ORIE 5355

Lecture 9: Algorithmic pricing: capacity, price differentiation, and competition

Nikhil Garg

Announcements

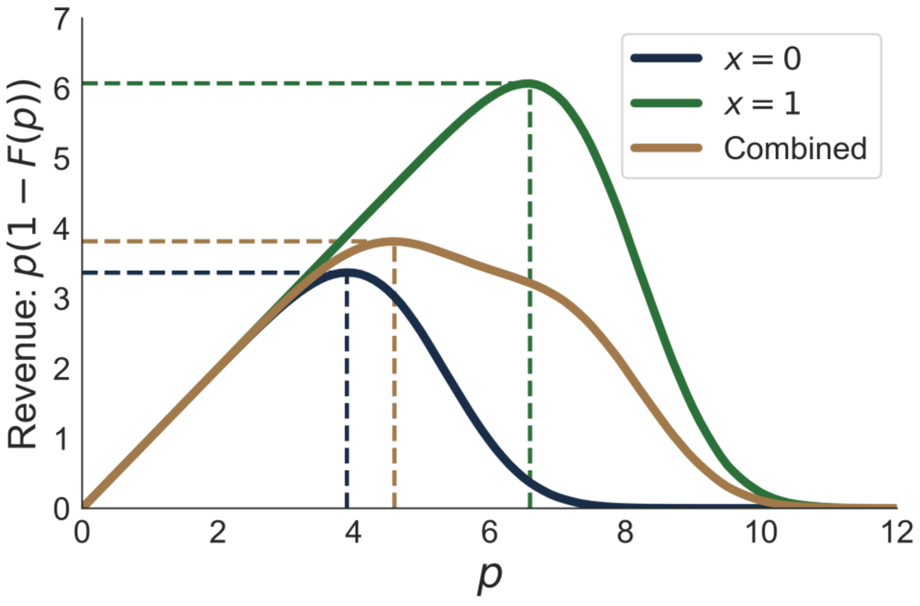
- HW2 due tomorrow
- Quiz 2 released – due Friday evening
- HW 3 released – due Tuesday 10/18
  - Conceptual component of HW due by class time on 10/05

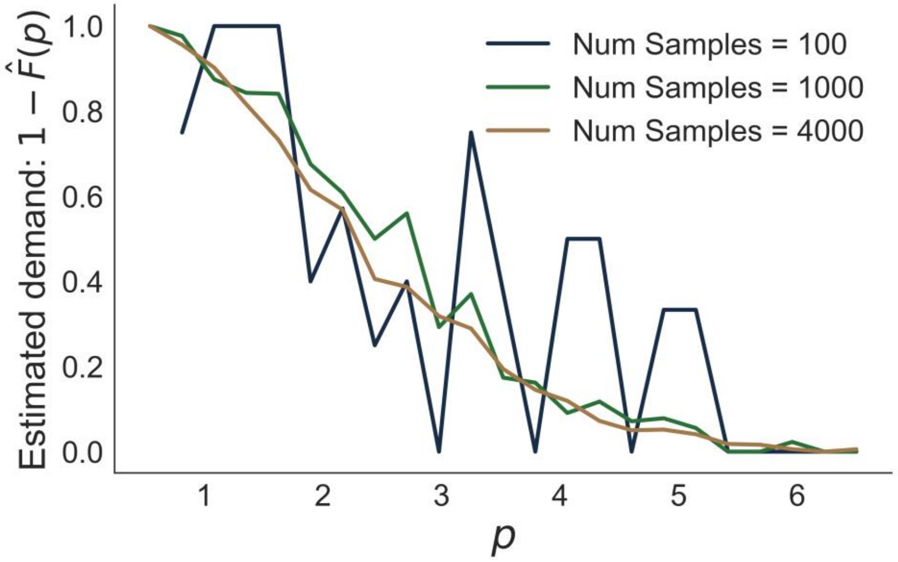
Last time

- 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





More on demand estimation

- We want to estimate $\mathrm{d(p,x)} \stackrel{\text{def}}{=} 1 - F_{p|X}\,(p \mid X = x)$
- Last time: Logistic regression
  - Target (Y variable) is purchase decision $\mathrm{d(p,x)}$
  - Covariates $\mathrm{(p,x)}$ are: price offered, user covariates, interactions between price and covariates or between covariates
- Challenge: what if you have many items you're selling (separately)? This wastes information (can't use models across items)
- Alternative: Use idea from recommendations! Suppose you have user vector $\mathrm{u_i}$ and item vector $\mathrm{w_j}$. Then, ML model to learn with covariates: $(\mathrm{p}, u_i \cdot w_j)$
  - Can learn demand for items you haven't sold before at certain prices!
  - (Or completely new items, using KNN approach from recommendations)
  - Allows incorporating other information you have about items, that helped you learn the item vectors

Plan for today

Many assumptions last time:

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

We'll peel back some of these assumptions today

Capacity constraints and pricing over time

Setting and examples

You often are trying to sell limited quantities of a good, to many potential customers over time

- Airline tickets – the airline “wastes” a seat that’s unsold
  - Same for concerts, sports, any event with a fixed date
  - Clothes that are going out of season/fashion
  - Electronics that become obsolete over time
- Any retail setting with inventory constraints
- Often 2 competing effects:
  - The items become less valuable over time, or you have a deadline to sell them
  - You have less stock over time

Simplified example

- You have 1 copy of the item to sell
- There are 2 time periods, today and tomorrow
  - One customer will come in today, a different one tomorrow
- No covariates
- No “discounting” (a dollar tomorrow is as valuable as a dollar today)
- You already have a good estimate of $\mathrm{d}(\mathrm{p})$

What price p_1 do you set today? What price p_2 do you set tomorrow?

A couple of observations

What I do today depends on what I can/will do tomorrow.

- I can't set p_1 unless I know how I will set p_2 in *each scenario*. (whether I sold the item today, or whether I didn't).
- I have to "simulate" the future

If I don't sell the item today, then *tomorrow* I am solving the same problem that we solved in class last time:

- Maximizing revenue for a single buyer/without capacity considerations
- => The price for tomorrow will be same as simple revenue maximizing price

$$p_2 = \arg\max_p \, [p \times d(p)]$$

Not true for the price today:

- If I sell the item today, then I lose out on a potential sale tomorrow
- If I don't sell the item today, I get another chance tomorrow

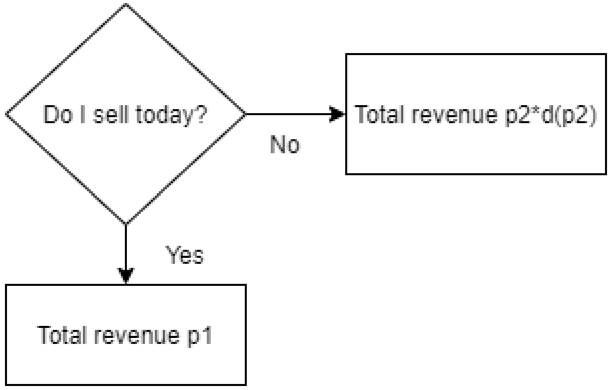
=> I should "take a risk" today to try to sell at a higher price

Solving the example: “Bellman equation”

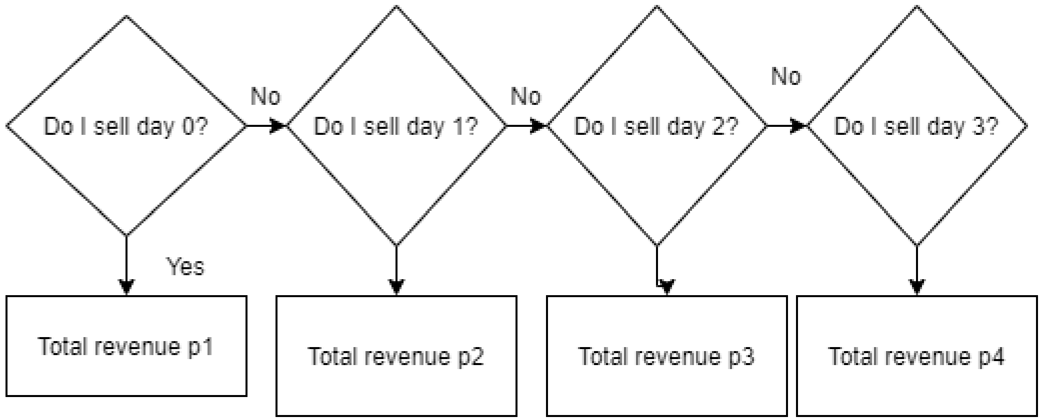
- If I don’t sell today: (happens with probability $1 - d(p_1)$)
  - Then my revenue today is 0
  - Then the expected revenue tomorrow is: $p_2 d(p_2)$
- If I *do* sell today: (happens with probability $d(p_1)$)
  - My revenue today is p_1
  - Then the expected revenue tomorrow is 0
- So, my overall expected revenue is:

$$d(p_1)(p_1 + 0) + \left(1 - d(p_1)\right)\left(0 + p_2 d(p_2)\right)$$

- 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 => *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 $\mathrm{T}-1$: Given Day T price, you know expected reward if you still have an item to be sold after Day $\mathrm{T}-1$. And so, you can calculate optimal price for Day $\mathrm{T}-1$.
- Now, you have the expected reward if you still have an item to be sold after Day $\mathrm{T}-2$...

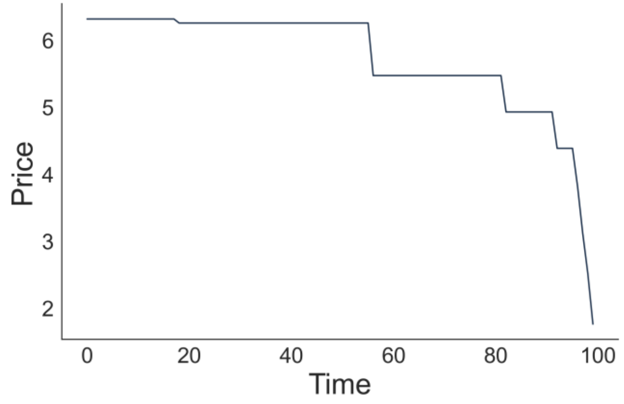
More Bellman equation

- Let $\mathrm{V_t}$ denote: "Expected profit if I still have an item to sell on day t"

$$\mathrm{V_T} = p_T \times d(p_T)$$

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

- 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 $p_0 \ldots p_T$
- Resulting $\mathrm{V_0}$ 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:

$$V_{t,0} = 0 \text{ for all } t$$

$$V_{t,k} = \max_{p_{t,k}} d(p_{t,k})\left[p_{t,k} + V_{t+1,k-1}\right] + \left(1 - d(p_{t,k})\right) V_{t+1,k}$$

  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 → 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?