AI Lab - Lesson 6

Deep Reinforcement Learning

Davide Corsi
Alessandro Farinelli

University of Verona
Department of Computer Science

January 20^{th} 2022

UNIVERSITÀ
di VERONA
Dipartimento
di INFORMATICA

Start Your Working Environment

Start the previously installed (lesson 1) conda environment *ai-lab*

Listing 1: Update Environment

```
cd AI-Lab
git stash (NB: remember to backup the previous lessons before this step!)
git pull
git stash pop
conda activate ai-lab
pip install tensorflow
pip install keras
jupyter notebook
```

Listing 2: Open Lesson

To open the tutorial navigate with your browser to:
lesson_6/lesson_6_problem.ipynb

Keras

What is it

Keras is a high-level neural networks APIs. It is written in Python and supports multiple back-end neural network computation engines:

- *built on top of TensorFlow 2.0*
- *optimized to work both on CPU and GPU*
- *simple functions to create, train and modify neural networks with state of the art architecture*

What is it for

Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible library.

Where to find it

https://keras.io/

Assignments

- Your assignments for this lesson are: *lesson_6/lesson_6_problem.ipynb*. You will be required to implement Deep Reinforcement Learning Algorithms, in particular the main loop and the "train" function
- In the following you can find the pseudocode

Deep Q-Learning

```
Input: $environment, neural\_network, trials, expl\_param, score\_queue$
Output: $neural\_network, score\_queue$
initialize the experience buffer                              ▷ A fixed size queue
initialize the score queue                                    ▷ An infinite size queue
for $i \leftarrow 0$ to $trials$ do
    initialize s observe current state
    repeat
        Select and execute action a                           ▷ $\epsilon$-greedy approach
        Observe new state s' and receive immediate reward r
        Add (s, a, s', r) to experience buffer
        TRAIN_FUNC($neural\_network, experience\_buffer$)
        update state s ← s'
    until $s$ is terminal
    update $score\_queue$
    if $score\_queue[i] > goal\_score$ then
        break loop
return $neural\_network, score\_queue$
```

Train Function

```
Input: $neural\_network, experience\_buffer(MB), gamma$
Output: $neural\_network$
Sample mini-batch MB of experiences from buffer
for $s, a, s', r \in$ MB do                                ▷ (state, action, next_state, reward)
    target $\leftarrow$ PREDICT($neural\_network, s$)
    if s' is terminal then
        target[a] = r
    else
        max-q = max(PREDICT($neural\_network, s'$))        ▷ max q-value from s'
        target[a] = r + (max-q * gamma)
    neural_network $\leftarrow$ FIT($neural\_network, s, target$)    ▷ back-propagation
return $neural\_network$
```