A Glimpse of Auction Theory (continued) + Distinct Elements

Anna Karlin
AGENDA

- Finish up glimpse of auction theory
- Distinct Elements
Auctions

- Companies like Google and Facebook make most of their money by selling ads.
- The ads are sold via auction.

**Facebook Ads bidding... 😐 Is this an auction?**

Yes! That’s the first thing you need to understand to master bidding management of Facebook Ads. When you’re creating a new campaign, you’re joining a huge, worldwide auction.

You’ll be competing with hundreds of thousands of advertisers to buy what Facebook is selling: Real estate on the News Feed, Messenger, Audience Network, and mobile apps to display your ads to the users.
An auction is a ...

- Game
  - Players: advertisers
  - Strategy choices for each player: possible bids
  - Rules of the game – made up by Google/Facebook/whoever is running the auction

- What do we expect to happen? How do we analyze mathematically?
Special case: Sealed Bid single item auction

- Say I decide to run an auction to sell my laptop and I let you be the bidders.
- If I want to make as much money as possible – what should the rules of the auction be?

Some possibilities:
- **First price auction:** highest bidder wins; pays what they bid.
- **Second price auction:** highest bidder wins; pays second highest bid.
- **All pay auction:** highest bidder wins: all bidders pay what they bid.

Which of these will make me the most money?

<table>
<thead>
<tr>
<th>Bidder</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bids</td>
<td>100</td>
<td>81</td>
<td>35</td>
<td>24</td>
</tr>
</tbody>
</table>
**Bidder Model**

Each bidder has a value, say $v_i$ for bidder $i$. 

Bidder is trying to maximize their “utility” – the value of the item they get – price they pay.

<table>
<thead>
<tr>
<th>Payment</th>
<th>1st price</th>
<th>2nd price</th>
<th>All pay</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>100</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>81</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>81</td>
<td>35</td>
</tr>
</tbody>
</table>
In a second price auction, it is always in your best interest to bid your value: 

\[ b_i = v_i \]

2nd price auction is truthful
$V_1 \sim U[0, 100]$ 

$V_2 \sim U[0, 100]$ 

Sealed bids 

$V_1$ 

$V_2$ 

2 bidders 

Winner 

Payments 

Auction 

2nd price: bid = $V$ 

1st price: bid = $\frac{V}{2}$ 

All pay auction: bid = $\frac{V^2}{200}$ 

Equal bidding strategy.
Bayes - Nash eq

1st price auction. If each bidder bids half their value, then they maximize

\[ E(\text{utility}) \]

value of other bidder

but response in expectation.

\[ V_1 \sim U[0,100] \quad V_2 \sim U[0,100] \]

\[ \text{bid } V_1, \quad \text{bid } V_2 \]

\[ \text{payment} = \min(V_1, V_2) \]

Exp auctioneer revenue in 2nd price auction

a) \[ E(\max(V_1, V_2)) \]

b) \[ E\left( \frac{V_1 + V_2}{2} \right) \]

c) \[ E(\min(V_1, V_2)) \]

d) I don't know
\begin{align*}
\text{2nd price} & \quad \text{1st price} & \quad \text{All pay each} \\
\text{bid} = v & \quad \text{bid} = \frac{v}{2} & \quad \text{bid} = \frac{v^2}{200} \\
E(\min(v_1, v_2)) & \quad E(\max(\frac{v_1}{2}, \frac{v_2}{3})) & \quad E\left[\frac{v_1^2}{200} + \frac{v_2^3}{200}\right] \\
\end{align*}

\begin{align*}
&= \frac{160}{3} \\
&= \frac{160}{3} \\
&= \frac{160}{3} \\
E[\min(v_1, v_2)] + E[\max(v_1, v_2)] = & E(v_1 + v_2)
\end{align*}
\[
E[\text{mm}(V_{11}, V_{22})] = 100 - \frac{2}{3} \cdot 100 = \frac{100}{3}
\]
In equilibrium, \( E(\text{auctioneer } \text{win}) \) is the same in all 3 auctions.

\[
E[\text{payment of each bidder same in all 3 auctions}]
\]

2nd price auction with reserve price \( r \):
- Highest bidder wins if \( b > r \) and pays \( \max(b - r, 0) \)
- Winner is the bidder with the highest bid
- Payment vector: \( (81, 75, 70) \)
- Reserve price: \( r = 70 \)
Distinct elements

Anna Karlin
With many slides by Luxi Wang, Shreya Jayaraman, Alex Tsun and Jeff Ullman
**Data Mining**

- In many data mining situations, the data is not known ahead of time.

- Examples:
  - Google queries
  - Twitter or Facebook status updates
  - Youtube video views

- In some ways, best to think of the data as an infinite stream that is non-stationary (distribution changes over time)
Stream model

- Input elements (e.g. Google queries) enter/arrive one at a time.
- We cannot possibly store the stream.

Question: How do we make critical calculations about the data stream using a limited amount of memory?
Sources of this kind of data

- **Sensor data**
  - E.g. millions of temperature sensors deployed in the ocean

- **Image data from satellites or surveillance cameras**
  - E.g. London

- **Internet and web traffic**
  - E.g. millions of streams of IP packets

- **Web data**
  - E.g. Search queries on Google, clicks on Bing, etc.
Example Applications

- **Mining query streams**
  - Google wants to know which queries are more frequent today than yesterday.

- **Mining click streams**
  - Facebook wants to know which of its ads are getting an unusual number of hits in the last hour.

- **Mining social network news feeds**
  - E.g., looking for trending topics on Twitter and Facebook, trending videos on TikTok
MORE APPLICATIONS

- Sensor networks
  - Many sensors feeding into a central controller.

- IP packets
  - Gather congestion information for optimal routing
  - Detect denial-of-service attacks
Problem

- Input: sequence of $N$ elements $x_1, x_2, ..., x_N$ from a known universe $U$ (e.g., 8-byte integers).

- Goal: perform a computation on the input, in a single left to right pass where
  - Elements processed in real time
  - Can’t store the full data. => use minimal amount of storage while maintaining working “summary”
What can we compute?

- Some functions are easy:
  - Min
  - Max
  - Sum
  - Average

32, 12, 14, 32, 7, 12, 32, 7, 32, 12, 4,
Counting distinct elements

$X_1, X_2, X_3, X_4$

32, 12, 14, 32, 7, 12, 32, 7, 32, 12, 4

Applications:

- **IP packet streams**: How many distinct IP addresses or IP flows (source+destination IP, port, protocol)
  - Anomaly detection, traffic monitoring
- **Search**: How many distinct search queries on Google on a certain topic yesterday
- **Web services**: how many distinct users (cookies) searched/browsed a certain term/item
  - Advertising, marketing trends, etc.
Another application

You are the content manager at YouTube, and you are trying to figure out the distinct view count for a video. How do we do that?

Note: A person can view their favorite videos several times, but they only count as 1 distinct view!
Counting distinct elements

32, 12, 14, 32, 7, 12, 32, 7, 32, 12, 4,

- Want to compute number of distinct keys in the stream.
- How to do this without storing all the elements?

- Yet another super cool application of probability (and hashing)
A naive solution, counting!

Store the n distinct user IDs in a hash table.

Space requirement: $O(n)$
Considering the number of users of YouTube, and the number of videos on YouTube, this is not feasible.

Consider a hash function \( h : U \rightarrow [0,1] \). For distinct values in \( U \), the function maps to iid (independent and identically distributed) \( \text{Unif}(0,1) \) random numbers.

Note that, if you were to feed in two equivalent elements, the function returns the same number.

\[
\begin{align*}
32, & \quad 12, \quad 14, \quad 32, \quad 7, \quad 12, \quad 32, \quad 7, \quad 32, \quad 12, \quad 4, \\
0.43, & \quad 0.19, \quad 0.53, \quad 0.43, \quad 0.26, \quad 0.19
\end{align*}
\]
 Min of IID Uniforms

If \( Y_1, \ldots, Y_m \) are iid \( \text{Unif}(0,1) \), where do we “expect” the points to end up?

\[
m = 1
\]

\[
0 \quad \underline{\text{x}} \quad 1
\]

\[
m = 2
\]

\[
0 \quad \underline{\text{x}} \quad \underline{\text{x}} \quad 1
\]

\[
m = 4
\]

\[
0 \quad \underline{\text{x}} \quad \underline{\text{x}} \quad \underline{\text{x}} \quad \underline{\text{x}} \quad 1
\]
I don't know