Online Learning Applications

Project Requirements

Recap on evaluation

The evaluation is composed by two parts:

Written exam

- max 16 points
- Questions about theory

Project

- max 16 points
- Groups of at most 4-5 students
- The goal is to develop an algorithm for a complex problem
- Includes modeling and coding

Overview of the project

Overview of the project

The goal of the project is to design online learning algorithms to handle a **marketing campaign to sell products**. This includes:

- An **advertising** campaign
- A **pricing** problem

Overview of the interaction

A company runs a marketing campaign for some months. The interaction goes as follows:

At each day:

1. The company chooses a price p
2. The company faces a sequence of auctions. For each auction:
    1. The company chooses a bid b
    2. A slot is (possibly) assigned to the company depending on b, the competing bids, and the auction format
    3. If the ad is clicked, an user visit the company web page
    4. The user buy the company product with a probability that depends on the price p

Overview of the interaction

A company runs a marketing campaign for some months. The interaction goes as follows:

At each day:

1. The company chooses a price p
2. The company faces a sequence of auctions. For each auction:
    1. The company chooses a bid b
    2. A slot is (possibly) assigned to the company depending on b, the competing bids, and the auction format
    3. If the ad is clicked, an user visit the company web page
    4. The user buy the company product with a probability that depends on the price p

⚠ **Notice that the price p and the bid b can be changed at different time. A company usually cannot change the price on the website every minute!**

Requirement 1: stochastic environment

Stochastic environment

Build a **stochastic** environment. At an high level, it should include:

- A distribution over the bids of the other agents
- A function specifying the probability with which an user buys for every price

Stochastic environment

Build a **stochastic** environment. At an high level, it should include:

- A distribution over the bids of the other agents
- A function specifying the probability with which an user buys for every price

⚠ **Of course there are other stochastic components like the probability λ_s that slot s is observed. We assume that these are data of the problem and does not change (even in adversarial environments).**

Design algorithms for stochastic environments

Pricing algorithm

Build a pricing strategy using the **continuous** set of prices $p \in [0, 1]$ and **Gaussian Processes**.

Bidding algorithm

Consider a sequence of second-price auctions. Build two learning algorithms to deal with the bidding problem:

- A primal-dual algorithm for **truthful** auctions
- A UCB-like algorithm

Design algorithms for stochastic environments

Pricing algorithm

Build a pricing strategy using the **continuous** set of prices $p \in [0, 1]$ and **Gaussian Processes**.

Bidding algorithm

Consider a sequence of second-price auctions. Build two learning algorithms to deal with the bidding problem:

- A primal-dual algorithm for **truthful** auctions
- A UCB-like algorithm

Run the algorithms separately. Then, run the algorithms together as specified in the interaction in Slide 5. Keep in mind that the number of customers seeing the price depends on the advertising campaign.

Requirement 2: adversarial environment

Adversarial (highly non-stationary) environment

Build an **highly non-stationary** environment. At an high level, it should include:

- A sequence of competing bids of the other agents (e.g., sampled from a distribution that changes **quickly** over time)
- For each day, a function specifying the probability with which an user buys for every price (this function changes **quickly** over time)

Adversarial learning algorithms

The right way to deal with highly non-stationary environment is to use adversarial regret minimizers.

Build a pricing strategy **discretizing** the continuous set of prices $p \in [0, 1]$

Consider a **generalized first-price auction**. Build a learning algorithms to deal with the bidding problem. In particular, a primal-dual algorithm for **non-truthful** auctions.

Adversarial learning algorithms

The right way to deal with highly non-stationary environment is to use adversarial regret minimizers.

Build a pricing strategy **discretizing** the continuous set of prices $p \in [0, 1]$

Consider a **generalized first-price auction**. Build a learning algorithms to deal with the bidding problem. In particular, a primal-dual algorithm for **non-truthful** auctions.

Run the algorithms separately. Then, run the algorithms together as specified in the interaction in Slide 5. Keep in mind that the number of customers seeing the price depends on the advertising campaign.

Requirement 3: two extensions for pricing

Two extensions for pricing

We extend the pricing problem along two directions. Since we focus only on pricing, we directly consider a demand curve **with noise** (differently from previous points in which the demand curve depends on the number of visits and hence on the advertising step).

Non-stationary environment

Build a **non-stationary** environment for the pricing problem. At an high level:

- Days are partitioned in intervals
- In each interval the demand curve is different
- The demand curve **+ noise** specifies how many buyers while buy for every price depending on the **current interval**

Build a pricing strategy using the **discretization** of the prices $p \in [0, 1]$ and:

- **sliding-window**
- **CUSUM**

Multiple products

⚠️ **Bonus point (not required to complete the project)**

Multiple products

Bonus point (not required to complete the project)

Build a **two-item stochastic** pricing environment. In particular, the seller proposes two items i_1 and i_2. The model should include:

- A demand curve $D(p_1, p_2)$ **+ noise** that specifies how many buyers will buy product i_1 and how many will buy product i_2 depending on price p_1 of product i_1 and price p_2 of product i_2

Build a regret minimizer for the continuous action set $[0, 1]^2$ using two-dimensional Gaussian processes.

Requirement 4: Compare different bidding algorithms

Competition among different algorithms

The goal is to compare the performances of different algorithms.

Consider the different algorithms that we have seen for bidding and let them play one against the others in a **generalized first-price auction**. In particular, consider:

- A primal-dual algorithm for **truthful** auctions
- A primal-dual algorithm for **non-truthful** auctions
- A UCB-like approach

Competition among different algorithms

The goal is to compare the performances of different algorithms.

Consider the different algorithms that we have seen for bidding and let them play one against the others in a **generalized first-price auction**. In particular, consider:

- A primal-dual algorithm for **truthful** auctions
- A primal-dual algorithm for **non-truthful** auctions
- A UCB-like approach

⚠ **Note that you can include more copies of each regret minimizer to increase competition.**

Competition among different algorithms

The goal is to compare the performances of different algorithms.

Consider the different algorithms that we have seen for bidding and let them play one against the others in a **generalized first-price auction**. In particular, consider:

- A primal-dual algorithm for **truthful** auctions
- A primal-dual algorithm for **non-truthful** auctions
- A UCB-like approach

Consider different problem parameters. Which regret minimizer performs best in each context?

Deliverable and presentation

Deliverable

Before the presentation you should deliver:

- Some slides describing in details the project and the results achieved:
  - **High level details** about the implemented algorithms
  - **graphs** of the empirical results
- A link to a GitHub repository with the code of the project

The presentation should be conducted as follows:

- 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)

Details about the project presentation schedule

You have three possible sessions to present the project in **July, September, December**;

- The deadline for the July project submission is **12th of July**
- The presentations will be in the days following the submission