Matrix Factorization with Scale-Invariant Parameters

Guangxiang Zeng[1], Hengshu Zhu[2], Qi Liu[1,*], Ping Luo[3], Enhong Chen[1], Tong Zhang[2]
[1]School of Computer Science and Technology, University of Science and Technology of China,
zgx@mail.ustc.edu.cn, qiliuql@ustc.edu.cn, cheneh@ustc.edu.cn
[2]Baidu Research-Big Data Lab, zhuhengshu@baidu.com, zhangtong10@baidu.com
[3]Key Lab of Intelligent Information Processing of Chinese Academy of Sciences, Institute of Computing Technology, Chinese Academy of Sciences, luop@ict.ac.cn

Abstract

Tuning hyper-parameters for large-scale matrix factorization (MF) is very time consuming and sometimes unacceptable. Intuitively, we want to tune hyper-parameters on small sub-matrix sample and then exploit them into the original large-scale matrix. However, most of existing MF methods are scale-variant, which means the optimal hyper-parameters usually change with the different scale of matrices. To this end, in this paper we propose a scale-invariant parametric MF method, where a set of scale-invariant parameters is defined for model complexity regularization. Therefore, the proposed method can free us from tuning hyper-parameters on large-scale matrix, and achieve a good performance in a more efficient way. Extensive experiments on real-world dataset clearly validate both the effectiveness and efficiency of our method.

1 Introduction

Matrix Factorization (MF) is among the most important machine learning techniques for real-world *Collaborative Filtering* (CF) applications, which attracts more and more attention in recent years [Koren *et al.*, 2009; Ge *et al.*, 2011; Wu *et al.*, 2012; Zhu *et al.*, 2014]. The main idea behind MF is that an $m \times n$ *user-item* rating matrix, where n items is assigned to m users, is modeled by the product of an $m \times K$ user factor matrix and the transpose of an $n \times K$ item factor matrix [Srebro *et al.*, 2003; Rennie and Srebro, 2005].

A variety of effective MF methods have been proposed, which can be mainly grouped into non-convex methods [Mnih and Salakhutdinov, 2007; Pilászy *et al.*, 2010; 2010] and convex methods [Bach *et al.*, 2008; Journée *et al.*, 2010; Bouchard *et al.*, 2013]. The main difference between these two types of methods is that the number of factors can be self-determined in convex methods while it has to be set manually in non-convex methods. Indeed, both previous convex and non-convex methods can achieve good performance through carefully tuning their hyper-parameters. However, the cost of tuning hyper-parameters is often ignored. In fact, tuning hyper-parameters for large-scale MF problems is very time consuming and sometimes unacceptable [Chan *et al.*, 2013]. Intuitively, a straightforward solution to this problem is to tune the hyper-parameters on small sub-matrix and then directly exploit them into the original large matrix. However, most of existing MF methods cannot work well for this idea due to the fact that their hyper-parameters are sensitive to the scale of matrix. In other words, the optimal hyper-parameters usually change with the different scale of matrices. We will both theoretically and experimentally analyze this issue in Section 5 and Section 6, respectively.

To address the above challenge, in this paper we propose a *scale-invariant* parametric MF method for facilitating model selection. Specifically, we assume that: *the k-th latent factor values of any rating matrix and its sub-matrices follow the same normal distribution $\mathcal{N}(0, \sigma_k^2)$*. With this assumption, we can first estimate a set of suitable $\{\sigma_k^2\}_{k=1}^K$, namely *Factorization Variances*, by a randomly drawn sub-matrix. Then we use them as constraint parameters to formulate a new MF problem such that the variance of the k-th dimension latent factor values is not bigger than σ_k^2, which can be used to conduct MF for the original large-scale matrix and any of its sub-matrices. Since the latent factor values of the sub-matrix and the original matrix are generated from the same distributions, we may still get a good performance on the original large matrix. In this process, factorization variances can introduce the similar effect of model complexity regularization, which is irrelevant to the scale of training matrices. In this sense, the proposed method is *Scale-Invariant*, which can free us from tuning hyper-parameters on large-scale matrix and achieve a good performance in a more efficient way. Specifically, our contributions can be summarized as follows.

First, to the best of our knowledge, we are the first to formulate the problem of MF with scale-invariant parameters. The unique perspective to this problem is to use the variance of the latent factor values from each dimension to control model complexity. All these parameters can be estimated on the resultant latent matrix output from any previous MF method on a random-sampled sub-matrix.

Second, we find that the formulated problem can be transformed into an equivalent problem, where the feasible set for each factor dimension is within a sphere. To solve this problem, we first initialize each dimension with the steepest decent eigenvector of the gradient matrix of the objective function. Then, we optimize the factors by the gradient decent

*Corresponding author.

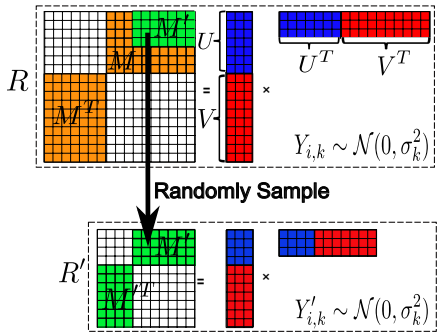

Figure 1: An Example of Factorization Variances.

method with projection to the feasible set.

Third, we both theoretically and experimentally prove that previous MF methods (i.e., in the form of both trace trade-off and trace bounding) are scale-variant. Moreover, extensive experiments on a real-world data set also empirically prove the effectiveness of our method, i.e., the hyper-parameters tuned by small sub-matrix can still achieve good performance on the original large-scale matrix.

2 Preliminaries & Assumption

In this paper, we follow the *Positive Semi-Definite* (PSD) style MF model [Bach *et al.*, 2008; Candès and Recht, 2009], which is shown in the top of Figure 1. Indeed, it is also equivalent to those SVD style formulations for MF [Mnih and Salakhutdinov, 2007; Koren, 2008]. Specifically, with PSD formulation the input matrix $M = UV^T \in \mathbb{R}^{m\times n}$ is put at the top-right corner of the PSD matrix R, and M^T is put at the bottom-left corner of R. Here, we aim to solve the factorization problem $R = YY^T$, where the top part of Y equals to U, and the bottom part of Y equals to V.

Different with previous studies, in this paper we consider the shared property among the column values of Y, and characterize the values in Y in a more sophisticated way. Specifically, we assume the values of the k-th column of Y satisfy the following distribution:

$$Y_{i,k} \sim \mathcal{N}(0, \sigma_k^2), i.i.d., i = 1, ..., N. \tag{1}$$

As we can see, values lie in the same column (e.g., the k-th column) share a same variance σ_k^2, and the $\{\sigma_k^2\}_{k=1}^K$ are named *Factorization Variances*. In particular, we assume:

Assumption 2.1. *The factorization variances $\{\sigma_k^2\}_{k=1}^K$ are scale-invariant for MFs on different sized sub-matrices which are randomly drawn from the same matrix.*

In other words, in our approach, the sampled sub-matrices share the same factorization variances with the original large matrix (i.e., as shown in Figure 1). With the above assumption, we can estimate a set of optimal parameters $\{\sigma_k^2\}_{k=1}^K$ on the sampled small sub-matrix M', which is relatively efficient. Since the latent matrix Y and Y' are generated from the same distributions, we can exploit the estimated $\{\sigma_k^2\}_{k=1}^K$ into the original large matrix M and hopefully obtain comparable performance. Along this line, there are two major challenges:

- How to utilize the scale-invariant parameters, i.e, factorization variances, for large-scale MF?
- How to estimate the factorization variances (including the number of factors, namely K)?

None of the two challenges is trivial task. For the first one, we will propose a new problem, matrix FActorization with factorization VAriances (FAVA for short), with the factorization variances as the constraints, and solve it in Section 3. Furthermore, the second one will be discussed in Section 4.

3 MF with Factorization Variances

In this section, we assume the factorization variances $\{\sigma_k^2\}_{k=1}^K$ are known in advance. The estimation method for them will be detailed in Section 4.

3.1 The Formulation of FAVA

Here, we introduce how to adapt the factorization variances as the constraints for MF process. Specifically, in order to utilize the Assumption 2.1, we impose the following constraints:

$$\frac{1}{N}\sum_{i=1}^{N} Y_{i,k}^2 \le \sigma_k^2, k = 1, ..., K. \tag{2}$$

The intuition behind these constraints is that we let the variance of the k-th column elements no larger than the known factorization variance. In these constraints, $\{\sigma_k^2\}_{k=1}^K$ serve as the model complexity controllers. The bigger the value of $\{\sigma_k^2\}_{k=1}^K$ is, the more freedom the values of the corresponding column have. Thus, it actually defines the feasible set for searching solution and guarantees the generalization ability of the model. With these constraints, we formulate the FAVA problem as follows:

$$\begin{aligned} \min_{Y} \quad & G(YY^T) = \sum_{(i,j)\in\Omega} (R_{i,j} - Y_{i,:} \cdot Y_{j,:})^2 \\ s.t. \quad & \tfrac{1}{N}\sum_{i=1}^{N} Y_{i,k}^2 \le \sigma_k^2, k = 1, ..., K, \end{aligned} \tag{3}$$

where $Y_{i,:}$ means the i-th row of matrix Y. In order to turn the contraints into a consistent form, we let:

$$\Sigma_K = diag\{\sigma_1, \sigma_2, .., \sigma_K\}, \tag{4}$$

and then we can rewrite the variable Y as follows:

$$Y = P\Sigma_K \tag{5}$$

where $P \in \mathbb{R}^{N\times K}$. Then, it is obvious that the constraints in Equation (2) are equivalent to $||P_{:,k}||^2 \le N$, where $P_{:,k}$ means the k-th column of P, $k = 1, ..., K$. Then, we get the following equivalent form of FAVA:

$$\begin{aligned} \min_{P} \quad & G(YY^T) = \sum_{(i,j)\in\Omega} (R_{i,j} - Y_{i,:} \cdot Y_{j,:})^2 \\ s.t. \quad & Y = P\Sigma_K \\ & ||P_{:,k}||^2 \le N, k = 1, ..., K. \end{aligned} \tag{6}$$

If the above problem is solved, we can get $Y = P\Sigma_K$ immediately. However, there are still some challenges:

- Firstly, simple gradient based method could not guarantee the searched solutions always meet the constraints of in Equation (6).

- Secondly, how to initialize the values of P? Randomly initialization can be a solution, but far from the best.

Next we will first show how we search solution for problem in Equation (6) in Section 3.2, then we give an effective initialization method for P in Section 3.3.

3.2 Sphere Projection for FAVA

As we can see in the constrains of problem in Equation (6), the $P_{:,k}$ is within a ball with radius $\sqrt{N}$ and its sphere, we optimize the latent factors by gradient decent method with projection to the feasible set of FAVA. Firstly, the gradient of objective function in Equation (6) is:

$$\triangle P = \frac{\partial G(YY^T)}{\partial P} = 2\nabla_X G(YY^T)P\Sigma_K^2, \tag{7}$$

from above we also know that:

$$\triangle P_{:,k} = \frac{\partial G(YY^T)}{\partial P_{:,k}} = 2\sigma_k^2 \nabla_X G(YY^T)P_{:,k}. \tag{8}$$

Assume α_t is the t-th iteration decent step size, then we update each column $P_{:,k}$ by the following rule:

$$P_{:,k}^{(t+1)} = \Pi_N(P_{:,k}^{(t)} - \alpha_t \cdot \triangle P_{:,k}^{(t)}), \tag{9}$$

where

$$\Pi_N(P'_{:,k}) = \begin{cases} \frac{\sqrt{N}\cdot P'_{:,k}}{\sqrt{tr(P'_{:,k}{P'_{:,k}}^T)}}, & tr(P'_{:,k}{P'_{:,k}}^T) > N, \\ P'_{:,k}, & else. \end{cases} \tag{10}$$

As the projection only happens when $tr(P'_{:,k}{P'_{:,k}}^T) > N$, and the projection makes $P'_{:,k}$ project onto the sphere of ball with radius $\sqrt{N}$ in $\mathbb{R}^N$. Thus, we call this method *Sphere Projection*. In addition, the step size α_t can be searched by the binary search method **BiSearch**($P^{(t)}$, $\triangle P^{(t)}$) as follows:

1. Set $\alpha_t = 1.0$;
2. Update each column of $P^{(t)}$ by Equation (9) and the result is denoted as $P^{(t+1)}$;
3. Let $Y^{(t+1)} = P^{(t+1)}\Sigma_K$, $X^{(t+1)} = Y^{(t+1)}{Y^{(t+1)}}^T$. If $G(X^{(t+1)}) \geq G(X^{(t)})$, let $\alpha_t = \frac{1}{2}\alpha_t$ repeat 2 and 3, else return α_t.

Finally, we stop the optimization iteration when $\alpha_t < \varepsilon$, where ε is a given accuracy level.

3.3 Steepest Initialization for FAVA

In this subsection, we will show the process that initializes columns of P with eigendecomposition of gradient matrix $\nabla_X G(X)$. The intuition behind this process is that we initialize the columns of P with the first K steepest decent direction of $G(X)$ at $X = 0$.

Firstly, by the definition in Equation (6), we know that $\nabla_X G(X)$ is always a symmetric matrix of the block form $\nabla_X G(X) = \begin{bmatrix} 0 & M \\ M^T & 0 \end{bmatrix}$, when $X \succeq 0$. As mentioned in [Jaggi *et al.*, 2010], we have the following proposition:

Proposition 3.1. *The spectrum of $\nabla_X G(X)$ is always symmetric: whenever $\begin{bmatrix} u \\ v \end{bmatrix}$ is an eigenvector for some eigenvalue ρ, then $\begin{bmatrix} u \\ -v \end{bmatrix}$ is an eigenvector for $-\rho$.*

So we know that the smallest eigenvalue of $\nabla_X G(X)$ is always $\rho_{min} \leq 0$. Then we have the following theorem.

Theorem 3.2. *Given $Y_k \in \mathbb{R}^{N\times k}$, let $Y_{k+1} = [Y_k|\sqrt{\beta}\cdot\vec{y}] \in \mathbb{R}^{N\times(k+1)}$, where $\beta \in \mathbb{R}^+$, $\vec{y} \in \mathbb{R}^N$ and $||\vec{y}|| = 1$, $\vec{y}$ is called attached vector for Y_k. Then, the unit eigenvector $\vec{y}_{min}$ for the smallest eigenvalue ρ_{min} of $\nabla_X G(Y_k Y_k^T)$ is the steepest decent attached vector of Y_k, which means that $f(\beta, \vec{y}) = G(Y_{k+1}Y_{k+1}^T)$ decent the fastest when $\vec{y} = \vec{y}_{min}$ at $\beta = 0$.*

Proof. Firstly, let $X_k = Y_k Y_k^T$ and $\triangle X = \vec{y}\vec{y}^T$. By definition we have:

$$\begin{aligned} f(\beta, \vec{y}) = &\ G(Y_{k+1}Y_{k+1}^T) = G(Y_k Y_k^T + \beta\vec{y}\vec{y}^T) \\ = &\ G(X_k + \beta\triangle X). \end{aligned} \tag{11}$$

Then we have:

$$f'_\beta(\beta, \vec{y}) = \nabla_X G(X_k + \beta\triangle X) \circ \triangle X, \tag{12}$$

where $A \circ B = tr(AB)$. Finally we have:

$$\begin{aligned} f'_\beta(0, \vec{y}) = &\ \nabla_X G(X_k) \circ \triangle X \\ = &\ \vec{y}^T \nabla_X G(Y_k Y_k^T)\vec{y} \\ \geq &\ \vec{y}_{min}^T \nabla_X G(Y_k Y_k^T)\vec{y}_{min} = \rho_{min}. \end{aligned} \tag{13}$$

And we also know that:

$$\begin{aligned} f'_\beta(0, \vec{y}_{min}) = &\ \vec{y}_{min}^T \nabla_X G(Y_k Y_k^T)\vec{y}_{min} \\ = &\ \rho_{min} \leq 0, \end{aligned} \tag{14}$$

which means $f(\beta, \vec{y})$ is decent at $\beta = 0$ when $\vec{y} = \vec{y}_{min}$, and for any $\vec{y} \neq 0$ and $||\vec{y}|| = 1$ we have:

$$f'_\beta(0, \vec{y}) \geq f'_\beta(0, \vec{y}_{min}). \tag{15}$$

$\vec{y}_{min}$ is the steepest decent attached vector of Y_k is proved. □

After we getting the steepest decent attached vector, now we focus on how to get the steepest step size β. Actually, that is quite straight forward, let $X_k = Y_k Y_k^T$ and $\triangle X_{min} = \vec{y}_{min}\vec{y}_{min}^T$. By Equation (12), let $f'_\beta(\beta, \vec{y}_{min}) = 0$, then:

$$\begin{aligned} 0 = &\ \nabla_X G(X_k + \beta\triangle X_{min}) \circ \triangle X_{min} \\ = &\ \nabla_X G(X_k) \circ \triangle X_{min} \\ &\ +2\beta(I_\Omega \odot \triangle X_{min}) \circ \triangle X_{min}, \end{aligned} \tag{16}$$

where I_Ω is a matrix that when $(i,j) \in \Omega$, $I_\Omega(i,j) = 1$, else $I_\Omega(i,j) = 0$, and $A \odot B = C$ means $C_{i,j} = A_{i,j} \times B_{i,j}$. Then we immediately have:

$$\begin{aligned} \beta = &\ -\frac{\nabla_X G(X_k)\circ\triangle X_{min}}{2(I_\Omega\odot\triangle X_{min})\circ\triangle X_{min}} \\ = &\ -\frac{\rho_{min}}{2tr((I_\Omega\odot\triangle X_{min})\triangle X_{min})}. \end{aligned} \tag{17}$$

Now, we can summarize the eigenvector initialization method **EigenInitial**(R, $\{\sigma_k\}_{k=1}^K$) for P as follows:

1. Let $k = 0$, $P_0 = []$, $Y_0 = []$ and denote $X_0 = 0$;

2. Find $\{\rho_{min}, \vec{y}_{min}\}$ of matrix $\nabla_X G(X_k)$;
3. Compute β by Equation (17), let $k = k + 1$;
4. Let $\beta' = \frac{\sqrt{\beta}}{\sigma_k}$ when $\frac{\sqrt{\beta}}{\sigma_k} \leq \sqrt{N}$, else $\beta' = \sqrt{N}$;
5. Let $P_k = [P_{k-1}|\beta' \cdot \vec{y}_{min}]$;
6. Let $\Sigma_k = diag\{\sigma_1, ..., \sigma_k\}$, $Y_k = P_k\Sigma_k$, $X_k = Y_k Y_k^T$;
7. Repeat 2 to 7 until $k = K$;
8. Return $P = P_K$.

In addition, line 4 is to ensure the initial values of P meet the constraints of problem in Equation (6). Line 2 is to find $\{\rho_{min}, \vec{y}_{min}\}$ of matrix $\nabla_X G(X_k)$, we do not need to conduct full eigendecomposition to the matrix $\nabla_X G(X_k)$ as it is very time consuming which does not fit for sparse large scale matrix, we can use the sparse power method [Yuan and Zhang, 2013] with a shift to the original matrix $\nabla_X G(X_k)$. Let c be a constant large enough, we construct a shifted matrix $D = c \cdot I - \nabla_X G(X_k)$, then we can easily get the dominant eigen value-vector pair $\{\rho_d, \vec{y}_d\}$ of matrix D through a few power iterations (less than 20 with accuracy level 10^{-5} in most circumstances), then we get $\{\rho_{min} = c - \rho_d, \vec{y} = \vec{y}_d\}$. Finally, we summarize the FAVA method in Algorithm 1.

Algorithm 1 Scale-Invariant Matrix Factorization (FAVA).

Input:
Accuracy level ε, Incomplete matrix R;
Factorization Variances $\{\sigma_1^2, \sigma_2^2, ..., \sigma_K^2\}$;
Output: Y;
1: Let $P = EigenInitial(R, \{\sigma_k\})$;
2: Let $t = 0$, $\alpha_t = 10 \times \varepsilon$ and $P^{(t)} = P$;
3: **while** $\alpha_t \geq \varepsilon$ **do**
4: Compute $\triangle P^{(t)}$ by Equation (7);
5: Let $\alpha_{t+1} = BiSearch(P^{(t)}, \triangle P^{(t)})$;
6: Update each column of $P^{(t)}$ by Equation (9), and the result is denoted as $P^{(t+1)}$;
7: $t = t + 1$;
8: **end while**
9: **return** $Y = P^{(t)}\Sigma_K$.

4 Estimating Factorization Variances

In this section, we discuss about how to estimate the *Factorization Variances* and also the number of them, i.e.,K. Before getting into the detail, it is worth mentioning that the algorithms introduced in this section are only applied to small sampled sub-matrices for estimating suitable factorization variances through hyper-parameter tuning.

As the number of factors of non-convex methods must be set manually, we focus our discussion on using convex methods for estimating factorization variances. The most widely used two forms of convex formulation are the *trace trade-off form* [Bach *et al.*, 2008; Candes and Plan, 2010]:

$$\begin{aligned} \min_X \quad & F(X) = \sum_{(i,j)\in\Omega} (R_{i,j} - X_{i,j})^2 + \lambda tr(X) \\ s.t. \quad & X \succeq 0, \end{aligned} \tag{18}$$

and the *trace bounding form* [Hazan, 2008; Jaggi *et al.*, 2010]:

$$\begin{aligned} \min_X \quad & G(X) = \sum_{(i,j)\in\Omega} (R_{i,j} - X_{i,j})^2 \\ s.t. \quad & X \succeq 0, \\ & tr(X) \leq \gamma. \end{aligned} \tag{19}$$

Actually, as mentioned in [Jaggi *et al.*, 2010], the above two formulations are equivalent in the following sense.

Theorem 4.1. *If X is an optimal solution to Problem in Equation* (18)*, let $\gamma = tr(X)$, then X is also an optimal solution to Problem in Equation* (19)*. On the other hand, if X is an optimal solution to Problem in Equation* (19)*, let $\lambda = -\frac{\nabla_X G(X) \circ X}{tr(X)}$, then X is also an optimal solution to Problem in Equation* (18)*.*

Next, we choose the formulation form of trace trade-off in Equation (18) as an example to discuss its solution. Firstly, let $X = YY^T$, then the constraint in Equation (18) is canceled and the problem can be rewritten into a non-convex form [Mnih and Salakhutdinov, 2007] (with $\lambda = \lambda_u = \lambda_v$):

$$\min_Y \sum_{(i,j)\in\Omega} (R_{i,j} - Y_{i,:} \cdot Y_{j,:})^2 + \lambda tr(YY^T). \tag{20}$$

Then, we use the method proposed in [Journée *et al.*, 2010] to solve this problem in Equation (18) through finding the local optimum of the problem in Equation (20), and increasing the number of Y's columns one by one until the condition of the following Theorem 4.2 is met to achieve its global optimum.

Theorem 4.2. *A local minimizer Y of the non-convex problem in Equation* (20) *provides a global minimum point $X = YY^T$ of the convex problem in Equation* (18) *if and only if $S_Y = \nabla_X F(YY^T) \succeq 0$.*

After we get Y, $X = YY^T$ is the global optimum for Problem in Equation (18). Then, by the assumption in Equation (1), we can approximate the factorization variances $\{\sigma_k^2\}_{k=1}^K$ as follows:

$$\sigma_k^2 \approx \frac{1}{N} \sum_{i=1}^{N} Y_{i,k}^2, k = 1, 2, ..., K. \tag{21}$$

On the small sampled sub-matrix we can tune the best parameter λ^* for Problem in Equation (18) by cross-validation. Then, with the latent factors resulted from λ^* we can estimate the corresponding factorization variances. Hopefully, these factorization variances work well on the original large matrix. The experiments will empirically validate this.

5 Theoretical Analysis on Scale-Variant and Scale-Invariant Methods

Now, we give the theoretical analysis on the relationship between the scale-invariant formulation FAVA (proposed in this paper) and the scale-variant ones (in the form of both trace trade-off and trace bounding).

Firstly, let's take a look at the relationship between FAVA and the trace trade-off formulation. Before detailing this relationship, we need to first present the following two lemmas.

Lemma 5.1. *X is an optimal solution of problem in Equation* (18) *if and only if the following conditions hold:*

$$\begin{array}{lrcl} 1^{\#} & X & \succeq & 0, \\ 2^{\#} & S & \succeq & 0, \\ 3^{\#} & SX & = & 0, \end{array} \tag{22}$$

where $S = \nabla_X F(X)$.

Lemma 5.2. *If P is a local minimum solution of problem in Equation* (6)*, then the following conditions hold:*

$$\triangle P_{:,k} \cdot P_{:,k} \leq 0, k = 1, ..., K, \tag{23}$$

where $\triangle P$ is the gradient of objective function in Equation (6)*, and $\triangle P_{:,k}$ means the k-th column of $\triangle P$.*

The conditions in Lemma 5.1 are the KKT conditions of the problem in Equation (18) and similarly the conditions in Lemma 5.2 are the KKT conditions of the problem in Equation (6) (see [Boyd and Vandenberghe, 2009]).

Theorem 5.3. *Let $Y = P\Sigma_K$, where P is a local minimum solution of FAVA in Equation* (6)*, and $X = YY^T$. If X is an optimal solution to problem in Equation* (18) *for some λ, then we have:*

$$\lambda = -\frac{\nabla_X G(X) \circ X}{tr(X)} \geq -\frac{\nabla_X G(X) \circ X}{N \cdot \sum_{k=1}^{K} \sigma_k^2} \geq 0 \tag{24}$$

Proof. Firstly, we know that $F(X) = G(X) + \lambda tr(X)$, then by $\mathbf{3}^{\#}$ of Lemma 5.1 we know that:

$$\nabla_X G(X) X + \lambda X = 0. \tag{25}$$

Both sides of the above equation get trace, then we have:

$$\lambda = -\frac{tr(\nabla_X G(X) X)}{tr(X)} = -\frac{\nabla_X G(X) \circ X}{tr(X)}. \tag{26}$$

And by constraints in Equation (2) we also know that:

$$0 < tr(X) = tr(YY^T) \leq N \cdot \sum_{k=1}^{K} \sigma_k^2. \tag{27}$$

Next, we prove $\nabla_X G(X) \circ X \leq 0$. As $X = YY^T$ and $Y = P\Sigma_K$, then we have:

$$\begin{aligned} tr(\nabla_X G(X) X) &= tr(\nabla_X G(YY^T) YY^T) \\ &= tr(\nabla_X G(YY^T) P\Sigma_K^2 P^T) \\ &= tr(P^T \nabla_X G(YY^T) P\Sigma_K^2). \end{aligned} \tag{28}$$

By Equation (7) we have:

$$tr(\nabla_X G(X) X) = \frac{1}{2} tr(P^T \triangle P) = \frac{1}{2} \sum_{k=1}^{K} \triangle P_{:,k} \cdot P_{:,k}. \tag{29}$$

By conditions in Lemma 5.2 we have:

$$\nabla_X G(X) \circ X = tr(\nabla_X G(X) X) \leq 0. \tag{30}$$

Now we have already proved:

$$\lambda = -\frac{\nabla_X G(X) \circ X}{tr(X)} \geq -\frac{\nabla_X G(X) \circ X}{N \cdot \sum_{k=1}^{K} \sigma_k^2} \geq 0. \tag{31}$$

□

Theorem 5.3 clarifies the correlation between FAVA and the trace trade-off form for MF. It actually provides a lower bound for the parameter λ by considering the equivalence of the two problems. We can clearly see that this lower bound on λ is not only dependant on the scale of the matrix N, but also the sparsity of the incomplete matrix R (the number of non-zero terms in $\nabla_X G(X) \circ X$ is $|\Omega|$). That is the reason why the best working λ^* on a sub-matrix may not perform well on large scale matrices. Similarly, we show the correlation between FAVA and *bounding trace* formulation with the following theorem.

Theorem 5.4. *Let $Y = P\Sigma_K$, where P is a local minimum solution of FAVA in Equation* (6)*, and $X = YY^T$. If X is an optimal solution for problem in Equation* (19) *for some γ, then we have:*

$$\gamma \geq N \sum_{k=1}^{K} \sigma_k^2. \tag{32}$$

The above theorem can be got immediately by the bounding trace constraint in Equation (19), as $N \sum_{k=1}^{K} \sigma_k^2$ is the upper bound of $tr(X)$ by the constraints in Equation (6). It states that the parameter γ is also dependant on the scale of the matrix N. This is the reason why the best working γ^* on a sub-matrix may not perform well on large matrices.

6 Experiment

In this section, we validate the effectiveness and efficiency of our approach FAVA based on a real-world dataset.

Baselines. We choose two widely used convex MF methods for comparison, namely

TTF: *Trace Trade-off Form* [Bach *et al.*, 2008; Bouchard *et al.*, 2013], which is formulated in Equation (18).

TBF: *Trace Bounding Form* [Hazan, 2008; Jaggi *et al.*, 2010], which is formulated in Equation (19).

In the experiments, FAVA uses TTF to estimate the $\{\sigma_k^2\}_{k=1}^K$ by randomly drawn sub-matrix and operates on the target matrix. The convergence level of all methods is set to 10^{-5}.

Table 1: Basic Statistics of Datasets.

Dataset	m	n	#rating	Dataset	m	n	#rating
$M_{0.04,1}$	179	124	4,126	$M_{0.08,1}$	899	319	27,858
$M_{0.04,2}$	182	126	4,091	$M_{0.08,2}$	1,092	356	35,547
$M_{0.04,3}$	197	123	4,475	$M_{0.08,3}$	902	363	28,769
$M_{0.04,4}$	184	143	4,521	$M_{0.08,4}$	922	326	28,852
$M_{0.04,5}$	126	105	2,584	$M_{0.08,5}$	899	337	27,246
$M_{0.16,1}$	3,930	928	176,117				
$M_{0.16,2}$	3,422	864	144,552				
$M_{0.16,3}$	3,297	841	136,066				
$M_{0.16,4}$	3,929	875	166,827				
$M_{0.16,5}$	3,538	914	149,484				

Dataset	MovieLens10M
m	69,878
n	10,677
#rating	10,000,054

Dataset. Here we use MovieLens10M [Miller *et al.*, 2003] dataset for validation. Specifically, we randomly splited the large dataset into training set and test set (80% for training, 20% for test). All the sub-matrices was sampled from the training set. Let $M \in \mathbb{R}^{m \times n}$ be the original large scale matrix, we sampled each sub-matrix dataset as follows: given a ratio r, we randomly sampled $m \times r$ rows and $n \times r$ columns

from the M first; then we collected the elements that are both covered by the sampled rows and columns into a set Ω; finally, we removed rows and columns whose numbers of elements in Ω are less than 10, and the remaining rows and columns and their covering elements in Ω formed our resulting sub-matrix dataset. For each ratio r, we sampled 5 sub-matrices independently, and the id-th sample is denoted as $\mathrm{M}_{r,id}$. The statistics of datasets are summarized in Table 1. Unless otherwise noted, all methods conducted 5-fold cross-validation when they run on the sub-matrix datasets.

Overall comparison on sub-matrices. In this experiment, we tuned parameters for TTF and TBF on the sub-matrices first, let λ variate from 0.1 to 4.0 by step size 0.1, and $\gamma \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000\}$. The left part of Table 2 shows the parameter tuning results. Then we used the best parameters estimated by $\mathrm{M}_{0.04,1}$ for all the methods, and let them run on all the sub-matrices, results are shown in the right part of Table 2. Figure 2 shows the total time of parameter tuning for both TTF and TBF methods.

As we can see from the right part of Table 2, for effectiveness comparison, FAVA performs the best across all sub-matrices. While for efficiency comparison, TTF method has superiority when the scale of sub-matrices is small (e.g., $\mathrm{M}_{0.04,*}$ series sub-matrices). However, when the scale of sub-matrices become larger, the efficiency of TTF becomes worse rapidly under the parameter setting that are estimated by $\mathrm{M}_{0.04,1}$. In contrast, the efficiencies of TBF and FAVA are always comparable across all sub-matrices. It is worth to point out that although the results of FAVA are not as good as the best results of TTF and TBF on $\mathrm{M}_{0.08,*}$ and $\mathrm{M}_{0.16,*}$ series sub-matrices, together with Figure 2 we can see that the amount of running time of FAVA to achieve such results (i.e., TTF parameter tuning time on $\mathrm{M}_{0.04,1}$ + FAVA running time) is much less than the parameter tuning time of TTF and TBF on these sub-matrices. Also, from Figure 2 we can find the parameter tuning time grows enormously as the matrix scale increases, which indicates tuning parameter on large-scale matrix is very time consuming.

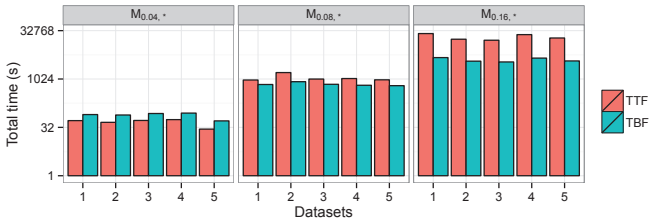

Figure 2: Parameters Tuning Time.

Comparison on MovieLens10M. In this experiment, we used the best parameters estimated by $\mathrm{M}_{0.04,1}$ for all the methods, then let they run on the original large-scale MovieLens10M matrix (results are shown in Table 3). Then, we used the $\mathrm{M}_{0.04,1}$, $\mathrm{M}_{0.08,1}$ and $\mathrm{M}_{0.16,1}$ to estimate 3 different sets of $\{\sigma_k^2\}_{k=1}^K$, and feed them into our FAVA method for MF on MovieLens10M dataset (results are shown in Table 4).

Table 3: Baselines Comparison.

$\mathrm{TTF}_{\lambda=1.0}$		$\mathrm{TBF}_{\gamma=70.0}$		FAVA	
RMSE	Time	RMSE	Time	RMSE	Time
0.92111	11516m5.600s	1.0513	48m5.631s	**0.81530**	95m21.948s

From Table 3 we can see that, once again FAVA plays the best on effectiveness, i.e., its RMSE is much better than that of baselines. The efficiencies of TBF and FAVA are comparable, while that of TTF on MovieLens10M dataset is much worse than the other two methods. Moreover, from Table 4 we can see that the RMSE is almost unchange as the scale of sub-matrices changes, however, the running time of FAVA increases as the scale of sub-matrices becomes large. That is because when the scale of sub-matrices becomes larger and larger, the structure of matrix becomes more and more complex and the TTF method needs more latent factors to capture them, which leads to more computation cost for FAVA on the original MovieLens10M matrix.

Table 4: Sub-Matrices Estimation Test for FAVA.

$\mathrm{M}_{0.04,1}$		$\mathrm{M}_{0.08,1}$		$\mathrm{M}_{0.16,1}$	
RMSE	Time	RMSE	Time	RMSE	Time
0.81530	95m21.948s	0.81513	180m26.588s	0.81885	322m18.697s

7 Related Work

We briefly review works about MF in terms of collaborative filtering here. Generally speaking, the MF methods can be divided into two categories, Bayesian (e.g., [Lim and Teh, 2007; Salakhutdinov and Mnih, 2008; Freudenthaler *et al.*, 2011; Rendle, 2013] and non-Bayesian (e.g., [Mnih and Salakhutdinov, 2007; Pilászy *et al.*, 2010; Rendle, 2010]) methods. The main difference between these two kinds of methods is that the Baysian methods introduce Baysian inference to the inferring of latent factors, while the non-Bayesian methods give MAP approximation to them. The method proposed in this paper belongs to the non-Bayesian method. All these methods can achieve good performance through carefully tuning their hyper-parameters. However, the cost of hyper-parameters tuning is ignored.

Although there are Baysian methods that claim they are non-parametric [Blei *et al.*, 2010; Ding *et al.*, 2010; Xu *et al.*, 2012], they are only non-parametric on how to set the number of latent factors, they still have to tune hyper-parameters (e.g., regularization parameters, prior sampling distribution parameters) to a achieve good performance. Actually, the non-Baysian convex MF methods [Bach *et al.*, 2008; Jaggi *et al.*, 2010; Journée *et al.*, 2010; Bouchard *et al.*, 2013] also can determine the number of latent factors automatically.

The proposed method is different from all these works. Firstly, to the best of our knowledge, we are the first to formulate a problem of MF with scale-invariant parameters. Secondly, we propose a new optimization problem called FAVA for scale-invariant parametric MF, and solve it both effectively and efficiently. Thirdly, we present the theoretical properties show that the previous MF methods (in the form of both trace trade-off and trace bounding) are scale-variant.

Table 2: Overall Comparison on Sub-Matrices.

Dataset	Parameter Tuning				Parameters Estimated by $M_{004,1}$					
	TTF		TBF		$TTF_{\lambda=1.0}$		$TBF_{\gamma=70.0}$		FAVA	
	λ_{best}	RMSE	γ_{best}	RMSE	RMSE	Time (s)	RMSE	Time (s)	RMSE	Time (s)
$M_{004,1}$	1.0	0.91507±0.02324	70.0	0.91689±0.02283	0.91507±0.02324	1.656	0.91689±0.02283	4.276	**0.90697±0.02317**	2.466
$M_{004,2}$	1.0	0.89232±0.02154	60.0	0.89327±0.02207	0.89232±0.02154	1.538	0.89342±0.02182	4.204	**0.88595±0.01709**	2.382
$M_{004,3}$	1.1	0.90167±0.02855	70.0	0.90265±0.02851	0.90213±0.02877	1.679	0.90265±0.02851	4.685	**0.89793±0.03421**	2.590
$M_{0.04,4}$	1	0.90275±0.01600	80.0	0.90511±0.01606	0.90275±0.01600	1.818	0.90554±0.01568	4.78	**0.89171±0.01470**	2.531
$M_{0.04,5}$	0.9	0.96147±0.03611	60.0	0.96256±0.03636	0.96219±0.03662	0.830	0.96418±0.03567	2.767	**0.95109±0.03162**	1.294
$M_{0.08,1}$	1.6	0.89540±0.00678	300.0	0.89663±0.00707	0.91132±0.00640	37.66	0.93555±0.00872	30.816	**0.90303±0.00387**	25.438
$M_{0.08,2}$	1.7	0.88941±0.00520	400.0	0.89083±0.00557	0.89588±0.00548	71.006	0.93697±0.00424	36.022	**0.88909±0.00387**	35.524
$M_{0.08,3}$	1.7	0.89779±0.00748	300.0	0.89910±0.00714	0.91524±0.00700	40.108	0.93447±0.00831	33.47	**0.90663±0.00000**	26.953
$M_{0.08,4}$	1.7	0.90125±0.00616	300.0	0.90278±0.00600	0.91768±0.00520	40.927	0.94458±0.00700	28.437	**0.90581±0.00424**	25.942
$M_{0.08,5}$	1.7	0.91429±0.00995	300.0	0.91552±0.00990	0.92521±0.00975	36.897	0.94752±0.01082	27.05	**0.92075±0.00906**	24.901
$M_{0.16,1}$	2.6	0.87246±0.00200	1000.0	0.87565±0.00200	0.91254±0.00200	948.059	0.99213±0.00245	174.225	**0.88960±0.00245**	233.074
$M_{0.16,2}$	2.4	0.86465±0.00283	1000.0	0.86686±0.00283	0.90014±0.00283	613.385	0.97799±0.00424	140.29	**0.88712±0.00141**	174.156
$M_{0.16,3}$	2.3	0.88353±0.00173	1000.0	0.88571±0.00200	0.91835±0.00245	597.935	0.99485±0.00245	129.596	**0.90078±0.00447**	180.950
$M_{0.16,4}$	2.5	0.87162±0.00346	1000.0	0.87421±0.00374	0.90975±0.00346	854.419	0.99198±0.00424	163.595	**0.89998±0.00316**	256.393
$M_{0.16,5}$	2.4	0.88460±0.00316	1000.0	0.88669±0.00332	0.92096±0.00316	687.802	0.99236±0.00245	144.217	**0.90935±0.00500**	222.723

8 Conclusion

In this paper, we proposed a *scale-invariant* parametric MF method for addressing the idea that tuning hyper-parameters on small sub-matrix and then use them on the original large scale matrix. Specifically, we can use any of the previous MF methods to estimate the best working factorization variances on small sampled sub-matrix, and then use them for the proposed method to conduct both effective and efficient MF on original large scale matrix. Extensive experiments also show that the proposed method can achieve good performance on the original large scale matrix with the estimated factorization variances on small sampled sub-matrix.

Acknowledgments

This research was partially supported by grants from the National Science Foundation for Distinguished Young Scholars of China (Grant No. 61325010), the National High Technology Research and Development Program of China (Grant No. 2014AA015203), the Fundamental Research Funds for the Central Universities of China (Grant No. WK2350000001) and the Natural Science Foundation of China (Grant No. 61403358). Ping Luo was supported by the National Natural Science Foundation of China (No. 61473274) and National 863 Program (No.2014AA015105).

References

[Bach *et al.*, 2008] Francis Bach, Julien Mairal, and Jean Ponce. Convex sparse matrix factorizations. *arXiv preprint arXiv:0812.1869*, 2008.

[Blei *et al.*, 2010] David M Blei, Perry R Cook, and Matthew Hoffman. Bayesian nonparametric matrix factorization for recorded music. In *ICML '10*, pages 439–446, 2010.

[Bouchard *et al.*, 2013] Guillaume Bouchard, Dawei Yin, and Shengbo Guo. Convex collective matrix factorization. In *AISTATS '13*, pages 144–152, 2013.

[Boyd and Vandenberghe, 2009] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2009.

[Candes and Plan, 2010] Emmanuel J Candes and Yaniv Plan. Matrix completion with noise. *Proceedings of the IEEE*, 98(6):925–936, 2010.

[Candès and Recht, 2009] Emmanuel J Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717–772, 2009.

[Chan *et al.*, 2013] Simon Chan, Philip Treleaven, and Licia Capra. Continuous hyperparameter optimization for large-scale recommender systems. In *BigData '13*, pages 350–358, 2013.

[Ding *et al.*, 2010] Nan Ding, Rongjing Xiang, Ian Molloy, Ninghui Li, et al. Nonparametric bayesian matrix factorization by power-ep. In *AISTATS '10*, pages 169–176, 2010.

[Freudenthaler *et al.*, 2011] Christoph Freudenthaler, Lars Schmidt-Thieme, and Steffen Rendle. Bayesian factorization machines. 2011.

[Ge *et al.*, 2011] Yong Ge, Qi Liu, Hui Xiong, Alexander Tuzhilin, and Jian Chen. Cost-aware travel tour recommendation. In *KDD '11*, pages 983–991, 2011.

[Hazan, 2008] Elad Hazan. Sparse approximate solutions to semidefinite programs. In *LATIN 2008: Theoretical Informatics*, pages 306–316. Springer, 2008.

[Jaggi *et al.*, 2010] Martin Jaggi, Marek Sulovsk, et al. A simple algorithm for nuclear norm regularized problems. In *ICML '10*, pages 471–478, 2010.

[Journée *et al.*, 2010] Michel Journée, Francis Bach, P-A Absil, and Rodolphe Sepulchre. Low-rank optimization on the cone of positive semidefinite matrices. *SIAM Journal on Optimization*, 20(5):2327–2351, 2010.

[Koren *et al.*, 2009] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[Koren, 2008] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD '08*, pages 426–434, 2008.

[Lim and Teh, 2007] Yew Jin Lim and Yee Whye Teh. Variational bayesian approach to movie rating prediction. In *Proceedings of KDD Cup and Workshop*, volume 7, pages 15–21, 2007.

[Miller *et al.*, 2003] Bradley N Miller, Istvan Albert, Shyong K Lam, Joseph A Konstan, and John Riedl. Movielens unplugged: experiences with an occasionally connected recommender system. In *IUI '03*, pages 263–266, 2003.

[Mnih and Salakhutdinov, 2007] Andriy Mnih and Ruslan Salakhutdinov. Probabilistic matrix factorization. In *NIPS '07*, pages 1257–1264, 2007.

[Pilászy *et al.*, 2010] István Pilászy, Dávid Zibriczky, and Domonkos Tikk. Fast als-based matrix factorization for explicit and implicit feedback datasets. In *RecSys '10*, pages 71–78, 2010.

[Rendle, 2010] Steffen Rendle. Factorization machines. In *ICDM '10*, pages 995–1000. IEEE, 2010.

[Rendle, 2013] Steffen Rendle. Scaling factorization machines to relational data. In *VLDB '13*, pages 337–348, 2013.

[Rennie and Srebro, 2005] Jasson DM Rennie and Nathan Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *ICML '05*, pages 713–719, 2005.

[Salakhutdinov and Mnih, 2008] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *ICML '08*, volume 25, 2008.

[Srebro *et al.*, 2003] Nathan Srebro, Tommi Jaakkola, et al. Weighted low-rank approximations. In *ICML '03*, volume 3, pages 720–727, 2003.

[Wu *et al.*, 2012] Le Wu, Enhong Chen, Qi Liu, Linli Xu, Tengfei Bao, and Lei Zhang. Leveraging tagging for neighborhood-aware probabilistic matrix factorization. In *CIKM '11*, pages 1854–1858, 2012.

[Xu *et al.*, 2012] Minjie Xu, Jun Zhu, and Bo Zhang. Nonparametric max-margin matrix factorization for collaborative prediction. In *NIPS '12*, pages 64–72, 2012.

[Yuan and Zhang, 2013] Xiao-Tong Yuan and Tong Zhang. Truncated power method for sparse eigenvalue problems. *The Journal of Machine Learning Research*, 14(1):899–925, 2013.

[Zhu *et al.*, 2014] Hengshu Zhu, Enhong Chen, Hui Xiong, Kuifei Yu, Huanhuan Cao, and Jilei Tian. Mining mobile user preferences for personalized context-aware recommendation. *ACM Trans. Intell. Syst. Technol.*, 5(4):58:1–58:27, December 2014.