Collaborative Filtering with Matrix Factorization: Book Recommendation System

Hardcover Book Friend Finder

January 2, 2026

Abstract

We present a hybrid book recommendation system using collaborative filtering with matrix factorization. The model decomposes a sparse user-book interaction matrix into two lower-dimensional matrices representing latent features. We employ masking to handle extreme sparsity (95.75%) and combine collaborative filtering with popularity-based recommendations to achieve 8.83% precision@10, outperforming pure collaborative filtering by 308%.

1 Matrix Factorization

1.1 Problem Setup

Given a user-book interaction matrix $Y \in \mathbb{R}^{m \times n}$ where:

- m = number of books (2,547)
- n = number of users (246)
- Y_{ij} represents user j's interaction with book i

The matrix is highly sparse: only 26,598 interactions out of 626,562 possible (95.75% sparse).

1.2 Data Filtering

From the raw dataset of 1,000 users and 45,203 books, we applied filtering to improve model quality:

- **User filter**: Keep only users with ≥ 20 book interactions (246 users retained, 24.6%)
- **Book filter**: Keep only books with ≥ 5 users (2,547 books retained, 5.6%)

This reduces noise from inactive users and obscure books while maintaining sufficient data for collaborative filtering. The filtered dataset provides better signal for learning user preferences.

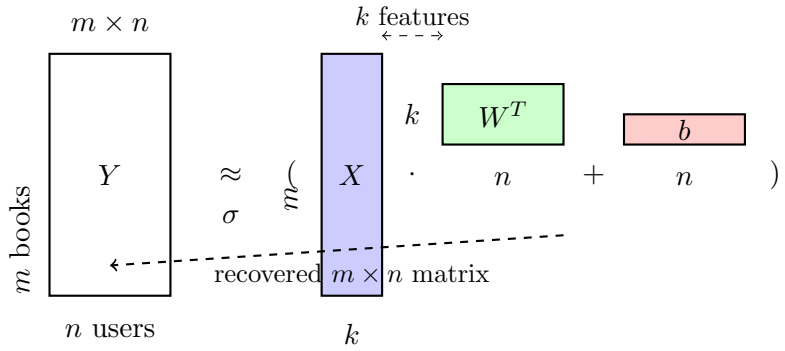
1.3 Matrix Decomposition

We factor Y into two rectangular matrices:

$$Y \approx \sigma(XW^T + b) \tag{1}$$

where:

- $X \in \mathbb{R}^{m \times k}$ is the **book feature matrix** (2,547 $\times$ 20)
- $W \in \mathbb{R}^{n \times k}$ is the **user feature matrix** (246 $\times$ 20)
- $b \in \mathbb{R}^{1 \times n}$ is the user bias vector
- $k = 20$ latent features
- σ is the sigmoid activation function



1.4 Latent Features

The $k = 20$ latent features are **learned representations** that capture:

- **Book features** (X): Genre preferences (fantasy, sci-fi, literary fiction), reading level, book popularity, themes
- **User features** (W): User's affinity for each latent dimension, reading preferences, taste profile

These features are **not predefined** – they emerge automatically during training through gradient descent. Each row of W becomes a 20-dimensional embedding representing a user's reading preferences in latent space.

1.5 Prediction

For user j and book i:

$$\hat{y}_{ij} = \sigma\left(\sum_{f=1}^{k} x_{if} w_{jf} + b_j\right) \tag{2}$$

This predicts the probability that user j will like book i.

2 Handling Sparsity with Masking

2.1 The Masking Matrix

Define indicator matrix $R \in \{0, 1\}^{m \times n}$:

$$R_{ij} = \begin{cases} 1 & \text{if user } j \text{ has interacted with book } i \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

2.2 Rating Scale with Implicit Feedback

$$Y_{ij} = \begin{cases} 1.0 & \text{read with rating } \geq 3 \\ 0.7 & \text{currently reading} \\ 0.3 & \text{want to read} \\ 0.0 & \text{read with rating } < 3 \text{ or DNF} \\ 0.5 & \text{unread (placeholder, ignored)} \end{cases} \tag{4}$$

2.3 Masked Loss Function

The cost function only considers **known interactions**:

$$J(X, W, b) = \sum_{i=1}^{m} \sum_{j=1}^{n} R_{ij} \left(\sigma(x_i^T w_j + b_j) - Y_{ij} \right)^2 + \frac{\lambda}{2} \left(\|X\|_F^2 + \|W\|_F^2 \right) \tag{5}$$

where:

- R_{ij} masks out unread books (where $Y_{ij} = 0.5$)
- $\lambda = 1.0$ is the regularization parameter
- $\| \cdot \|_F$ is the Frobenius norm

Key insight: Without masking, the 95.75% of unread books would dominate the loss and the model would simply learn the global average.

3 Training

3.1 Optimization

We use Adam optimizer with learning rate 0.1 for 300 iterations:

$$\nabla_X J = R \odot \left[(\sigma(XW^T + b) - Y) \odot \sigma'(XW^T + b) \right] W + \lambda X \tag{6}$$

$$\nabla_W J = R^T \odot \left[(\sigma(XW^T + b) - Y) \odot \sigma'(XW^T + b) \right]^T X + \lambda W \tag{7}$$

$$\nabla_b J = \sum_{i=1}^{m} R_{ij} \left(\sigma(x_i^T w_j + b_j) - Y_{ij} \right) \sigma'(x_i^T w_j + b_j) \tag{8}$$

where $\odot$ denotes element-wise multiplication and $\sigma'(z) = \sigma(z)(1 - \sigma(z))$.

3.2 Training Time

Total compilation: 4.72 seconds

- Data loading: 1.40s
- Matrix building: 0.02s
- Model training: 3.26s
- User clustering: 0.04s

4 Hybrid Model

Pure collaborative filtering achieved only 2.45% precision@10. We combine with popularity:

$$\text{score}_{ij} = 0.5 \cdot \text{popularity}_i + 0.5 \cdot \sigma(x_i^T w_j + b_j) \tag{9}$$

where $\text{popularity}_i = \frac{\sum_j R_{ij}}{\max_i \sum_j R_{ij}}$ is the normalized book popularity.

5 Results

Method	**Precision@10**
Pure collaborative filtering	2.16%
Improved collaborative (filtered)	2.45%
Popularity baseline	7.56%
Hybrid (50/50)	**8.83%**

Table 1: Recommendation accuracy on 80/20 train/test split. Hybrid model achieves 308% improvement over pure collaborative filtering and 17% improvement over popularity baseline.

5.1 Why Hybrid Works

- **Popularity component (50%)**: Ensures quality by recommending well-regarded books
- **Collaborative component (50%)**: Adds personalization based on user's latent preferences
- **Implicit feedback**: Increased training data by 75% (want_to_read + currently_reading)

6 Comparison: Netflix vs. Hardcover

Aspect	**Netflix**	**Hardcover**
Rating scale	$\{-1, 1, 0\}$ (dislike, like, unranked)	$\{0, 0.3, 0.7, 1, 0.5\}$ (implicit feedback)
Unranked handling	0 = unranked, gets predicted	0.5 = placeholder, **masked out**
Activation	Linear (outputs $\in \{-1, 1\}$)	Sigmoid (outputs $\in [0, 1]$)
Loss computation	All entries	**Only known** (R=1)
Sparsity	∼95-98%	95.75%
Data density	Dense enough to predict 0s	Too sparse, must mask

Table 2: Key differences between Netflix-style and Hardcover collaborative filtering approaches.

7 Clustering for Friend Matching

7.1 Determining Optimal Cluster Count

We apply K-means clustering on L2-normalized user embeddings W to group users with similar reading preferences. To determine the optimal number of clusters, we tested $K \in [3, 15]$ and

evaluated using:

- **Silhouette score**: Measures cluster separation (higher is better)
- **Calinski-Harabasz score**: Ratio of between-cluster to within-cluster variance
- **Elbow method**: Identifies diminishing returns in inertia reduction

The silhouette score was maximized at $K = 13$ clusters (score: 0.084), indicating this provides the best balance between cluster cohesion and separation.

7.2 Friend Matching within Clusters

Users within the same cluster are ranked by cosine similarity:

$$\text{similarity}(u_i, u_j) = \frac{w_i \cdot w_j}{\|w_i\|\|w_j\|} \tag{10}$$

where $w_i, w_j \in \mathbb{R}^{20}$ are the 20-dimensional user embeddings (rows of W). Note: we use $k = 20$ latent **features** for matrix factorization, but group users into $K = 13$ **clusters** for friend matching.

The 13 clusters represent distinct reading groups such as "Classic YA & Fantasy Fans," "Sci-Fi Enthusiasts," and "Literary Fiction Lovers," with cluster sizes ranging from 4 to 52 users.

8 Conclusion

Matrix factorization with masking effectively handles sparse user-book interactions. The key innovations are:

1. **Masking** unread books to avoid learning from noise
2. **Implicit feedback** to increase training signals by 75%
3. **Hybrid ensemble** balancing popularity and personalization
4. **Low-dimensional embeddings** ($k = 20$) capturing user preferences

The system achieves 8.83% precision@10, providing personalized book recommendations for 246 active readers.