
Action selecton: In its simplest form, the agent decides about the next action greedily by selecting the one with the highest q-value . In general, a combination of *exploitation* of what you already know and *exploration* of new actions is applied. You will be asked to experiment with different options.
DQN training: The choice of the loss function to train the DQN is part of your work. Typically the mean squared error between the predicted q-value (by the network) and the target q-value is used, where the target q-value is defined as the direct reward for the transition to some next state plus the max q-value of the next state downgraded by γ.
The DQN architecture involving all the steps of action, observation, and replay buffer is depicted in Fig. 2.
To update the DQN we need to use a gradient descent based optimization method to back propagate the Q-estimation error. To estimate the Q-values, we use the Bellman equation:

$$Q^*(s, a) = r + \gamma * \max_{a'} Q(s', a') \tag{1}$$

The Bellman equation returns the value of taking an action in a state. This is based on the reward in the current state (r), and then the maximum value of the next state (s') discounted by a factor γ.

3 Technical Requirements

To make the project hardware (GPU and system memory) independent, so that everyone have equal opportunity to train, we advice to use Google Colab: `https://colab.research.google.com/?utm_source=scs-index`. In the provided `pacman.ipynb` file all the installation code for libraries that one may require to run the task in colab is already provided. For running the code

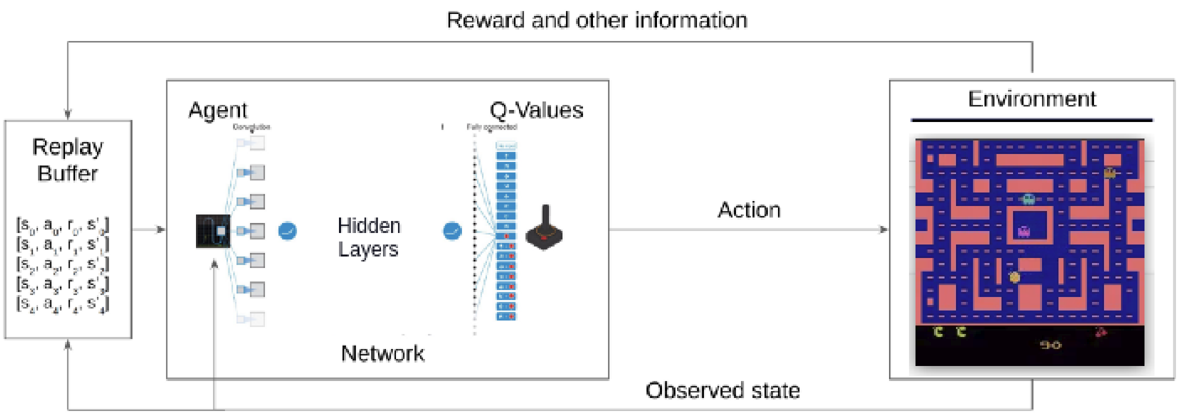

Figure 2: An overview of DQN for the Pacman game.

locally you will mainly require **Python 3.7** environment, **gym 0.21**, **gym[atari]**, **autorom**, and the deep learning library you intend to use. The part of the codes that you may need to modify/change are mentioned in the notebook either within comment quote ("'Your Code HERE "') or with # comment. But they are not necessarily limited to predefined structure i.e if you want to define some new functions or class for your convenience, it is acceptable. Thus, feel free to be more creative.

4 Task: Learn to play Packman with DQN

- The first sub-task is to learn a DQN that can predict the q-values for each state action combination, i.e., $Q(s,a)$. This includes the DQN architecture and the loss function. Justify the network architecture and the loss choice in your report. Also, report on the convergence of training.

  , a loss function based on Eq. 1 to evaluate Q-value estimation error, and a learning strategy (e.g SGD, Adam, etc.) to learn the DQN using the loss function. Plot the training loss and avg score per epoch. Write down your observation about the training loss convergence in your submission pdf.

- Experiment with different exploration-exploitation strategies including the extreme cases (only exploitation aka greedy) or only exploration. Report on your observations,

Further tasks This is only for interested students and not obligatory to pass the project.

- We are using the same DQN to both estimate the target Q-values and to predict the current Q-values. This is a problem because as we update the DQN, the targets also change. You can think of this as chasing a moving target going everywhere, which may hamper training stability. Try finding some strategy to fix this issue (explain why your strategy will work better).