Learning in Big Data Analytics
Lecture 3

Alexander Schönhuth

UNIVERSITÄT BIELEFELD
Faculty of Technology

Bielefeld University
December 1, 2020

Web Advertising

On-Line Advertising: Introduction

- Web applications support themselves through advertising, rather than subscriptions
  - Radio and television use ads as primary resource
  - Newspapers and magazines make use of hybrid approches
- Most lucrative venue for advertising is search
  - The *adwords* model is about matching ads with search queries
  - Algorithms are *greedy* and *online*
  - We will treat this here
- Advertising items in online stores: *collaborative filtering*
  ☞ treated in lecture *Big Data Analytics*, SS 2020

Online Advertising Opportunities

- *Direct placement* of ads for fee/commission (Craig's List; eBay; auto trading)
- Displaying ads at *fixed rate per impression* (display + download of ad)
- Online stores display ads to maximize user interest (display for free)
- Ads are placed among results in response to search query
  - Advertisers bid for right to have ad shown in response to queries
  - Pay only if ad is clicked on (impression)
  - Ads selected by complex process, involving
    - search terms
    - amount of bid
    - click-through rate of particular ad
    - total budget spent by advertiser

Direct Ad Placement

- Ads displayed in response to query terms
  - use inverted index of words in analogy to search engine itself
  - alternatively, advertiser specifies parameters to be stored in database
- Applicable ads are ranked by appropriateness
  - Beware of advertiser spam, filter ranking for ads that are too similar
- Ranking by *attractiveness* is an alternative approach. Consider:
  - Placement of ads in ranking enhances attractiveness
  - Attractiveness works relative to query terms
  - Ads whose attractiveness cannot be estimated (because of being new) deserve to be shown until attractiveness can be measured

Display Ads: Issues

- Ads should be shown to interested people
- Traditional media work with newspapers, magazines, broadcasts catering to particular interests
- The Web works with exploring individual user interests. For example:
  - Screen Facebook group membership
  - Screen emails (in gmail account) for frequently used terms
  - Time spent on sites serving particular topics
  - Screen search queries for frequently occurring terms
  - Browse through bookmark folders
- Raises (enormous!) privacy issues. Trade-off:
  - No subscription fees for various services
  - Automatically raised information can get into hands of real people

Online Algorithms and the Competitive Ratio

Online Algorithms

- Matching ads with queries are often *online algorithms*
- *Offline Algorithms:*
  - All data needed by algorithm is available initially
  - Algorithm can access data in arbitrary order
  - Algorithm produces answer accordingly
- *Online Algorithms:*
  - Not all data can be accessed before answer is required
  - *Recall data stream mining*: data appears in particular order, not all data can be stored etc.
- Selecting ads for queries easy offline:
  - E.g. consider a month full of search queries
  - *Issue:* Assign ads to queries in a most profitable way
  - Offline: assign ads to queries that maximizes both
    - search engine revenue
    - number of impressions for each advertiser
  - *But:* cannot wait for a month until displaying ad on query

Example: Online versus Offline Algorithm

- Manufacturer A_1 and A_2 both have 100 EUR budget to spend
- A_1 bids 10 cents on search term 'chesterfield'
- A_2 bids 20 cents on search terms 'chesterfield' and 'sofa'
- *Imagine*:
  - *Scenario 1:* Lots of queries for 'sofa', few for 'chesterfield'
    ☞ Need to assign 'chesterfield' to A_1
  - *Scenario 2:* Lots of search queries for 'chesterfield'
    ☞ Queries can be given to A_2; both will spend entire budget
- *Offline:* Knowing all queries beforehand allows to assign them to bids optimally
- *Online:* Mistakes are possible; overspending A_2's bids on chesterfield queries

Greedy Algorithms

- Many online algorithms are *greedy algorithms*
- Greedy algorithms decide based on actual and past input
- They maximize some appropriate function

Example: Greedy Algorithm

Consider earlier situation, involving manufacturers A_1 and A_2 and their bids on search terms ’chesterfield’ and ’sofa’.

Greedy Algorithm:

Assign each query to the highest bidder. That is,

- Assign query to A_2 if A_2 has budget left.
- Continue assigning queries to A_1 as long as A_1 has budget.
- *Result:* Assign first 500 ’chesterfield’ and ’sofa’ queries to A_2; continue to assign following 1000 ’chesterfield’ queries to A_1
- *Extreme scenario:* 500 ’chesterfield’ queries arrive followed by 500 ’sofa’ queries
  - *Offline* algorithm assigns chesterfield queries to A_1, and sofa queries to A_2
  - *Online* algorithm assigns chesterfield queries to A_2, nothing to A_1

Online Algorithms: The Competitive Ratio

- Online algorithms can only be worse than best offline algorithms
- How much worse are they? Good online algorithms differ only by little from the offline version
- Consider a particular problem, and input I
- Let $C_{\text{opt}}(I)$ be the value that one obtains when running the optimum offline algorithm
- Let $C_{\text{on}}(I)$ that one obtains when running the online algorithm under consideration

Online Algorithms: The Competitive Ratio

- Consider a particular problem, and input I
- Let $C_{\text{opt}}(I)$ be the value that one obtains when running the optimum offline algorithm
- Let $C_{\text{on}}(I)$ that one obtains when running the online algorithm under consideration

Definition [Competitive Ratio]

The *competitive ratio* of an online algorithm is (if it exists) a constant $c < 1$, such that for any input I

$$C_{\text{on}}(I) \geq c \cdot C_{\text{opt}}(I) \tag{1}$$

Online Algorithms: The Competitive Ratio

Definition [Competitive Ratio]

The *competitive ratio* of an online algorithm is (if it exists) a constant $c < 1$, such that for any input I

$$C_{\text{on}}(I) \geq c \cdot C_{\text{opt}}(I)$$

Explanation: For an online algorithm with competitive ratio c, the value of the objective function is at least c times the optimal value one can achieve using an offline algorithm.

Example: Competitive Ratio I

Consider earlier situation, involving manufacturers A_1 and A_2 and their bids on search terms 'chesterfield' and 'sofa'.

- *Extreme scenario:* 500 'chesterfield' queries arrive followed by 500 'sofa' queries
- *Offline* algorithm assigns chesterfield to A_1, and sofa to A_2
  ☞ Revenue: 150 EUR
- *Online* algorithm assigns chesterfield to A_2, nothing to A_1
  ☞ Revenue: 100 EUR
- So, on this instance, $C_{\text{on}}(I) = \frac{2}{3} \cdot C_{\text{opt}}(I)$
- That means that for the competitive ratio c, we have

$$c \leq \frac{2}{3}$$

Example: Competitive Ratio II

Consider earlier situation, involving manufacturers A_1 and A_2 and their bids on search terms 'chesterfield' and 'sofa'.

- *Extreme scenario:* 500 'chesterfield' queries arrive followed by 500 'sofa' queries
- Consider to raise A_1's bid to $20 - \epsilon$ cents per bid, then:
  - *Offline* algorithm assigns chesterfield to A_1, and sofa to A_2
    ☞ Revenue now: $200 - 500 \cdot \epsilon \overset{\epsilon \to 0}{\longrightarrow} 200$ EUR
  - *Online* algorithm assigns chesterfield to A_2, nothing to A_1, because still A_2's bid is greater than A_1's
    ☞ Revenue still: 100 EUR
- On this instance, c approaches $\frac{1}{2}$
- One can indeed show that

$$c = \frac{1}{2}$$

The Matching Problem

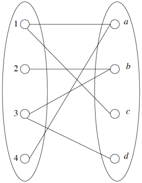
MATCHES AND PERFECT MATCHES

DEFINITION [BIPARTITE GRAPHS]

A bipartite graph $G = (V, E)$ with vertices V and edges E is referred to as *bipartite* iff

- there are $V_1, V_2 \subset V$ such that

$$V = V_1 \mathbin{\dot\cup} V_2 \quad \text{and} \quad E \subset (V_1 \times V_2)$$



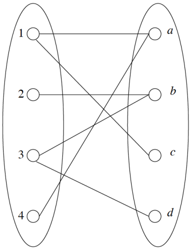
Bipartite graph with $E \subset \{1, 2, 3, 4\} \times \{a, b, c, d\}$

Adopted from `mmds.org`

MATCHES AND PERFECT MATCHES

DEFINITION [MATCHINGS]

- A *matching* $M \subset E$ is a set of edges such that for each vertex $v \in V$ there is at most one $e \in M$ in which v appears
- A *perfect matching* is a matching that covers every node
- A matching is *maximal* iff any other matching is at most as large



Adopted from `mmds.org`

- $(1, a), (2, b), (3, d)$ is a matching, but not a perfect matching
- $(1, c), (2, b), (3, d), (4, a)$ is a perfect matching
- $(1, c), (2, b), (3, d), (4, a)$ is also maximal
- *Note:* every perfect matching is also maximal

GREEDY ALGORITHM FOR MAXIMAL MATCHING

- *Offline algorithms* for maximal matchings have been studied for decades
- The algorithms run in nearly $O(n^2)$ time for graphs on n vertices
- Here, we consider online algorithms (also well studied)
- Greedy algorithm for maximal matching:
  - Consider edges in any order
  - Add edge to matching iff both ends are not yet covered by any edge collected so far
- *Example:*
  - Consider vertices from example before in order $(1, a), (1, c), (2, b), (3, b), (3, d), (4, a)$
  - This yields non-maximal matching $(1, a), (2, b), (3, d)$
  - Any order starting with $(1, a), (3, b)$ implies matching of size 2

Competitive Ratio for Greedy Matching

- In the example, we had optimal matching of size 4 and greedy matching of size 2
- That implies that $\frac{1}{2}$ is an upper bound for the competitive ratio for Greedy matching
- We would like to prove that $\frac{1}{2}$ is the competitive ratio

Competitive Ratio for Greedy Matching

Notation

- Let M_o be a maximal matching
- Let M_g be the matching computed by the Greedy algorithm
- Let L be the left nodes matched in M_o, but not in M_g
- Let R be the right nodes connected by edges to any vertex in L

Claim: Every vertex from R is matched in M_g.

Proof: Suppose that $r \in R$ is not matched in M_g. At some point, the greedy algorithm considers (l, r) with $l \in L$. At that point, however, neither $l \in L$ nor $r \in R$ were encountered by the Greedy algorithm. So (l, r) will be included in the matching, a contradiction! □

Conclusion: Every node from R is matched in M_g.

UNIVERSITÄT BIELEFELD

Competitive Ratio for Greedy Matching

Notation/Facts

- Let M_o be a maximal matching
- Let M_g be the matching computed by the Greedy algorithm
- Let L be the left nodes matched in M_o, but not in M_g
- Let R be the right nodes connected by edges to any vertex in L
- We proved that every node from R is matched in M_g

- In M_o, all nodes in L are matched with nodes from R, implies

$$|L| \leq |R| \tag{2}$$

- Every node in R is matched in M_g, implies

$$|R| \leq |M_g| \tag{3}$$

- Together, this yields

$$|L| \leq |M_g| \tag{4}$$

Competitive Ratio for Greedy Matching

- From before, we have
$$|L| \leq |M_g| \tag{5}$$
- Only nodes in L can be matched in M_o, but not in M_g, implies
$$|M_o| \leq |M_g| + |L| \tag{6}$$
- (5) and (6) together imply
$$|M_o| \leq 2|M_g| \quad \text{or} \quad |M_g| \geq \frac{1}{2}|M_o| \tag{7}$$

That means that the competitive ratio c is at least $\frac{1}{2}$, so with the above example, that
$$c = \frac{1}{2}$$

General / Further Reading

Literature

- Mining Massive Datasets, Sections 8.1, 8.2, 8.3: `http://infolab.stanford.edu/~ullman/mmds/ch8.pdf`