ESPIDAM epi AI project

Authors: Bram Silue, Sebastiaan Weytjens, prof. Pieter Libin, prof. Niel Hens

Acknowledgments: Veronica Biancacci, Alexandra Cimpean

After these two days, we now have some baggage on epi modeling and reinforcement learning. Time to combine this knowledge!

Background

We will consider the problem of school closures, more specifically, given a particular school closure budget (in weeks), how can we allocate these school closure weeks in an optimal way. We will look at a simple setting, with many simplifying assumptions, yet by the end of this exercise, we will have the background on how to target epi problems using reinforcement learning.

Model and Markov decision process

We will use an age-structured SIR ODE model, where we will implement school closures by adjusting the contact matrix (as studied during the practical of the first day of ESPIDAM). We will consider the optimisation with respect to the number of new infections.

To formalize this setting, we consider:

- a state space that directly corresponds to the state space of the epi model
- a transition function that corresponds to the dynamics of the ODE model
- actions that correspond to the closure of schools on a weekly basis (i.e., decide every week of the epidemic whether to keep the schools open or close them)
- a reward function that considers the difference in susceptibles at every time step (-[(Sa+Sc)-(S'a+S'c)]); as detailed in [1]).

We provide you with a function *run_sir_model* in *sir.py* that allows you to simulate the epi model until a given timestep. If we optimize school closures without any constraints, the reinforcement learning agent has no incentive to limit the number of weeks the schools are closed, while there is clearly an important burden involved in such choices. Ideally, we would tackle this from a multi-criteria perspective (more on that later today), but, for the sake of simplicity, we will consider a *budget* of school closures. This means that, if the agent chooses to close schools when the budget is spent, this action will have to be ignored.

Implement the gym(nasium) environment

So, to take the first step, let's code a gym environment that implements the above modalities. Recall that on the second day of ESPIDAM, we studied a simple gym environment [2], that can act as inspiration for your implementation.

We will give you some tips, to give you a head start:

- use a Discrete action space (gym.spaces.Discrete)
- use a Box observation space (spaces.Box)
- no need to implement a render function

Test your gym environment!

If there are bugs in your gym environment, the agent will learn bogus things. As such it is *very* important to properly test the environment, by interfacing with it via the step function, and verify the output of some example policies (i.e., sequences of school closure decisions).

Learning an optimal policy with PPO

We will use Stable Baseline's Proxima Policy Optimization (PPO) implementation to learn an optimal policy for different budgets. Implement a script that instantiates your gym environment and initializes a PPO agent to learn a policy. Develop this script such that you can parameterize it for different school closure budgets and different PPO seeds. You can use inspiration from the Jupyter notebook where PPO was used with the LunarLander environment.

Some important tips:

- For PPO, we only adjust the learning rate hyperparameter, which is set to *2e-3*. This was hand-tuned (we will get back to that later today).
- To learn efficiently in this environment, it is advised to normalize the observation space (why do you think this is the case?) and reward function. This can be done by wrapping your environment (i.e., NormalizeObservation and NormalizeReward in the gymnasium.wrappers module).
- Stable baselines allow you to write tensorboard logs [3], which make it easy to inspect your learning curves. Look for the "tb_log_name" parameter in [3] to add the budget to your tensorboard run names.

With this script, run some experiments (250.000 steps should be enough to learn something), with budget={2,4,6} and 2 different seeds. Visualize these results through tensorboard and interpret these results. We also provided functions to visualize the learned policy in *plot_policy.py*.

References:

[1] https://arxiv.org/pdf/2003.13676
[2] https://www.gymlibrary.dev/content/environment_creation/
[3] https://stable-baselines.readthedocs.io/en/master/guide/tensorboard.html