CMSC 35401: The Interplay of Economics and ML (Winter 2024)

Introduction

Instructor: Haifeng Xu

Outline

- Course Overview
- Administrivia
- An Example

Single-Agent Decision Making

- A decision maker picks an action $x \in X$, resulting in utility $f(x)$
- Typically an optimization problem:

$$\begin{array}{ll} \text{minimize (or maximize)} & f(x) \\ \text{subject to} & x \in X \end{array}$$

  - x: decision variable
  - $f(x)$: objective function
  - X: feasible set/region
  - Optimal solution, optimal value

- Example 1: minimize x^2, s.t. $x \in [-1,1]$

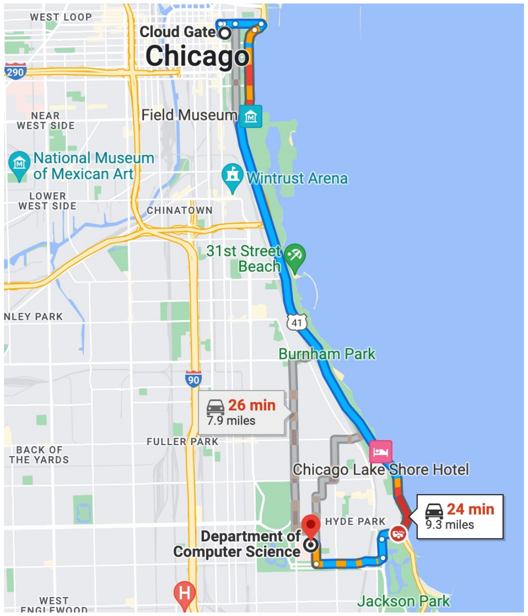
Single-Agent Decision Making

- A decision maker picks an action $x \in X$, resulting in utility $f(x)$
- Typically an optimization problem:

$$\begin{array}{ll} \text{minimize (or maximize)} & f(x) \\ \text{subject to} & x \in X \end{array}$$

- x: decision variable
- $f(x)$: objective function
- X: feasible set/region
- Optimal solution, optimal value

- Example 1: minimize x^2, s.t. $x \in [-1,1]$
- Example 2: pick a road to school



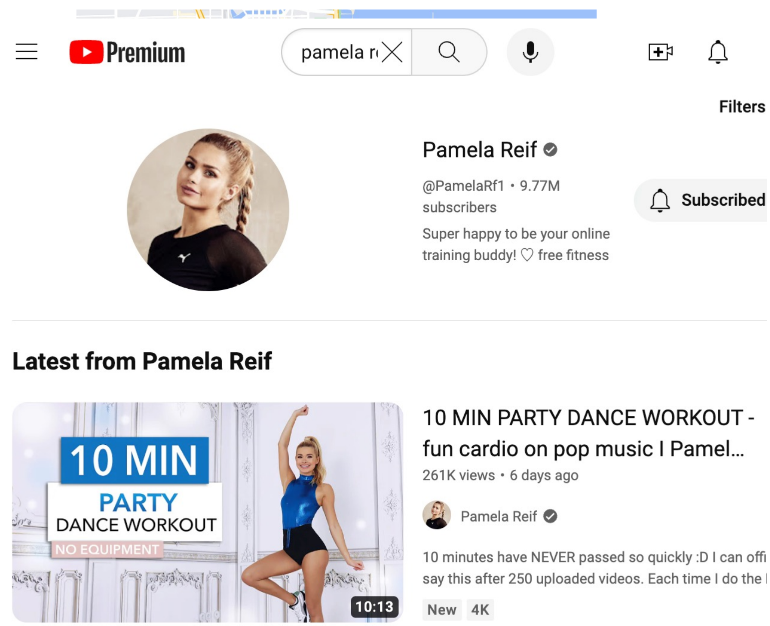
Single-Agent Decision Making

- A decision maker picks an action $x \in X$, resulting in utility $f(x)$
- Typically an optimization problem:

$$
\begin{array}{ll}
\text{minimize (or maximize)} & f(x) \\
\text{subject to} & x \in X
\end{array}
$$

  - x: decision variable
  - $f(x)$: objective function
  - X: feasible set/region
  - Optimal solution, optimal value

- Example 1: minimize x^2, s.t. $x \in [-1,1]$
- Example 2: pick a road to school
- Example 3: build a Youtube channel



Multi-Agent Decision Making

- Usually, your payoffs affected not only by your actions, but also others'
- Agent i's utility $f_i(x_i, x_{-i})$ depends on his own action x_i, as well as other agents' actions x_{-i}
- Is this still an optimization problem? Should each agent i just pick $x_i \in X_i$ to minimize $f_i(x_i, x_{-i})$?
  - x_{-i} is not under i's control
  - Think of rock-paper-scissor game
- Examples: build a Youtube channel, routing, sales, even taking courses…

Example I: Prisoner's Dilemma

- Two members A,B of a criminal gang are arrested
- They are questioned in two separate rooms
  - No communications between them

B / A	B stays silent	B betrays
A stays silent	-1 $\boxed{-1}$	0 -3
A betrays	-3 $\boxed{0}$	-2 -2

Q: How should each prisoner act?

- Betray is always the best action

Example I: Prisoner's Dilemma

- Two members A,B of a criminal gang are arrested
- They are questioned in two separate rooms
  - No communications between them

B A	B stays silent	B betrays
A stays silent	-1 -1	0 $\boxed{-3}$
A betrays	-3 0	-2 $\boxed{-2}$

Q: How should each prisoner act?

- Betray is always the best action

Example I: Prisoner's Dilemma

- ➢ Two members A,B of a criminal gang are arrested
- ➢ They are questioned in two separate rooms
  - ❖ No communications between them

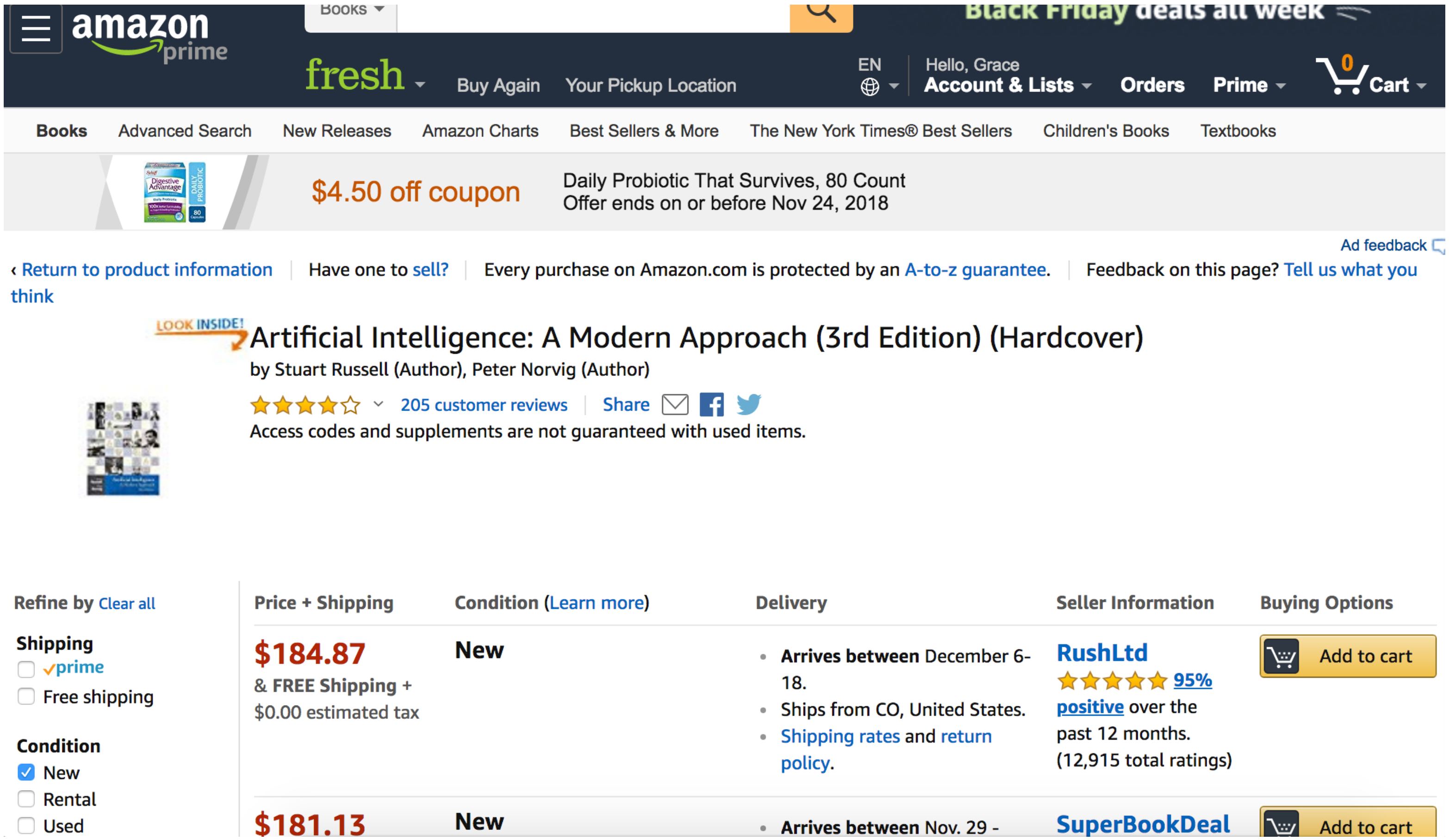
B A	B stays silent	B betrays
A stays silent	-1 -1	0 -3
A betrays	-3 0	$\boxed{-2 \quad -2}$

equilibrium

Q: How should each prisoner act?

- ➢ Betray is always the best action
- ➢ But, (-1,-1) is a better outcome for both
- ➢ Why? What goes wrong?
  - • Selfish behaviors lead to inefficient outcome

Example II: Markets on Amazon

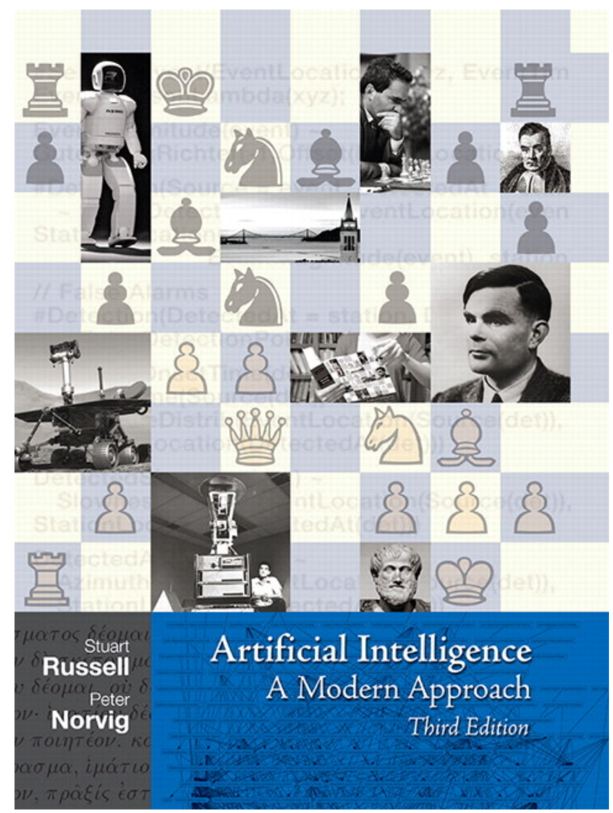


Example II: Markets on Amazon

- Assume people will buy if the book price ≤ $200
- Product cost = $20

If the market has only one book seller…

Q: What price should this monopoly set?

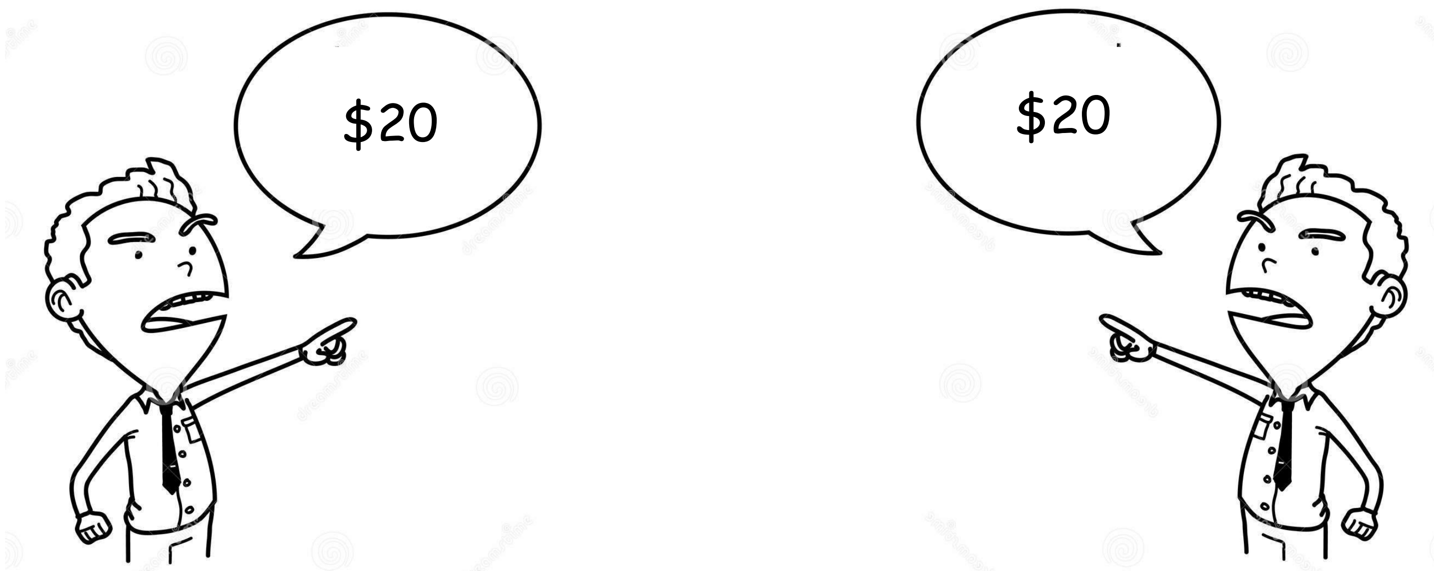


Example II: Markets on Amazon

- Assume people will buy if the book price $\leq$ \$200
- Product cost = \$20

What if the market has two book sellers…

Q: What price should each seller set?



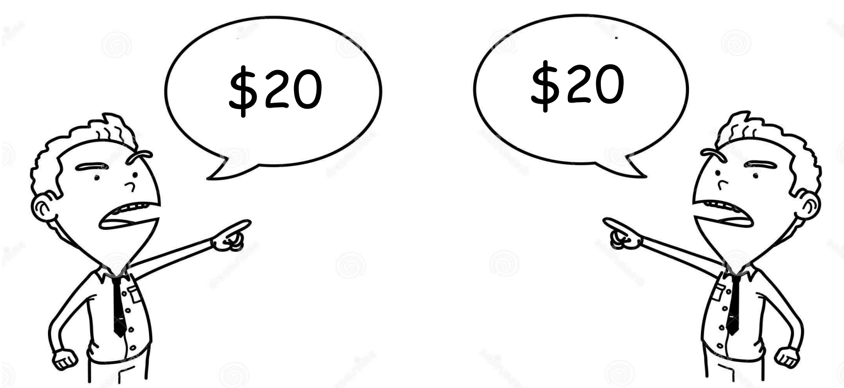
Example II: Markets on Amazon

- Assume people will buy if the book price $\leq$ \$200
- Product cost = \$20

What if the market has two book sellers…

Q: What price should each seller set?

- The market reaches a "stable status" (a.k.a., equilibrium)
- Nobody can benefit via *unilateral deviation*



- Bertrand competition
- Seller's revenue-maximizing behaviors lead to low revenue

Economic Analysis and Game Theory

Game Theory studies economic/multiple-agent decision making in scenarios where an agent's payoff depends on other agents' actions.

- Fundamental concept --- Equilibrium
  - A "stable status" at which any agent cannot improve his payoff through unilateral deviation
  - A solution concept (i.e., outcome) used to describe the system
  - Resembles "optimal decision" in single-agent case
- A central theme in game theory is to study the equilibrium
  - Different "types" of equilibria
  - May not exist; even exist, not necessarily unique
  - Understand properties of equilibrium, compute equilibria, how to improve inefficiency of equilibrium . . .

Machine Learning

- Difficult to give a universal definition
- At a high level, the task is to learn a function $f: X \rightarrow Y$, where $(\mathrm{x}, \mathrm{y}) \in X \times Y$ is drawn from some distribution D
  - Input: a set of samples $\{(x_i, y_i)\}_{i=1,2,\cdots,n}$ drawn from D
  - Output: an algorithm $A: X \rightarrow Y$ such that $A(x) \approx f(x)$ (usually measured by some loss function)
- Examples
  - Classification: $X =$ feature vectors; $Y = \{0,1\}$
  - Regression: $X =$ feature vectors; $Y = \mathbb{R}$
  - Reinforcement learning has a slightly different setup, but can be thought as $X =$ state space, Y = action space

Problems at Interface of Learning and Game Theory

- If a game is unknown or too complex, can players learn to play the game optimally?
  - Yes, sometimes – no regret learning and convergence to equilibrium
- Can game-theoretic models inspire machine learning models?
  - Yes, GANs which are zero-sum games
- Data is the fuel for ML – can we quantify economic value of data?
  - Yes, using ideas from coalitional game theory
- We know how to learn to recognize faces or languages, but can we also learn the design of games to achieve some goal?
  - Yes, learning optimal auctions, product pricing schemes, etc
- Gaming behaviors in ML? How to handle them? Societal impact?
  - Yes, e.g, learn whether to give loans to someone or whether to admit a student to Uchicago based on their features
- . ..

Goodhart's Law

When a measure becomes a target, it ceases to be a good measure

Main Topics of This Course

First Half: Machine learning for economic problems

- Basics of linear programming and game theory
- Online learning and its convergence to equilibrium

Second Half: Economic aspects of machine learning

- Economic principles for the valuation and pricing of data
- Handle gaming behaviors in machine learning
  - Particularly, algorithms, fairness, societal impacts
- The economy of online content creation and new challenges under generative AI

Main Topics of This Course

First Half: Machine learning for economic problems

- Basics of linear programming and game theory
- Online learning and its convergence to equilibrium

Second Half: Economic aspects of machine learning

- Economic principles for the valuation and pricing of data
- Handle gaming behaviors in machine learning
  - Particularly, algorithms, fairness, societal impacts
- The economy of online content creation and new challenges under generative AI

Only cover fundamentals of each direction

Main Topics of This Course

	PART I:	**Learning for Economic Problems**
1 (Jan 4: I)	Introduction [slides]	Kleinberg/Leighton paper
2 (Jan 4: II)	Basics of LPs [slides]	Chapter 2.1, 2.2, 4.3 of Convex Optimization by Boyd and Vandenberghe
3 (Jan 11: I)	LP duality [slides]	Lecture notes 5 and 6 of an optimization course by Trevisan
4 (Jan 11: II)	Intro to Game Theory (I) [slides]	Section 3.1, 3.2, 3.3 of an game theory book by Shoham and Leyton-Brown
5 (Jan 18: I)	Intro to Game Theory (II) [slides]	Equilibrium analysis of GANs by Arora et al.
6 (Jan 18: II)	Intro to Online Learning [slides]	
7 (Jan 25: I)	Multiplicative Weight [slides]	A survey paper on MWU and its applications by Arora et al.
8 (Jan 25: II)	Swap Regret [slides]	A note by Balcan on converting regret to swap regret
9 (Feb 1: I)	Multi-Armed Bandits [slides]	Section 2, 3 of the Book by Bubeck and Cesa-Bianch on Bandits
	PART II:	**Economic Aspects of Machine Learning**
10 (Feb 1: II)	Information Design [slides]	Bayesian Persuasion and Information Design paper
11 (Feb 8: I)	Valuation and Pricing of Information	Quantifying information and Optimal Pricing of Information
12 (Feb 8: II)	Shapley Value, Data Valuation	Shapley's original paper and its applications to valuating data
13 (Feb 15: I)	Strategic Learning I	PAC-learning for Strategic Classification paper
14 (Feb 15: II)	Strategic Learning II	How Can ML Induce Right Efforts paper
15 (Feb 22: I)	Tradeoffs of Fairness	Inherent Trade-Offs in the Fair Determination
16 (Feb 22: II)	Performative Prediction	Performative Prediction: Past and Future
17 (Feb 29: I)	Economics of Generative AI	Mechanism Design for Large Language Models
18 (Feb 29: II)	Project presentations	

Course Goal

- Get familiar with basics of economic principles and learning
- Understand machine learning questions in economic settings, and how to deal with some of them
- Understand the value of data, online contents, recommendation
- Aware of gaming behaviors in machine learning applications, and how to deal with some of them
- Can understand cutting-edge research papers in relevant areas

Targeted Audience of This Course

- Anyone planning to do research at the interface of economics (or algorithm design) and machine learning
  - This is a new research direction with many opportunities/challenges
  - Recent breakthrough in no-limit poker is an example

Targeted Audience of This Course

- Anyone planning to do research at the interface of economics (or algorithm design) and machine learning
  - This is a new research direction with many opportunities/challenges
  - Recent breakthrough in no-limit poker is an example
- Anyone interested in theoretical ML, strategic reasoning, human factors in learning, AI
  - As more and more ML systems interact with human beings, such strategic reasoning becomes increasingly important
  - With more techniques developed for ML, they also broadened our toolkits for designing and solving games
- Anyone interested in understanding basics of economics and learning

Who May not Be Suitable for This Course?

- Those who do not satisfy the prerequisites "in practice"
- Those who are looking for a recipe to implement ML/DL algorithms, or want to learn how to use TensorFlow, PyTorch, etc.
  - This is primarily a theory course
  - We will mostly focus on simple/basic yet theoretically insightful problems
  - The course is proof based – we will not write code

Outline

- Course Overview
- Administrivia
- An Example

Basic Information

- Course time: Thursday, 2:00 pm – 4:50 pm, with 15 mins break at the middle
- Lecture: in person (unless further instruction)
- Instructor: Haifeng Xu
  - Email: haifengxu@uchicago.edu
  - Office Hour: 4:50 to 5:50 pm Thur (rightly after class)
  - Can add more office hour, depending on demand
- TAs
  - **No TA** curently
- Couse website: www.haifeng-xu.com/cmsc35401win24/index.htm
  - Easier way is to search my personal website and navigates to course
- References: linked papers/notes on website, no official textbooks
  - Slides will be posted *after* lecture

Prerequisites

- Mathematically mature: be comfortable with proofs
- Sufficient exposures to probabilities and algorithms/optimization
  - CMSC 27200/27220 and equivalent
  - We will cover basics of optimization

Requirements and Grading

- Part I: 10% participation
- Part II: research project, 45% of grade. Project instructions will be posted on website later.
  - Team up: 2 – 4 people per team
  - Raise *novel* technical questions and provide some *nontrivial* answers
  - Deliverables: a presentation + a technical report in PDF
  - Grading is based on novelty + non-triviality

Requirements and Grading

- Part III: 3~4 homework, 45% of grade.
  - Proof based
  - Discussion allowed, even encouraged, but must write up solutions independently
  - Must be written up in Latex – hand-written solutions will not be accepted
  - One late homework allowed, at most 2 days
- Taking for electives
  - Need to additionally complete bonus questions (often more challenging) in each HW
  - HW still counts for 45%
- FYI: no need to worry about your grade if you do invest time

If you have any suggestions/comments/concerns, feel free to email me.

Outline

- Course Overview
- Administrivia
- An Example

Learning to Sell a Product

- You are a product seller facing N unknown buyers
- These buyers all value your product at the same $v \in [0,1]$, which however is *unknown* to you
- Buyers come in sequence $1,2,\cdots,N$; For each buyer, you can choose a price p and ask him whether he is willing to buy the product
  - If $v \geq p$, she/he purchases; otherwise leaves the queue

Learning to Sell a Product

- You are a product seller facing N unknown buyers
- These buyers all value your product at the same $v \in [0,1]$, which however is *unknown* to you
- Buyers come in sequence $1,2,\cdots,N$; For each buyer, you can choose a price p and ask him whether he is willing to buy the product
  - If $v \geq p$, she/he purchases; otherwise not
- How to quickly learn these buyers' value v within precision $\epsilon = \frac{1}{N}$?
  - This is a pure learning problem
  - (Well, you may directly ask a buyer's value, but guess what will happen?)
- Answer: $\log(N)$ rounds via BinarySearch

Learning to Sell a Product

- You are a product seller facing N unknown buyers
- These buyers all value your product at the same $v \in [0,1]$, which however is *unknown* to you

Let us move to a natural game-theoretic setup ……

- You have an ultimate objective of maximizing your revenue, but do not really care about learning v (though you may have to)
- How much revenue can BinarySearch secure?
  - May get really unlucky in first $\log(N)$ rounds and no sale happened
  - After $\log(N)$ rounds, can set a price $p \geq \tilde{v} - 1/N$ ($\tilde{v}$ is learned value)

$$\text{Rev} = \underbrace{0}_{\text{First } \log(N) \text{ rounds}} + \underbrace{(N - \log N)(v - \frac{2}{N})}_{\text{Remaining rounds}} \approx vN - v\log N \text{ -2}$$

Regret as Performance Measure

- To measure algorithm performance, we use regret

Regret $:=$ how much less is an algorithm's utility compared to the (idealized) case where we know v.

- Had we know v, should just price the product at $p = v$, earning vN
- The regret is then

$$\text{Regret(binary search)} \approx vN - [vN - v\log N - 2] = v\log N + 2$$

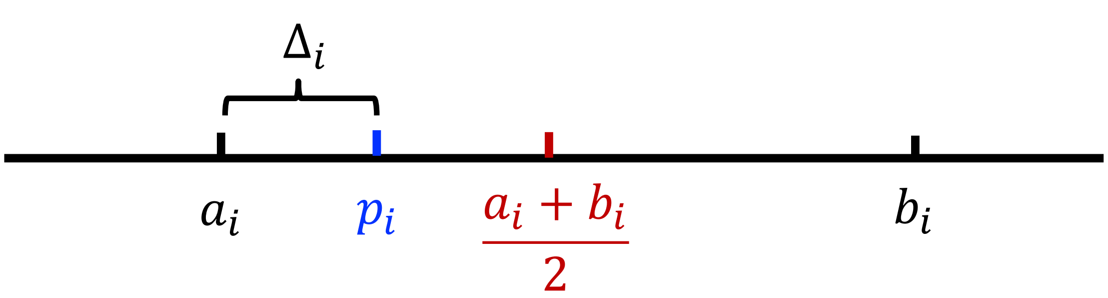
Q: Is this the best (i.e., the smallest) regret?

An Algorithm with Smaller Regret

Theorem [Kleinberg/Leighton, FOCS'03] : there is an algorithm achieving regret at most $(1 + 2 \log \log N)$

Why BinarySearch may be bad?

- For buyer i, BinarySearch maintains an interval bound $[a_i, b_i]$ and use $p_i = (a_i + b_i)/2$ for buyer i
  - This learns v as quickly as possible
  - But maybe bad for revenue since we will get 0 revenue if $p_i > v$, and $p_i = (a_i + b_i)/2$ may be too high/aggressive

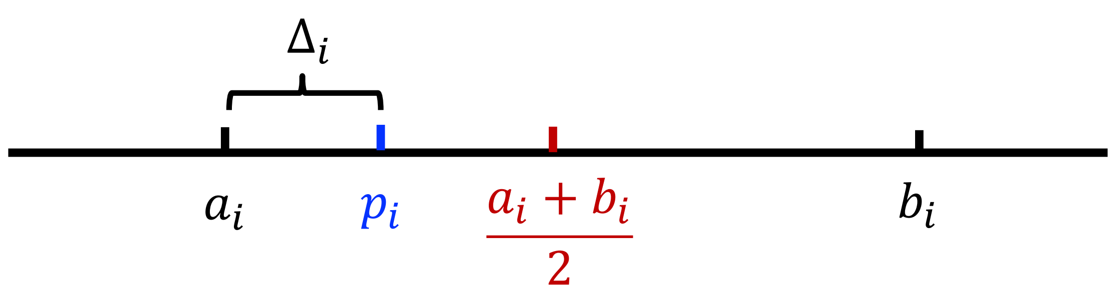


An Algorithm with Smaller Regret

Theorem [Kleinberg/Leighton, FOCS'03] : there is an algorithm achieving regret at most $(1 + 2 \log \log N)$

Why BinarySearch may be bad?

- For buyer i, BinarySearch maintains an interval bound $[a_i, b_i]$ and use $p_i = (a_i + b_i)/2$ for buyer i
  - This learns v as quickly as possible
  - But maybe bad for revenue since we will get 0 revenue if $p_i > v$, and $p_i = (a_i + b_i)/2$ may be too high/aggressive
- Algorithm idea: use more conservative prices



An Algorithm with Smaller Regret

Theorem [Kleinberg/Leighton, FOCS'03] : there is an algorithm achieving regret at most $(1 + 2\log\log N)$

The Algorithm (note $v \in [0,1]$):

- Maintains an interval bound $[a_i, b_i]$ and a step size Δ_i
- Offer price $p_i = a_i + \Delta_i$ for buyer i

- If i accepts, update $a_{i+1} = p_i$, $b_{i+1} = b_i$, $\Delta_{i+1} = \Delta_i$

An Algorithm with Smaller Regret

Theorem [Kleinberg/Leighton, FOCS'03] : there is an algorithm achieving regret at most $(1 + 2\log\log N)$

The Algorithm (note $v \in [0,1]$):

- Maintains an interval bound $[a_i, b_i]$ and a step size Δ_i
- Offer price $p_i = a_i + \Delta_i$ for buyer i

- If i accepts, update $a_{i+1} = p_i$, $b_{i+1} = b_i$, $\Delta_{i+1} = \Delta_i$

An Algorithm with Smaller Regret

Theorem [Kleinberg/Leighton, FOCS'03] : there is an algorithm achieving regret at most $(1 + 2 \log \log N)$

The Algorithm (note $v \in [0,1]$):

- Maintains an interval bound $[a_i, b_i]$ and a step size Δ_i
- Offer price $p_i = a_i + \Delta_i$ for buyer i

- If i accepts, update $a_{i+1} = p_i$, $b_{i+1} = b_i$, $\Delta_{i+1} = \Delta_i$
- Otherwise, update $a_{i+1} = a_i$, $b_{i+1} = p_i$, $\Delta_{i+1} = (\Delta_i)^2$

An Algorithm with Smaller Regret

Theorem [Kleinberg/Leighton, FOCS'03] : there is an algorithm achieving regret at most $(1 + 2 \log \log N)$

The Algorithm (note $v \in [0,1]$):

- Maintains an interval bound $[a_i, b_i]$ and a step size Δ_i
- Offer price $p_i = a_i + \Delta_i$ for buyer i

- If i accepts, update $a_{i+1} = p_i$, $b_{i+1} = b_i$, $\Delta_{i+1} = \Delta_i$
- Otherwise, update $a_{i+1} = a_i$, $b_{i+1} = p_i$, $\Delta_{i+1} = (\Delta_i)^2$

An Algorithm with Smaller Regret

Theorem [Kleinberg/Leighton, FOCS'03] : there is an algorithm achieving regret at most $(1 + 2\log\log N)$

The Algorithm (note $v \in [0,1]$):

- Maintains an interval bound $[a_i, b_i]$ and a step size Δ_i
- Offer price $p_i = a_i + \Delta_i$ for buyer i

- If i accepts, update $a_{i+1} = p_i$, $b_{i+1} = b_i$, $\Delta_{i+1} = \Delta_i$
- Otherwise, update $a_{i+1} = a_i$, $b_{i+1} = p_i$, $\Delta_{i+1} = (\Delta_i)^2$
- Start with $a_1 = 0$, $b_1 = 1$, $\Delta_1 = 1/2$; Once $b_i - a_i \le \frac{1}{N}$, always use $p = a_i$ afterwards

Remark: searching smaller region with smaller step size.

An Algorithm with Smaller Regret

Theorem [Kleinberg/Leighton, FOCS'03] : there is an algorithm achieving regret at most $(1 + 2\log\log N)$

Algorithm analysis:

Claim 1: The step size Δ_i takes values 2^{-2^j} for $j = 0, 1, \cdots$. Moreover, whenever $\Delta_{i+1} = (\Delta_i)^2$ happens, $b_{i+1} - a_{i+1} = \sqrt{\Delta_{i+1}}$.

Proof

- Recall $\Delta_1 = \frac{1}{2} = 2^{-2^0}$, and step size update $\Delta_{i+1} = (\Delta_i)^2$
- If $\Delta_i = 2^{-2^j}$, then $(\Delta_i)^2 = 2^{-2^j - 2^j} = 2^{-2^{j+1}}$
- When $\Delta_{i+1} = (\Delta_i)^2$ happens, $b_{i+1} - a_{i+1} = \Delta_i = \sqrt{\Delta_{i+1}}$

An Algorithm with Smaller Regret

Theorem [Kleinberg/Leighton, FOCS'03] : there is an algorithm achieving regret at most $(1 + 2 \log \log N)$

Algorithm analysis:

- After $b_i - a_i \leq \frac{1}{N}$, the total regret is at most 1
  - Because (1) regret of each step is at most $\frac{1}{N}$; (2) there are at most N rounds
- Main step is to bound regret before reaching $b_i - a_i = \frac{1}{N}$

An Algorithm with Smaller Regret

Theorem [Kleinberg/Leighton, FOCS'03] : there is an algorithm achieving regret at most $(1 + 2\log\log N)$

Algorithm analysis:

- How many step size value updates needed to reach $b_i - a_i = \frac{1}{N}$?
  - $\log\log N$: set $2^{-2^i} = \frac{1}{N}$ → $i = \log\log N$
  - The following claim then completes the proof of the theorem

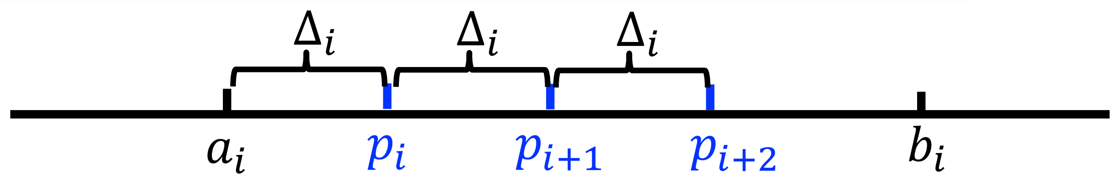
Claim 2: total regret from any step size value Δ is at most 2.

- No sale happens only once for any step size → regret at most 1

An Algorithm with Smaller Regret

Theorem [Kleinberg/Leighton, FOCS'03] : there is an algorithm achieving regret at most $(1 + 2\log\log N)$

Algorithm analysis:



➢ How many step size value updates needed to reach $b_i - a_i = \frac{1}{N}$?

- $\log\log N$: set $2^{-2^i} = \frac{1}{N}$ → $i = \log\log N$
- The following claim then completes the proof of the theorem

Claim 2: total regret from any step size value Δ is at most 2.

➢ No sale happens only once for any step size → regret at most 1

➢ What about the regret when sales happen?

- Can happen at most $\sqrt{\Delta}/\Delta$ times since $b_i - a_i \leq \sqrt{\Delta}$; regret from each time is at most $b_i - a_i (\leq \sqrt{\Delta})$
- Regret from sales is at most $(\sqrt{\Delta}/\Delta) \times \sqrt{\Delta} = 1$

An Algorithm with Smaller Regret

Remarks

- $O(\log \log N)$ is also the order-wise best regret [KL, FOCS'13]
- This is an example of exploration vs exploitation
  - Exploration: want to learn v
  - Exploitation: but ultimate goal is to utilize learned v to maximize revenue
  - More in later lectures…
- $\mathsf{BinarySearch}$ is best for exploration, but did not balance the two
- The "optimal" algorithm uses less step value updates, but more interval updates
  - Less step value updates are to be conservative about prices in order for revenue maximization
  - More interval updates mean interacting with more buyers to learn v
  - That is, **slower learning** but **higher revenue**

Well, This is Not the End Yet . . .

- Here, it is crucial that each buyer only shows up once
- What if the same buyer shows up repeatedly?
  - In fact, this is more realistic
  - E.g., in online advertising, buyer = an advertiser
- How should a (repeatedly showing up) buyer behave if he knows seller is learning her value v and then uses it to set a price for her?

Open Research Questions:

1. How to design pricing schemes for a repeatedly showing up buyer to maximize revenue when the buyer knows you are learning his value?
2. How to generalize to selling multiple products?

Thank You

Haifeng Xu

University of Chicago

haifengxu@uchicago.edu