Estimation of the Click Volume by Large Scale Regression Analysis

Yury Lifshits[1] and Dirk Nowotka[2]

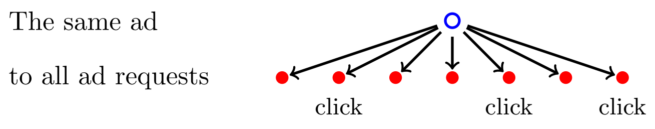
[1] Steklov Institute of Mathematics St.Petersburg, Russia, yura@logic.pdmi.ras.ru
[2] FMI, Universität Stuttgart, Germany, nowotka@fmi.uni-stuttgart.de

Abstract. How could one estimate the total number of clicks a new advertisement could potentially receive in the current market? This question, called the *click volume estimation problem* is investigated in this paper. This constitutes a new research direction for advertising engines. We propose a model of computing an estimation of the click volume. A key component of our solution is the application of linear regression to a large (but sparse) data set. We propose an iterative method in order to achieve a fast approximation of the solution. We prove that our algorithm always converges to optimal parameters of linear regression. To the best of our knowledge, it is the first time when linear regression is considered in such a large scale context.

1 Introduction

In general, an *advertising engine* (AE) (1) maintains a database of advertisements, (2) receives ad requests "some person is accessing some media", and (3) returns several ads that are most relevant to this request. Google AdWords, Yahoo! Search Marketing, and Microsoft adCenterare the most prominent advertising engines for sponsored search. Google AdSenseis an example of an AE for contextual advertisements. Finally, the Amazon.com recommendation system [8] is a particular case for an e-commerce recommendations AE. We expect that specialized advertising engines will be introduced very soon for blogspace, social networks, computer games and virtual reality, and even supermarket bills.

In this paper we start a new research direction for advertising engines. Consider the following question: How could one estimate the total number of clicks a new advertisement can potentially receive in the current market? We call this the **click volume estimation problem** and use $CV(a)$ to denote the click volume of an advertisement a. Knowledge about advertisements (the *ad space*), requests for ads (the *request space*), and *historical information* can be used to calculate an estimation of $CV(a)$. The click volume estimation problem has not yet been investigated in the literature to the best of our knowledge.



There are plenty of reasons to be interested in the click volume of an ad. Let's consider some of them.

- Maintainers of advertising engines might wish to understand how many clicks they can (approximately) sell for any given advertisement. Click volume information can be useful for setting optimal prices.
- The click volume can measure the current effectiveness of advertising engines.
- Advertising engines might use different strategies for the cases when the click volume is smaller than the demand from advertisers and when it is larger. In the latter case an AE can *skip* some ad request even if advertisers mark it as belonging to their target group.
- The real goal of an AE is to recognize the whole interested audience for the given ad and to display it *only* to these people. Hence, estimating the volume of that audience is the first step towards the recognition problem.
- Using click volume estimation, an advertiser can predict the necessary resources needed to cover a given fraction of market.
- Comparison of overall click volume and click volume restricted to the target subspace of ad requests provides a kind of "target coverage" value. It can help advertisers to understand whether their target description is good enough.
- Assuming we have a purchase history table. Then applying similar techniques we can estimate the purchase market for a new product.

Results. This paper is the first step towards setting a formal definition of click volume and constructing efficient algorithms for computing reliable CV estimations. Our main contributions are (1) a general model of an advertising engine and its history table, (2) a methodology for calculating the click volume using linear regression, (3) a fast iterative algorithm for solving the linear regression problem on large and sparse data sets based on [11, 18], (4) a complexity bound for one round of iterations, and (5) a proof of convergence for the algorithm. Finally, we pose a series of open problems and suggest directions for further research in Section 4.

Let us describe our solution for the click volume problem in an informal way. We take a history table and transform it into the list of pairs: event vector, empirical value of click-through rate. Here, an event vector characterizes a displayed ad, an ad request and their relationship. It belongs to a high dimensional euclidian space, but only few of its components are nonzero. We make an assumption that the click-through rate (more precisely, *logit* of click-through rate) can be calculated as a scalar product between the event vector and some unknown vector α. Then we have to find the α that minimizes the prediction error over the whole history table. This is a well-known linear regression problem. Unfortunately, classical methods, like direct method and SVD-based method [15] are infeasible in our settings. As far as we know, all previous algorithms have time complexity $\Omega(mn)$ where mn is the size of underlying matrix. In order to get better complexity, we have to use the sparseness of the underlying matrix.

In essence, we need just to compute the projection of one vector to the linear hull of some family of *sparse* vectors. While solving systems of linear equations and computing eigenvectors are well studied in sparse settings [13], no particular method was suggested for the *projection* problem. In this paper we propose an iterative algorithm based on componentwise descent method [18]. In [18] only one component of the current approximation is updated in every turn. Inspired by the sequential minimal optimizations method [11] used for training support vector machines, we make a modification to componentwise descent method. That is, we calculate the optimal shift value *analytically.* We show that shift computation is linear just to the number of *nonzero* components of the corresponding term vector.

In this paper we just start the investigation of solving the large scale regression problem on sparse data. But we are convinced that this method can find much more applications in web computing than just click volume estimation.

Related research. The algorithm for predicting the click-through rate in [12] was the main inspiration source for our research. However, there are some important differences. In [12] the click-through rate (hence, click volume) is estimated for all ads shown for a given search term (ad request in our terminology). We present a solution for the dual problem: fixing an ad and assuming that it is displayed on all ad requests. Moreover, we estimate the click through rate for any pair of ad and ad request. In our case the problem of insufficient history can not be solved *solely* by the assumption that two content-similar ads have a similar click volume. Indeed, a collection "ad-to-ad-request" is far from the completeness for *every* ad and even for every ad cluster. Next, we present a model which captures various industrial solutions at the same time, while [12] addresses only sponsored search technology.

Let us compare our regression based solution with other possible approaches. For every newcomer event vector one can locate the nearest events from our history table and use their click-through rates as the basis of estimation. However, the nearest neighbors algorithms either are two slow [6], or use assumptions that clearly do not hold [17] for our case. Alternatively, one can, for example, treat ads as people, ad requests as books and empirical click-through rates as ratings. Than we can apply the well developed theory of collaborative filtering [10]. Unfortunately, one needs at least few appearances of the newcomer ad in the history table to apply this method. Also, collaborative filtering does not use term similarity between ads and ad requests.

We refer to the papers [4, 16] for a general introduction to sponsored search. Algorithms for optimal ad choosing are constructed in [9]. Auction design for sponsored search is well presented in [3, 5]. Many open problems around advertising engines were posed on the SSA'06 panel discussion [1]. Finally, various versions of similarity for search terms (i.e. ad requests in our terminology) are investigated in the paper [2].

2 Basic Model: Advertising Engine and History Table

The general setting discussed in this paper is an advertising engine which, given some historical information about placing advertisements and a new advertisement from an advertiser, has to estimate the number of clicks (click volume) this new ad would get over a certain period of time. Let's define our notation.

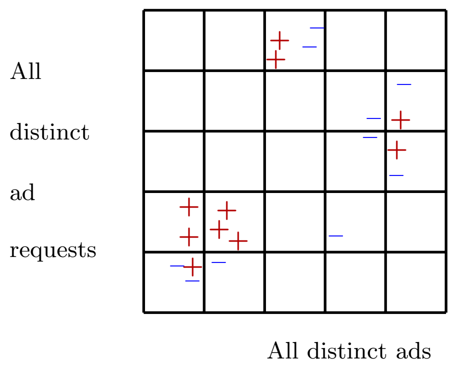
Let A be the set of all *advertisements*. By convention we denote elements of A by a and its subscripted versions. Let $a_{new} \in A$ denote the advertisement whose click volume will be estimated. In principle an element of A might be defined by several properties like the content of the advertisement (for example text with a link and a phone number), a target audience description, some keywords describing the ad, and so on. However, all we assume here is that an ad is represented by a vector of reals. For example, an entry of a might denote the membership of the ad in some property where 1 means that a has that property and 0 means that a does not have that property.

Furthermore, we need a notion of ad request. The AE is contacted every time a person is contacting some media (in most cases websites). Let R be the set of all *ad requests* whose elements are denoted by r and its subscripted versions in the following. Requests, like advertisements, could be described by a number of parameters like a person (or rather IP address), media (for example a search web page), and an action (for example an entered search query). But again, all we require is r being a vector of reals, like advertisements.

Advertisements and requests together form an exposition *event*. An event should be represented by features of the advertisement (for example, keywords, language, target group), features of the ad request (for example, location, search phrase), and the relation between ad and ad request (for example, whether or not the language of the ad is spoken at the location the request came from). Let us assume that an event is represented as a vector such that each component denotes whether or not that event possesses that feature. Let E denote the set of all events. Given an ad a and a request r, let $e(a, r)$ denote the event representing the ad, the ad request, and the relation between them.

The data we use for our estimation of the click volume is called *history table* over $E \times \{0, 1\}$ where $(e(a, r), b)$ denotes an advertisement a shown on ad request r which received a click if $b = 1$ and no click if $b = 0$. Let us fix an arbitrary history table HT with n entries for the rest of this paper. We have

$$HT = (e(a_1, r_1), b_1), (e(a_2, r_2), b_2), \ldots, (e(a_n, r_n), b_n) \ .$$



We can visualize the history table as a set of $+$ and $-$ symbols in the table whose columns correspond to ads and rows correspond to ad request. Actually every cell represents the set of all possible unique exposition events that are observed at the same pair (a, r). Our history table is a collection of "click-occurrence" values on some points in the table. We want to stress here that there is an important difference between columns and rows. Namely, in our history ad requests are taken randomly from standard daily (weekly, annual) distribution. But ads were chosen by the advertising engine and they are by no means random. Typically, for every row our history covers only few cells. Therefore, we can assume that only a linear (more precisely, proportional to the number of all requests) number of cells is covered by the history.

The *click volume* $CV(a)$ is the total number of clicks over a fixed period of time (for example one week) that we would expect for a to get if shown at *all* ad requests taken from the same distribution as in HT. The click-through rate is a standard term for internet advertising. We recall it in the setting of our model. Given an event e the *click-through rate* (CTR) $CTR(e)$ is the ratio between the number of clicks for e and the total number of times e occurs (impressions), or in other words, the probability that a is clicked on in conjunction with the ad request r when $e = e(a, r)$.

3 Reduction to the Regression Problem

The history table HT records for each event e either a click or no click. What we really want to know, though, is the probability of an event being clicked, that is, we need the actual CTR of e. The CTR of an event can in principle be calculated from the number of times e occurs in HT and the number of times when e received a click. However, this simple approach does not work well, since most of the vectors in HT are unique.

The history table HT is assumed to be our given input. Although it is a natural way to present the input data, it is not suitable for calculating the click volume of a new ad quickly. Therefore, HT is pre-processed. The first step in the pre-processing phase is a *dimensionality reduction* of HT. There exist a number of different approaches to reduce the dimensionality of E; see [14] for a survey. We suggest dimensionality reduction by some term extraction method, for example *term clustering*. Latent semantic indexing is another method for term extraction. However, we do not suggest latent semantic indexing here since it will turn out that sparseness of the event vector set is a desired property for applying the linear regression method to our problem as suggested below; see the next section in particular. Let us assume that DR denotes the dimensionality reduction function of our choice, that is, $DR(e)$ denotes the event derived from e by reducing its dimensionality.

The function DR is now used to transform the history table HT into a *reduced history table* RHT. Firstly, we replace every entry (e, b) of the history table HT by $(DR(e), b)$, and secondly, all entries (e', b_1), ..., (e', b_k) of the same reduced event e' are combined to one entry (e', b') where b' denotes the click through rate $(\sum_{1 \leq i \leq k} b_i)/k$ for e'. Let the set of all reduced events in RHT be denoted by a matrix T and the set of all click through rates by a vector β, that is $RHT = T\beta$.

Given an event e_{new}, the problem of estimating the click through rate of e_{new} can be formulated as the problem of fitting the function that relates reduced events to their click through rate. The *linear regression* analysis is a standard method for curve fitting where a linear function is used to describe the relation between two variables; in our case, these are reduced events and click through rate.

However, it is not guarantied that the function resulting from the linear regression analysis yields a value between 0 and 1 for e_{new}. To avoid this problem, we apply a one-to-one mapping from $[0, 1]$ to $[-\infty, +\infty]$ to all click through rate values in RHT. Such a mapping is for example logit where $\text{logit}(p) = \log(p/(1-p))$. Let now $\gamma = \text{logit}(\beta)$, and we perform a linear regression analysis on $T\gamma$. The regression analysis yields a vector α such that $\|T\alpha - \gamma\|$ is minimal, that is, α approximates the relation between reduced events and the (logit of the) click through rate such that the sum of the quadratic error of all entries is minimal. The click through rate of e_{new} is now estimated by $\text{logit}^{-1}(\alpha \cdot DR(e_{new}))$, that is (1) the dimensionality of e_{new} is reduced, (2) this reduced event is combined with α, and (3) the resulting value is mapped into the interval $[0, 1]$ by logit^{-1}.

We would like to know the sum of all clicks a new ad a_{new} should get when exposed to all requests of the history (for a given time period). That is, a query consists of an ad a_{new} which needs to be combined with all requests in HT. We get the following formula.

$$CV(a_{new}) = \sum_{1 \leq i \leq n} \text{logit}^{-1}(\alpha \cdot DR(e(a_{new}, r_i)))$$

All these steps are summarized in the following methodology.

Methodology for Click Volume Estimation

Pre-processing:

1. Dimensionality reduction of E, e.g. by term clustering
2. Calculation of $RHT = T\gamma$
3. Approximate calculation of α such that $\|(\alpha T) - \gamma\|$ is minimal (see next section)

Query: Calculate $CV(a_{new}) = \sum_{1 \leq i \leq n} \text{logit}^{-1}(\alpha \cdot DR(e(a_{new}, r_i)))$

4 Solving the Large Scale Regression Problem

The methodology of estimating the click volume of a_{new} of the previous section employs the linear regression analysis for calculating $CV(a_{new})$. Consider the history table entries of reduced dimensionality, let $e = DR(e(a_{new}, r_i))$. The direct calculation of parameters of linear regression requires inverting a huge and non-sparse matrix. Since we assume to work with large data sets and under strict time constraints this method is not available to us. Instead, we suggest an alternative method to estimate α which is iteratively approximating the optimal solution and which also has good convergence properties for sparse settings as the ones we consider for click volume. Intuitively, α is learned by using the reduced history table as a training set.

The formal setting. We consider the history table of reduced dimensionality *RHT* where we have a logit value for click through rate assigned to every event entry e. Let m denote the dimension of e, and let T denote the $n \times m$ matrix of entries in *RHT*, and let γ be an n vector where γ_i denotes the logit value of the click through rate the i-th entry in *RHT*. Let t_j denote the j-th column vector (of dimension n) of T for all $1 \leq j \leq m$. The goal is to find the α such that $\|T\alpha - \gamma\|$ is minimal. Certainly, $\alpha = (T \cdot T^*)^{-1} T^* \gamma$. However, calculating α directly this way is too expensive for our purpose. Therefore we describe an iterative method in the next paragraph which generates a sequence $\alpha^{(1)}$, $\alpha^{(2)}$, ... converging towards α.

The iterative algorithm. Let $\alpha^{(1)}$ be the zero vector. Assume $\alpha^{(k)}$ is known and we would like to calculate $\alpha^{(k+1)}$. First, one component j of $\alpha^{(k)}$ is chosen. Then $\alpha^{(k+1)}$ is such that $\alpha_i^{k+1} = \alpha_i^{(k)}$ for all $i \neq j$ and

$$\alpha_j^{(k+1)} = \alpha_j^{(k)} + \frac{(\alpha^{(k)} T - \gamma) \cdot t_j}{\|t_j\|^2} \ .$$

Intuitively, we minimize the error of the "prediction" of all CTR values in our history table by $\alpha^{(k)}$ by adjusting only the j-th component $\alpha^{(k)}$ while all other components are left unchanged.

A geometric interpretation. Let us reformulate our algorithm in terms of *discrepancy* vector φ:

$$\varphi^{(k)} = \gamma - \alpha^{(k)} T$$

$$\varphi^{(0)} = \gamma, \quad \varphi^{(k+1)} = \varphi^{(k)} - \frac{\varphi^k \cdot t_j}{\|t_j\|^2} t_j$$

Consider the linear hull *span*(T) of T. We would like to come as close to γ as possible, while all our estimations belong to *span*(T). Hence, our optimal estimation is the orthogonal projection of γ to *span*(T), let us denote it by φ. A minimal error achieved by a vector that is equal to *span*(T)-orthogonal component of γ. In every step of our algorithm we do the following. Take our current estimation $\varphi^{(k)} \in$ *span*(T), draw a line parallel to t_j and going through $\varphi^{(k)}$, then project γ (or, with the same result, φ) to this line. This projection point is our new estimation $\varphi^{(k+1)}$.

Algorithm analysis.

Theorem 1 (Convergence theorem). *Consider euclidian space, vector $\varphi^{(0)}$, family of vectors $T = \{t_1, \ldots, t_m\}$, and fixed infinite order of updates J : $j_1, j_2, \ldots$ that contains every index infinitely many times. Let us construct the sequence $\varphi^{(k)}$ by the following rule:*

$$\varphi^{(k+1)} = \varphi^{(k)} - \frac{\varphi^{(k)} \cdot t_{j_k}}{\|t_{j_k}\|^2} t_{j_k}$$

Then the sequence $(\varphi^{(k)})$ converges to φ, where φ is the span(T)-orthogonal component of $\varphi^{(0)}$.

Proof. Note that $\varphi^{(k+1)} \perp t_{j_k}$. Indeed,

$$\varphi^{(k+1)} \cdot t_{j_k} = \varphi^k \cdot t_{j_k} - \frac{\varphi^{(k)} \cdot t_{j_k}}{\|t_{j_k}\|^2} t_{j_k} \cdot t_{j_k} = 0.$$

Since $\varphi^{(k)} = \varphi^{(k+1)} + \frac{\varphi^{(k)} \cdot t_{j_k}}{\|t_{j_k}\|^2} t_{j_k}$ and $\varphi^{(k+1)} \perp t_{j_k}$, we have $\|\varphi^{(k)}\| \geq \|\varphi^{(k+1)}\|$.

Assume now that $(\varphi^{(k)})$ does not converge to φ. Since $\|\varphi^{(k)}\|$ is bounded, we can choose a subsequence $(\varphi^{(k_i)})$ converging to some $\psi \neq \varphi$. Let us divide all T family to those vectors that are orthogonal to ψ (subfamily T_1) and those that are not orthogonal (subfamily T_2). Then T_2 is nonempty, otherwise ψ coincides with φ.

Let $c = \min_{j \in T_2} \frac{\psi \cdot t_j}{\|t_j\|}$. Since there exists a subsequence of $(\varphi^{(k)})$ converging to ψ, there are infinitely many members, that belong to the $c/2$ neighborhood of ψ. Consider one such visit $\varphi^{(k)}$. Let us look at the next updates. If we use $j_k \in T_1$ we can come only closer to ψ. Indeed, as was shown above, $\varphi^{(k+1)} \perp t_{j_k}$, moreover $\psi \perp t_{j_k}$. Since

$$\varphi^{(k)} - \psi = (\varphi^{(k+1)} - \psi) + \frac{\varphi^{(k)} \cdot t_{j_k}}{\|t_{j_k}\|^2} t_{j_k}$$

and the first and second terms are orthogonal, $\|\varphi^{(k)} - \psi\| \geq \|\varphi^{(k+1)} - \psi\|$.

Since every index occurs infinitely often we will finally apply an update for some $j_k \in T_2$, still being in the $c/2$ neighborhood of ψ. Let us estimate the "size" of this step.

$$\frac{\varphi^{(k)} \cdot t_{j_k}}{\|t_{j_k}\|^2} t_{j_k} = \frac{\psi \cdot t_{j_k} + (\varphi^{(k)} - \psi) \cdot t_{j_k}}{\|t_{j_k}\|^2} t_{j_k} \geq \frac{c\|t_{j_k}\| - \frac{c}{2}\|t_{j_k}\|}{\|t_{j_k}\|^2} t_{j_k} \geq \frac{c}{2\|t_{j_k}\|} t_{j_k}.$$

Since the shift was orthogonal to $\varphi^{(k+1)}$, we have

$$\|\varphi^{(k+1)}\|^2 \leq \|\varphi^{(k)}\|^2 - \frac{c^2}{4}.$$

Let us summarize our observations. Sequence $(\varphi^{(k)})$ visits $c/2$ neighborhood of ψ infinitely often, once coming inside it cannot escape by T_1 updates, while T_2 update leads to a substantial decrease of $\|\varphi^{(k)}\|$. We get a contradiction with the facts that $\|\varphi^{(k)}\|$ is nonnegative and monotonically decreasing. □

Lemma 1 (Round complexity). *It is possible to make a single update for every j (one round of updates) in time linearly depending from the number of nonzero elements in the history table.*

Proof. At the beginning, we precompute norms of all t_j. We will maintain both current regression vector $\alpha^{(k)}$ and current discrepancy vector $\varphi^{(k)} = \gamma - \alpha^{(k)} T$. We start from $\alpha^{(k)} = 0$ and $\varphi^{(0)} = \gamma$. Consider some j and let q_j be the number of nonzero components in t_j. At first we update $\alpha_j^{(k)}$ in $\mathcal{O}(q_j)$ time. Indeed, we need to calculate the scalar product between t_j and $\varphi^{(k)}$ and this can be done by few corresponding look-ups. Then we update the discrepancy vector by the rule $\varphi^{(k+1)} = \varphi^{(k)} - \frac{\varphi^{(k)} \cdot t_{j_k}}{\|t_{j_k}\|^2} t_{j_k}$. Again, scalar product can be computed in $\mathcal{O}(q_j)$ time and only q_j components of $\varphi^{(k)}$ should be modified. Summing over all j we get the required round complexity. □

Some other complexity remarks:

1. A vector $\alpha^{(k)}$ can be safely updated in two components j_1 and j_2 *in parallel* if we have $t_{j_1} \cap t_{j_2} = \emptyset$. Again the sparseness assumption would allow for a high degree of parallelism in practice.
2. Note that in the case of orthogonal column vectors t_i α is reached by updating every component exactly once.
3. Also the joint update of two components i and j is not expensive. In order to calculate

$$\left(\alpha_i^{(k+1)} \alpha_j^{(k+1)}\right)^* = \left(\alpha_i^{(k)} \alpha_j^{(k)}\right)^* - \left((t_i t_j) \cdot (t_i t_j)^*\right)^{-1} (t_i t_j)^* (\alpha^{(k)} T - \gamma)$$

we need to invert $(t_i t_j) \cdot (t_i t_j)^*$ which is only a 2×2 matrix.

5 Further Work

In this paper we state the click volume problem, show how one can reformulate it as the large scale regression problem and propose an iterative algorithm for solving the latter. It is the first time this has been done and many questions follow from that. How do we estimate the convergence speed of our algorithm? How should we choose the next component for update? Assume that we allow the output "cannot predict CTR for this event". Can we improve the accuracy by solving the problem with this relaxation? Can we combine the regression approach with other methods, e.g. clustering?

Experimental validation. Of course, one wants to verify our regression based approach in industrial applications (e.g. Google AdWords system). Forthcoming experiments should answer the following questions. What is the convergence speed of our iterative algorithm in practice? What is the overall error of linear regression estimation on the history table? What event vector representation and dimensionality reduction routines lead to the most accurate CTR prediction? Finally, it is interesting to apply the large scale linear regression algorithm to other problems, e.g. predicting news article popularity on `digg.com`.

Related problems for on-line advertisements. The dual problem for click volume estimation is the *ad volume estimation.* Namely, to estimate the total amount of advertisements that could get a positive response on the given ad request. *Click volume for the whole market*: What is the fastest way to estimate the click volume for every advertisement in the system? More precisely, can we do it faster than doing a separate click volume estimation for every ad?

Finally, the click volume problem is just a single member of our list of web-related algorithmic problems. In [7], one can find more theoretical questions in web computing.

Acknowledgments. The authors would like to thank Javier Esparza, Stefan Göller, Benjamin Hoffmann, Stefan Kiefer, Mikhail Lifshits, Markus Lohrey, Hinrich Schütze, Kirill Shmakov, Jason Utt and anonymous referees of the previous version of this paper for their useful comments and fruitful discussions.

References

1. Models for sponsored search: What are the right questions? Panel Discussion at SSA'06, 2006.
2. K. Bartz, V. Murthi, and S. Sebastian. Logistic regression and collaborative filtering for sponsored search term recommendation. In *SSA'06*, 2006.
3. B. Edelman, M. Ostrovsky, and M. Schwarz. Internet advertising and the generalized second price auction: Selling billions of dollars worth of keywords. In *SSA'06*, 2006.
4. D. Fain and J. Pedersen. Sponsored search: a brief history. In *SSA'06*, 2006.
5. J. Feng, H. Bhargava, and D. Pennock. Implementing sponsored search in web search engines: Computational evaluation of alternative mechanisms. *Informs Journal on Computing*, 2006.
6. J. M. Kleinberg. Two algorithms for nearest-neighbor search in high dimensions. In *STOC '97*, pages 599–608, 1997.
7. Y. Lifshits. A Guide to Web Research. Materials of mini-course at Stuttgart University. Available at `http://logic.pdmi.ras.ru/~yura/webguide.html`, 2007.
8. G. Linden, B. Smith, and J. York. Amazon.com recommendations item-to-item collaborative filtering. *Internet Computing*, 2003.
9. A. Mehta, A. Saberi, U. Vazirani, and V. Vazirani. AdWords and generalized on-line matching. In *FOCS'05*. IEEE, 2005.
10. M. O'Connor and J. Herlocker. Clustering items for collaborative filtering. SIGIR '01, Workshop on Recommender Systems., 2001.
11. J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in kernel methods: support vector learning*, pages 185–208, Cambridge, MA, USA, 1999. MIT Press.
12. M. Regelson and D. Fain. Predicting clickthrough rate using keyword clusters. In *SSA'06*, 2006.
13. Y. Saad. *Iterative methods for sparse linear systems (2nd edition)*. SIAM, 2003.
14. F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.
15. G. A. F. Seber and A. J. Lee. *Linear Regression Analysis*. Wiley, 2003.
16. A. Tuzhilin. The Lane's Gifts v. Google report, 2006.
17. P. N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *SODA '93*, pages 311–321, 1993.
18. V. Zhdanov. The componentwise descent method. *Mathematical Notes*, 22(1):566–569, 1977.