Autonomus Reinforcement Learning

Authors: **Francisco Giancarelli** (UTN - FRSF), **Juan C. Barsce** (UTN - FRVM - INGAR - CONICET), **Ernesto Martinez** (INGAR - CONICET)

Starting point
1

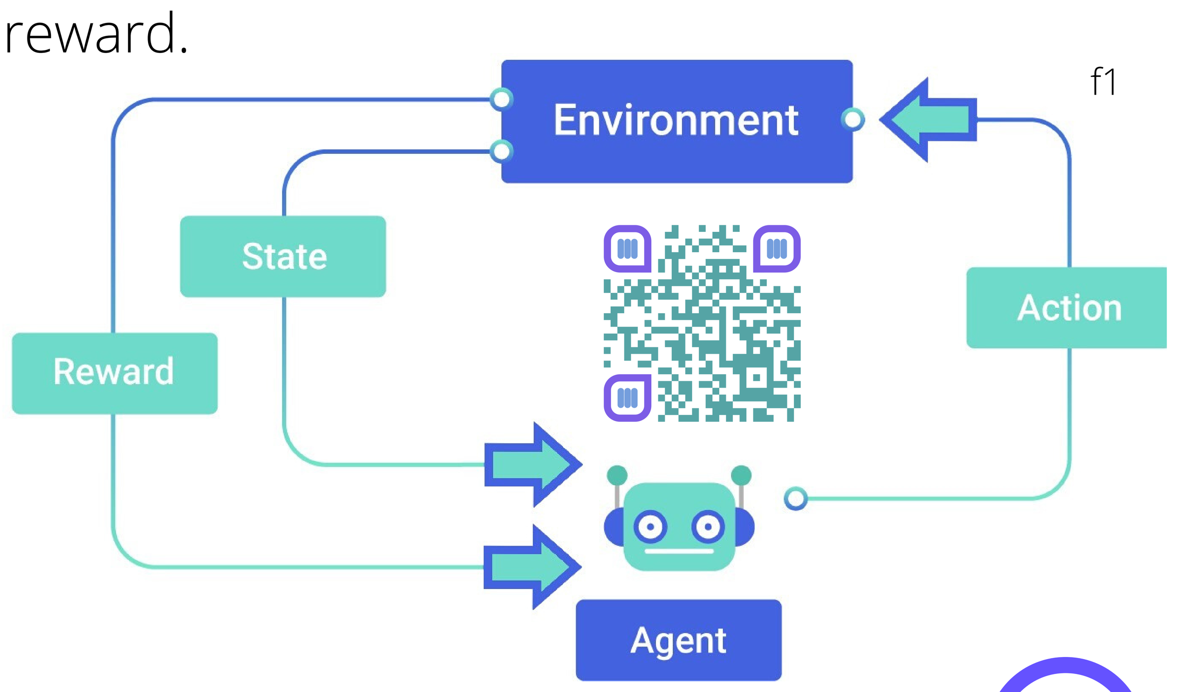
The use of Reinforcement Learning (RL) algorithms to solve practical problems, rapidly becomes a complex task, being the main problem to be addressed the proper selection of hyper-parameters.

In order to make algorithms more autonomous, a tight integration of reinforcement learning with Bayesian Optimization is proposed for the automatic selection of hyperparameters.

What is reinforcement learning?
2

Reinforcement learning is one of the three main paradigms of algorithms we can use when solving a machine learning problem (along side Supervised and Unsupervised Learning).

RL is based on the idea of letting an agent interact with an environment, and through a series of actions, and observing state transitions and corresponding reward, learn a policy to maximize cumulative or average reward.



How does it work?
3

The agent takes an action, and the environment makes a transition to a new state and provides a reward or hint about the goodness or the action taken. The sequence action-state-reward (f1) is what the agent uses to learn its policy and maximize the cumulative (or average) reward.

The learning rule the agent use to modify its policy depend or each specific algorithm.

For example in the **Q-learning algorithm:**

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha * (r_t + \gamma * maxQ(s_{t+1},a_{t+1}) - Q(s_t,a_t))$$

We can already see the appearance of two hyper-parameters:

Alpha: The learning rate of our agent.

Gamma: The discount factor (How much should the agent care for future rewards).

The importance of hyper-parameters?
4

In RL the value of hyper-parameters define the informative content of data generation. Hyper-parameters can be related to real value parameters like the learning rate (alpha) or the discount factor (gamma), or more structural decisions like the algorithm to use and the space discretization.

We will refer to the former as **algorithm hyper-parameters**, and the latter as structural **hyper-parameters.**

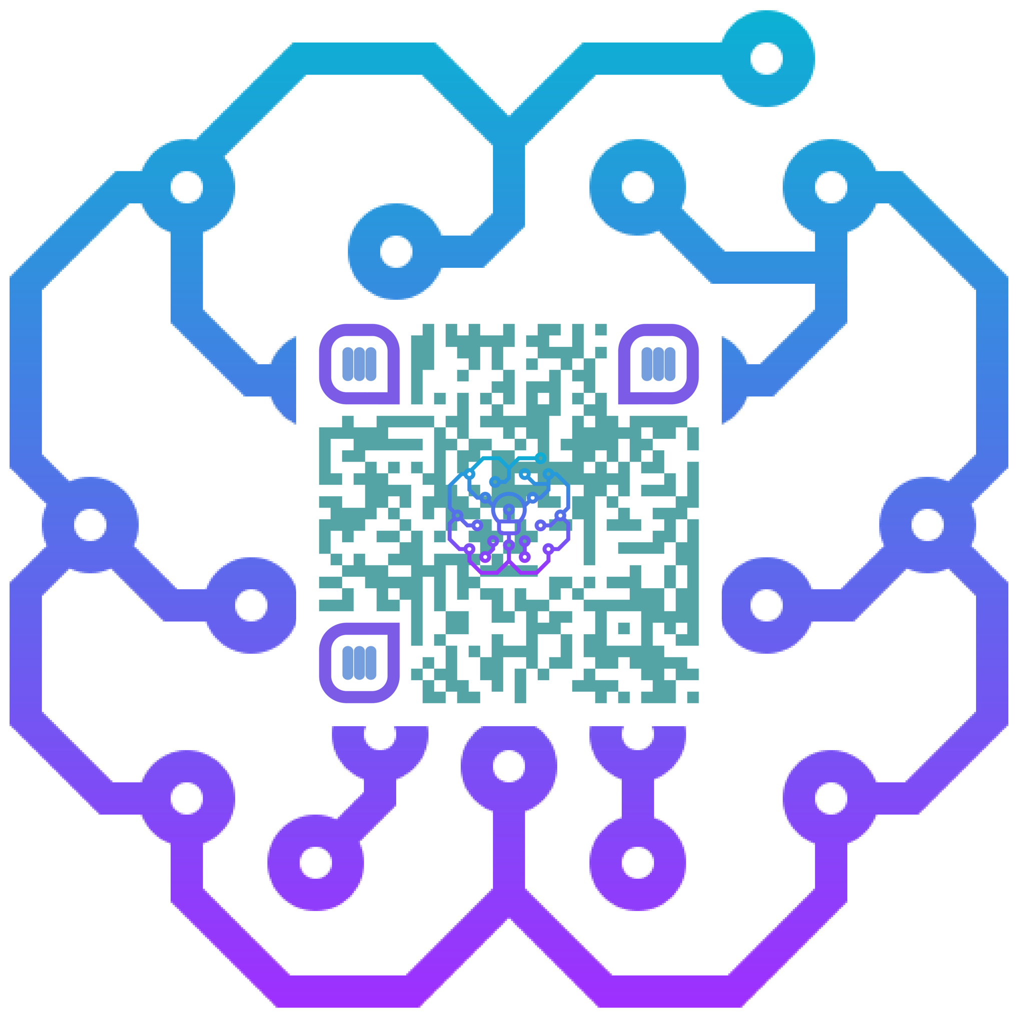
Bayesian Optimization
5

Bayesian Optimization is actually as strategy, that in the case of our study, consists on treating the achieved learning function (given a set of hyper-parameters) as random, and place a prior over it. Then we evaluate it with random hyper-parameters and update our function model with the learning achieved. Finally we use the posterior distribution to form an **acquisition function** that help us determine our next query point based on a given criteria, and the cycle starts again.

The algorithm:

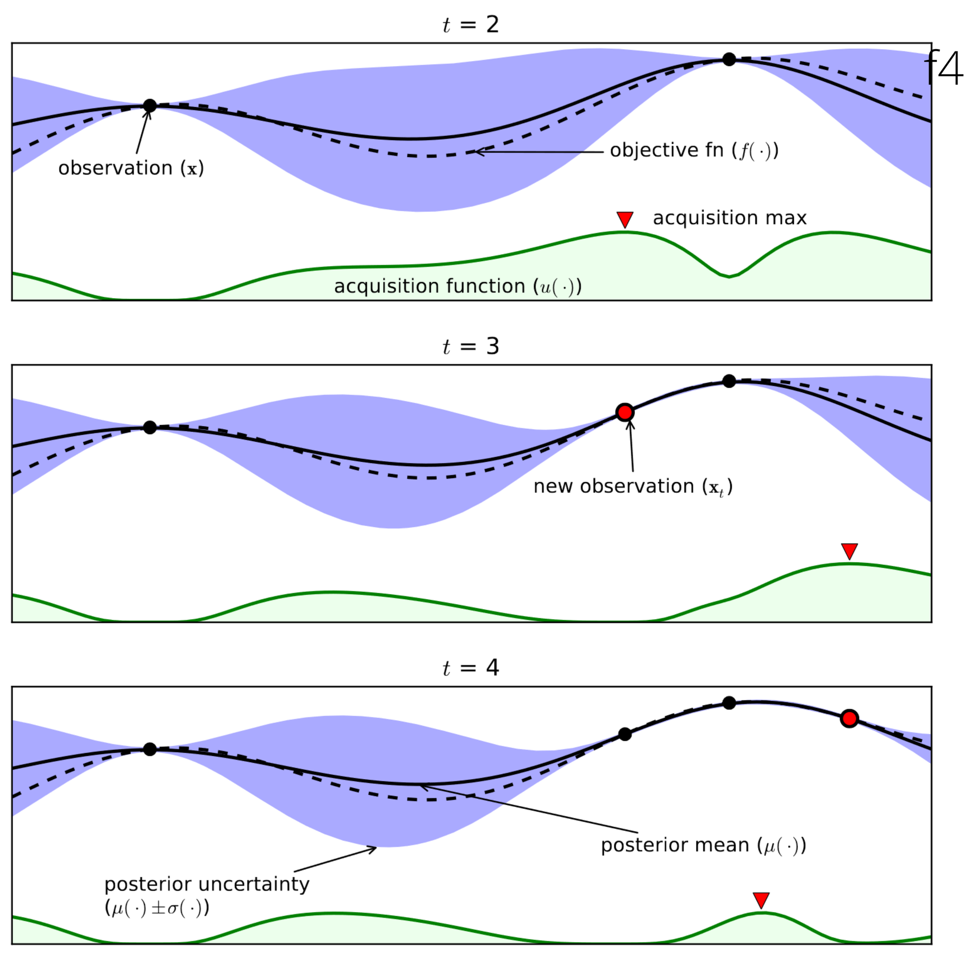
for n loops do
-> select new x_{n+1} by optimization of α which is an acquisition function x_{n+1} = max α(x; D_n)
-> get new observation y_{n+1} from objective function
-> augment data $D_{n+1} = \{D_n, (x_{n+1}, y_{n+1})\}$
-> update model
end for

At the end of the column there's a more graphical example of the algorithm (f2).



How do we update our model?
6

To update our model we use something called **Gaussian Processes**. These can be described as a set of random variables indexed by time or space. Why do we use this for our model? because it allows us to know not only the average, but the level of uncertainty we have in every point of the domain of our function.



Our proposal
7

Given the complexity of the calculations we have to make in order to optimize many hyper-parameters, we propose to **divide the optimization into two stages**.

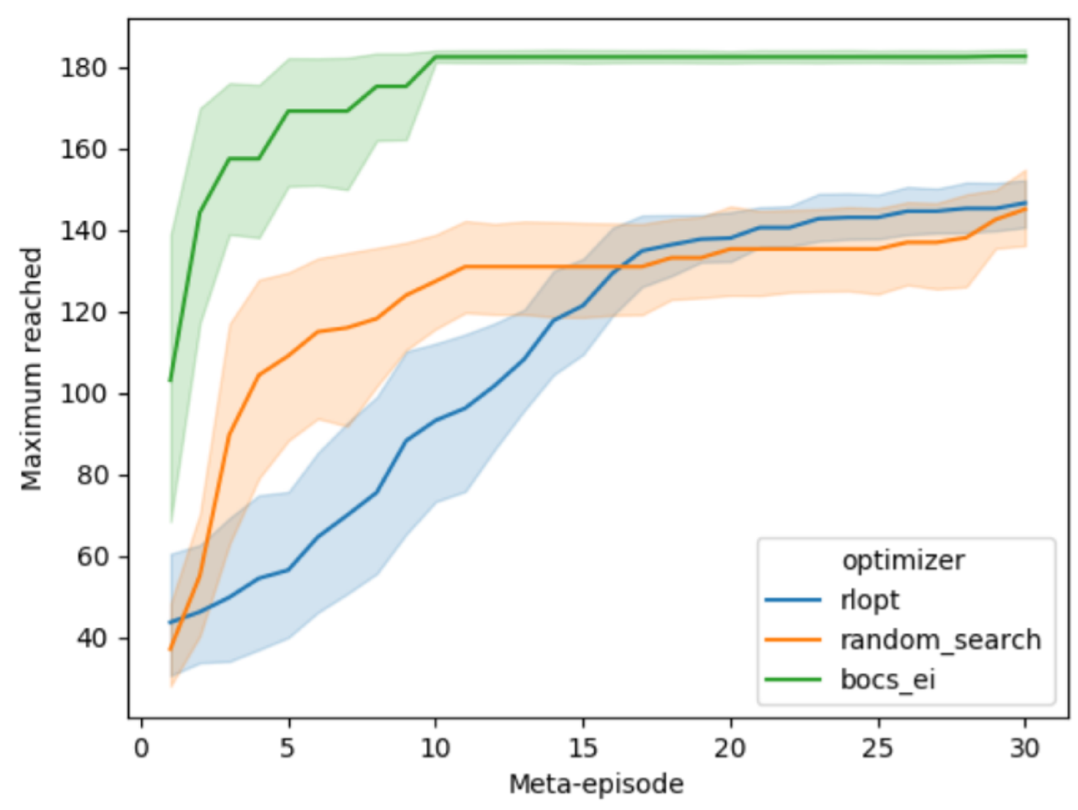
First we optimize the structural hyper-parameters, fixing the algorithm hyper-parameters to a random selection. And then we use the resulting performance to optimize the algorithm hyper-parameters with the first ones fixed. Then is just a matter of repeating this two tier procedure.

Input : Set of prior algorithm hyper-parameters Θ_p
for *structural evaluation* $= 1$ *to* N **do**
  Obtain Θ_n that optimizes $\alpha_{BOCS}(.) \rightarrow \mathbb{R}$ over structural hyper-parameters, with prior algorithm hyper-parameters preset as Θ_p
  Query the objective function f at the point $\Theta_n = \Theta_s \cup \Theta_p$
  $D_1 \leftarrow D_1 \cup \{\Theta_n, f(\Theta_n)\}$,
  Initialize $D_2 \leftarrow \{\Theta^+, f(\Theta^+)\}$ with the best $\Theta^+ = \Theta_s^+ \cup \Theta_p$ point and corresponding maximum of f found
  for *hyper-parameters evaluation* $= 1$ *to* M **do**
    Obtain $\Theta_m = \Theta_s^+ \cup \Theta_a$ that optimizes $\alpha_{EI}(.) \rightarrow \mathbb{R}$, with structural hyper-parameters preset as Θ_s^+
    Query the objective function f at the point $\Theta_m = \Theta_s^+ \cup \Theta_a$
    Add the result $f(\Theta)$ to D_2
  end
  Set prior algorithm hyper-parameters $\Theta_p \leftarrow \Theta_a^+$, being Θ_a^+ the best set of algorithm hyper-parameters
end
return $(\arg\max_\Theta f, \max f)$

Our test
8

We will compare our algorithm with a random search and a decoupled Bayesian Optimization.

Further information about the experiment is detailed in the QR in the center.



Future?
9

Bayesian Optimization gives us a faster learning process, while being more certain and consistent than the thumb's rules normally used. Autonomous RL allows the user to focus on defining the correct reward function for the task at hand.

Acknowledgements
10

The work is part of research activities funded by PID UTN UTI4375TC project titled "Abstracciones, Modelos y Algoritmos para Optimización Bayesiana y Control Inteligente de Sistemas integrando Aprendizaje por Refuerzos con Procesos Gaussianos." F. Giancarelli has a student allowance grant associated with this project and J. C. Barsce has posdoctoral grant funded by UTN.