Scaling Matrix Factorization for Recommendation with Randomness

Lei Tang
@WalmartLabs
San Bruno, CA 94066, USA
leitang@acm.org

Patrick Harrington
@WalmartLabs
San Bruno, CA 94066, USA
pharrington@walmartlabs.com

ABSTRACT

Recommendation is one of the core problems in eCommerce. In our application, different from conventional collaborative filtering, one user can engage in various types of activities in a sequence. Meanwhile, the number of users and items involved are quite huge, entailing scalable approaches. In this paper, we propose one simple approach to integrate multiple types of user actions for recommendation. A two-stage randomized matrix factorization is presented to handle large-scale collaborative filtering where alternating least squares or stochastic gradient descent is not viable. Empirical results show that the method is quite scalable, and is able to effectively capture correlations between different actions, thus making more relevant recommendations.

Categories and Subject Descriptors

H.2.8 [**Information Technology and Systems**]: Database Applications—*Data Mining*; H.3.3 [**Information Storage and Retrieval**]: Information Filtering

Keywords

Randomness; Matrix Factorization; Randomized Matrix Factorization; Recommendation; eCommerce;

1. INTRODUCTION

Recommendation has been one of the core problems in eCommerce and other online services. It is a critical factor for personalization and user engagement. Since the Netflix Prize Challenge, collaborative filtering has attracted lots of attention, and matrix factorization (MF) gained its momentum [1, 3, 4]. In this work, we focus on MF techniques for recommendation due to its outstanding performance.

In our particular application, we recommend products to Walmart customers. Of course, this recommendation can target users through any available channels like emails, the walmart.com website or display ads showing on partner websites. Both the number of users and products are huge. Nevertheless, the eCommerce domain presents distinctive challenges that were seldom addressed in previous research. 1) Most of existing work focuses on collaborative filtering with one rating-like matrix. For example, movie ratings, transactions etc. While in many cases, we have different types of activities associated with users, which could be used for recommendation. In our setting, we have transactions, browsing activities and search activities for Walmart customers. Additional user profile information is also available. We need a smart way to integrate all these different signals without complicating a model too much. 2) On the other hand, it is not trivial to solve the highly non-convex MF problem in large scale. Common approaches involve iterative process through alternating least squares or stochastic gradient descent. And some implementations are available in MapReduce. But current MapReduce framework involves too much overhead. In our application, just one iteration of alternating least squares takes almost half day. It is unbearable to run multiple iterations if only limited cluster resources are available.

In this work, we propose one simple way to integrate multiple signals about users without any tuning parameter. We also propose a two-stage approach with randomness to solve the matrix factorization such that we are able to compute recommendations for tens of millions of users within hours.

WWW 2013 Companion, May 13–17, 2013, Rio de Janeiro, Brazil.
ACM 978-1-4503-2038-2/13/05.

2. SCALING MATRIX FACTORIZATION

We collect different types of events associated with individuals: purchases and online views. The distributions of all the events are quite different yet correlated. Our goal is to recommend products online to users. All raw events are represented as a quadruple $< u_i, a_k, p_j, t >$ with u, a, p and t denoting user, action, product, and time respectively. Suppose the time we compute recommendation is t_0. Then the corresponding weight

$$A_{ij}^{(k)} = \exp\left\{-\frac{t_0 - t}{\beta}\right\} \log \frac{N}{N_j} \tag{1}$$

β is a parameter to control the decay speed. We set to 60 days in our experiments. N is the total number of users, and N_j is the number of users who have purchased or viewed product p_j depending on the action. The first term captures the recency of the event, and the latter, like the inverse document frequency in text mining, promotes those less popular products because they are more specific in capturing user interests.

To integrate all types of activities associated with one user, we propose to concatenate matrixes of different actions into one. That is, $A = [A^{(1)}, A^{(2)}, A^{(3)}]$, with $A^{(k)}$ indicating different events. Note that the size of matrix A is huge but extremely sparse. In our setting, we ended up with a matrix of tens of millions of rows (users) and millions of columns (products in different actions). A standard approach is to approximate the observed entries by low-rank latent factors

Input: Given an $n \times m$ matrix A and a target rank k
output: latent factor Q

1. draw a random Gaussian $m \times k$ matrix G;
2. $Y = AG$, $Y \in R^{n \times k}$;
3. orthonomralize Y by QR decomposing $Y = Q_Y R_Y$;
4. $B = Q_Y^T A$, $B \in R^{k \times m}$;
5. Compute the SVD of $B = U \Sigma V^T$;
6. return $Q = V \Sigma^{1/2}$.

Figure 1: Algorithm: randomized SVD to compute Q

P and Q as follows:

$$\min_{P,Q} \sum_{(i,j) \in \Omega} (A_{ij} - p_i^T q_j)^2 + \lambda(||P||_F^2 + ||Q||_F^2) \qquad (2)$$

where $P \in R^{m \times k}, Q \in R^{n \times k}$ and Ω denotes observed entries. The problem itself is non-convex with no analytical solution. Yet, it becomes a least squares problem when P or Q is fixed [5]. Alternating least squares randomly initializes P or Q and then iteratively updates P and Q given each other. But even so, one iteration of our data takes nearly half day in our medium-sized cluster. Moreover, the cluster is very busy running many different product applications. Such a computationally intensive procedure is not an option. We need a more scalable solution.

2.1 Two-Stage MF with Randomness

We hope that through some procedure, we are able to obtain Q accurately, so that we need only one iteration of least squares fit in order to compute P. After discarding the regularization term and the loss constrained to only observed entries, Eq (2) becomes the following low-rank approximation: $\min_{P,Q} \sum_{(i,j)} (A_{ij} - p_i^T q_j)^2$, which has a global optimal solution, i.e. the truncated singular value decomposing (SVD) of matrix A. Therefore, we suggest the following two-stage matrix factorization:

1. Compute the truncated SVD of matrix $A \approx U \Sigma_k V^T$ and obtain $Q = V \Sigma^{1/2}$ following the algorithm in Figure 1.
2. Given Q, solve P. As p_i can be computed independently of each other, this step can be accomplished through one map-reduce job.

Stage 1 requires the computation of truncated SVD. But all deterministic methods are too expensive. Thanks to recent advancement of randomized algorithms, it is possible to obtain an approximate SVD by projecting the original matrix into a small-sized space so that the SVD computation is viable [2]. There are many variants of the randomized SVD as to achieve efficiency and better theoretical error bounds. Please refer to [2] for detailed treatment and approximation error bound. Here we list one straightforward implementation in Figure 1. It first aims to find a Q_Y such that $||A - Q_Y Q_Y^T A|| \leq \epsilon$. Then SVD can be performed on a much smaller-sized thin matrix B in line 4 and 5.

3. EXPERIMENTS

We collect 15-month historical data about user purchases and browsing activities at Walmart.com for training, and use a small portion of 1-week email click traffic in February 2013 for testing. *The whole analysis is privacy-friendly as all customer ids are randomly anonymized into integers.* In each email, one user will be exposed to eight different products as recommended by some rules in existing production systems. We believe that a better recommendation algorithm should rank those items being clicked higher among the recommended products. In order to compare the relative ranking of different models, mean average precision (MAP) and normalized discounted cumulative gain (NDCG)[4] are adopted as performance metric.

Table 1: Performance Comparison of Different Methods

Method	MAP	NDCG
Popularity	.377098	.534320
2-stage MF with irlba	.398874	.551616
2-stage MF with randomness	**.404095**	**.555462**
alternating least squares	NA	NA

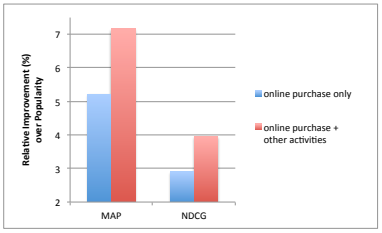

Figure 2: Performance of Integrating Multiple Actions

First, we would like to see whether the two-stage matrix factorization is making sense. We use recommendation based on popularity (denoted as Popularity) as a baseline. For comparison, we also tried to compute SVD using deterministic methods with the irlba (implicitly-restarted Lanczos bidiagonalization) package in R. In order to obtain any output, we have to feed into a much smaller matrix. So we randomly sample 1% of users (rows) from the matrix A. When the low rank is set to 400, the irlba takes two days to compute Q, while randomized SVD takes only 3 hours. Not to mention that irlba deals with a much smaller-sized matrix. As shown in Table 1, our proposed two-stage matrix factorization with randomness achieves the best performance.

Secondly, we want to verify that integrating multiple signals yields better result. It is shown in Figure 2 that our method is able to inject all types of user actions and achieves a much better performance than using transaction alone.

The result above shows that the proposed two-stage matrix factorization with randomness is quite scalable. It is able to capture correlations between different types of actions, thus making more relevant recommendations.

4. REFERENCES

[1] A. S. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. In *WWW* , pages 271–280, 2007.

[2] N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.*, 53(2):217–288, May 2011.

[3] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[4] G. Takács, I. Pilászy, B. Németh, and D. Tikk. Scalable collaborative filtering approaches for large recommender systems. *J. Mach. Learn. Res.*, 10:623–656, June 2009.

[5] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan. Large-scale parallel collaborative filtering for the netflix prize. In *AAIM*, pages 337–348, 2008.