Advanced Topics in Machine Learning and Algorithmic Game Theory

Lecture 9: Sponsored Search Optimization

Lecturer: Yishay Mansour *Scribe: Eyal Gofer*

9.1 Lecture Overview

This lecture discusses the problem of allocating ad slots given search queries, where the advertisers have constrained budgets, and the search queries arrive online. An online algorithm is tasked with allocating each term to an advertiser, with the aim of maximizing the overall revenue, subject to the budget constraints. The main result described is an online algorithm that, for a random permutation of the ads, achieves a profit of $(1-\epsilon)OPT$ with high probability, where OPT is the profit of the optimal offline algorithm. The value of ϵ is an input to the algorithm and depends on problem parameters.

9.2 Online Adword Allocation

Suppose a user types the query term "flowers" in a search engine. In addition to search result, the search engine returns related ads that are placed at the top and on the side of the results page. In the background, various advertisers bid for the right to show their ads in response to specific search terms, and these advertisers are ranked according to their bids. A bid is the amount of money an advertiser is willing to pay whenever a user clicks on an ad. Note that an advertiser may sell several different products. In addition, each advertiser has a budget constraint, for example, up to \$100 to be spent in total on a given day.

The advertiser that wins such an auction pays the second highest price (a second price auction). This type of auction has the desirable property of being strategy proof, namely, that advertisers have no incentive to state a price other than their true valuation. Real-life auctions on the internet are more complex, but the general principal is the same.

Formally, there are n advertisers, m queries, and B_i is the budget of advertiser i, for $1 \leq i \leq n$. The queries arrive in an online fashion, and for each query j, $1 \leq j \leq m$, we get as input the advertisers' bids for the query, denoted $b_{1,j}, \ldots, b_{n,j}$. It is assumed for simplicity that $b_{i,j} \in [0,1]$ for every i and j. The online algorithm is required to assign each query to an advertiser. For query j, this choice is represented by a vector $x_{1,j}, \ldots, x_{n,j}$, s.t. $x_{i,j} = 1$ if i is the chosen advertiser and 0 otherwise. The profit of the algorithm is then $ALG = \sum_{i=1}^{n} \min\{B_i, \sum_{j=1}^{m} b_{i,j}x_{i,j}\}$. We will assume that $b_{\max} = \max_{i,j}\{b_{i,j}\}$ satisfies $b_{\max} \ll B_i$ for every $1 \leq i \leq n$. In other words, we assume that the importance of any single ad allocation is negligible.

The source of the queries may be modeled in various ways. We will consider allocation algorithms for three different source models.

9.3 A Fully Adversarial Model

We begin with the case where the frequency and order of the terms is arbitrary.

We will make the simplifying assumption that $b_{i,j} \in \{0,1\}$. Denote $spent(i,j) = \sum_{t=1}^{j-1} x_{i,t} b_{i,t}$ for the amount spent by advertiser i up until round j. Consider the algorithm GREEDY that, given the bid vector $b_{*,j}$ for the j-th query, chooses the highest possible bid, that is the advertiser in $\arg\max_i \{b_{i,j} : spent(i,j) < B_i\}$.

We will bound the competitive ratio of GREEDY. Let $y_{i,j}$ be the allocations by an optimal offline algorithm OPT. The revenue of any algorithm is the sum of the revenues from each of the advertisers. In order for OPT to do better than GREEDY, it has to do better on some of the advertisers. There are some advertisers whose budget is completely exhausted under GREEDY's allocations. For those advertisers, OPT clearly cannot do better than GREEDY. We then consider an advertiser i whose budget is not all used under GREEDY's allocations. Any excess revenue OPT gets from i must come from times j s.t. $y_{i,j} = 1$ and $x_{i,j} = 0$. Since i did not use up its budget, GREEDY could have chosen i, but instead allocated query j to some advertiser $k \neq i$. By definition of GREEDY, $b_{i,j} \leq b_{k,j}$. Therefore, any advantage gained by OPT over GREEDY may be mapped to some greater or equal revenue obtained by GREEDY. Thus, $OPT - GREEDY \leq GREEDY$, which means that GREEDY is 2-competitive.

Note that with minor changes, the same greedy algorithm and analysis also work if $b_{i,j} \in [0,1]$, and the algorithm is 2-competitive.

9.4 The Other Extreme: A Stochastic Model

We assume that there is a finite set L of possible bid vectors $b_{*,l}$, each with its own known probability q_l. On each round j, a bid vector is chosen independently at random from the set, according to the above probability. We solve the following linear program:

$$\begin{array}{lll} \text{maximize} & & \sum_{i,l} b_{i,l} x_{i,l} q_l m \\ \text{subject to} & \forall i & \sum_l b_{i,l} x_{i,l} q_l m \leq B_i \\ & \forall l & \sum_i x_{i,l} \leq 1 \\ & \forall i,l & x_{i,l} \geq 0. \end{array}$$

Each of the variables $x_{i,l}$ has the interpretation of being the probability of allocating the ad to advertiser i, given that we get the bid vector $b_{*,l}$. The program maximizes the expected

revenue. This leaves the possibility that for some samples, advertisers will outspend their budgets. We show how this may be avoided with high probability.

If we use the above solution, what is the probability that an advertiser i overspends its budget?

Let $\hat{q}_l$ be the observed frequency of query $l \in L$. For each query l we select the advertiser using the probability distribution $x_{*,l}$. Let $\hat{x}_{i,l}$ be the frequency of times we selected advertiser i for query l. By Hoeffding's inequality,

$$\Pr\left(\exists l:\ |q_l - \hat{q}_l| > \epsilon\right) \leq |L|e^{-2\epsilon^2 m}, \quad \text{and} \quad \Pr\left(\exists l, i:\ |x_{i,l} - \hat{x}_{i,l}| > \epsilon\right) \leq n|L|e^{-2\epsilon^2 m}$$

This implies that the probability we exceed the budget is small, i.e.,

$$\Pr\left(\exists i:\ \sum_{i,l} \hat{q}_l \hat{x}_{i,l} b_{i,l} m \geq (1+\epsilon)B_i\right) \ \leq \ 2n|L|e^{-2(\epsilon/3)^2 m},$$

since $(1+\epsilon/3)^2 < 1+\epsilon$. Equivalently, if $m = \Omega(\frac{1}{\epsilon^2}\ln\frac{n|L|}{\delta})$, that probability is upper bounded by δ. This gives a regret of at most $\sum_{i=1}^{n} \epsilon B_i \leq \epsilon \cdot OPT$.

9.5 An Intermediary Model: Random Permutations

We now introduce a model in which an adversary is allowed to choose the bid vectors, but their order is chosen uniformly at random. This model seems more realistic than the previous stochastic setting. The main idea will be to "sacrifice" the first ϵm bids, learn an optimal rule, and then use it on the remaining $(1-\epsilon)m$ bids. For this purpose, we will assume that m is known in advance.

It is important to note that knowing m in advance is necessary for achieving a competitive ratio that tends to 1. To see why this is true, consider a scenario with two bidders and two query types, a and b. Each bidder has a budget of 150, and the bids are $(1,0)$ for a and $(2,1)$ for b. We receive a stream of 50 a's, then 50 b's, and then again 50 a's and 50 b's. If the stream stops after the first 100 queries, OPT allocates all queries to bidder 1, obtaining a revenue of $OPT_1 = 150$. If we process all the queries, OPT allocates 25 b's and all a's to bidder 1, and 75 b's to bidder 2, a total revenue of $OPT_2 = 225$. Let ALG be some online algorithm. Denote a_1 for the number of a's allocated to bidder 1 out of the first 100 queries, b_1 for the number of b's allocated to bidder 1 out of the first 100 queries, and similarly, a_2 and b_2 for the allocations to bidder 1 out of the last 100 queries. $R_1 = b_1 + a_1 + 50$ is the revenue of ALG from the first 100 queries. Therefore, $R_2 = b_1 + a_1 + b_2 + a_2 + 100$ is the revenue from all the queries. The revenue from bidder 1 on all queries is $2b_1 + 2b_2 + a_1 + a_2$, and this is ≤ 150, by its budget constraints. Thus, $R_1 + R_2 = (2b_1 + 2b_2 + a_1 + a_2) + (150 + a_1 - b_2) \leq 300 + a_1 - b_2 \leq 350$. Observe that

$$\min\{R_1/OPT_1, R_2/OPT_2\} \ \leq \ \frac{R_1 + R_2}{OPT_1 + OPT_2} \leq 350/375 \sim 0.93,$$

so ALG is suboptimal either on the first 100 queries or on all of them.

We now proceed to explain the idea behind the algorithm. We consider the following linear program:

$$\begin{array}{lllrcl} \text{maximize} & & \sum_{i,j} b_{i,j} x_{i,j} & & & \\ \text{subject to} & \forall i & \sum_j b_{i,j} x_{i,j} & \leq & B_i \\ & \forall j & \sum_i x_{i,j} & \leq & 1 \\ & \forall i,j & x_{i,j} & \geq & 0. \end{array}$$

Its dual is the program

$$\begin{array}{lllrcl} \text{minimize} & & \sum_i \alpha_i B_i + \sum_j p_j & & & \\ \text{subject to} & \forall j & \alpha_j & \geq & 0 \\ & \forall j & p_j & \geq & 0 \\ & \forall i,j & p_j & \geq & b_{i,j}(1-\alpha_i). \end{array}$$

Consider the optimal dual solution. At the optimum, $\alpha_i \leq 1$ for every i, because given the constraints of the form $p_j \geq 0$, there is no advantage in taking $\alpha_i > 1$. Thus, for every j, $p_j = \max_i\{b_{i,j}(1-\alpha_i)\}$. The dual objective therefore has the form $D(\alpha) = \sum_i \alpha_i B_i + \sum_j \max_i\{b_{i,j}(1-\alpha_i)\}$ at the optimum. By complementary slackness, if $x_{i,j} > 0$, then $p_j = b_{i,j}(1-\alpha_i)$, and by all the other constraints on p_j, we get $p_j = \max_i\{b_{i,j}(1-\alpha_i)\}$. Given the optimal α, we should therefore allocate j to the bidder $\arg\max_i\{b_{i,j}(1-\alpha_i)\}$. In fact, every $\alpha \in [0,1]^n$ defines a rule that assigns j to $\arg\max_i\{b_{i,j}(1-\alpha_i)\}$.

The algorithm will "learn" the optimal α on the first ϵm queries, and then apply the allocation rule described above to the rest of the queries. We use the notation $D(\alpha, S) = \sum_{i=1}^n \alpha_i \epsilon B_i + \sum_{j \in S} \max_i\{b_{i,j}(1-\alpha_i)\}$, where $S \subseteq \{1,\ldots,n\}$, $|S| = \epsilon m$.

Algorithm 1 Learn-Weights(ϵ)

for $j = 1 \rightarrow \epsilon m$ **do**
  Observe the bids $b_{i,j}$
  Allocate arbitrarily (e.g. GREEDY)
end for
$\alpha^* \leftarrow \arg\min_\alpha\{D(\alpha, S)\}$
for $j = \epsilon m + 1 \rightarrow m$ **do**
  Observe the bids $b_{i,j}$
  Allocate to $\arg\max_i\{b_{i,j}(1-\alpha_i^*)\}$
end for

Note that the algorithm makes the implicit assumption that the expression $b_{i,j}(1-\alpha_i^*)$ has a unique maximum. We continue under this assumption and comment that the problem of ties may be resolved by the introduction of small random perturbations that bring the probability of ties to 0.

Theorem 9.1 *The algorithm Learn-Weights(ϵ) has a $1-O(\epsilon)$ competitive ratio if for every $1 \leq i \leq n$, $\frac{n \ln(mn)}{\epsilon^3} \leq B_i$.*

To prove Theorem 9.1, we use ideas based on the PAC learning model. The PAC model setting consists of a sample space X, a concept class $\mathcal{C}$, a target function $f : X \to \{0,1\}$, and a distribution D over X.

Definition A class $\mathcal{C}$ is PAC-learnable with a sample of size k if for every $f \in \mathcal{C}$, we can find a hypothesis $h \in \mathcal{C}$ s.t. $\Pr_D\{error(h) < \epsilon\} > 1-\delta$, where $error(h) = D(f \neq h)$.

Theorem 9.2 *A finite class $\mathcal{C}$ is PAC-learnable with a sample of size $k = O(\frac{1}{\epsilon} \ln \frac{|\mathcal{C}|}{\delta})$, using a consistent hypothesis.*

Proof: Assume $error(h) > \epsilon$. The probability that h is consistent is $< (1-\epsilon)^k$. By the union bound,

$$\Pr(\exists h : h \text{ is consistent}, error(h) > \epsilon) \quad \leq \quad |\mathcal{C}|(1-\epsilon)^k \leq \delta.$$

□

For our problem, $\mathcal{C} = [0,1]^n$, h is our α, and $f = \alpha^*$. Every α is an allocation of the queries to the advertisers $\{1, \ldots, n\}$, and $error(\alpha) = 1 - \frac{ALG(\alpha)}{OPT}$. We will show that if $error(\alpha) > \epsilon$, then $\Pr(\alpha \text{ is optimal on } S) \leq \delta/|\Pi| = \delta'$, where S is the query learning set and Π is the set of possible allocations by some $\alpha' \in \mathcal{C}$.

Let $x_{i,j}(\alpha)$ be the allocations using α. For every $1 \leq i \leq n$, denote $R_i(\alpha) = \sum_j b_{i,j} x_{i,j}(\alpha)$ and

$$d_i(\alpha) \quad = \quad \begin{cases} |R_i(\alpha) - B_i| & \alpha_i > 0 \\ \max\{0, R_i(\alpha) - B_i\} & \alpha_i = 0. \end{cases}$$

Denote also $D_i(\alpha) = \alpha_i B_i + (1-\alpha_i) R_i(\alpha)$. We need some auxiliary claims before we can prove the main result.

Claim 9.3 *It holds that $D(\alpha) - ALG(\alpha) \leq \sum_i d_i(\alpha)$.*

Proof: We have that

$$\begin{aligned} D(\alpha) &= \sum_i \alpha_i B_i + \sum_j \max_i \{b_{i,j}(1-\alpha_i)\} \\ &= \sum_i \alpha_i B_i + \sum_j \sum_i (1-\alpha_i) b_{i,j} x_{i,j}(\alpha) \\ &= \sum_i \alpha_i B_i + \sum_i (1-\alpha_i) \sum_j b_{i,j} x_{i,j}(\alpha) \\ &= \sum_i \alpha_i B_i + \sum_i (1-\alpha_i) R_i(\alpha). \end{aligned}$$

In addition, $ALG(\alpha) = \sum_i \min\{B_i, R_i(\alpha)\}$, and together we have that

$$D(\alpha) - ALG(\alpha) \;=\; \sum_i \left(\alpha_i B_i + (1-\alpha_i)R_i(\alpha) - \min\{B_i, R_i(\alpha)\}\right).$$

Consider the expression $\alpha_i B_i + (1-\alpha_i)R_i(\alpha) - \min\{B_i, R_i(\alpha)\}$. If $\alpha_i > 0$, then in case $B_i \le R_i(\alpha)$, we get $(1-\alpha_i)(R_i - B_i)$, and if $B_i > R_i(\alpha)$, we get $\alpha_i(B_i - R_i)$. In both cases the result is $\le d_i(\alpha)$. If $\alpha_i = 0$, then we get $R_i(\alpha) - \min\{B_i, R_i(\alpha)\}$. If $B_i \le R_i(\alpha)$, then we get $R_i(\alpha) - B_i$, and if $B_i > R_i(\alpha)$, we get 0. In any case, the result is $\le d_i(\alpha)$. □

Recall that we want to show that if $error(\alpha) > \epsilon$, then $\Pr(\alpha \text{ is optimal on S}) \le \delta/|\Pi| = \delta'$. To do that, we show:

1. If $error(\alpha) > \epsilon$, then $\exists i\ d_i(\alpha) > \epsilon D_i(\alpha)$.
2. If $\exists i\ d_i(\alpha) > \epsilon D_i(\alpha)$, then $\Pr(\alpha \text{ is optimal on } S) \le \delta'$.

Suppose that for every i, $d_i(\alpha) < \epsilon D_i(\alpha)$. By Claim 9.3, we would have that

$$D(\alpha) - ALG(\alpha) \le \epsilon \sum_i D_i(\alpha).$$

We have already seen that

$$D(\alpha) = \sum_i \alpha_i B_i + \sum_i (1-\alpha_i)R_i(\alpha) = \sum_i D_i(\alpha),$$

so $D(\alpha) - ALG(\alpha) \le \epsilon D(\alpha)$, and therefore, $error(\alpha) \le \epsilon$, proving (1).

We now proceed to show (2). The following claim bounds the size of the concept class (proof omitted).

Claim 9.4 *It holds that* $|\Pi| = (mn)^{O(n)}$.

We will use the following variant of Bernstein's inequality:

Lemma 9.5 *Let* $b_1, \ldots, b_n \in [0,1]$, *and let* $S \subseteq \{1, \ldots, n\}$ *be a random subset of size* k. *Define* $X = \sum_{j \in S} b_j$ *and let* $\mu = \mathbb{E}[X] = (k/m)\sum_i b_i$. *Then with probability* $> 1 - \delta$, $|X - \mu| = O(\sqrt{\mu \ln(1/\delta)} + \ln(1/\delta))$.

Denote $b_j = b_{i,j} x_{i,j}(\alpha)$ (and suppose that $\alpha_i > 0$). By complementary slackness, $X = \epsilon B_i$ and $\mu = \epsilon R_i(\alpha)$, and therefore, $|X - \mu| = \epsilon d_i(\alpha)$. Since $d_i(\alpha) > \epsilon D_i(\alpha)$, we have that $\epsilon d_i(\alpha) > \epsilon^2 D_i(\alpha) = \epsilon^2 z$, where $z = \alpha_i B_i + (1-\alpha_i)R_i(\alpha)$, and $B_i \simeq R_i(\alpha) = z$. We also have that $\epsilon^2 z \ge \sqrt{\epsilon z \ln(1/\delta')} + \ln(1/\delta')$ for $B \simeq z \ge (1/\epsilon^3)\ln(1/\delta') = \frac{\ln|\Pi| + \ln(1/\delta)}{\epsilon^2}$. We know that for $O(\sqrt{\epsilon R/z \ln(1/\delta')} + \ln(1/\delta'))$ the probability is δ'. Therefore, with probability at most δ', the vector α is optimal if $\epsilon D_i(\alpha) < d_i(\alpha)$.