Top-N Recommender System via Matrix Completion

Zhao Kang Chong Peng Qiang Cheng
Department of Computer Science, Southern Illinois University, Carbondale, IL 62901, USA
{Zhao.Kang, pchong, qcheng}@siu.edu

Abstract

Top-N recommender systems have been investigated widely both in industry and academia. However, the recommendation quality is far from satisfactory. In this paper, we propose a simple yet promising algorithm. We fill the user-item matrix based on a low-rank assumption and simultaneously keep the original information. To do that, a nonconvex rank relaxation rather than the nuclear norm is adopted to provide a better rank approximation and an efficient optimization strategy is designed. A comprehensive set of experiments on real datasets demonstrates that our method pushes the accuracy of Top-N recommendation to a new level.

Introduction

The growth of online markets has made it increasingly difficult for people to find items which are interesting and useful to them. Top-N recommender systems have been widely adopted by the majority of e-commerce web sites to recommend size-N ranked lists of items that best fit customers' personal tastes and special needs (**?**). It works by estimating a consumer's response for new items, based on historical information, and suggesting to the consumer novel items for which the predicted response is high. In general, historical information can be obtained explicitly, for example, through ratings/reviews, or implicitly, from purchase history or access patterns (**?**).

Over the past decade, a variety of approaches have been proposed for Top-N recommender systems (**?**). They can be roughly divided into three categories: neighborhood-based collaborative filtering, model-based collaborative filtering, and ranking-based methods. The general principle of neighborhood-based methods is to identify the similarities among users/items (**?**). For example, item-based k-nearest-neighbor (ItemKNN) collaborative filtering methods first identify a set of similar items for each of the items that the consumer has purchased, and then recommend Top-N items based on those similar items. However, they suffer from low accuracy since they employ few item characteristics.

Model-based methods build a model and then generate recommendations. For instance, the widely studied matrix factorization (MF) methods employ the user-item similarities in their latent space to extract the user-item purchase patterns. Pure singular-value-decomposition-based (PureSVD) matrix factorization method (**?**) characterizes users and items by the most principal singular vectors of the user-item matrix. A weighted regularized matrix factorization (WRMF) (**?**; **?**) method applies a weighting matrix to differentiate the contributions from observed purchase/rating activities and unobserved ones.

The third category of methods rely on ranking/retrieval criteria. Here, Top-N recommendation is treated as a ranking problem. A Bayesian personalized ranking (BPR) (**?**) criterion, which is the maximum posterior estimator from a Bayesian analysis, is used to measure the difference between the rankings of user-purchased items and the rest items. BPR can be combined with ItemKNN (BPRkNN) and MF method (BPRMF). One common drawback of these approaches lies in low recommendation quality.

Recently, a novel Top-N recommendation method SLIM (**?**) has been proposed. From user-item matrix X of size $m \times n$, it first learns a sparse aggregation coefficient matrix $W \in \mathcal{R}_+^{n\times n}$ by encoding each item as a linear combination of all other items and solving an l_1-norm and l_2-norm regularized optimization problem. Each entry w_{ij} describes the similarity between item i and j. SLIM has obtained better recommendation accuracy than the other state-of-the-art methods. However, SLIM can only capture relations between items that are co-purchased/co-rated by at least one user, while an intrinsic characteristic of recommender systems is sparsity due to the fact that users typically rate only a small portion of the available items.

To overcome the above limitation, LorSLIM (**?**) has also imposed a low-rank constraint on W. It solves the following problem:

$$\min_W \frac{1}{2}\|X - XW\|_F^2 + \frac{\beta}{2}\|W\|_F^2 + \lambda\|W\|_1 + \gamma\|W\|_* \\ s.t. \quad W \geq 0, \quad diag(W) = 0,$$

where $\|W\|_*$ is the nuclear norm of W, defined as the sum of its singular values. Low-rank structure is motivated by the fact that a few latent variables from F that explain items' features in factor model $W \approx FF^T$ are of low rank. After obtaining W, the recommendation score for user i about an un-purchased/-rated item j is $\hat{x}_{ij} = \mathbf{x}_i^T\mathbf{w}_j$, where $x_{ij} = 0$, $\mathbf{x}_i^T$ is the i-th row of X, and $\mathbf{w}_j$ is the j-th column of W.

Thus $\hat{X} = XW$. LorSLIM can model the relations between items even on sparse datasets and thus improves the performance.

To further boost the accuracy of Top-N recommender systems, we first fill the missing ratings by solving a nonconvex optimization problem, based on the assumption that the user' ratings are affected by only a few factors and the resulting rating matrix should be of low rank (**?**), and then make the Top-N recommendation. This is different from previous approaches: Middle values of the rating ranges, or the average user or item ratings are commonly utilized to fill out the missing ratings (**?**; **?**); a more reliable approach utilizes content information (**?**; **?**; **?**), for example, the missing ratings are provided by autonomous agents called filterbots (**?**), which rate items based on some specific characteristics of their content; a low rank matrix factorization approach seeks to approximate X by a multiplication of low rank factors (**?**). Experimental results demonstrate the superior recommendation quality of our approach.

Due to the inherent computational complexity of rank problems, the non-convex rank function is often relaxed to its convex surrogate, i.e. the nuclear norm (**?**; **?**). However, this substitution is not always valid and can lead to a biased solution (**?**; **?**). Matrix completion with nuclear norm regularization can be significantly hurt when entries of the matrix are sampled non-uniformly (**?**). Nonconvex rank approximation has received significant attention (**?**; **?**). Thus we use log-determinant ($logdet$) function to approximate the rank function and design an effective optimization algorithm.

Problem Formulation

The incomplete user-item purchases/ratings matrix is denoted as M of size $m \times n$. M_{ij} is 1 or a positive value if user i has ever purchased/rated item j; otherwise it is 0. Our goal is to reconstruct a full matrix X, which is supposed to be low-rank. Consider the following matrix completion problem:

$$\begin{aligned} &\min_X logdet((X^TX)^{1/2} + I) \\ &s.t. \quad X_{ij} = M_{ij}, \quad (i,j) \in \Omega, \end{aligned} \tag{1}$$

where Ω is the set of locations corresponding to the observed entries and $I \in \mathcal{R}^{n\times n}$ is an identity matrix. It is easy to show that $logdet((X^TX)^{1/2} + I) \leq \|X\|_*$, i.e., $logdet$ is a tighter rank approximation function than the nuclear norm. $logdet$ also helps mitigate another inherent disadvantage of the nuclear norm, i.e., the imbalanced penalization of different singular values (**?**). Previously $logdet(X + \delta I)$ was suggested to restrict the rank of positive semidefinite matrix X (**?**), which is not guaranteed for more general X, and also δ is required to be small, which leads to significantly biased approximation for small singular values. Compared to some other nonconvex relaxations in the literature (**?**), our formulation enjoys the simplicity and efficacy.

Methodology

Considering that the user-item matrix is often nonnegative, we add nonnegative constraint $X \geq 0$, i.e., element-wise positivity, for easy interpretation of the representation. Let $\mathcal{P}_\Omega$ be the orthogonal projection operator onto the span of matrices vanishing outside of Ω (i.e., Ω^c) so that

$$(\mathcal{P}_\Omega(X))_{ij} = \begin{cases} X_{ij}, & \text{if } X_{ij} \in \Omega; \\ 0, & \text{if } X_{ij} \in \Omega^c. \end{cases}$$

Problem (1) can be reformulated as

$$\begin{aligned} \min_X &logdet((X^TX)^{1/2} + I) + l_{\mathcal{R}_+}(X) \\ &s.t. \quad \mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M), \end{aligned} \tag{2}$$

where $l_{\mathcal{R}_+}$ is the indicator function, defined element-wisely as

$$l_{\mathcal{R}_+}(x) = \begin{cases} 0, & \text{if } x \geq 0; \\ +\infty, & \text{otherwise.} \end{cases}$$

Notice that this is a nonconvex optimization problem, which is not easy to solve in general. Here we develop an effective optimization strategy based on augmented Lagrangian multiplier (ALM) method. By introducing an auxiliary variable Y, it has the following equivalent form

$$\begin{aligned} &\min_{X,Y} logdet((X^TX)^{1/2} + I) + l_{\mathcal{R}_+}(Y) \\ &s.t. \quad \mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M), \quad X = Y, \end{aligned} \tag{3}$$

which has an augmented Lagrangian function of the form

$$\begin{aligned} \mathcal{L}(X,Y,Z) = &logdet((X^TX)^{1/2} + I) + l_{\mathcal{R}_+}(Y) + \\ &\frac{\mu}{2}\|X - Y + \frac{Z}{\mu}\|_F^2 \quad s.t. \quad \mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M), \end{aligned} \tag{4}$$

where Z is a Lagrange multiplier and $\mu > 0$ is a penalty parameter.

Then, we can apply the alternating minimization idea to update X, Y, i.e., updating one of the two variables with the other fixed.

Given the current point X^t, Y^t, Z^t, we update X^{t+1} by solving

$$\begin{aligned} X^{t+1} = \arg\min_X &logdet((X^TX)^{1/2} + I) + \\ &\frac{\mu^t}{2}\|X - Y^t + \frac{Z^t}{\mu^t}\|_F^2 \end{aligned} \tag{5}$$

This can be converted to scalar minimization problems due to the following theorem (**?**).

Theorem 1 *If $F(Z)$ is a unitarily invariant function and SVD of A is $A = U\Sigma_A V^T$, then the optimal solution to the following problem*

$$\min_Z F(Z) + \frac{\beta}{2}\|Z - A\|_F^2 \tag{6}$$

is Z^* with SVD $U\Sigma_Z^* V^T$, where $\Sigma_Z^* = diag\,(\sigma^*)$; moreover, $F(Z) = f \circ \sigma(Z)$, where $\sigma(Z)$ is the vector of nonincreasing singular values of Z, then σ^* is obtained by using the Moreau-Yosida proximity operator $\sigma^* = prox_{f,\beta}(\sigma_A)$, where $\sigma_A := diag(\Sigma_A)$, and

$$prox_{f,\beta}(\sigma_A) := \arg\min_{\sigma \geq 0} f(\sigma) + \frac{\beta}{2}\|\sigma - \sigma_A\|_2^2. \tag{7}$$

Algorithm 1 Solve (3)

Input: Original imcomplete data matrix $M_\Omega \in \mathcal{R}^{m\times n}$, parameters $\mu^0 > 0, \gamma > 1$.
Initialize: $Y = \mathcal{P}_\Omega(M)$, $Z = 0$.
REPEAT
1: Obtain X through (10).
2: Update Y as (12).
3: Update the Lagrangian multipliers Z by

$$Z^{t+1} = Z^t + \mu^t(X^{t+1} - Y^{t+1}).$$

4: Update the parameter μ^t by $\mu^{t+1} = \gamma\mu^t$.
UNTIL stopping criterion is met.

According to the first-order optimality condition, the gradient of the objective function of (7) with respect to each singular value should vanish. For $logdet$ function, we have

$$\frac{1}{1+\sigma_i} + \beta(\sigma_i - \sigma_{i,A}) = 0 \; s.t. \; \sigma_i \geq 0. \tag{8}$$

The above equation is quadratic and gives two roots. If $\sigma_{i,A} = 0$, the minimizer σ_i^* will be 0; otherwise, there exists a unique minimizer. Finally, we obtain the update of X variable with

$$X^{t+1} = Udiag(\sigma^*)V^T. \tag{9}$$

Then we fix the values at the observed entries and obtain

$$X^{t+1} = \mathcal{P}_{\Omega^c}(X^{t+1}) + \mathcal{P}_\Omega(M). \tag{10}$$

To update Y, we need to solve

$$\min_Y \quad l_{\mathcal{R}_+}(Y) + \frac{\mu^t}{2}\|X^{t+1} - Y + \frac{Z^t}{\mu^t}\|_F^2, \tag{11}$$

which yields the updating rule

$$Y^{t+1} = \max(X^{t+1} + Z^t/\mu^t, 0). \tag{12}$$

Here $\max(\cdot)$ is an element-wise operator. The complete procedure is outlined in Algorithm 1.

To use the estimated matrix $\hat{X}$ to make recommendation for user i, we just sort i's non-purchased/-rated items based on their scores in decreasing order and recommend the Top-N items.

Experimental Evaluation

Datasets

We evaluate the performance of our method on six different real datasets whose characteristics are summarized in Table 1. These datasets are from different sources and at different sparsity levels. They can be broadly categorized into two classes.

The first class includes Delicious, lastfm and BX. These three datasets have only implicit feedback (e.g., listening history), i.e., they are represented by binary matrices. In particular, Delicious was derived from the bookmarking and tagging information from a set of $2K$ users from Delicious social bookmarking system[1] such that each URL was bookmarked by at least 3 users. Lastfm corresponds to music artist listening information which was obtained from the last.fm online music system[2], in which each music artist was listened to by at least 10 users and each user listened to at least 5 artists. BX is a subset from the Book-Crossing dataset[3] such that only implicit interactions were contained and each book was read by at least 10 users.

Table 1: The datasets used in evaluation

dataset	#users	#items	#trns	rsize	csize	density	ratings
Delicious	1300	4516	17550	13.50	3.89	0.29%	-
lastfm	8813	6038	332486	37.7	55.07	0.62%	-
BX	4186	7733	182057	43.49	23.54	0.56%	-
ML100K	943	1682	100000	106.04	59.45	6.30%	1-10
Netflix	6769	7026	116537	17.21	16.59	0.24%	1-5
Yahoo	7635	5252	212772	27.87	40.51	0.53%	1-5

The "#users", "#items", "#trns" columns show the number of users, number of items and number of transactions, respectively, in each dataset. The "rsize" and "csize" columns are the average number of ratings for each user and on each item (i.e., row density and column density of the user-item matrix), respectively, in each dataset. Column corresponding to "density" shows the density of each dataset (i.e., density=#trns/(#users$\times$#items)). The "ratings" column is the rating range of each dataset with granularity 1.

The second class contains ML100K, Netflix and Yahoo. All these datasets contain multi-value ratings. Specifically, the ML100K dataset corresponds to movie ratings and is a subset of the MovieLens research project[4]. The Netflix is a subset of data extracted from Netflix Prize dataset[5] and each user rated at least 10 movies. The Yahoo dataset is a subset obtained from Yahoo!Movies user ratings[6]. In this dataset, each user rated at least 5 movies and each movie was rated by at least 3 users.

Evaluation Methodology

We employ 5-fold Cross-Validation to demonstrate the efficacy of our proposed approach. For each run, each of the datasets is split into training and test sets by randomly selecting one of the non-zero entries for each user to be part of the test set[7]. The training set is used to train a model, then a size-N ranked list of recommended items for each user is generated. The evaluation of the model is conducted by comparing the recommendation list of each user and the item of that user in the test set. For the following results reported in this paper, N is equal to 10.

Top-N recommendation is more like a ranking problem rather than a prediction task. The recommendation quality is measured by the hit-rate (HR) and the average reciprocal hit-rank (ARHR) (**?**). They directly measure the performance of the model on the ground truth data, i.e., what users have already provided feedback for. As pointed out in (**?**), they are the most direct and meaningful measures in Top-N recommendation scenarios. HR is defined as

$$HR = \frac{\#\text{hits}}{\#\text{users}}, \tag{13}$$

[1] http://www.delicious.com

[2] http://www.last.fm

[3] http://www.informatik.uni-freiburg.de/ cziegler/BX/

[4] http://grouplens.org/datasets/movielens/

[5] http://www.netflixprize.com/

[6] http://webscope.sandbox.yahoo.com/catalog.php?datatype=r

[7] We use the same data as in (**?**), with partitioned datasets kindly provided by Yao Cheng.

Table 2: Comparison of Top-N recommendation algorithms

method	Delicious						lastfm					
	params				HR	ARHR	params				HR	ARHR
ItemKNN	300	-	-	-	0.300	0.179	100	-	-	-	0.125	0.075
PureSVD	1000	10	-	-	0.285	0.172	200	10	-	-	0.134	0.078
WRMF	250	5	-	-	0.330	0.198	100	3	-	-	0.138	0.078
BPRKNN	1e-4	0.01	-	-	0.326	0.187	1e-4	0.01	-	-	0.145	0.083
BPRMF	300	0.1	-	-	0.335	0.183	100	0.1	-	-	0.129	0.073
SLIM	10	1	-	-	0.343	0.213	5	0.5	-	-	0.141	0.082
LorSLIM	10	1	3	3	0.360	0.227	5	1	3	3	0.187	0.105
Our	250	4	-	-	**0.382**	**0.241**	0.03	1.5	-	-	**0.206**	**0.113**
method	BX						ML100K					
	params				HR	ARHR	params				HR	ARHR
ItemKNN	400	-	-	-	0.045	0.026	10	-	-	-	0.287	0.124
PureSVD	3000	10	-	-	0.043	0.023	100	10	-	-	0.324	0.132
WRMF	400	5	-	-	0.047	0.027	50	1	-	-	0.327	0.133
BPRKNN	1e-3	0.01	-	-	0.047	0.028	2e-4	1e-4	-	-	0.359	0.150
BPRMF	400	0.1	-	-	0.048	0.027	200	0.1	-	-	0.330	0.135
SLIM	20	0.5	-	-	0.050	0.029	2	2	-	-	0.343	0.147
LorSLIM	50	0.5	2	3	0.052	0.031	10	8	5	3	0.397	0.207
Our	1.2e-3	1.3	-	-	**0.065**	**0.043**	6e-3	2.5	-	-	**0.428**	**0.215**
method	Netflix						Yahoo					
	params				HR	ARHR	params				HR	ARHR
ItemKNN	200	-	-	-	0.156	0.085	300	-	-	-	0.318	0.185
PureSVD	500	10	-	-	0.158	0.089	2000	10	-	-	0.210	0.118
WRMF	300	5	-	-	0.172	0.095	100	4	-	-	0.250	0.128
BPRKNN	2e-3	0.01	-	-	0.165	0.090	0.02	1e-3	-	-	0.310	0.182
BPRMF	300	0.1	-	-	0.140	0.072	300	0.1	-	-	0.308	0.180
SLIM	5	1.0	-	-	0.173	0.098	10	1	-	-	0.320	0.187
LorSLIM	10	3	5	3	0.196	0.111	10	1	2	3	0.334	0.191
Our	0.015	1.2	-	-	**0.226**	**0.127**	5e-3	1.1	-		**0.367**	**0.218**

The parameters for each method are as follows: ItemKNN: the number of neighbors k; PureSVD: the number of singular values and the number of iterations during SVD; WRMF: the dimension of the latent space and the weight on purchases; BPRKNN: the learning rate and regularization parameter λ; BPRMF: the dimension of the latent space and learning rate; SLIM: the l_2-norm regularization parameter β and the l_1-norm regularization parameter λ; LorSLIM: the l_2-norm regularization parameter β, the l_1-norm regularization parameter λ, the nuclear norm regularization parameter z and the auxiliary parameter ρ. Our: auxiliary parameters μ^0 and γ. N in this table is 10. Bold numbers are the best performance in terms of HR and ARHR for each dataset.

where #hits is the number of users whose item in the test set is recommended (i.e., hit) in the size-N recommendation list, and #users is the total number of users. An HR value of 1.0 indicates that the algorithm is able to always recommend the hidden item, whereas an HR value of 0.0 denotes that the algorithm is not able to recommend any of the hidden items.

A drawback of HR is that it treats all hits equally regardless of where they appear in the Top-N list. ARHR addresses it by rewarding each hit based on where it occurs in the Top-N list, which is defined as follows:

$$ARHR = \frac{1}{\#\text{users}} \sum_{i=1}^{\#\text{hits}} \frac{1}{p_i}, \tag{14}$$

where p_i is the position of the test item in the ranked Top-N list for the i-th hit. That is, hits that occur earlier in the ranked list are weighted higher than those occur later, and thus ARHR measures how strongly an item is recommended. The highest value of ARHR is equal to the hit-rate and occurs when all the hits occur in the first position, whereas the lowest value is equal to HR/N when all the hits occur in the last position of the list.

Comparison Algorithms

We compare the performance of the proposed method[8] with seven other state-of-the-art Top-N recommendation algorithms, including the item neighborhood-based collaborative filtering method ItemKNN (**?**), two MF-based methods PureSVD (**?**) and WRMF (**?**), two ranking/retrieval criteria based methods BPRMF and BPRKNN (**?**), SLIM (**?**), and LorSLIM (**?**).

Experimental Results

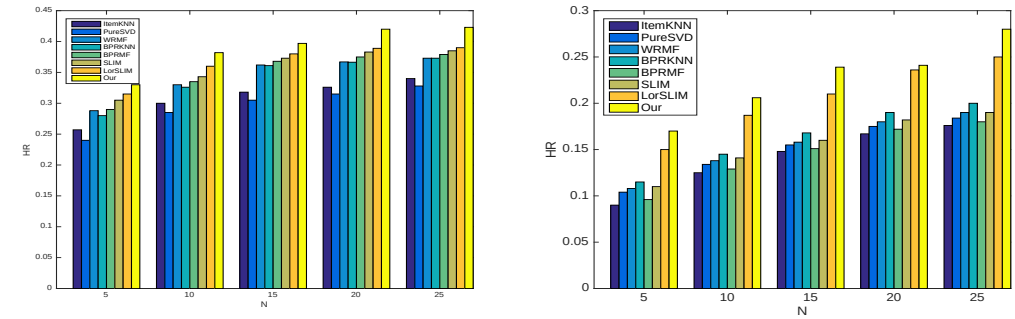
Top-N recommendation performance

We report the comparison results with other competing methods in Table 2. The results show that our algorithm performs better than all the other methods across all the datasets [9]. Specifically, in terms of HR, our method outperforms ItemKNN, PureSVD, WRMF, BPRKNN, BPRMF, SLIM and LorSLIM by 41%, 48.14%, 35.40%, 28.69%, 36.57%, 26.26%, 12.38% on average, respectively, over all the six datasets; with respect to ARHR, ItemKNN, PureSVD, WRMF, BPRKNN, BPRMF, SLIM and LorSLIM are improved by 48.55%, 60.38%, 48.58%, 37.14%, 49.47%, 31.94%, 14.15% on average, respectively.

Among the seven state-of-the-art algorithms, LorSLIM is substantially better than the others. Moreover, SLIM is a little better than others except on lastfm and ML100K among the rest six methods. Then BPRKNN performs best among the remaining five methods on average. Among the three MF-based models, BPRMF and WRMF have similar performance on most datasets and are much better than PureSVD on all datasets except on lastfm and ML100K.

[8] The implementation of our method is available at: https://github.com/sckangz/recom_mc.

[9] A bug is found, so the result in published version is updated. We apologize for any inconvenience caused.



(a) Delicious (b) lastfm

(c) BX (d) ML100K

(e) Netflix (f) Yahoo

Figure 1: Performance for Different Values of N.

Recommendation for Different Top-N

Figure 1 shows the performance achieved by the various methods for different values of N for all six datasets. It demonstrates that the proposed method outperforms other algorithms in all scenarios. What is more, it is evident that the difference in performance between our approach and the other methods are consistently significant. It is interesting to note that LorSLIM, the second most competitive method, may be worse than some of the rest methods when N is large.

Matrix Reconstruction

We compare our method with LorSLIM by looking at how they reconstruct the user-item matrix. We take ML100K as an example, whose density is 6.30% and the mean for those non-zero elements is 3.53. The reconstructed matrix from LorSLIM $\hat{X} = XW$ has a density of 13.61%, whose non-zero values have a mean of 0.046. For those 6.30% non-zero entries in X, $\hat{X}$ recovers 70.68% of them and their mean value is 0.0665. This demonstrates that lots of information is lost. On the contrary, our approach fully preserves the original information thanks to the constraint condition in our model. Our method recovers all zero values with a mean of 0.554, which is much higher than 0.046. This suggests that

our method recovers X better than LorSLIM. This may explain the superior performance of our method.

Parameter Tunning

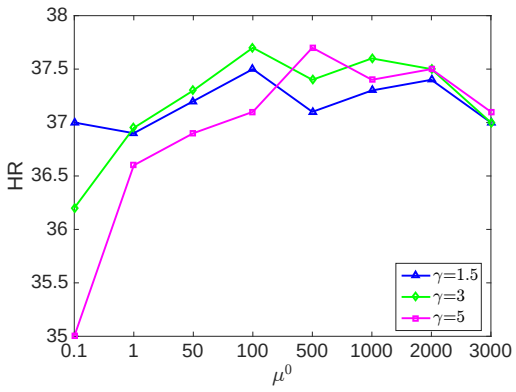


Figure 2: Influence of μ^0 and γ on HR for Delicious dataset.

Although our model is parameter-free, we introduce the auxiliary parameter μ during the optimization. In alternating direction method of multipliers (ADMM) (**?**), μ is fixed and it is not easy to choose an optimal value to balance the computational cost. Thus, a dynamical μ, increasing at a rate of γ, is preferred in real applications. $\gamma > 1$ controls the convergence speed. The larger γ is, the fewer iterations are to obtain the convergence, but meanwhile we may lose some precision. We show the effects of different initializations μ^0 and γ on HR on dataset Delicious in Figure 2. It illustrates that our experimental results are not sensitive to them, which is reasonable since they are auxiliary parameters controlling mainly the convergence speed. In contrast, LorSLIM needs to tune four parameters, which are time consuming and not easy to operate.

Efficiency Analysis

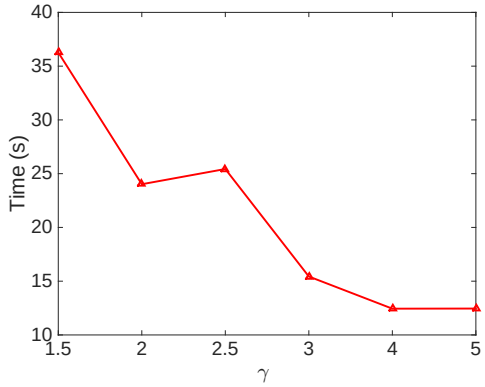


Figure 3: Influence of γ on time.

The time complexity of our algorithm is mainly from SVD. Exact SVD of a $m \times n$ matrix has a time complexity of $O(\min\{mn^2, m^2n\})$, in this paper we seek a low-rank matrix and thus only need a few principal singular vectors/values. Packages like PROPACK (**?**) can compute a rank k SVD with a cost of $O(min\{m^2k, n^2k\})$, which can be advantageous when $k \ll m, n$. In fact, our algorithm is much faster than LorSLIM. Among the six datasets, ML100K and lastfm datasets have the smallest and largest sizes, respectively. Our method needs 9s and 5080s, respectively, on these two datasets, while LorSLIM takes 617s and 32974s. The time is measured on the same machine with an Intel Xeon E3-1240 3.40GHz CPU that has 4 cores and 8GB memory, running Ubuntu and Matlab (R2014a). Furthermore, without losing too much accuracy, γ can speed up our algorithm considerably. This is verified by Figure 3, which shows the computational time of our method on Delicious with varying γ.

Conclusion

In this paper, we present a matrix completion based method for the Top-N recommendation problem. The proposed method recovers the user-item matrix by solving a rank minimization problem. To better approximate the rank, a non-convex function is applied. We conduct a comprehensive set of experiments on multiple datasets and compare its performance against that of other state-of-the-art Top-N recommendation algorithms. It turns out that our algorithm generates high quality recommendations, which improves the performance of the rest of methods considerably. This makes our approach usable in real-world scenarios.

Acknowledgements

This work is supported by US National Science Foundation Grants IIS 1218712. Q. Cheng is the corresponding author.

Tempora eos provident aperiam earum quibusdam sint perferendis optio non animi sed, nemo quam culpa nulla possimus rem distinctio asperiores quae eveniet