Instructions of Running the Codes for Linear-EXP4/D2-EXP4 Experiments

Jianyu Xu, and Yu-Xiang Wang

February 2022

This is a document of instructions for running our Linear-EXP4 and D2-EXP4 codes for agnostic contextual pricing. We separate the implementation into three procedures: data generating, pricing, and result plotting. We first generate both random x_t's and adversarial ones. Secondly, we take either of the two datasets as inputs of both Linear-EXP4 and D2-EXP4. Thirdly, we set up a program that plots the regret curve of Linear-EXP4 for a geometric series of time horizons T_k's. Since D2-EXP4 is very time-consuming and we cannot run it for multiple T_k's, we do not have its plot (but would instead have a tentative cumulative regret curve for a fixed T while running the function program).

The general process of running our codes is as follows:

1. (Optional) Run `x_t_generator.py` to generate stochastic/adversarial $\{x_t\}$ series.
2. For Linear-EXP4: Run `LinearEXP4_02.py` for the experiment of Linear-EXP4 we conduct in our paper. Or run `linear_exp4.py` (i.e. in the console) for a brief demonstration of it, and there will be a tentative cumulative regret curve.
3. If we have run `LinearEXP4_02.py` for the previous step, then we run `plotting.py` here. Remember to match every parameter in `plotting.py` as we adopted in running `LinearEXP4_02.py` previously.
4. (Optional) Run `D2-EXP4.py` in the console for a brief demonstration of the D2-EXP4 algorithm. There will be a tentative cumulative regret curve.

1 Data Generating

The program `x_t_generator.py` can generate 2 different $\{x_t\}_{t=1}^T$ series: a stochastic series of $\{x_t\}$ generated by a `random agent`, and an adversarial series of $\{x_t\}$ generated by a `super strategic agent`. Both of the agents are defined in the `x_t_agent.py` file. The `random agent` generates i.i.d. $x_t \sim \mathcal{N}(\mu, \Sigma)$ with random vector μ and Σ , and normalizes it such that $||x_t|| \leq \text{normbound}_{x_t}$. The `super strategic agent` generates an $x_{k,t}$ for epoch k and time period t, with $x_{k,t} = [0, 0, 0, \ldots, 0, 1, 0, \ldots, 0]^T$ of which the only 1 at position ($k \mod d$).

In our numerical experiments, we let $d = 2$ and generate 20 stochastic series and 20 adversarial ones, with each of them a length of $70,000$. They are stored as `xtrandom.pkl` and `xtadversarial.pkl` separately. It is worth mentioning that we have also designed another `strategic` agent: it outputs the average of all failed x_t's in history and a random x_t that balances exploration and exploitation. However, this works not as well as we expected, because it would "converge" to some fixed x as T large enough. (To the audience: you can also design your own stochastic and/or strategic x_t agents and generate.)

2 Linear-EXP4

We implement Linear-EXP4 in `linear_exp4.py` as a function `linearexp4()`. The experiment conducted in our paper is written as `LinearExp4.py` that imports and uses this function from `linear_exp.py`.

In function `linearexp4()`, we have the following basic parameters for initialization:

- T is the time horizon. (restricted on $T < 70000$, defaulted as 4096)
- d is the dimension of x_t. (defaulted as 2)
- *rounds* is the number of repeating. (restricted on $rounds \leq 20$, defaulted as 5)
- bound_beta and bound_x are the L_2 norm bounds of parameter β and features. (defaulted as 1)

For the valuation model, we let $y_t = J^{-1}(x_t^\top \beta^*) + N_t$, where β^* is randomly chosen and fixed ahead of time and N_t is a zero-mean Gaussian noise with standard deviation a `sigma` (which is an input parameter of the function `linearexp4()`, defaulted as 0.25). γ is the exploration parameter for the EXP-4 learner inside our Linear-EXP4 algorithm, defaulted as 0.11. There are also some auxiliary parameters: mode_indicator for using stochastic/adversarial feature sequences, ifregression for calculating the linear fit of the tentative cumulative regret in a log-log scale, ifplot for ploting the regret curve and the linear fit, ifprint for printing the monitoring amounts, and start_point for the start point of plotting or calculating log-log linear fits. The function `linearexp4()` outputs the cumulative regret of each round with time horizon T, i.e., it outputs a numpy array of size `rounds` with each element a cumulative regret at time T of a specific round.

The program `LinearEXP4.py` is designed for running `linearexp4()` on a geometric sequence of $T = T_0, T_1, \ldots, T_K$ with $T_{i+1} = 2^c \cdot T_i$. Here we choose $T_0 = 512, c = \frac{1}{3}$ and $K = 21$(=epoches-1). We set $d = 2, rounds = 20$,bound_beta=bound_x=1, sigma=0.25, $gamma = 0.11$. After running these numerical experiments, we plot the cumulative regret for each T_k with empirical mean and 0.95-confidence error bar according to the 20-times repeating. In order to show the regret rate, we plot the results on a log-log diagram and draw a linear-fit that indicates the empirical asymptotic regret bound.

3 D2-EXP4

We implement D2-EXP4 in `D2-EXP4.py` as a function `d2exp4()`. The parameters of `d2exp4()` are similar to those in `linearexp4()` except for those related to the valuation model: In `D2-EXP4.py` we implemented, we let $y_t = x_t^\top \theta^* + N_t$ where θ^* is randomly chosen and fixed ahead of time and N_t is a $[0, 1]$-truncated Gaussian noise with $\mu = 0.5$ and $\sigma = \frac{1}{8}$.

We may run `d2exp4()` and get a plot of tentative cumulative regret curve, but this does not indicate the any-time regret rate while applying D2-EXP4 (since we have to know T in advance for discritization), and nor does the linear fit. It is not hard to implement the experiment of D2-EXP4 on a series of T_k's with multiple repeating, but it must be very time-consuming as the policy set is exponentially large in our setting.