Q-learning algorithms applied to Ball Balancing & Two Link Rigid Manipulator

Ettore Maria Celozzi
E-mail address
ettore.celozzi@stud.unifi.it

Luca Ciabini
E-mail address
luca.ciabini1@stud.unifi.it

Abstract

In this summary, the experimentation of the hysteretic Q-Learning approach for Reinforcement learning is presented. Two different examples are used and for those a comparison between centralized, decentralized and hysteretic algorithms is done.

1. Introduction

A **Multi Agent System** (MAS) is a set of multiple, interacting, intelligent agents that share the same environment. Nowadays this kind of systems have high relevance, indeed there is a broad collection of problems for which the presence of one agent is not enough to find a solution.
The problems analyzed here are two examples that belong to this collection: **Ball Balancing** and **Two Link Rigid Manipulator**.
One of the former attributes of the agents is very important: **intelligence**. The agents have to control the system choosing some actions, and this choice have to be smart enough to achieve the problem goal.
In order to educate the agents, the **reinforcement learning** paradigms is used. In this context, through a reward, that the agents want to maximize, is possible to teach the agents on how to solve the problem. The algorithm used to learn the policy is the **Q-Learning** algorithm, a model-free[1] reinforcement learning algorithm. The functions used to calculate the quality of the state-action combination is:

$$Q : S \times A \rightarrow \mathbb{R} \tag{1}$$

where S and A represent the set of states and actions respectively. Given some states ($s \in S$), the execution of some actions ($a \in A$) provide to the agents a reward and result in a transition to another state.
To remember the history of the choices (and the reward associated) a matrix called **Q-table** is used. This is updated in each step of the training phase and then used by the agents, in the test phase, in order to choose the best action and reach the goal.

2. Q-Learning algorithms

There are different type of Q-Learning algorithms, the first distinction is between the **centralized** and **decentralized** implementation. In the first case the agents of the system are controlled by one "agent" that chooses the action for all the agents and the result of this choices is saved in a Q-table that has a dimension equal to $|S| \times |A|^{|C|}$ where C is the set of the controls.
For the decentralized implementation instead, each agent has his own Q-Table (dimension: $|S| \times |A|$) and chooses his own actions.
Even though the centralized case is easier to implement, the space required grows exponentially, so could be prohibitive to apply for some problems. These issues are solved by the decentral-

[1] It does not require a model of the environment

ized approaches.
The decentralized approach has different implementations, here two of these types are presented: **Decentralized** and **Hysteretic**.
All the algorithm have two parameters:

- **learning rate**: $\alpha \in]0; 1]$
- **discount factor**: $\gamma \in [0; 1[$

The last one determines the importance of future reward; a factor of 0 will make the agent shortsighted because only current rewards are considered, while a factor approaching 1 will make it strive for a long-term high reward.

2.1. Centralized

The main difference between the implementations depends on how the **Q-value** is computed. This is the value that is stored in the Q-table.
[1] In the centralized case the update rule is:

$$Q(s, a_1, ..., a_n) = (1 - \alpha)Q(s, a_1, ..., a_n) + \alpha[r + \gamma \max_{a'_1,...,a'_n} Q(s', a'_1, ..., a'_n)] \quad (2)$$

As explained before, in this case the Q-value depends on all the actions since there is a "central" agent that take the decisions for the others. The notation s', a' means new state and new actions respectively.

2.2. Decentralized

Although the decentralized name is used to represent a family of algorithms, here it represents the algorithm that is similar to the centralized case, but with multiple Q-tables:

$$Q_i(s, a_i) = (1 - \alpha)Q_i(s, a_i) + \alpha[r + \gamma \max_{a'_i} Q(s', a'_i)] \quad (3)$$

In this case we have one Q-table for each agent, hence the q-value depends just on the action chosen and associated to the i-th agent.

2.3. Hysteretic

[1] The idea behind this algorithm is that an agent can be punished because of a bad choice of the team even if it has chosen an optimal action. Then the agent had better to attach less importance to a punishment received after the choice of an action which has been satisfying in the past. This leads to an update rule of the form:

$$\delta = r - Q_i(s, a_i)$$
$$Q_i(s, a_i) = \begin{cases} Q_i(s, a_i) + \alpha\delta & \text{if } \delta \geq 0 \\ Q_i(s, a_i) & \text{else} \end{cases} \quad (4)$$

The problem with the former implementation is that the agents do not care to achieve the coordination between multiple optimal joint actions. Agents must not be blind to penalties at the risk of staying in sub-optimal equilibrium or miscoordinating on the same optimal joint action. In order to solve this issue two learning rates are used, according to the result of a joint action.

$$\delta = r + \gamma \max_{a'} Q_i(s', a') - Q_i(s, a_i)$$
$$Q_i(s, a_i) = \begin{cases} Q_i(s, a_i) + \alpha\delta & \text{if } \delta \geq 0 \\ Q_i(s, a_i) + \beta\delta & \text{else} \end{cases} \quad (5)$$

Where $\beta \in]0, 1]$, is another learning rate (usually much lower than α).

3. Problem 1: Ball balancing

The first problem is showed in fig 3: given two robots that can control the height of a plane on which a ball is placed, the aim is to keep the ball balanced in the center of the plane without make it falls.
So both the robots can interact with the environment setting, the heights $h1$ and $h2$, and takes actions based on the current state given by the position and speed of the ball $s = (x, \dot{x})$.
The system is described by the dynamic:

$$m\ddot{x} = -c\dot{x} + mg(\frac{h_1 - h_2}{l}) \quad (6)$$

where m is the weight, g the gravity acceleration and l the length of the plane.
The used reward is:

$$r = 0.8e^{-\frac{x^2}{0.25^2}} + 0.2^{-\frac{\dot{x}^2}{0.25^2}} \quad (7)$$

that penalized the states far from zero.

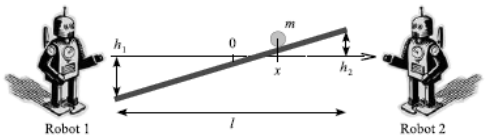

Figure 1. Ball balancing

3.1. Experiments

As suggested in [1] the state space and the action space has been discretized in the following way:

- The sample time is 0.03 seconds
- For the state space the position can assume 100 values between $-1, 1$ while the speed 50 values between $-3, 3$
- The actions can assume 15 values between $-1, 1$

This discretization leads to a Q-table with dimension: $100 \times 50 \times 15^2$ for the centralized case or to two Q-tables with dimensions $100 \times 50 \times 12$ in the decentralized and hysteretic case.

The training stage is done by letting the system go for 20 seconds for 5000 times called trials.
In each trial the starting state of the system is $s = (0.5, 0.1)$. The new action is then calculated using the $\epsilon - greedy$ approach with an ϵ that goes from 0.8 (at the start) to 0.1 (at the end).
Given the new action is possible to extract the acceleration of the ball (using the dynamic) and consequently the new state of the system and the reward.
After that, the Q-table/s is/are updated following the rules of the different algorithms with $\alpha = 0.9$, $\beta = 0.1$, $\gamma = 0.9$ and another trial start.

The training stage is done 5 times and as a final result the average of the reward sum from each trial (meaning the sum of all the rewards of a certain trial) is saved.

3.2. Results

The results for centralized, decentralized and hysteretic approach can be seen in 2, 3 and 4.
The results perfectly reflects those in [1] with the difference that, due to a lower number of training execution (5 vs 20), there is more noise. We can see from the sum of reward a better convergence for the hysteretic approach w.r.t the other two methods.
Also decentralized and hysteretic offer a much faster and space-efficient way to reach the results also due to the dimension of the Q-table.
All the states evolution charts shows that the values converges after few seconds to the requested values.

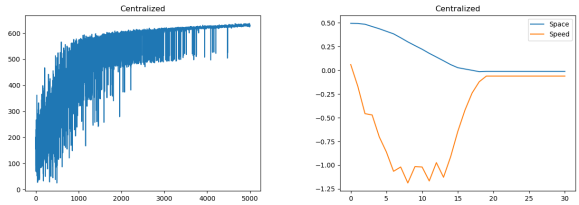

Figure 2. Sum of rewards (left) and states evolution for centralized approach

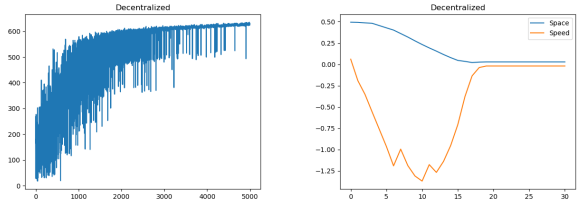

Figure 3. Sum of rewards (left) and states evolution for decentralized approach

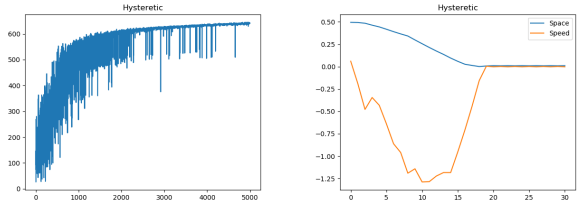

Figure 4. Sum of rewards (left) and states evolution for hysteretic approach

4. Problem 2: Two Link Rigid Manipulator

In this section the algorithms are applied to the **Two Link Rigid Manipulator** depicted in Fig 5. For this kind of problem there are two agents and four states: the angles θ_1, θ_2 and the angular speeds $\dot{\theta}_1$, $\dot{\theta}_2$. The controls are τ_1 and τ_2, for the two joints respectively. The system is described by the nonlinear fourth-order model:

$$\tau = M(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} + G(\theta) \tag{8}$$

where $M(\theta)$ represent the mass matrix, $C(\theta, \dot{\theta})$ the Coriolis and centrifugal forces and $G(\theta)$ the gravity vector.

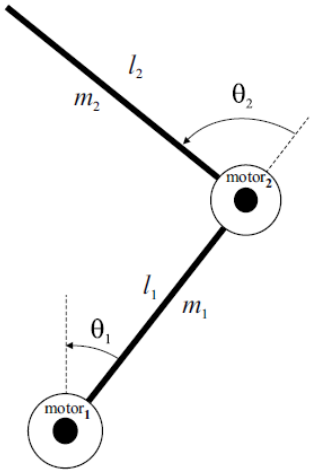

Figure 5. Two Link Rigid Manipulator

As a consequence of some simplifications the dynamics of the manipulator can be expressed as:

$$x_1 = \frac{1}{P_2(P_1 + P_2 + 2P_3 cos\theta_2)}$$
$$x_2 = [P_2(\tau_1 - b_1\dot{\theta}_1) - (P_2 + P_3 cos\theta_2)\tau_2]$$
$$\ddot{\theta}_1 = x_1 \cdot x_2 \tag{9}$$
$$\ddot{\theta}_2 = \frac{\tau_2}{P_2} - b_2\dot{\theta}_2 \tag{10}$$

See [2] for further explanation.

The goal of this problem is the stabilization of the system around $\theta = \dot{\theta} = 0$ in minimum time, with a tolerance of $\pm 5 \cdot \frac{\pi}{180}$ rad for the angles and $\pm\, 0.1$ rad/sec for the speeds.

4.1. Experiments

Even for this problem a discretization technique have to be used because the problem is continuous.
The sampling time used is 0.05 and to discretize the states, the idea is the same of [2] and consists of interpolating continuous states through bins based on some fixed centers. The bins' centers are:

- $[-180; -130; -80; -30; -15; -5; 0; 5; 15; 30; 80; 130] \cdot \pi/180$ rad for the angles
- $[-360; -180; -30; 0; 30; 180; 360] \cdot \pi/180$ rad/sec for the speeds

The weights are computed as:

$$\mu_{s,n}(s) = \begin{cases} \max(0, \frac{c_{n+1}-s}{c_{n+1}-c_n}), \text{ if } n = 1 \\ \max[0, \min(\frac{s-c_{n-1}}{c_n-c_{n-1}}, \frac{c_{n+1}-s)}{c_{n+1}-c_n})], \\ \qquad \text{if } 1 < n < |S| \\ \max(0, \frac{s-c_{n-1}}{c_n-c_{n-1}}), \text{ if } n = |S| \end{cases}$$

Where s is one of the four states (θ_1, θ_2, $\dot{\theta}_1$ and $\dot{\theta}_2$) and n represent the n-th center.
To notice that there is no 'first' or 'last' bin since the angles centers evolve on a circle manifold, so the 'last' bin is neighbor of the 'first'.
The actions can assume 16 values between $-0.2, 0.2$ for τ_1 and 16 between $-0.1, 0.1$ for τ_2.
This discretization leads to a Q-table with dimension: $(12 \times 7 \times 16)^2$ for the centralized case and to two $12 \times 7 \times 16$ Q-tables in the decentralized and hysteretic case.

The training phase is done through 10000 trials each of which lasts 5 seconds. The starting state is $(-1, -3, 0, 0)$ and during each of these trials the actions are chosen with the same ϵ-greedy approach of the former problem (with the same range of values for ϵ, see 3.1).
For all the algorithms the values of the parameters are the same: α=0.9, β=0.1, γ=0.98.

4.2. Results

The results showed have been obtained with a **quadratic reward** of the form:

$$r = \theta^2 - 0.05 \cdot \dot{\theta}^2 \tag{11}$$

The figures 6, 7 and 8 show the states evolution after the training phase. The results are the same obtained in the former problem but, here, the differences between the centralized and decentralized approaches are more evident. The convergence of the hysteretic approach is the fastest with a huge difference with respect of the centralized approach, that is the slowest ($\simeq$ 420 iterations more) and quite close (see fig. 9) to the decentralized one ($\simeq$ 20 iterations).

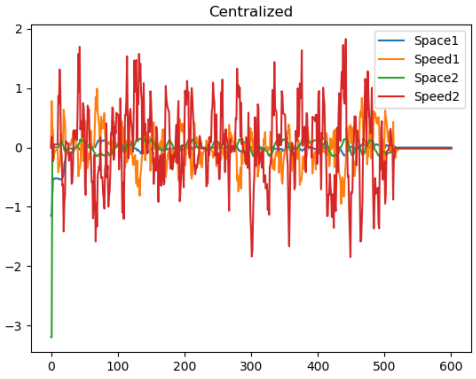

Figure 6. States evolution for centralized approach

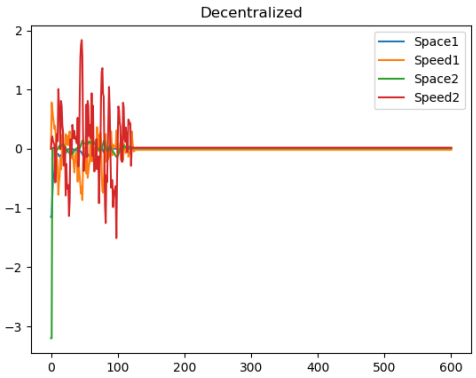

Figure 7. States evolution for decentralized approach

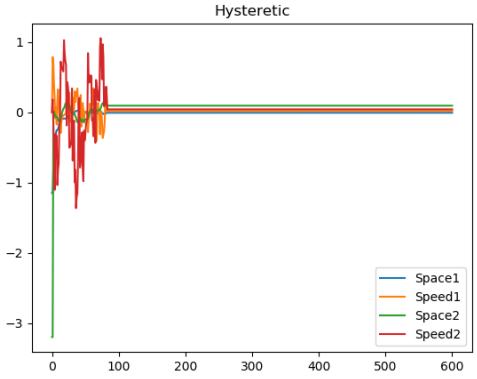

Figure 8. States evolution for hysteretic approach

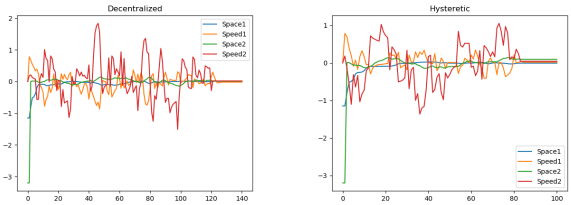

Figure 9. Focused comparison between decentralized (left) and hysteretic (right) approaches

As happened for the ball balancing example, the values are not exactly zero because of the discretization, so the speed is too small to cause a "visible" (for the discretization) change in the space that consequently stays stationary.
To solve the problem it would be possible to use a fine grained discretization but this would lead to a much bigger state-space.
Another aspect to keep in mind is that since we use a $\epsilon - greedy$ approach with a small number of trials (like in our case) different execution of the training can lead to a slightly different test results.

5. Conclusion

The example shown here have proved that through reinforcement Q-learning is possible to teach (multiple) agents such that they are able to accomplish one desired task.
The different Q-algorithms presented have pros and cons. At first there is an huge difference between centralized and decentralized (as category)

implementations, indeed the first one requires a Q-table that is much bigger than the decentralized one: $100 \times 50 \times 15^2$ vs $100 \times 50 \times 15$ (first example, see 3.1) and $(12 \times 7 \times 16)^2$ vs $12 \times 7 \times 16$ (second example, see 4.1). Furthermore the bigger space make more difficult the state space exploration, so for big space the convergence could be harder.
Despite the dimensions, the centralized method is easier to implement.

As showed with the tests, the hysteretic approach requires the lowest number of iterations to converge to the goal.

References

[1] N. L. F.-P. Laetitia Matignon, Guillaume J. Laurent. Hysteretic q-learning : an algorithm for decentralized reinforcement learning in cooperative multi-agent teams. *IEEE*, 2007.

[2] R. B. Lucian Busoniu, Bart De Schutter. Decentralized reinforcement learning control of a robotic manipulator. *IEEE*, 2006.