Robot Learning Homework 2

Carena Simone, s309521
s309521@studenti.polito.it

1 Introduction

The purpose of this experiment is to develop a controller to stabilize a *cartpole* system. First, a classical control approach is used, designing an LQR controller for the linearized system. Then a reinforcement learning strategy is adopted to control the system and to impose it a a desired behavior through the shaping of a reward function. Advantages and disadvantages of both approaches are presented throughout this report.

2 System Description

The system under study is composed of a cart with a rigid pole on top of it. The goal of the control problem is to stabilize the pole so it does not fall over.

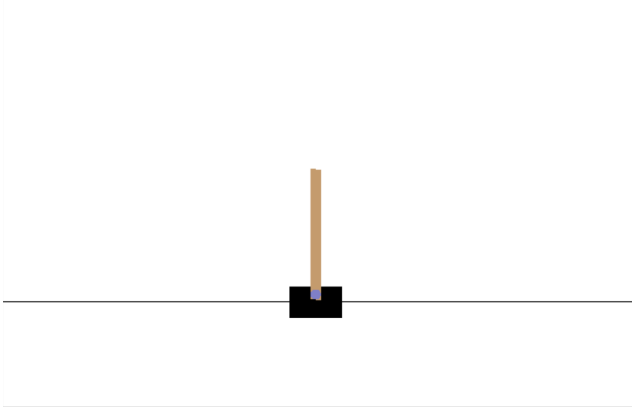


Figure 1: Carpole System

The dynamics of the cartpole, described by the state variable $\mathbf{x} = \begin{bmatrix} x & \dot{x} & \theta & \dot{\theta} \end{bmatrix}^{\mathrm{T}}$, are nonlinear. A linearization is performed around the equilibrium point and the resulting linear system is studied

$$\dot{\mathbf{x}} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\frac{g}{l\left(\frac{m_p}{m_p+m_k}-\frac{4}{3}\right)} & 0 \end{pmatrix} \mathbf{x} + \begin{pmatrix} 0 \\ \frac{1}{m_k+m_p} \\ 0 \\ \frac{1}{l(m_k+m_p)\left(\frac{m_p}{m_p+m_k}-\frac{4}{3}\right)} \end{pmatrix} \mathbf{u}$$

3 LQR Approach

The *Linear Quadratic Regulator* (LQR) is an optimal linear controller, where the optimality of the control action is defined by the matrix $\mathbf{Q}$ and $\mathbf{R}$, which describe, respectively, what should be

minimized between the state convergence and the command effort, i.e. higher entries of $\mathbf{Q}$ yield a faster convergence, while greater entries of $\mathbf{R}$ result in a lower effort of the actuators.

At first the controller is designed with $\mathbf{Q} = \mathbf{I}_4$ and $R = 1$. This choice of parameters results in the following system's evolution

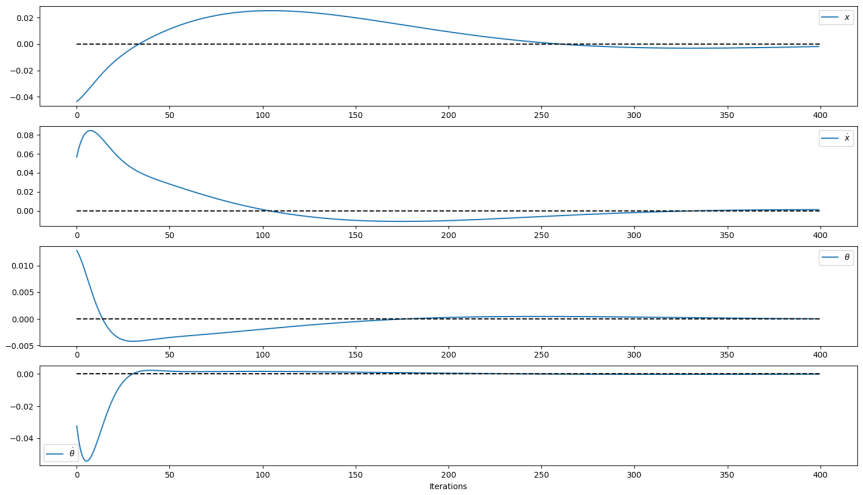


Figure 2: Evolution of the LQR-controlled cartpole

All system states converge to the equilibrium point, with ± 0.05 tolerance, within 27 iterations.

Secondly, to show the effect of $\mathbf{R}$ on the system's actuation, different values of such matrix are tried, in particular with $\mathbf{R} = 0.01$, 0.1, 10 and 100

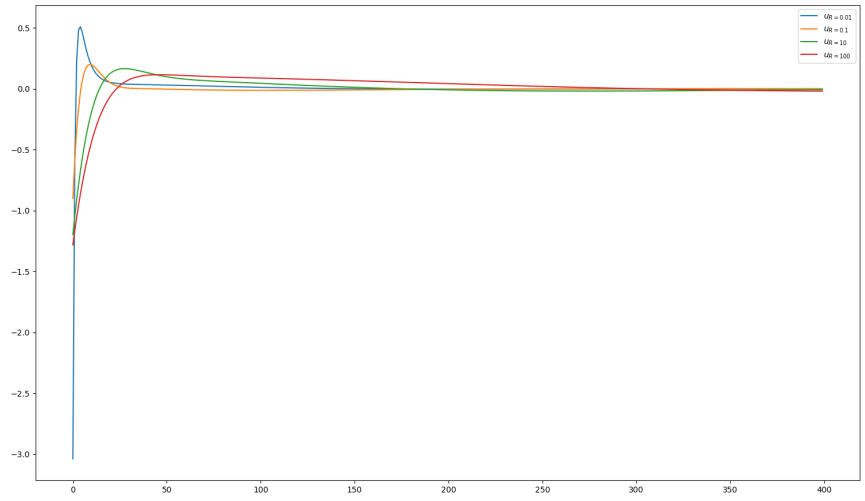


Figure 3: Command Effort with Different Values of $\mathbf{R}$

As it is possible to see from (3), an higher value of R results in a lower peak of command activity, though it also leads to an higher convergence time.

4 Reinforcement Learning Approach

4.1 Training and Testing

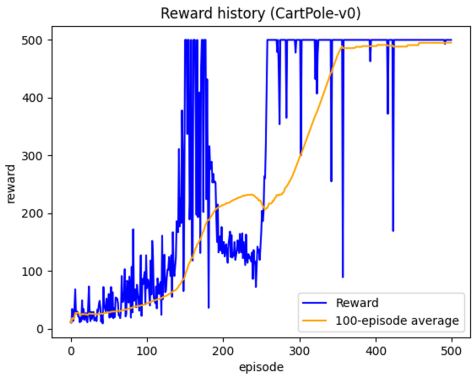
The reinforcement learning environment runs several episodes in order to make the agent learn to balance the cartpole system. Each episode concludes when either of these event occurs:

- The cartpole angle is greater or equal than $\pm 12°$
- The cartpole position is greater or equal than 2.4, i.e. the cartpole leaves the screen
- The episode length is greater than a fixed value define by the env attribute `_max_episode_steps`

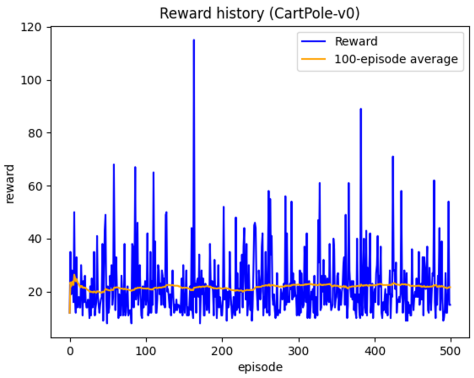
In the basic configuration, a reward of `+1` is given to the agent every step it manages to keep the episode running.

4.1.1 Random Policy

A model can be trained using a random policy. This means that we do not consider the actions predicted by the agents, but instead the action performed is sample randomly from the action space. The result of training, along with a standard training result, is presented below



Standard Policy Training Results



Random Policy Training Results

As it is possible to see, using a random policy, in contrast to a more structured one, the agent does not learn and thus its reward does not increase over time. This happens because the action to take is randomly sampled from the action space, instead of being the result of a trail and error procedure that should shape the desired behavior. In particular the model trained without the random policy, during the testing phase, yield a mean reward of 500.0, while the agent trained with the random policy performs poorly.

4.1.2 Training and Testing with Different Timesteps

We can try to train the model over `200` timesteps and test it with `500` to see if it can still manage to balance the pole. The results show that, if the agent can learn to balance the system within 200 timesteps, it can usually keep it balanced also for 500 timesteps. This mean that if the agent can learn a method for balancing the pole for 200 timesteps, it is probably generalizable to 500 iterations.

4.2 Repetability

If we train several models with the same reward function we can notice a large variance between the various trainings as it is possible to see from the plot below

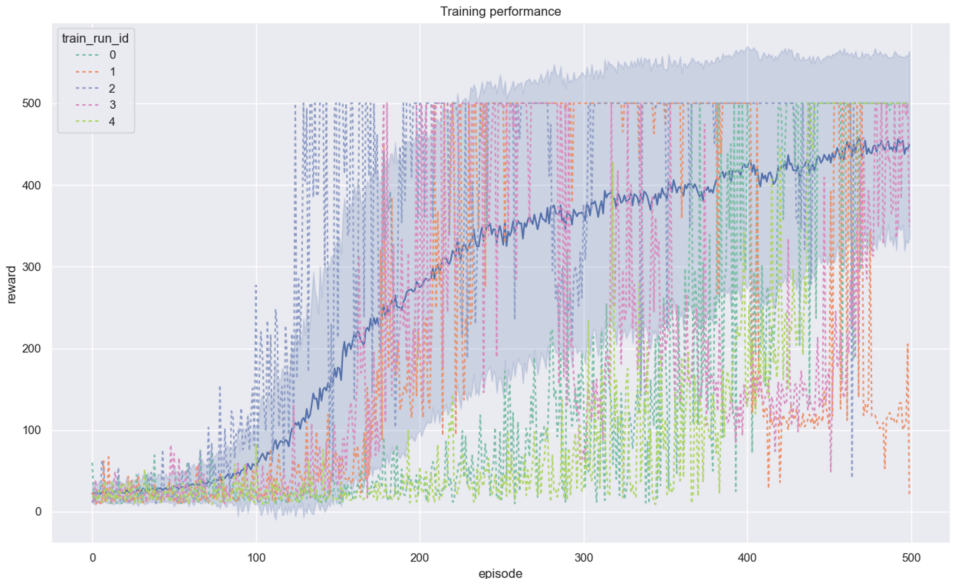


Figure 4: Multiple Cartpole Training Results

This large variability among the rewards of the various training examples shows how different models can learn in different ways the policy to execute. In particular this may lead some models to perform better than others, and some even not to learn. This is due to the fact that some agents don't predict well enough the reward they could get and act greedily or it take them longer to determine a good strategy to maximize the reward.

4.3 Reward Functions

The basic reward function of giving `+1` every time the agent manages to avoid the episode termination is not well suited for trying to impose a particular behavior to the agent. Thus, new reward functions are shape in order to make the cartpole follow a desired behavior.

4.3.1 Cartpole Balancing Specifying the Position ($x = x_0$)

To keep the cartpole fixed in a desired position $x = x_0$ while preventing the pole from falling, we could design a reward that reaches its maximum when the position $x \approx x_0$ and the pole angle $\theta \approx 0$. To model such behavior the reward function used is

$$r = e^{-|x-x_0|-|\theta|}$$

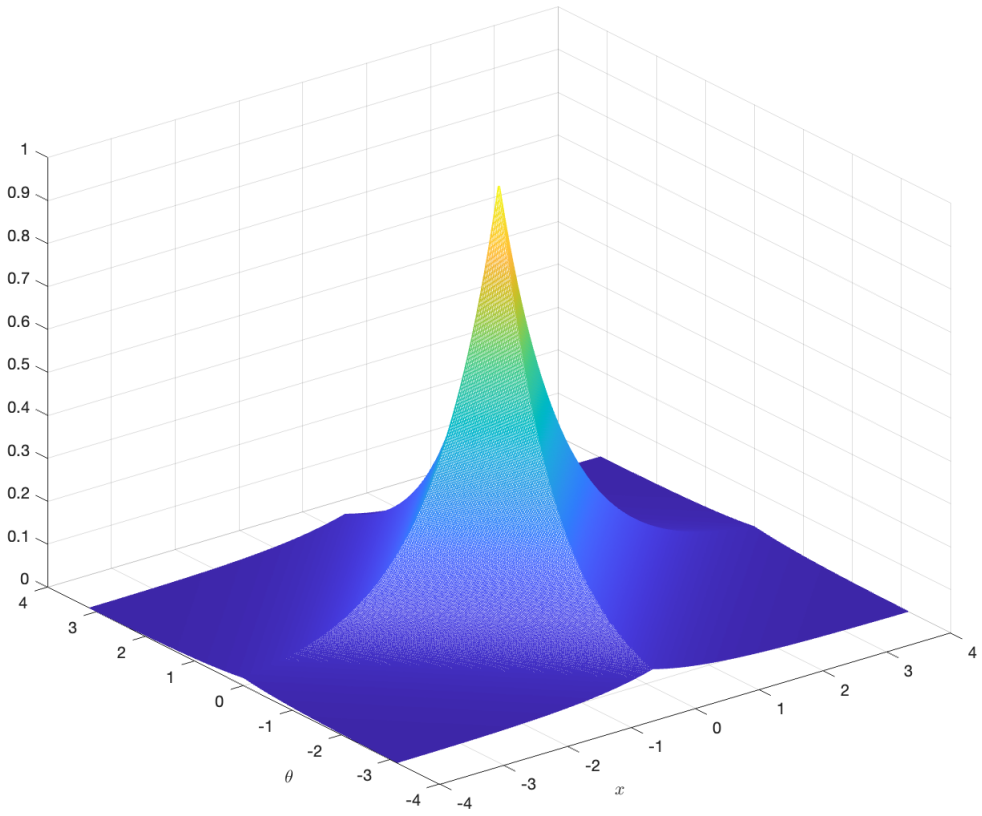


Figure 5: Reward Function to Position the Cartpole to $x = x_0$

4.3.2 Making the Cartpole Move from One End of the Screen to the Other

To make the cartpole move from one end of the screen to the other as fast as possible two reward functions have been shaped. These two reward achieved acceptable results once but other attempts to recreate the desired behavior using those failed multiple times. Both function try to maximize the speeed of the cartpole by giving more reward the faster it goes, but using two different approaches.

The first reward is

$$\tanh\left(|\dot{x}|^4\right) + 0.5$$

To avoid having the agent indefinitely increasing its speed, an hyperbolic tangent function is used to limit the speed reward from 0 to 1. The central area of the curve, around $\dot{x} = 0$ is almost flat, this is done to penalize the cartpole from staying still. A constant of `+0.5` is added to reward the agent for keeping the episode from ending prematurely.

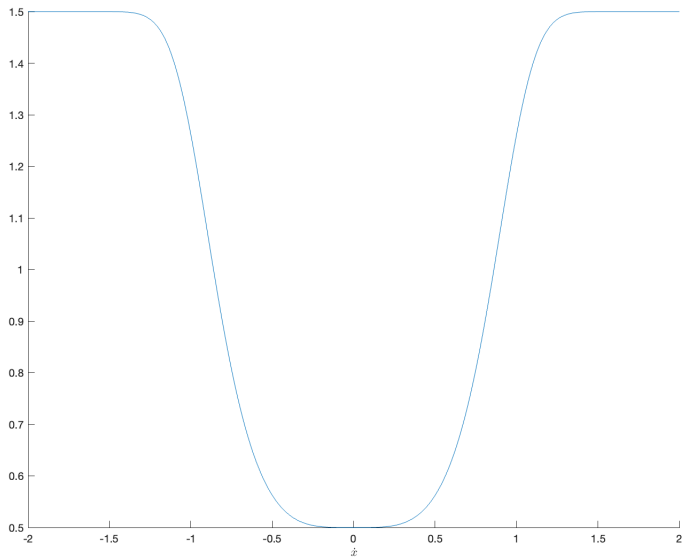
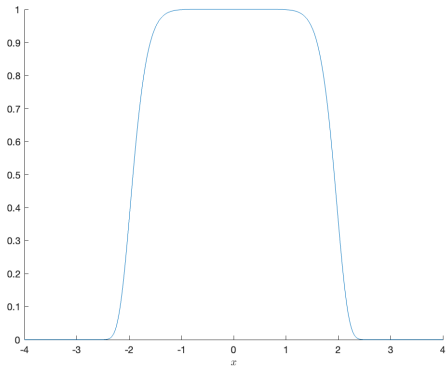


Figure 6: Reward Function 1

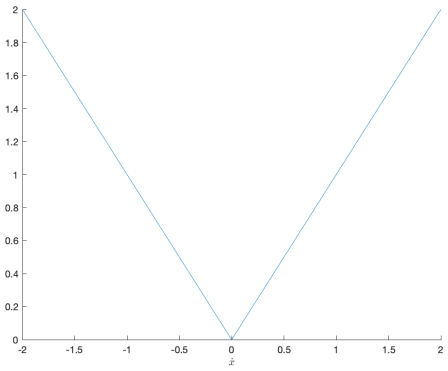
The second reward function is

$$0.5 + |\dot{x}|\, e^{-0.001x^{10}}$$

This function is similar to the former, but it allows the speed reward to grow indefinitely. To keep the cartpole from building up too much speed and leaving the screen, a penalization is added if $x < -2.2$ or $x > 2.2$. A constant of `+0.5` is also added to reward the agent for keeping the episode from ending prematurely.



Penalization Part of the Reward



Reward for Going Fast

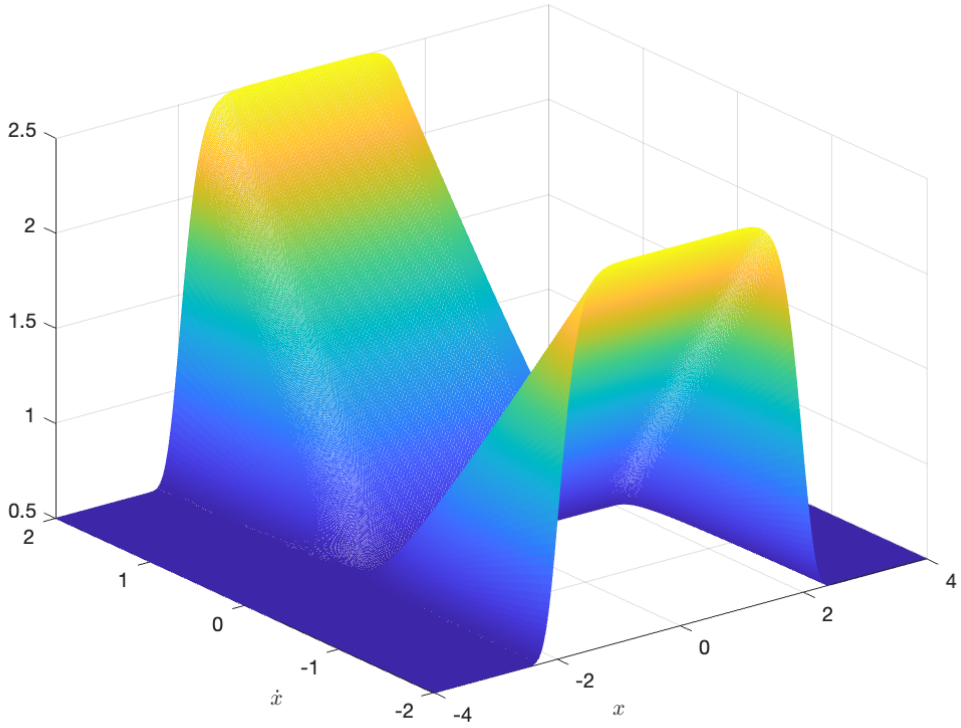


Figure 7: Complete Resulting Reward Function 2

As mentioned before, the training using these reward function was successful only once, and further attempts to recreate the desired behavior failed. This can be seen from the training reward of multiple agents

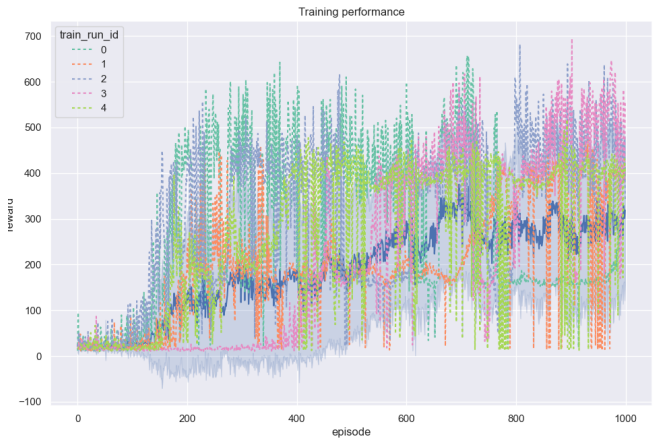


Figure 8: Training Reward of 10 Agents Using the Second Reward Function

The behavior of the cartpole observed for unsuccessful training where that the cart either found an higher reward by standing still, it could not balance the pole or it greatly increased its speed and left the screen.

5 Conclusions

By taking into account the result obtained from the various test performed, and the realization complexity resulting from both the LQR and the Reinforcement Learning approaches, several considerations can be made.

Using an LQR to control a system is easier to develop, if the system is no excessively complex. Though the LQR is an optimal linear controller it proved to be effective even on the linearized cartpole system. The easier realization also comes with a better understanding of the behavior of the controlled system: the effect of the $\mathbf{Q}$ and $\mathbf{R}$ matrices is more easily justifiable. The main drawback of a pure control approach comes from the limitations it comes with. In particular imposing a specific behavior to the system is much more complicated, if not impossible, in some cases.

The Reinforcement Learning approach comes with several advantages, but also some drawbacks. The major advantage is that the system can learn to perform particular actions that would be tedious to implement with a classical control approach. Given that, a great disadvantage of the approach is the great variability that comes from the training of the agents: agents may learn different policies, in different amount of time, and some may not even learn to perform the desired actions. The learning variability is accentuated by the difficulty of finding a proper reward function to control the system as desired. The more complex the task to undertake, the more articulated the reward, the higher the probability of having to perform several training to find an agent that learns to impose the behavior we are interested in. Furthermore, if the agent does not behave as expected it is more complicated to find the cause of the behavior: it could be a badly-shaped reward function, as well as the fact that not enough trainings have been performed and no agent learned properly. On the contrary, the LQR effects are more easily perceptible and easier to change (if possible), by tuning the values of $\mathbf{Q}$ and $\mathbf{R}$.

In conclusion both approaches bear advantages and disadvantages, but from the experiments performed, for a simple system, a classical control approach is preferable as it is simpler to tune.