Note: See ORIE 5355 Fall 2024 Project instructions for an overview of the project, as well as logistics. This document is intended purely as informational/suggestive for the strategy that you might undertake in each part – the project is open ended in terms of strategies you take.

Part 1: Personalized pricing with capacity constraints

Demand estimation and pricing

Using the provided train set customer prices and decisions, you can adapt your code from HW3 - Problem 1 to tackle the problem of demand estimation and pricing *without* capacity constraints. For example, you might want to train a logistic regression Purchased ~ Price + Covariates, which would capture the dynamic of varying demand in response to price and customer covariates. Then, you could do a grid search and find some value p, such that for a given set of covariates, p maximizes

$$p * \text{Purchased}(p, \text{Covariates})$$

That is, p is the single-step revenue maximizing price to offer.

- Question: since we enforce a 0.5 second limit on the time you are allowed to generate a decision for each customer, can you do it faster than a grid search? Can you do this extremely efficiently under a logistic regression model?

Pricing with capacity constraints

When you have capacity constraints, a very natural thing to consider is *should I offer the product to this customer at all?* Imagine a scenario where the current customer is predicted to have a really low willingness to pay at higher prices, so the expected revenue from the current customer is low. Would it be better to 'reject' the current customer (e.g., by offering an extremely high price) and hope you will get better luck with the next one?

Here, a dynamic programming approach might be useful, which you practiced in HW3 - Problem 3 (the setting is more similar to 3(c)).

To make things simpler, you can assume that all the customers are drawn from the same distribution you observe in the train set.

- Question: how do you integrate dynamic programming into real-time pricing decisions, and make it fast?
- Question: can you simplify the problem? Essentially you are comparing the expected value of a unit of inventory for the current customer versus an 'average' customer in the future. Intuitively, you may want to "save" a product for a future customer if you are low on stock and the current customer has a low valuation for the item.

Testing your strategy

You can test your strategy using the notebook we provide. Again, you can assume that the customers that we will grade your strategy on come from the same distribution as what we have programmed into the code in the notebook, and you can assume in the final grading we will be looking at the mean of your strategy's profit over a large number of customers. (2500 timesteps)

- Question: if you have two strategies A and B, and you know A performs better than B, how would you iterate for an even better one?

Part 2: Pricing competition

In this part, you must consider pricing in the presence of competition from your opposing team: if they offer low prices, then the customer will buy from them instead of buying from you. Thus, your task is to price high enough that you make revenue from a sale, but low enough such that the customer buys from you and not your competitor.

[Note that it does not matter how many *individual games* you win, just your ***revenue*** across games; thus, pricing near 0 for every item in every round is unlikely to be an effective strategy].

Note that after each customer, you will observe the price your opponent set, as well as the customer's decision.

Responding to your competitor's action

Even before considering that your opponent is strategic and so can react to your choices, consider that they have some demand prediction/revenue maximization model that is outputting prices for the customer, just like you do. Their model might be similar or different to your model - after all, there are only a few observable features of each customer, and there is a limit to the demand estimation you can do.

- Question: Can you learn/incorporate the tendencies of their model, such as the tendency to price items for certain customers higher than you do? In other words, can you predict the prices that your opponent will set for a given customer if your opponent faced no competition?
- Further question: How can you evaluate such a model during the competition? How does the output of this model help you?

Consider the following naïve heuristic. Suppose the customer valuation is v, and so optimal pricing without competition is exactly $p = v$. Then, with competition, instead of setting that price, instead set the price as $\alpha * p$, where α depends on how high your opponent prices tend to be. For example, if your opponent always prices things at infinity (equivalently, you have no opponent), then $\alpha = 1$. If instead your opponent sets very low prices, then set α close to 0. When running the competition, you can further tune your α parameter adaptively. Though quite simple, we found this to be a good starting point for many teams in previous years' similar competitions. This draws a link to the "adaptive experimentation" idea we briefly discussed in class. However, one pitfall of this strategy without modification is that it may lead to both teams pricing at near α = 0, which will lead to low prices and revenue.

Competition versus cooperation

You might have noticed the interesting choice of inventory: 12 units for both parties, for every 20 arriving customers. This leaves room for cooperation between you and your competitors: if the final goal is to maximize revenue, instead of fighting tooth and nail to win over every customer,

why don't each of us sell to 10 customers, and try to maximize our revenue from them? However, such perfect cooperation can easily break: suppose you and your competitor 'agrees' to alternate your offers, i.e., you will offer [1000, v1], [v2, 1000], [1000, v3], [v4, 1000], and so on, so ideally the customers will alternative between you two, and you both sell to 10 customers for every 20 rounds. Now note that your competitor will still have 2 units of inventory left from this, so they might think they can beat you by surprise in two of the rounds that they were supposed to offer 1000, and instead offer a really low, but positive price. This is guaranteed to make them better off. This is similar to the classical prisoner's dilemma problem in two-person games.

- Question: How to achieve a good balance between cooperation and competition between you and your competitor? We would encourage you to think about the iterated prisoner's dilemma problem, where a lot of different strategies have been proposed. One popular strategy is the so-called 'tit-for-tat': roughly speaking, cooperate when your opponent is cooperative, and compete when your opponent is competitive.
- Question: How do you determine the intent of your competitor during the competition?

Testing strategy

One thing you might be wondering is how to test your agent? Your opponents might take a variety of strategies, and your agent should do well against many of them. (Remember, only overall revenue across games matters). We recommend playing your agent against itself and against other dummy agents, for example:

- An agent that just outputs the optimal price without competition.
- An agent that outputs optimal price, with some adaptive adjustments.
- An agent created by another team in the class (this is an exception to the rule that you should not possess code from other groups; you still cannot copy code, but you are allowed to transfer the agents into the same computer in order to simulate a game).

Speeding up your strategy

In Part 2, the 0.5s/customer time constraint is really going to matter. Now, in addition to pricing for yourself, you need to take into account the best response to your competitor, and that adds a lot of work.

We will be posting leaderboards for Part 2 before the deadline as well, so you can also refer to that as an evaluation of your agent's overall performance.