Exercise 2a

Ludovica Mazzucco
Department of Control and Computer Engineering
Politecnico di Torino
Italy
`s315093@studenti.polito.it`

Abstract: The purpose of this document is to show the results obtained from the study of the Linear Quadratic Regulator (LQR) first and Reinforcement Learning then control techniques applied to the Cart-pole agent, offered by the Gym environment.

Keywords: Exercise2a, LQR, RL

1 Introduction

The main focus of this report is to analyze and compare two different control techniques, LQR and Reinforcement Learning, evaluating their performance on the Cart-pole system offered by the Gym environment. It basically consists in a moving cart on the horizontal axis with a pole on top of it; the goal is to move the cart avoiding the pole to fall down. The state of the agent is a four-dimensional vector with components position and velocity of the cart and angle and angular velocity of the pole with respect to the vertical axis, like this:

$$state = \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix}$$

1.1 LQR: General Considerations

Before going deeper in to the explanation of the drawn results, an initial clarification about the mechanism behind the LQR control is needed.
So, the main idea is to design an input signal $u(t)$ which solves a minimization problem concerning a certain cost function J, that in this case is composed by the energy of the state weighted by a diagonal positive definite matrix Q and the energy of the input itself weighted by another diagonal positive definite matrix R. Putting things in a formally way:

$$u(t) = arg \min_{u(t)} \int_0^{\infty} (x^T Q x + u^T R u) dt \tag{1}$$

$$s.t.\ \dot{x}(t) = Ax(t) + Bu(t)$$

For given Q and R, the optimal solution to [1] is

$$u(t) = -Kx(t)$$

$$K = R^{-1} B^T P$$

Where P is the positive definite matrix satisfing the Algebraic Riccati Equation (ARE):

$$A^T P + PA + Q - PBR^{-1} B^T P = 0 \tag{2}$$

As it is highlighted by the formulas above, LQR control provides for a system which is linear, hence the matrices A and B are well defined (these are the dynamic and state-input matrices, respectively).

Therefore, in the implementation of the regulator for the cart-pole a linearized model is considered. Besides, the matrices Q and R could be designed in order to manage the trade-off between performance of control and energy consumption, i.e increasing R over Q would induce a ”energy-saving mode” with a limited input.

1.2 Reinforcement Learning: General Considerations

In what this second control technique is concerned, it is a sort of unsupervised learning in which an agent (the dynamic system, alias the cart-pole) is not provided with an outer command to follow as in LQR, nevertheless it is guided toward the desired behaviour through a reward if its actions are correct or a penalty otherwise. More precisely, the agent interacts continuously with the environment through a set of possible actions and it receives back from it the corresponding reward. The ultimate goal is, obviously, to maximize not also the immediate reward but even the future ones, possibly discounted by a factor γ if it aims at weighting differently the contribution of next rewards. It is paramount to notice that the whole process is stochastic, hence the feasible states of the agent follow a certain probability distribution as well as the actions that it could perform in a given state. In fact, it changes state according to a transition matrix and takes actions obeing to a so called policy π, which embodies a critic role in this context since ”learning” the best policy implies taking every time the correct actions.

2 Experimental Results

2.1 LQR: Application to the Cart-pole

In order to apply the LQR to the cart-pole, as it was clarified in section 1.1, the system has to be preliminarly linearized so that the matrices A and B can be exploited for the calculation of the optimal control gain K. Then, for each time step the algorithm loops as follows:

- evaluation of the control input u given K and the current observation vector (the state) by means of $action$ (0 or 1 depending on the sign) and $force$(its magnitude). This is due to the fact that the gym environment accepts only those values to be passed to the function encharged to apply the control input to the agent.
- passing control input to the agent.
- update the observation.

After 400 steps the evolution in time of the state variables are shown in the figures below.

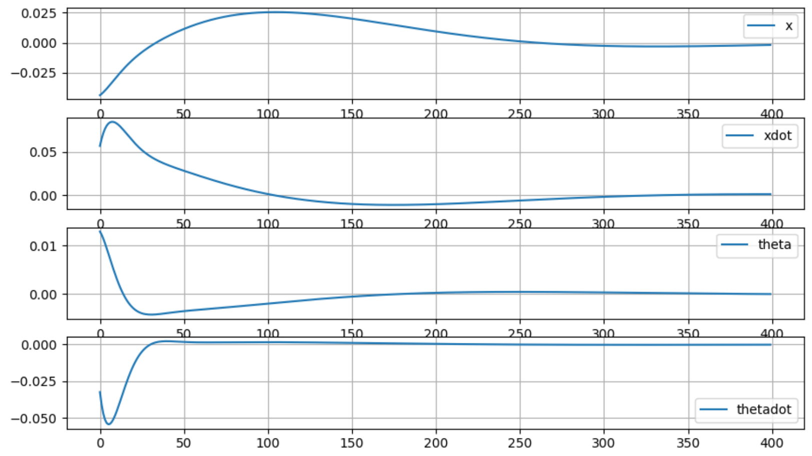

Figure 1: state variables over 400 steps

All the signals ranges are in the neighborhood of zero, more precisely, after t=23 all of them are included in the area 0 ± 0.05. This means that the regulator is acting properly to force every state variable to reach zero, in other terms, to let the cart-pole remain in the center of the box with the pole steadily up. This happens since appling such input signal the vector $obs = [0 \quad 0 \quad 0 \quad 0]^T$ corresponds to an asymptotically stable equilibrium.
To prove the assertion done in the introduction about the role of matrix R, saying it controls the energy of the input signal, it is reported here the plot representing the forces provided to the system for different values of R. It is not difficult to notice a decreasing trend as response to the increasing R, representing the fact that in this way the minimization of J acts harshly on them.

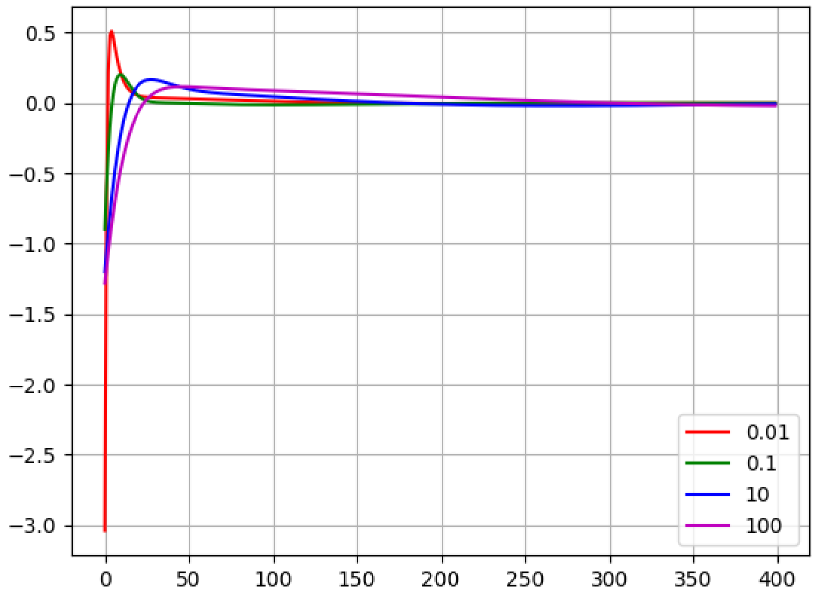

Figure 2: Forces corresponding to different values of R

2.2 Reinforcement Learning: Application to the Cart-pole

The cart-pole controlled in the last section with the LQR is now regulated by means of Reinforcement Learning following the main ideas explained in paragraph [1.2]. More in detail, it passes through two phases, training and testing, in which the cryterion followed is that for each episode (the number of episodes is chosen a-priori by the programmer) the observation is passed to a function returning the action the agent will take based on the running policy (0 or 1 depending on the direction of the force applied to the cart), after that the input is passed to the agent to actually perform that action and returning the reward obtained due to that action. It has to be remarked that episodes have not the same length because there are different termination conditions:

- The episodes reach the maximum number of time steps, tunable through the corresponding field in the ”env” object (500 by default);
- The pole angle exceeds $\pm 12°$ with respect to the vertical axis;
- The cart position surpasses ± 2.4.

Besides, the pre-defined reward function consists in the number of time steps during which the agent remains alive, in other words, the reward is +1 each time step the cartpole does not verify the exit conditions.
We said that the agent behaviour is driven by a certain policy that gives precise probabilities to perform an action rather then an other, if we try to impose a random policy substituting the one used by gym environment we will notice that on average the episodes last very little compared to the ones with the default version, since the cart continues to drift away or the pole to fall apart. Clearly, a random policy does not encourage the agent to learn by its experience, there is no evolution in its behaviour nor a rationale behind it. The technical results of the simulations are reported below:

- **Default policy**: Average test reward: 499.62 episode length: 499.62;
- **Random policy**: Average test reward: 168.59 episode length: 168.59.

We said that an exit condition for the episode is that the cart-pole runs out of time, nonetheless the field in env reporting the maximum number of time steps granted to the agent can be adjusted properly. This lets us decide to set different time boundaries in train and test phases. In particular, if $\_max\_episodes\_steps$ is fixed at 200 for the first and at 500 for the second, it emerges from the simulations that the cart-pole able to stabilize the pole during the training still balances it pretty well in the test, given that the agent does not learn a temporal law but it is stimulated to behave correctly through the reward it gets from what it does. Hence, once it understands which actions lead to the best income it does not matter how long the episodes last, what is crucial is that we pay attention that the training episode is long enough to allow it to effectively experience those. Definitely we can say that in 200 steps the cart-pole reaches the goal to maintain up the pole without too many oscillations and so, as it was said, even the test goes fine, however we have to take into account the stochasticity and non optimality of this kind of control that causes large variance in the results, as it is proved launching with the same settings five simulations one after the other:

1. Average test reward: 387.02 episode length: 387.02
2. Average test reward: 459.13 episode length: 459.13
3. Average test reward: 328.27 episode length: 328.27
4. Average test reward: 500.0 episode length: 500.0
5. Average test reward: 180.2 episode length: 180.2

This highlights how different the outcomes of each training session can be with respect to each other although the reward function always remains the same simple one of giving one every step the agent remains alive.
Another way we can drive the behaviour of the cart-pole with is to change the reward function. We would like to fulfill the following goals by designing proper functions:

1. Balance the pole close to the center of the screen;
2. Balance the pole in an arbitrary point of the screen;
3. Keep the cart moving from the leftmost to rightmost side of the track as fast as possible, while still balancing the pole.

The cryterion used to solve the above task is basically to honour the agent whenever it obeys to the expectations and penalizing it otherwise, giving it less or more "points". Moreover, since the objective is to maximize the overall return, it is not so relevant to assign negative rewards up to zero (when it assumes the desired configuration) rather than positive rewards where, instead, zero is given to bad decisions, so the latter way is considered to try to influence the agent.
In what regards the first point, the reward function used was

$$reward = 1 - ||state||_2^2$$

The idea here is to disregard all the configurations in which the observation vector assumes components different from zero, given that the goal is to keep the cart in the zero position, null velocity, no pole inclination nor angular velocity. This proved to work well, giving average test reward as 281 and episode length 495. Below is shown the performance during the training phase.

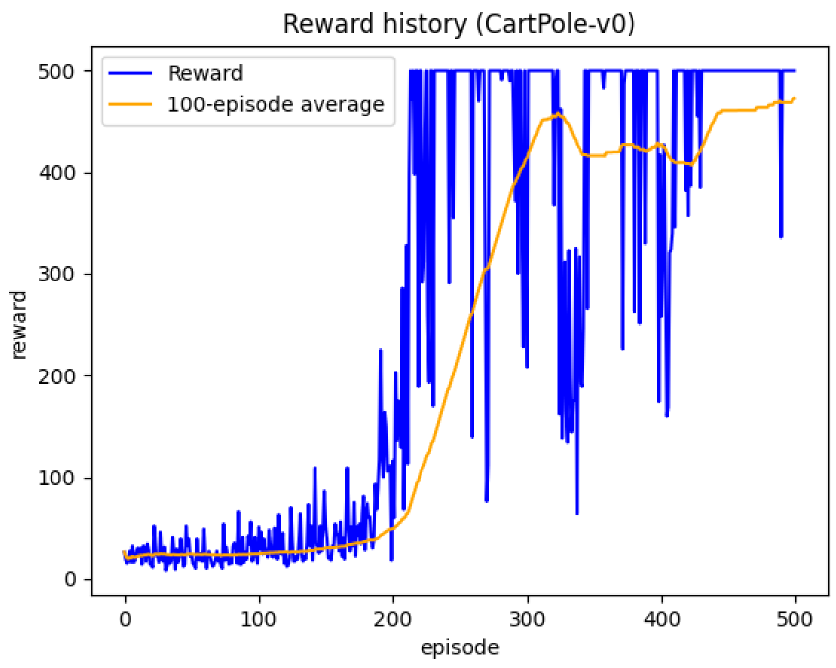

Figure 3: Training for the first reward function

The second reward function remarks the previous one since this case can be transposed to it considering as central point an arbitrary one instead of zero. Nevertheless, it has been slightly modified to increase the performance given that exploiting the 2-norm was not returning satisfing results; the zero-norm was used rather than the squared 2-norm. The mentioned formula is

$$reference\_state = [x_0 \quad 0 \quad 0 \quad 0]$$

$$reward = 5 - ||state - reference\_state||_0$$

The sense is that the agent must be pushed like the former case to a configuration in which the observation vector contains as many null components as possible, thus minimizing the zero-norm and maximizing the reward. Anyway it is very difficult to have all of them exactly set to zero since a little error is always present, that's why the formula has been considered with the 5 instead of 4 (the dimention of the state), in this way the agent receives 1 as reward even when the zero norm gives 4 as result, to repay it for remaining alive. Hence, the average test reward is 474 as the episode length while the training performance is shown in this figure:

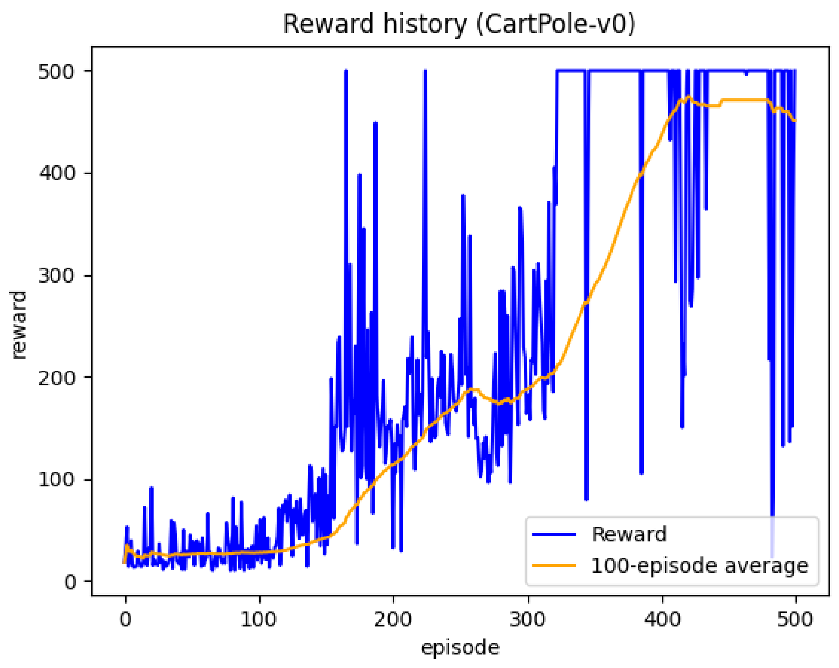

Figure 4: Training for the second reward function

In what the third reward function is concerned a lot of methods have been tried, given that, even if the drawn functions seemed to be logically efficient, the cart-pole was not reacting well deviating from the desired behaviour.
The first attempt was based on the idea of building the reward function giving different weights to the contributions of the reward for the high velocity, reward for the position that must not pass by the borders of the screen and reward for the balance of the pole. Each of them, in turn, starts from value 1 and decreases depending on how much penalized the agent has to be in the specific case. In other words, this reward function is represented by the convex combination of the fore-mentioned rewards and looks like

$$reward+ = 0.8 * reward\_velocity + 0.1 * reward\_pole + 0.1 * reward\_position$$

Despite this well-structured shape, the test on the cart-pole worked poorly with it just moving in random directions or just staying fixed in some position inside the screen, even though at certain point in the training session it seemed to start to behave correctly.
The next attempt is similar to the previous in what regards the convex combination part but differs in the way the "sub-rewards" are computed. In fact, now the previous state is taken into account as well as the current one in order to honour the agent if the pole moves toward the vertical axis, each position in absolute value is greater than the previous (meaning that the cart moves toward the edges) and the velocity increases. Sadly, even in this case the agent does not pass the test properly, sometimes it drifts in one direction, sometimes it goes to the border and than comes back, but every movement is done very slowly.
Eventually, the third and last attempt seemed to report nice results. Here the new approach is to consider two boolean variables $barrier$ and $change$ which are controlled in the train/test function at each iteration of the loop inside each episode and set in a way to keep track of when the cart

has to start decreasing velocity (to avoid crushing at the borders) and of the direction change the cart-pole has to do whenever it reaches the edge of the screen, in order to not drift out. A third sentinel variable ok is used to introduce the high velocity condition in the center only after a few episodes, otherwise the cart-pole struggles to balance the pole at the beginning of the training. The carring idea is that the method through which the rewards are assigned has to be different between the two movements of going toward a border and then inverting the route and turning back, in other words, if the cart is moving left the reward function must verify that the velocity is increasing and is negative until it bypasses the barrier and starts decelerating, at this point, after it reaches the left-most feasible position, the $change$ variable is flagged and the function begins to control that the velocity increases toward positive values. The same process happens in the right side but with signs switched. Proceeding like that the agent during the test starts moving to take a run-up and start bouncing between the borders remaining alive with increasing velocity. Moreover, the possible number of steps per episode has been augmented to 1500 in order to allow the cart-pole to experience the complete motion and the number of episodes has been set to 500. The average test reward was in this case 497 while the episode length 1205, the maximum velocity reached is in modulus 1.6. The image below shows the result of 15 parallel trainings using the script ”multiple_cartpoles_rl.py”.

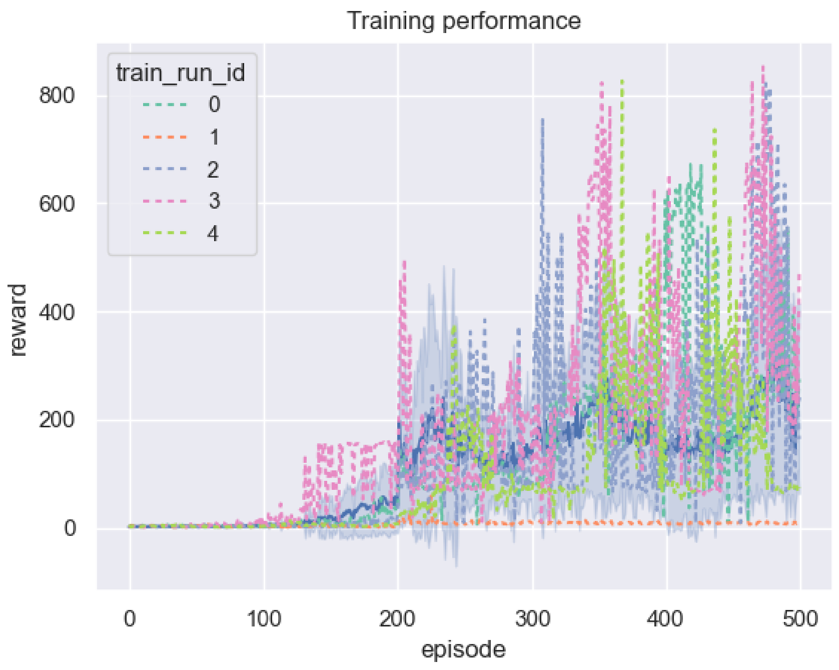

Figure 5: 15 training runs on the cart-pole under the third reward function

3 Conclusion

To conclude, as the results on the simulations highlight, LQR and Reinforcement Learning work very differently on the system taken in consideration. On one hand, LQR is an optimal control approach and through a few calculations it is possible to draw out the value to give to the controller so that the regulated system is asymptotically stable in a desired configuration. On the other side, Reinforcement Learning is a heuristic and it is not certain it will work perfectly, since it is based on stochasticity and the way the reward function is designed deeply changes the outcome of the experiment. However, if one can build the latter carefully, the system can be driven to assume behaviour difficult to achieve with other control approaches and, also, it is even possible to regulate a hypothetic agent without knowing its dynamic model.