Pricing-Advertising

Consider a setting in which an e-commerce website sells a product and can control both the price and the advertising strategy.

Environment

We assume that a round corresponds to one day. The users are characterized as follows.

- Two binary features can be observed by the advertising platform, call them F1 and F2; users can be of three different classes according to these features, call them C1, C2, C3; these three classes differ in terms of:
    - the function that expresses the number of daily clicks as the bid varies, and
    - the function that assigns the cumulative daily cost of the clicks as the bid varies.
- The three classes (C1, C2, and C3) also distinguish in terms of purchase conversion rate. More precisely, they differ in terms of the function expressing how the conversion probability varies as the price varies.

The construction of the environment can be done as follows.

- For every user class, specify a concave curve expressing the average dependence between the number of clicks and the bid; then add Gaussian noise to the average curve that is used whenever a sample is drawn (that is, when the number of daily clicks is drawn given a bid).
- For every user class, specify a concave curve expressing the average cumulative daily click cost for the bid and add a Gaussian noise over the average to draw a sample (that is, when the cumulative daily click cost is drawn given a bid).
- For every user class, consider 5 different possible prices and use a Bernoulli distribution for every price. This specifies whether the user buys or not the item at that specific price. A sample of the Bernoulli must be independently drawn for every user who landed on the e-commerce website.

The time horizon to use in the experiments is 365 rounds long.

Clairvoyant optimization algorithm

The objective function to maximize is defined as the reward. For one class, the reward is defined as the number of daily clicks multiplied by the conversion probability multiplied by the margin minus the cumulative daily costs due to the advertising. With multiple classes of users, the reward is just the sum of the rewards provided by the single classes.

The continuous set of possible bids can be approximated by a finite set of bids. In particular, the seller can choose among 100 possible bids.

Given a fixed structure of contexts according to which the users are split, the optimization algorithm we suggest using is:

- for every single class find the best price, independently from the other classes;
- then optimize the bid for each class independently from the other classes.

Such an algorithm requires an exhaustive search over the prices for every class and, subsequently, an exhaustive search over the bids for every class. Thus, the algorithm runs in linear time in the number of prices, bids, and contexts.

Step 0: Motivations and environment design

Imagine and motivate a realistic application fitting with the scenario above. Describe all the parameters needed to build the simulator.

Step 1: Learning for pricing

Consider the case in which all the users belong to class C1. Assume that the curves related to the advertising part of the problem are known, while the curve related to the pricing problem is not. Apply the UCB1 and TS algorithms, reporting the plots of the average (over a sufficiently large number of runs) value and standard deviation of the cumulative regret, cumulative reward, instantaneous regret, and instantaneous reward.

Step 2: Learning for advertising

Consider the case in which all the users belong to class C1. Assume that the curve related to the pricing problem is known while the curves related to the advertising problems are not. Apply the GP-UCB and GP-TS algorithms when using GPs to model the two advertising curves, reporting the plots of the average (over a sufficiently large number of runs) value and standard deviation of the cumulative regret, cumulative reward, instantaneous regret, and instantaneous reward.

Step 3: Learning for joint pricing and advertising

Consider the case in which all the users belong to class C1, and no information about the advertising and pricing curves is known beforehand. Apply the GP-UCB and GP-TS algorithms when using GPs to model the two advertising curves, reporting the plots of the average (over a sufficiently large number of runs) value and standard deviation of the cumulative regret, cumulative reward, instantaneous regret, and instantaneous reward.

Step 4: Contexts and their generation

Consider the case in which there are three classes of users (C1, C2, and C3), and no information about the advertising and pricing curves is known beforehand. Consider two scenarios. In the first one, the structure of the contexts is known beforehand. Apply the GP-UCB and GP-TS algorithms when using GPs to model the two advertising curves, reporting the plots with the average (over a sufficiently large number of runs) value and standard deviation of the cumulative regret, cumulative reward, instantaneous regret, and instantaneous reward. In the second scenario, the structure of the contexts is not known beforehand and needs to be learnt from data. Important remark: the learner does not know how many contexts there are, while it can only observe the features and data associated with the features. Apply the GP-UCB and GP-TS algorithms when using GPs to model the two advertising curves paired with a context generation algorithm, reporting the plots with the average (over a sufficiently large number of runs) value and standard deviation of the cumulative regret, cumulative reward, instantaneous regret, and instantaneous reward. Apply the context generation algorithms every two weeks of the simulation. Compare the performance of the two algorithms --- the one used in the first scenario with the one used in the second scenario. Furthermore, in the second scenario, run the GP-UCB and GP-TS algorithms without context generation, and therefore forcing the context to be only one for the entire time horizon, and compare their performance with the performance of the previous algorithms used for the second scenario.

Step 5: Dealing with non-stationary environments with two abrupt changes

Consider the case in which there is a single-user class C1. Assume that the curve related to the pricing problem is unknown while the curves related to the advertising problems are known. Furthermore, consider the situation in which the curves related to pricing are non-stationary, being subject to seasonal phases (3 different phases spread over the time horizon). Provide motivation for the phases. Apply the UCB1 algorithm and two non-stationary flavors of the UCB1 algorithm defined as follows. The first one is passive and exploits a sliding window, while the second one is active and exploits a change detection test. Provide a sensitivity analysis of the parameters employed in the algorithms, evaluating different values of the length of the sliding window in the first case and different values for the parameters of the change detection test in the second case. Report the plots with the average (over a sufficiently large number of runs) value and standard deviation of the cumulative regret, cumulative reward, instantaneous regret, and instantaneous reward. Compare the results of the three algorithms used.

Step 6: Dealing with non-stationary environments with many abrupt changes

Develop the EXP3 algorithm, which is devoted to dealing with adversarial settings. This algorithm can be also used to deal with non-stationary settings when no information about the specific form of non-stationarity is known beforehand. Consider a simplified version of Step 5 in which the bid is fixed. First, apply the EXP3 algorithm to this setting. The expected result is that EXP3 performs worse than the two non-stationary versions of UCB1. Subsequently, consider a different non-stationary setting with a higher non-stationarity degree. Such a degree can be modeled by having a large number of phases that frequently change. In particular, consider 5 phases, each one associated with a different optimal price, and these phases cyclically change with a high frequency. In this new setting, apply EXP3, UCB1, and the two non-stationary flavors of UBC1. The expected result is that EXP3 outperforms the non-stationary version of UCB1 in this setting.

Matching-Social Influence

Consider a setting in which a company wants to increase the visibility of its products by using social influence techniques to reach possible customers. Then, it has to match the reached customers with some of its products.

Environment

We assume that a round corresponds to one day. The network of customers is organized in a graph defined by:

- a set of 30 customers;
- a set of edges connecting the customers. These edges describe the influence among the customers; we assume that only ~10% (50) of the possible edges are present;
- each edge has a possibly different activation probability;
- for each user, two binary features can be observed by the company, call them F1 and F2; customers can be of three different classes according to these features, call them C1, C2, C3; these three classes differ in terms of the reward of matching the customer with the items;
- at each round, the company can choose three seeds to activate in the social network.

Moreover, the company has three classes of products D1, D2, and D3, and:

- for every product of type Dj and class of customer Ci, specify a reward distribution F(Dj, Ci) of matching the product “j” with the customer “i”; each reward distribution is a Gaussian distribution;
- for every product of type Dj, specify the number of units of this product; we suggest that each type of product has 3 units;
- each unit of product can be matched only with one customer, and each customer can be matched with a single product.

The time horizon to use in the experiments is 365 rounds long. At each round, after the set of seeds is selected, the information cascade and matching is repeated sufficiently many times.

Clairvoyant optimization algorithm

Consider the case in which the company can directly observe the type of each customer Ci. The objective function to maximize is defined as the sum of the expected reward of the couples of matched customers to products. In particular, for each influenced customer of type Ci matched with a product of type Di the reward is the expected value of the distribution F(Dj, Ci).

The optimization algorithm that we suggest is divided into two steps:

1. Find the node that, when it is a seed, gives the highest marginal increase in the number of total activated nodes. Then, fix it as a seed and find the one among the remaining ones that, when added, gives the highest increase. Repeat the same

procedure also for the last node. This is called the greedy algorithm. When looking for the nodes that give the highest increase in the number of total activated nodes, simulate the social influence process by using a Monte Carlo technique with a sufficiently large number of runs.

2. When the optimal set of seeds is fixed, compute the value of the optimum by simulating multiple runs of the social influence process and, for each set of activated nodes, compute the value of the optimal matching. The value of the optimum is computed as an expectation over these runs. If there are more activated users than products, define an opportune number of dummy items such that the total number of items equals the number of users. There is no reward when a user of any class is matched with a dummy item. The case in which there are more items than users can be handled in a similar way.

Step 0: Motivations and environment design

Imagine and motivate a realistic application fitting with the scenario above. Describe all the parameters needed to build the simulator.

Step 1: Learning for social influence

Assume that all the properties of the graph are known except for the edge activation probabilities. Apply the greedy algorithm to the problem of maximizing the expected number of activated customers, where each edge activation probability is replaced with its upper confidence bound (in a UCB1-like fashion). Furthermore, apply the greedy algorithm to the same problem when estimating edge activation probabilities with Beta distributions and sampling is used (in a TS-like fashion). Report the plots with the average (over a sufficiently large number of runs) value and standard deviation of the cumulative regret, cumulative reward, instantaneous regret, and instantaneous reward.

Step 2: Learning for matching

Consider the case in which the company can observe the type of each customer Ci. Moreover, assume that the set of seeds is fixed to the optimal solution found when the activation probabilities are known. On the other hand, suppose that the reward distributions F(Dj, Ci) for the matching are unknown. Apply an upper confidence bound matching algorithm in which the value of a matching is substituted with its upper confidence bound. Do the same using a TS-like algorithm. Report the plots with the average (over a sufficiently large number of runs) value and standard deviation of the cumulative regret, cumulative reward, instantaneous regret, and instantaneous reward.

Step 3: Learning for joint social influence and matching

Consider the case in which the company can observe the type of each customer Ci. Moreover, assume that both the edge activation probabilities and reward distributions F(Dj, Ci) are unknown. Apply jointly the greedy algorithm (for influence maximization) and the matching algorithm using upper confidence bound in place of the edge activation probabilities and the expected reward of each match. Apply jointly the greedy algorithm (for influence maximization) and the matching algorithm using the TS algorithm to estimate the edge activation probabilities and the expected reward of each match. Report the plots of the average value and standard deviation of the cumulative regret, cumulative reward, instantaneous regret, and instantaneous reward.

Step 4: Contexts and their generation

Consider the case in which the company cannot observe the type of each customer Ci, but only the features F1 and F2. Moreover, no information about the edge activation probabilities and the reward distributions F(Dj, Ci) is known beforehand. The structure of the contexts is not known beforehand and needs to be learned from data. Important remark: the learner does not know how many contexts there are, while it can only observe the features and data associated with the features. Apply the UCB and TS algorithms (as in Step 3) paired with a context generation algorithm, reporting the plots with the average (over a sufficiently large number of runs) value and standard deviation of the cumulative regret, cumulative reward, instantaneous regret, and instantaneous reward. Apply the context generation algorithms every two weeks of the simulation. Compare the performance of the designed algorithm with the one in Step 3 (that can observe the context).

Step 5: Dealing with non-stationary environments with two abrupt changes

Assume that all the properties of the graph are known except for the edge activation probabilities. Assume that the edge activation probabilities are non-stationary, being subject to seasonal phases (3 different phases spread over 365 days). Provide motivation for the phases. Apply the greedy algorithm to the problem of maximizing the expected number of activated customers, where each edge activation probability is replaced with its upper confidence bound (in a UCB1-like fashion). Moreover, apply two non-stationary flavors of the algorithm. The first one is passive and exploits a sliding window, while the second one is active and exploits a change detection test. Provide a sensitivity analysis of the algorithms, evaluating different values of the length of the sliding window in the first case and different values for the parameters of the change detection test in the second case. Report the plots of the average value and standard deviation of the cumulative regret, cumulative reward, instantaneous regret, and instantaneous reward.

Step 6: Dealing with non-stationary environments with many abrupt changes

Develop the EXP3 algorithm, which is devoted to dealing with adversarial settings. This algorithm is also used to deal with non-stationary settings when no information about the specific form of non-stationarity is known beforehand. Consider a simplified version of Step 5 in which the company chooses a single seed to activate in the social network at each round. First, apply the EXP3 algorithm and the algorithms designed in Step 5 to this simplified version of the setting. The expected result is that EXP3 performs much worse than the two non-stationary versions of UCB1. Subsequently, consider a different non-stationary setting with a higher non-stationarity degree. Such a degree can be modeled by having a large number of phases that frequently change. In particular, consider 5 phases, each one associated with a different optimal price, and these phases cyclically change with a high frequency. In this new setting, apply EXP3, UCB1, and the two non-stationary flavors of UBC1. The expected result is that EXP3 outperforms the non-stationary flavors of UCB1.

Additional Information about Project Submission and Presentation

The submission should contain the following:

- Some slides describing in detail the project and the results achieved;
- A link to a GitHub repository with the code of the project (do not send notebooks);
- The report should focus on the design of the algorithms and their properties. Details on the actual implementation of the code (e.g., classes and their relationships, packages used) are not required;

The presentation should be conducted as follows:

- The slides should be a subset of those used for the project submission (we expect that the submission will include many more details);
- You will have 20 minutes to deliver your presentation; additional ~10 minutes will be used for questions and answers;
- A detailed discussion of unexpected results is appreciated (e.g. the algorithm does not achieve a sublinear regret).

The dates for project submission and presentation are as follows:

- We have three sessions (end of July, end of September, end of December);
- The deadlines for the project submission are: 21st of July, 22nd of September and 15th of December;
- The presentation will be in the days following the submission;
- We prefer that the presentations will be delivered physically, but it is not mandatory (it is also allowed that the team is partially online).