Online Algorithms

Performance-based Advertising

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University
Online Algorithms

- **Classic model of algorithms**
  - You get to see the entire input, then compute some function of it
  - In this context, “offline algorithm”

- **Online Algorithms**
  - You get to see the input one piece at a time, and need to make irrevocable decisions along the way
  - Similar to the data stream model
Online Algorithms

Bipartite Matching

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University
Example: Bipartite Matching

Nodes: Boys and Girls; Edges: Compatible Pairs

Goal: Match as many compatible pairs as possible
Example: Bipartite Matching

\[ M = \{(1,a),(2,b),(3,d)\} \text{ is a matching} \]

Cardinality of matching = \(|M| = 3\)
Example: Bipartite Matching

\[ M = \{(1,c),(2,b),(3,d),(4,a)\} \] is a perfect matching

**Perfect matching** ... all vertices of the graph are matched

**Maximum matching** ... a matching that contains the largest possible number of matches
Problem: Find a maximum matching for a given bipartite graph

- A perfect one if it exists

There is a polynomial-time offline algorithm based on augmenting paths (Hopcroft & Karp 1973, see http://en.wikipedia.org/wiki/Hopcroft-Karp_algorithm)

But what if we do not know the entire graph upfront?
Initially, we are given the set boys.

In each round, one girl’s choices are revealed.

That is, girl’s edges are revealed.

At that time, we have to decide to either:

- Pair the girl with a boy.
- Do not pair the girl with any boy.

Example of application:
Assigning tasks to servers.
Online Graph Matching: Example

```
1 ---- a
     |
  2 ---- b
     |    (1,a)
  3 ---- c
     |
  4 ---- d
     (2,b)
     (3,d)
```
Greedy Algorithm

Greedy algorithm for the online graph matching problem:

- Pair the new girl with any eligible boy
  - If there is none, do not pair girl

How good is the algorithm?
For input $I$, suppose greedy produces matching $M_{\text{greedy}}$ while an optimal matching is $M_{\text{opt}}$.

Competitive ratio = 

$$\min_{\text{all possible inputs } I} \left( \frac{|M_{\text{greedy}}|}{|M_{\text{opt}}|} \right)$$

(what is greedy’s worst performance over all possible inputs $I$)
Analyzing the Greedy Algorithm

- Suppose $M_{greedy} \neq M_{opt}$
- Consider the set $G$ of girls matched in $M_{opt}$ but not in $M_{greedy}$
- $(1) \ |M_{opt}| \leq |M_{greedy}| + |G|$

- Every boy $B$ adjacent to girls in $G$ is already matched in $M_{greedy}$
- $(2) \ |M_{greedy}| \geq |B|$
Analyzing the Greedy Algorithm

- So far:
  - $G$ matched in $M_{opt}$ but not in $M_{greedy}$
  - Boys $B$ adjacent to girls $G$
  - (1) $|M_{opt}| \leq |M_{greedy}| + |G|$
  - (2) $|M_{greedy}| \geq |B|$

- Optimal matches all the girls in $G$ to boys in $B$
  - (3) $|G| \leq |B|$

- Combining (2) and (3):
  - (4) $|G| \leq |B| \leq |M_{greedy}|$
Analyzing the Greedy Algorithm

- So we have:
  - (1) $|M_{opt}| \leq |M_{greedy}| + |G|$
  - (4) $|G| \leq |B| \leq |M_{greedy}|$

- Combining (1) and (4):
  - $|M_{opt}| \leq |M_{opt}| + |M_{greedy}|$
  - $|M_{opt}| \leq 2|M_{greedy}|$
  - $|M_{greedy}|/|M_{opt}| \geq 1/2$
Worst-case Scenario

1  
2  
3  
4  

(1, a)  
(2, b)  

Online Algorithms
Performance-based Advertising
The AdWords Problem

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University
History of Web Advertising

- **Banner ads (1995-2001)**
  - Initial form of web advertising
  - Popular websites charged $X for every 1,000 “impressions” of the ad
    - Called “**CPM**” rate (Cost per thousand impressions)
    - Modeled similar to TV, magazine ads
  - From **untargeted** to **demographically targeted**
  - **Low click-through rates**
    - Low ROI for advertisers
Performance-based Advertising

- Introduced by Overture around 2000
  - Advertisers bid on search keywords
  - When someone searches for that keyword, the highest bidder’s ad is shown
  - Advertiser is charged only if the ad is clicked on

- Similar model adopted by Google with some changes around 2002
  - Called Adwords
Performance-based advertising works!
- Multi-billion-dollar industry

What ads to show for a given query?
- (Today’s lecture)

If I am an advertiser, which search terms should I bid on and how much should I bid?
- (Not focus of today’s lecture)
A stream of queries arrives at the search engine: $q_1, q_2, \ldots$

Several advertisers bid on each query

When query $q_i$ arrives, search engine must pick a subset of advertisers whose ads are shown

**Goal:** Maximize search engine’s revenues

**Clearly we need an online algorithm!**
# Expected Revenue

<table>
<thead>
<tr>
<th>Advertiser</th>
<th>Bid</th>
<th>CTR</th>
<th>Bid * CTR</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>$1.00</td>
<td>1%</td>
<td>1 cent</td>
</tr>
<tr>
<td>B</td>
<td>$0.75</td>
<td>2%</td>
<td>1.5 cents</td>
</tr>
<tr>
<td>C</td>
<td>$0.50</td>
<td>2.5%</td>
<td>1.125 cents</td>
</tr>
</tbody>
</table>

**Click through rate**

**Expected revenue**
Instead of sorting advertisers by bid, sort by expected revenue!

<table>
<thead>
<tr>
<th>Advertiser</th>
<th>Bid</th>
<th>CTR</th>
<th>Bid * CTR</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>$0.75</td>
<td>2%</td>
<td>1.5 cents</td>
</tr>
<tr>
<td>C</td>
<td>$0.50</td>
<td>2.5%</td>
<td>1.125 cents</td>
</tr>
<tr>
<td>A</td>
<td>$1.00</td>
<td>1%</td>
<td>1 cent</td>
</tr>
</tbody>
</table>
Given:
- A set of bids by advertisers for search queries
- A click-through rate for each advertiser-query pair
- A budget for each advertiser (say for 1 day, month...)
- A limit on the number of ads to be displayed with each search query

Respond to each search query with a set of advertisers such that:
- The size of the set is no larger than the limit on the number of ads per query
- Each advertiser has bid on the search query
- Each advertiser has enough budget left to pay for the ad if it is clicked upon
### Limitations of Simple Algorithm

Instead of sorting advertisers by bid, sort by expected revenue!

<table>
<thead>
<tr>
<th>Advertiser</th>
<th>Bid</th>
<th>CTR</th>
<th>Bid * CTR</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>$0.75</td>
<td>2%</td>
<td>1.5 cents</td>
</tr>
<tr>
<td>C</td>
<td>$0.50</td>
<td>2.5%</td>
<td>1.125 cents</td>
</tr>
<tr>
<td>A</td>
<td>$1.00</td>
<td>1%</td>
<td>1 cent</td>
</tr>
</tbody>
</table>

- CTR of an ad is unknown
- Advertisers have limited budgets and bid on multiple ads (BALANCE algorithm)
Estimating CTR

- Clickthrough rate (CTR) for a query-ad pair is measured historically
  - Averaged over a time period

- Some complications we won’t cover in this lecture
  - CTR is position dependent
    - Ad #1 is clicked more than Ad #2
  - Explore v Exploit: Keep showing ads we already know the CTR of, or show new ads to estimate their CTR?
Online Algorithms

The BALANCE Algorithm

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University
Adwords Problem

- **Given:**
  - A set of bids by advertisers for search queries
  - A click-through rate for each advertiser-query pair
  - A budget for each advertiser (say for 1 day, month...)
  - A limit on the number of ads to be displayed with each search query

- **Respond to each search query with a set of advertisers such that:**
  - The size of the set is no larger than the limit on the number of ads per query
  - Each advertiser has bid on the search query
  - Each advertiser has enough budget left to pay for the ad if it is clicked upon
Our setting: Simplified environment

- There is 1 ad shown for each query
- All advertisers have the same budget $B$
- All ads are equally likely to be clicked
- Value of each ad is the same (=1)

Simplest algorithm is greedy:

- For a query pick any advertiser who has bid 1 for that query
- Competitive ratio of greedy is 1/2
Two advertisers A and B

- A bids on query x, B bids on x and y
- Both have budgets of $4

Query stream: x x x x y y y y

- Worst case greedy choice: B B B B _ _ _ _
- Optimal: A A A A B B B B
- Competitive ratio = ½

This is the worst case!

- Note: Greedy algorithm is deterministic – it always resolves draws in the same way
**BALANCE Algorithm [MSVV]**

- **BALANCE** Algorithm by Mehta, Saberi, Vazirani, and Vazirani
  - For each query, pick the advertiser with the largest unspent budget
  - Break ties arbitrarily *(but in a deterministic way)*
Two advertisers A and B
- A bids on query $x$, B bids on $x$ and $y$
- Both have budgets of $4$

Query stream: $x \ x \ x \ x \ y \ y \ y \ y$

BALANCE choice: A B A B B B _ _
- Optimal: A A A A A B B B B

Competitive ratio = $\frac{3}{4}$
- For BALANCE with 2 advertisers
Consider simple case
- 2 advertisers, $A_1$ and $A_2$, each with budget $B \geq 1$
- Optimal solution exhausts both advertisers’ budgets

BALANCE must exhaust at least one advertiser’s budget:
- If not, we can allocate more queries
- Assume BALANCE exhausts $A_2$’s budget
Analyzing Balance

**Case 1:** BALANCE assigns at least \( \frac{B}{2} \) blue queries to \( A_1 \). So \( y \geq \frac{B}{2} \).

**Case 2:** BALANCE assigns more than \( \frac{B}{2} \) blue queries to \( A_2 \).
Consider the last blue query assigned to \( A_2 \).
At that time, \( A_2 \)'s unspent budget must have been at least as big as \( A_1 \)'s.
That means at least as many queries have been assigned to \( A_1 \) as to \( A_2 \).
At this point, we have already assigned at least \( \frac{B}{2} \) queries to \( A_2 \).
So \( y \geq \frac{B}{2} \).
Analyzing BALANCE

Queries allocated to $A_1$ in the optimal solution

Queries allocated to $A_2$ in the optimal solution

Optimal revenue $\text{OPT} = 2B$

Balance revenue $\text{BAL} = B + y$

We have shown that $y \geq B/2$

$\text{BAL} \geq B + B/2 = 3B/2$

$\text{BAL}/\text{OPT} \geq 3/4$
In the general case, worst competitive ratio of BALANCE is $1 - \frac{1}{e} = \text{approx. 0.63}$

- Interestingly, no online algorithm has a better competitive ratio!

- Let's see the worst case example that gives this ratio
Worst case for BALANCE

- **N advertisers**: $A_1, A_2, \ldots, A_N$
  - Each with budget $B > N$

- **Queries**:
  - $N \cdot B$ queries appear in $N$ rounds of $B$ queries each

- **Bidding**:
  - Round 1 queries: bidders $A_1, A_2, \ldots, A_N$
  - Round 2 queries: bidders $A_2, A_3, \ldots, A_N$
  - Round $i$ queries: bidders $A_i, \ldots, A_N$

- **Optimum allocation**:
  Allocate round $i$ queries to $A_i$
  - Optimum revenue $N \cdot B$
If we find the smallest $k$ such that $S_k \geq B$, then after $k$ rounds we cannot allocate any queries to any advertiser.
**Fact:** for large $n$

- Result due to Euler

$$\frac{1}{1} \quad \frac{1}{2} \quad \frac{1}{3} \quad \ldots \quad \frac{1}{(N-(k-1))} \quad \ldots \quad \frac{1}{(N-1)} \quad \frac{1}{N}$$

$$\ln(N)$$

$$\ln(N) - 1 = \ln(N-k)$$

$$\ln(N/(N-k)) = 1$$

$$N/(N-k) = e$$

$$k = N(1 - 1/e)$$
So after the first $k = N(1-1/e)$ rounds, we cannot allocate a query to any advertiser

- Revenue = $B \cdot N \cdot (1-1/e)$
- Competitive ratio = $1-1/e$
So far: all bids = 1, all budgets equal (=B)

In a general setting BALANCE can be terrible

Consider query q, two advertisers $A_1$ and $A_2$

- $A_1$: bid = 1, budget = 110
- $A_2$: bid = 10, budget = 100

Suppose we see 10 instances of q

BALANCE always selects $A_1$ and earns 10

Optimal earns 100
Consider query $q$, bidder $i$
- Bid = $x_i$
- Budget = $b_i$
- Amount spent so far = $m_i$
- Fraction of budget left over $f_i = 1 - m_i/b_i$
- Define $\psi_i(q) = x_i(1-e^{-f_i})$

Allocate query $q$ to bidder $i$ with largest value of $\psi_i(q)$

Same competitive ratio (1-1/e)