Homework 5

Francesco Grimaldi, 1206035
Department of Mathematics, University of Padua

Neural Networks and Deep Learning
Professor: Dr. Alberto Testolin
Assistant: Dr. Federico Chiariotti

January 2020

1 Introduction

This report describes the work done for the 5th home assignment of the course in Neural Networks and Deep Learning.

In this homework we used *Q-learning* and *SARSA* algorithms, given in the *Lab06* code, to perform Reinforcement Learning.
In particular, an agent has been trained to reach a goal in an environment with *obstacles* (objects that cannot be passed over) and *sand* (objects that can be passed over but with some penalization).

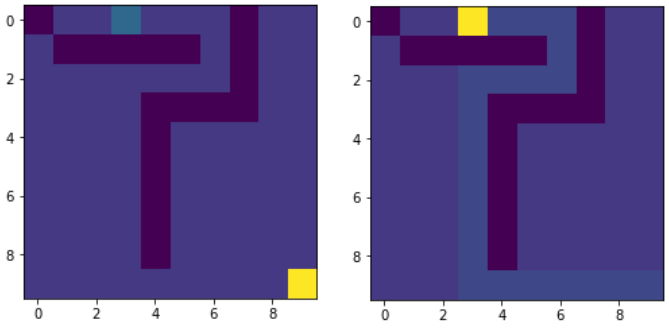

Fig 1. We can see the starting and ending environments. In yellow we have the agent, in dark blue the obstacle, in blue the terrain and in light blue the goal (in left images) and the path done (in the right image).

2 First environment: Obstacles

We started by creating an environment with only obstacles. These obstacles were implemented in such a way that choosing an action that lead to the encounter of an obstacle brings two consequences:

1. The agent doesn't perform the action but remain in the previous position;
2. The reward for that decision is *-1*.

After tuning a bit the hyper-parameters alpha ($\alpha = 0.5$) and epsilon ($\epsilon = 0.1$) the agent, trained with 3000 epochs, managed to reach its goal by avoiding the obstacles (see *fig.1*, *fig.2*).

The algorithm used is *Q-Learning*, but it must be said that, between *SARSA* and *Q-Learning*, the differences in the results were none.

Please see *obstacle_only.gif* in the *GIFS* folder to see the animated version of *fig.2*.

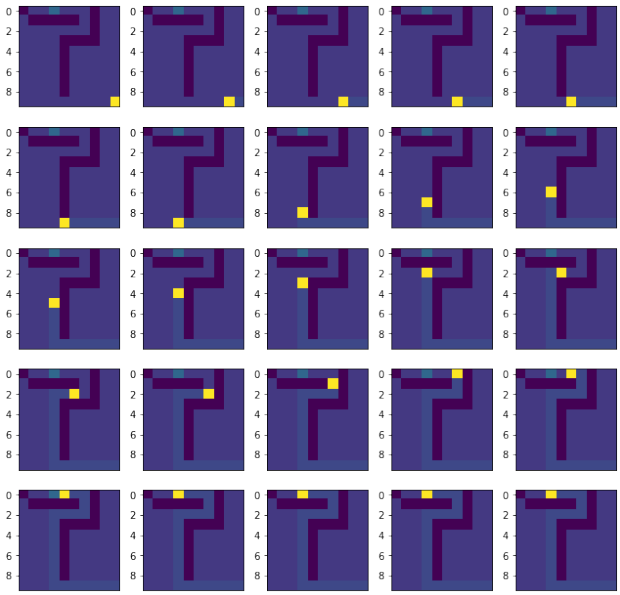

Fig 2. Path of our agent (yellow). We can see that it has followed the optimal path.

3 Second Environment: Obstacles and Sand

A second environment has been created by adding the *sand* object. As explained in the introduction, the sand object can be crossed, but it leads to a negative reward.

The idea is that the *agent* will decide to cross the sand only if it will make the path shorter enough.

3.1 Low penalization factor

The first test was done by using a reward for crossing the sand of value *-1*.

Good training parameters were, as before, $\alpha = 0.5$ and $\epsilon = 0.1$ while the number of epochs has been increased to 5000.

Both *SARSA* and *Q-Learning* have led, approximately, to the same final *average reward* of 0.80, but in the end **SARSA with softmax** has been chosen since it led to faster convergence (see *fig.3*).

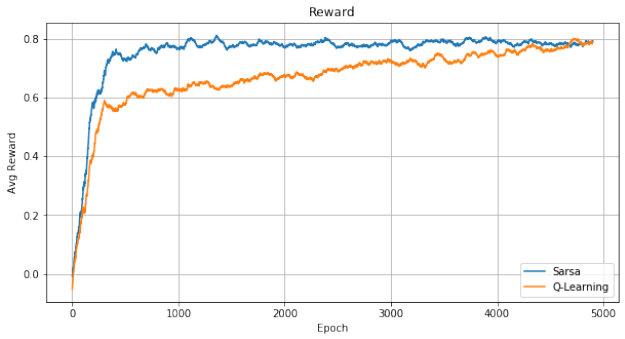

Fig 3. Average reward by number of epochs of an Agent implementing SARSA (blue) and Q-Larning (orange).

3.1.1 Sand avoidance example

The agent, after 5000 epochs of training and with a negative reward of minus 1 for crossing the sand, have proven to have learned the optimal path and to avoid the sand if this wouldn't have led to too much longer path (see *fig.4*).

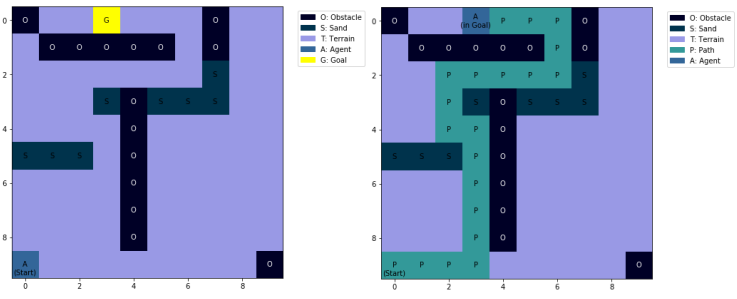

Fig 4. Example of path. Left, we can see the starting situation and right, we can see the ending situation where the agent has reached its goal and it has decided to circumnavigate the sand in position (3, 3).

A *gif* showing the actions taken in the example illustrated in *fig. 4* is in the *GIFS* folder under the name *low_penalization_sand_avoided.gif*.

3.1.2 Sand crossed example

The agent also managed to learn to cross the sand when this led to a much shorter path (see *fig. 5*).

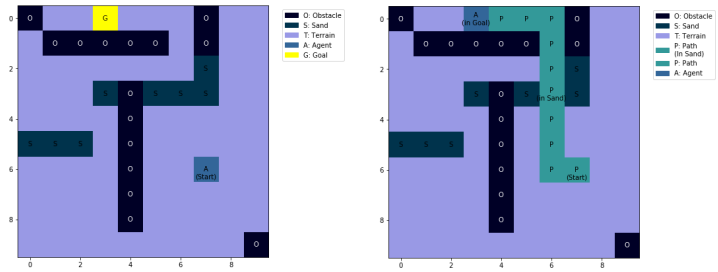

Fig 5. Example of path for an agent which has decided to cross the sand

The *gif* showing the actions taken in the example illustrated in *fig. 5* is in the *GIFS* folder under the name *low_penalization_sand_crossed.gif*.

3.2 High penalization factor

For this second test a negative reward for crossing the sand of value *-5* was used, while the same hyper-parameters ($\alpha = 0.5$, $\epsilon = 0.1$) and the same algorithm (**SARSA with softmax**) were kept. See *fig. 6* for the average reward in function of episodes of SARSA and Q-Learning.

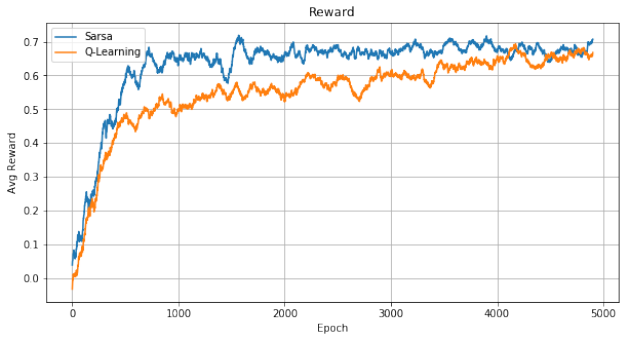

Fig 6. Average reward of SARSA (blue) and Q-Learning (orange).

If the starting position was the same of *fig. 4* the same path occurred, because obviously the agent was encouraged even more to avoid the sand.

In the second case, illustrated in *fig. 5*, this time the agent preferred a different strategy (see *fig. 7*), where it chooses to take the longer path and to avoid the sand at every cost.

This policy of avoiding the sand of all cost occurred even if the total reward at the end was less than the one given by an optimal path. One hypothesis of this behaviour is that the agent didn't manage to see long term rewards.

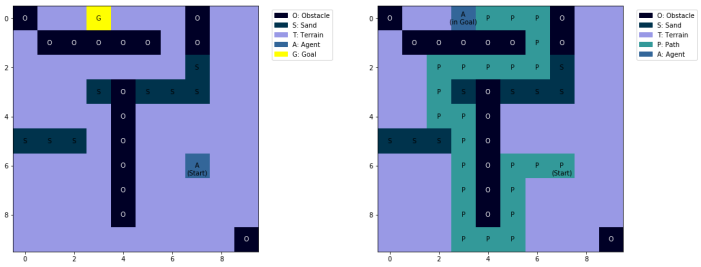

Fig 7. Example of an agent that avoid the sand if the penalization factor is set too much high.

The *gif* showing the actions taken in the example illustrated in *fig. 5* is in the *GIFS* folder under the name *high_penalization_sand_avoided.gif*.

4 Guide to the code

The zip file contain 5 files and the GIFS folder where all the gifs produced are saved:

1. *agent.py* is the script to build the agent (almost unchanged from the one given in the lab);
2. *environment.py* is the script to build the environment with obstacle and sand;
3. *training.py* is the script that run the training;
4. *matrixlib.py* is a script for some graphic functions;
5. this pdf file.

The *training.py* train an agent to reach the goal in position (0, 3). The training consists of 1500 epochs and uses SARSA with softmax. The position of sand and obstacle is fixed but can be manually changed inside the training.py like all the other parameters.

The training.py will print the average reward of the last 100 episodes every 100 episodes and, as an output, will create a *trained_model* folder where inside can be found 2 files:

1. *learner.obj*: the saved agent object that can be retrieved for eventual test;
2. *log.txt*: a file where for every episodes is reported the average reward and the initial random position.