*2025 International Conference on Intelligent Computing*
*July 26-29, Ningbo, China*
https://www.ic-icc.cn/2025/index.php

# BPMFARD: A Bayesian Probabilistic Matrix Factorization Algorithm with Automatic Rank Determination in Recommender Systems

Hao Wang[1[0009-0008-7185-9118]], Junfeng Yan[1[0009-0009-6294-2021]], Jingyuan Xiao[3], Guangzhi Qu[2[0000-0003-4047-9514]], and Feng Zhang[1(✉)[0000-0002-0506-9440]]

[1] China University of Geosciences, 388 Lumo Road, Wuhan, Hubei 430074, China
[2] Oakland University, 201 Meadow Brook Rd, Rochester, MI 48309-4401, USA
[3] University of Wisconsin–Madison, 500 Lincoln Dr, Madison, WI 53706, USA

**Abstract.** Matrix factorization is a prevalent and effective technique for building recommender systems. However, traditional matrix factorization methods demand manual setting and tuning of hyperparameters, like regularization coefficients, learning rates, and the dimension of the feature matrix (rank). To automate this, we introduce BPMFARD, a Bayesian probabilistic matrix factorization algorithm with automatic rank determination. By setting a prior distribution for factor matrices and devising an effective parameter elimination strategy, BPMFARD enables automatic parameter adjustment during training to significantly alleviate overfitting, enhancing recommendation accuracy. Experiments with benchmark datasets show that BPMFARD outperforms the benchmark methods. Since matrix factorization can be seen as a simple neural network, the rank determination strategy in matrix factorization may provide a valuable and interesting research perspective for the embedding size learning in neural collaborative filtering.

**Keywords:** Recommender Systems, Matrix Factorization, Automatic Rank Determination, Bayesian Inference, Optimization.

## 1 Introduction

Recommender systems are essential in today's internet, aiding users in navigating overwhelming choices across music, books, videos, products, and more. They power major platforms like Amazon, Netflix, Facebook, Pandora, and YouTube, assisting users make informed decisions.

A multitude of design methodologies exist for recommender systems, encompassing content-based filtering [1], collaborative filtering (CF) [2,3]and hybrid methods[4]. Among these, CF is widely recognized for its effectiveness, predicting user preferences by leveraging the ratings and behaviors of others.

CF generates personalized recommendations by leveraging similarities between users or items and encompasses several variants, including user-based CF [5], item-based CF [6] and matrix factorization approaches [7,8].

Matrix factorization reduces data dimensionality by embedding users and items into lower-dimensional vector spaces. This technique augments CF by uncovering the latent features of users and items, thereby transcending mere reliance on their direct interactions.

Various matrix factorization techniques have been developed. Early methods include gradient descent-based algorithms, notably Funk SVD [9] and BiasSVD [10]. Subsequent advancements introduced probabilistic frameworks, including Probabilistic Matrix Factorization (PMF) [11] and Bayesian Probabilistic Matrix Factorization (BPMF) [12]. Additionally, Nonnegative Matrix Factorization (NMF) [13] provides an alternative by constraining factorized components to nonnegative values.

A key challenge in these methodologies lies in the necessity to pre-determine hyperparameters, such as learning rates, regularization coefficients, and crucially, the dimensionality of the feature matrix (rank). The selection of rank has a substantial influence on model performance. Hyperparameter tuning, typically conducted via cross-validation, is both time-consuming and resource-demanding [14].

To address these challenges, we introduce BPMFARD: a Bayesian Probabilistic Matrix Factorization algorithm integrated with Automatic Rank Determination for recommender systems. Its main contributions are summarized as follows:

(1) Unlike existing CF recommender systems that utilize a fixed, pre-set rank for matrix factorization, BPMFARD automatically determines the optimal rank during training. By deactivating non-informative dimensions, it adeptly addresses the challenges posed by the high sparsity and dynamic nature of user-item interactions, substantially enhancing recommendation accuracy at scale.

(2) Unlike traditional matrix factorization methods that lack the ability to dynamically adjust parameters and are prone to overfitting in large-scale recommender systems, BPMFARD employs a parameter elimination strategy to discard near-zero columns while also avoiding the need for additional tuning of hyperparameters such as learning rate or regularization factor, thereby more effectively mitigating overfitting.

## 2 Related work

Matrix factorization gained popularity in recommender systems after the Netflix Prize in 2006, proven effective for large, sparse datasets [15,16]. Key algorithms, including NMF [13], Funk SVD [9], BiasSVD [10], minimized loss functions involving squared error and regularization terms. However, tuning hyperparameters like regularization coefficients and learning rates often necessitates manual trial and error, which is time-consuming.

To address these challenges, Salakhutdinov et al. [12] introduced a fully Bayesian model based on probabilistic matrix factorization (PMF). The model automatically manages its capacity by integrating all parameters and hyperparameters, trained using Markov Chain Monte Carlo (MCMC) for enhanced predictive accuracy. However, it

still needs predefined factor matrix dimensions despite eliminating the need for regularization parameters and learning rates.

Anelli et al. [14] studied how factor matrix dimension, learning rate, and iterations affect experimental results. They discovered that the factor matrix rank has the greatest impact, especially in the MovieLens dataset. This finding highlights the significance of rank determination during hyperparameter tuning.

Chan et al. [17] adapted hyperparameters to evolving business dynamics, improving large-scale recommender system accuracy. Beutel et al. [18] introduced an impact function that optimizes predictions based on hyperparameters. However, both still required multiple attempts at parameter tuning.

Tipping et al. [19] proposed Bayesian Principal Component Analysis (Bayesian PCA), which employs a probabilistic generative latent variable model to automatically determine the number of principal components, thus addressing the limitation of traditional PCA requiring manual dimensionality specification. The concepts and framework introduced in their work have laid a significant foundation for subsequent research on automatic rank determination in the field of matrix factorization.

Tan et al. [20] proposed a Bayesian $\beta$-NMF method based on automatic relevance determination (ARD), introducing shared scale parameters in the prior to automatically determine the latent dimensionality of NMF. They developed efficient MM algorithms for parameter inference, achieving good results on multiple datasets. However, due to the non-negativity constraint on factor matrices, this approach cannot address rank determination in models that allow negative values, such as Funk SVD and BiasSVD.

Zhao et al. [21] used a hierarchical probabilistic model for CP decomposition with sparse priors on latent factors and hyperpriors on hyperparameters for automatic rank determination. Their Bayesian inference algorithm scales linearly with data size, enabling parameter-free inference of low-rank latent factors and predictive distributions for missing data. However, this method is tailored for image processing with multi-dimensional tensors, not directly applicable to recommender systems using matrix factorization for two-dimensional user-item interaction matrices.

In this study, we integrated techniques from prior research to achieve, for the first time, efficient recommendations in matrix factorization-based systems without the need for selecting model training parameters or hyperparameters.

## 3 Preliminaries

### 3.1 Notions

Following the notation conventions from [22], we denote matrices with bold uppercase letters (e.g., $\mathbf{X}$), vectors with bold lowercase letters (e.g., $\mathbf{x}$), and scalars with lowercase letters (e.g., $x$). The inverse and transpose of a matrix are represented as $\mathbf{X}^{-1}$ and $\mathbf{X}^{T}$, respectively. The rating matrix in the recommender system is denoted as $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2}$, where $I_1$ and $I_2$ represent the numbers of users and items, respectively. Since the rating matrix is incomplete, we define $\Omega$ as the set of observed ratings and $\bar{\Omega}$ as the set of missing ratings, each containing the corresponding indices. Here, $M$ denotes the

number of observed ratings. Additionally, we introduce the indicator matrix $\mathbf{O}$, where $\mathbf{O}_{ij} = 1$ if user $i$ has rated item $j$, and $\mathbf{O}_{ij} = 0$ otherwise.

Matrix factorization in recommender systems decomposes the rating matrix $\mathbf{A}$ into low-dimensional user and item feature matrices. Specifically, we represent the user feature matrix as $\mathbf{U} \in \mathbb{R}^{I_1 \times R}$ and the item feature matrix as $\mathbf{V} \in \mathbb{R}^{I_2 \times R}$. The feature vectors of user $i$ and item $j$ are denoted as $\mathbf{U}_i \in \mathbb{R}^{R \times 1}$ and $\mathbf{V}_j \in \mathbb{R}^{R \times 1}$, respectively. Here, $R$ represents the rank of the original matrix $\mathbf{A}$ and determines the dimensionality of the feature space. The predicted rating of user $i$ for item $j$ is given by $\mathbf{X}_{ij} = \langle \mathbf{U}_i, \mathbf{V}_j \rangle$, where $\langle \cdot, \cdot \rangle$ denotes the inner product, defined as $\langle \mathbf{u}, \mathbf{v} \rangle = \sum_k u_k v_k$.

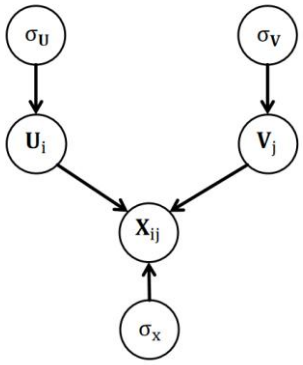



**Fig. 1.** Graphical model of PMF.

### 3.2 Probilistic Matrix Factorization

Probability Matrix Factorization (PMF) [11] integrates probabilistic graphical models with matrix factorization to predict user-item ratings. It models the relationships between users and items as probability distributions and leverages these distributions' parameters for rating predictions. The distributions can be Gaussian, Poisson, or any other appropriate type, with their parameters typically learned via maximum likelihood estimation or Bayesian inference.

In PMF, as depicted in Fig. 1, user ratings for items, denoted as $\mathbf{X}$, are presumed to adhere to a Gaussian distribution. Similarly, the user feature matrix $\mathbf{U}$ and the item feature matrix $\mathbf{V}$ are also assumed to follow Gaussian distributions. Consequently, their conditional probability distributions can be readily expressed as:

$$
\begin{aligned}
p(\mathbf{X}|\mathbf{U}, \mathbf{V}, \sigma_{\mathbf{X}}) &= \prod_{i=1}^{I_1} \prod_{j=1}^{I_2} \left[ \mathcal{N}\left(\mathbf{X}_{ij} \middle| \mathbf{U}_i^T \mathbf{V}_j, \sigma_{\mathbf{X}}^{-1}\right) \right]^{\mathbf{O}_{ij}} \\
p(\mathbf{U}|\sigma_{\mathbf{U}}) &= \prod_{i=1}^{I_1} \mathcal{N}(\mathbf{U}_i|0, \sigma_{\mathbf{U}}^{-1}\mathbf{I}) \\
p(\mathbf{V}|\sigma_{\mathbf{V}}) &= \prod_{j=1}^{I_2} \mathcal{N}\left(\mathbf{V}_j \middle| 0, \sigma_{\mathbf{V}}^{-1}\mathbf{I}\right),
\end{aligned}
\tag{1}
$$

Here, $\mathcal{N}(x|\mu,\sigma^{-1})$ represents a Gaussian distribution, where $\mu$ represents the mean and $\sigma$ represents the precision. $\sigma_{\mathbf{X}}$, $\sigma_{\mathbf{U}}$, and $\sigma_{\mathbf{V}}$ control the precision of the observed ratings, user features, and item features, respectively. Using Bayesian inference, we can then derive the posterior distributions of each parameter:

$$p(\mathbf{U},\mathbf{V}|\mathbf{X},\sigma_{\mathbf{X}},\sigma_{\mathbf{U}},\sigma_{\mathbf{V}}) \propto p(\mathbf{X}|\mathbf{U},\mathbf{V},\sigma_{\mathbf{X}})p(\mathbf{U}|\sigma_{\mathbf{U}})p(\mathbf{V}|\sigma_{\mathbf{V}}) \tag{2}$$

# 4 Bayesian Probabilistic Matrix Factorization with Automatic Rank Determination

BPMF models matrix factorization as a probabilistic process, using Bayesian inference to estimate the distribution of the latent factor matrix. It treats the rating matrix as observed, assuming each user-item rating is generated by a probability distribution over hidden factors. Parameters are treated as random variables, and Variational Inference is used to estimate the posterior distribution of the latent factor vectors. Next, Subsection 4.1 introduces the model, while Subsection 4.2 outlines the procedure for learning the model parameters.

## 4.1 Probabilistic Model and Priors

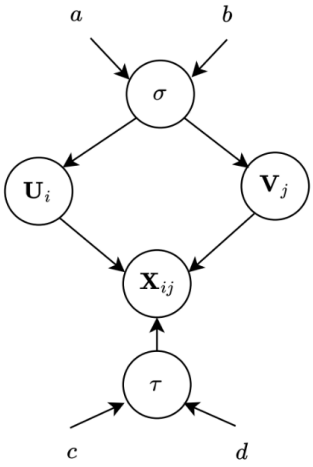



**Fig. 2.** The graphical model of the method proposed in this paper.

The Bayesian probability matrix model proposed in this paper, as shown in Fig. 2, assumes that the prior distributions of user and item feature matrices follow Gaussian distributions. Here, the hyperparameter $\boldsymbol{\sigma} = [\sigma_1, \sigma_2, \ldots, \sigma_R]$ represents the diagonal elements of the covariance matrix in the multivariate Gaussian distribution of $\mathbf{U}_i$ or $\mathbf{V}_i$. Each $\sigma_i$ controls the corresponding column of the factor matrices $\mathbf{U}$ and $\mathbf{V}$. This design ensures that by varying $\sigma$, one can automatically determine the rank by effectively

deleting a column in all factor matrices. Next, we can easily derive the conditional probability formulas for the user feature matrix **U** and the item feature matrix **V**.

$$\begin{aligned} p(\mathbf{U}|\boldsymbol{\sigma}) &= \prod_{i=1}^{I_1} \mathcal{N}(\mathbf{U}_i|0, \boldsymbol{\Lambda}^{-1}) \\ p(\mathbf{V}|\boldsymbol{\sigma}) &= \prod_{j=1}^{I_2} \mathcal{N}\left(\mathbf{V}_j\middle|0, \boldsymbol{\Lambda}^{-1}\right), \end{aligned} \tag{3}$$

where $\boldsymbol{\Lambda}$ represents the precision matrix, and $\boldsymbol{\Lambda} = \mathrm{diag}(\boldsymbol{\sigma})$.

We further set a Gamma prior for $\sigma$ as:

$$p(\boldsymbol{\sigma}) = \prod_{r=1}^{R} Ga(\sigma_r|a_0^r, b_0^r), \tag{4}$$

where $\mathrm{Ga}(x|a, b)$ denotes the Gamma distribution, given by $Ga(x|a, b) = \frac{b^a x^{a-1} e^{-bx}}{\Gamma(a)}$, with $\Gamma(\cdot)$ being the Gamma function.

Additionally, we set a Gamma prior to the noise parameter $\tau$ to fully implement the Bayesian approach throughout the process. The prior for $\tau$ is defined as:

$$p(\tau) = \mathrm{Ga}(\tau|c, d) = \frac{d^c \tau^{c-1} e^{-d\tau}}{\Gamma(c)}. \tag{5}$$

Therefore, based on Fig. 2, the complete joint distribution is given by:

$$p(\mathbf{X}_\Omega, \mathbf{U}, \mathbf{V}, \boldsymbol{\sigma}, \tau) = p(\mathbf{X}_\Omega|\mathbf{U}, \mathbf{V}, \tau) p(\mathbf{U}|\boldsymbol{\sigma}) p(\mathbf{V}|\boldsymbol{\sigma}) p(\boldsymbol{\sigma}) p(\tau). \tag{6}$$

### 4.2 Model Learning via Bayesian Inference

Our next objective is to infer the posterior distributions of the variables **U**, **V**, $\sigma$, and $\tau$ given the observed data. We denote the set of parameters as $\vartheta = \{\mathbf{U}, \mathbf{V}, \sigma, \tau\}$. The posterior distribution of parameter $\vartheta$ is given as follows:

$$p(\vartheta|\mathbf{X}_\Omega) = \frac{p(\vartheta, \mathbf{X}_\Omega)}{\int p(\vartheta, \mathbf{X}_\Omega) d\vartheta}. \tag{7}$$

Once the posterior distributions of the variables are determined, the missing values $\mathbf{X}_{ij}$ in $\mathbf{X}_{\bar{\Omega}}$ can be estimated using the inner product of $\mathbf{U}_i$ and $\mathbf{V}_j$.

Due to the complexity of the posterior distributions, precisely evaluating this predictive distribution is challenging, necessitating the use of approximate inference methods. We employ variational methods [23] to infer model parameters.

Our objective is to find a distribution $q(\vartheta)$ that closely approximates the true posterior distribution $p(\vartheta, \mathbf{X}_\Omega)$ by minimizing the Kullback-Leibler (KL) divergence, given by:

$$\begin{aligned}\min_{\vartheta} KL(q(\vartheta)||p(\vartheta|\mathbf{X}_\Omega)) &= \int q(\vartheta) ln \frac{q(\vartheta)}{p(\vartheta|\mathbf{X}_\Omega)} d\vartheta \\ &= ln\, p(\mathbf{X}_\Omega) - \int q(\vartheta) ln \frac{p(\mathbf{X}_\Omega, \vartheta)}{q(\vartheta)} d\vartheta,\end{aligned} \tag{8}$$

where $p(\mathbf{X}_\Omega)$ denotes the model evidence, while $lb = \int q(\vartheta) ln \frac{p(\mathbf{X}_\Omega, \vartheta)}{q(\vartheta)} d\vartheta$ represents the lower bound. Since the model evidence is constant, minimizing Equation (8) is equivalent to maximizing $lb$.

To facilitate the computations, we introduce the mean-field variational family [23]. It assumes independence among latent variables, implying that within the distribution $q(\vartheta)$ we seek, the variables mutually independent. Despite its seemingly restrictive nature, this assumption is broadly applicable. For scenarios where hidden variables exhibit genuine correlations, we can enhance our model by clustering correlated variables and representing each cluster as a product of joint distributions.

$$q(\vartheta) = q_{\boldsymbol{\sigma}}(\boldsymbol{\sigma}) q_\tau(\tau) q_{\mathbf{U}}(\mathbf{U}) q_{\mathbf{V}}(\mathbf{V}). \tag{9}$$

Upon establishing these foundational assumptions, we proceed with the Coordinate Ascent Variational Inference (CAVI) technique. The update rules for the variables are formulated as:

$$q_j^*(\vartheta_j) \propto e^{\mathbb{E}_{\vartheta - \vartheta_j}[ln\, p(\vartheta, \mathbf{X}_\Omega)]}, \tag{10}$$

Here, $E_{\vartheta - \vartheta_j}[\cdot]$ calculates the expectation of the distribution with respect to all variables except $\vartheta_j$.

**update feature matrices.** Leveraging Equation (10), we formulate the posterior distribution of the factor matrix **U** as:

$$q_{\mathbf{U}}(\mathbf{U}) = \prod_{i=1}^{I_1} \mathcal{N}\left(\mathbf{U}_i \middle| \widetilde{\mathbf{U}}_i, \boldsymbol{\Sigma}_i^{\mathbf{U}}\right). \tag{11}$$

The variance and mean of its posterior distribution can be updated using the following formulas:

$$\begin{aligned}\tilde{\Sigma}_i^U &= \left(\mathbb{E}_q(\tau) \mathbb{E}_q\left[\mathbf{V}_{s(\mathbf{O}_i=1)}^T \mathbf{V}_{s(\mathbf{O}_i=1)}\right] + \mathbb{E}_q[\boldsymbol{\Lambda}]\right)^{-1}, \\ \widetilde{\mathbf{U}}_i &= \tilde{\Sigma}_i^{\mathbf{U}} \mathbb{E}_q[\tau] \mathbb{E}_q\left[\mathbf{V}_{s(\mathbf{O}_i=1)}^T\right] vec\left(\mathbf{X}_{s(\mathbf{O}_i=1)}\right).\end{aligned} \tag{12}$$

Concurrently, the posterior distribution of the factor matrix **V** is formulated as:

$$q_{\mathbf{V}}(\mathbf{V}) = \prod_{j=1}^{I_2} \mathcal{N}\left(\mathbf{V}_j \middle| \widetilde{\mathbf{V}}_j, \boldsymbol{\Sigma}_j^{\mathbf{V}}\right), \tag{13}$$

and its parameters are updated as:

$$\begin{aligned}\tilde{\Sigma}_j^V &= \left(\mathbb{E}_q(\tau)\mathbb{E}_q\left[\mathbf{U}_{s(\mathbf{O}_{j=1})}^T\mathbf{V}_{s(\mathbf{O}_{j=1})}\right] + \mathbb{E}_q[\boldsymbol{\Lambda}]\right)^{-1},\\ \tilde{\mathbf{V}}_i &= \tilde{\Sigma}_i^{\mathbf{U}}\mathbb{E}_q[\tau]\mathbb{E}_q\left[\mathbf{U}_{s(\mathbf{O}_{j=1})}^T\right] vec\left(\mathbf{X}_{s(\mathbf{O}_{j=1})}\right).\end{aligned} \tag{14}$$

where $s(\mathbf{O}_{i*} = 1)$ denotes the subset of column indices in $\mathbf{O}$ for which the elements in the 1. $(\cdot)_{s(\mathbf{O}_i=1)}$ denotes the matrix formed by the columns corresponding to this subset. Similarly, $(\cdot)_{s(\mathbf{O}_i=1)}$ signifies the matrix composed of the rows where the $j$-th column's elements are 1.

Calculating $\mathbb{E}_q[\mathbf{U}_{s(\mathbf{O}_{j=1})}^T\mathbf{U}_{s(\mathbf{O}_{j=1})}]$ presents a significant challenge. However, as stated in Theorem 3.1 of [21], its expectation can be reformulated as follows.

$$\begin{aligned}\mathbb{E}_q\left[\mathbf{U}_{s(\mathbf{O}_{j=1})}^T\mathbf{U}_{s(\mathbf{O}_{j=1})}\right] &= \sum_{\mathbf{u}_i\in\mathbf{U}_{s(\mathbf{O}_{j=1})}} \mathbb{E}[\mathbf{u}_i\mathbf{u}_i^T]\\ &= \sum_{\mathbf{u}_i\in\mathbf{U}_{s(\mathbf{O}_{j=1})}} \left(\mathbb{E}[\mathbf{u}_i]\mathbb{E}[\mathbf{u}_i^T] + Var(\mathbf{u}_i)\right).\end{aligned} \tag{15}$$

In summary, the previously challenging computation of complex terms, such as $\mathbb{E}_q[\mathbf{U}_{s(\mathbf{O}_{j=1})}^T\mathbf{U}_{s(\mathbf{O}_{j=1})}]$ and $\mathbb{E}_q[\mathbf{U}_{s(\mathbf{O}_{i=1})}^T\mathbf{U}_{s(\mathbf{O}_{i=1})}]$, is now feasible. The posterior distribution of the factor matrices can be efficiently updated using Equations (12) and (14).

**update hyperparameters $\boldsymbol{\sigma}$.** Similarly, through Equation (10), we can obtain the posterior distribution for $\sigma$ as follows:

$$q_\sigma(\sigma) = \prod_{r=1}^{R} Ga\left(\sigma_r\middle|\tilde{a}_r, \tilde{b}_r\right), \tag{16}$$

where $\tilde{a}_r$ and $\tilde{b}_r$ represent the parameters for the posterior distribution of $\sigma_r$, which are updated as:

$$\begin{aligned}\tilde{a}^r &= a_0^r + \frac{1}{2}(I_1 + I_2)\\ \tilde{b}^r &= b_0^r + \frac{1}{2}(\mathbb{E}_q[\mathbf{u}_{\cdot r}^T\mathbf{u}_{\cdot r}] + \mathbb{E}_q[\mathbf{v}_{\cdot r}^T\mathbf{v}_{\cdot r}]).\end{aligned} \tag{17}$$

Since the expected value of a Gamma distribution is given by $a/b$, we have $\mathbb{E}(\boldsymbol{\sigma}) = [\tilde{a}^1/b^1, \ldots, \tilde{a}^R/\tilde{b}^R]$.

**update hyperparameters $\boldsymbol{\tau}$.** The posterior distribution of the hyperparameter $\tau$ can also be derived using Equation (10):

$$q_\tau(\tau) = Ga\left(\tau\middle|\tilde{c}, \tilde{d}\right). \tag{18}$$

The parameters of the posterior distribution are updated using the following two equations:

$$\begin{aligned}\tilde{c} &= c_0 + \frac{1}{2}\sum_{i}^{I_1}\sum_{j}^{I_2}\mathbf{O}_{i,j} \\ \tilde{d} &= d_0 + \frac{1}{2}\mathbb{E}_q\left[\|\mathbf{O} \circledast (\mathbf{X} - \mathbf{U}\mathbf{V}^T)\|_F^2\right].\end{aligned} \tag{19}$$

**update lower bound.** The objective is to minimize the value in Equation (8), thereby maximizing the lower bound. Model convergence signifies that the lower bound $lb$ remains relatively stable across successive iterations, computed using the formula below:

$$\begin{aligned} lb \quad &= \int q(\vartheta) ln \frac{p(\mathbf{X}_\Omega, \vartheta)}{q(\vartheta)} d\vartheta \\ &= \mathbb{E}_{q(\vartheta)}\left[ln\left(p(\mathbf{X}_\Omega, \vartheta)\right) - ln\left(q(\vartheta)\right)\right] \\ &= \frac{1}{2}\left(\sum_i l\,n\left|\tilde{\Sigma}_i^{\mathrm{U}}\right| + \sum_j l\,n\left|\tilde{\Sigma}_j^{\mathrm{V}}\right|\right) \\ &+ ln\,\Gamma(\tilde{c}) + \tilde{c}\left(1 - ln\,\tilde{d} - \frac{d_0}{\tilde{d}}\right) \\ &+ \sum_r l\,n\,\Gamma(\tilde{a}^r) + \sum_r (\tilde{a}^r)\left(1 - \frac{b_0^r}{\tilde{b}^r} - ln\,(\tilde{b}^r)\right) \\ &- \frac{\tilde{c}}{2\tilde{d}}\mathbb{E}_q\left[\|\mathbf{O} \circledast (\mathbf{X} - \mathbf{U}\mathbf{V}^T)\|_F^2\right] \\ &- \frac{1}{2}Tr\left(\Lambda\left(\tilde{\mathbf{U}}^T\tilde{\mathbf{U}} + \tilde{\mathbf{V}}^T\tilde{\mathbf{V}} + \sum_i \tilde{\Sigma}_i^{\mathrm{U}} + \sum_j \tilde{\Sigma}_j^{\mathrm{V}}\right)\right) + C. \end{aligned} \tag{20}$$

Up to this point, we have derived the update rules for all relevant parameters within the model training process.

Algorithm 1 outlines the complete procedural flow of BPMFARD. By setting prior distributions for both model parameters and hyperparameters, these priors functions as regularization factors, eliminating the need for manual adjustment. BPMFARD employs variational Bayesian inference for model learning, which avoids the manual tuning of learning rates required in traditional matrix factorization. Consequently, there is no need for explicit specification of regularization factors and learning rates.

Equation (17) shows that as $\mathbf{u}_{\cdot r}$ and $\mathbf{v}_{\cdot r}$ decrease, $\tilde{b}^r$ declines, increasing $\boldsymbol{\sigma}_r$. Equation (12) reveals $\boldsymbol{\Lambda}$ inversely correlates with $\tilde{\Sigma}_i^{\mathrm{U}}$, decreasing $\tilde{\mathbf{U}}_i$'s magnitude. This interplay causes columns in $\mathbf{U}$ and $\mathbf{V}$ to attenuate, exacerbating $\boldsymbol{\sigma}$'s growth towards zero. During training, this achieves an ARD effect, pruning unnecessary columns and determining the effective rank automatically.

BPMFARD uses this mechanism to dynamically adapt its complexity to the data, pruning redundant parameters, mitigating overfitting, and enhancing efficiency. This ensures a balanced trade-off between model complexity and generalization.

**Algorithm 1** Bayesian Probabilistic Matrix Factorization with Automatic Rank Determination (BPMFARD)

**Input:** rating matrix **X**

**Initialization:** **U**, **V**, $\tau$, $\boldsymbol{\sigma}$, $a_0$, $b_0$, $c_0$, $d_0$, *eps*

1: **while** $lb_t - lb_{t-1}$ >= *eps* **do**

2: update feature matrix **U** using Equation (12)

3: update feature matrix **V** using Equation (14)

4: update hyperparameters $\sigma$ using Equation (17)

5: update hyperparameters $\tau$ using Equation (19)

6: compute lower bound $lb$ using Equation (20)

7: **Parameter Elimination Strategy**: Remove any common latent dimension in **U** and **V** if the sum of its squared entries is below the pruning threshold:

$$\text{pruneTol} = (I_1 + I_2) \cdot \epsilon(\| Z_{\text{all}} \|_F),$$

where $\epsilon$ $(\cdot)$ denotes machine epsilon evaluated at the Frobenius norm scale.

8: **end while**

9: To predict the missing elements in the rating matrix **X** using the feature matrices **U** and **V**.

### 4.3 Computational Complexity

The computational cost of updating factor matrices as shown in Equations (12) and (14) is $O(R^2M + R^3(I_1 + I_2))$. Due to the pruning of latent components with zero information contribution during training, the rank $R$ decreases rapidly in the early iterations. The cost of updating the hyperparameter $\boldsymbol{\sigma}$ in Equation (17) is $O(R^2(I_1 + I_2))$, and the cost for updating the noise precision $\boldsymbol{\tau}$ in Equation (19) is $O(R^2M)$. Therefore, the overall complexity of BPMFARD is $O(R^2M + R^3)$, which scales linearly with data size and polynomially with model complexity.

## 5 Experiment

This section presents the experimental evaluation of the proposed method. Section 5.1 describes the datasets and benchmark methods used for comparison. Section 5.2 details the quantitative metrics for assessing recommender system performance. Section 5.3 reports the empirical results, highlighting the proposed method's superiority in recommendation performance and its capability to automatically determine the feature matrix rank.

### 5.1 Datasets

**MovieLens 100K**: The data was collected through the MovieLens website over a seven-month period, from September 19th, 1997, to April 22nd, 1998. This data has

been cleaned by removing users who provided fewer than 20 ratings or lacked complete demographic information. This final dataset consists of 100,000 ratings (on a scale of 1 to 5) from 943 distinct users on 1,682 movies, with a density of 6.3\%.

**MovieLens 1M**: The dataset contains a total of 1,000,209 anonymous ratings for approximately 3,900 movies. These ratings were contributed by 6,040 MovieLens users who joined MovieLens in the year 2,000. The MovieLens 1M dataset has a density of 4.19\%, and the ratings provided by users range from 1 to 5.

**ml-latest-small**: Similar to MovieLens 100K and MovieLens 1M, this dataset includes 100,836 ratings from 610 users on 9,742 movies, with ratings ranging from 1 to 5 and has a density of 0.17%. All selected users have rated at least 20 movies.

### 5.2 Evaluation metrics

We evaluate recommender system algorithms using two commonly adopted metrics: Root Mean Square Error (RMSE) and Mean Absolute Error (MAE).

**RMSE**: RMSE calculates the square root of the average squared differences between predicted values and actual observations, formulated as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{x_{ij}\in\Omega}(x_{ij} - < \mathbf{u}_i, \mathbf{v}_j >)^2}. \tag{21}$$

**MAE**: MAE calculates the average of the absolute deviations between predicted values and actual observations, formulated as:

$$MAE = \frac{1}{n}\sum_{x_{ij}\in\Omega}\left|x_{ij} - < \mathbf{u}_i, \mathbf{v}_j >\right|, \tag{22}$$

Here, $x_{ij}$ denotes the observed value in the rating matrix, and the predicted value is calculated as the inner product of $\mathbf{u}_i$ and $\mathbf{v}_j$.

### 5.3 Experiment Results

This section evaluates the algorithm's performance based on recommendation accuracy and its ability to automatically determine rank for parameter self-tuning. Benchmarks include Funk SVD [9], BiasSVD [10], NMF [13], PMF [11], IRNN [24], WNNR [25], MSS [26], ISVTA [27].

**Accuracy of Rating Prediction.** We evaluate the MSE and RMSE under various hyperparameter settings. The learning rate and regularization factor are selected from the range {0.1, 0.05, 0.01, 0.005, 0.001} using grid search to determine the optimal combination. For low-rank matrix factorization algorithms, we adopt the default parameters provided in the original papers or official implementations whenever possible to ensure the accuracy of the results. The dataset is split into training and testing sets in an 80:20 ratio.

In BPMFARD, the hyperparameters $a_0$, $b_0$, $c_0$, $d_0$ are all set to $1 \times 10^{-6}$ to impose a non-informative prior. We initialize $E[\Sigma]$ as the identity matrix and $E[\tau]$ as 1. The

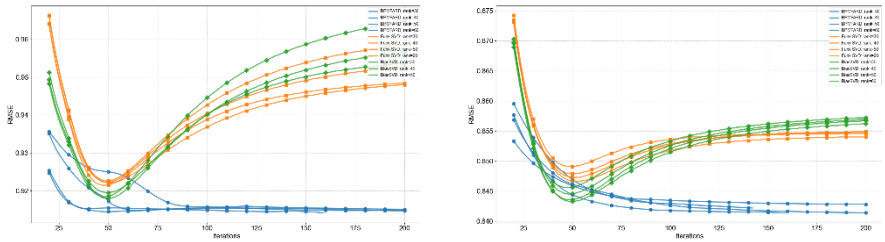
latent factor matrices are randomly initialized with initial ranks of 30, 40, 50, and 60 for subsequent experiments.

Comparative results are presented in Table 1. This table shows that BPMFARD consistently outperforms comparison algorithms across all datasets, as evidenced by lower RMSE and MAE values. This consistent performance across datasets suggests BPMFARD effectively addresses recommendation data sparsity, offering more pronounced benefits than other algorithms.

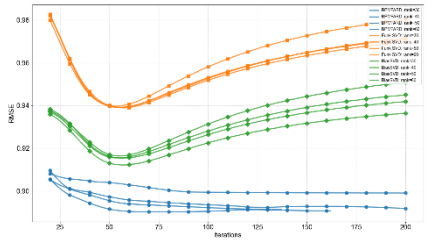
Fig. 3 shows Funk SVD and BiasSVD overfit during training, causing accuracy to drop. In contrast, BPMFARD remains stable, avoiding overfitting due to innovative priors for model parameters that act as regularization. Our method also includes a rational parameter elimination strategy, pruning irrelevant columns from the feature matrix during training, enhancing robustness compared to conventional matrix factorization.

**Table 1.** Comparison of RMSE and MAE Results of Different Recommendation Models on the MovieLens Dataset.

| Model | MovieLens 100K | | MovieLens 1M | | ml-latest-small | |
|---|---|---|---|---|---|---|
| | RMSE | MAE | RMSE | MAE | RMSE | MAE |
| NMF [13] | 2.9007 | 2.6628 | 2.7248 | 2.4557 | 3.0105 | 2.7812 |
| PMF [11] | 0.9413 | 0.7417 | 0.8577 | 0.6761 | 1.0047 | 0.7400 |
| Funk SVD [9] | 0.9214 | 0.7216 | 0.8466 | 0.6658 | 0.9389 | 0.7210 |
| BiasSVD [10] | 0.9180 | 0.7172 | 0.8433 | 0.6617 | 0.9122 | 0.6817 |
| IRNN [24] | 0.9375 | 0.7178 | 0.9255 | 0.7325 | 0.9234 | 0.6849 |
| DNNR[25] | 0.9539 | 0.7404 | 0.9738 | 0.7496 | 0.9481 | 0.7138 |
| MSS [26] | 0.9417 | 0.7427 | 0.8727 | 0.6845 | 0.9299 | 0.7170 |
| ISVTA [27] | 1.0138 | 0.7867 | 0.9377 | 0.6727 | 1.0019 | 0.7605 |
| **BPMFARD** | **0.9142** | **0.7147** | **0.8414** | **0.6589** | **0.8903** | **0.6811** |



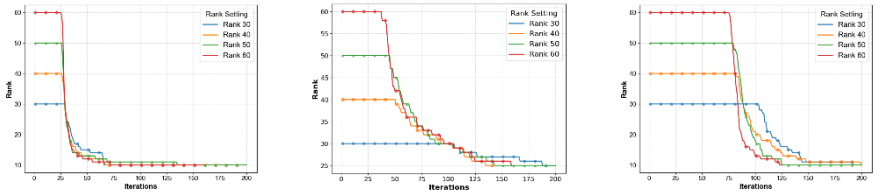
(a) MovieLens 100K (b) MovieLens 1M



(c) ml-latest-small

**Fig. 3.** RMSE varies with the number of iterations across different datasets.

**The Process of Rank Determination.**
This section highlights BPMFARD's advantage in determining optimal feature matrix rank, distinguishing it from manual setting methods. BPMFARD learns rank from data, dynamically adjusting to find the suitable final rank starting with a large initial rank. Table 2 demonstrates BPMFARD's adaptability across datasets. Fig. 4 visualizes rank evolution, showing the algorithm's automatic refinement based on data.



(a) MovieLens 100K (b) MovieLens 1M (c) dml-latest-small

**Fig. 4.** Rank changes during model training across different datasets.

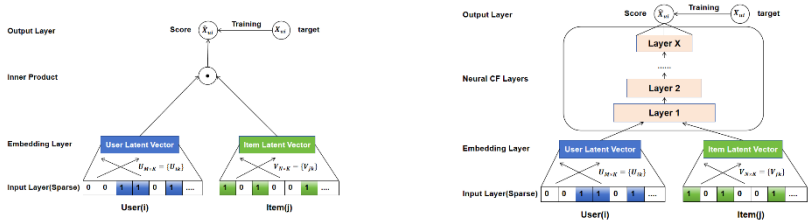
Across the three datasets, BPMFARD consistently converges to optimal ranks: 10 for MovieLens 100K, 25 for MovieLens 1M, and 10 for ml-latest-small, regardless of initial rank. Experiments in Section 5.3 show initial rank variations do not significantly affect RMSE and MAE, reinforcing BPMFARD's robust rank determination and effectiveness in identifying the appropriate rank for the data.

**Table 2.** For different datasets, BPMFARD determines the final rank under different initial rank settings.

| Init Rank | 30 | 40 | 50 | 60 |
|---|---|---|---|---|
| MovieLens 100K | 10 | 10 | 10 | 10 |
| MovieLens 1M | 25 | 25 | 25 | 25 |
| ml-latest-small | 10 | 10 | 10 | 10 |

## 6 Conclusion and Future Work

BPMFARD automates feature-vector dimensionality selection in recommender systems. It automatically finds the optimal rank during training, obviating the need for extensive rank trials required by traditional methods. It outperforms approaches such as Funk SVD, BiasSVD, PMF, and NMF in recommendation accuracy and shows strong resilience against overfitting.




(a) Matrix factorization: a perspective of neural network.

(b) Neural collaborative filtering framework.

**Fig. 5.** Matrix Factorization and Neural Collaborative Filtering Framework.

From a neural network perspective, Neural Collaborative Filtering (NCF) [28] extends matrix factorization by using neural networks to learn nonlinear user-item interactions. Fig. 5 illustrates this extension. Leveraging embeddings and multilayer perceptions, NCF captures more complex interaction patterns than MF. Future work could integrate BPMFARD's ARD strategy into NCF, enabling automatic identification and pruning of uninformative neurons within embeddings and intermediate layers. This integration could create a hybrid model that combines Bayesian uncertainty control with the expressive power of deep networks.

## References


1. Wang, D., Liang, Y., Xu, D., Feng, X., Guan, R.: A content-based recommender system for computer science publications. Knowledge-Based Systems **157**, 1–9 (2018)
2. Kim, S., Kang, H., Choi, S., Kim, D., Yang, M., Park, C.: Large language models meet collaborative filtering: An efficient all-round llm-based recommender system. In: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (SIGKDD). pp. 1395–1406 (2024)
3. Hou, Y., Park, J.D., Shin, W.Y.: Collaborative filtering based on diffusion models: Unveiling the potential of high-order connectivity. In: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR). pp. 1360–1369 (2024)
4. Walek, B., Fajmon, P.: A hybrid recommender system for an online store using a fuzzy expert system. Expert Systems with Applications **212**, 118565 (2023)
5. Zhao, Z.D., Shang, M.S.: User-based collaborative-filtering recommendation algorithms on hadoop. In: 2010 third international conference on knowledge discovery and data mining (KDD). pp. 478–481 (2010)

6. Deshpande, M., Karypis, G.: Item-based top-n recommendation algorithms. ACM Transactions on Information Systems **22**(1), 143–177 (2004)
7. Wang, Y., Gao, M., Ran, X., Ma, J., Zhang, L.Y.: An improved matrix factorization with local differential privacy based on piecewise mechanism for recommendation systems. Expert Systems with Applications **216**, 119457 (2023)
8. Shaikh, S., Kagita, V.R., Kumar, V., Pujari, A.K.: Data augmentation and refinement for recommender system: A semi-supervised approach using maximum margin matrix factorization. Expert Systems with Applications **238**, 121967 (2024)
9. Funk, S.: Netflix update: Try this at home (2006)
10. Paterek, A.: Improving regularized singular value decomposition for collaborative filtering. In: Proceedings of KDD cup and workshop (KDD Cup). pp. 5–8 (2007)
11. Salakhutdinov, R., Mnih, A.: Probabilistic Matrix Factorization. In: Proceedings of the 21st International Conference on Neural Information Processing Systems (NeurIPS). pp. 1257–1264 (2007)
12. Salakhutdinov, R., Mnih, A.: Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In: Proceedings of the 25th international conference on Machine learning (ICML). pp. 880–887 (2008)
13. Lee, D.D., Seung, H.S.: Algorithms for non-negative matrix factorization. In: Proceedings of the 14th International Conference on Neural Information Processing Systems(NeurIPS). pp. 535–541 (2000)
14. Anelli, V.W., Di Noia, T., Di Sciascio, E., Pomo, C., Ragone, A.: On the discriminative power of hyper-parameters in cross-validation and how to choose them. In: Proceedings of the 13th ACM conference on recommender systems (RecSys). pp. 447–451 (2019)
15. Bennett, J., Lanning, S., et al.: The netflix prize. In: Proceedings of KDD cup and workshop (KDD Cup). p. 35 (2007)
16. Bell, R.M., Koren, Y.: Lessons from the Netflix prize challenge. Acm Sigkdd Explorations Newsletter **9**(2), 75–79 (2007)
17. Chan, S., Treleaven, P., Capra, L.: Continuous hyperparameter optimization for large-scale recommender systems. In: 2013 IEEE international conference on big data (IEEE BigData). pp. 350–358 (2013)
18. Beutel, A., Chi, E.H., Cheng, Z., Pham, H., Anderson, J.: Beyond globally optimal: Focused learning for improved recommendations. In: Proceedings of the 26th International Conference on World Wide Web (WWW). pp. 203–212 (2017)
19. Bishop, C.M.: Bayesian PCA. In: Proceedings of the 11th International Conference on Neural Information Processing Systems(NeurIPS). p. 382 (1999)
20. Tan, V.Y., Févotte, C.: Automatic relevance determination in nonnegative matrix factorization with the/spl beta/-divergence. IEEE transactions on pattern analysis and machine intelligence **35**(7), 1592–1605 (2012)
21. Zhao, Q., Zhang, L., Cichocki, A.: Bayesian CP factorization of incomplete tensors with automatic rank determination. IEEE transactions on pattern analysis and machine intelligence **37**(9), 1751–1763 (2015)
22. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM review **51**(3), 455–500 (2009)
23. Jordan, M.I., Ghahramani, Z., Jaakkola, T.S., Saul, L.K.: An introduction to variational methods for graphical models. Machine learning **37**, 183–233 (1999)
24. Lu, C., Tang, J., Yan, S., Lin, Z.: Generalized nonconvex nonsmooth low-rank minimization. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR). pp. 4130–4137 (2014)

25. Zhang, H., Gong, C., Qian, J., Zhang, B., Xu, C., Yang, J.: Efficient recovery of low-rank matrix via double nonconvex nonsmooth rank minimization. IEEE Transactions on Neural Networks and Learning Systems **30**(10), 2916–2925 (2019)
26. Xu, C., Lin, Z., Zha, H.: A unified convex surrogate for the schatten-p norm. In: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI). pp. 926–932 (2017)
27. Zhang, H., Qian, J., Zhang, B., Yang, J., Gong, C., Wei, Y.: Low-rank matrix recovery via modified Schatten-p norm minimization with convergence guarantees. IEEE Transactions on Image Processing **29**, 3132–3142 (2019)
28. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.S.: Neural collaborative filtering. In: Proceedings of the 26th international conference on world wide web (WWW). pp. 173–182 (2017)