Homework 5

Anna Braghetto - 1205200

22/02/2020

Introduction

The aim of the proposed homework is to explore the reinforcement learning algorithms and to train and test an agent using Python. The behaviour of the agent is analyzed when trained by using *SARSA* and *Q-learning* with different policies and discount values on a costumized environment. In particular, the hyperparameters for exploration policies are tuned.

Environment

The environment consists into a grid world, defined as a 10x10 matrix, containing different objects.

- **Goal**: it is the destination of the agent and its reward is +1.
- **Wall**: it is an obstacle that cannot be traversed and its reward is -1.
- **Sand**: it is an obstacle that can be traversed but leads to a penalty and its reward is -$\frac{1}{2}$.
- **Portal**: it is an object that permits to teleport in another portal of the grid and its reward is $-\frac{1}{4}$. There are just 2 portals in the map.

The environment is fixed during the whole training and its representation is shown in Fig. 1.

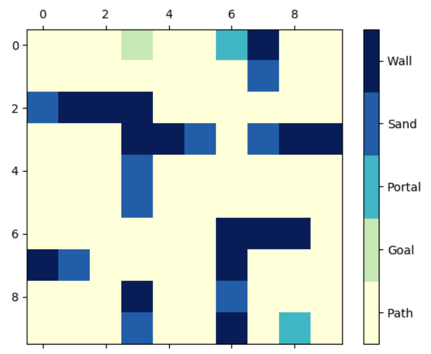


Figure 1: Environment. It is possible to observe the different objects inside the environment: goal in (0,3) in light green, walls in many coordinates in dark blue, sands in many coordinates in blue and portals in (9,8) and (0,6) in light blue, while the path is represented in yellow.

Agent

The agent is traned in the environment where it can move using 5 basic actions: stay, move up, move down, move right and move left. Its final aim is to reach the goal of the environment.

The initial coordinates, corresponding to the initial state of the agent, are chosen randomly, while the selection of the next action, among the available five, and the update of the q-table depend on the policy and on the algorithm considered to train the agent.

Training

The training of the agent is performed considering different algorithms and policies: SARSA and Q-learning algorithms, softmax and ϵ-greedy policies. SARSA is an on-policy algorithm, thus the considered policy refers to both behaviour and update one, while Q-learning is an off-policy algorithm, thus the considered policy just refers to the behaviour policy; the update one is greedy. Comparing the two policies; the ϵ-greedy policy allows to choose the best action (with higher q-value) with probability $1-\epsilon$

and any another action with probability ϵ, while through the softmax policy, the probability to choose an action depeds always on its q-value.

For a fixed algorithm and policy, the initial state (coordinates) of the agent is chosen randomly and it can move inside the environment for a number of steps fixed to 50; when it reaches the *goal*, it stays there until the end. Then, each episode, with a random initial state, is repeated for 2000 times.

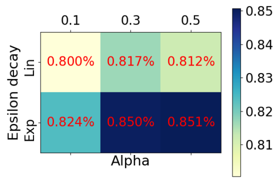
Tuning of the Hyperparameters

At first, a tuning of the hyperparameters for the exploration policies, ϵ and α, is performed by fixing the value of the discount to 0.9. Due to the random extraction of the initial states, the learning is stochastic, thus the rewards are averaged over 10 runs.

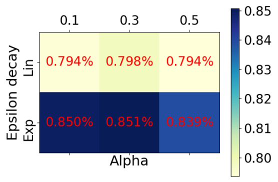
The tuning is performed on the learning rate α, fixed during the training, for two different ϵ schedules: linear decay and exponential decay. In particular, the initial value for ϵ is set to 0.8 to assure exploration at the initial phases. The considered combinations analyzed through the search are collected below.

- **Algorithm**: Q-learning and SARSA
- **Policy**: ϵ-greedy and softmax
- **Decay of ϵ**: linear and exponential from 0.8 to 0.001
- **Learning rate α**: 0.1, 0.3 and 0.5

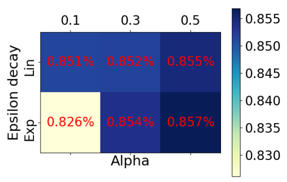
The average reward over the last 100 episodes, averaged over 10 runs, is shown for each configuration of parameters, for discount set to 0.9, in Fig. 2.



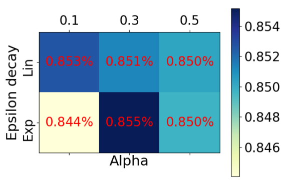
(a) Q-learning and ϵ-greedy policy.



(b) Sarsa and ϵ-greedy policy.



(c) Q-learning and softmax policy.



(d) Sarsa and softmax policy.

Figure 2: Results of the tuning of the parameters. It is possible to observe that, for both the algorithms, the kind of decay of the parameter ϵ does not affect the reward in case of softmax policy even if it is considered in the computation (see code), while the exponetial decay shows an improvement in the reward, compared with the linear one, if the the ϵ-greedy policy is considered.

Thanks to a systematic search, the best learning rate for each schedule of the parameter ϵ, considering all the combinations of algorithm and policy, is defined and the results are collected in Tab. 1.

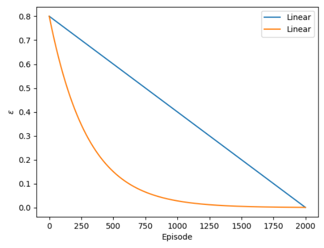
Algorithm	**Policy**	**Decay of ϵ**	**Learning rate α**
Q-learning	ϵ-greedy	Linear	0.3
Q-learning	ϵ-greedy	Exponential	0.5
Q-learning	Softmax	Linear	0.3
Q-learning	Softmax	Exponential	0.3
SARSA	ϵ-greedy	Linear	0.5
SARSA	ϵ-greedy	Exponential	0.5
SARSA	Softmax	Linear	0.1
SARSA	Softamx	Exponential	0.3

Table 1: Best learning rate for each configuration.

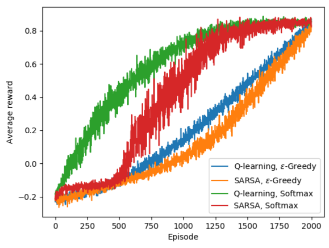
Final Training

Defined the best learning rate for each configuration, the final training with the given parameters are performed as defined above.

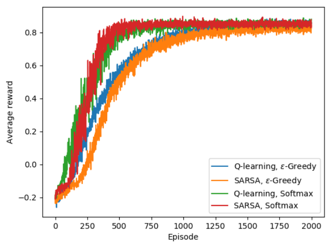
In particular, to compare well the results, the evolution of the average reward for 30 runs, considering all the configurations, is defined and shown, for both the kind of decay, in Fig. 3.



(a) ϵ decay.



(b) Linear decay.



(c) Exponential decay.

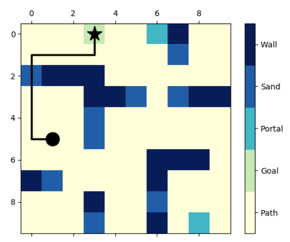
Figure 3: Evolution of ϵ (on the left) and of the average reward for linear (on the center) and exponential (on the right) decay of ϵ. In both cases, for the ϵ-greedy policy, the convergence is slower than softmax policy, while fixing the policy and comparing the algorithms, it is possible to observe the Q-learning reaches the convergence faster than SARSA due to the fact that the first is an off-policy algorithm and the second an on-policy one. Comparing the two cases, the exponential one reaches the convergence faster than the linear: the exponential decrease of ϵ permits to perform a shorter, but enough, inital exploration phase, reducing then the randomness of the selection of the action (both algorithm) and of the update (just for SARSA).

As shown above, the best schedule for ϵ decay is the exponential decay.

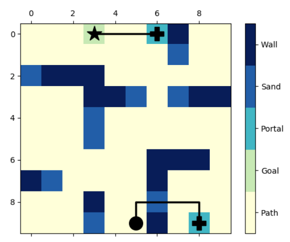
Test

Finally, it is interesting to show how the trained agent moves inside the environment.

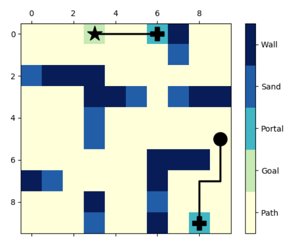
As example, the test is performed just for the exponential decay of ϵ, and considering all the algorithms and policies. In particular, the training is performed just for one run and after it, the agent is tested setting ϵ to 0, in order to avoid random action selections; the results, for different initial conditions, with discount factor set to 0.9, are shown in Fig. 4.



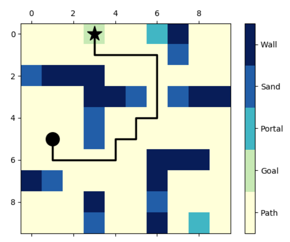
(a) Initial state at (5,1): Q-learning and ϵ-greedy policy.



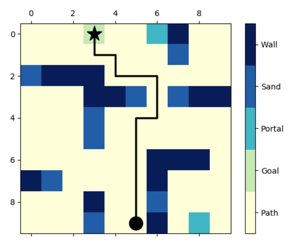
(b) Initial state at (9,5): Q-learning and ϵ-greedy policy.



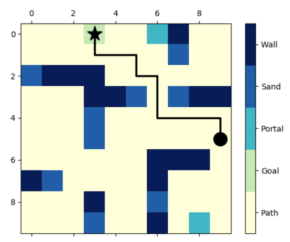
(c) Initial state at (5,9): Q-learning and ϵ-greedy policy.



(d) Initial state at (5,1): SARSA and ϵ-greedy policy.



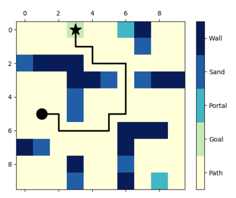
(e) Initial state at (9,5): SARSA and ϵ-greedy policy.



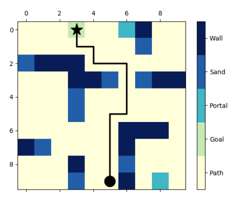
(f) Initial state at (5,9): SARSA and ϵ-greedy policy.

Figure 4: Path for differerent initial states with discount set to 0.9. Neverthless all the configurations are tested, just two cases are shown. The results relative to Q-learning with ϵ-greedy policy are shown in (a), (b) and (c) because for the other combinations, except for SARSA with ϵ-greedy policy, the results are similar; showing that the agent takes the shortest way to reach the goal, even if it has to pass through sands or portals. The results relative to SARSA with ϵ-greedy policy are shown in (d), (e) and (f), showing that it takes less risks avoiding the sands and the portals, because it is an on-policy algorithm and the considered policy is ϵ-greedy, as explained above.

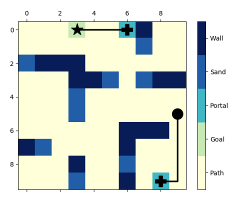
Shown the results for discount set to 0.9, it is also interesting compare the behaviour given a lower discount factor. Thus, the test is performed also setting it to 0.7: it is observed that the reach of the convergence is more difficult that considering an higher discount factor. The results are shown in Fig. 5.



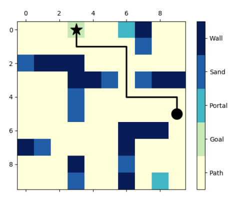
(a) Initial state at (5,1): Q-learning and ϵ-greedy policy.



(b) Initial state at (9,5): Q-learning and ϵ-greedy policy.



(c) Initial state at (5,9): Q-learning and ϵ-greedy policy.



(d) Initial state at (5,9): SARSA and ϵ-greedy policy.

Figure 5: Path for differerent initial states with discount set to 0.7. Just two cases are shown, since for Q-learning with ϵ-greedy, Q-learning with softmax and SARSA with softmax, the results are similar as the ones shown in (a), (b) and (c): the agent avoids sands but passes through portals. The results relative to SARSA with ϵ-greedy policy are similar to the ones shown in (a) and (b) with initial states at (5,1) and (9,5), while for the initial state (5,9) shown in (d), the agent avoids all the obstacles in the environment, as observed in Fig. 4.

Setting the discount factor to 0.7 it is possible to observe that the agent never passes through the sand due to the fact that it gives an higher *weight* to present rewards. While, by increasing the discount factor to 0.9, it is possible to notice that the agent gives more importance to long-term rewards, allowing the passage through the sand (just for the first combinations of algorithm and policy, as explained above).

Finally, it is observed, as explained above, that the results are really stochastic due to the dependence on the random initial conditions considered in the training.