Value Function Approximation with ANN for solving Pong game (two sessions)

This lab will spread on two sessions. You are to submit your solution IN GROUPS OF TWO, before the deadline April 19 2021, 18h00. Submission is mandatory, ipynp/py/zip file, one per group, make sure to write both names in the submitted files.

Ask your parents, they probably have played to the Pong game when they were young. The purpose of this lab (this week and next week) is to program an intelligent player using ANN. This artificial player will learn from you, by observing when you play.

More specifically, we want to use an ANN as a Value Function Approximator that could estimate the next action to play given a state. The possible actions are up, down, and nope (we need to add this one to give the possibility to remain still!). The state will be given ONLY by the current image (matrix of pixels) of the game.

Of course, in this simple game, a simpler and better solution would be to predict the right position of the paddle using the fact that all trajectories are linear. The pong is only an example of game and we want to test if we can learn the action to be performed using an ANN.

1. Necessary installations

python3, numpy

If you don't have it already...

- install python3
- then install numpy

First, upgrade pip to the latest version:

```
python3 -m pip install --upgrade pip

Then install numpy
python3 -m pip install numpy
```

For window, the name of the executable file for python3 may be different (python, py, ...). You need to be sure to use the version 3 of python (you can have different versions of python installed on the same computer). If you want to check the version of python, use this command:

```
python3 –version
```

tensorflow.keras

We will use keras, that is now part of tensorflow. Il you are lucky, you just need to use pip:

```
python3 -m pip install tensorflow
python3 -m pip install keras
```

For some systems (some MacOS), binary installations of tensorflow may not work. An alternative is to install pytorch instead of tensorflow (then use keras will on top of pytorch) **or** to compile the sources of tensorflow.

For the lucky students who have GPUs on their computer, make sure tensorflow and keras are using it. Don't worry if you don't have GPUs, everything works on any laptop without GPU. It is also why we will use very simple games like pong.

gym

You should already have **gym** installed. This environment offers the possibility to compare different Reinforcement Learning (RL) algorithms on similar applications. We will focus on **atari based games** but all other applications could be used.

If you are lucky:

```
python3 -m pip install gym
```
and

```
python3 -m pip install gym[atari]
```
If you encounter issue in the installation with Windows, try these 2 solutions (thanks Kienan Bachwa):

- atari fork: https://stackoverflow.com/questions/42605769/openai-gym-atari-on-windows
- Or the WSL solution:
  If you don't have the WSL you can find it on the Microsoft Store (install Ubuntu or the other linux). Then you can launch Xming (**https://sourceforge.net/projects/xming/**) on windows,
  type export DISPLAY=:0 on your WSL and then you can launch anything graphical on your WSL and it will be displayed in Windows.

If you get a missing file error during import atari_py (thanks to Amine Legrifi):

- Try to do this : uninstall atari_py
- and try : pip3 install --no-index -f https://github.com/Kojoley/atari-py/releases atari_py

2. Let's play

No more than 15 minutes if you want to have time for completing this work.

```
import gym
from gym.utils.play import play

env = gym.make('Pong-v4')
env.reset()
play(env, zoom=3, fps=12)

env.close()
```
You will easily discover that you are the green player and you need to use keys 'a' and 'd' to move the paddle up or down. Modify zoom parameter in the code to be at ease (the value will depend on your screen resolution; 3 is generally a good value) and fps (frame per second) depending on your caffeine level.

3. Record games

If we want to learn from played games, we need to register some **X** (data) and **y** (labels) values that will be used as training data for a neural network.

The play function we used previously admits another parameter *callback* whose value is a function that will be called at each step of the game:

```
gym.utils.play.play(env, zoom=3, fps=12, callback=mycallback)
```
You'll need to write the function *mycallback* with the following specific signature:

```
def mycallback(obs_t, obs_tp1, action, rew, done, info):
```

labels of data, for training

Using a very simple callback function:

```
def mycallback(obs_t, obs_tp1, action, rew, done, info):
    print("action = ", action, " reward = ", rew, "done = ", done)
```
play again and observe the values of action, rew and done parameters.

You should now have an idea for the values to put in the vector y (labels for training).

data X for training

Now, let's focus on X values.

obs_t is the array of pixels of the image you see on the screen. You can try and add this line in mycallback function (without forgetting to *import imageio* in the headlines):

```
    imageio.imwrite("jeu.jpg",obs_t)
```
Of course, you will obtain only one image, for last state you have seen on your screen. Play shortly to obtain one image and observe the dimensions of the image.

What will be the dimension of one sample in the X vector ? Post your results on #si4-ia channel in the slack.

You probably observed that the dimension is very high. We should therefore reduce the size:

- among the 3 different color channels, could you take only one ? which one ?
- are every pixels needed in the screen ?
- try to downsample the pixels as much as possible but saving the ball position
- crop the array of pixels

Try and explain all numbers in this line of code:

```
    imageio.imwrite('outfile.png', obs_t[34:194:4,12:148:2,1])
```
Publish answers on slack. If you agree with a previous answer, just comment by :+1: otherwise, comment by :-1: and publish your answer in the same thread.

select the information to be saved

- will you need bare images? how to include temporal information?
- will you need to use all the episode data in case of game lost?

It might be useful to shuffle the data so that they are less consecutively correlated.

Recording games

You need to save into files the games you have played. Here is an example of how to save the values of pixels in a text file (other options like saving the object using pickle are also to be considered):

```
 with open('X.txt', 'a') as outfileX:
        np.savetxt(outfileX,delimiter='', X=obs_t[parameters you choosed], fmt='%d')
```

You also need to save the y (labels) values.

Now, it's time to play as best as possible ! Keep these training files as you will use them this week AND next week.

4. Teach your agent how to play

For each state, you have chosen an action up, down or nope. Let's train your ANN.

Training

Now, you will build a neural network that will choose the action depending of the state.

- input: image saved previously
- output: action
- number of layers: start with 1 or 2
- number of neurons: try different values, like 10, 50, 100, 500, etc.
- nature of layers: first layer could be dense or conv2d
- preprocessing of input data: normalise values, reshape the samples according to your first layer (1d if dense, 2d if conv2d, ...)

Testing

The test is in two steps:

1. do not use all your data for the training; spare 10% or 20% for testing. How are the performances ?
2. play new games with your ANN agent playing the green player. Is he a good player ? Who won ?

Technically, you already know everything to answer to step 1. For step 2, you need to specify your action.

Here is an example of random game:

```
env = gym.make('Pong-v4')
env.reset()
while True:
    env.render()
    obs,rew,d,inf=env.step(env.action_space.sample()) # take a random action
    if rew != 0:
        print("reward: ", rew)

env.close()
```

It will stop at the end of a set of 21 games. In the loop, you need to predict, from *obs*, the action to be performed at the next iteration. Observe the scores for several games.

Further directions (not optional for students who took "CVML" or "From shallow to DL" in Semester 1)

You can explore different ways:

- Network topology: plot the accuracy when varying the topology (number of layers, type of layers, number of neurons per layer)

- Compare the performance of the network against the performance of the agent, and estimate the correlation
- What is the minimal size of data to allow efficient learning and generalisation?
- Try and use another estimator (use sklearn)