Magnitude Bounded Matrix Factorisation for Recommender Systems

Shuai Jiang, Kan Li, and Richard Yi Da Xu

Abstract—Low rank matrix factorisation is often used in recommender systems as a way of extracting latent features. When dealing with large and sparse datasets, traditional recommendation algorithms face the problem of acquiring large, unrestrained, fluctuating values over predictions. Imposing bounding constraints has been proven an effective solution. However, existing bounding algorithms can only deal with one pair of fixed bounds, and are very time-consuming when applied on large-scale datasets. In this paper, we propose a novel algorithm named Magnitude Bounded Matrix Factorisation (MBMF), which allows different bounds for individual users/items and performs very quickly on large scale datasets. The key idea of our algorithm is to construct a model by constraining the magnitudes of each individual user/item feature vector. By converting coordinate system with radii set as the corresponding magnitudes, MBMF allows the above constrained optimisation problem to become an unconstrained one, which can be solved by unconstrained optimisation algorithms such as the stochastic gradient descent. We also explore an acceleration approach and the choice of magnitudes are given in detail as well. Experiments on synthetic and real datasets demonstrate that in most cases the proposed MBMF is superior over all existing algorithms in terms of accuracy and time complexity.

Index Terms—Magnitude bounded matrix factorisation, coordinate conversions, recommender systems.

1 Introduction

The prevalence of recommender systems has become evident over recent years. Amongst many of its algorithms, collaborative filtering (CF) has been one of the most widely used [1], [2], [3]. The two primary categories of CF are the neighbourhood methods and latent factor models. The former computes the relationships between users or items [4], [5], while the latter tries to explain the ratings by characterising both users and items on factors inferred from the ratings patterns [6], [7], which has become more popular in recent years. The most successful latent factor models used low rank matrix factorisation (MF) to obtain the feature vectors, in which the goal is to fill the unknown entries of the data matrix by calculating the inner product of factorised feature matrices [8], [9]. After obtaining the prediction values, recommendations on items can be made by ranking the values associated with the user in question.

Although existing MF-based CF algorithms have been proved effective for general recommender systems, such as Movielens [10] and Jester [11] datasets, when it comes down to datasets with high sparsity where many users only provide very few ratings, such as BookCrossing [12], Dating [13] and Netflix Prize [14] datasets, the predictions of missing entries are often unstable and out of range [15].

We demonstrate this effect through an example of a synthetic recommender system in Fig. 1. In this example, we randomly generated three sparse matrices with the same size of 500×500 and range of $[1, 10]$ but different densities: 0.8, 0.2 and 0.05 (denoted by blue dots in Fig. 1 (a), (c) and (e)). Then we ran the MF method [15] on each sparse matrix to get predicted values (denoted by red dots in Fig. 1 (b), (d), (f) and (g)). Meanwhile, we recorded the maximal and minimal predicted values (denoted by $\max(\hat{V})$ and $\min(\hat{V})$), and calculated the maximal and average standard deviations (denoted by $\max(\sigma)$ and ave(σ)), which are presented in TABLE 1. As seen from the results, the increase in missing entries led towards not only more out-of-bounds predicted values for more red dots appeared, but also higher prediction fluctuations as both the $\max(\sigma)$ and ave(σ) became greater. Furthermore, according to Fig. 1 (f) and (g) and the last two rows in TABLE 1, repeating the factorisation algorithm and using the average predicted values can significantly leverage the bounding issues. However, this will cause an increase in computational cost, especially when large data sets are involved.

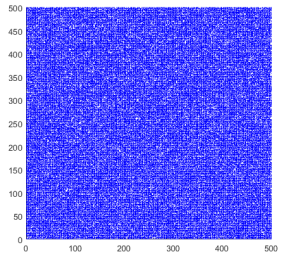
Bounding predicted values and reducing fluctuation on predictions can help improve recommendation accuracy on large and sparse datasets. Since many recommendation systems have a single pre-defined range for all entries, one straightforward way to reduce fluctuation over predictions is to limit the product of the factorised matrices, to make them fall within a pre-defined range. There are two types of algorithms in literature seeking to address this bounding problem: one is to project the products that are out of range to the boundaries [16], [17], and the other one is to impose strong penalties in the objective function if the products are out of range [18]. Projecting (or mapping) methods require in range or out of range checks on each of the product entries obtained by the multiplication of factorised matrices, which causes the issue of high computation time especially on large scale dataset (see the Section of Complexity Analysis for more details), while the performance of the penalty methods highly depends on the setting of their penalty coefficients.

Apart from the above mentioned recommendation systems with pre-defined single range, there are many other ones where data does not share a single range, i.e. users/items' data are not in the same magnitude. For instance, in "user-location check-

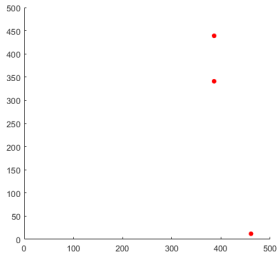
S. Jiang is with the School of Computer Science, Beijing Institute of Technology (BIT), Beijing, China, and the Faculty of Engineering and Information Technology, University of Technology Sydney (UTS), Australia (email: jiangshuai@bit.edu.cn; shuai.jiang-1@student.uts.edu.au).

K. Li is with the School of Computer Science, Beijing Institute of Technology (BIT), Beijing, China (email: likan@bit.edu.cn).

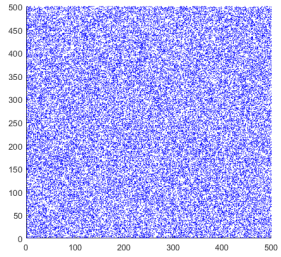
R. Y. D. Xu is with the Faculty of Engineering and Information Technology, University of Technology Sydney (UTS), Australia (email: yida.xu@uts.edu.au).



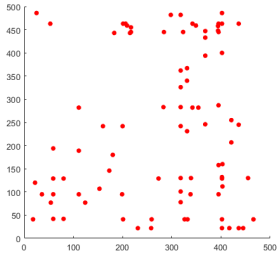
(a) sparse matrix density $= 0.8$.



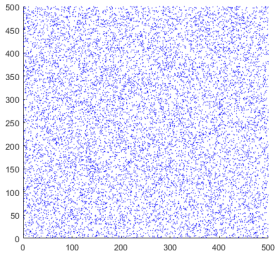
(b) average of 10 repeats, only 3 predicted values (0.001%) are out of bounds.



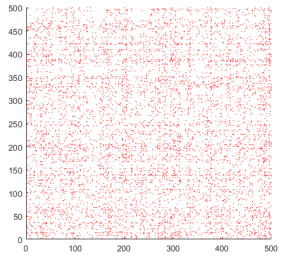
(c) sparse matrix density $= 0.2$.



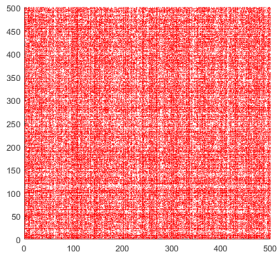
(d) average of 10 repeats, several predicted values (0.034%) are out of bounds.



(e) sparse matrix density $= 0.05$.



(f) average of 10 repeats, many predicted values (2.781%) are out of bounds.



(g) a single run, a large proportion of predicted values (34.69%) are out of bounds.

Fig. 1: Predicted values obtained from Matrix Factorisation (MF) algorithm [15] with respect to different densities of factorised matrices. Synthetic data are denoted by blue dots in (a), (c) and (e) and the corresponding out-of-bounds (oob) predicted values are denoted by red dots in (b), (d), (f) and (g). Note that the red dots in (b) and (d) are depicted 20 times bigger than those in other subfigures to ensure they are clearly visible.

TABLE 1: Statistics of Predicted Values in the Synthetic Example

density	error	#oob	$\max(\hat{V})$	$\min(\hat{V})$	$\max(\delta)$	ave(δ)
0.8	7.25	3	10.69	-0.38	2.08	0.51
0.2	5.25	86	11.67	-1.84	8.98	1.68
0.05	0.29	6953	18.88	-8.67	22.76	5.36
0.05(1)	0.27	86720	62.72	-45.00	-	-

in frequency" dataset [19], the matrix contains the frequency counts of users appearing in different locations, where the scales of frequencies vary from user to user and location to location. Another example is the Ali Mob dataset, where some people browsed and bought a lot of products while others might only do a few. In these cases, the above two bounding methods may fail because they can only address datasets with single range: imposing a single pair of bounds, for example, using the maximum and minimum of all known values, will have very little effect on bounding most of the entries, as they are all far from the end points of this range. From another aspect, even when dealing with datasets which have a single pre-defined range, such as movie ratings in Movielens which are between 1 and 5, bounding the entries in the product matrix into different and more concise ranges (e.g. related to the corresponding user or item) is also likely to gain better recommendations because users/items can have different preferences which can be reflected by their own bounds. We demonstrate the benefits and improvements of individual concise ranges compared to fixed pre-defined ranges in our Experiments Section.

In this paper, we propose a novel model called, Magnitude Bounded Matrix Factorisation (MBMF), which is a bounding model designed for large and sparse recommender systems. As with other bounding approaches, MBMF deals with the fluctuation of recommendations by bounding the entries in the product matrix into pre-defined ranges. The main difference is that MBMF can be applied to datasets with or without pre-defined single bounds, and allows the product entries to have individually different bounds. In the proposed MBMF, we first devise an optimisation algorithm where individual range constraints are placed onto the magnitude of latent feature vectors. This allows MBMF to deal with any datasets for which the magnitudes can be set using context-specific information. Secondly, in order to solve this constrained optimisation problem, we convert the original constrained problem into an unconstrained one by using cartesian to spherical coordinate conversions such that the problem becomes that of the unconstrained task. These conversions make MBMF unique from all other existing bounding algorithms. It neither projects the product values to the boundaries nor imposes penalty terms to constrain the factorisation, and it can be solved efficiently by unconstrained optimisation solvers or algorithms, for example, the widely used stochastic gradient descent (SGD) approach.

Although the unconstrained optimisation task seems easy to be solved, the representation complexity brought by the spherical coordinate makes it hard to compute the derivatives. Therefore, we propose an elegant acceleration approach for MBMF based on the discovery of patterns of derivatives which involves tensor construction and multiplication. Further, we also explore in detail the choice of magnitudes according to different types of data. In order

to demonstrate the efficiency of the proposed MBMF, we conduct sufficient numerical experiments to analyse its time complexity with comparative algorithms. In the experiments, both synthetic and real-world datasets are used to verify the effectiveness of MBMF, and the results show that MBMF is superior to existing bounding algorithms regarding recommendation accuracies in most cases.

The main contributions of this paper are:

1. We propose a novel algorithm named MBMF, a bounding model which deals with the fluctuation of recommendations on large and sparse datasets with or without pre-defined single bounds. Different from existing studies, MBMF bounds all entries using individual ranges. To the best of our knowledge, MBMF is the first bounding model which allows different bounds for each entry based on MF framework for recommender systems.

2. The methodology used in MBMF, magnitude constraints and coordinate conversions, is totally different from all existing bounding algorithms which are either projecting product values to boundaries or imposing penalty constraints. Furthermore, with the help of a proposed acceleration approach which mainly uses tensor construction and multiplication, the unconstrained MBMF can be solved both effectively and efficiently.

3. We also propose a MBMF variant, Non-negative MBMF, to deal with non-negative data and bound the predicted values to be within a non-negative range determined by magnitude constrains. For the original MBMF, we further distinguish Fixed MBMF and Centred MBMF from it based on the centring approach and magnitude constraints it uses. Fixed MBMF can work on datasets with pre-defined bounds where magnitudes are set identical, while Centred MBMF can bound predicted values using a newly proposed centring approach with respect to magnitude constraints instead of the conventional pre-defined range.

4. The proposed MBMF is the fastest bounding algorithm against the state-of-art bounding algorithms. On very large synthetic datasets (e.g. $200,000$ by $500,000$), MBMF is the only bounding algorithm which has comparable efficiency against classic matrix factorisation algorithms (MF and Non-negative Matrix Factorisation (NMF)).

5. We introduce a feasible way to incorporate historical data into our proposed model, which considers both the global influence and the individual effect. Experiments on three real world recommendation systems show that MBMF is capable for different types of datasets and can achieve the best performance in most cases.

The rest of this paper is organised as follows: in the next section, we review related literature to our work. The Method section contains the details of our algorithm, followed by the Experiments section in which we evaluate our algorithm as well as the Conclusions.

2 Related Work

For recommender systems, MF algorithms are mostly used in the latent factor models, where the features of users/items are represented by the factorised matrices. Many variations have been proposed based on the MF framework, with different additional information, such as user/item bias [1], [15], meta information collected from users and items [6], and temporal dynamics [20], [21].

As for studies on the bounds of product matrix, Bounded Matrix Factorisation (BMF) proposed by [16] was the first one focusing on limitations of ranges over predictive values in recommender systems. The key thought of BMF is the following updating formula:

$$\boldsymbol{W}_{ab} = \begin{cases} \max(\mathbf{l}), & if \quad \boldsymbol{W}^*_{ab} < \max(\mathbf{l}) \\ \min(\mathbf{u}), & if \quad \boldsymbol{W}^*_{ab} > \min(\mathbf{u}) \\ \boldsymbol{W}^*_{ab}, & otherwise, \end{cases} \tag{1}$$

where $\mathbf{l}$ is the lower bound vector, $\mathbf{u}$ is the upper bound vector, $\boldsymbol{W}_{ab}$ is the element of the left factorised matrix in question and $\boldsymbol{W}^*_{ab}$ is the optimal solution derived from the objective function by letting the derivatives vanish. Under the above updating rules, BMF is proved to achieve better performance than unbounded methods. However, BMF has been proven not to converge to stationary points [17]. Apart from the convergence, its core idea is similar to a truncated approach: instead of directly truncating entries to the pre-defined bounds, BMF moves the updating values to the inner bounds (the minimum of the upper bound vector and the maximum of the lower bound vector) if they are outside of the given range. As for the time complexity, BMF has to compute for each entry a complementary matrix whose size is close to the observation matrix, which can be time consuming when big datasets are involved.

Recently, an improved bounded matrix completion method was proposed in [17] based on Alternating Direction of Multiplier Method (ADMM) framework (so is called BMC-ADMM). The optimisation problem in their work is defined as:

$$\begin{aligned} \min_{\boldsymbol{X},\boldsymbol{P},\boldsymbol{T}} & \frac{1}{2}\|\boldsymbol{Z} * (\boldsymbol{V} - \boldsymbol{X})\|_F^2 + \lambda\|\boldsymbol{P}\|_* \\ s.t. & \boldsymbol{X} + \boldsymbol{E} = \boldsymbol{P}, \neg\boldsymbol{Z} * \boldsymbol{X} = \mathbf{0}, \boldsymbol{Z} * \boldsymbol{E} = 0, \\ & \boldsymbol{P} = \boldsymbol{T}, r_{\min} \leqslant \boldsymbol{T} \leqslant r_{\max}, \end{aligned} \tag{2}$$

where $\|\cdot\|_F$ is the Frobenius norm, $\boldsymbol{Z}$ is a mask matrix, $*$ denotes the element-wise multiplication, $\|\cdot\|_*$ is the nuclear norm commonly used to approximate the matrix rank, and $\neg$ denotes logical NOT. The authors aimed to improve the theoretical foundation in BMF, and to reformulate the model so that it can take advantage of the ADMM framework (split the loss, regularisation and constraints) which guarantees their algorithm to converge to the optimal solution. However, the paper does not differ fundamentally from BMF as far as bounding the predicted entries is concerned: during each iteration, BMF-ADMM checks the product of the two factorised matrices and projects any entries which are outside of the range to the boundaries. This checking mechanism is also the most time consuming step in the model which was reported in their work. Another potential obstacle for BMC-ADMM to be applied on very large datasets is the fact that, the updating procedure involves the calculation of partial singular value decomposition (SVD) for a non-sparse matrix having the same size of the observation matrix. Although the authors used several tricks to accelerate each of the updating steps, its efficiency still falls behind of the classic methods and our proposed MBMF (see Complexity Analysis for more details).

An alternative way to bound the entries is the so-called, Bounded-SVD [18]. In their work, the authors imposed a penalty regularisation term to penalise the updating values which tend to fall outside of the given range:

$$\begin{aligned} F = \sum_{ij} & (\boldsymbol{V}_{ij} - \boldsymbol{W}_{i:}\boldsymbol{H}_{:j}) + \\ & \lambda(e^{\alpha(\boldsymbol{W}_{i:}\boldsymbol{H}_{:j} - r_{\max})} + e^{\alpha(r_{\min} - \boldsymbol{W}_{i:}\boldsymbol{H}_{:j})}), \end{aligned} \tag{3}$$

where λ and α are parameters controlling the strength of penalties, $r_{\max}$ and $r_{\min}$ are the given upper and lower bounds. It also introduces another parameter η as the learning rate when updating the factor matrices. This regularisation method is complex, and cannot guarantee all product values are bounded within the given range simultaneously: therefore, the objective function only considers a small set of entries instead of bounding all of them. Besides, from our experiments, Bounded-SVD is very sensitive to its penalty coefficients: when they are set to be large (such as 10), all product values become very small, while when they are set to have small values (such as 0.01), most of the entries cannot be bounded within the pre-defined range. This algorithm also faces a time complexity issue when applied on large datasets.

MBMF is different from all the above existing bounding algorithms. It neither shifts the product values to fall between the boundaries, nor imposes any penalties in the objective function. MBMF introduces the magnitude constraints which guarantee the product of factorised matrices to be within ranges determined by these magnitude constraints. With the help of coordinates conversions, it manages to model the problem with an unconstrained optimisation task which can be solved efficiently by the well-known SGD method.

3 METHOD

In this section, we introduce a novel matrix factorisation algorithm called Magnitude Bounded Matrix Factorisation (MBMF) to deal with the prediction fluctuation for users/item in recommender systems. We first define the magnitude bounded matrix factorisation problem, followed by conversions from the constrained task to an unconstrained one. After that, we apply a stochastic gradient descent method to solve the unconstrained task.

3.1 Magnitude Bounded Matrix Factorisation Problem

Given an observation matrix $\boldsymbol{V} \in \mathbb{R}^{N\times M}$, the original low rank matrix factorisation task is to find two factor matrices $\boldsymbol{W} \in \mathbb{R}^{N\times K}$ and $\boldsymbol{H} \in \mathbb{R}^{K\times M}$ which satisfy the approximation $\boldsymbol{V} \approx \boldsymbol{W}\boldsymbol{H}$, where K is commonly referred to as the latent dimension and usually set a smaller value than N and M. According to the definition of matrix multiplication and the geometric interpretation of vector inner product in Euclidean space, each element of $\boldsymbol{V}$, say $\boldsymbol{V}_{ij}$, is approximated by the inner product of the i^{th} row of $\boldsymbol{W}$ and the j^{th} column of $\boldsymbol{H}$:

$$\boldsymbol{V}_{ij} \approx \hat{\boldsymbol{V}}_{ij} = \boldsymbol{W}_{i:} \cdot \boldsymbol{H}_{:j} = |\boldsymbol{W}_{i:}||\boldsymbol{H}_{:j}| \cos\alpha_{ij} \tag{4}$$

where $\boldsymbol{W}_{i:}$ denotes the i^{th} row of $\boldsymbol{W}$ (so does $\boldsymbol{H}_{:j}$), $|\cdot|$ denotes the vector magnitude and α_{ij} is the angle between vectors $\boldsymbol{W}_{i:}$ and $\boldsymbol{H}_{:j}$. As $\cos\alpha_{ij} \in [-1, 1]$, the prediction entry $\hat{\boldsymbol{V}}_{ij}$ is bounded by the product of magnitudes:

$$-|\boldsymbol{W}_{i:}||\boldsymbol{H}_{:j}| \leqslant \hat{\boldsymbol{V}}_{ij} \leqslant |\boldsymbol{W}_{i:}||\boldsymbol{H}_{:j}|. \tag{5}$$

Definition 1 (Magnitude Bounded Matrix Factorisation Problem)**.** *Given an observation matrix $\boldsymbol{V} \in \mathbb{R}^{N\times M}$ and two constant positive magnitude vectors $\boldsymbol{r}^{\boldsymbol{W}} \in \mathbb{R}_+^N$ and $\boldsymbol{r}^{\boldsymbol{H}} \in \mathbb{R}_+^M$, magnitude bounded matrix factorisation aims to find two factor matrices $\boldsymbol{W}$ and $\boldsymbol{H}$ such that*

$$\begin{aligned} &\boldsymbol{V} \approx \hat{\boldsymbol{V}} = \boldsymbol{W}\boldsymbol{H} \\ s.t.\quad &|\boldsymbol{W}_{i:}| = \boldsymbol{r}_i^{\boldsymbol{W}}, \forall i = 1, 2, \cdots, N, \\ &|\boldsymbol{H}_{:j}| = \boldsymbol{r}_j^{\boldsymbol{H}}, \forall j = 1, 2, \cdots, M. \end{aligned} \tag{6}$$

According to the above definition, the prediction entry $\hat{\boldsymbol{V}}_{ij}$ is now bounded by the product of its two corresponding magnitudes:

$$-\boldsymbol{r}_i^{\boldsymbol{W}}\boldsymbol{r}_j^{\boldsymbol{H}} \leqslant \hat{\boldsymbol{V}}_{ij} \leqslant \boldsymbol{r}_i^{\boldsymbol{W}}\boldsymbol{r}_j^{\boldsymbol{H}}. \tag{7}$$

In fact, the two magnitude vectors can form a rank one range matrix $\boldsymbol{R} = \boldsymbol{r}^{\boldsymbol{W}}(\boldsymbol{r}^{\boldsymbol{H}})^T \in \mathbb{R}^{N\times M}$. Thus the prediction entry is bounded by its corresponding entry in the range matrix $\boldsymbol{R}$:

$$-\boldsymbol{R}_{ij} \leqslant \hat{\boldsymbol{V}}_{ij} \leqslant \boldsymbol{R}_{ij}. \tag{8}$$

Notice that all of the entries in the product matrix can be bounded within individually different ranges which are determined by the give constant magnitude vectors, though the ranges are not totally independent.

According to the definition of MBMF, no matter how few observations a user/item has, its corresponding predictions are always bounded as long as the magnitude vectors are set properly. We assume that the magnitude vectors can reflect a relatively stable preference of each user or item for the current factorisation task. So the magnitudes can also be regarded as user/item preferences in recommender systems.

3.2 Conversions from Constrained to Unconstrained Task

The magnitude bounded matrix factorisation defined in Eq.(6) is equal to a constrained optimisation problem:

$$\begin{aligned} &\min_{\boldsymbol{W},\boldsymbol{H}} \|\boldsymbol{Z} * (\boldsymbol{V} - \boldsymbol{W}\boldsymbol{H})\|_F^2 \\ s.t.\quad &|\boldsymbol{W}_{i:}| = \boldsymbol{r}_i^{\boldsymbol{W}}, \forall i = 1, 2, \cdots, N, \\ &|\boldsymbol{H}_{:j}| = \boldsymbol{r}_j^{\boldsymbol{H}}, \forall j = 1, 2, \cdots, M. \end{aligned} \tag{9}$$

where $\boldsymbol{Z}$ is an indicator matrix in which ones denote the existing observations and zeros otherwise. It is not straightforward to work it out because of the magnitude constraints.

Let us now imagine the geometric representation of the magnitude constraints first. Given a two-dimensional variable $\boldsymbol{x} = [x_1, x_2]^T$ in Euclidean space, setting its magnitude to a constant $|\boldsymbol{x}| = r$ limits the possible space of $\boldsymbol{x}$ onto a circle with radius r. By introducing an angle variable ϕ, the vector can then be converted into the Polar coordinate system as $\boldsymbol{x} = [r\cos\phi, r\sin\phi]^T$, which embeds the magnitude constraint into the Polar representation such that calculations on the converted variable do not need to consider the magnitude constraint. Similarly, for higher dimensional variables, the magnitude constraints limit their possible space onto a sphere or hypersphere in the Euclidean system, and conversion to spherical coordinates will incorporate the magnitudes into their new representations, which "removes" the magnitude constraints.

Thus we introduce two angle matrices $\boldsymbol{\Phi} \in \mathbb{R}^{N\times(K-1)}$ and $\boldsymbol{\Theta} \in \mathbb{R}^{(K-1)\times M}$ to help represent each row of $\boldsymbol{W}$ and each column of $\boldsymbol{H}$ in spherical space with the radii set as corresponding magnitudes. A n-dimensional variable $\boldsymbol{x} = [x_1, x_2, \cdots, x_n]^T$ can be represented by a radial coordinate r and $n-1$ angular coordinates $\boldsymbol{\phi} = [\phi_1, \phi_2, \cdots, \phi_{n-1}]^T$ where $\phi_1, \phi_2, \cdots, \phi_{n-2} \in [0, \pi]$ and $\phi_{n-1} \in [0, 2\pi)$:

$$\begin{aligned} x_1 &= r\cos\phi_1 \\ x_2 &= r\sin\phi_1\cos\phi_2 \\ &\vdots \\ x_{n-1} &= r\sin\phi_1\cdots\sin\phi_{n-2}\cos\phi_{n-1} \\ x_n &= r\sin\phi_1\cdots\sin\phi_{n-2}\sin\phi_{n-1}. \end{aligned} \tag{10}$$

Note that the constraints on these angles are to ensure the uniqueness of conversion from cartesian to spherical, however, MBMF does the conversion the other way around, i.e. from spherical to cartesian which does not have any angle constraints.

Define auxiliary vectors of functions $\boldsymbol{s}(\boldsymbol{\phi}) = [s_1(\boldsymbol{\phi}), s_2(\boldsymbol{\phi}), \cdots, s_n(\boldsymbol{\phi})]^T$ and $\boldsymbol{c}(\boldsymbol{\phi}) = [c_1(\boldsymbol{\phi}), c_2(\boldsymbol{\phi}), \cdots, c_n(\boldsymbol{\phi})]^T$ such that

$$s_i(\boldsymbol{\phi}) = \begin{cases} 1, & i = 1 \\ \prod_{j=1}^{i-1} \sin\phi_j, & i > 1 \end{cases}, \tag{11}$$

$$c_i(\boldsymbol{\phi}) = \begin{cases} \cos\phi_i, & i < n \\ 1, & i = n \end{cases}, \tag{12}$$

then we have

$$x_i = r s_i(\boldsymbol{\phi}) c_i(\boldsymbol{\phi}), \tag{13}$$

thus

$$\boldsymbol{x} = r\boldsymbol{s}(\boldsymbol{\phi}) * \boldsymbol{c}(\boldsymbol{\phi}). \tag{14}$$

Then both of the factor matrices can be converted into the spherical space:

$$\boldsymbol{W}_{i:} = \boldsymbol{r}_i^{\boldsymbol{W}} \boldsymbol{s}(\boldsymbol{\Phi}_{i:}) * \boldsymbol{c}(\boldsymbol{\Phi}_{i:}), \tag{15}$$

$$\boldsymbol{H}_{:j} = \boldsymbol{r}_j^{\boldsymbol{H}} \boldsymbol{s}(\boldsymbol{\Theta}_{:j}) * \boldsymbol{c}(\boldsymbol{\Theta}_{:j}), \tag{16}$$

Define auxiliary matrices of functions $\boldsymbol{S}(\boldsymbol{\Phi})$, $\boldsymbol{S}(\boldsymbol{\Theta})$ such that

$$\boldsymbol{S}_{i:}(\boldsymbol{\Phi}) = \boldsymbol{s}(\boldsymbol{\Phi}_{i:}), \quad \boldsymbol{S}_{:j}(\boldsymbol{\Theta}) = \boldsymbol{s}(\boldsymbol{\Theta}_{:j}), \tag{17}$$

and $\boldsymbol{C}(\boldsymbol{\Phi})$, $\boldsymbol{C}(\boldsymbol{\Theta})$ such that

$$\boldsymbol{C}_{i:}(\boldsymbol{\Phi}) = \boldsymbol{c}(\boldsymbol{\Phi}_{i:}), \quad \boldsymbol{C}_{:j}(\boldsymbol{\Theta}) = \boldsymbol{c}(\boldsymbol{\Theta}_{:j}), \tag{18}$$

then we have

$$\boldsymbol{W} = \boldsymbol{r}^{\boldsymbol{W}} \odot \boldsymbol{S}(\boldsymbol{\Phi}) * \boldsymbol{C}(\boldsymbol{\Phi}), \tag{19}$$

$$\boldsymbol{H} = \boldsymbol{r}^{\boldsymbol{H}} \odot \boldsymbol{S}(\boldsymbol{\Theta}) * \boldsymbol{C}(\boldsymbol{\Theta}), \tag{20}$$

where $\odot$ denotes the Khatri-Rao product. Notice that we omit some transpose signs in the formulas from Eq. (15) to (20) to make them concise and consistent. We refer to this kind of conversion as *Spherical2Cartesian*.

Thus, the constrained optimisation problem in Eq. (9) is equivalent to the following unconstrained optimisation task:

$$\min_{\boldsymbol{\Phi},\boldsymbol{\Theta}} \|\boldsymbol{Z} * (\boldsymbol{V} - (\boldsymbol{r}^{\boldsymbol{W}} \odot \boldsymbol{S}(\boldsymbol{\Phi}) * \boldsymbol{C}(\boldsymbol{\Phi}))(\boldsymbol{r}^{\boldsymbol{H}} \odot \boldsymbol{S}(\boldsymbol{\Theta}) * \boldsymbol{C}(\boldsymbol{\Theta})))\|_F^2. \tag{21}$$

3.3 Solving Unconstrained Optimisation

The objective functions defined in Eq. (9) and Eq. (21) both are non-convex. Fortunately, we found that it can be solved efficiently by the SGD method [22]. According to Eq.(21), the objective function is defined as

$$\mathcal{F} = \|\boldsymbol{Z} * (\boldsymbol{V} - (\boldsymbol{r}^{\boldsymbol{W}} \odot \boldsymbol{S}(\boldsymbol{\Phi}) * \boldsymbol{C}(\boldsymbol{\Phi}))(\boldsymbol{r}^{\boldsymbol{H}} \odot \boldsymbol{S}(\boldsymbol{\Theta}) * \boldsymbol{C}(\boldsymbol{\Theta})))\|_F^2. \tag{22}$$

First, we calculate the partial derivatives of $\mathcal{F}$ with respect to a single element, say $\boldsymbol{\Phi}_{ab}$. Here we regard $\boldsymbol{W}$ as a function of $\boldsymbol{\Phi}$ (so does $\boldsymbol{H}$ to $\boldsymbol{\Theta}$). Thus by chain rule, we have:

$$\frac{\partial \mathcal{F}}{\partial \boldsymbol{\Phi}_{ab}} = \sum_{j=1}^{K} \frac{\partial \mathcal{F}}{\partial \boldsymbol{W}_{aj}} \frac{\partial \boldsymbol{W}_{aj}}{\partial \boldsymbol{\Phi}_{ab}} = \frac{\partial \mathcal{F}}{\partial \boldsymbol{W}_{a:}} (\frac{\partial \boldsymbol{W}_{a:}}{\partial \boldsymbol{\Phi}_{ab}})^T, \tag{23}$$

where the summation is due to that $\boldsymbol{\Phi}_{ab}$ participants in the representation of all elements of vector $\boldsymbol{W}_{a:}$.

The partial derivative of $\mathcal{F}$ with respect to $\boldsymbol{W}_{aj}$ is

$$\frac{\partial \mathcal{F}}{\partial \boldsymbol{W}_{aj}} = 2((\boldsymbol{Z} * (\boldsymbol{W}\boldsymbol{H}))\boldsymbol{H}^T - (\boldsymbol{Z} * \boldsymbol{V})\boldsymbol{H}^T)_{aj}. \tag{24}$$

The partial derivative of $\boldsymbol{W}_{aj}$ with respect to $\boldsymbol{\Phi}_{ab}$ is

$$\frac{\partial \boldsymbol{W}_{aj}}{\partial \boldsymbol{\Phi}_{ab}} = \boldsymbol{r}_a^{\boldsymbol{W}} \cdot \begin{cases} 0, & j < b \\ -\prod_{p=1}^{j} \sin\boldsymbol{\Phi}_{ap}, & j = b \\ \cos\boldsymbol{\Phi}_{ab} \cos\boldsymbol{\Phi}_{aj} \prod_{p=1;p\neq b}^{j-1} \sin\boldsymbol{\Phi}_{ap}, & b < j < K \\ \cos\boldsymbol{\Phi}_{ab} \prod_{p=1;p\neq b}^{j-1} \sin\boldsymbol{\Phi}_{ap}, & j = K \end{cases}. \tag{25}$$

Define auxiliary vectors of functions $\boldsymbol{s}(\boldsymbol{\Phi}_{a:}, b) = [s_1(\boldsymbol{\Phi}_{a:}, b), \cdots, s_K(\boldsymbol{\Phi}_{a:}, b)]^T$ and $\boldsymbol{c}(\boldsymbol{\Phi}_{a:}, b) = [c_1(\boldsymbol{\Phi}_{a:}, b), \cdots, c_K(\boldsymbol{\Phi}_{a:}, b)]^T$ such that

$$s_j(\boldsymbol{\Phi}_{a:}, b) = \prod_{p=1}^{b-1} \sin\boldsymbol{\Phi}_{ap} \cdot \begin{cases} 0, & j < b \\ -\sin\boldsymbol{\Phi}_{aj}, & j = b \\ 1, & j = b+1 \\ \prod_{p=b+1}^{j-1} \sin\boldsymbol{\Phi}_{aj}, & b+1 < j \leqslant K \end{cases}, \tag{26}$$

$$c_j(\boldsymbol{\Phi}_{a:}, b) = \begin{cases} 0, & j < b \\ 1, & j = b \\ \cos\boldsymbol{\Phi}_{aj} \cos\boldsymbol{\Phi}_{ab}, & b < j < K \\ \cos\boldsymbol{\Phi}_{ab}, & j = K \end{cases}, \tag{27}$$

then we have

$$\frac{\partial \boldsymbol{W}_{a:}}{\partial \boldsymbol{\Phi}_{ab}} = (\boldsymbol{r}_a^{\boldsymbol{W}} \boldsymbol{s}(\boldsymbol{\Phi}_{a:}, b) * \boldsymbol{c}(\boldsymbol{\Phi}_{a:}, b))^T, \tag{28}$$

thus

$$\begin{aligned} \frac{\partial \mathcal{F}}{\partial \boldsymbol{\Phi}_{ab}} =& 2((\boldsymbol{Z} * (\boldsymbol{W}\boldsymbol{H}))\boldsymbol{H}^T - (\boldsymbol{Z} * \boldsymbol{V})\boldsymbol{H}^T)_{a:} \\ & \cdot (\boldsymbol{r}_a^{\boldsymbol{W}} \boldsymbol{s}(\boldsymbol{\Phi}_{a:}, b) * \boldsymbol{c}(\boldsymbol{\Phi}_{a:}, b)). \end{aligned} \tag{29}$$

Similarly the derivative of $\mathcal{F}$ with respective to $\boldsymbol{\Theta}_{ab}$ is

$$\begin{aligned} \frac{\partial \mathcal{F}}{\partial \boldsymbol{\Theta}_{ab}} =& 2((\boldsymbol{W}^T(\boldsymbol{Z} * (\boldsymbol{W}\boldsymbol{H})) - \boldsymbol{W}^T(\boldsymbol{Z} * \boldsymbol{V}))_{:b})^T \\ & \cdot (\boldsymbol{r}_b^{\boldsymbol{H}} \boldsymbol{s}(\boldsymbol{\Theta}_{:b}, a) * \boldsymbol{c}(\boldsymbol{\Theta}_{:b}, a)). \end{aligned} \tag{30}$$

So the updating rules of $\boldsymbol{\Phi}$ and $\boldsymbol{\Theta}$ using stochastic gradient descent are

$$\begin{aligned} \boldsymbol{\Phi}_{ab} \leftarrow \boldsymbol{\Phi}_{ab} - \lambda^{\boldsymbol{\Phi}} & ((\boldsymbol{Z} * (\boldsymbol{W}\boldsymbol{H}))\boldsymbol{H}^T - (\boldsymbol{Z} * \boldsymbol{V})\boldsymbol{H}^T)_{a:} \\ & \cdot (\boldsymbol{r}_a^{\boldsymbol{W}} \boldsymbol{s}(\boldsymbol{\Phi}_{a:}, b) * \boldsymbol{c}(\boldsymbol{\Phi}_{a:}, b)), \end{aligned} \tag{31}$$

$$\begin{aligned} \boldsymbol{\Theta}_{ab} \leftarrow \boldsymbol{\Theta}_{ab} - \lambda^{\boldsymbol{\Theta}} & ((\boldsymbol{W}^T(\boldsymbol{Z} * (\boldsymbol{W}\boldsymbol{H})) - \boldsymbol{W}^T(\boldsymbol{Z} * \boldsymbol{V}))_{:b})^T \\ & \cdot (\boldsymbol{r}_b^{\boldsymbol{H}} \boldsymbol{s}(\boldsymbol{\Theta}_{:b}, a) * \boldsymbol{c}(\boldsymbol{\Theta}_{:b}, a)), \end{aligned} \tag{32}$$

where $\lambda^{\boldsymbol{\Phi}}$ and $\lambda^{\boldsymbol{\Theta}}$ are the learning rate parameters.

3.4 Acceleration for MBMF

According to our experiments, it is inefficient to individually update each element of $\boldsymbol{\Phi}$ and $\boldsymbol{\Theta}$ by Eq. (31) and (32). A feasible solution is to update the whole matrices at the same time as long as the derivatives of $\mathcal{F}$ with respect to $\boldsymbol{\Phi}$ and $\boldsymbol{\Theta}$ can be computed directly.

Referring to Eq. (23), we know that the derivative of $\mathcal{F}$ with respect to a certain element is a result of a dot product. That is, the derivative of $\mathcal{F}$ with respect to $\boldsymbol{\Phi}$ is

$$D^{\boldsymbol{\Phi}} = \frac{\partial \mathcal{F}}{\partial \boldsymbol{\Phi}} = \begin{bmatrix} \frac{\partial \mathcal{F}}{\partial \boldsymbol{W}_{1:}}(\frac{\partial \boldsymbol{W}_{1:}}{\partial \boldsymbol{\Phi}_{11}})^T & \cdots & \frac{\partial \mathcal{F}}{\partial \boldsymbol{W}_{1:}}(\frac{\partial \boldsymbol{W}_{1:}}{\partial \boldsymbol{\Phi}_{1(K-1)}})^T \\ \frac{\partial \mathcal{F}}{\partial \boldsymbol{W}_{2:}}(\frac{\partial \boldsymbol{W}_{2:}}{\partial \boldsymbol{\Phi}_{21}})^T & \cdots & \frac{\partial \mathcal{F}}{\partial \boldsymbol{W}_{2:}}(\frac{\partial \boldsymbol{W}_{2:}}{\partial \boldsymbol{\Phi}_{2(K-1)}})^T \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathcal{F}}{\partial \boldsymbol{W}_{N:}}(\frac{\partial \boldsymbol{W}_{N:}}{\partial \boldsymbol{\Phi}_{n1}})^T & \cdots & \frac{\partial \mathcal{F}}{\partial \boldsymbol{W}_{N:}}(\frac{\partial \boldsymbol{W}_{N:}}{\partial \boldsymbol{\Phi}_{n(K-1)}})^T \end{bmatrix}. \tag{33}$$

Notice that the k^{th} column of $D^{\boldsymbol{\Phi}}$ is a vector of dot products of two matrices—$\frac{\partial \mathcal{F}}{\partial \boldsymbol{W}}$ and $\frac{\partial \boldsymbol{W}}{\partial \boldsymbol{\Phi}_{:k}}$ on the row dimension. Thus we can compute $D^{\boldsymbol{\Phi}}$ in the following way:

1) calculate the derivative of $\boldsymbol{W}$ with respect to $\boldsymbol{\Phi}$, which is a $N \times K \times (K-1)$ three-way tensor (indeed it is a four-way tensor with a singleton dimension);

2) duplicate $\frac{\partial \mathcal{F}}{\partial \boldsymbol{W}}$, a $N \times K$ matrix $(K-1)$ times to form the other $N \times K \times (K-1)$ three-way tensor;

3) $D^{\boldsymbol{\Phi}}$ is then the resultant matrix of dot products of the two tensors on the second dimension (dot products of corresponding mode-2/row fibres).

The tensor in 2) is straightforward but not in 1). According to Eq. (25), the derivative of $\boldsymbol{W}$ with respect to the a^{th} row of $\boldsymbol{\Phi}$ is a $(K-1) \times K$ matrix which is nearly upper triangular, and the derivative of $\boldsymbol{W}$ with respect to the b^{th} column of $\boldsymbol{\Phi}$ is a $N \times K$ matrix where each row shares the same format. Thus, we can construct the tensor in 1) by calculating the derivatives of $\boldsymbol{W}$ with respect to each column of $\boldsymbol{\Phi}$ and integrate them together.

In the algorithm of MBMF, the calculation of the derivative of $\boldsymbol{W}$ with respect to $\boldsymbol{\Phi}$ is referred as *GradWwrtPhi*.

The derivative of $\mathcal{F}$ with respect to $\boldsymbol{\Theta}$ can be obtained in a similar way, and the calculation of the derivative of $\boldsymbol{H}$ with respect to $\boldsymbol{\Theta}$ is referred as *GradHwrtTheta*. Now the updating rules of $\boldsymbol{\Phi}$ and $\boldsymbol{\Theta}$ become

$$\boldsymbol{\Phi} \leftarrow \boldsymbol{\Phi} - \lambda^{\boldsymbol{\Phi}} D^{\boldsymbol{\Phi}}, \tag{34}$$

$$\boldsymbol{\Theta} \leftarrow \boldsymbol{\Theta} - \lambda^{\boldsymbol{\Theta}} D^{\boldsymbol{\Theta}}. \tag{35}$$

According to the conversion from the spherical to cartesian coordinate systems (see *Spherical2Cartesian*), $\boldsymbol{W}$ and $\boldsymbol{H}$ then can be easily updated with the renewed $\boldsymbol{\Phi}$ and $\boldsymbol{\Theta}$.

3.5 MBMF Algorithm and Its Variant

MBMF can be extended to versions catering for different applicant scenarios. In this section, apart from the original MBMF algorithm demonstrated in the previous section, we introduce a MBMF variant with additional non-negative constraints, followed by the discussion on parameter settings.

3.5.1 Original MBMF

The complete algorithm of the original MBMF is shown in Algorithm 1.

Algorithm 1 Magnitude Bounded Matrix Factorisation (MBMF)

Input: $\boldsymbol{V} \in \mathbb{R}^{N \times M}$ (factorising matrix),
$\boldsymbol{Z} \in \mathbb{R}^{N \times M}$ (indicator matrix),
$K \in \mathbb{N}^+$ (latent dimension),
$\boldsymbol{r}^{\boldsymbol{W}}, \boldsymbol{r}^{\boldsymbol{H}}$ (magnitudes for $\boldsymbol{W}$ and $\boldsymbol{H}$)
$\lambda_{\boldsymbol{\Phi}} \in \mathbb{R}^+, \lambda_{\boldsymbol{\Theta}} \in \mathbb{R}^+$ (learning coefficients)
Output: $\boldsymbol{W} \in \mathbb{R}^{N \times K}$ (the left factorised matrix),
$\boldsymbol{H} \in \mathbb{R}^{K \times M}$ (the right factorised matrix)

Initialise $\boldsymbol{\Phi}$ and $\boldsymbol{\Theta}$ as random angle matrices
$\boldsymbol{W} \leftarrow$ *Spherical2Cartesian*$(\boldsymbol{\Phi}, \boldsymbol{r}^{\boldsymbol{W}})$
$\boldsymbol{H} \leftarrow$ *Spherical2Cartesian*$(\boldsymbol{\Theta}, \boldsymbol{r}^{\boldsymbol{H}})$
while Terminal conditions not satisfied **do**
  GradW $\leftarrow (\boldsymbol{Z} * (\boldsymbol{W}\boldsymbol{H}))\boldsymbol{H}^T - (\boldsymbol{Z} * \boldsymbol{V})\boldsymbol{H}^T$
  GradW $\leftarrow$ duplicate(GradW, $K-1$)
  GradPhi $\leftarrow$ *GradWwrtPhi*$(\boldsymbol{\Phi}, \boldsymbol{r}^{\boldsymbol{W}})$
  $\boldsymbol{\Phi} \leftarrow \boldsymbol{\Phi} - \lambda^{\boldsymbol{\Phi}} \cdot$ dot(GradW, GradPhi, 2)
  GradH $\leftarrow \boldsymbol{W}^T(\boldsymbol{Z} * (\boldsymbol{W}\boldsymbol{H})) - \boldsymbol{W}^T(\boldsymbol{Z} * \boldsymbol{V})$
  GradH $\leftarrow$ duplicate(GradH, $K-1$)
  GradTheta $\leftarrow$ *GradHwrtTheta*$(\boldsymbol{\Theta}, \boldsymbol{r}^{\boldsymbol{H}})$
  $\boldsymbol{\Theta} \leftarrow \boldsymbol{\Theta} - \lambda^{\boldsymbol{\Theta}} \cdot$ dot(GradH, GradTheta, 1)
  $\boldsymbol{W} \leftarrow$ *Spherical2Cartesian*$(\boldsymbol{\Phi}, \boldsymbol{r}^{\boldsymbol{W}})$
  $\boldsymbol{H} \leftarrow$ *Spherical2Cartesian*$(\boldsymbol{\Theta}, \boldsymbol{r}^{\boldsymbol{H}})$
end while

3.5.2 Non-negative MBMF

MBMF can be easily modified to deal with non-negative data with additionally imposed non-negative constraints. We define non-negative MBMF problem as

$$\begin{aligned} \boldsymbol{V} &\approx \hat{\boldsymbol{V}} = \boldsymbol{W}\boldsymbol{H} \\ s.t. \quad |\boldsymbol{W}_{i:}| &= \boldsymbol{r}_i^{\boldsymbol{W}}, \forall i = 1, 2, \cdots, N, \\ |\boldsymbol{H}_{:j}| &= \boldsymbol{r}_j^{\boldsymbol{H}}, \forall j = 1, 2, \cdots, M, \\ \boldsymbol{W} &\geqslant 0, \quad \boldsymbol{H} \geqslant 0. \end{aligned} \tag{36}$$

Because of the non-negativity in both factor matrices, $\cos \alpha_{ij} > 0$ for the angle α_{ij} between any feature vectors $\boldsymbol{W}_{i:}$ and $\boldsymbol{H}_{:j}$. Thus, we have

$$0 \leqslant \hat{\boldsymbol{V}}_{ij} \leqslant \boldsymbol{r}_i^{\boldsymbol{W}} \boldsymbol{r}_j^{\boldsymbol{H}}. \tag{37}$$

We can solve the above problem by combining the coordinate conversions used in MBMF and projected gradient descent (PGD). PGD is a variation of SGD. It updates variables according to SGD formulas and projects updated results which fail to satisfy imposed constraints to the constraint boundary. After the conversion from cartesian to spherical coordinate system, the magnitude constraints are removed and only the non-negative constraints are left. Therefore, we can update Φ and Θ according to Eq. (34) and (35), convert them back to $\boldsymbol{W}$ and $\boldsymbol{H}$ by Spherical2Cartesian and then simply project all negative values in $\boldsymbol{W}$ and $\boldsymbol{H}$ to 0. We refer to MBMF with additional non-negative constraints as Non-negative MBMF (MBMF-N) which is shown in Algorithm 2.

3.5.3 Parameter Settings

The parameters of MBMF are the learning coefficients $\lambda_{\boldsymbol{\Phi}}$ and $\lambda_{\boldsymbol{\Theta}}$ which are introduced by the SGD method. Among literature about SGD, the learning coefficients or the step sizes can be set in three different ways: (1) set the steps the value that produce the best performance after several tries; (2) change the steps dynamically during the updating procedure by checking the change of the value of objective function: if the function value decreases, make

Algorithm 2 Non-negative Magnitude Bounded Matrix Factorisation (MBMF-N)

Input: $\boldsymbol{V} \in \mathbb{R}_{\geqslant 0}^{N\times M}$ (factorising matrix),
$\boldsymbol{Z} \in \mathbb{R}^{N\times M}$ (indicator matrix),
$K \in \mathbb{N}^{+}$ (latent dimension),
$\boldsymbol{r^W}, \boldsymbol{r^H}$ (magnitudes for $\boldsymbol{W}$ and $\boldsymbol{H}$)
$\lambda_{\boldsymbol{\Phi}} \in \mathbb{R}^{+}, \lambda_{\boldsymbol{\Theta}} \in \mathbb{R}^{+}$ (learning coefficients)
Output: $\boldsymbol{W} \in \mathbb{R}_{\geqslant 0}^{N\times K}$ (the left factorised matrix),
$\boldsymbol{H} \in \mathbb{R}_{\geqslant 0}^{K\times M}$ (the right factorised matrix)

Initialise $\boldsymbol{\Phi}$ and $\boldsymbol{\Theta}$ as random angle matrices
$\boldsymbol{W} \leftarrow$ *Spherical2Cartesian*$(\boldsymbol{\Phi}, \boldsymbol{r^W})$
$\boldsymbol{H} \leftarrow$ *Spherical2Cartesian*$(\boldsymbol{\Theta}, \boldsymbol{r^H})$
while Terminal conditions not satisfied **do**
  GradW $\leftarrow (\boldsymbol{Z} * (\boldsymbol{WH}))\boldsymbol{H}^T - (\boldsymbol{Z} * \boldsymbol{V})\boldsymbol{H}^T$
  GradW $\leftarrow$ duplicate(GradW, $K-1$)
  GradPhi $\leftarrow$ *GradWwrtPhi*$(\boldsymbol{\Phi}, \boldsymbol{r^W})$
  $\boldsymbol{\Phi} \leftarrow \boldsymbol{\Phi} - \lambda^{\boldsymbol{\Phi}} \cdot$ dot(GradW, GradPhi, 2)
  GradH $\leftarrow \boldsymbol{W}^T(\boldsymbol{Z} * (\boldsymbol{WH})) - \boldsymbol{W}^T(\boldsymbol{Z} * \boldsymbol{V})$
  GradH $\leftarrow$ duplicate(GradH, $K-1$)
  GradTheta $\leftarrow$ *GradHwrtTheta*$(\boldsymbol{\Theta}, \boldsymbol{r^H})$
  $\boldsymbol{\Theta} \leftarrow \boldsymbol{\Theta} - \lambda^{\boldsymbol{\Theta}} \cdot$ dot(GradH, GradTheta, 1)
  $\boldsymbol{W} \leftarrow$ *Spherical2Cartesian*$(\boldsymbol{\Phi}, \boldsymbol{r^W})$
  $\boldsymbol{H} \leftarrow$ *Spherical2Cartesian*$(\boldsymbol{\Theta}, \boldsymbol{r^H})$
  $\boldsymbol{W}(\boldsymbol{W} < 0) \leftarrow 0, \quad \boldsymbol{H}(\boldsymbol{H} < 0) \leftarrow 0$
end while

the steps a bit bigger (e.g. multiplied by 1.1), otherwise shrink them (e.g. divided by 2) and roll back; (3) apply a line search method (e.g. bisection search or backtracking search with Wolfe conditions) to determine more reasonable steps. We used the dynamic setting for parameters in our implementation, and the initial values for both parameters are set at 0.1.

3.6 Preliminaries of Data

3.6.1 Data Centring

Eq. (6) defines MBMF as an algorithm bounding each entry of the product matrix in a centrosymmetric range with respect to the origin point zero. It requires the input matrix to be centred. To centre a data matrix with a pair of pre-defined bounds $[r_{\text{min}}, r_{\text{max}}]$, one only needs to shift all the observed values by

$$\boldsymbol{V}_{ij}^C = \boldsymbol{V}_{ij} - \frac{r_{\text{min}} + r_{\text{max}}}{2}, \forall i, j, \boldsymbol{Z}_{ij} = 1. \tag{38}$$

For example, movie ratings in Netflix are one of $\{1, 2, 3, 4, 5\}$, and the centred ratings should be $\{-2, -1, 0, 1, 2\}$ correspondingly. In this case, one can simply set the magnitudes as any values satisfying the following equation:

$$\boldsymbol{r}_i^{\boldsymbol{W}} \boldsymbol{r}_j^{\boldsymbol{H}} = \frac{r_{\text{max}} - r_{\text{min}}}{2}, \forall i, j. \tag{39}$$

The easiest way is to set $\boldsymbol{r}_i^{\boldsymbol{W}} = \boldsymbol{r}_j^{\boldsymbol{H}} = \sqrt{\frac{r_{\text{max}} - r_{\text{min}}}{2}}$. For example, when applying MBMF on Netflix dataset, after centring the observed matrix by Eq. (38) we can set all the magnitudes to be $\sqrt{2}$. The entries in the product matrix will be naturally bounded within $[-2, 2]$. We refer to MBMF using pre-defined bounds and setting identical magnitudes as Fixed MBMF (MBMF-F).

However, since MBMF can bound each entry to be within different individual bounds, the centring approach for MBMF should also reflect and utilise this feature. Thus, we propose a new centring approach with respect to the individual bounds determined by the magnitude constraints:

$$\boldsymbol{V}_{ij}^C = \boldsymbol{V}_{ij} - \min(\boldsymbol{V}) - \boldsymbol{r}_i^{\boldsymbol{W}} \boldsymbol{r}_j^{\boldsymbol{H}}, \forall i, j, \boldsymbol{Z}_{ij} = 1. \tag{40}$$

Also take the movie ratings in Netflix as an example where $\min(\boldsymbol{V}) = 1$. If the magnitude constraints $\boldsymbol{r}_i^{\boldsymbol{W}} \boldsymbol{r}_j^{\boldsymbol{H}} = 2.5$, then $\boldsymbol{V}_{ij}$ will be mapped from $\{1, 2, 3, 4, 5\}$ to $\{-2.5, -1.5, -0.5, 0.5, 1.5\}$. Note that there might be contradictions when centring data in this way. If $\boldsymbol{r}_i^{\boldsymbol{W}} \boldsymbol{r}_j^{\boldsymbol{H}} = 1.5$ and $\boldsymbol{V}_{ij} = 5$, Eq. (40) gives $\boldsymbol{V}_{ij}^C = 2.5$ which is out of the magnitude bounds $[-1.5, 1.5]$. It happens when $\boldsymbol{V}_{ij} > 2\boldsymbol{r}_i^{\boldsymbol{W}} \boldsymbol{r}_j^{\boldsymbol{H}} + \min(\boldsymbol{V})$ and one needs to modify the magnitude constraints. Otherwise, the corresponding approximation error will produce an unresolvable gap (in the above example, the gap is 1 because the best approximation for $\boldsymbol{V}_{ij}^C$ is 1.5). We refer to MBMF using this centring approach and magnitudes from historical data (see next section) as Centred MBMF (MBMF-C).

By using the newly proposed centring approach, MBMF-C does not requires the awareness of any pre-defined systematic bounds once the magnitude constraints are properly set. In the Experiments Section, we can see that MBMF-C can work on not only datasets with pre-defined ranges, such as Jester ($[-10, 10]$) and BookCrossing ($[1, 10]$), but also datasets which do not have such pre-defined ranges like Ali Mob.

3.6.2 Choice of Magnitudes

The magnitude constraints imported in MBMF can be set according to Eq. (39) if pre-defined bounds $[r_{\text{min}}, r_{\text{max}}]$ are available. When the bounds are unavailable or to fully utilise the advantages of MBMF, the magnitudes can be obtained in other ways, such as from historical records, user/item profiles and/or social connections. Among them, using historical records is relatively feasible and practical since most of the real world recommender systems collect data in time streams. Here, we introduce what we have done in our implementations to obtain good magnitudes based on historical data.

Assume that $\boldsymbol{V}$ is the current factorising matrix and $\boldsymbol{V'}$ is the corresponding historical matrix. Consider the recommendation for user i. A possible way to calculate the magnitude of this user is

$$\boldsymbol{r}_i^{\boldsymbol{W}} = \sqrt{\mathbb{E}(\boldsymbol{V}'_{i:}) + \sigma(\boldsymbol{V}'_{i:})}, \tag{41}$$

where $\mathbb{E}(\cdot)$ is the mean value and $\sigma(\cdot)$ is the standard deviation. Note that this equation requires the historical matrix to be non-negative, because if there are negative values, the mean cannot reflect the degree of goodness. Although this magnitude calculation is based on the individual average rating and allows possible variations as well, there might be cases in which a user has very few historical records, and applying Eq. (41) to calculate the magnitude in these cases can cause over-fitting issues. For example, if user i only has rated a movie 1 out of 5 in the past, Eq. (41) gives $\boldsymbol{r}_i^{\boldsymbol{W}} = 1$ since the mean is 1 and the standard deviation is 0, which means this user will be super strict for all of its future ratings. Thus, we introduce the global influence for individuals to mitigate such issues:

$$\boldsymbol{r}_i^{\boldsymbol{W}} = \omega_i \sqrt{\mathbb{E}(\boldsymbol{V}'_{i:}) + \sigma(\boldsymbol{V}'_{i:})} + (1 - \omega_i) \sqrt{\mathbb{E}(\boldsymbol{V'}) + \sigma(\boldsymbol{V'})}, \tag{42}$$

TABLE 2: Running Time (s) on Different Scales of Matrix.

N	M	K	density	BMF	BMC-ADMM	Bounded-SVD	MBMF-N
20	50	10	1	**0.1497**	2.2388	0.2631	0.3018
100	100	10	1	**0.5007**	8.8932	2.6718	0.7857
200	500	10	1	5.7100	4.4400	23.291	**3.0845**
1,000	1,000	10	1	313.75	**14.908**	201.71	18.293
2,000	5,000	10	1	4,753.4	128.73	2,101.3	**91.020**
Avg.				1,014.7	31.841	465.84	**22.697**
100,000	100,000	10	0.001	-	34,025	-	**5,728.3**
200,000	500,000	10	0.0002	-	442,250	-	**12,251**
Avg.				-	238,138	-	**8989.7**

where ω_i is the weight of how much impact in the magnitude is from the user itself. In our implementation, we use the following formula to determine the value of ω_i:

$$\omega_i = \begin{cases} 0, & \text{user } i \text{ has no historical data} \\ \min(\sum \boldsymbol{Z}'_{i:}/(\rho M), 1), & \text{otherwise} \end{cases} \tag{43}$$

where $\boldsymbol{Z}'$ is the indicator matrix for historical data, $\rho \in (0, 1]$ is a consistent percentage denoting how much historical data a user needs to completely represent its own preferences, and M is the number of items. More specifically, if a user has no records (i.e. $\omega_i = 0$), MBMF uses the global impact from all other users to define its magnitude. If a user has rated more than ρ of all the items in the past (i.e. $\omega_i = 1$), MBMF will fully use its historical ratings to determine its magnitude; the more historical data a user has, the more weight will be imposed on the impact from its own, and vice versa.

For our proposed MBMF-N, we can directly use Eq. (42) to compute the magnitudes required. For MBMF-C, the magnitudes $\boldsymbol{r}^C$ have a relationship with those of MBMF-N ($\boldsymbol{r}^N$):

$$\boldsymbol{r}^C = \boldsymbol{r}^N / \sqrt{2} \tag{44}$$

which provides a feasible approach to obtain magnitudes for MBMF-C.

3.7 Complexity Analysis

We analyse the computation complexity of the proposed MBMF algorithm both theoretically and numerically.

For each iteration, MBMF first calculates the partial derivatives of $\mathcal{F}$ with respect to $\boldsymbol{W}$ and $\boldsymbol{H}$, which costs $2NMK + 2(N + M)K^2$ addition and $2NMK + 2(N + M)K^2 + (N + M)K$ multiplication operations; then it computes the partial derivatives of $\boldsymbol{W}$ and $\boldsymbol{H}$ with respect to $\boldsymbol{\Phi}$ and $\boldsymbol{\Theta}$, and requires $2(N + M)(K - 1)$ trigonometric operations plus $2(N + M)(K - 2)(K - 1)$ times multiplication; it also operates $2(N + M)(K - 2)$ multiplication to transform $\boldsymbol{\Phi}$ and $\boldsymbol{\Theta}$ to $\boldsymbol{W}$ and $\boldsymbol{H}$. Thus the overall complexity of MBMF is $O(NMK)$, which is the same as the traditional matrix factorisation approach.

We conduct experiments to show the time consumed of MBMF on 7 different scales of matrices against other bounding algorithms BMF, BMC-ADMM and Bounded-SVD. The matrix scale varies from 10^3 to 10^{11}. The experiments are divided into two groups. One is conducted on small and full matrices with all 7 algorithms (repeated 5 times), and the other is conducted on large and sparse matrices (run once) without BMF and Bounded-SVD since they are extremely time consuming. All complexity experiments are conducted on a computer with i7-6900K CPU, 16G memory and Windows 10. The average results are presented in TABLE 2.

The results show that, (i) BMF and Bounded-SVD are quite fast when the matrix is small, but their running time increases when the matrix size becomes larger. Since the third case study, they start to fall far behind to the other bounding algorithms, and for the group of large scale matrices, they are even incapable of converging in a reasonable time. (ii) BMF-ADMM is not the fastest algorithm on small matrices, but it can still be applied to large scale instances, despite its long running time. (iii) MBMF-N is the overall fastest algorithm not only on small matrices, but also on large scale ones. This demonstrates that MBMF is the most efficient bounding algorithm and has great advantages when applied on large-scale recommender systems.

4 Experiments

We explore prediction variation of algorithms on synthetic datasets and compare the recommendation performance against related algorithms on several real datasets including Jester [11], BookCrossing [12], and Tianchi Mobile Record datasets. The statistics of these datasets after preprocessing (i.e. removing invalid records) are presented in TABLE 3. The baseline algorithms we chose for comparison with the proposed MBMF-F, MBMF-C and MBMF-N are:

- *Matrix Factorisation for recommender systems (MF)* [15]. The conventional approach used for recommender systems.
- *Non-negative Matrix Factorisation algorithms (NMF)* [23]. Using "multiplicative rules" instead of addition formula in the SGD, NMF becomes popular when dealing with non-negative observations.
- *Feature-wise updated Cyclic Coordinate Descent method (CCD++)* [24]. It is a new and fast non-bounding approach to handle large-scale datasets.

- *Bounded Matrix Factorisation (BMF)* [16]. This is the first algorithm studying bounding issues in matrix factorisation for recommender systems.
- *Bounded Matrix Completion in Alternating Direction of Multiplier Method (BMC-ADMM)* [17]. It solved the convergence issue in BMF with the help of ADMM framework.
- *Bounded Singular Value Decomposition (Bounded-SVD)* [18]. This approach tries to bound factorising entries by imposing exponential penalties.

TABLE 3: Statistics of Datasets used for Experiments

dataset	users	items	density	range	ordinal
Synthetic	500	500	0.2	[0,10]	N
Jester	73,421	100	0.5634	[-10,10]	N
BookCrossing	92,184	270,169	0.00004	[1,10]	Y
Ali Mob	10,000	8,916	0.0101	[1,6853]	Y

Note that NMF and MBMF-N require the input data to be non-negative, MBMF-C requires the data to be centred with respect to magnitudes, MBMF-F requires the data to be centred with respect to systematic bounds, while other algorithms have no additional requirements. Thus, each dataset has been pre-processed into a non-negative and two centred ones.

Three metrics are used to evaluate the performance of recommendations:

(1) *Root Mean Square Error (RMSE)* [25], which evaluates difference between the recovered ratings and their corresponding original ratings:

$$\text{RMSE} = (\sum_{ij} \boldsymbol{M}_{ij})^{-1/2} \|\boldsymbol{M} * (\boldsymbol{V} - \boldsymbol{W}^* \boldsymbol{H}^*)\|_F, \qquad (45)$$

where $\boldsymbol{W}^*$ and $\boldsymbol{H}^*$ are the final factorised matrices, and $\boldsymbol{M}$ is a binary matrix in which one denotes the chosen entry for cross validation.

(2) *Mean Absolute Error (MAE)* [26], which evaluates the absolute difference between the recovered ratings and their corresponding original ratings:

$$\text{MAE} = \frac{1}{\sum_{ij} \boldsymbol{M}_{ij}} |\boldsymbol{M} * (\boldsymbol{V} - \boldsymbol{W}^* \boldsymbol{H}^*)|. \qquad (46)$$

(3) *F1 score* [27], which is a percentage value that evaluates how much proportion of correct recommendations is with respect to users:

$$\text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \qquad (47)$$

In our experiments, for each user we use its average rating as a threshold, and ratings greater than it imply that the corresponding items are recommended. Before and after factorisation, we obtain two binary recommendation vectors which can be used to calculate the F1 score.

For RMSE and MAE, lower is better, while for F1 Score, higher is better.

The maximal iteration times for all of the algorithms are set at 500. An early termination condition is also applied: if the decrease of the objective function value is smaller than 10^{-5} for 10 consecutive iterations, the algorithm is regarded as converged.

The source codes of MBMF as well as all the preprocessed datasets we used for experiments can be downloaded at: https://github.com/shawn-jiang/MBMF.

4.1 Prediction Variation

We conduct experiments on synthetic datasets to explore the prediction variation of algorithms. In this experiment, K varies from 5 to 50 with step 5. For each K, we adopt the following experimental procedure:

- Randomly generate a factorising matrix $\boldsymbol{V}$ of size 500×500 and scale its entries to $[0, 10]$;
- Remove 80% of the values (set them to missing data) in $\boldsymbol{V}$ randomly while making sure the remaining part does not have zero rows/columns;
- Run each of the algorithms on the remaining matrix 10 times, and record their recovered matrices;
- Calculate the standard deviation (σ) on missing entries.

The average and maximal standard deviation over predictions is presented in Fig. 2. The lower the standard deviation is, the more stable the predictions are. Algorithms with stable predictions are more operational on large datasets.

According to Fig. 2 (note that in Fig. 2 (b) the maximal σ of NMF is greater than the limit of the y-axis in many cases, since we would like to show more details for other algorithms), we can see that four bounding algorithms (BMF, BMC-ADMM, Bounded-SVD and MBMF-N) all successfully reduced the prediction variation compared with the traditional matrix factorisation methods (MF and NMF) which have no bounding conditions. MBMF-N achieves both the lowest average σ and the lowest maximal σ in all cases, followed by CCD++ (the performance of CCD++ is impressive although it has no bounding constraints), BMF and Bounded-SVD. This experiment demonstrates that MBMF is so far the best bounding algorithm in terms of reducing prediction variation, and can be easily applied on real recommender systems: it does not require repeated runs.

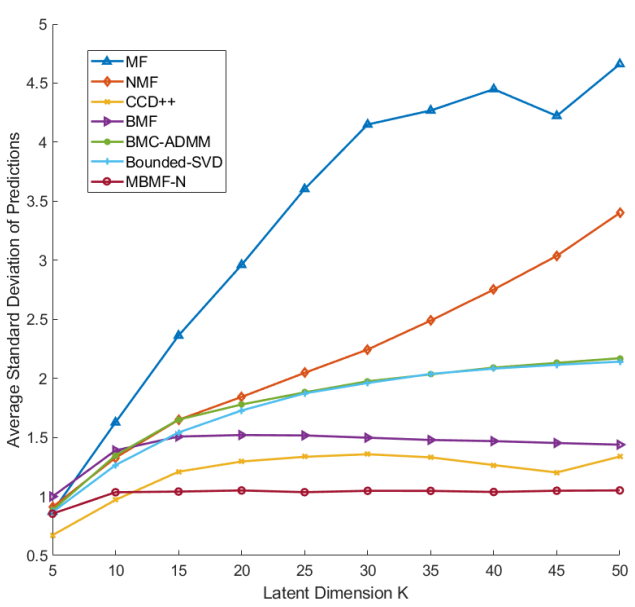
4.2 Performance on Real Recommender Systems

As for the experiments on real recommendation datasets, we first randomly divided each of them into historical data and present data. The historical data can have zero rows/columns while the present data cannot. Then in each run we randomly picked 10% entries from the present data as a validation fold. The latent dimension K varies in $10, 20, 50$. We ran 5 folds for each K and report the average results.

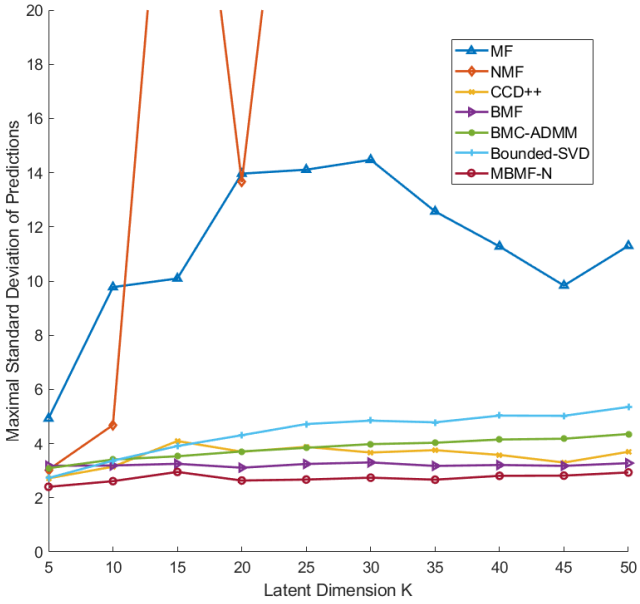
4.2.1 Jester Dataset

Jester is a free online dataset containing about 4.1 million continuous ratings ($[-10, 10]$) of 100 jokes from $73,421$ users collected between April 1999 and May 2003. Due to the high density, the continuity in ratings and the centrosymmetric bounds, which are rarely seen in other recommender systems, this dataset becomes popular to test recommendation algorithms. Also because of this reason, we used 80% as historical data and 20% as present data. Parameters set for this dataset are: MF, $\lambda = 1e^{-4}$; CCD++, $\lambda = 0.1$; BMC-ADMM, $\lambda = 100, p_1 = p_2 = 1$; Bounded-SVD, $\eta = 1e^{-4}, \alpha = 1$; MBMF, $\rho = 0.05, \lambda = 1$. The results are showed in TABLE 4.

From TABLE 4, we can see that among non-bounding algorithms, NMF fails and gets very high RMSE and MAE, and



(a) ave(σ) of predictions.



(b) max(σ) of predictions.

Fig. 2: Prediction variation on Synthetic dataset.

TABLE 4: Recommendation Performance on Jester Dataset

	RMSE				MAE				F1 Score			
K	10	20	50	Avg.	10	20	50	Avg.	10	20	50	Avg.
MF	5.6107	4.9242	4.6770	5.0710	4.2949	3.8589	3.7919	3.9820	60.54	62.26	64.35	62.38
NMF	333.18	98.349	10.161	147.20	9.3592	9.3339	5.6652	8.1190	57.96	56.02	58.07	57.35
CCD++	5.6643	5.5892	5.1609	5.4710	4.2840	4.2523	3.9493	4.1620	60.40	60.08	60.46	60.31
BMF	5.4271	5.5027	5.3717	5.4340	4.2851	4.3581	4.2744	4.3060	58.62	58.96	58.79	58.79
BMC-ADMM	4.9162	4.9477	4.7541	4.8730	3.8149	3.8807	3.7961	3.8310	60.52	57.42	55.22	57.72
Bounded-SVD	6.3798	6.7069	5.7458	6.2780	4.7758	5.1434	4.5270	4.8150	59.90	58.32	58.82	59.01
MBMF-F	4.9785	5.0395	5.2365	5.0840	3.9484	4.0252	4.2259	4.0670	58.79	58.62	56.48	57.96
MBMF-C	4.6816	4.6695	4.6773	4.6760	3.8045	3.8014	3.8039	3.8030	64.08	64.47	64.16	64.24
MBMF-N	**4.6474**	**4.6399**	**4.6344**	**4.6410**	**3.7467**	**3.7473**	**3.7404**	**3.7450**	**65.58**	**65.71**	**65.60**	**65.63**

also the lowest F1 Score. MF is a bit better than CCD++, but overall their performance is close. For bounding algorithms, Bounded-SVD obtains the highest (worst) RMSE and MAE, while BMF gets the lowest F1 Score. The performance of MBMF-N on this dataset is the best over all three metrics. It achieves 4.6410 RMSE, 3.7450 MAE and 65.63% F1 Score, followed by MBMF-C with 4.6760 RMSE, 3.8030 MAE and 64.24% F1 Score. The performance of MBMF-F is better than BMF and Bounded-SVE, but worse than BMC-ADMM, MBMF-C and MBMF-N. This demonstrates that bounding the predicted values in product matrix in individually concise ranges can help improve the recommendation performance. Another observation is that, the improvement of MBMF to comparative algorithms is massive on this dataset. We believe this is because the magnitudes obtained from the historical ratings can represent users and jokes very well. From TABLE 3 we know that this dataset is the smallest and most dense one compared to the other two datasets, which means the proposed MBMF is superior to all other existing algorithms on small and dense systems.

4.2.2 *BookCrossing Dataset*

Bookcrossing is a large online dataset which contains 278,858 users with 1,149,780 ratings (explicit / implicit) about 271,379 books. However, neither all users nor all books have corresponding ratings in this dataset. From the ratings file, we constructed a rating matrix with $92,184$ rows (users) and $270,169$ columns (books). It has been randomly split into 50% historical and 50% present data. Parameters set for this dataset are: MF, $\lambda = 1e^{-4}$; CCD++, $\lambda = 0.1$; BMC-ADMM, $\lambda = 0.1, p_1 = p_2 = 1$; Bounded-SVD, $\eta = 1e^{-4}, \alpha = 1$; MBMF, $\rho = 0.05, \lambda = 1$. We only did one experiment using BMF for each K on this dataset since it is too time consuming (the experiment with K set 50 took about 30 days). The results are showed in TABLE 5.

TABLE 5: Recommendation Performance on BookCrossing Dataset

	RMSE				MAE				F1 Score			
K	10	20	50	Avg.	10	20	50	Avg.	10	20	50	Avg.
MF	1.7566	1.3539	1.6726	1.5940	1.3164	0.9979	1.2578	1.1910	**87.25**	86.85	77.31	83.80
NMF	1.8827	1.6211	1.4146	1.6390	1.2889	1.1479	0.9779	1.1380	82.68	83.98	86.43	84.36
CCD++	1.3137	1.3191	1.3131	1.3150	0.8713	0.8753	0.8699	0.8722	87.05	87.07	**87.12**	**87.08**
BMF	1.3409	1.2863	1.2990	1.3090	0.9329	0.8563	0.8653	0.8848	86.29	86.29	86.06	86.21
BMC-ADMM	3.3757	3.0368	2.8886	3.1000	3.0033	2.6781	2.5543	2.7450	81.93	78.93	67.27	76.04
Bounded-SVD	1.6221	1.3541	1.6021	1.5260	1.1873	0.9712	1.2017	1.1200	87.02	**87.47**	77.94	84.14
MBMF-F	1.3898	1.3929	1.3951	1.3930	0.9909	0.9893	0.9943	0.9915	83.98	84.01	84.02	84.00
MBMF-C	**1.1783**	**1.1846**	**1.1813**	**1.1810**	**0.7810**	**0.7849**	**0.7817**	**0.7825**	85.56	85.64	85.63	85.61
MBMF-N	1.2598	1.2639	1.2628	1.2620	0.8503	0.8534	0.8526	0.8521	86.33	86.30	86.32	86.32

From TABLE 5, BMC-ADMM gets the worst performance, with the highest RMSE and MAE and the lowest F1 Score. Except for our proposed MBMF-C, MBMF-F and MBMF-N, the best algorithms on this dataset are CCD++ (non-bounding) and BMF (bounding). Their performance is close and CCD++ also obtains the best F1 Score among all algorithms. However, when it comes to RMSE and MAE, our MBMF-C is the best which achieves the lowest RMSE (1.1810) and MAE (0.7825) in all cases, followed by our MBMF-N. Even over the F1 Score, MBMF-N is the second best, scoring 86.32% just after CCD++ (87.08%). Notice that this dataset is not only large-scale ($92,184 \times 270,169$) but very sparse (0.00004 density) as well (see TABLE 3). Thus, the results demonstrate that MBMF is also suitable to be applied on large-scale and sparse datasets. The improvements from MBMF-F to MBMF-C/MBMF-N verify again the effectiveness of our individual bounding motivations.

4.2.3 Ali Mob Dataset

Ali Mob is a public recommendation dataset provided by Tianchi platform. It recorded the online shopping behaviours of users on Alibaba's M-Commerce platforms in 2014. The dataset consists of $10,000$ users and $2,876,947$ commodities in $8,916$ categories. $12,256,906$ valid records were collected in this dataset which contained users' behaviours (click, collect, add-to-cart and payment) towards commodities. We preprocessed this dataset as following:

- Count the number of each behaviour (b_1, b_2, b_3, b_4) for each user (u) on each category of commodities (c);
- Calculate the interest points (p) of user u on category c:

$$p = w_1 b_1 + w_2 b_2 + w_3 b_3 + w_4 b_4, \tag{48}$$

  where w_1, w_2, w_3, w_4 are behaviour weights. We set $w_1 = 1, w_2 = 2, w_3 = 3, w_4 = 5$ in our experiments according to the increase of interest from "click" to "payment".
- Construct the data matrix with interest points whose rows denote users and columns denote categories.

This dataset does not have a pre-defined pair of bounds and all values in the factorising matrix are non-negative. Since the range of observations varies among users/items, we use the maximal value and minimal value as the upper and lower bounds respectively for algorithms (i.e. BMF, BMC-ADMM, Bounded-SVD and MBMF-F) which can only handle one single pair of bounds. Half ratings are used as historical data while the other half is used as the present. Parameters set for this dataset are: MF, $\lambda = 1e^{-4}$; CCD++, $\lambda = 0.1$; BMC-ADMM, $\lambda = 1, p_1 = p_2 = 1$; Bounded-SVD, $\eta = 1e^{-4}, \alpha = 0.01$; MBMF, $\rho = 0.05, \lambda = 1$. The results are showed in TABLE 6.

As shown in TABLE 6, among non-bounding algorithms, MF and NMF fail to get reasonable approximations and recommendations, while CCD++ works somewhat. Both MF and CCD++ obtain better results when K is set at 50. This implies that the performance of these algorithms is affected by the value of K. Among bounding algorithms, both BMF and BMC-ADMM fail. They are projection-based algorithms which project the product values that are out of range to the exact boundaries. So this bad performance implies that the projection approach cannot handle datasets without pre-defined bounds. As for Bounded-SVD, we observed from the dataset that most of the interest points are small, and only a tiny part of users on specific categories of commodities have ridiculously high points. Therefore, we think the way that Bound-SVD restricts all the predictions to be small mitigates the inaccurate recommendation for those extreme users and leads to good results. However, our MBMF-C and MBMF-N considers all users' interest points over all categories and sets different magnitudes for them. From the results we can see that, MBMF-C achieves the best performance over MAE (14.378 in average) and F1 score (43.51% in average) in all cases. On the other hand, MBMF-F is still workable on this dataset, although its performance falls far behind the ones with magnitude constraints. This experiment demonstrates that our proposed MBMF can properly handle datasets which do not have pre-defined bounds.

From all the experiments on real-world datasets, the proposed MBMF achieves the best performance in most cases (29 out of 36 if we split the three metrics RMSE, MAE and F1 Score into four cases of $K = 10$, $K = 20$, $K = 50$ and $Ave.$), and comparable in the remaining ones.

4.3 Discussion

From the experimental results, we found an interesting and noteworthy phenomenon: the performance of MBMF does not change

TABLE 6: Recommendation Performance on Ali Mob Dataset

	RMSE				MAE				F1 Score			
K	10	20	50	Avg.	10	20	50	Avg.	10	20	50	Avg.
MF	115.93	95.286	61.138	90.784	29.930	34.871	29.975	31.592	38.99	38.22	38.76	38.66
NMF	235511	4540.0	11238	131098	2686.0	959.17	1223.1	1622.8	35.90	34.29	34.77	34.99
CCD++	63.087	59.334	56.933	59.785	22.667	25.103	26.128	24.633	39.58	38.66	38.29	38.84
BMF	549.66	455.34	380.98	461.99	507.57	428.79	362.07	432.81	32.07	32.28	31.69	32.01
BMC-ADMM	319.65	392.39	593.43	435.16	157.90	205.80	358.47	240.73	17.70	18.39	22.85	19.65
Bounded-SVD	43.381	**42.810**	**41.766**	**42.652**	15.827	15.953	16.438	16.073	41.83	42.41	42.68	42.31
MBMF-F	50.548	52.530	50.532	51.200	21.459	20.910	21.757	21.380	35.06	34.51	34.76	34.78
MBMF-C	43.273	43.671	43.881	43.608	**14.405**	**14.329**	**14.400**	**14.378**	**43.47**	**43.51**	**43.56**	**43.51**
MBMF-N	**42.540**	42.903	42.999	42.814	18.963	18.948	18.896	18.935	39.64	39.70	39.72	39.69

much when K varies, while other methods obtain their best results with different K when applied on different datasets. This observation reveals the high stability, a unique feature of MBMF which other methods do not have. It implies that the selection of latent dimension K has little impact on the performance of MBMF. This is very useful when applying it on large scale datasets because it means we do not have to repeat experiments to determine the best suitable K which is usually required by existing methods. The reason of the high stability in it is the magnitude constraint: no matter what K's value is, the magnitude of row/column in the factorised matrices is the same.

To some degree, however, the magnitude constraints are quite strict as well: since all the rows/columns in the factorised matrices have to conform to these magnitude constraints, there is not much space for the SGD method to approximate the product of factorised matrices to be as close to the factorising matrix as in other algorithms. In our future work, we will relax the magnitude constraints from equalities to inequalities (i.e. $|\boldsymbol{W}_{i:}| \leqslant \boldsymbol{r}_i^{\boldsymbol{W}}$), which will make MBMF more flexible but harder to solve as well. We believe that imposing magnitude constraints is an effective way to solve bounded matrix factorisation.

5 Conclusions

In this paper, we propose a novel matrix factorisation algorithm for recommender systems called MBMF. It is a bounded algorithm which reduces the fluctuation of predictions especially for large and sparse datasets. The magnitude constraints allow MBMF to have individually different bounds for each entry and make the algorithm capable of dealing with data that has no fixed bounds. The conversions of coordinates used in the method of MBMF turn the magnitude constrained optimisation into an equivalent unconstrained one by setting the radii as corresponding magnitudes. As for the unconstrained task, we apply a stochastic gradient descent method to solve it efficiently. In order to update the variables efficiently, we introduce an acceleration approach which involves tensor construction and multiplication based on the derivative patterns. We also propose a MBMF variant, Non-negative MBMF to deal with non-negative data. Compared with the unconstrained MF, NMF and CCD++ as well as the-state-of-art bounding methods BMF, BMC-ADMM and Bounded-SVD, MBMF achieved superior performance in most cases over all evaluation metrics on both synthetic datasets and real recommendation datasets. Moreover, the improvements from MBMF-F to MBMF-C/MBMF-N verify the effectiveness and benefits of bounding product entries in different individual ranges.

References

[1] V. W. Zheng, B. Cao, Y. Zheng, X. Xie, and Q. Yang, "Collaborative filtering meets mobile recommendation: A user-centered approach," in *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI-10)*, vol. 10, 2010, pp. 236–241.

[2] C. Wang and D. M. Blei, "Collaborative topic modeling for recommending scientific articles," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '11. ACM, 2011, pp. 448–456.

[3] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutirrez, "Recommender systems survey," *Knowledge-Based Systems*, vol. 46, pp. 109–132, 2013.

[4] R. M. Bell and Y. Koren, "Scalable collaborative filtering with jointly derived neighborhood interpolation weights," in *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, Oct 2007, pp. 43–52.

[5] Y. Koren, "Factorization meets the neighborhood: A multifaceted collaborative filtering model," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '08. ACM, 2008, pp. 426–434.

[6] Q. Gu, J. Zhou, and C. Ding, "Collaborative filtering: Weighted nonnegative matrix factorization incorporating user and item graphs," in *Proceedings of the 2010 SIAM International Conference on Data Mining*, 2010, pp. 199–210.

[7] L. Baltrunas, B. Ludwig, and F. Ricci, "Matrix factorization techniques for context aware recommendation," in *Proceedings of the Fifth ACM Conference on Recommender Systems*, ser. RecSys '11. ACM, 2011, pp. 301–304.

[8] J. D. M. Rennie and N. Srebro, "Fast maximum margin matrix factorization for collaborative prediction," in *Proceedings of the 22Nd International Conference on Machine Learning*, ser. ICML '05. ACM, Oct 2005, pp. 713–719.

[9] X. Luo, Y. Xia, and Q. Zhu, "Applying the learning rate adaptation to the matrix factorization based collaborative filtering," *Knowledge-Based Systems*, vol. 37, pp. 154–164, 2013.

[10] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, pp. 19:1–19:19, Dec 2015.

[11] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, "Eigentaste: A constant time collaborative filtering algorithm," *Information Retrieval*, vol. 4, no. 2, pp. 133–151, Jul 2001.

[12] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen, "Improving recommendation lists through topic diversification," in *Proceedings of the 14th International Conference on World Wide Web*, ser. WWW '05. ACM, 2005, pp. 22–32.

[13] L. Brozovsky and V. Petricek, "Recommender system for online dating service," *arXiv preprint cs/0703042*, 2007.
[14] "Netflix prize dataset," https://www.netflixprize.com/.
[15] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug 2009.
[16] R. Kannan, M. Ishteva, and H. Park, "Bounded matrix factorization for recommender system," *Knowledge and Information Systems*, vol. 39, no. 3, pp. 491–511, 2014.
[17] H. Fang, Z. Zhang, Y. Shao, and C.-J. Hsieh, "Improved bounded matrix completion for large-scale recommender systems," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 2017, pp. 1654–1660.
[18] B. H. Le, K. Q. Nguyen, and R. Thawonmas, "Bounded-svd: A matrix factorization method with bound constraints for recommender systems," in *2015 International Conference on Emerging Information Technology and Engineering Solutions*, Feb 2015, pp. 23–26.
[19] C. Cheng, H. Yang, I. King, and M. R. Lyu, "Fused matrix factorization with geographical and social influence in location-based social networks," in *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI-12)*, 2012, pp. 17–23.
[20] Y. Koren, "Collaborative filtering with temporal dynamics," *Commun. ACM*, vol. 53, no. 4, pp. 89–97, 2010.
[21] V. W. Zheng, Y. Zheng, X. Xie, and Q. Yang, "Collaborative location and activity recommendations with gps history data," in *Proceedings of the 19th International Conference on World Wide Web*, ser. WWW '10. ACM, 2010, pp. 1029–1038.
[22] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers*. Physica-Verlag HD, 2010, pp. 177–186.
[23] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Advances in Neural Information Processing Systems 13*. MIT Press, 2001, pp. 556–562.
[24] H.-F. Yu, C.-J. Hsieh, S. Si, and I. Dhillon, "Scalable coordinate descent approaches to parallel matrix factorization for recommender systems," in *2012 IEEE 12th International Conference on Data Mining*, Dec 2012, pp. 765–774.
[25] A. G. Barnston, "Correspondence among the correlation, rmse, and heidke forecast verification measures; refinement of the heidke score," *Weather and Forecasting*, vol. 7, no. 4, pp. 699–709, 1992.
[26] C. J. Willmott and K. Matsuura, "Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance," *Climate research*, vol. 30, no. 1, pp. 79–82, 2005.
[27] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," 2011.

Shuai Jiang received the bachelors degree in computer science and technology from Beijing Institute of Technology, Beijing, China, in 2013. Currently he is working towards the dual doctoral degree in both Beijing Institute of Technology and University of Technology Sydney. His main interests include machine learning, optimisation and data analytics.

Kan Li is currently a Professor in the School of Computer at Beijing Institute of Technology. He has published over 50 technical papers in peer-reviewed journals and conference proceedings. His research interests include machine learning and pattern recognition.

Richard Yi Da Xu received the B.Eng. degree in computer engineering from the University of New South Wales, Sydney, NSW, Australia, in 2001, and the Ph.D. degree in computer sciences from the University of Technology at Sydney (UTS), Sydney, NSW, Australia, in 2006. He is currently an Associate Professor of School of Electrical and Data Engineering, UTS. His current research interests include machine learning, deep learning, data analytics and computer vision.