Robust Matrix Factorization for Collaborative Filtering in Recommender Systems

Christos G. Bampis, Cristian Rusu, Hazem Hajj, Alan C. Bovik

***Abstract*—Recently, matrix factorization has produced state-of-the-art results in recommender systems. However, given the typical sparsity of ratings, the often large problem scale, and the large number of free parameters that are often implied, developing robust and efficient models remains a challenge. Previous works rely on dense and/or sparse factor matrices to estimate unavailable user ratings. In this work we develop a new formulation for recommender systems that is based on projective non-negative matrix factorization, but relaxes the non-negativity constraint. Driven by a simple yet instructive intuition, the proposed formulation delivers promising and stable results that depend on a minimal number of parameters. Experiments that we conducted on two popular recommender system datasets demonstrate the efficiency and promise of our proposed method. We make available our code and datasets at https://github.com/christosbampis/PCMF_release.**

***Index Terms*—collaborative filtering, matrix factorization**

I. Introduction

Recommender systems [1] commonly deploy latent factor models [2], [3] that use available rating information to create a lower dimensional structure on which unavailable ratings are predicted. The underlying assumption is that the rating matrix can be represented by a small number of independent variables, called latent factors. Unlike memory-based techniques, latent factor models use vectors that describe the relationships between users and the latent factors. Items being rated are described by the amount of each latent factor they possess. Latent factor models have been shown to outperform their memory-based counterparts [2] and have thus become a main focus of current research on recommender systems.

Here we develop an improved latent factor model that relies on a minimization scheme that is based on projective non-negative matrix factorization [4]–[6]. However, we take a step further by relaxing the non-negativity constraint. As shown in the experimental section, our method delivers robust and promising results using a minimum number of parameters. Most state of the art methods may suffer from convergence issues and require a large set of free parameters; hence careful and time consuming tuning is required. By contrast, our adaptive learning scheme controls the learning rate, and converges to a high quality prediction result without the need for a regularization parameter.

A unique feature of the proposed method is that it demonstrates improved bias-variance tradeoff by allowing some loss of accuracy but much higher gain in robustness. The experiments demonstrate robustness while achieving accuracies comparable to previous methods. This robustness can be seen in two aspects: The first aspect is that the proposed method converges in terms of the objective function and the prediction performance, while previous methods diverge. The second aspect is the stability of the proposed method against different learning parameters, by being sensitive only to the number of latent factors, while other methods require delicate tuning of more learning parameters, which can adversely affect the prediction performance.

The rest of this paper is organized as follows. Section II discusses previous work on latent factor models, while Section III details the proposed method. Section IV presents the results of the experiments we conducted, including comparisons with other state of the art methods. Section V concludes with discussions of future work.

C. G. Bampis and A. C. Bovik are with the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, USA (e-mail: bampis@utexas.edu; bovik@ece.utexas.edu). C. Rusu is with the Institute for Digital Communications, University of Edinburgh, UK (c.rusu@ed.ac.uk). H. Hajj is with the Department of Electrical and Computer Engineering, American University of Beirut, Lebanon (hazem.hajj@aub.edu.lb).

II. Prior work on Latent Factor models

The use of latent factor models in recommender systems is based on the premise that the overall rating matrix can be represented by a small number of latent factors (independent variables). Under this low dimensionality assumption, a user is represented by a single vector that captures the degree of affinity that a user has for the different latent factors. Further, an item is also assigned to a feature vector, which describes its affinity to each latent factor that the item possesses. For example, in a movie recommendation setting, the users refer to human viewers, items correspond to movies of interest to users, and a possible latent factor can be related to the type of the movie (action, drama etc.) that is being rated. Therefore, the rating of user i on movie j can be modeled as a vector product between the ith user vector factor and the jth item vector factor:

$$R_{ij} = \langle U_i, V_j \rangle \tag{1}$$

where $\mathbf{R} = [R_{ij}]$ is a $m \times n$ rating matrix, m is the number of users, n is the number of items, U_i is the latent factor vector for user i and V_j is the latent factor vector for item j.

However, it has been established [2] that actual user preferences are characterized not only by this user-item interaction but also by a number of biases. As an example, a popular item, such as an award-winning movie, is likely to be rated more favorably by users. Further, a given user may inherently rate movies in an individual way, i.e. by assigning lower ratings on all items. These are two typical examples of item and user biases (denoted by b_j and b_i respectively), which are combined with the previous interaction term using an additive model:

$$R_{ij} = \mu_g + b_i + b_j + \langle U_i, V_j \rangle \tag{2}$$

where R_{ij} is the rating of user i of the movie j, b_i is the ith user's bias, b_j is the jth item's bias, and μ_g is the global mean of the available ratings. The $m \times n$ matrix $\mathbf{Y} = [Y_{ij}]$ represents the interaction between user latent factors and item latent factors and can be computed as follows:

$$Y_{ij} = R_{ij} - b_i - b_j - \mu_g. \tag{3}$$

The most common approach to latent factor models [2] computes the latent factor matrices $\mathbf{U}$ and $\mathbf{V}$ by using the following minimization scheme [2], [7]:

$$\min_{\mathbf{U},\mathbf{V}} \left\|\mathbf{Y} - \mathbf{A}(\mathbf{UV})\right\|_2^2 + \lambda_u \|\mathbf{U}\|_F^2 + \lambda_v \|\mathbf{V}\|_F^2 \tag{4}$$

where $\mathbf{U} = [U_{ij}]$ and $\mathbf{V} = [V_{ij}]$ are the $m \times r$ user and $r \times n$ item factor matrix respectively, r is the number of latent factors, λ_u and λ_v are regularization parameters, $\mathbf{A}$ is a $m \times m$ linear operator, and $\|\cdot\|_F$ denotes the Frobenius matrix norm. We assume that r is either given or is a free parameter that needs to be determined by grid search.

An alternative strategy was proposed in [8], where instead of assuming that both the user and item matrices are dense, the authors argue that the item matrix may instead often be sparse: items either possess a certain latent factor or not. They consequently proposed a formulation they call Blind Compressive Sensing framework for Collaborative Filtering (BCS-CF):

$$\min_{\mathbf{U},\mathbf{V}} \left\|\mathbf{Y} - \mathbf{A}(\mathbf{UV})\right\|_2^2 + \lambda_u \|\mathbf{U}\|_F^2 + \lambda_v \|\mathbf{V}\|_1 \tag{5}$$

Realization of this approach yields a small energy dense matrix $\mathbf{U}$ and enforces the sparseness of the item matrix $\mathbf{V}$. Building on this baseline model, an expanded method incorporating metadata was later developed [9].

A Majorization Minimization (MM) approach was used in [8] to compute the factor matrices $\mathbf{U}$ and $\mathbf{V}$. Two sub-problems are considered and solved in an alternating fashion. The first computes the dense user factor matrix given the item matrix i.e.

$$\mathbf{U}_{k+1} = \arg\min_{\mathbf{U}_k} \left\|\mathbf{Y} - \mathbf{A}(\mathbf{U}_k\mathbf{V}_k)\right\|_2^2 + \lambda_u \|\mathbf{U}_k\|_F^2 \tag{6}$$

Minimizing an underdetermined system of equations yields a least squares solution, which requires computing the pseudoinverse matrix and the inversion of a (usually) large matrix. The Majorization Minimization (MM) approach deals with large matrix inversions by finding a different minimizer to replace the original function to be minimized. Given $\mathbf{Y}$, $\mathbf{U}_k$ and $\mathbf{V}_k$ at some iteration k, (6) is solved as follows:

$$\mathbf{Z}_k = \mathbf{U}_k\mathbf{V}_k + \frac{1}{\beta}\mathbf{A}^\top\big(\mathbf{Y} - \mathbf{A}(\mathbf{U}_k\mathbf{V}_k)\big) \tag{7}$$

$$\mathbf{U}_{k+1} = \arg\min_{\mathbf{U}_k} \|\mathbf{Z}_k - \mathbf{U}_k\mathbf{V}_k\|_F^2 + \lambda_u \|\mathbf{U}_k\|_F^2 \tag{8}$$

where $\beta = \lambda_{\max}(\mathbf{A}^\top\mathbf{A})$ and $\mathbf{U}_k$ denotes the kth iteration of $\mathbf{U}$. Equation (8) can be solved using an efficient least squares approach. The second sub-problem is then solved to compute the sparse item matrix based on the computed user matrix [8].

Gogna *et al.* [8] compute an estimate $\hat{\mathbf{Y}} = [\hat{Y}_{ij}]$ of the interaction matrix, that is commingled with the underlying minimization scheme via $\hat{\mathbf{Y}} = \mathbf{UV}$, where $\mathbf{U}$ and $\mathbf{V}$ are computed via (5). Given a train and test pair and the estimated interaction values, (3) may be re-written:

$$\hat{R}_{ij} = \hat{Y}_{ij} + \mathbf{b}_u(i) + \mathbf{b}_m(j) + g \tag{9}$$

where $\hat{\mathbf{R}} = [\hat{R}_{ij}]$ is the matrix of estimated ratings. Then, by using an appropriate measure of error between $\mathbf{R}$ and $\hat{\mathbf{R}}$, the quality of the estimated ratings can be evaluated.

Another approach that has been taken is to combine the BCS-CF framework of (5) with an elastic term:

$$\min_{\mathbf{U},\mathbf{V}} \left\|\mathbf{Y} - \mathbf{A}(\mathbf{UV})\right\|_F^2 + \lambda_3 \|\mathbf{U}\|_F^2 + \lambda_1 \|\mathbf{V}\|_1 + \lambda_2 \|\mathbf{V}\|_F^2$$

where the free parameters λ_1, λ_2, λ_3 control the weighting of the regularizing terms. We will refer to this method as BCS-CF-EL and will use the publicly available code provided by the authors when making comparisons [8]. We will also compare against the metadata enriched version of BCS-CF, which uses both user and item metadata (BCS-CF-UI) [9].

III. Proposed Collaborative Filtering Formulation

An alternative approach that we propose here is to impose the projective constraint [4], [5] on the item matrix i.e. $\mathbf{V} = \mathbf{U}^\top\mathbf{Y}$, then re-cast the optimization as a quadratic minimization problem. One immediate advantage of this is that, instead of learning two different matrices $\mathbf{U}$ and $\mathbf{V}$, only a single matrix $\mathbf{U}$ with fewer parameters is learned. Unlike prior formulations, $\mathbf{U}$ no longer represents the user factor matrix, but is instead a user-item interaction matrix which is learned via a quadratic scheme.

Consider the problem of recovering a low rank matrix $\hat{\mathbf{Y}} = \mathbf{U}\mathbf{U}^\top\mathbf{Y}$ that satisfies:

$$\min \left\|\mathbf{A}(\mathbf{Y} - \hat{\mathbf{Y}})\right\|_F^2 \tag{10}$$

where the operator $\mathbf{A}(\cdot)$ ensures that only informative entries are used when calculating $\mathbf{U}$ and the reconstruction error. Our ultimate goal is to determine the optimal $\mathbf{U}$ that solves

$$\begin{aligned}\mathbf{U} &= \arg\min_{\mathbf{U}} \left\|\mathbf{A}(\mathbf{Y} - \mathbf{U}\mathbf{U}^\top\mathbf{Y})\right\|_F^2 \\ &= \arg\min_{\mathbf{U}} \left\|\mathbf{Y} - \mathbf{A}(\mathbf{U}\mathbf{U}^\top\mathbf{Y})\right\|_F^2\end{aligned} \tag{11}$$

Note that (11) can be related to (4) by setting $\mathbf{V} = \mathbf{U}^\top\mathbf{Y}$ and $\lambda_u = \lambda_v = 0$. Similar to [8], we solve an equivalent MM formulation:

$$\min_{\mathbf{U}} \left\|\mathbf{Z} - \mathbf{U}\mathbf{U}^\top\mathbf{Z}\right\|_F^2 \tag{12}$$

where we iteratively estimate $\mathbf{Z}_k$:

$$\mathbf{Z}_k = \mathbf{U}_k\mathbf{U}_k^\top\mathbf{Y} + \frac{1}{\beta}\mathbf{A}^\top\big(\mathbf{Y} - \mathbf{A}(\mathbf{U}_k\mathbf{U}_k^\top\mathbf{Y})\big) \tag{13}$$

Given $\mathbf{Z_k}$, we propose to solve (12) by using a multiplicative update rule (MUR) similar to the one proposed in [4]:

$$U_{ij} \leftarrow U_{ij} \frac{(\mathbf{Z}\mathbf{Z}^\top \mathbf{U})_{ij}}{(\mathbf{U}\mathbf{U}^\top \mathbf{Z}\mathbf{Z}^\top \mathbf{U})_{ij} + (\mathbf{Z}\mathbf{Z}^\top \mathbf{U}\mathbf{U}^\top \mathbf{U})_{ij}}. \tag{14}$$

The next step is to normalize $\mathbf{U}$ by its norm [4], [5]. Alternative MUR strategies were proposed in [10] by introducing a learning rate parameter η, which determines the learning rate of the update rule [6]: small values guarantee the monotonic decrease of the objective function by sacrificing the convergence speed. Consider the following MUR:

$$U_{ij} \leftarrow U_{ij} \left(\frac{(\mathbf{Z}\mathbf{Z}^\top \mathbf{U})_{ij}}{(\mathbf{U}\mathbf{U}^\top \mathbf{Z}\mathbf{Z}^\top \mathbf{U})_{ij} + (\mathbf{Z}\mathbf{Z}^\top \mathbf{U}\mathbf{U}^\top \mathbf{U})_{ij}} \right)^\eta \tag{15}$$

where $\eta = \frac{1}{3}$ has been used [10]. A similar approach was applied in [11] in an image segmentation application. However, (15) can not be applied directly, since the interaction values $\mathbf{Y}_{ij}$ and the corresponding $\mathbf{Z}_{ij}$ elements can take negative values. This observation is at the core of our method: unlike previous works where the data matrix consisted of non-negative values [11], e.g., pixel values, the MUR we develop here has to account for negative data values. Therefore, we propose to split the update rule as follows:

1) set

$$Q_{ij} = \frac{(\mathbf{Z}\mathbf{Z}^\top \mathbf{U})_{ij}}{(\mathbf{U}\mathbf{U}^\top \mathbf{Z}\mathbf{Z}^\top \mathbf{U})_{ij} + (\mathbf{Z}\mathbf{Z}^\top \mathbf{U}\mathbf{U}^\top \mathbf{U})_{ij}} \tag{16}$$

2) if $Q_{ij} \geq 0$: $U_{ij} \leftarrow U_{ij} Q_{ij}^\eta$, else $U_{ij} \leftarrow -U_{ij} |Q_{ij}|^\eta$

where $\mathbf{Q} = [Q_{ij}] \in \mathbb{R}^{m \times r}$. This step allows us to reduce large norms while learning $\mathbf{U}_k$. We fixed $\eta = \frac{1}{2}$; as shown in the experimental section, the proposed method is not sensitive to this parameter. The update rule (15) can be repeatedly applied until convergence is achieved or a predetermined number number of iterations occurs. In the proposed algorithm, the inner loop - minimizing (12) - is repeated at most $\max_2$ times, while the outer loop (to re-compute $\mathbf{Z}_k$ given $\mathbf{U}_k$) is repeated at most $\max_1$ times. The number of maximum iterations ($\max_{1,2}$) should be large enough to ensure convergence. We set $\max_1 = 50$ and $\max_2 = 20$. The initial matrix $\mathbf{U}_0$ is randomly generated although other initialization schemes could be deployed to advantage. We refer to the following proposed method as Projective-Constrained Matrix Factorization for Collaborative Filtering (PCMF-CF).

Algorithm for PCMF-CF

```
input: observed interaction values (train set)
initialize $\mathbf{U}_0$ to a random $m \times r$ matrix
$k \leftarrow 0$, $\eta \leftarrow \frac{1}{2}$
while $k \leq \max_1$ do
    compute $\mathbf{Z}_k$ using (13)
    $l \leftarrow 0$
    $\mathbf{U} \leftarrow \mathbf{U}_k$
    while $l \leq \max_2$ do
        compute $\mathbf{Q}$ using (16)
        for $i = 1 : m$ do
            for $j = 1 : n$ do
                if $Q_{ij} \geq 0$ then
                    $U_{ij} \leftarrow U_{ij} Q_{ij}^\eta$
                else
                    $U_{ij} \leftarrow -U_{ij} |Q_{ij}|^\eta$
                end
            end
        end
        $l \leftarrow l + 1$
    end
    $\mathbf{U}_{k+1} \leftarrow \mathbf{U}$
    $k \leftarrow k + 1$
end
$\hat{\mathbf{Y}} = \mathbf{U}_k \mathbf{U}_k^\top \mathbf{Y}$
output: unobserved interaction values (test set)
```

IV. Experimental results

We evaluated our proposed method on the MovieLens 100K and 1M datasets [12]. These publicly available datasets are extensively used in the recommender systems literature. The available ratings take values between 1 and 5. Unavailable ratings are assigned the value 0. The MovieLens 100K dataset consists of 10^5 ratings on 1682 movies assigned by 943 users (93.70% sparse), while the 1M dataset consists of 10^6 ratings on 3952 movies assigned by 6040 users (95.81% sparse). The following methods were tested for comparison: BCS-CF [8], BCS-CF-UI [9], and PCMF-CF (our proposed model). We were not able to find metadata for the 1M dataset, hence we do not report BCS-CF-UI results on the 1M dataset. Note that we only considered methods using baseline estimation since it has been demonstrated [9] that they yield improved results over methods that use raw rating information, such as BCD-NMF [13] or PMF [7]. We performed a 5-fold split on the dataset, by setting 80% of the ratings contained in $\mathbf{R}$ as the train set and the rest 20% as the test set. Given a particular train and test pair and the available ratings matrix $\mathbf{R} = [R_{ij}]$, let S be the set of available ratings i.e. $S = \{i, j : R_{ij} > 0\}$. The interaction values $\hat{Y}_{ij}$ were computed using (3), where μ_g is the global mean i.e. $\mu_g = \frac{1}{|S|} \sum_S R_{ij}$ and $|\cdot|$ denotes set cardinality. Then, by applying the corresponding matrix factorization method, the unavailable (test set) interaction values are estimated. Using (9), estimates of the unobserved (test set) ratings can then be computed from the interaction values, the global mean and the bias terms . We used the mean absolute error (MAE) and root mean squared error (RMSE) to compare the quality of prediction (QoP) between the different methods:

$$\text{MAE} = \frac{1}{mn} \sum_{ij} \left| R_{ij} - \hat{R}_{ij} \right|$$

$$\text{RMSE} = \sqrt{\frac{1}{mn} \sum_{ij} (R_{ij} - \hat{R}_{ij})^2}$$

First, we evaluated each method by picking their best performance over the number of latent factors (r) used by each method. We used the default values suggested by the authors of every compared method. Different initializations will produce different results for each method; hence we repeated each experiment 10 times and report the averages of each result. Tables I and II tabulate the obtained values of the error metrics

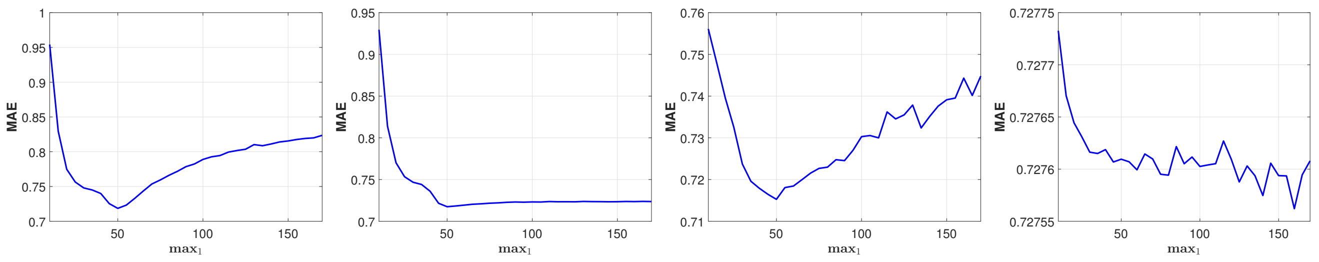

Fig. 1: Average MAE over 5 folds of different compared methods against number of iterations. Left to right: BCS-CF, BCS-CF-EL, BCS-CF-UI, PCMF-CF. Note that the y axis for PCMF-CF is significantly tighter compared to the other methods.

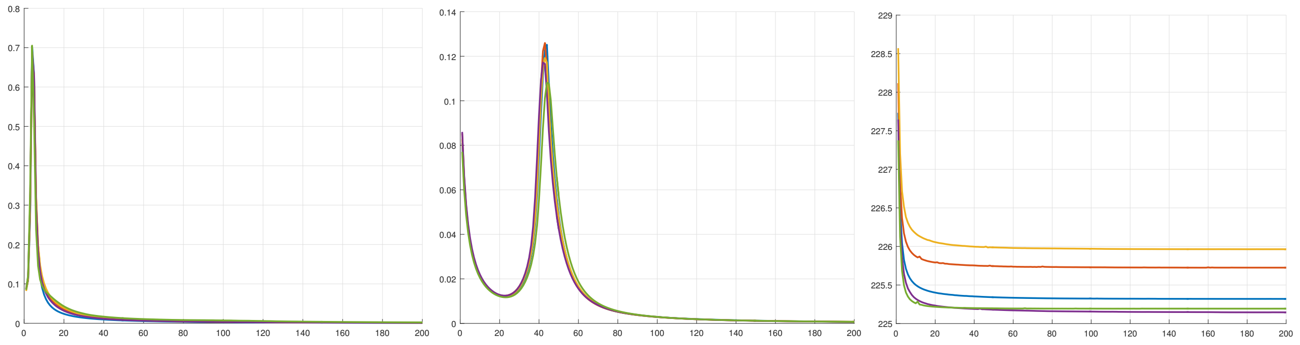

Fig. 2: Objective function evolution at each iteration. Left to right: BCS-CF, BCS-CF-EL, PCMF-CF. Each dataset fold is denoted by a different color.

for each model on both datasets. It may be seen that PCMF-CF delivers competitive performance but is inferior to BCS-CF on both databases.

TABLE I: Results on 100K dataset - 5 folds. The results are averaged across 10 experiments.

Method	MAE (↑)	RMSE (↑)	best r
BCS-CF [8]	0.7185	0.9182	64
BCS-CF-EL	0.7174	0.9165	60
BCS-CF-UI [9]	0.7166	0.9178	10
PCMF-CF	0.7276	0.9271	12

TABLE II: Results on 1M dataset - 5 folds. The results are averaged across 10 experiments.

Method	MAE (↑)	RMSE (↑)	best r
BCS-CF [8]	0.6776	0.8669	8
BCS-CF-EL	0.6773	0.8655	10
PCMF-CF	0.6927	0.8808	32

However, an important property of PCMF-CF is that it is a parameter-robust alternative to collaborative filtering. To investigate this claim, we examined the convergence properties of BCS-CF, BCS-CF-elastic and BCS-CF-UI, by varying the number of iterations ($\max_1$) for each matrix factorization scheme. As shown in Fig. 1, only BCS-CF-EL exhibits fairly stable behavior (in the MAE sense) as the number of iterations is increased. Overall, PCMF-CF delivers very robust behavior, since the MAE varies considerably less compared to the other methods.

Apart from studying the error convergence of each method, it is also important to examine convergence of the objective function. As shown in Fig. 2, only PCMF-CF monotonically decreased the objective function across different folds.

We also studied the sensitivity of PCMF-CF to the value of η. In our experiments, we found that values $\eta \in [0.1\ 0.6]$ produced stable results across all five folds in both datasets (see Fig. 3); values outside this range may deliver worse results. The extreme case of $\eta = 1$ leads to instabilities in the PCMF-CF computations. However, given that values $\eta \leq \frac{1}{2}$ were also used in very different applications [6], [10], [11], we believe that the η parameter can be fixed a priori. Naturally, reducing the number of tuned parameters is highly desirable. However, the BCS related methods require more tuning parameters which presents two challenges: first, tuning these parameters is time consuming, and second, the recommendation results may be adversely affected by these parameter variations. To demonstrate the effect of parameter tuning, Fig. 4 shows two cases where the BCS-CF-EL method delivered recommendation results which were quite sensitive to the choice of those parameters; hence careful tuning is required.

One shortcoming of PCMF-CF is its computational cost, given the underlying quadratic optimization. Table III presents the execution times of the different methods as the number of iterations was increased, for a given train/test pair, on both datasets. It may be seen that both the dataset size and the maximum number of iterations $\max_1$ increases the computational cost. However, it should be emphasized that the execution times in Table III assume that the best set of parameters for BCS-related methods is known a priori. In practice, carrying out a parameter grid search would induce much larger computational costs; while PCMF-CF is robust to parameter selection and is dependent only on a single

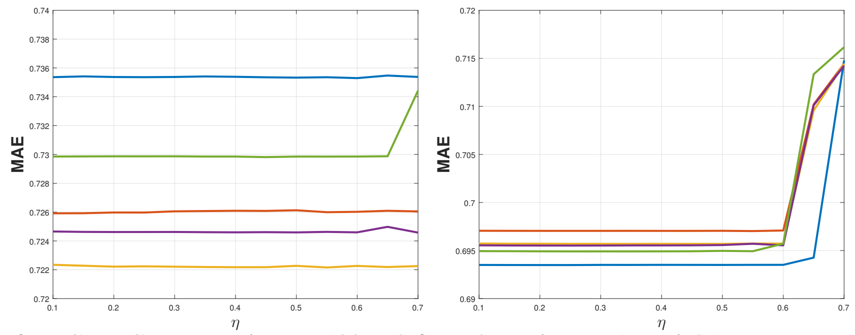

Fig. 3: MAE against η for PCMF-CF on MovieLens 100K (left) and MovieLens 1M (right). For $\eta \leq 0.6$, PCMF-CF delivered stable results.

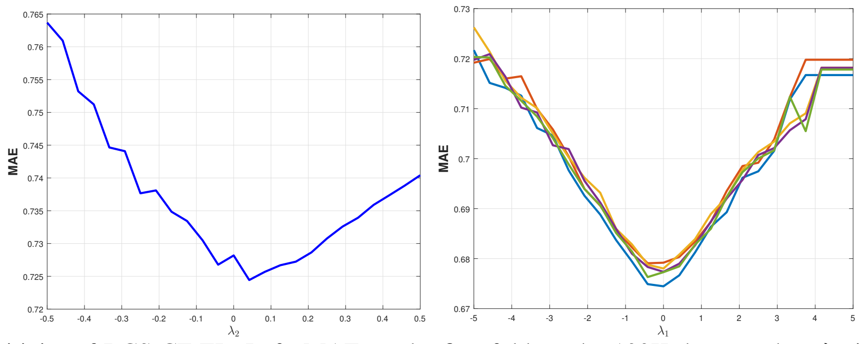

Fig. 4: Parameter sensitivity of BCS-CF-EL. Left: MAE on the first fold on the 100K dataset when λ_2 is varied. Right: MAE on all five folds on the 1M dataset when λ_1 is varied.

parameter (number of latent factors r). The stability of PCMF-CF makes up for the larger (per iteration) computational cost, hence PCMF-CF becomes a highly attractive alternative.

Overall, we found that PCMF-CF is a robust method in that it requires only a single parameter to be determined, converges empirically in terms of the objective function and its prediction performance is comparable to other methods that suffer from convergence issues and require manual tuning of a larger set of parameters which further influence the prediction performance.

TABLE III: Computational cost (in sec.) on both datasets over 50 and 200 iterations for all methods.

Database	100K		1M	
# iterations	50	200	50	200
BCS-CF [8]	3.3890	12.8671	48.9619	188.0133
BCS-CF-EL	3.1292	11.9565	44.3501	172.3313
BCS-CF-UI [9]	2.3499	8.8018	-	-
PCMF-CF	5.5029	26.3256	199.3489	869.8755

V. Conclusions

We developed a robust alternative to collaborative filtering methods based on projective-constrained matrix factorization. PCMF-CF delivered stable, promising results relative to other state of the art methods. Its unique feature is that it uses only one parameter, while other methods require many parameters. The proposed method could serve as a core framework for other methods that incorporate additional information such as user or item metadata. Our planned future work will be to integrate such additional information to further improve the quality of prediction of the collaborative filtering approach. Another future challenge would be to decrease the computational complexity of our proposed approach.

References

[1] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich, *Recommender Systems: an Introduction*. Cambridge University Press, 2010.
[2] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, no. 8, pp. 30–37, 2009.
[3] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowledge-Based Systems*, vol. 46, pp. 109–132, 2013.
[4] Z. Yuan and E. Oja, "Projective nonnegative matrix factorization for image compression and feature extraction," in *Image Analysis*, 2005, pp. 333–342.
[5] Z. Yang and E. Oja, "Linear and nonlinear projective nonnegative matrix factorization," *IEEE Trans. Neural Netw.*, vol. 21, no. 5, pp. 734–749, 2010.
[6] H. Zhang, Z. Yang, and E. Oja, "Adaptive multiplicative updates for quadratic nonnegative matrix factorization," *Neurocomputing*, vol. 134, pp. 206–213, 2014.
[7] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization." in *NIPS*, 2008, pp. 1257–1264.
[8] A. Gogna and A. Majumdar, "Blind compressive sensing framework for collaborative filtering," *arXiv preprint arXiv:1505.01621*, 2015.
[9] A. Gogna and A. Majumdar, "Blind compressive sensing formulation incorporating metadata for recommender system design," *APSIPA Transactions on Signal and Information Processing*, vol. 4, p. e2, 2015.
[10] Z. Yang and E. Oja, "Unified development of multiplicative algorithms for linear and quadratic nonnegative matrix factorization," *IEEE Trans. Neural Netw.*, vol. 22, no. 12, pp. 1878–1891, Dec. 2011.
[11] C. G. Bampis, P. Maragos, and A. C. Bovik, "Projective non-negative matrix factorization for unsupervised graph clustering," in *IEEE Int' l Conf. on Image Processing (ICIP)*, 2016, pp. 1255–1258.
[12] http://grouplens.org/datasets/movielens/.
[13] Y. Xu and W. Yin, "A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion," *SIAM Journal on imaging sciences*, vol. 6, no. 3, pp. 1758–1789, 2013.