# CMSC 35401: The Interplay of Learning and Game Theory (Autumn 2022)

# 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:

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

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

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

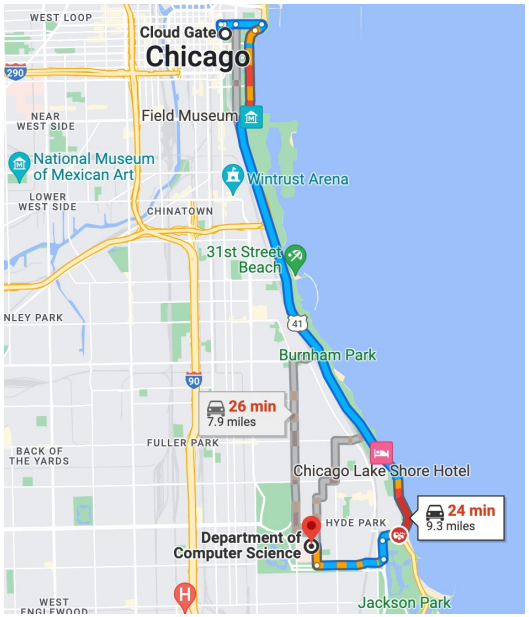
# Single-Agent Decision Making

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

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

- $x$: 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



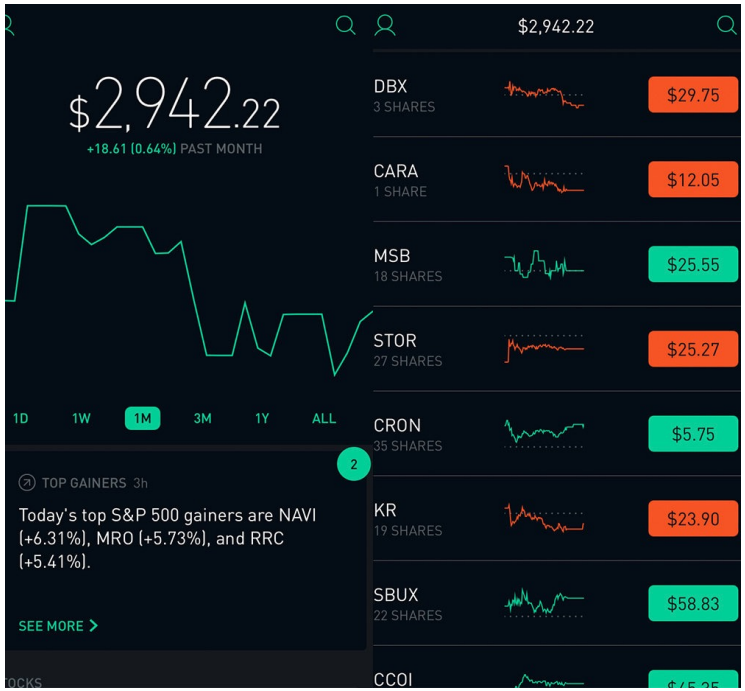
# Single-Agent Decision Making

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

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

- $x$: 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: invest a subset of stocks



# 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: stock investment, 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

| A \ B | B stays silent | B betrays |
|---|---|---|
| A stays silent | -1, -1 | 0, -3 |
| A betrays | -3, 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

| A \ B | B stays silent | B betrays |
|---|---|---|
| A stays silent | -1, -1 | 0, -3 |
| A betrays | -3, 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

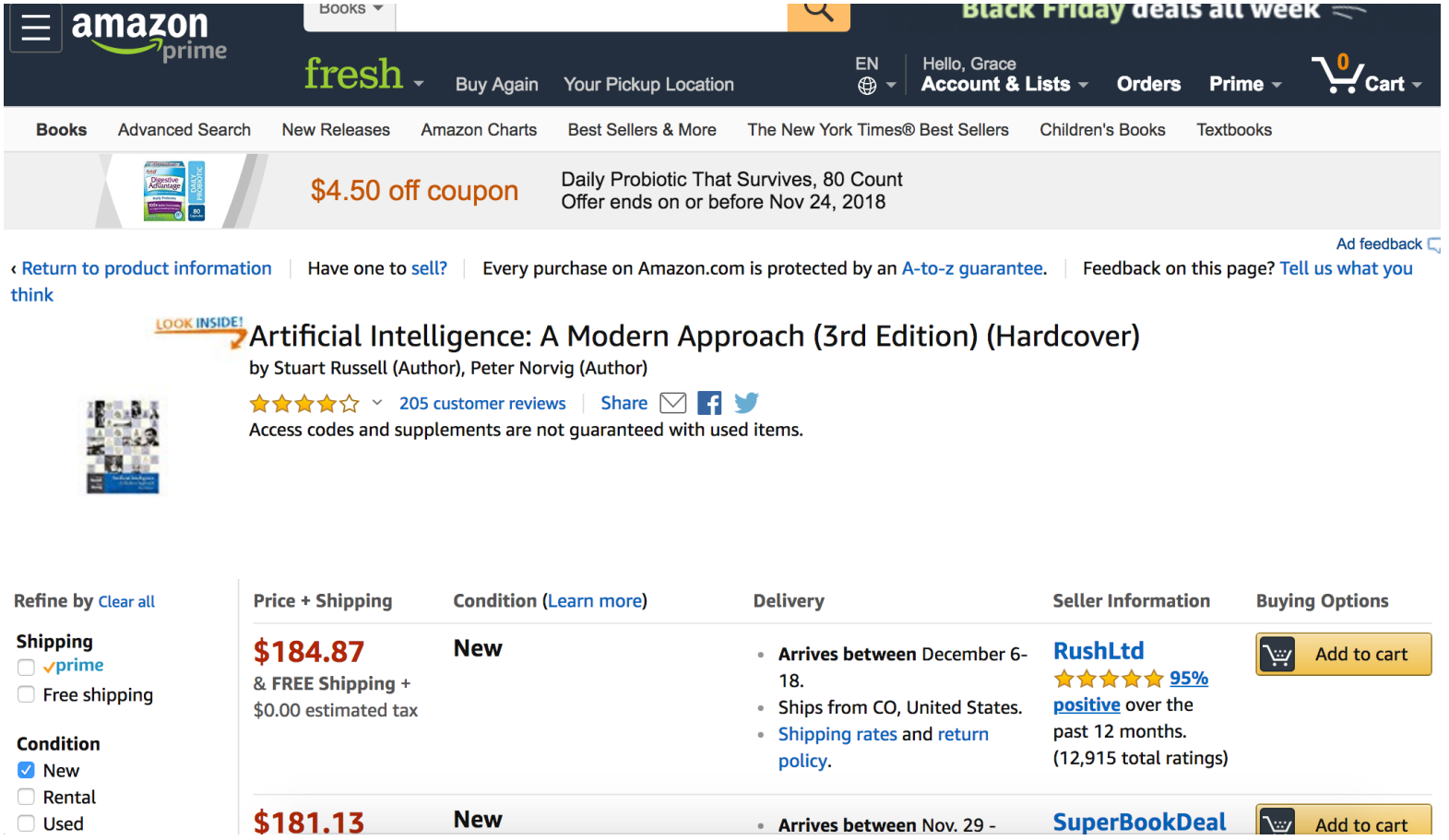
| A \ B | B stays silent | B betrays |
|---|---|---|
| A stays silent | -1, -1 | 0, -3 |
| A betrays | -3, 0 | -2, -2 |

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

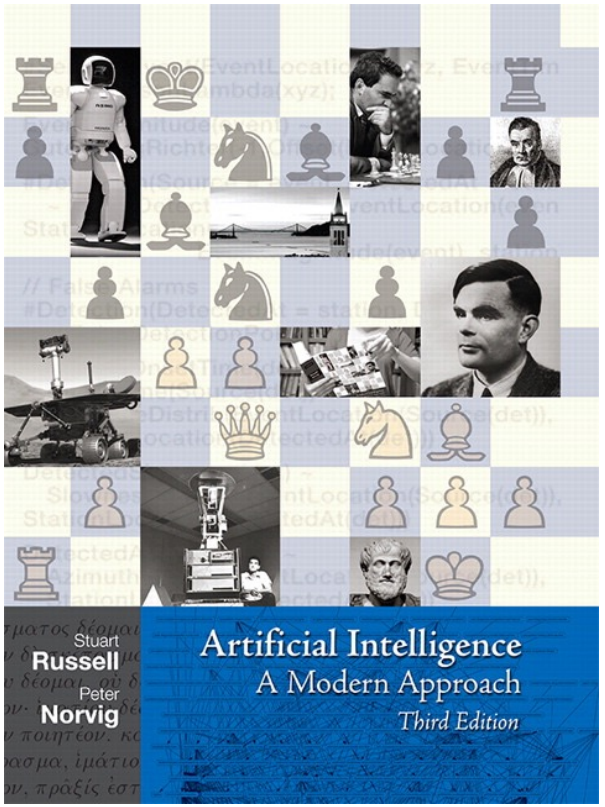


# Example II: Markets on Amazon

- 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?

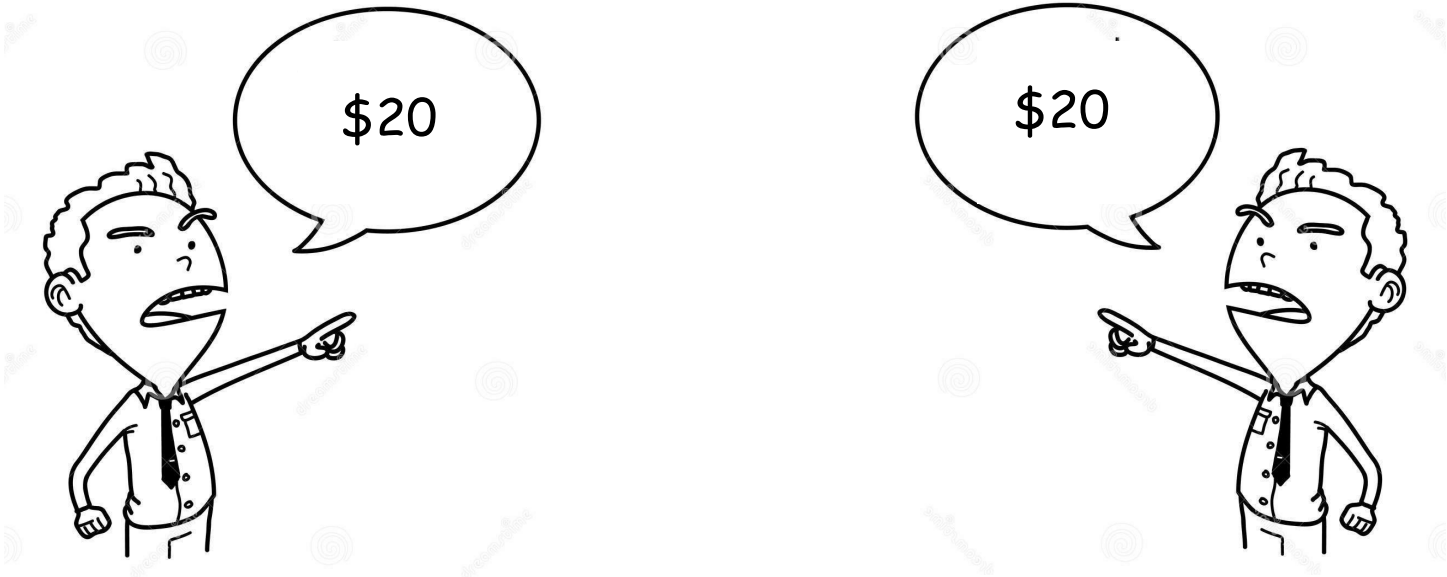



# Example II: Markets on Amazon

- 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?



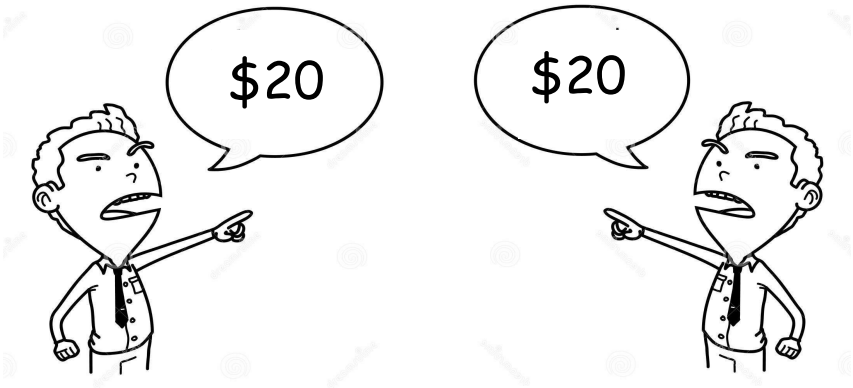
# Example II: Markets on Amazon

- 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

# Game Theory

Game Theory studies multiple-agent decision making in competitive 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
  - If exits, it is the solution concept (i.e., outcome) that we expect to happen
  - 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 $f: X \to Y$, 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 $A: X \to Y$ 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 collect high-quality data from crowd?
  - Yes, via information elicitation mechanisms
- 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/strategic behaviors in ML? How to handle them?
  - 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 game theory

- Basics of linear programming and game theory
- No regret learning and its convergence to equilibrium

Second Half: Game theory for machine learning

- Incentivize high-quality data via information elicitation (a.k.a., crowdsourcing)
- Handle gaming behaviors in machine learning
  - Particularly, learning from strategic data sources, and fairness

# Main Topics of This Course

First Half: Machine learning for game theory

- Basics of linear programming and game theory
- No regret learning and its convergence to equilibrium

Second Half: Game theory for machine learning

- Incentivize high-quality data via information elicitation (a.k.a., crowdsourcing)
- Handle gaming behaviors in machine learning
  - Particularly, learning from strategic data sources, and fairness

Only cover fundamentals of each direction

# Main Topics of This Course

| | | |
|---|---|---|
| 1 (Sep 27) | Introduction | Kleinberg/Leighton paper |
| 2 (Sep 29) | Basics of LPs | Chapter 2.1, 2.2, 4.3 of Convex Optimization by Boyd and Vandenberghe |
| 3 (Oct 4) | LP duality | Lecture notes 5 and 6 of an optimization course by Trevisan |
| 4 (Oct 6) | Intro to Game Theory (I) | Section 3.1, 3.2, 3.3 of an game theory book by Shoham and Leyton-Brown |
| 5 (Oct 11) | Intro to Game Theory (II) | Equilibrium analysis of GANs by Arora et al. |
| 6 (Oct 13) | Intro to Online Learning | |
| 7 (Oct 18) | Multiplicative Weight | A survey paper on MWU and its applications by Arora et al. |
| 8 (Oct 20) | Swap Regret | A note by Balcan on converting regret to swap regret |
| 9 (Oct 25) | Multi-Armed Bandits | Section 2 and 3 of the Book by Bubeck and Cesa-Bianch on Bandits |
| 10 (Oct 27) | Prediction Markets (PMs) | Notes from a similar course by Waggoner |
| 11 (Nov 1) | PMs and Scoring Rules | The original paper including all presented results |
| 12 (Nov 3) | Peer Prediction | Bayesian Truth Serum paper |
| 13 (Nov 8) | Bayesian Persuasion | The original Bayesian Persuasion paper |
| 14 (Nov 10) | Pricing of Information | The Selling Information Through Consulting paper |
| 15 (Nov 15) | Strategic Learning I | PAC-learning for Strategic Classification paper |
| 16 (Nov 17) | Strategic Learning II | How Can ML Induce Right Efforts paper |
| (Nov 22) | Thanksgiving, No Lecture | |
| (Nov 24) | Thanksgiving, No Lecture | |
| 17 (Nov 29) | Tradeoffs of Fariness | Inherent Trade-Offs in the Fair Determination |
| (Dec 1) | Project presentations | |

# Course Goal

- Get familiar with basics of game theory and learning
- Understand machine learning questions in game-theoretic settings, and how to deal with some of them
- Understand 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 game theory (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 game theory (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, game theory, human factors in learning, AI
  - As more and more ML systems interact with human beings, such game-theoretic 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 game theory 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: Tuesday/Thursday, 2:00 pm – 3:20 pm
- Lecture: in person (unless further university guidance is given)
- Instructor: Haifeng Xu
  - Email: haifengxu@uchicago.edu
  - Office Hour: TBD
- TAs
  - **Minbiao Han**: office hour Wed/Fri 1 – 2 pm, in person (room TBD)
- Depending on demand, can add more office hours (let us know!)
- Couse website: www.haifeng-xu.com/cmsc35401fa22/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 algorithms/optimization
  - CMSC 27200/27220 and equivalent
  - We will cover some basics of optimization

# Requirements and Grading

- 3 homeworks, 50% of grade.
  - Proof based, and will be challenging
  - 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
- Research project, 50% of grade. Project instructions will be posted on website later.
  - Team up: 2 – 4 people per team
  - Can thoroughly survey a research field, or
  - Study a relevant research question, e.g., arising from your own research
  - Presentation form: a report in PDF
- 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

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

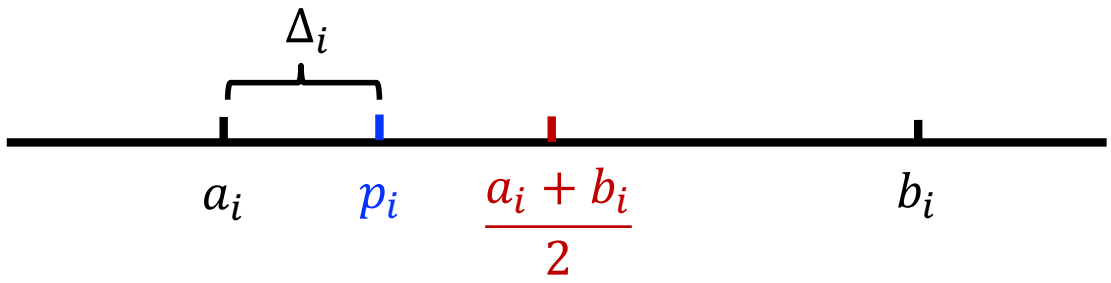
Q: Is this the best (i.e., the smallest) regret?

# 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

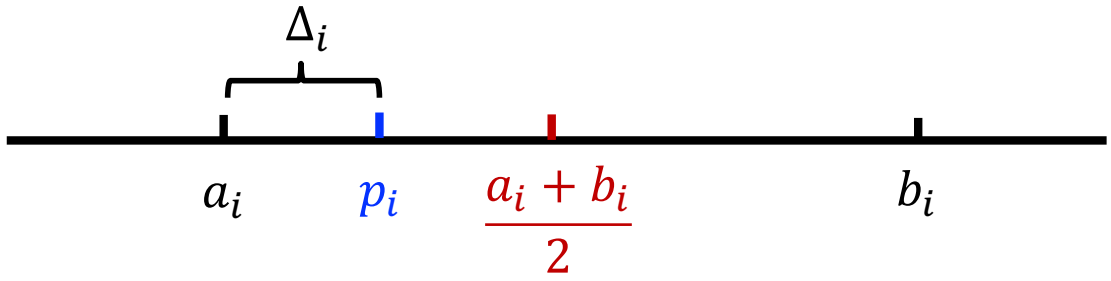


# An Algorithm with Smaller Regret

**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

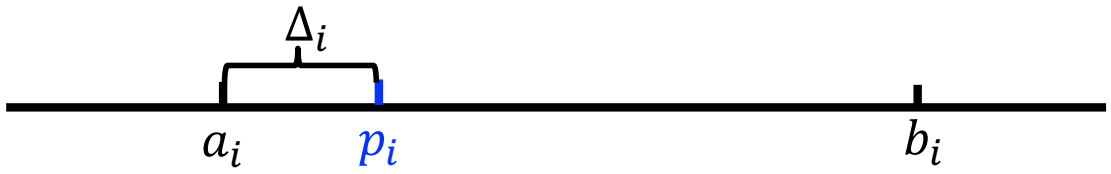


# An Algorithm with Smaller Regret

**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 $\Delta_i$
- Offer price $p_i = a_i + \Delta_i$ for buyer $i$



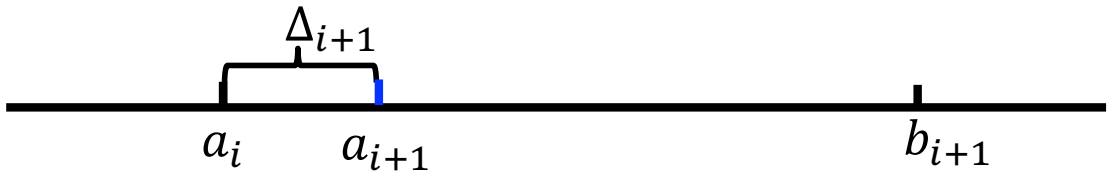

- If $i$ accepts, update $a_{i+1} = p_i$, $b_{i+1} = b_i$, $\Delta_{i+1} = \Delta_i$

# 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 $\Delta_i$
- Offer price $p_i = a_i + \Delta_i$ for buyer $i$



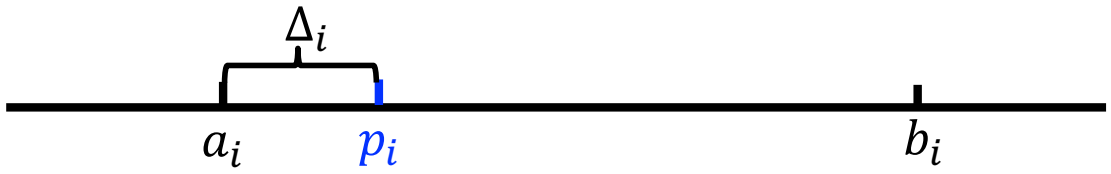

- If $i$ accepts, update $a_{i+1} = p_i$, $b_{i+1} = b_i$, $\Delta_{i+1} = \Delta_i$

# 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 $\Delta_i$
- 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 $\Delta_i$
- 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 $\Delta_i$
- 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 $\Delta_i$ 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

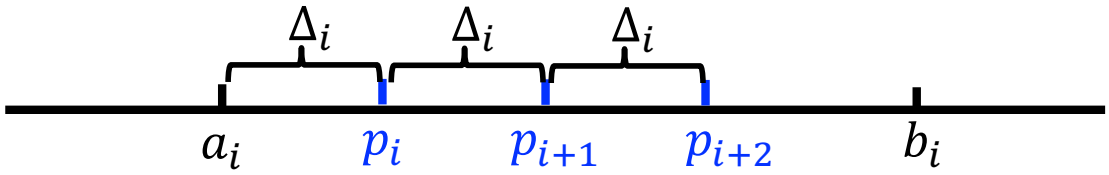
**Claim 2:** total regret from any step size value $\Delta$ is at most $2$.

- 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 $\Delta$ 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 $b_i - a_i \le \sqrt{\Delta}$; regret from each time is at most $b_i - a_i (\le \sqrt{\Delta})$
  - 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…
- 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