

ORIE 5355

Lecture 10: Algorithmic pricing: price differentiation, competition, and practice

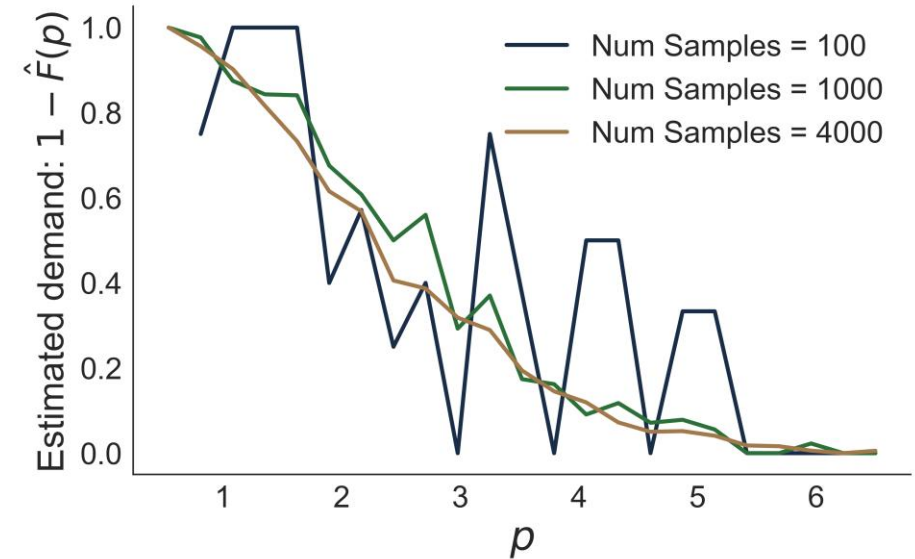
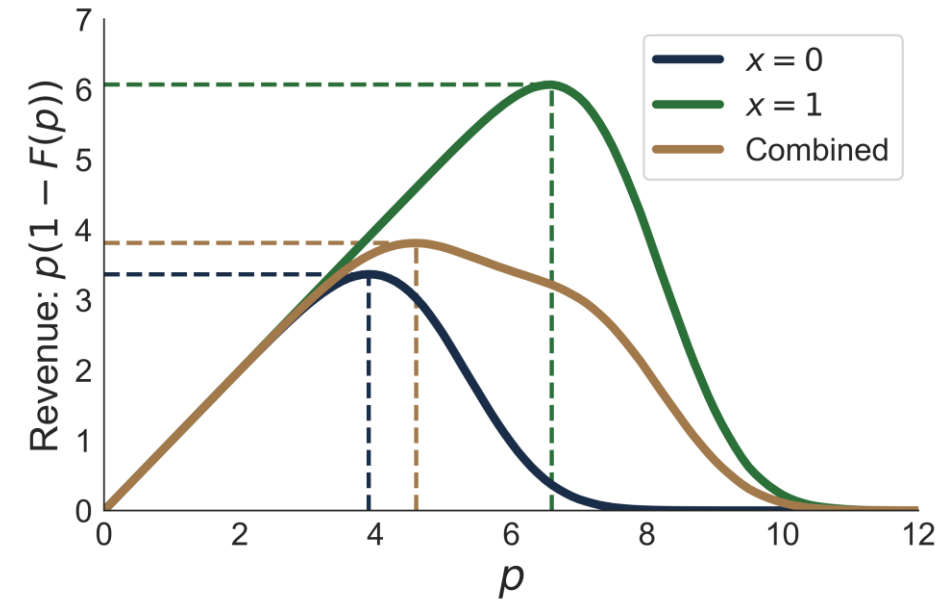
Nikhil Garg

Announcements & reminders

- HW 3 posted
- In person pricing ethics discussion 10/22! **Important**
- [PollEv.com/nikhilgarg713](https://poll-ev.com/nikhilgarg713)

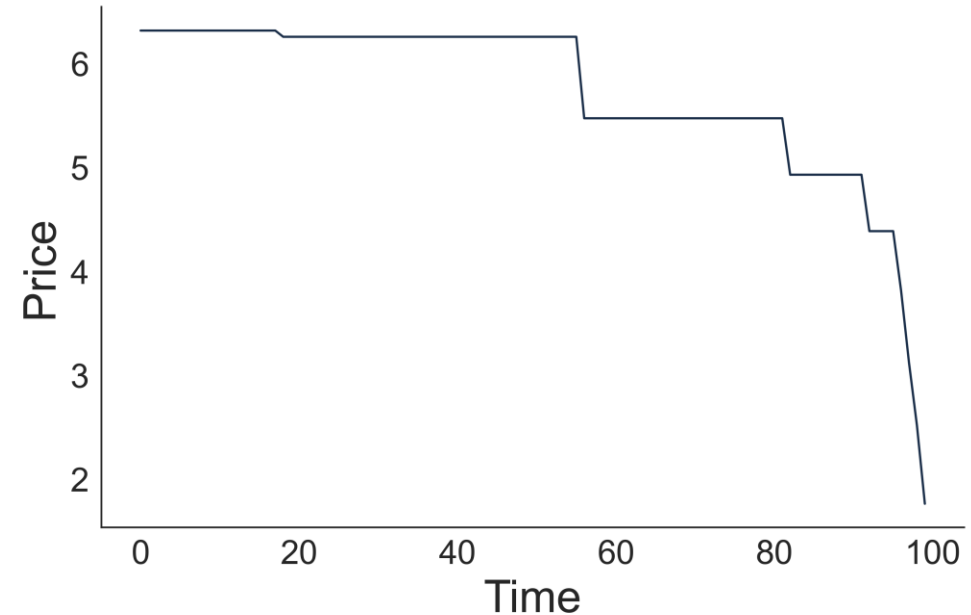
Pricing so far

- Given a demand distribution $d(p) = 1 - F(p)$, how to calculate optimal prices
$$\arg \max_p [p \times d(p)]$$
- How to estimate demand distributions, potentially as a function of covariates



Capacity constraints and pricing over time

- Dynamic programming approach
- If you have T time periods to sell an item and want to maximize expected revenue, what prices $p_1 \dots p_T$ do you set?
- Key idea: optimize backwards
 - First decide price p_T
 - Then decide price p_{T-1}
- Posted additional notes; come to OHs for additional questions



Then: with $T-1=3$ as last day to sell item

$$V_{T-1} = V_3 = d(P_3)P_3 + \cancel{(1-d(P_3))0}$$

$$V_2 = d(P_2)P_2 + (1-d(P_2))V_3$$

$$V_1 = d(P_1)P_1 + (1-d(P_1))V_2$$

$$V_0 = d(P_0)P_0 + (1-d(P_0))V_1$$

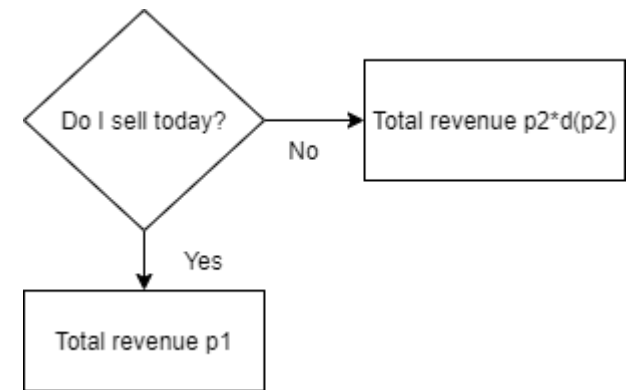
General equation:

$$V_T = 0$$

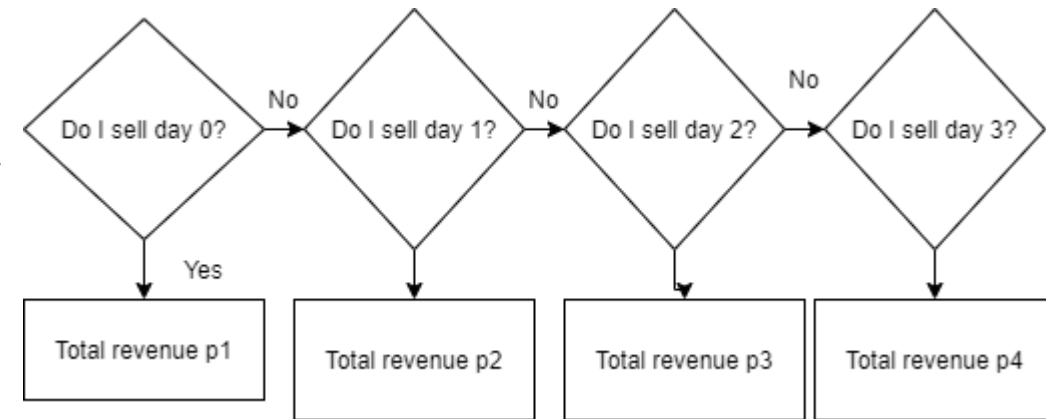
for $t=0 \dots t=T-1$:
$$V_t = d(P_t)P_t + (1-d(P_t))V_{t+1}$$

Solving the example: “Bellman equation”

- If I don't sell today: (happens with probability $1 - d(p_1)$)
 - Then my revenue today is 0
 - Then the expected revenue tomorrow is: $p_2 d(p_2)$
- If I do sell today: (happens with probability $d(p_1)$)
 - My revenue today is p_1
 - Then the expected revenue tomorrow is 0
- So, my overall expected revenue is:
$$d(p_1)(p_1 + 0) + (1 - d(p_1))(0 + p_2 d(p_2))$$
- p_2 easy to solve – does not depend on p_1
- Given p_2 , the above revenue function is only a function of $p_1 \Rightarrow$ Can optimize p_1



Bellman equation generally



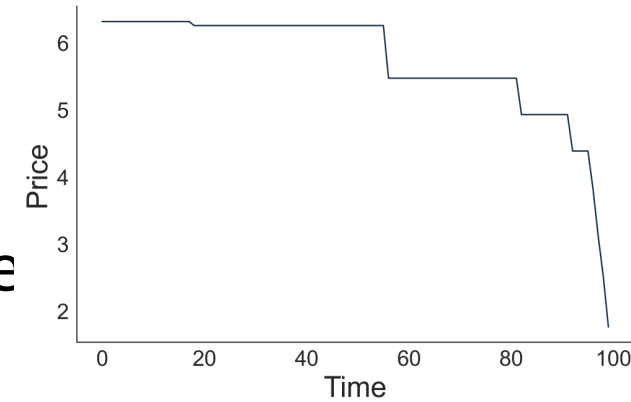
- You can generalize this idea to selling any number of items sequentially for T days
- Start from Day T : If you still have an item, do single-shot maximization
- Day $T - 1$: Given Day T price, you know expected reward if you still have an item to be sold after Day $T - 1$. And so, you can calculate optimal price for Day $T - 1$.
- Now, you have the expected reward if you still have an item to be sold after Day $T - 2$...
- All the way until Day $T = 1$

More Bellman equation

- Let V_t denote: “Expected profit if I still have an item to sell on day t ”

$$V_T = p_T \times d(p_T)$$
$$V_{T-1} = [p_{T-1} \times d(p_{T-1})] + (1 - d(p_{T-1})) V_T$$

- Above means: “Value today is revenue today if I sell the item today, or tomorrow’s expected revenue if I don’t sell the item today”
- For each t , given V_{t+1} we can calculate optimal price p_t
- Keep iterating until you have prices $p_1 \dots p_T$
- Resulting V_1 is my expected revenue given these prices



Bellman equations: a general idea

- Constructing a tree to reason about what happens tomorrow, and then iterating backwards, is a powerful + flexible algorithmic technique: “dynamic programming”
- Example: What if you have 5 copies of each item?

Let k denote how many copies of the item I have. Then:

$$V_{t,0} = 0 \text{ for all } t$$
$$V_{t,k} = \max_{p_{t,k}} d(p_{t,k})[p_{t,k} + V_{t+1,k-1}] + (1 - d(p_{t,k})) V_{t+1,k}$$

If I sell an item today: Revenue today, plus future revenue from 1 less item

If I don't sell: Future revenue from same number of items

Competing effects: Now, less capacity over time $\rightarrow$ prices should go up (but less time to sell, so prices should go down).

Capacity constraints + over-time pricing in practice

- Dynamic programs/bellman equations are powerful, but often the real world is too complicated
 - Uncertainty in future capacity
 - Future actions of competitors
 - Future demand distributions
 - “Long time horizons” (T is big)
- In theory, dynamic programming can handle the above. In practice, hard to know how to calculate future value.

Approximating dynamic programming

- In the recommendations module, we created “score”(or “index”) functions:
 - Consider future users, through capacity and avg ratings terms in the score function
- With 1 item: V_{t+1} represents my “opportunity cost” if I sell an item today that I could have sold tomorrow.

Also interpret as “safety net”: if fail to sell the item today, still earn V_{t+1} in expectation
- Instead of doing a full Bellman equation, estimate V_{t+1} through some other means, then plug into the decision problem for today (finding price p_t)
 - Can construct it like we did score functions for recommendations
 - AlphaGo to play Go: V_{t+1} is partially estimated via a neural network

Pricing with capacity summary

- Just like in recommendations, have to think about potential future demand
- Here, potential future demand lets us be “more aggressive” by pricing higher today
- If I can summarize future revenue (V_{t+1}) effectively, then I can optimize today's prices
- Dynamic programming: start from the end!
- We assumed that customers can't strategize on when to come – not true!

Questions?

Plan for rest of today

So far:

- A little bit on using side-information (user and item vectors) to estimate personalized demand
- Capacity constraints over time

Many assumptions from so far:

- Only one item
- Allowed to explicitly give different prices to different users
 - Or give different prices over time
- No competition from other sellers
- No over-time dynamics

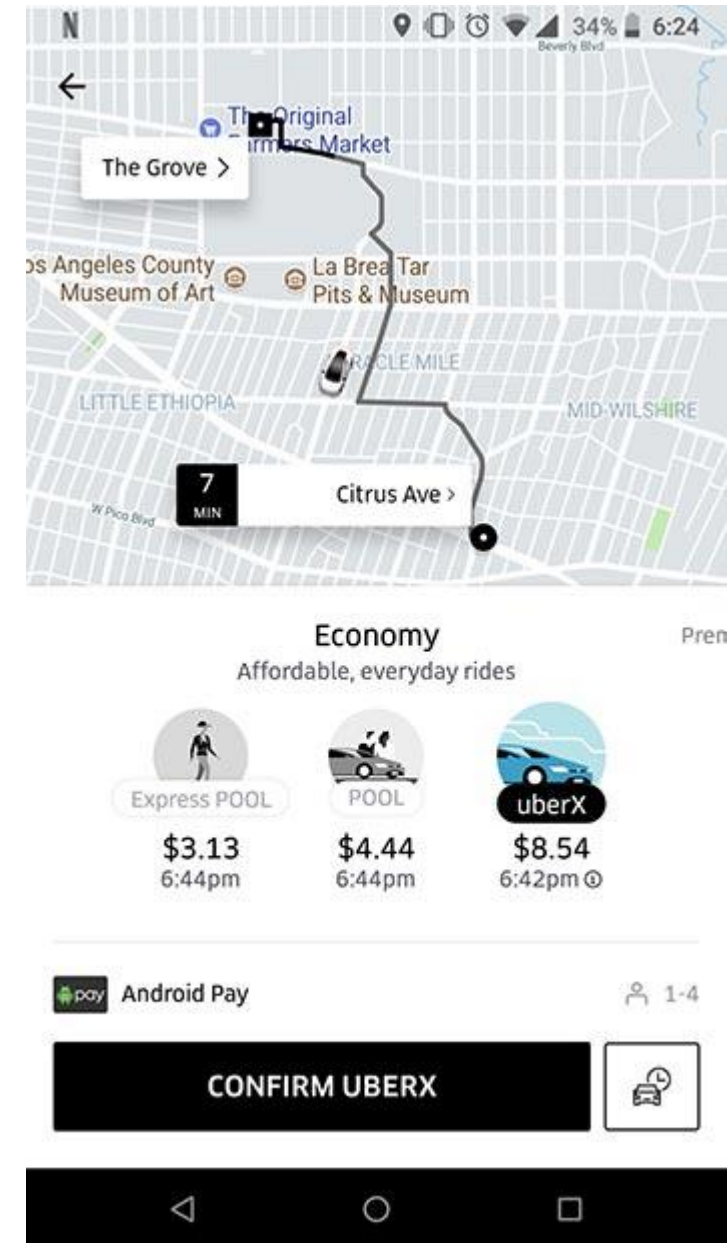
We'll peel back some more of these assumptions today

Selling multiple kinds of items

Price differentiation

Example

- Ride-hailing offers different “tiers” of service
- UberPool cheaper than UberX
 - Also costs less for the platform
- How do we price these items together?
 - What happens if we do simple revenue maximizing price for each item separately?
 - What happens if we make UberPool cheaper?



Motivation

Motivation 1:

You simply have multiple kinds of products to sell. Different types of clothes, laptops, airline seats, furniture, etc.

Motivation 2:

- Earlier: personalized pricing with covariates
- Challenge: Often you can't (technically, ethically, legally, ...) give different prices for the same product to different users based on covariates
- Now: Different "tiers" of service.
 - High quality: First class seats, faster service in Uber/Lyft, luxury goods versions, get item "now"
 - Lower quality: Economy seats, UberPool/Lyft Wait and Save, ...

=> Purposely create tiers of service to earn more money from richer people while earning something from others

Challenges

- Just like pricing over time, now prices for the 2 items depend on each other
 - Unlike pricing to different demographic segments without capacity constraints
- Cannibalization: Customers who would have bought the luxury good instead buy the cheaper good because it is available

2-item user behavior model

- Suppose you're selling 2 types of items
- Each person will buy at most one item
 - Each person has a *private valuation* v_1 for item 1 and v_2 for item 2
- Suppose you offer the items at price p_1 and p_2 , respectively
- How does the person make their decision?

Utility from item j at price p_j is $v_j - p_j$

- Person i buys

Neither item if $v_1 < p_1$ and $v_2 < p_2$

Item 1 if $v_1 \geq p_1$ and $v_1 - p_1 \geq v_2 - p_2$

Item 2 if $v_2 \geq p_2$ and $v_2 - p_2 \geq v_1 - p_1$

Assumption on customer's "choice model." More generally, customer could buy randomly, with choice probabilities that depend on

$v_j - p_j$

In more detail

How does the person make their decision? Person i buys

Neither item if $v_1 < p_1$ and $v_2 < p_2$

Item 1 if $v_1 \geq p_1$ and $v_1 - p_1 \geq v_2 - p_2$

Item 2 if $v_2 \geq p_2$ and $v_2 - p_2 \geq v_1 - p_1$



Revenue in 2 item model

For a set of prices (p_1, p_2) , let

$d_1(p_1, p_2)$ be fraction of people who buy item 1
(Yellow Region)

$d_2(p_1, p_2)$ be fraction of people who buy item 2
(Blue Region)

Then, revenue is:

$$p_1 \times d_1(p_1, p_2) + p_2 \times d_2(p_1, p_2)$$

Given functions d_1, d_2 , can solve for optimal prices

For a set of prices (p_1, p_2) , let

$d_1(p_1, p_2)$ be fraction of people who buy item 1
(Yellow Region)

$d_2(p_1, p_2)$ be fraction of people who buy item 2
(Blue Region)

Then, revenue is:

$$p_1 \times d_1(p_1, p_2) + p_2 \times d_2(p_1, p_2)$$

Given functions d_1, d_2 , can solve for optimal prices

Cannibalization

Now, each price affects other item.

Revenue: $p_1 \times d_1(p_1, p_2) + p_2 \times d_2(p_1, p_2)$

Suppose decrease p_1 (make item 1 cheaper)

Then:

- Earn less money in yellow region ↓
- Yellow region becomes bigger
 - White region becomes smaller ↑
 - Blue region becomes smaller ↓**



Demand estimation with multiple items

- With a single item, we suggested machine learning approach to estimate: $d(p, x) \stackrel{\text{def}}{=} 1 - F_{p|X}(p | X = x)$
- Assume we have user i with covariates x_i
- Now, would need to estimate $d_1(p_1, p_2, x_i)$ and $d_2(p_1, p_2, x_i)$
- Gets very hard, very quickly
- Approach 1: Use a *multi-class* classification algorithm $g(p_1, p_2, x_i)$
[Buy nothing, buy item 1, buy item 2] and then extract class probabilities
(sci-kit learn: use **predict_proba** with any multi-class classifier)
- Approach 2: (Extend idea from previous class)
 - Use user and item vectors, i.e., $(p_1, p_2, u_i \cdot w_{\text{item } 1}, u_i \cdot w_{\text{item } 2})$

Sidenote: Substitutes and complements

- So far: motivation -- we have multiple products to sell, that appeal to different customers
 - “cheaper” and “more expensive” product
- Items are “substitutes”: people only buy at most one kind of item
- Sometimes, items are “complements” – buying one item makes the other item more attractive
 - Soda + popcorn at movie theater
 - iPhone and Macbook and Apple Watch and Apple TV and ...
- Then, reducing one item’s price might induce you to buy more overall
 - An item is a “loss leader”

Putting pieces together: class
competition

So far we've covered

- Recommendation systems
 - Given past user and item data, predicting how much each user would like each item
 - How to turn these predictions into *recommendations* (with capacity constraints)
- Pricing
 - Single item revenue maximization
 - Estimating demand at each price, potentially with covariates
 - Potentially with multiple items, and with using user and item vectors
 - Pricing over time with capacity constraints
 - Pricing multiple items

Overview: Real-life algorithmic pricing

- You and a single competitor (your classmates) each are selling two types of items, Book A and Book B.
 - (Potentially: suppose you get K copies of each item every 10 steps)
- A customer walks in and you observe some personal data
 - Just demographic covariates
 - Demographic covariates & user vector trained using their past experiences
- You and your competitor post prices for each item
- The customer at most 1 item and leaves
- Repeat for many customers over time

Basic case

- For now, let's ignore competition
- For each user, you have either just demographic covariates x_i or also a trained user vector u_i from their past interactions on your site
- You would predict demand for each item, $d_1(p_A, p_B, x_i, u_i)$ and $d_2(p_A, p_B, x_i, u_i)$ for each set of prices (p_A, p_B)
 - Your choice on how to estimate this demand
 - What do you do for customers with no user vector u_i ?
- Set prices to maximize your expected revenue

Complication 1: Capacity constraints

- Now, have K copies of each item for each $T=10$ customers.
- Now, the price that you set for each item should depend on opportunity cost: what if you can sell that item to a different customer in the future?
- 3-d Bellman equation: time, capacity of Book A, capacity of Book B
- Set up your Bellman equation:

$$V_{t,k_A,k_B} = A + B + C$$

A: If I sell Book A today: Revenue today, plus future revenue from 1 less Book A

B: If I sell Book B today : Revenue today, plus future revenue from 1 less Book B

C: If I don't sell anything: future revenue from same number of copies

How to calculate future revenue?

- As before, future revenue depends on future prices that you set
- ...Think about prices you'd set on last day $T-1=10$
 - For each combination of capacities left k_A, k_B
- Complication: on day $t < T - 1$ you don't yet know the customer x_{T-1}, u_{T-1} that will show up on the last day $T - 1$!
 - You only know customer who has shown up on day t
- When calculating future *expected* value V_{t+1, k_A, k_B} , you need to consider the *distribution* of customers that *could* show up
 - Use training data to consider possible customers that could show up
 - Then calculate the prices that you *would* show each of them

Complication: Competition

- You and your opponent both do the same thing, and calculate the exact same prices p_A, p_B at the current time step
- Your opponent is clever, and so decides to *undercut* you slightly, and so sets prices $p_A - \$0.01, p_B - \0.01
- ...but you're cleverer, and know your opponent will do this, and so you set prices $p_A - \$0.02, p_B - \0.02
- There's now a game theory component: you need to anticipate what your opponent will do when setting prices
- More complicated: it's a repeated setting
 - You can *learn parameters* for how your opponent behaves

Questions?