Departamento de Informática
Universidad Carlos III de Madrid

Bachelor in Data Science and Engineering

Machine Learning I
Course 2022-2023

Tutorial 1: The Pac-Man Platform

Feature extraction and manual programming of an agent.

February 6, 2023

- The aim of this tutorial is to get used with the Pac-Man platform, which will be used in the practical lessons of the course.
- The tutorial can be done in Linux (recommended), Windows or Mac.
- It is important to solve the exercises in order.

1 Introduction

The practical part of this course is based on Pac-Man[1], the classical videogame. We will use a simplified version of the game to develop autonomous agents while exploring and understanding some machine learning concepts.

In this version of the game, we have ghosts and food points called "pac dots". Pac-Man and the ghosts can move one cell per turn or "tick". On every tick that Pac-Man does not eat neither a ghost nor a pac dot, -1 is substracted from the score. Each eaten ghost is awarded with 200 points, and each pac dot eaten is awarded with 100 points. The aim of the game is to eat all the ghosts appearing in the screen, maximizing the score. In this simplified version, Pac-Man can always eat the ghosts.

During the course of the game, we will have access to the game state, information we will use to generate autonomous agents based on different machine learning techniques.

2 Python Programming Language

The Pac-Man platform is developed in Python, so we will use this tutorial also to get used with this programming language. Python is a interpreted programming language, which means that we do not need to compile any file to execute the programs. The platform is programmed in Python 3 (usually pre-installed in some OS), and that is the version we will use in the lab sessions. You can download it from here:

`https://www.python.org/downloads`

On the internet, there are many examples that can be consulted to solve specific problems. The following official Python link is a good reference when questions arise and to see code samples:

`https://docs.python.org/3/tutorial/index.html`

Note: for executing the exercise code, you may need to install also the following dependencies:

sudo apt install python3-pip

pip3 install future

sudo apt-get install python3-tk

[1]https://en.wikipedia.org/wiki/Pac-Man

3 Code Execution

1. Download the Pac-Man code from Aula Global (**pacman.zip** file).
2. Unzip the *pacman* folder.
3. Open a terminal and enter to the *pacman* directory.
4. Launch the game by executing the following command in the terminal:

```
python busters.py
```

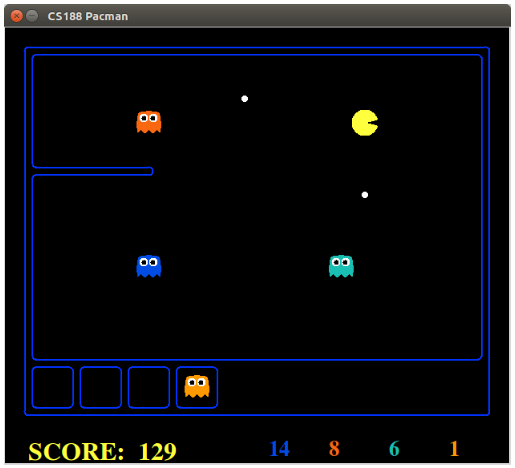

Figure 1: Pac-Man interface.

By default, the game starts with Pac-Man being controlled with the keyboard, ghosts are standing still and the maze is *oneHunt*. To check all the available options, you can execute the following command:

```
python busters.py --help
```

The main arguments you can change are:

- **-n GAMES** Number of games (1 by default).
- **-l LAYOUT_FILE** Maze (*oneHunt* by default).
- **-p TYPE** Type of Pac-Man agent (*BustersKeyBoardAgent* by default).
- **-g TYPE** Type of ghosts.
- **-k NUMGHOSTS** Number of ghosts, from 1 to 4 (4 by default).
- **-t** Time delay between frames.

4 Exercises

You should answer the following questions. Before that, you should explore different options and try to understand how the code works. You can take a look to the Appendix section, where the most important files are briefly described.

1. What information is shown in the interface? And in the terminal? Which position does Pac-Man occupy initially?
2. In your opinion, which data could be useful to decide what Pac-Man should do on each tick?
3. Take a look to the *layout* folder. How are the mazes defined? Create a new maze, save it and execute the game on it. Include a screenshot of the maze in the document.

4. Execute the agent *BasicAgentAA* with the following command: `python busters.py -p BasicAgentAA`

   Describe which information is shown in the screen about the game state. Which information do you consider the most useful?

5. Program a method called *printLineData()* inside the *BasicAgentAA* agent from the *bustersAgents.py* file. This method should return a string with the information from the Pac-Man state you consider relevant. Then, you call this method from the main loop of the game in *Game.py* to write the information to a file.

   For each tick, you should store a line with all the considered information, splitting each data with commas.

   Moreover, each time a new game starts, the new lines must be appended below the old ones. You should not rewrite the file when a new game starts.

6. Program an "intelligent" Pac-Man using your own rules. To do so, modify the method *chooseAction()* from the *BasicAgentAA* class. As you can see, by now it just act randomly. Your new Pac-Man should chase and eat all the ghosts in the screen. Compare the results obtained by executing the game with the static and random ghosts (*-g RandomGhost*). Play several games and determine the number of ticks (on average) that your agent needs to eat the ghosts.

7. The agent you just programmed does not use any machine learning technique. What advantages do you think machine learning may have to control Pac-Man?

5 Files to Submit

All the lab assignments **must** be done in groups of 2 people. You must submit a .zip file containing the required material through Aula Global before the following deadline: **Tuesday, February 15th at 8:00**. The name of the zip file must contain the last 6 digits of both student's NIA, `i.e., tutorial1-123456-234567.zip`

The zip file must contain the following files:

1. A **PDF** document with:
   - Cover page with the names and NIAs of both students.
   - Answers to all the questions appearing in the Exercises section.
   - Description of the method implemented to save the game state on each tick.
   - Description of the behavior programmed for the `BasicAgentAA`.
   - Conclusions.
2. Include in a folder called "material" the following files:
   - The new maze.
   - The file generated with the method that saves the game state on each tick.
   - The files modified to create the intelligent agent.

Please, **be very careful and respect the submission rules**.

Appendix: Pac-Man Platform Files

- **bustersAgents.py**: Pac-Man agents with different behaviors. Each agent is included as a new class.
- **inference.py**: code to compute the distance to the ghosts.
- **busters.py**: main class of the game.
- **bustersGhostAgents.py**: class that defines the behavior of the ghosts.
- **distanceCalculator.py**: class that computes distances.
- **game.py**: class where the main loop of the game runs ("*def run:loop*"). In this file are also defined: the maze, in the **Grid** class; the Pac-Man actions in the **Actions** class; and the information about the game state in the class **GameStateData**.

- **ghostAgents.py**: agents that control the ghosts (**RandomGhost** and **DirectionalGhost**).
- **graphicsDisplay.py**: graphical interface of the game.
- **graphicsUtils.py**: auxiliary class dedicated to the interface of the game.
- **keyboardAgents.py**: keyboard control of Pac-Man.
- **layout.py**: code to read the mazes and save their content.
- **util.py**: auxiliary code with different support methods.