Estimation of the Click Volume by Large Scale Regression Analysis

Yuri Lifshits, Dirk Nowotka [International Computer Science Symposium in Russia '07, Ekaterinburg]

Presented by: Aditya Menon

UCSD

May 15, 2008

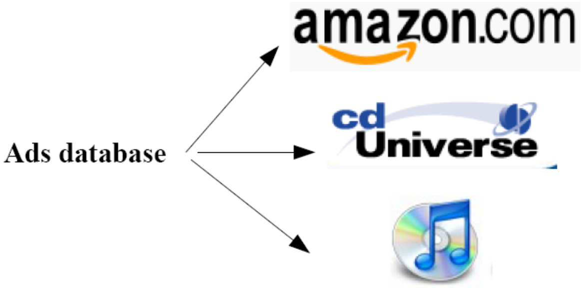
Outline

The setting: online advertising

- Most web services rely on advertising as a source of income
- Choosing the right ad to display is important
  - Try to tailor it to the customer
- An advertising engine is used to choose the ads
  - Some software that dynamically extracts ads based on user requests
- **Goal**: Maximize the number of user clicks

How an advertising engine operates

- The advertising engine keeps a database of ads, and given a request, returns a set of relevant ads



Measuring usefulness: Clickthrough rate

- A natural quantity of interest is the clickthrough rate
- The clickthrough rate represents the fraction of times that an ad was clicked when it is displayed:

$$\text{Clickthrough rate}(a) = \frac{\#\text{ times ad } a \text{ is clicked}}{\#\text{ times } a \text{ is displayed}}$$

- A high click through rate signifies a well-chosen ad

Measuring usefulness: Clickthrough rate

- Theoretically, we can think of the clickthrough rate as being the probability of a click given some information about an ad:

$$\text{Theoretical clickthrough rate}(a) = \text{Pr}(\text{Click}|f(a))$$

where $f(a)$ captures properties of an ad, surrounding pages, the viewer, etc.

Measuring usefulness: Click volume

- A related theoretical quantity is the click volume
- This is the number of times that an ad would be clicked, assuming it is shown in all requests
- We can think of a request as being a potential opportunity for display; the click volume is just a sum of clickthrough rates over all requests r:

$$\text{Click volume}(a) = \sum_r \text{Clickthrough rate}(a, r)$$

Measuring usefulness: Click volume

- More theoretically, the click volume can be expressed

$$\text{Click volume}(a) = \sum_r \Pr(\text{Click}|a, r)$$

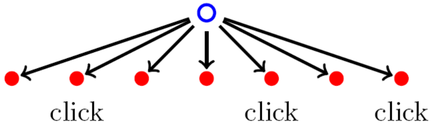
- This can be thought of as an unweighted average

How would knowing the click volume help us?

- Intuitively, the click volume helps us answer many relevant questions
  - The maintainer of the engine can estimate how many clicks an ad can expect to receive, and hence set the optimal price
  - The volume of an ad can help determine the target audience for an ad
  - We can use the volume to predict the purchase market for an ad
  - etc.
- **In short**: it is a useful thing to estimate!
  - So how do we do this?

Estimating click volume

- A natural question to ask is how we can estimate the click volume for an ad
- Specifically, given an ad a, assuming it is displayed at all requests, how many of those displays will result in a click?



Aside: is the assumption justified?

- We said that we assumed an ad is displayed on all requests; is such an assumption justified?
- If "all" means a random sample, then

$$\begin{aligned}\text{Click volume}(a) &= \sum_r \text{Clickthrough rate}(a, r) \\ &\approx \Pr(\text{Click}|a)\end{aligned}$$

Where this paper fits in

- This paper looks at how we can estimate click volume for a new ad, given the past history of clicks on different ads
- The key idea is to use linear regression on this history to predict the new click volume
- The computational constraints are that the data here is large-scale and sparse
- An algorithm is proposed that solves the regression problem efficiently, with guaranteed convergence
  - **Critique #1**: There is no experimental testing of the algorithm
  - **Critique #2**: There is no mention of other work on linear regression

Outline

Representing ads and requests

- The two important components of the formal model are ads and requests
- We represent an ad as a vector $a \in \mathcal{A}$, each component indicating a particular feature (e.g. text, link, phone number, ...):

$$a = (a_1, \ldots, a_m)$$

- We also represent a request as a vector $r \in \mathcal{R}$, similarly representing a collection of features (e.g. IP address, referrer, query, ...)

$$r = (r_1, \ldots, r_n)$$

- Note that these vectors may be sparse e.g. bag of words model

Introducing events

- We define an event to be the result of the interaction between an ad and a request
- We can think of an event e being a function of a, r, which we also represent by a vector:

$$e(a, r) = (e_1, \ldots, e_p)$$

Creating a history table

- We need some structure to help study the behaviour of ads and their requests
- A history table can be thought of as a collection of events, and whether or not they resulted in a click
- If we let $b_i \in \{0, 1\}$ denote a click, then

$$HT = \{(e(a_1, r_1), b_1), \ldots, (e(a_n, r_n), b_n)\}$$

Intuitive representation

- As the name suggests, we can represent a history table in a tabular form
- Let rows denote ads and columns ad requests

	Ad requests				
Ads					
	-		+ - +		
				- + +	
		+			
					- +

- Note that each cell denotes the b_i's for all events formed by the pair (a, r)

Sparsity of table

- An important property of the history table, when viewed in matrix form, is that it is sparse along the columns
- The reason is that each column usually covers only a few cells: the ads for a given request are generated by the advertising engine
- It is assumed that if the size of the table is $m \times n$, there are $O(n)$ nonzero entries
  - This is an important assumption used later to ease computation

Problem: estimating the click volume

- We can more formally specify our problem now

Problem

Given an ad a, we wish to estimate Click volume(a), assuming:

- the distribution of requests follows that of the history table
- the ad a would be shown on *all* the requests

Outline

Approaches that don't work: nearest neighbour

- Nearest neighbour solutions
  - We could try to find nearby ads and use their click volumes to estimate the new volume
  - **Problem # 1**: Similarity of ads does not tell us anything about that of volumes
  - **Problem # 2**: They are too slow!

Approaches that don't work: collaborative filtering

- Collaborative filtering
  - If we view the requests as users, and ads as movies, this is similar to a prediction problem!
  - **Problem # 1**: We have absolutely no data about the new "movie", or ad in this case (the cold-start problem)
  - **Problem # 2**: This ignores extra information, such as term similarity between ads and requests

Suggested approach

- The approach followed in the paper is three-fold:
  1. Perform dimensionality reduction on the history table
  2. Aggregate the values in the reduced table
  3. Perform a least-squares regression
- We look at the steps in turn

Reduction of table

- We can convert the table into a matrix by store clickthrough rates for events, rather than storing separate click values for each event
- We can also create generalized events $e' = D(e)$ that reduce the dimensionality of events by some function D
  - e.g. D might merge events with similar words
- Transform

$$\{(e_1, b_1), (e_2, b_2), \ldots, (e_k, b_k)\} \mapsto \left(e', \frac{\sum_i b_i}{k}\right)$$

where $e' = D(e_1) = D(e_2) = \ldots = D(e_k)$ represents the equivalence class of $e_1, \ldots, e_k$

Reduced history table

- Dimensionality reduction and aggregation gives us a reduced history table
- We can think of it as being represented by a vector T of generalized events, and a vector β of click through rates for each event

$$T = \begin{bmatrix} \boldsymbol{e'_1} \\ \boldsymbol{e'_2} \\ \vdots \\ \boldsymbol{e'_n} \end{bmatrix}, \beta = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix}$$

Estimating clickthrough rate

- **Problem**: Given a new generalized event $\boldsymbol{e'_m}$, we want to estimate its clickthrough rate β_m
- But this is equivalent to asking: is there a function f so that

$$f(T) \approx \beta?$$

  - T serves as our training set
- In our case, we assume that there is a linear relation between T, β
- This suggests a specific choice for f...

Estimating click volume

- **Aim**: We'd like to find a vector $\boldsymbol{\alpha}$ so that

$$T\boldsymbol{\alpha} \approx \boldsymbol{\beta}$$

- This is a problem of linear regression
  - There are more equations than variables, hence this is an overconstrained problem
  - No exact solution in general, so we want one with minimal discrepancy

Formulating problem as logistic regression

- We need to address a technical point: regression works over $[-\infty, \infty]$, whereas we want our $\boldsymbol{\beta}$ to be over $[0, 1]$
- **Solution**: We perform a logit transform to get a new vector $\boldsymbol{\gamma} = \mathrm{logit}(\boldsymbol{\beta})$, which now takes values in $[-\infty, \infty]$
  - **Recall**: The logit function is $\mathrm{logit} : x \mapsto \log \frac{x}{1-x}$
- The equivalent problem is

$$\min ||T\boldsymbol{\alpha} - \boldsymbol{\gamma}||$$

- **Note**: This is really logistic regression!

Solution summary

1. Find the equivalence classes of events, get a matrix T
2. Aggregate click values for equivalent events, get a vector $\boldsymbol{\beta}$ of clickthrough proportions
3. Find the vector $\boldsymbol{\alpha}$ which minimizes

$$||T\boldsymbol{\alpha} - \boldsymbol{\gamma}||$$

4. The predicted click volume is then simply

$$CV(a) = \sum_i \text{logit}^{-1}(\boldsymbol{\alpha}.e'(a, r_i))$$

Are we done?

- Intuitively, this method will work
- **Question**: But is it efficient?

Are we done?

- Intuitively, this method will work
- **Question**: But is it efficient?
- **Answer**: No!
  - Standard regression techniques involves inverting a large, non-sparse matrix
- We need a new technique to efficiently solve this problem

Note on sampling requests

- Notice that we need to sum over all requests to estimate the click volume
- This is potentially an intractable operation
- **Solution**: Consider only a sample of requests, e.g. over a fixed time window

Outline

1. Background: Sponsored search
2. Formal description: the history table
3. Estimating click volume with linear regression
4. Efficient sparse linear regression

A geometric interpretation of $T\alpha$

- If we write out the matrix product $T\boldsymbol{\alpha}$, it is clear that

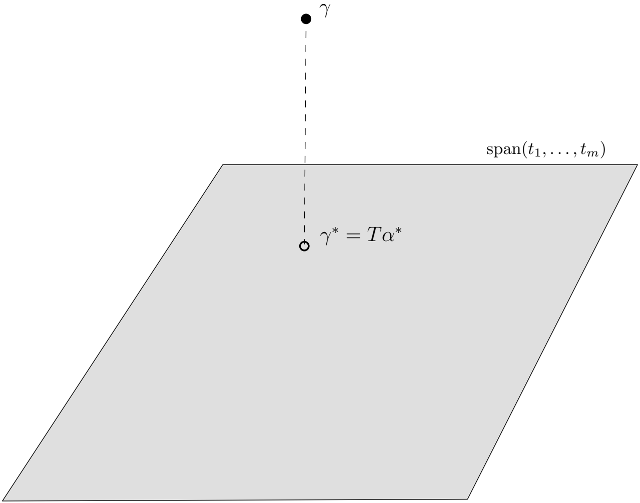
$$T\boldsymbol{\alpha} = \begin{bmatrix} T_{11}\alpha_1 + \ldots + T_{1m}\alpha_m \\ \vdots \\ T_{n1}\alpha_1 + \ldots + T_{nm}\alpha_m \end{bmatrix}$$
$$= \alpha_1 t_1 + \ldots + \alpha_m t_m$$

- So, $T\boldsymbol{\alpha}$ is nothing but a projection of γ onto the span of the columns of T:

$$T\boldsymbol{\alpha} \in \mathsf{span}(t_1, \ldots, t_m)$$

Geometric framework for our problem

- When we want $||T\boldsymbol{\alpha} - \boldsymbol{\gamma}||$ to be minimal, we really want to find the optimal projection of $\boldsymbol{\gamma}$ onto the span of T's columns



Sequential solution

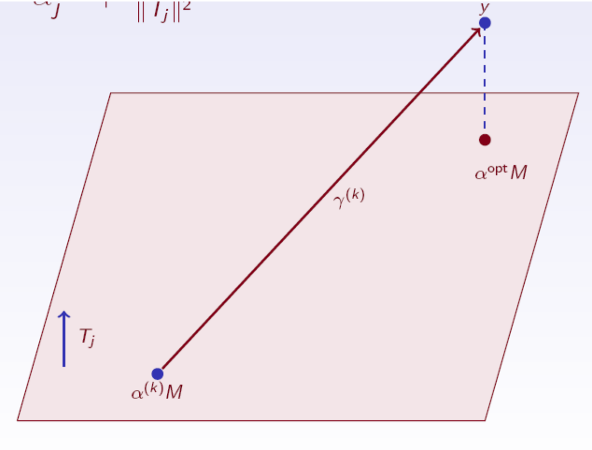
- We will look at a sequence of vectors $\{\boldsymbol{\alpha}^{(k)}\}_{k \in \mathbb{N}}$, starting from the guess $\boldsymbol{\alpha}^{(0)} = \mathbf{0}$
- **Idea**: Solve this problem by focussing on one column at a time

Geometrically inspired update

- **Update rule**: Choose any j, and fix all columns but j; now seek the best projection of $\boldsymbol{\gamma}$ onto the jth column of T
- To do this, we project the current error in each step of our solution onto the column vector t_j
  - This is the same as projecting γ, by linearity

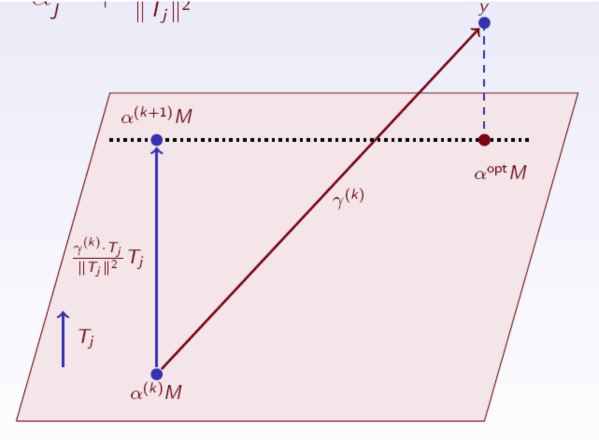
Visualizing the update

Find the current error in our solution



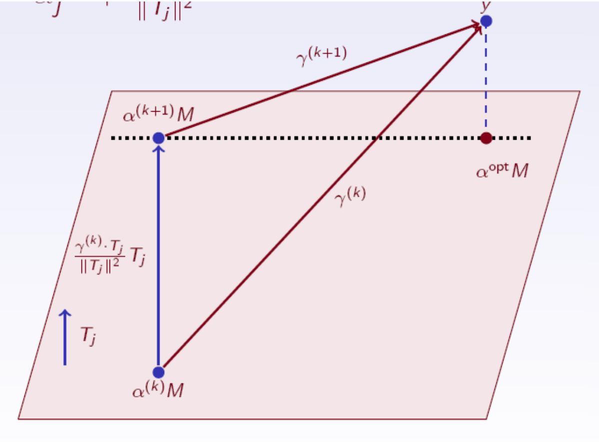
Visualizing the update

Project the error onto the vector t_j



Visualizing the update

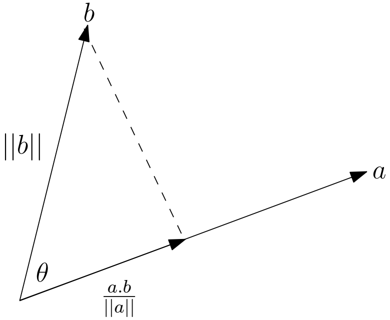
Now repeat this operation



Recap: vector projection

- Recall that the optimal projection of vector $\boldsymbol{b}$ onto $\boldsymbol{a}$ is

$$\mathrm{proj}_{\boldsymbol{a}}\boldsymbol{b} = \frac{\boldsymbol{a}.\boldsymbol{b}}{||\boldsymbol{a}||^2}\boldsymbol{a}$$



Measuring how close we are to optimal

- Suppose that the discrepancy at step k is denoted by

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

- Suppose also that the columns we choose are $J = \{j_1, j_2, \ldots\}$
- We can derive the discrepancy at step $(k+1)$:

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

  - This means we project the previous discrepancy onto the currently chosen column

Update rule

- There is a duality to the changes to our α vectors
- Our update for α works only on its jth component:

$$\alpha_j^{(k+1)} = \alpha_j^{(k)} + \frac{\varphi^{(k)}.t_j}{||t_j||^2}$$

The algorithm

while change in $||\varphi|| > \epsilon$

Choose $j \in [1, m]$ arbitrarily

$$\alpha_j \leftarrow \alpha_j + \frac{\varphi.t_j}{||t_j||^2}$$

$$\varphi \leftarrow \varphi - \frac{\varphi.t_j}{||t_j||^2} t_j$$

Convergence of the updates

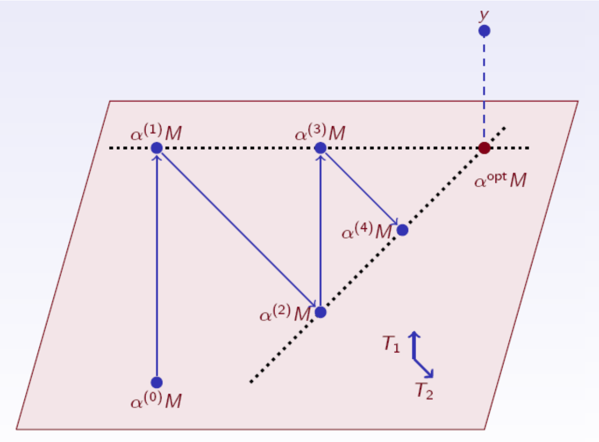
- Our updates are intuitive, but how well do they work?
- We have the following guarantee

[Convergence theorem]

Suppose we have a set of columns $J = \{j_1, j_2, \ldots, \}$, where every column appears infinitely many times. Then, the sequence $\varphi^{(k)}$, as defined above, will converge to φ^*, the minimal orthogonal projection error of γ onto the span of T

Convergence of the updates

Intuitively, our updates should converge: e.g. in the case where we just have two columns



Proof outline

- The proof is in three steps
  1. Show that $||\varphi^{(k)}||$ is bounded and monotone decreasing for every k
  2. Assume that $||\varphi^{(k)}||$ does not converge to φ^*, but instead converges to ψ
  3. Show that $||\varphi^k||$ does not converge to ψ

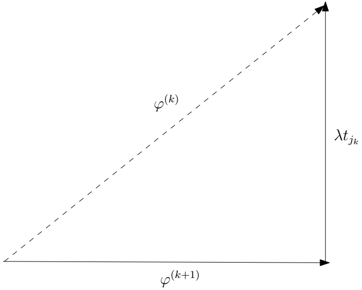
Bounded nature of $||\varphi^{(k)}||$

- First, note that the vectors $\varphi^{(k+1)}$ and t_{j_k} are always orthogonal:

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

- Then, Pythagoras' theorem tells us that

$$0 \leq ||\varphi^{(k+1)}|| \leq ||\varphi^{(k)}|| \leq ||\varphi^{(0)}||$$



Towards a contradiction

- Since we know that $||\varphi^{(k)}||$ is bounded and monotone, it must converge [Monotone Convergence theorem]
- Say that $||\varphi^{(k}||$ converges to $\psi \neq \varphi^*$

Towards a contradiction

- We now show that ψ cannot be a good enough bound on $||\varphi^{(k)}||$
- At some stage, we know that $||\varphi^{(k)}||$ must come within distance $\frac{c}{2}$ of φ^*, for any constant c
- Now look at the potential updates
  1. $t_{j_k} \perp \psi$: this will mean we get closer to ψ
  2. $t_{j_k} \not\perp \psi$: this will mean the size of $||\varphi^{(k)}||$ dramatically reduces

Case 1: $t_{j_k} \perp \psi$

- If $t_{j_k} \perp \psi$, then $t_{j_k}.\psi = 0$
- We have

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

- But the terms on the RHS are orthogonal, which means that

$$||\varphi^{(k)} - \psi|| \geq ||\varphi^{(k+1)} - \psi||$$

- This means we cannot escape ψ, as we only get closer to it

Case 2: $t_{j_k} \not\angle \psi$

- If $t_{j_k} \not\angle \psi$, we can lower bound the shift by

$$\begin{aligned}
\frac{\varphi^{(k)}.t_{j_k}}{||t_{j_k}||^2} t_{j_k} &= \frac{\psi.t_{j_k} + (\varphi^{(k)} - \psi).t_{j_k}}{||t_{j_k}||^2} t_{j_k} \\
&\geq \frac{c||t_{j_k}|| - \frac{c}{2}||t_{j_k}||}{||t_{j_k}||^2} t_{j_k} \\
&\geq \frac{c}{2||t_j||} t_{j_k}
\end{aligned}$$

- But $\varphi^{(k)} \perp \varphi^{(k+1)}$, so

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

The contradiction

- But these results lead to a contradiction
- We have said that
  1. $\varphi^{(k)}$ visits the $c/2$ neighbourhood of ψ infintely often
  2. Once it's inside the neighbourhood, $j_k \perp \psi$ updates cannot let it escape
  3. Once it's inside the neighbourhood, $j_k \not\perp \psi$ updates cause a substantial decrease in the size of $||\varphi^{(k)}||$
- But $||\varphi^{(k)}||$ is nonnegative and monotone, so this cannot be the case
- Contradiction!

Where are we now?

- We have shown that the given approach does converge to the optimal solution
- This is good, but there is another natural question to ask...

Runtime analysis

- How efficient is this approach?
- We show that we can exploit sparsity of the history table:

[Runtime bound]

We can perform each update j in time that is linear in the number of nonzero elements in the history table

How to do the update

- Suppose there are q_j non-zero elements in t_j
- We do the update as follows
  1. Precompute all norms $||t_j||$
  2. Update α_j^k in $\mathcal{O}(q_j)$ time: we just need to compute the dot-product of t_j and $\varphi^{(k)}$
  3. Update φ^k in $\mathcal{O}(q_j)$ time: again we just need to compute a dot-product

Note on convergence

- Important caveat: We don't have any guarantee on the convergence rate
- All we know is that it *does* converge
- Authors state this as an open problem
  - More concerned with introducing a new idea into the literature

Outline

Conclusion

- The click volume is a quantity of interest in web advertising
- Given past history of clicks, we can estimate the click volume of a new ad using linear regression
- When the data is sparse, we need a new technique to efficiently compute the regression
- Paper provides an algorithm for efficiently solving this problem, with proof of correctness (but lacking a convergence rate bound)