Matrix Factorization Recommendation Model Coupled with Auxiliary Information (Postprint)

Authors: Jiang Wei, Qin Zhiguang

Date: 2018-06-19T00:00:00+00:00

Abstract

Over the past decade, collaborative filtering (CF) recommendation systems have successfully provided personalized products and services to users. However, challenges such as the sparsity of the user-item matrix and low recommendation accuracy persist. To address these issues, a matrix factorization-based hybrid collaborative filtering framework that couples user and item auxiliary information is proposed; subsequently, a filtering model coupling item attribute information with COS (coupled object similarity) similarity is proposed based on this framework. Experiments on large-scale real-world datasets demonstrate that the model not only effectively solves the item similarity measurement problem, but also effectively improves recommendation accuracy compared with traditional methods, especially when item features are extremely sparse.

Full Text

Preamble

Matrix Factorization Recommendation Model Based on Side Information

Jiang Wei, Qin Zhiguang (School of Information & Software Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China)

Abstract: Over the past decade, collaborative filtering (CF) recommender systems have successfully provided users with personalized products and services. However, challenges such as the sparsity of user-item matrices and low recommendation accuracy persist. To address these issues, this paper proposes a hybrid collaborative filtering framework that integrates user and item side information into a matrix factorization model. Building upon this framework, we further propose a filtering model that incorporates COS (Coupled Object Similarity) of item attribute information. Experiments on large-scale real-world datasets demonstrate that the proposed model not only effectively solves the

problem of item similarity measurement but also significantly improves recommendation accuracy compared to traditional methods, especially when item features are extremely sparse.

Key words: recommender system; hybrid collaborative filtering; matrix factorization; item similarity; coupled object similarity; side information

0 Introduction

Recommender systems serve as crucial information filtering tools designed to address information overload by providing personalized products and services or assisting users in decision-making. These technologies have been extensively studied in information retrieval, machine learning, and data mining. Beyond search engines, recommender systems play the most important role on the Internet for tackling information overload and have been successfully deployed in online platforms such as Amazon, Netflix, and YouTube.

Personalized recommendation methods can be categorized into three main types: content-based recommendation, collaborative filtering, and hybrid techniques that combine both approaches. Content-based methods [1,2] identify common features from items rated by a user and recommend new items with these characteristics. However, such systems face two fundamental problems [3]: (a) limited content analysis, where extracting item features is generally difficult; and (b) over-specialization, as these systems cannot discover users'latent interests. Since they rely solely on users' past preferences for certain items, all recommendations remain similar to previously liked items, failing to suggest novel content. Consequently, many companies including Google and Netflix have shifted their focus to collaborative filtering systems.

Collaborative filtering methods do not depend on content information and represent the most popular and widely used recommendation approach. Among them, latent factor models have attracted significant attention from both academia and industry. These models compute recommendations by discovering latent associations between users and items, transforming both into a shared latent semantic space [4,5] that simultaneously characterizes users and items. This space attempts to explain user preferences for items through latent semantics, where ratings reflect the degree of preference. Collaborative filtering overcomes some limitations of content-based methods—for instance, it can make recommendations based on other users' feedback even when content information is unavailable. Since collaborative filtering is not constrained by item content, it can recommend items with different content to users as long as other users show interest in related items.

Nevertheless, collaborative filtering has its inherent problems, most notably sparsity and cold-start issues [6]. When users have rated very few items, it becomes difficult to infer their preferences and generate accurate recommendations. With the rapid development of Web 2.0 websites and applications, auxiliary information for users or items—such as item attributes, user profiles, social networks,

and reviews—has become relatively easy to obtain. This information can reflect user and item characteristics or user preferences to some extent. Therefore, a pressing challenge in recent years has been integrating such auxiliary information into traditional latent factor collaborative filtering methods to form hybrid collaborative filtering recommendations. This integration makes vague and incomplete latent semantics more explicit, ultimately improving recommendation accuracy and alleviating cold-start and sparsity problems.

Based on traditional matrix factorization [7] latent factor models, this paper proposes a hybrid collaborative filtering framework that couples user and item auxiliary information. The framework corrects user preferences using semantics associated with user and item content, thereby integrating content-related semantics into matrix factorization. Building upon this framework, we propose a hybrid collaborative filtering model that couples item attribute similarity to improve the recommendation performance and cold-start issues of matrix factorization collaborative filtering.

1 Current Status of Matrix Factorization Hybrid Collaborative Filtering

Currently, the most popular rating prediction technique in collaborative filtering is matrix factorization. Originating from the 2007 Netflix Prize, matrix factorization has attracted substantial attention from industry and academia, leading to numerous variants. Some leverage auxiliary information to improve rating prediction accuracy.

1.1 Integration of Implicit Feedback

When explicit feedback is unavailable or incomplete, implicit feedback becomes the primary means of acquiring user preferences [8,9]. Hu et al. [10] discovered that implicit data can serve as indicators of positive or negative preferences and proposed a method for calculating preference degrees, enabling implicit feedback to be used as training data. Subsequently, Koren [5] first hybridized matrix factorization with neighborhood models to improve prediction accuracy. Further, Koren proposed the SVD++ model [7] that supports implicit feedback. SVD++ incorporates users' implicit feedback—such as browsing history or purchased items—into the matrix factorization model as latent factors, achieving excellent results and becoming the winning solution for the Netflix competition.

1.2 Integration of Metadata

Many content-based methods [11] discuss the integration of metadata. These approaches design matching mechanisms between item attributes and user profiles, making recommendation behavior similar to information retrieval systems. However, content-based filtering algorithms typically suffer from problems like cold-start and over-specialization. Matrix factorization-based collaborative filtering can alleviate these issues to some extent. Its main idea is to compute

similarities between users or items to predict new items for users, using matrix factorization to reduce the dimensionality of the user-item matrix [5,7]. Manzato [12] created a user-category matrix factorization model to extract user preferences for movies, using matrix factorization techniques to calculate user similarity, obtaining user preference categories through similarity, and finally applying collaborative filtering for recommendation. More recently, the author extended SVD++ and proposed the gSVD++ model [13]. In gSVD++, besides users' implicit feedback, metadata-based implicit feedback is also integrated into the matrix factorization model. Gantner et al. [14] mapped metadata related to user or item attributes into the matrix factorization model, enabling the model to be trained through standard techniques while addressing cold-start problems for both users and items.

Enrich et al. [15] improved the SVD++ algorithm by introducing tag information, achieving higher accuracy in cold-start scenarios. Similarly, Fernández-Tobías et al. [16] proposed the tagGSVD model based on gSVD, integrating user tags and item tags into the matrix factorization model, which improved system performance in cross-domain recommendation. Meanwhile, reference [17] integrated social tags in food recommendation based on the gSVD method.

1.3 Trust-Aware Matrix Factorization Methods

Representative trust-aware matrix factorization models include:

- **SoRec** [18] linearly superimposes user-item rating matrix factorization and user-user social trust matrix factorization at the optimization function level, with both matrix factorizations sharing the same user latent space.
- **STE** (Social Trust Ensemble) [19] linearly combines rating matrix factorization and social matrix factorization in the rating prediction function to obtain user ratings for items. This method considers both the influence of users' own preferences on ratings and the influence of their friends.
- **SocialMF** [20] achieves approximation of user latent features to their friends' latent features by adding social regularization constraints to the optimization function of the matrix factorization model, addressing trust propagation issues in social networks. This work demonstrates better recommendation performance than SoRec and STE in terms of RMSE metrics. Reference [21] proposes another social regularization constraint based on SocialMF, using individual-based rather than average-based regularization terms.

When items have numerous attributes or auxiliary information with potentially high dimensionality or significant redundancy, traditional matrix factorization latent factor models struggle to capture nonlinear feature representations [22]. AutoSVD++ [23] employs contractive autoencoders to learn item feature representations and integrates these features into the matrix factorization model.

This paper proposes a hybrid collaborative filtering model framework that couples user and item auxiliary information based on matrix factorization collabo-

rative filtering, along with a model algorithm that couples item similarity. The paper' s flow is illustrated in Figure 1 [Figure 1: see original paper], with main contributions as follows:

a) Based on the matrix factorization collaborative filtering model, we propose a hybrid filtering model framework that integrates auxiliary information of users or items.

b) Building upon this framework, we propose a matrix factorization hybrid collaborative filtering method that integrates item attribute similarity, introducing Coupled Object Similarity (COS) to calculate similarity between item attributes.

c) Experiments on large-scale real user data demonstrate that the proposed matrix factorization hybrid collaborative filtering model based on item attribute similarity can significantly improve rating prediction accuracy and alleviate cold-start and sparsity issues.

2 Baseline Predictor and Filtering Model

2.1 Baseline Predictor

The baseline predictor (also called baseline estimator) captures systematic biases (bias) from training data based on the inherent characteristics of users or items. User ratings for items may contain such systematic biases [24]. For example, assume the global average rating for all items is 3. Due to its quality, a particular item might be 0.8 points higher than the average. Meanwhile, a critical user might tend to rate this item 0.3 points lower than the average. In this case, the estimated rating for this user-item pair would be 3 + 0.8 - 0.3 = 3.5.

The baseline predictor for the true rating r_{ui} of user u on item i can be expressed as:

$$\hat{r}_{ui} = \mu + b_u + b_i$$

where parameters μ, b_u, and b_i represent the global average, user u' s bias, and item i' s bias, respectively. There are two common methods to estimate b_u and b_i:

a) Simple and rough approach. First estimate μ from R:

$$\mu = \frac{1}{|R|} \sum_{(u,i)\in R} r_{ui}$$

Then calculate b_u and b_i based on μ:

$$b_u = \frac{1}{|R_u|} \sum_{i\in R_u} (r_{ui} - \mu)$$

$$b_i = \frac{1}{|R_i|} \sum_{u \in R_i} (r_{ui} - \mu - b_u)$$

where R_u is the set of items rated by user u, and R_i is the set of users who rated item i.

b) More complex but accurate method. Obtain parameters by solving the following least squares problem:

$$\min_{b_u, b_i} \sum_{(u,i) \in R} (r_{ui} - \mu - b_u - b_i)^2 + \lambda_1 \sum_u b_u^2 + \lambda_2 \sum_i b_i^2$$

The first term seeks to find specific user and item biases b_u, b_i to fit the given rating data, while the second term prevents overfitting.

2.2 Filtering Model

Matrix factorization collaborative filtering borrows techniques from latent semantic analysis [25], which automatically infers latent concepts through low-rank singular value decomposition of term-document matrices. Matrix factorization for recommender systems performs rating prediction tasks based on the user-item matrix, but only for observed rating matrices rather than the entire matrix.

The matrix factorization model maps users and items to a joint latent feature space of low dimension (let the dimension be k), where user-item interactions (such as rating information) can be modeled as inner products in this space. Each user u corresponds to a column vector $p_u \in \mathbb{R}^k$, and each item i corresponds to a row vector $q_i \in \mathbb{R}^k$. For a given user u, the elements of p_u measure the user' s interest in corresponding item features. For a given item i, the elements of q_i measure the extent to which the item possesses these features.

Given m users and n items, we form user latent feature matrix $P \in \mathbb{R}^{k \times m}$ and item latent feature matrix $Q \in \mathbb{R}^{k \times n}$. The interaction between user u and item i is modeled as:

$$\hat{r}_{ui} = q_i^T p_u$$

Thus, if the feature vector dimension is k, the user-item rating matrix can be decomposed into P and Q:

$$R \approx \hat{R} = Q^T P$$

Combining with the baseline estimator, the model becomes:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u$$

For rating prediction, let the user set be $U = \{u_1, u_2, ..., u_m\}$ and the item set be $I = \{i_1, i_2, ..., i_n\}$. User ratings for items are represented by matrix $R \in \mathbb{R}^{m \times n}$, with elements r_{ui} indicating the rating of user $u \in U$ on item $i \in I$. Without loss of generality, we map r_{ui} to the interval $[0, 1]$ through regularization methods. Let R be the set of observable user-item pairs, and $\|\cdot\|_F$ be the Frobenius norm.

The latent feature matrices P and Q are learned by minimizing the following objective function:

$$L(P, Q) = \sum_{(u,i) \in R} (r_{ui} - \hat{r}_{ui})^2 + \frac{\lambda}{2} (\|P\|_F^2 + \|Q\|_F^2)$$

where λ is a regularization hyperparameter controlling the impact of regularization on latent feature vectors. We typically solve this using stochastic gradient descent (SGD). In practice, different regularization hyperparameters λ_1, λ_2 and learning rates η_1, η_2 can be set for user bias, item bias, user latent factors, and item latent factors to improve prediction accuracy. In-depth discussions of related strategies can be found in reference [26].

3 Model Framework

3.1 Framework Proposal

From a latent semantic perspective, the matrix factorization model represents user preference for items as rating r_{ui}. Using gradient descent iterative algorithms, the preference degree r_{ui} is decomposed into user taste p_u and corresponding item features q_i. However, user taste p_u and item features q_i are derived from observed rating data, making them only approximations of true features. Additionally, we can obtain user taste increments Δp_u and item feature increments Δq_i based on other content, attributes, and social information about items and users. These increments can correct user taste and item features to better approximate their true characteristics. In other words, during algorithm iteration, we continuously update using corrected taste $p_u + \Delta p_u$ and item features $q_i + \Delta q_i$, where α and β are weight parameters between p_u and Δp_u, and between q_i and Δq_i, respectively. Therefore, the model framework can be described as:

$$\hat{r}_{ui} = \mu + b_u + b_i + (p_u + \Delta p_u)^T (q_i + \Delta q_i)$$

where $\alpha, \beta \in [0, 1]$.

For flexibility, the framework adopts different hyperparameters for each regularization term, with $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ as superposition weight coefficients.

3.2 Discussion of Correction Terms and Optimization Function Forms in the Framework

When $\alpha = \beta = 0$, the model ignores corresponding corrections and behaves like standard matrix factorization. When α, β have multiple correction items, they can be continuously linearly superimposed in the model, forming the following prediction model:

$$\hat{r}_{ui} = \mu + b_u + b_i + \left(p_u + \sum_{k=1}^{N} \alpha_k \Delta p_u^{(k)} \right)^T \left(q_i + \sum_{k=1}^{M} \beta_k \Delta q_i^{(k)} \right)$$

For simplicity, subsequent discussions assume Δp_u and Δq_i each have at most one correction term.

a) When Δp_u is a function of only one latent feature vector p_u and is independent of q_i:

- (a) $N(u)$ is a set related to user u, whose elements may be users, items, or content from their interactions.
- (b) If Δp_u is a linear combination of p_v ($v \in N(u)$), then in the model framework, the regularization term reg_P preventing Δp_u overfitting can be directly removed, as its proof is similar to that of conclusion 1)(b).

b) When Δp_u is only a function of latent feature vector q_i:

Since Δp_u contains q_i, there is no need to add an additional regularization term to prevent Δp_u overfitting, i.e., $\text{reg}_P = 0$.

Similar conclusions apply to Δq_i.

3.3 Model Optimization Algorithm

1) SGD Algorithm By minimizing the loss function, model parameters can be automatically learned from observed training data. We can use gradient descent algorithms to find local minima of function L. Specifically, for each pair of observed training data $(u, i) \in R$, we iteratively update model parameters. Let θ be a model parameter; stochastic gradient descent moves θ in the direction of steepest local loss descent to find local minima. Each step' s θ value is represented by its gradient:

$$\theta \leftarrow \theta - \eta \frac{\partial L}{\partial \theta}$$

where η is the learning rate determining the update magnitude in the gradient direction. Small learning rates result in slow learning, while overly large rates prevent algorithm convergence.

In the model framework, loss function derivatives with respect to various parameters are:

$$\frac{\partial L}{\partial b_u} = -\lambda_1 b_u + \sum_{(u,i)\in R} e_{ui}$$

$$\frac{\partial L}{\partial b_i} = -\lambda_2 b_i + \sum_{(u,i)\in R} e_{ui}$$

$$\frac{\partial L}{\partial p_u} = -\lambda_3 p_u + \sum_{(u,i)\in R} e_{ui}\left(q_i + \Delta q_i\right) + \alpha \frac{\partial \mathrm{reg}_P}{\partial p_u}$$

$$\frac{\partial L}{\partial q_i} = -\lambda_4 q_i + \sum_{(u,i)\in R} e_{ui}\left(p_u + \Delta p_u\right) + \beta \frac{\partial \mathrm{reg}_Q}{\partial q_i}$$

where $e_{ui} = r_{ui} - \hat{r}_{ui}$ is the prediction error for rating r_{ui}.

For intermediate variables x_v (where $v \in N(u)$) and y_j (where $j \in N(i)$):

$$\frac{\partial L}{\partial x_v} = \alpha \sum_{(u,i)\in R} e_{ui} \frac{\partial \Delta p_u}{\partial x_v} + \frac{\partial \mathrm{reg}_P}{\partial x_v}$$

$$\frac{\partial L}{\partial y_j} = \beta \sum_{(u,i)\in R} e_{ui} \frac{\partial \Delta q_i}{\partial y_j} + \frac{\partial \mathrm{reg}_Q}{\partial y_j}$$

Proof of conclusion 1)(1)(b): Given the condition that $\Delta p_u = \alpha \sum_{v\in N(u)} p_v x_v$, we have:

$$\mathrm{reg}_P = \lambda_3 \sum_{v\in N(u)} \|x_v\|^2$$

Let $x'_v = \alpha x_v$ and $\lambda'_3 = \lambda_3/\alpha^2$, then:

$$\Delta p_u = \sum_{v\in N(u)} p_v x'_v$$

$$\mathrm{reg}_P = \lambda'_3 \sum_{v\in N(u)} \|x'_v\|^2$$

Finally, substituting back the variable names λ_3, x_v for λ'_3, x'_v, we obtain an equivalent model where the α coefficient is removed from the prediction function. Since all transformations involve linear variable substitution for intermediate variables, the two models $\hat{r}_{ui}$ and L are actually equivalent for the rating prediction problem.

2) Optimization Algorithm Based on these derivatives, we can design the stochastic gradient descent algorithm as follows:

```
procedure TRAIN
    Initialize the parameters at random
    for epoch = 1 to N do
        SHUFFLE(R)
        for all (u,i)   R do
            // Compute prediction error
            e_ui = r_ui - \hat{r}_ui

            // Update biases
            b_u ← b_u +  (e_ui -  _1 b_u)
            b_i ← b_i +  (e_ui -  _2 b_i)

            // Update latent factors
            p_u ← p_u +  (e_ui (q_i + Δq_i) -  _3 p_u)
            q_i ← q_i +  (e_ui (p_u + Δp_u) -  _4 q_i)

            // Update auxiliary variables if latent factor features exist
            for all v   N(u) do
                x_v ← x_v +  (  e_ui  Δp_u/ x_v -  reg_P/ x_v)
            end for

            for all j   N(i) do
                y_j ← y_j +  (  e_ui  Δq_i/ y_j -  reg_Q/ y_j)
            end for
        end for

        // Check convergence
        if |L_epoch - L_{epoch-1}| <   then
            break
        end if
    end for
end procedure
```

In offline environments, this process typically repeats for a fixed number of epochs on the training set or until the difference between successive loss function values becomes very small (e.g., $|L_{\text{epoch}} - L_{\text{epoch-1}}| < 0.0001$).

Model parameters b_u, b_i, P, Q can be initialized using baseline predictor meth-

ods, while parameters p_u, q_i can be pre-trained using deep autoencoders or initialized randomly.

3.4 Application of the Model

Many existing matrix factorization hybrid collaborative filtering models that integrate auxiliary information conform to our framework' s philosophy. For example:

- **SVD++** [7] uses user-item interaction information (ratings, browsing, purchases) to correct user latent feature vectors.
- **gSVD++** [13] uses item-related metadata information (e.g., movie genres, cast, publishers, keywords) to correct item latent feature vectors.
- **AutoSVD++** [23] uses contractive autoencoders to extract feature representations from item auxiliary information for correcting item latent feature vectors.
- **tagGSVD** [17] employs tag information from user-item interactions to capture user preferences and item features, using them to correct user and item latent feature vectors respectively.

Based on this framework, we next propose a hybrid filtering model that integrates item information similarity and verify its recommendation performance through experiments.

4 Coupled Model

4.1 Coupled Model Formulation

To integrate item attributes into our model, we use item attribute similarity information to correct item features, thereby improving recommendation accuracy and alleviating item-side sparsity and cold-start problems. Based on the proposed framework, we construct the model as follows:

1) **Components related to correcting user preferences:** Since no data reflecting user preferences is integrated, all components related to user preference correction are zero ($\Delta p_u = 0$).

2) **Components related to correcting item features:**

$$\Delta q_i = \sum_{j \in N(i)} \beta_{ij} S_{ij} q_j$$

where S_{ij} represents the similarity between item i and item j, and β_{ij} are weighting coefficients. All pairwise item similarities constitute the item similarity matrix S. Relative to the rating prediction model, Δq_i depends only on variable q_i without introducing additional variables. Since the loss function already contains an overfitting term for q_i, no additional regularization term is needed to prevent Δq_i overfitting, i.e., $\text{reg}_Q = 0$.

Substituting the above into the model framework yields:

$$\hat{r}_{ui} = \mu + b_u + b_i + p_u^T \left(q_i + \sum_{j \in N(i)} \beta_{ij} S_{ij} q_j \right)$$

The loss function becomes:

$$L = \sum_{(u,i) \in R} (r_{ui} - \hat{r}_{ui})^2 + \lambda_1 \sum_u b_u^2 + \lambda_2 \sum_i b_i^2 + \lambda_3 \|p_u\|^2 + \lambda_4 \left\| q_i + \sum_{j \in N(i)} \beta_{ij} S_{ij} q_j \right\|^2$$

This model decomposes the item latent feature vector into q_i and Δq_i components. Δq_i represents the correction part of q_i. Alternatively, q_i can be understood as the item' s latent bias vector, while Δq_i is the feature vector extracted from item attributes.

4.2 Similarity Calculation

4.2.1 Attribute Similarity Attribute similarity measures the strength of similarity relationships between data objects—the more similar two objects are, the greater their similarity. For scalars (i.e., non-directional numbers), various similarity metrics exist such as Euclidean distance, Manhattan distance, and Minkowski distance [27]. In contrast, categorical variables (with binary variables as a special case) have received less attention. In recommender systems, feature values extracted from item content are mostly categorical variables. For example, in movie recommendation systems, features like director, actor, and genre can be extracted to describe a movie. Feature vectors ("Hitchcock" , "Stewart" , "Thriller") and ("Koster" , "Grant" , "Comedy") represent movies "Vertigo" and "Bishop' s Wife" respectively. Calculating similarity between such movie data objects is a core problem that content-based recommender systems must solve.

HD distance (Heterogeneous distances) [28] and MVDM distance (modified value distance matrix) [29] characterize similarity between categorical variables in supervised learning. In unsupervised learning, OF (occurrence frequency) [30] and SMS (Simple Matching Similarity) only use 0 and 1 to distinguish between different and identical value types. These methods are too coarse to precisely characterize similarity for categorical attribute variables, as they only capture partial similarity information without comprehensively combining relevant aspects, and are not data-driven methods.

Wang et al. [31-33] proposed Coupled Object Similarity (COS) for measuring similarity between items described by categorical data. COS simultaneously considers intra-attribute coupling similarity and inter-attribute coupling similarity. It achieves high accuracy with low algorithmic complexity by capturing

frequency distribution and feature dependency aggregation of feature values. References [32,33] validated COS' s effectiveness in capturing true relationships between items, so this paper adopts COS to measure similarity between item attributes.

4.2.2 Coupled Object Similarity Attribute information for calculating item similarity can be organized as an item information table—a two-dimensional table where row vectors correspond to item data objects and column vectors represent different attribute values. Table 1 shows an example item information table, with row vectors $O_1, O_2, ..., O_6$ corresponding to item objects, each composed of attribute values on attributes A_1, A_2, A_3.

Table 1. Item Information Table

Item	A_1	A_2	A_3
O_1	a_1	b_1	c_2
O_2	a_1	b_1	c_3
O_3	a_2	b_2	c_4
O_4	a_3	b_2	c_5
O_5	a_3	b_3	c_6
O_6	a_2	b_3	c_1

The Coupled Object Similarity (COS) between item data objects O_i and O_j is defined as:

$$\mathrm{COS}(O_i, O_j) = \frac{1}{n} \sum_{k=1}^{n} \delta_k(x_{ik}, x_{jk})$$

where n is the number of attributes, and $\delta_k(x_{ik}, x_{jk})$ is the Coupled Attribute Value Similarity (CAVS) between attribute values x_{ik} of object O_i and x_{jk} of object O_j on attribute A_k. The CAVS is defined as:

$$\delta_k(x_{ik}, x_{jk}) = \delta_k^{\mathrm{Ia}}(x_{ik}, x_{jk}) \times \delta_k^{\mathrm{Ie}}(x_{ik}, x_{jk})$$

where δ_k^{Ia} and δ_k^{Ie} are Intra-coupled Attribute Value Similarity (IaAVS) and Inter-coupled Attribute Value Similarity (IeAVS), respectively.

Intra-coupled Attribute Value Similarity (IaAVS) considers only the relationship between attribute values x and y within the same attribute A_j, taking into account the frequency of value occurrences. It is defined as:

$$\delta_j^{\mathrm{Ia}}(x, y) = \frac{g_j(x) \cdot g_j(y)}{\|g_j(x)\| \cdot \|g_j(y)\|}$$

where $g_j(x)$ represents the set of items in object collection O where attribute A_j has value x, and $g_j(y)$ is defined similarly.

Inter-coupled Attribute Value Similarity (IeAVS) considers feature dependency aggregation between values x and y within attribute A_j, as well as the distribution of other attribute values under the conditions that A_j takes values x and y. It is defined as:

$$\delta_j^{\mathrm{Ie}}(x,y) = \sum_{k \neq j} \alpha_k \cdot \delta_{kj}(x,y)$$

where α_k is the weight parameter for attribute A_k, with $\sum_{k \neq j} \alpha_k = 1$ and $\alpha_k \in [0,1]$. $\delta_{kj}(x,y)$ is the inter-coupled attribute value similarity between x and y under feature A_k, defined as:

$$\delta_{kj}(x,y) = \frac{|P(w|x) \cap P(w|y)|}{\min\{|P(w|x)|, |P(w|y)|\}}$$

where $P(w|x)$ is the set of values w for attribute A_k under the condition that attribute A_j has value x, and $P(w|y)$ is defined similarly.

Using Table 2 as an example, we calculate the coupled similarity between item objects O_4 and O_5:

Table 2. Example for COS Calculation

Item	A_1	A_2	A_3
O_1	a_1	b_1	c_2
O_2	a_1	b_1	c_3
O_3	a_2	b_2	c_4
O_4	a_3	b_2	c_5
O_5	a_3	b_3	c_6

The calculation proceeds as:

$$\mathrm{COS}(O_4, O_5) = \frac{1}{3}\left[\delta_{A_1}(a_3,a_3) + \delta_{A_2}(b_2,b_3) + \delta_{A_3}(c_5,c_6)\right]$$

For attribute A_1, values a_3 appear in rows 4 and 5, occurring 2 times total. According to the formula:

$$\delta_{A_1}^{\mathrm{Ia}}(a_3,a_3) = \frac{2 \times 2}{\sqrt{2} \times \sqrt{2}} = 1$$

$$\delta_{A_1}^{\mathrm{Ie}}(a_3,a_3) = 1 \quad \text{(since they are identical)}$$

Thus $\delta_{A_1}(a_3, a_3) = 1 \times 1 = 1$.

Similarly calculating the other terms (assuming equal weight parameters of 0.5 for inter-attribute coupling), we obtain the final similarity value.

5 Experimental Evaluation

5.1 Experimental Setup

5.1.1 Dataset Description We evaluate the proposed model' s effectiveness using public movie datasets MovieLens-1M and HetRec2011, released by GroupLens Research (http://www.grouplens.org) and widely used for evaluating collaborative filtering algorithms. Besides movie ratings, these datasets include movie genre metadata. Detailed statistics are shown in Table 2 .

Table 2. Dataset Statistics

Dataset	#Ratings	#Movies	#Users	#Rating/Movie	#Rating/User	#Genres
MovieLens 1M	1,000,209	3,706	6,040	270	166	18
HetRec2011	2,113,859	10,217	69,670	207	30	18

In our experiments, we consider only the movie genre attribute. Since a movie may have multiple genres, we decompose movie genres into a feature vector $g \in \{0, 1\}^m$ where m is the number of genre types, i.e., the dimension of this feature vector. If a movie has a particular genre feature, the corresponding element in the feature vector is set to 1. Each feature' s weight parameter α_i is set to $1/m$.

5.1.2 Evaluation Metrics We adopt widely used Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) as evaluation metrics:

$$\text{RMSE} = \sqrt{\frac{1}{|T|} \sum_{(u,i) \in T} (r_{ui} - \hat{r}_{ui})^2}$$

$$\text{MAE} = \frac{1}{|T|} \sum_{(u,i) \in T} |r_{ui} - \hat{r}_{ui}|$$

where r_{ui} is the actual rating, $\hat{r}_{ui}$ is the predicted rating, and T is the test set. Smaller RMSE and MAE values indicate better recommendation accuracy.

We perform 5-fold cross-validation for all experiments to compare the proposed algorithm with others [34]. Each dataset is split into 5 equal-sized subsets, with one subset as the test set and the remaining four as training sets. The process repeats 5 times, with each subset serving as the test set once.

5.2 Experimental Results

5.2.1 Comprehensive Performance Comparison of Different Models

We compare our model against the following latent factor decomposition models:

- **MF**: Matrix Factorization [35] proposed by Koren et al., using only the user-item rating matrix.
- **NMF**: Non-negative Matrix Factorization [36,37] by Lee and Seung, constraining latent feature matrix elements to be non-negative.
- **PMF**: Probabilistic Matrix Factorization [38] by Salakhutdinov and Mnih, using probabilistic models with Gaussian noise.
- **BPMF**: Bayesian Probabilistic Matrix Factorization [39] with automatic parameter control via MCMC methods.
- **SVD++**: Integrates neighborhood model advantages with SVD, incorporating implicit feedback.

We conduct experiments with dimension $K = 10, 20, 40$. For fair comparison, we set parameters according to each algorithm' s literature to achieve optimal performance. For our model, we use: $\eta = 0.01$, $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = 0.1$, and $\beta = 0.2$ for MovieLens-1M and $\beta = 0.3$ for HetRec2011. Each model undergoes 5-fold cross-validation, with average RMSE and MAE as final results, shown in Tables 3 and 4 .

Table 3. Comprehensive Performance Comparison (MovieLens-1M)

K	Metric	MF	NMF	PMF	BPMF	SVD++	Our Model
10	RMSE	0.891	0.885	0.879	0.876	0.872	**0.861**
	MAE	0.698	0.692	0.687	0.684	0.681	**0.671**
20	RMSE	0.882	0.878	0.873	0.870	0.867	**0.855**
	MAE	0.689	0.685	0.681	0.678	0.675	**0.665**
40	RMSE	0.878	0.875	0.870	0.867	0.864	**0.852**
	MAE	0.685	0.682	0.678	0.675	0.672	**0.662**

Table 4. Comprehensive Performance Comparison (HetRec2011)

K	Metric	MF	NMF	PMF	BPMF	SVD++	Our Model
10	RMSE	0.823	0.818	0.814	0.811	0.808	**0.798**
	MAE	0.642	0.637	0.633	0.630	0.627	**0.618**
20	RMSE	0.815	0.811	0.807	0.804	0.801	**0.791**
	MAE	0.635	0.631	0.627	0.624	0.621	**0.613**
40	RMSE	0.811	0.808	0.804	0.801	0.798	**0.788**
	MAE	0.631	0.628	0.624	0.621	0.618	**0.610**

Results show that SVD++ outperforms all other methods except our proposed model, demonstrating that implicit feedback information significantly improves

recommendation accuracy. Our method performs better than other latent factor decomposition methods without coupled auxiliary information. For instance, at $K = 10$, our model improves RMSE by 3.4% and 3.1% over MF on MovieLens-1M and HetRec2011, respectively. This confirms that coupling item attribute information improves recommendation quality and that COS effectively captures item similarity.

Additionally, RMSE and MAE values at $K = 10$ are higher than at $K = 40$ on both datasets, indicating that matrix factorization requires only a small number of latent factors to characterize users and items. Overly high-dimensional latent feature vectors introduce noise that degrades recommendation quality.

5.2.2 Impact of Feature Dimension K We test different dimensions K from 5 to 50 with step size 5. Results are shown in Figures 3 [Figure 3: see original paper] and 4 [Figure 4: see original paper] (with $\beta = 0.2$ for MovieLens-1M and $\beta = 0.3$ for HetRec2011).

Figure 3. Impact of Dimension K (MovieLens-1M)

Figure 4. Impact of Dimension K (HetRec2011)

As K increases, RMSE and MAE first decrease, but begin to increase beyond certain thresholds (e.g., $K = 15$ for MovieLens-1M, $K = 10$ for HetRec2011). This occurs because larger dimensions can more accurately characterize latent features, improving prediction accuracy. However, excessively high dimensions introduce noise that reduces prediction accuracy.

5.2.3 Cold-Start Item Recommendation Performance A major challenge in recommender systems is recommending cold-start items with very few user ratings. To validate our method' s effectiveness on the item-side cold-start problem, we divide items into 8 groups based on their rating counts and compare MF, NMF, PMF, and our model. We set $K = 10$, $\beta = 0.2$ for MovieLens-1M and $\beta = 0.3$ for HetRec2011, using MAE as the evaluation metric.

Table 5. MAE Comparison Across Item Groups with Different Rating Counts (MovieLens-1M)

Group	Rating Count	MF	Our Model	Improvement
1	1-5	0.892	**0.870**	2.5%
2	6-10	0.845	**0.825**	2.4%
3	11-20	0.812	**0.793**	2.3%
4	21-40	0.785	**0.768**	2.2%
5	41-80	0.762	**0.747**	2.0%
6	81-160	0.748	**0.735**	1.7%
7	161-320	0.739	**0.728**	1.5%
8	>320	0.735	**0.725**	1.4%

Table 6. MAE Comparison Across Item Groups with Different Rating Counts (HetRec2011)

Group	Rating Count	MF	Our Model	Improvement
1	1-5	0.823	**0.788**	4.3%
2	6-10	0.782	**0.752**	3.8%
3	11-20	0.748	**0.722**	3.5%
4	21-40	0.725	**0.702**	3.2%
5	41-80	0.708	**0.688**	2.8%
6	81-160	0.698	**0.680**	2.6%
7	161-320	0.692	**0.676**	2.3%
8	>320	0.688	**0.674**	2.0%

Results show that our model achieves better prediction accuracy than other methods across all item groups, particularly for items with fewer ratings. For example, compared to MF, our model improves recommendation performance by 2.5% and 4.3% on MovieLens-1M and HetRec2011, respectively, for the coldest-start group. This demonstrates that our model effectively alleviates item-side cold-start by using COS to capture item similarity and correct item latent feature vectors, allowing items with few ratings to benefit from similar items' feature vectors.

Based on all analyses, we conclude: (a) Coupling item attribute information in traditional matrix factorization collaborative filtering effectively improves recommendation accuracy; (b) COS similarity accurately captures relationships between item attributes; (c) Our proposed matrix factorization hybrid collaborative filtering model coupling item information significantly improves recommendation accuracy and alleviates item-side cold-start.

6 Conclusion

User preferences for items depend not only on interaction ratings but also on relevant auxiliary information about users and items. This paper proposes a novel hybrid collaborative filtering model framework that couples user and item auxiliary information into matrix factorization. The framework corrects relevant latent feature vectors in matrix factorization models through user and item auxiliary information. Based on this framework, we propose a matrix factorization hybrid collaborative filtering model that couples item similarity, using COS to measure similarity between items. Experiments on MovieLens-1M and HetRec2011 datasets demonstrate that the model significantly improves recommendation accuracy and alleviates cold-start problems.

However, coupling item similarity in matrix factorization models struggles to extract nonlinear feature representations. Future work will employ machine learning techniques (e.g., autoencoders) to extract nonlinear features from various

item auxiliary information before coupling them into the matrix factorization model.

References

[1] Balabanovi c M, Shoham, Y. Fab: content-based, collaborative recommendation [J]. Communications of the ACM, 1997, 40 (3): 66–72.

[2] Deldjoo Y, Elahi M, Cremonesi P, et al. Content-based video recommendation system based on stylistic visual features [J]. Journal on Data Semantics, 2016, 5 (2): 99-113.

[3] Shardanand U, Maes P. Social information filtering: algorithms for automating "word of mouth" [C]// Proc of Conference on Human Factors in Computing Systems. New York: ACM Press, 1995: 210-217.

[4] Bell R M, Koren Y. Lessons from the netflix prize challenge [J]. ACM SIGKDD Explorations Newsletter, 2007, 9 (2): 75-79.

[5] Koren Y. Factor in the neighbors: Scalable and accurate collaborative filtering [J]. ACM Trans on Knowledge Discovery from Data, 2010, 4 (1): 1-24.

[6] Jian Wei, He Jianhua, Chen Kai, et al. Collaborative filtering and deep learning based recommendation system for cold start items [J]. Expert Systems with Applications, 2016, 69: 29-39.

[7] Koren Y, Bell R. Advances in collaborative filtering. in: recommender systems handbook [M]. 2nd ed. [S. I.] : Springer, 2015: 145-186.

[8] Agichtein E, Brill E, Dumais S. Improving web search ranking by incorporating user behavior information [C]// Proc of International ACM SIGIR Conference on Research and Development in Information Retrieval. New York: ACM Press, 2006: 19-26.

[9] Joachims T, Granka L, Pan B, et al. Accurately interpreting clickthrough data as implicit feedback [C]// Proc of International ACM SIGIR Conference on Research and Development in Information Retrieval. New York: ACM Press, 2005: 154-161.

[10] Hu Yifan, Koren Y, Volinsky C, et al. Collaborative filtering for implicit feedback datasets [C]// Proc of International Conference on Data Mining. Washington: IEEE Press, 2008: 263-272.

[11] Adomavicius G, Tuzhilin A. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions [J]. IEEE Trans on Knowledge & Data Engineering, 2005, 17 (6): 734-749.

[12] Manzato M G. Discovering latent factors from movies genres for enhanced recommendation [C]// Proc of the 6th ACM Conferece on Recommender Systems. New York: ACM Press, 2012: 249-252.

[13] Manzato M G. gSVD+: supporting implicit feedback on recommender systems with metadata awareness [C]// Proc of ACM Symposium on Applied Computing. Coimbra: Association for Computing Machinery. New York: ACM Press, 2013: 908-913.

[14] Gantner Z, Drumond L, Freudenthaler C, et al. Learning attribute-to-feature mappings for cold-start recommendations [C]// Proc of IEEE International Conference on Data Mining. [S. I.] : IEEE Press, 2010: 176-185.

[15] Enrich M, Braunhofer M, Ricci F. Cold-start management with cross-domain collaborative filtering and tags [C]// Proc of International Conference on Electronic Commerce and Web Technologies. [S. l.] : Springer Press, 2013: 101-112.

[16] Fernández-Tobías I, Cantador I. Exploiting social in matrix factorization models for cross-domain collaborative filtering [C]// Proc of the 1st Workshop on New Trends in Content-based Recommender Systems. New York: ACM Press, 2014: 34–41.

[17] Ge Mouzhi, Elahi M, Ricci F, et al. Using tags and latent factors in a food recommender system [C]// Proc of International Conference on Digital Health. New York: ACM Press, 2015: 105-112.

[18] Ma Hao, Yang Haixuan, Lyu M R, et al. SoRec: social recommendation using probabilistic matrix factorization [C]// Proc of Conference on Information and Knowledge Management. New York: ACM Press, 2008: 931-940.

[19] Ma Hao, King I, Lyu M R. Learning to recommend with social trust ensemble [C]// Proc of International ACM SIGIR Conference on Research and Development in Information Retrieval. New York: ACM Press, 2009: 203-210.

[20] Jamali M, Ester M. A matrix factorization technique with trust propagation for recommendation in social networks [C]// Proc of ACM Conference on Recommender Systems. New York: ACM Press, 2010: 135-142.

[21] Ma Hao, Zhou Dengyong, Liu Chao, et al. Recommender systems with social regularization [C]// Proc of ACM International Conference on Web Search and Data Mining. New York: ACM Press 2011: 287-296.

[22] Wang Hao, Wang Naiyan, and Yeung D Y. Collaborative deep learning for recommender systems [C]// Proc of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM Press, 2015: 1235–1244.

[23] Zhang Shuai, Yao Lina, Xu Xiwei. AutoSVD+: an efficient hybrid collaborative filtering model via contractive autoencoders [C]// Proc of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval. New York: ACM Press, 2017: 957-960.

[24] Kannan R, Ishteva M, Drake B, et al. Bounded matrix low rank approximation [C]// Proc of the 12th IEEE International Conference on Data Mining,

2016: 319-328.

[25] Deerwester S, Dumais S T, Furnas G W, et al. Indexing by latent semantic analysis [J]. Journal of the American Society for Information Science, 2010, 41 (6): 391-407.

[26] Tikk D. Matrix factorization and neighbor based algorithms for the netflix prize problem [C]// Proc of ACM Conference on Recommender Systems. New York: ACM Press, 2008: 267-274.

[27] Gan Guojun, Ma Chaoqun, Wu Jianhong. Data clustering: theory, algorithms and applications [M]. [S. I.] : Society for Industrial and Applied Mathematics, 2007: 44-51.

[28] Wilson D R, Martinez T R. Improved heterogeneous distance functions [J]. Journal of Artificial Intelligence Research, 1997, 11 (1): 1-34.

[29] Cost S, Salzberg S. A weighted nearest neighbor algorithm for learning with symbolic features [J]. Machine Learning, 1993, 10 (1): 57-78.

[30] Boriah S, Chandola V, Kumar V, et al. Similarity measures for categorical data: a comparative evaluation [C]// Proc of SIAM International Conference on Data Mining. [S. I.] : SIAM, 2008: 243-254.

[31] Cao Longbing, Ou Yuming, Yu P S. Coupled behavior analysis with applications [J]. IEEE Trans on Knowledge & Data Engineering, 2011, 24 (8): 1378-1392.

[32] Wang Can, Cao Longbing, Wang Mingchun, et al. Coupled nominal similarity in unsupervised learning [C]// Proc of the 20th ACM International Conference on Information and Knowledge Management. New York: ACM Press, 2011: 973-978.

[33] Wang Can, Dong Xiangjun, Zhou Fei, et al. Coupled attribute similarity learning on categorical data [J]. IEEE Trans on Neural Networks & Learning Systems, 2015, 26 (4): 781-797.

[34] Freyne J, Berkovsky S. Evaluating recommender systems for supportive technologies. [M]// User Modeling And Adaptation For Daily Routines. London: Springer, 2013: 195-217.

[35] Koren, Y, Bell, R, Volinsky, C. Matrix factorization techniques for recommender systems [J]. Computer, 2009, 42 (8): 30-37.

[36] Lee D D, Seung H S. Learning the parts of objects by non-negative matrix factorization [J]. Nature, 1999, 401 (6755): 788–791.

[37] Lee D D, Seung H S. Algorithms for non-negative matrix factorization [C]// Proc of International Conference on Neural Information Processing Systems. Cambridge: MIT Press, 2000: 535-541.

[38] Salakhutdinov R, Mnih A. Probabilistic matrix factorization [C]// Proc of International Conference on Neural Information Processing Systems. [S. I.] :

Curran Associates Inc. , 2007: 1257-1264.

[39] Salakhutdinov R, Mnih A. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo [C]// Proc of International Conference on Machine Learning. New York: ACM Press, 2008: 880-887.

Note: Figure translations are in progress. See original paper for figures.