ESPIDAM project: Multi-armed bandits for IBMs

Authors: prof. Pieter Libin, prof. Niel Hens

Acknowledgments: Bram Silue

We now have some baggage on individual-based models and multi-armed bandits. Time to combine this knowledge!

Background

Torneri et al. developed an **individual-based stochastic model of SARS-CoV-2 transmission** tailored to primary school contact structures [1]. This model supports the simulation of three distinct intervention strategies: symptom-based isolation, reactive screening, and repetitive testing. The school model supports two outcome statistics: attack rate and number of school days lost, to support a balanced evaluation of intervention strategies. Ideally, we would tackle this from a multi-criteria perspective (more on that tomorrow), but, for the sake of simplicity, we will focus on the attack rate, which we will optimize using a **multi-armed bandit framework**.

As we discussed during the lecture, **evaluating preventive strategies by simulating each option equally is inefficient**, as it wastes computational resources on suboptimal choices and lacks clear guidance on how many simulations are needed per strategy [2]. Given the high cost of running individual-based models, using a multi-armed bandit framework is thus appealing. In this framework, finding the best prevention strategy corresponds to a best-arm identification problem [2].

However, returning only the **single best prevention strategy** can limit public health decision-making, as it offers no flexibility for political, legal, or logistical considerations. Providing a set of top-performing strategies allows experts to compare alternatives that may be equally effective in reducing infection but differ in practical aspects like logistics or cost. While such trade-offs could be handled with a multi-objective approach (we will come back to this tomorrow), it adds significant complexity to the analysis. This corresponds to the **m-top exploration** setting in multi-armed bandits. [3].

Therefore, in this project, we will approach the **optimization of prevention strategies** in a **school simulator**, with respect to the **attack rate**, using an **m-top exploration approach**.

Bernoulli bandit

We will conduct experiments using the python library that implements an m-top exploration framework[1], including two state-of-the-art algorithms AT-LUCB [3] and Boundary Focused Thompson Sampling (BFTS) [4]. BFTS is a Bayesian algorithm, which means we can incorporate prior knowledge in the sampling strategy.

We will start with a simple Bernoulli bandit, where each arm is defined by a Bernoulli distribution with an arm-specific mean. This will allow us to get acquainted with the BFTS framework and do some initial experiments.

Study the example script *bernoulli-exp.py*. In this script, we construct a Bernoulli bandit with 20 arms. We run an experiment for 2000 arm pulls, for three algorithms (AT-LUCB, BFTS and uniform sampling), considering 5 replicates.

Questions and experimentation:

- How are the ground-truth means of the arms determined, in this Bernoulli environment?
- Which prior was chosen in this experiment?

[1] https://github.com/plibin/bfts

- Run the script and observe the outcomes and give your interpretation. The plots visualize the performance of the bandit compared to the ground truth.
- As you will notice, using five replicates leaves us with quite high variance in the experimental results. However, increasing the number of replicates will induce a significant computational burden, especially when considering computationally intensive simulators, such as individual-based models. While *bernoulli-exp.py* is in python*,* the framework is fully prepared to support running experiments on a high-performance cluster. Propose how this script can be ported to your HPC cluster such that the replicates can be run in parallel.

Towards the school bandit

We will be looking at the intervention strategies as studied in [1], the list of strategies (or arms) that is considered can be found in *arms.py*.

Since we will continue using BFTS, we will need to **choose a prior and likelihood**. As we are modelling the mean of the attack rate, we will assume a central tendency with a limited, yet unknown, variance[2], i.e., we assume a Gaussian over the means of the reward distribution. For this reward distribution, when assuming a non-informative Jeffreys prior, this results in a T-distributed posterior [5].

Given the limited time we have during this session, it is **not possible to run the experiments with the real school simulator**. To allow experimentation, we simulated each arm 1000 times (i.e., a brute force analysis), which results in an **attack rate distribution** for each of the considered arms. We provide you with these empirical arm distributions of this brute force analysis in the *brute-force-ar.csv* CSV file.

Questions and experimentation:

- Visualize the attack rate distribution, using *plot-brute-force-ar.py,* and discuss.
- The attack rate distribution is not a reward distribution yet! Study the script *ar-to-reward.py* and use it to convert it to a proper reward distribution.

Building the school bandit

Now, based on the Bernoulli bandit example (*bernoulli-exp.py*), construct a *school-exp.py* script that performs 5 replicates for AT-LUCB, BFTS and uniform sampling, for the school attack rate distributions.

Some pointers:

- You can use the environment in bfts/environments/csv_dist.py, this environment will accept the reward distribution CSV (brute-force-reward.csv) and construct a bandit with the arms and their distribution as specified in the CSV. When an arm is pulled, it will uniformly random return one of the 1000 rewards established in the brute force experiment.
- You can find the available posteriors in bfts/algorithms/posteriors.
- You should be able to get some insights with 2000 arm pulls.

Running a school bandit analysis

Experiments as in the previous section are valuable to get more insights in bandit algorithms and simulators. However, the real use case of m-top exploration is to identify the m-top

[2] Here we make the statistical convenient assumption of a Gaussian reward distribution. However, if more dispersion is expected, a negative binomial could be warranted.

strategies, using only a limited of arm pulls. In such settings, there is **no ground truth available**. As such, it is tricky to decide when enough samples have been achieved, to come to a decision. Bayesian bandit algorithms have an advantage here, as we can inspect the posteriors to assess the uncertainty of the m-top exploration process.

To investigate this, draft a script that executes one BFTS run on the school model and inspect the posteriors at different time points.
Some pointers:

- Start from the run_bfts.py script, to run one replicate
- Visualize the posteriors, at different time steps:

```
def pdf(x, posterior, rewards):
  n = len(rewards)
  freedom = posterior.freedom(n)
  sigma = np.sqrt(np.var(rewards)/freedom)
  mu = np.mean(rewards)
  return sp.t.pdf((x - mu)/sigma, freedom) / sigma
```

References
[1] Andrea Torneri, Lander Willem, Vittoria Colizza, Cécile Kremer, Christelle Meuris, Gilles Darcis, Niel Hens*, Pieter JK Libin*. Controlling SARS-CoV-2 in schools using repetitive testing strategies, eLife, 2022.
[2] Libin, P. J., Verstraeten, T., Roijers, D. M., Grujic, J., Theys, K., Lemey, P., & Nowé, A. Bayesian best-arm identification for selecting influenza mitigation strategies, ECML, 2018.
[3] Jun, Kwang-Sung & Robert Nowak. "Anytime exploration for multi-armed bandits using confidence information." *International Conference on Machine Learning*, 2016.
[4] Libin, P., Verstraeten, T., Roijers, D. M., Wang, W., Theys, K., & Nowe, A. Bayesian anytime m-top exploration. International Conference on Tools with Artificial Intelligence (ICTAI). IEEE, 2019.
[5] Honda, Junya, and Akimichi Takemura. Optimality of Thompson sampling for Gaussian bandits depends on priors. Artificial Intelligence and Statistics. 2014.