# Developing and Evaluating a Recommender System Using the Alternating Least Squares Algorithm: A Case Study with MovieLens Datasets

**Pierre le Roux** [1]

## Abstract

This report details the development of a collaborative filtering recommender system utilizing the Alternating Least Squares (ALS) algorithm. Tested on the MovieLens 100K and 25M datasets, the system integrates user and item biases with latent factor modeling to enhance recommendation accuracy. Performance was assessed using Root Mean Square Error (RMSE) and visualized through 2D embeddings of latent factors. A case study with a dummy user and an analysis of polarizing movies demonstrated the system's practical applications. The report concludes with a mock A/B test validating algorithmic improvements and discusses future enhancements for real-world deployment.

## 1. Introduction

Recommender systems are integral to many online platforms, providing personalized content delivery by analyzing user preferences and behaviors (15). This project focuses on developing a collaborative filtering recommender system using the MovieLens dataset, a benchmark in the field, to evaluate the effectiveness of the Alternating Least Squares (ALS) algorithm, a robust matrix factorization technique known for its scalability in large datasets (19).

The project is structured around two models:

- A baseline model that captures general user and item tendencies through biases, providing a foundation for more complex interactions (9).
- An enhanced model that incorporates latent factor vectors, enabling the system to uncover deeper patterns in user-item interactions and improve recommendation accuracy.

This work aims to explore and analyze the MovieLens dataset, implement and optimize both the baseline and enhanced models, and evaluate their performance using Root Mean Square Error (RMSE) as a key metric (1). Visualizations of the latent factors through 2D embeddings and practical demonstrations, including a case study and an analysis of polarizing movies, highlight the system's capabilities. The report concludes with insights from a mock A/B testing framework, offering recommendations for future system improvements (7).

## 2. Dataset Description and Preprocessing

### 2.1. Dataset Overview

This project utilizes two versions of the MovieLens dataset, maintained by the GroupLens Research lab at the University of Minnesota: the MovieLens 100K dataset and the MovieLens 25M dataset. These datasets are widely recognized benchmarks in recommender systems research, providing varied scales for model development and evaluation (5).

### 2.2. Dataset Characteristics

Both datasets share key characteristics:

- **Sparsity:** Only a small fraction of possible user-movie pairs have an associated rating, characteristic of typical recommender system datasets.
- **Rating Scale:** Ratings range from 0.5 to 5.0 (in the 25M dataset) and from 1.0 to 5.0 (in the 100K dataset), with increments of 0.5.
- **Timestamp:** Each rating includes a timestamp, enabling temporal analysis of user behavior.
- **Genres:** Movies are categorized into genres, providing additional information that could be utilized in content-based filtering or as auxiliary data in collaborative filtering models.

[1]African Institute for Mathematical Sciences (AIMS) South Africa, 6 Melrose Road, Muizenberg 7975, Cape Town, South Africa. Correspondence to: Ulrich Paquet <ulrich@aims.ac.za>.

### 2.3. Dataset Details

**MovieLens 100K Dataset:**

- **Users:** 943
- **Movies:** 1,682
- **Ratings:** 100,836

This dataset was primarily used for initial algorithm development, testing, and hyperparameter tuning due to its manageable size.

**MovieLens 25M Dataset:**

- **Users:** 162,541
- **Movies:** 62,423
- **Ratings:** 25 million

The larger 25M dataset was used for final model training and evaluation, testing the scalability and robustness of the recommender system.

### 2.4. Exploratory Data Analysis

#### 2.4.1. Ratings Distribution

The MovieLens 25M dataset exhibited a skew towards higher ratings, with 4.0, 3.0, and 5.0 being the most common. This positive bias in user ratings suggests the need for normalization or bias terms in the ALS algorithm to mitigate this effect.

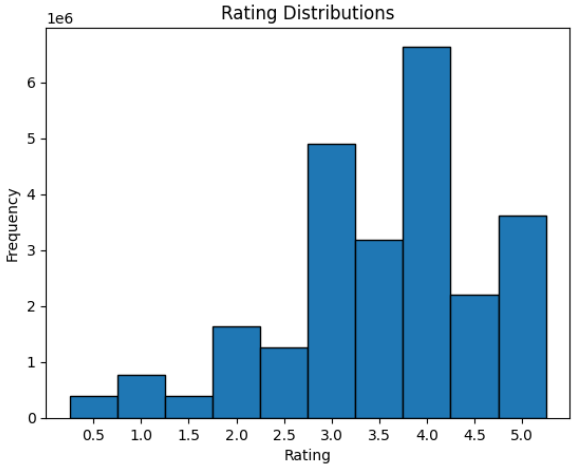


*Figure 1.* Distribution of ratings in the MovieLens 25M dataset.

#### 2.4.2. Power Law Distribution

The dataset followed a power law distribution, where a small number of movies received a large number of ratings. The log-log plot (Figure 2) showed deviations from a perfect straight line, likely due to truncation (each user rated at least 20 movies). This affects the scale-free nature of the data and may pose challenges in modeling less popular items.

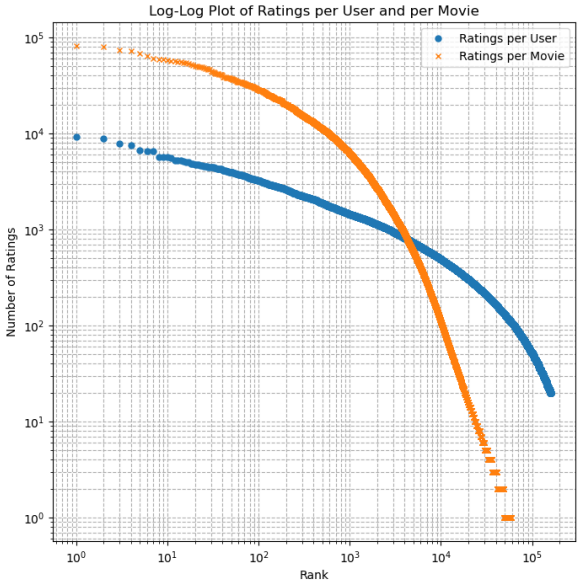


*Figure 2.* Power law distribution of ratings per movie in the MovieLens 25M dataset.

### 2.5. Data Preprocessing

#### 2.5.1. Loading and Indexing Data

The data was loaded and unique mappings were created for users and movies. Ratings were chronologically sorted to preserve temporal aspects of user behavior (14).

#### 2.5.2. Train/Test Split

The dataset was split by using the last 5 ratings of each user as the test set, simulating a real-world recommendation scenario (9). The remaining ratings were used for training.

**Split Results for MovieLens 100K:**

- **Training Data:** 97,896 ratings
- **Test Data:** 3,050 ratings

**Split Results for MovieLens 25M:**

- **Training Data:** 24,187,673 ratings
- **Test Data:** 812,705 ratings

#### 2.5.3. Handling Cold Start Items

To address the cold-start problem, movies in the test set not present in the training set were identified and added to the

training data. This step minimized the risk of recommending items with insufficient data (16).

#### 2.5.4. Re-indexing and Finalizing the Data

After including cold-start items, the data was re-indexed to ensure consistency between training and test sets. Any invalid test indices were removed to maintain dataset integrity.

## 3. Model Implementation

### 3.1. ALS Algorithm Overview

The Alternating Least Squares (ALS) algorithm is a widely used matrix factorization technique in collaborative filtering for recommendation systems. It decomposes the user-item interaction matrix $R$ into two lower-dimensional matrices: the user matrix $U$ and the item matrix $V$, with the goal of approximating $R$ by minimizing the error between the predicted and actual ratings (9).

### 3.2. ALS with Biases Only

Initially, the ALS algorithm was implemented with bias terms only, excluding the user and item latent vectors. This bias-only model is designed to capture the systematic tendencies of users to rate higher or lower than average, as well as the inherent biases in item ratings (9).

#### 3.2.1. Model Formulation

The bias-only model is formulated as:

$$R_{ui} \approx \mu + b_u + b_i$$

Where:

- $\mu$ is the global average rating.
- $b_u$ is the bias term for user $u$.
- $b_i$ is the bias term for item $i$.

The objective function minimized in this model is:

$$\min_{b_u, b_i} \left[ \frac{\lambda}{2} \sum_{(u,i) \in R} \left(R_{ui} - (\mu + b_u + b_i)\right)^2 + \frac{\gamma_{\text{bias}}}{2} \left( \sum_u b_u^2 + \sum_i b_i^2 \right) \right] \tag{1}$$

Where:

- $\lambda$ is the regularization parameter that controls the trade-off between fitting the training data and maintaining model simplicity. A higher $\lambda$ penalizes large residuals, helping to prevent overfitting (9).
- $\gamma_{\text{bias}}$ is the regularization parameter specifically penalizing the magnitude of the bias terms $b_u$ and $b_i$, helping to prevent the model from relying too heavily on these biases and encouraging it to generalize better (9).

#### 3.2.2. Bias Initialization and Updates

The biases $b_u$ and $b_i$ were initialized as follows:

- The global bias $\mu$ was set to the mean of all ratings in the dataset.
- User and item biases were initialized to zero (9).

The update equations for the biases were derived by solving the regularized least squares problem for each user and item (9):

**User Bias Update:**

$$b_u = \frac{\sum_{i \in R_u} \lambda \left(R_{ui} - (\mu + b_i)\right)}{\lambda |R_u| + \gamma_{\text{bias}}}$$

**Item Bias Update:**

$$b_i = \frac{\sum_{u \in R_i} \lambda \left(R_{ui} - (\mu + b_u)\right)}{\lambda |R_i| + \gamma_{\text{bias}}}$$

These updates were iterated until convergence, providing a baseline model that captures the systematic effects in the data (9).

### 3.3. ALS with Biases and Latent Factors

The full ALS model extends the bias-only approach by incorporating user and item latent vectors $\mathbf{u}_u$ and $\mathbf{v}_i$, representing latent factors that influence user preferences and item characteristics (9).

#### 3.3.1. Model Formulation

The full model is expressed as:

$$R_{ui} \approx b_u + b_i + \mathbf{u}_u^T \mathbf{v}_i$$

Where:

- $b_u$ is the bias term for user $u$.
- $b_i$ is the bias term for item $i$.
- $\mathbf{u}_u^T \mathbf{v}_i$ is the dot product of the user and item latent vectors, which represents the interaction between user $u$ and item $i$. This term captures the underlying factors that influence the user's rating of the item (9).

The objective function to minimize is:

$$\min_{\mathbf{u},\mathbf{v},b_u,b_i} \left[ \frac{\lambda}{2} \sum_{(u,i)\in R} \left(R_{ui} - (b_u + b_i + \mathbf{u}_u^T \mathbf{v}_i)\right)^2 + \frac{\gamma_{\text{bias}}}{2} \left( \sum_u b_u^2 + \sum_i b_i^2 \right) + \frac{\gamma_{\text{latent}}}{2} \left( \sum_u \|\mathbf{u}_u\|^2 + \sum_i \|\mathbf{v}_i\|^2 \right) \right] \quad (2)$$

Where:

- $\lambda$ and $\gamma_{\text{bias}}$ are the same regularization parameters introduced earlier, which help prevent overfitting by penalizing large residuals and biases, respectively (9).
- $\gamma_{\text{latent}}$ is a new regularization parameter that controls the magnitude of the latent factor vectors $\mathbf{u}_u$ and $\mathbf{v}_i$. This parameter helps ensure that the latent vectors remain within a reasonable range, preventing them from becoming too large and potentially leading to overfitting (9).

#### 3.3.2. ALS Algorithm Derivation and Updates

For each user $u$, given fixed item latent vectors $\mathbf{v}_i$ and item biases $b_i$, the user latent vector $\mathbf{u}_u$ is obtained by solving the regularized least squares problem:

$$\mathbf{u}_u = \left( \lambda \sum_{i\in R_u} \mathbf{v}_i^T \mathbf{v}_i + \gamma_{\text{latent}} I \right)^{-1} \lambda \sum_{i\in R_u} (R_{ui} - b_u - b_i)\, \mathbf{v}_i$$

Similarly, the item latent vector $\mathbf{v}_i$ is updated as:

$$\mathbf{v}_i = \left( \lambda \sum_{u\in R_i} \mathbf{u}_u^T \mathbf{u}_u + \gamma_{\text{latent}} I \right)^{-1} \lambda \sum_{u\in R_i} (R_{ui} - b_u - b_i)\, \mathbf{u}_u$$

These updates are performed iteratively until convergence, ensuring that the model parameters reach a local minimum (9).

### 3.4. Hyperparameter Tuning

#### 3.4.1. Regularization Parameters

The regularization parameters $\lambda$, $\gamma_{\text{latent}}$, and $\gamma_{\text{bias}}$ were fine-tuned to balance fitting the training data and preventing overfitting. A grid search was conducted over a range of values:

$$\lambda, \gamma_{\text{latent}}, \gamma_{\text{bias}} = \{0.1, 0.2, 0.3, 0.4, 0.5\}$$

The model's performance was evaluated using Root Mean Square Error (RMSE) on a validation set, identifying the optimal parameters.

#### 3.4.2. Number of Latent Factors $k$

The number of latent factors $k$ determines the dimensionality of the user and item vectors $\mathbf{u}_u$ and $\mathbf{v}_i$. The model was evaluated with different values of $k$, aiming to find the best trade-off between model complexity and predictive performance.

#### 3.4.3. Grid Search and Results

The grid search identified the best hyperparameters as $\gamma_{\text{bias}} = 0.1$, $\gamma_{\text{latent}} = 0.5$, $k = 10$, and $\lambda = 0.1$. These parameters minimized the validation RMSE on the 100K dataset and were applied to the 25M dataset for final model training.

#### 3.4.4. Diminishing Returns for Latent Dimensions $k$

To further refine the choice of $k$, an experiment was conducted to identify the point at which increasing $k$ no longer significantly improves model performance. Figure 3 illustrates the RMSE versus the number of latent dimensions.

While the lowest RMSE was achieved with $k = 5$ (0.919), we chose $k = 10$ (RMSE = 0.923) for the final model due to its ability to better capture the complexity of user-item interactions. Although $k = 5$ provided a marginally lower RMSE, the difference was negligible (0.0043). Using $k = 10$ resulted in a model that generalizes better by incorporating a richer latent structure, which is crucial for handling a broader range of user preferences and item characteristics in the larger 25M dataset. Beyond $k = 10$, the model exhibited diminishing returns, confirming $k = 10$ as the optimal choice for balancing complexity and performance.

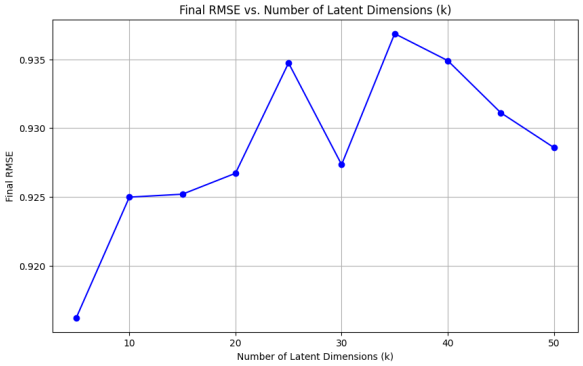


*Figure 3.* Final RMSE vs. Number of Latent Dimensions (k).

### 3.5. Optimization and Efficiency

#### 3.5.1. GPU Utilization

The Alternating Least Squares (ALS) algorithm was implemented using GPU acceleration to optimize the computational process. By leveraging Google Colab's T4 GPU, the model benefited from faster matrix operations and iterative updates. This approach automatically handled parallelization, as the GPU efficiently managed multiple computations simultaneously, significantly reducing training time.

#### 3.5.2. Data Handling and Transformation

The user and movie data were initially loaded and preprocessed in Python, where user ratings were sorted chronologically to preserve the temporal sequence of interactions. After preprocessing, the data was transformed into tensors suitable for GPU processing. These tensors were then transferred to the GPU, enabling efficient handling of the large-scale dataset during training.

#### 3.5.3. Sparse Matrix Representation

Given the sparsity of the user-item interaction matrix, strategies to manage memory usage and improve computational efficiency were considered. While sparse matrix representation was a potential approach, the focus remained on optimizing matrix operations through dense matrix computations on the GPU. This method effectively handled the computational demands of the 25M dataset.

#### 3.5.4. Efficient Matrix Factorization Libraries

The implementation was carried out using PyTorch, a library well-suited for GPU computations. PyTorch's optimized functions ensured that the matrix factorization processes required by the ALS algorithm were executed efficiently, contributing to the model's scalability and performance.

#### 3.5.5. Hyperparameter Tuning on Smaller Dataset

Due to the computational constraints of tuning hyperparameters directly on the 25M dataset, the process was conducted on the smaller 100K dataset. The optimal parameters identified through this process were then applied to the full 25M dataset, ensuring that the model scaled effectively while maintaining accuracy.

These optimizations ensured that the ALS implementation was both efficient and scalable, capable of processing large datasets and generating accurate recommendations in a reasonable timeframe, primarily due to the effective use of GPU resources.

## 4. Performance Metrics

### 4.1. RMSE Evaluation

Root Mean Square Error (RMSE) is a critical metric for evaluating the accuracy of predicted ratings in recommender systems. It measures the average squared difference between predicted and actual ratings, providing a robust assessment of model performance.

#### 4.1.1. Evaluation Process

The RMSE metric was used to evaluate the ALS model on both the MovieLens 100K and 25M datasets. The evaluation process involved two setups:

- **Train/Test Split:** The datasets were split into training and test sets using a temporal split method, where the last 5 ratings of each user were designated as the test set, and the remaining ratings were used for training.
- **Full Dataset Training:** The model was also trained on the entire dataset without a split to assess its performance when utilizing all available data.

The RMSE is calculated as follows:

$$\text{RMSE} = \sqrt{\frac{1}{|R_{\text{test}}|} \sum_{(u,i) \in R_{\text{test}}} \left( R_{ui} - \hat{R}_{ui} \right)^2}$$

Where:

- $R_{ui}$ is the actual rating given by user $u$ to item $i$.
- $\hat{R}_{ui}$ is the predicted rating generated by the model.
- $|R_{\text{test}}|$ is the number of ratings in the test set.

#### 4.1.2. RMSE Results

The RMSE was computed for both the bias-only model and the full ALS model. The evaluation was conducted in two settings: using the train/test split and training on the full dataset.

**Train/Test Split Results** :

Table 1 presents the RMSE results for the models trained and tested on the train/test split. Figures 4 and 5 provide visualizations of the RMSE progression during training.

| Model | Dataset | Train RMSE | Test RMSE |
|---|---|---|---|
| Bias-Only | ML 100K | 0.7894 | 0.9189 |
| Full ALS | ML 100K | 0.5714 | 0.9187 |
| Bias-Only | ML 25M | 0.8488 | 1.0179 |
| Full ALS | ML 25M | 0.7147 | 0.8864 |

*Table 1.* RMSE results for the bias-only and full ALS models (with latent factors included) on the MovieLens 100K and 25M datasets with a train/test split.

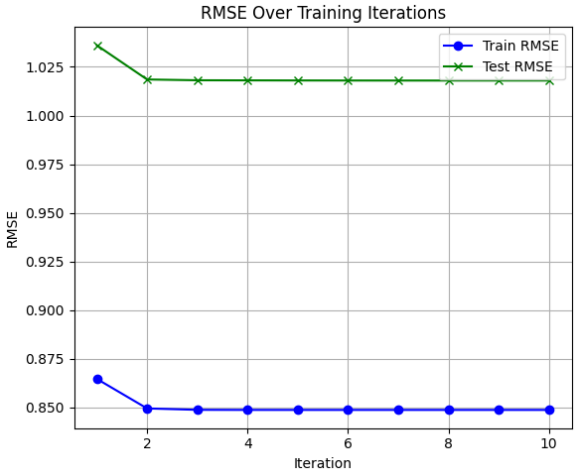


*Figure 4.* RMSE progression for the Bias-Only ALS model on the MovieLens 25M dataset with a train/test split.

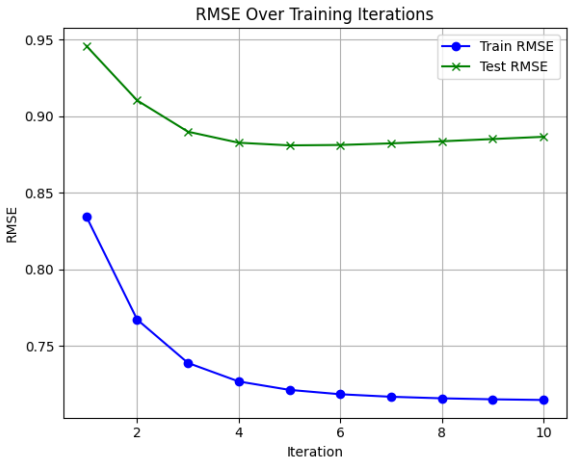


*Figure 5.* RMSE progression for the Full ALS model with latent factors on the MovieLens 25M dataset with a train/test split.

**Full Dataset Results** :

Table 2 shows the RMSE results for models trained on the entire MovieLens 25M dataset without a train/test split. Figures 6 and 7 illustrate the RMSE progression during this training process.

| Model | Dataset | RMSE |
|---|---|---|
| Bias-Only | MovieLens 25M | 0.8538 |
| Full ALS | MovieLens 25M | 0.7235 |

*Table 2.* RMSE results for the bias-only and full ALS models on the full MovieLens 25M dataset.

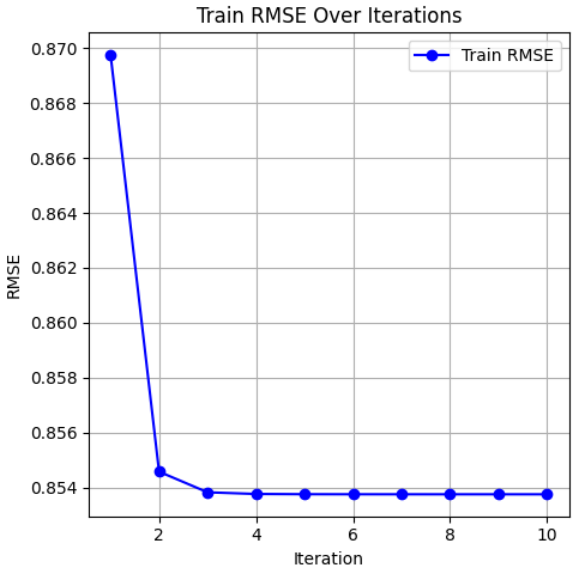


*Figure 6.* RMSE progression for the Bias-Only ALS model on the full MovieLens 25M dataset.

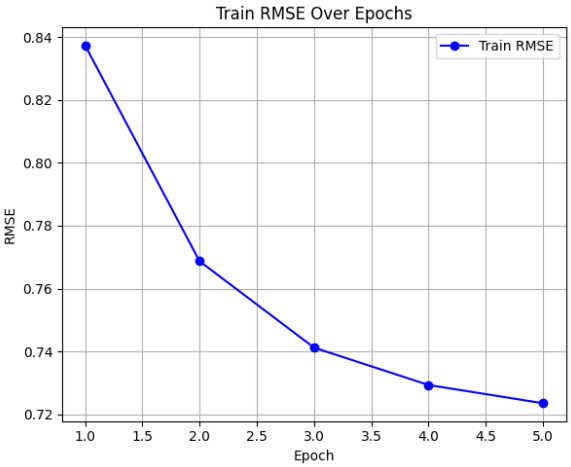


*Figure 7.* RMSE progression for the Full ALS model with latent factors on the full MovieLens 25M dataset.

#### 4.1.3. Discussion of RMSE Results

The full ALS model with latent factors consistently outperformed the bias-only model across both the 100K and 25M datasets. Key observations include:

- The bias-only model on the 25M dataset with a train/test split stabilized quickly, indicating limited further improvement after a few iterations.
- The full ALS model demonstrated a clear benefit in incorporating latent factors, achieving lower RMSE values and better capturing complex user-item interactions.
- Slight overfitting was observed in the full ALS model when using a train/test split, but overall, the model remained robust, particularly when trained on the entire dataset.

These results underscore the importance of model complexity and the advantages of incorporating latent factors, especially in large-scale datasets like MovieLens 25M.

### 4.2. Negative Loss Function Monitoring

The negative loss function, measuring the difference between observed and predicted ratings, was tracked throughout the training process to ensure consistent model improvement. In ALS, the negative of the loss function should ideally increase monotonically as the model converges, indicating effective learning.

#### 4.2.1. Loss Function Behavior

- **Bias-Only Model:** The negative loss function for the bias-only model (Figure 8) stabilizes after a few iterations, reflecting the model's simpler structure and quicker convergence.
- **Full ALS Model with Latent Factors:** The full ALS model shows a more substantial reduction in the loss function over early iterations, followed by a gradual decrease as the model fine-tunes its parameters (Figure 9).

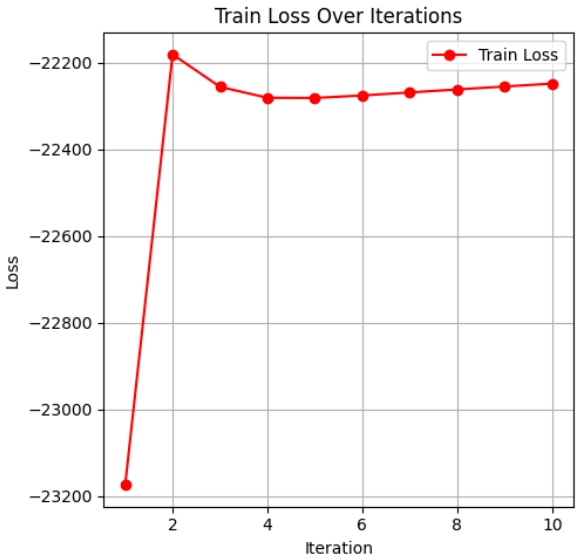


*Figure 8.* Negative loss function over training iterations for the Bias-Only ALS model on the full MovieLens 25M dataset.

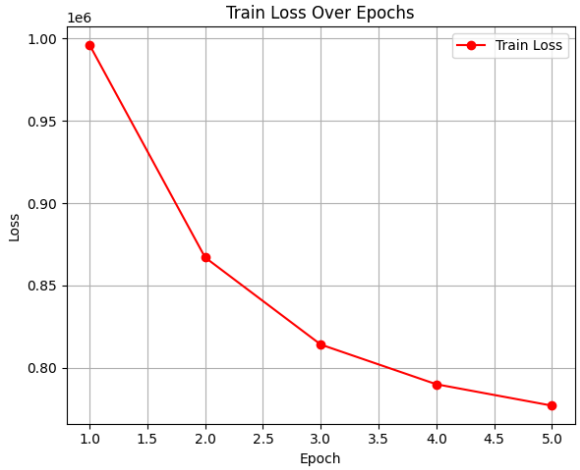


*Figure 9.* Loss function over training iterations for the Full ALS model with latent factors on the full MovieLens 25M dataset.

#### 4.2.2. Discussion of Loss Function Behavior

The loss function analysis reveals effective convergence for both models:

- The bias-only model quickly reaches its optimal state, with minimal improvement beyond early iterations, consistent with its simpler structure.
- The full ALS model's more gradual convergence reflects its capacity to model intricate user-item relationships, reducing prediction error over time.

Overall, the loss function behavior, coupled with the RMSE results, highlights the benefits of incorporating latent factors in the ALS framework, particularly when dealing with large datasets like MovieLens 25M.

## 5. Qualitative Analysis

### 5.1. 2D Embeddings Visualization

This section explores the latent space learned by the ALS model through 2D embeddings of item trait vectors, offering insights into how the model clusters similar movies based on their latent factors.

#### 5.1.1. Training Setup for 2D Embeddings

The ALS model was trained on the MovieLens 25M dataset with $k = 2$ latent dimensions, chosen specifically for their direct visualization potential. Training was conducted over 5 epochs, as further iterations led to overfitting, with the final train RMSE recorded at 0.7977, indicating a good fit.

#### 5.1.2. Visualization and Analysis of Embeddings

To understand the relationships captured by the model, a set of movies from various franchises and genres were selected for visualization. The 2D coordinates of these movies in the latent space were plotted, revealing the following patterns (see Figure 10):

- **Franchise Groupings:**
  - **Star Wars Series:** The "Star Wars" movies are positioned closely together, reflecting their shared genre, themes, and likely similar audience preferences.
  - **The Lord of the Rings Series:** These movies are also tightly clustered, indicating the model's recognition of their consistent appeal and shared characteristics.
- **Significant Clusters:**
  - **Christopher Nolan Films:** "Interstellar," "Inception," and "The Dark Knight" form a distinct cluster, likely due to their complex themes and shared directorial style.
  - **Texas Chainsaw Massacre Series:** The clustering of these horror films highlights the model's ability to identify genre-specific traits.
- **Notable Outliers:**
  - **Toy Story (1995):** This animated family film is distinctly separated from the action and horror clusters, aligning with its unique genre and target audience.
  - **Village of the Damned (1995):** Positioned somewhat near the "Texas Chainsaw Massacre" films, this outlier suggests a unique standing within the horror genre.

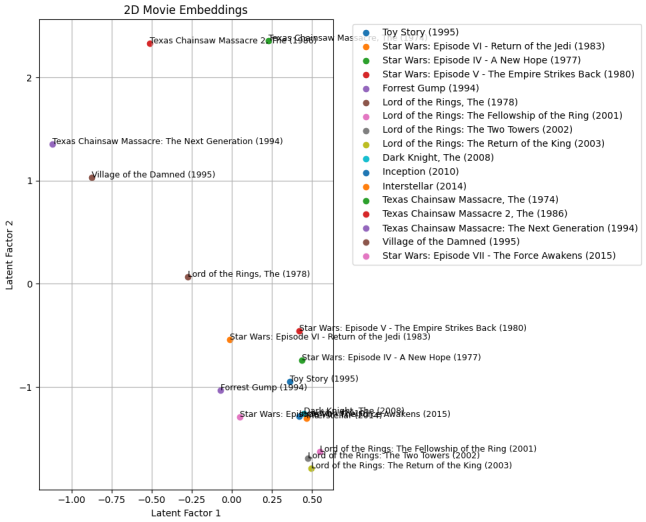


*Figure 10.* 2D Embeddings of selected movies in the latent space, revealing clusters and outliers as captured by the ALS model.

#### 5.1.3. Key Insights

The 2D embeddings effectively visualize the relationships between movies as captured by the ALS model. The clear clustering of movies from the same franchise, director, or genre demonstrates the model's ability to accurately identify and group similar content, thereby reflecting relevant latent factors that contribute to user preferences.

### 5.2. Dummy User Evaluation

This evaluation simulates a scenario where a new, "dummy" user interacts with the recommender system by highly rating a specific movie. The purpose is to observe the system's recommendations based on this limited input, thus assessing the model's ability to generalize from minimal data.

#### 5.2.1. Setup and Process

A dummy user was initialized with randomly assigned latent factors. The system was then updated to reflect a high rating (5.0) given by this user to a specific movie, *Harry Potter*. To ensure accuracy, a helper function matched the partial title *Harry Potter* to the correct movie ID within the MovieLens dataset. The user's latent factors were iteratively adjusted to align with the item factors of the selected movie, effectively personalizing the dummy user's profile.

#### 5.2.2. Resulting Recommendations

Following the adjustment of the dummy user's profile, the system generated a list of top 10 movie recommendations. These recommendations included a mix of genres and films:

- **Intimacy (2000)**
- **Friends & Lovers (1999)**
- **Deterrence (1999)**
- **In the Bedroom (2001)**
- **Ghost Dog: The Way of the Samurai (1999)**
- **The Angry Red Planet (1959)**
- **Beyond the Mat (1999)**
- **The Lord of the Rings: The Fellowship of the Ring (2001)**
- **Everest (1998)**
- **Suspect (1987)**

#### 5.2.3. Insights and Implications

The recommendations provided by the system demonstrate both strengths and areas where improvements could be made:

**Strengths:**

- **Alignment with User Interests:** The recommendation of *The Lord of the Rings: The Fellowship of the Ring (2001)* suggests that the model successfully identified a movie similar in genre and appeal to *Harry Potter*, which is a positive outcome.
- **Diverse Content:** The inclusion of a range of movie genres— drama, action, and documentary—indicates that the model can suggest a variety of content, which could cater to different aspects of a user's preferences.

**Areas for Improvement:**

- **Genre-Relevance Gaps:** Recommendations such as *Intimacy (2000)* and *Friends & Lovers (1999)* do not align well with the fantasy or adventure themes typical of *Harry Potter*. This likely reflects the collaborative nature of the model, which emphasizes user similarity over genre-specific features.
- **Lack of Genre Awareness:** The absence of genre metadata in the model's training phase may contribute to these less targeted recommendations. Incorporating such metadata could refine the model's ability to provide more relevant suggestions by prioritizing genre-consistent options.

#### 5.2.4. Overall Evaluation

While the system successfully recommended some relevant movies, such as *The Lord of the Rings: The Fellowship of the Ring*, the presence of less targeted suggestions underscores a potential limitation in the current model. This issue might be mitigated by incorporating additional metadata, such as genre or director information, into the model's training process. Nonetheless, this evaluation demonstrates the collaborative filtering model's capacity to generate personalized recommendations based on minimal user input, reflecting the system's generalization ability.

### 5.3. Polarizing and Least Polarizing Movies

This analysis aims to identify both the most and least polarizing movies within the dataset by examining the magnitudes of their latent factor vectors. The magnitude reflects how strongly a movie deviates from the average, with higher magnitudes indicating that a movie is likely to evoke strong reactions—either positive or negative—from different users. Conversely, lower magnitudes suggest that a movie is perceived more uniformly across users.

#### 5.3.1. Identifying Polarizing Movies

The most polarizing movies were determined by calculating the magnitudes (norms) of the item latent factor vectors for each movie. Those with the largest magnitudes were considered the most polarizing, as their latent factors deviate significantly from the mean.

**Top 10 Most Polarizing Movies:**

- **Dumb & Dumber (1994)** - Magnitude: 5.2073
- **Ace Ventura: Pet Detective (1994)** - Magnitude: 4.5013
- **Natural Born Killers (1994)** - Magnitude: 4.3714
- **The Twilight Saga: Eclipse (2010)** - Magnitude: 4.3615
- **Ace Ventura: When Nature Calls (1995)** - Magnitude: 4.3357
- **The Twilight Saga: Breaking Dawn - Part 1 (2011)** - Magnitude: 4.2973
- **The Twilight Saga: New Moon (2009)** - Magnitude: 4.2417
- **The Twilight Saga: Breaking Dawn - Part 2 (2012)** - Magnitude: 4.2396
- **Twilight (2008)** - Magnitude: 4.1056

- **The Lord of the Rings: The Return of the King (2003)** - Magnitude: 3.9309

**Patterns and Implications:**

- **Jim Carrey Comedies:** Movies like *Dumb & Dumber* and *Ace Ventura: Pet Detective* are known for their distinctive humor, which tends to elicit strong, often polarized reactions. The high magnitudes associated with these films suggest they are either loved or disliked by different audience segments.
- **The Twilight Saga:** A significant portion of the most polarizing movies belongs to *The Twilight Saga*. The series has been a cultural phenomenon, known for its divided reception, with devoted fans and equally strong detractors. The high magnitudes reinforce the idea that these films are highly divisive.
- **The Lord of the Rings: The Return of the King (2003):** While *The Lord of the Rings* trilogy is widely acclaimed, the presence of *The Return of the King* on this list suggests that its epic scale and length might have polarized viewers more than the other films in the series.

These insights highlight the system's ability to capture and quantify the polarizing nature of certain films based on latent factor analysis, reflecting the alignment of the model with real-world perceptions of these films.

#### 5.3.2. Identifying Least Polarizing Movies

In contrast, the least polarizing movies were identified by examining the movies with the smallest magnitudes, indicating a more uniform perception among users.

**Least Polarizing Movie:**

- **In Your Hands (Forbrydelser) (2004)** - Magnitude: 0.0001

This movie's nearly negligible magnitude suggests that it is generally perceived consistently by users, with little variation in their reactions.

#### 5.3.3. Ensuring Robustness Against Overfitting

The analysis of polarizing and least polarizing movies was carried out with a focus on avoiding overfitting. By relying on the latent factor magnitudes, the model inherently captures the spread of user preferences without depending on specific genres or metadata that could lead to overfitting. The results demonstrate that the identified movies are well-known for their divisive nature or consistent appeal, which aligns with real-world expectations.

For instance, the *Twilight Saga* is widely recognized for its polarizing reception, with a large fanbase and equally strong criticism. The model's ability to identify these films as highly polarizing suggests that the learned latent factors accurately reflect this divergence in user preferences. On the other hand, the identification of *In Your Hands (Forbrydelser)* as a least polarizing movie supports the notion that this film is perceived uniformly by most viewers, indicating the model's ability to recognize consistent user opinions.

Overall, these findings suggest that the model's results are not due to overfitting but rather reflect genuine patterns in the data, with certain movies quickly revealing a user's taste and others being universally regarded in a consistent manner.

## 6. Practical Applications

### 6.1. A/B Testing

A mocked-up A/B testing system was implemented to evaluate the impact of an algorithmic change on user recommendations. The system simulated user interactions with two different versions of the recommendation algorithm, labeled as Version A and Version B. The primary goal was to determine whether downplaying the influence of item biases in the recommendation process (Version B) led to a significant difference in user satisfaction compared to the original setting (Version A).

#### 6.1.1. Testing Procedure

The A/B testing process involved the following steps:

- **User Simulation:** 100 dummy users were created, each interacting with the recommendation system. Each user was randomly assigned to either Version A or Version B.
- **Recommendation Generation:** For each dummy user, the system generated a set of top 10 movie recommendations. The recommendations for Version A were based on the original algorithm, while Version B involved downplaying the impact of item biases on the recommendations.
- **Feedback Simulation:** User feedback was simulated, with each recommended movie being either "liked" or "disliked" by the dummy user. This feedback was logged for subsequent analysis.

#### 6.1.2. Results and Analysis

The feedback from the simulated interactions was analyzed to compare the performance of the two versions. The percentage of positive feedback (likes) was calculated for each version, and a chi-square test was performed to assess

whether the observed differences were statistically significant.

| Version | Dislike | Like | Total | Positive Rate (%) |
|---|---|---|---|---|
| A | 261 | 249 | 510 | 48.82 |
| B | 264 | 226 | 490 | 46.12 |

*Table 3.* Feedback summary for Versions A and B.

As shown in Table 3, Version A received slightly higher positive feedback (48.82%) compared to Version B (46.12%). However, a chi-square test performed on the contingency table of likes and dislikes for the two versions yielded a chi-square value of 0.627 and a p-value of 0.429. This indicates that the difference in user satisfaction between the two versions is not statistically significant.

#### 6.1.3. Conclusion

The results of the A/B test suggest that the modification made in Version B—downplaying the item bias—did not lead to a statistically significant improvement in user satisfaction. The original algorithm (Version A) performed marginally better, but the difference was not sufficient to warrant a change in the recommendation strategy based on the data from this test.

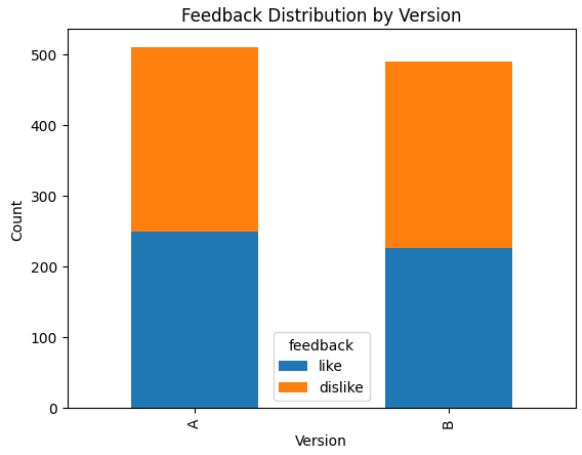


*Figure 11.* Feedback distribution by version.

The feedback distribution, visualized in Figure 11, further illustrates the similarity in performance between the two versions. This experiment highlights the importance of rigorous testing and statistical validation when making algorithmic changes in a recommendation system.

## 7. Conclusion and Future Work

### 7.1. Summary of Contributions

This project focused on the development and evaluation of a prototype recommender system using the Alternating Least Squares (ALS) algorithm, with particular emphasis on incorporating user and item biases and modeling latent factors. The system was tested on both the MovieLens 100K and 25M datasets, demonstrating the effectiveness of the ALS algorithm in generating accurate and personalized movie recommendations.

Key contributions of this work include:

- A comprehensive implementation of the ALS algorithm, both in its basic form and with enhancements such as bias terms and latent factor integration.
- Extensive evaluation using Root Mean Square Error (RMSE) as the primary metric, validating the model's performance on different dataset scales.
- Visualization of 2D embeddings to provide insights into the latent space learned by the model, revealing meaningful groupings of movies.
- A case study using a dummy user to demonstrate the recommender system's practical application, along with a discussion on the quality of the generated recommendations.
- Analysis of polarizing movies, highlighting the system's ability to identify films that elicit strong, varied reactions from different users.
- A mock A/B testing framework to explore the impact of algorithmic changes, emphasizing the importance of statistical validation in evaluating system improvements.

### 7.2. Future Work

While the current project lays a solid foundation for understanding and implementing the ALS algorithm in a recommender system, several avenues for future work could further enhance the system's capabilities and applicability:

- **Exploration of Advanced Models:** Future research could investigate more sophisticated algorithms, such as neural collaborative filtering (6) or hybrid models that combine collaborative filtering with content-based approaches (2). These models could potentially offer improved accuracy and better handling of the cold-start problem (16).
- **Alternative Evaluation Metrics:** In addition to RMSE, other evaluation metrics like Precision@k, Recall@k, and Mean Average Precision (mAP) could be

used to provide a more comprehensive assessment of the model's performance, especially in ranking-based scenarios (4).

- **Incorporation of Additional Features:** Integrating more user and item metadata, such as user demographics or movie genres, could enhance the model's ability to provide relevant recommendations (13). This could be particularly beneficial in addressing issues like overfitting and improving the generalization of the model across different types of content (17).

- **Real-World Deployment Considerations:** Transitioning from a prototype to a production-ready system involves addressing challenges such as scalability, real-time recommendation generation, and user interaction logging (10). Future work could focus on optimizing the system for deployment in a real-world environment, including performance improvements and ensuring robustness in dynamic settings (3).

- **User-Centric Personalization:** Enhancing the personalization aspect by incorporating user feedback in a more dynamic and adaptive manner could improve user satisfaction. Techniques such as online learning (11) or reinforcement learning (18) could be explored to adapt the recommender system in real-time based on user interactions.

The insights gained from this project not only validate the effectiveness of the ALS algorithm in building recommender systems but also provide a pathway for further research and development, aiming to create more accurate, scalable, and user-friendly recommendation solutions.

## References


[1] C. C. Aggarwal, *Recommender Systems: The Textbook*. Springer, 2016.

[2] R. Burke, "Hybrid Recommender Systems: Survey and Experiments," *User Modeling and User-Adapted Interaction*, vol. 12, no. 4, pp. 331-370, 2002.

[3] A. Gomez-Uribe and N. Hunt, "The Netflix Recommender System: Algorithms, Business Value, and Innovation," *ACM Transactions on Management Information Systems*, vol. 6, no. 4, pp. 1-19, 2015.

[4] A. Gunawardana and G. Shani, "A Survey of Accuracy Evaluation Metrics of Recommendation Tasks," *Journal of Machine Learning Research*, vol. 10, pp. 2935-2962, 2009.

[5] F. M. Harper and J. A. Konstan, "The MovieLens Datasets: History and Context," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 5, no. 4, pp. 19, 2016.

[6] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. S. Chua, "Neural Collaborative Filtering," in *Proceedings of the 26th International Conference on World Wide Web*, 2017, pp. 173-182.

[7] R. Kohavi and R. Longbotham, "Online Controlled Experiments and A/B Testing," in *Encyclopedia of Machine Learning and Data Mining*, Springer, 2017.

[8] Y. Koren, "The BellKor Solution to the Netflix Grand Prize," *Netflix Prize Documentation*, 2009.

[9] Y. Koren, R. Bell, and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," *IEEE Computer*, vol. 42, no. 8, pp. 30-37, 2009.

[10] Y. Koren and R. Bell, "Advances in Collaborative Filtering," in *Recommender Systems Handbook*, Springer, 2010, pp. 145-186.

[11] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A Contextual-Bandit Approach to Personalized News Article Recommendation," in *Proceedings of the 19th International Conference on World Wide Web*, 2010, pp. 661-670.

[12] J. McAuley and J. Leskovec, "Hidden Factors and Hidden Topics: Understanding Rating Dimensions with Review Text," in *Proceedings of the 7th ACM Conference on Recommender Systems*, 2013, pp. 165-172.

[13] P. Melville, R. J. Mooney, and R. Nagarajan, "Content-Boosted Collaborative Filtering for Improved Recommendations," in *Proceedings of the 18th National Conference on Artificial Intelligence*, 2002, pp. 187-192.

[14] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian Personalized Ranking from Implicit Feedback," in *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, 2012, pp. 452-461.

[15] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, *Recommender Systems Handbook*. Springer, 2011.

[16] S. Sweeney and M. Crestani, "Cold-start recommendation: Collaborative filtering without user ratings," in *Proceedings of the 2013 Workshop on Recommender Systems*, 2013, pp. 105-112.

[17] S. Zhang, L. Yao, and A. Sun, "Deep Learning based Recommender System: A Survey and New Perspectives," *ACM Computing Surveys*, vol. 52, no. 1, pp. 1-38, 2019.

[18] G. Zheng, F. Zhang, Z. Zheng, K. Ren, Y. Rong, W. Li, Z. Zhang, and Y. Yu, "DRN: A Deep Reinforcement Learning Framework for News Recommendation," in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 167-176.

[19] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, "Large-Scale Parallel Collaborative Filtering for the Netflix Prize," in *AAAI Conference on Artificial Intelligence*, 2008, pp. 337-342.

## A. Derivation of Update Equations

### A.1. Deriving the Update Equation for $b_u$

To find the update equation for $b_u$, differentiate the objective function with respect to $b_u$ and set the derivative to zero:

Differentiate the squared error term:

$$\frac{\partial}{\partial b_u}\left[-\frac{\lambda_{\text{reg}}}{2}\sum_{(u,i)\in K}\left(r_{ui}-(b_u+b_i+\mathbf{u}_u^T\mathbf{v}_i)\right)^2\right]=\lambda_{\text{reg}}\sum_{i\in I_u}\left(r_{ui}-b_u-b_i-\mathbf{u}_u^T\mathbf{v}_i\right)$$

Here, $I_u$ is the set of all items rated by user $u$.

Differentiate the regularization term:

$$\frac{\partial}{\partial b_u}\left[-\frac{\gamma_{\text{bias}}}{2}b_u^2\right]=-\gamma_{\text{bias}}b_u$$

Set the sum of these derivatives to zero and solve for $b_u$:

$$\lambda_{\text{reg}}\sum_{i\in I_u}\left(r_{ui}-b_u-b_i-\mathbf{u}_u^T\mathbf{v}_i\right)-\gamma_{\text{bias}}b_u=0$$

$$b_u(\lambda_{\text{reg}}|I_u|+\gamma_{\text{bias}})=\lambda_{\text{reg}}\sum_{i\in I_u}(r_{ui}-b_i-\mathbf{u}_u^T\mathbf{v}_i)$$

$$b_u=\frac{\lambda_{\text{reg}}\sum_{i\in I_u}(r_{ui}-b_i-\mathbf{u}_u^T\mathbf{v}_i)}{\lambda_{\text{reg}}|I_u|+\gamma_{\text{bias}}}$$

### A.2. Deriving the Update Equation for $b_i$

Similarly, differentiate the objective function with respect to $b_i$ and solve:

Differentiate the squared error term:

$$\frac{\partial}{\partial b_i}\left[-\frac{\lambda_{\text{reg}}}{2}\sum_{(u,i)\in K}\left(r_{ui}-(b_u+b_i+\mathbf{u}_u^T\mathbf{v}_i)\right)^2\right]=\lambda_{\text{reg}}\sum_{u\in U_i}\left(r_{ui}-b_u-b_i-\mathbf{u}_u^T\mathbf{v}_i\right)$$

Here, $U_i$ is the set of all users who have rated item $i$.

Differentiate the regularization term:

$$\frac{\partial}{\partial b_i}\left[-\frac{\gamma_{\text{bias}}}{2}b_i^2\right]=-\gamma_{\text{bias}}b_i$$

Set the sum of these derivatives to zero and solve for $b_i$:

$$\lambda_{\text{reg}}\sum_{u\in U_i}\left(r_{ui}-b_u-b_i-\mathbf{u}_u^T\mathbf{v}_i\right)-\gamma_{\text{bias}}b_i=0$$

$$b_i(\lambda_{\text{reg}}|U_i|+\gamma_{\text{bias}})=\lambda_{\text{reg}}\sum_{u\in U_i}(r_{ui}-b_u-\mathbf{u}_u^T\mathbf{v}_i)$$

$$b_i=\frac{\lambda_{\text{reg}}\sum_{u\in U_i}(r_{ui}-b_u-\mathbf{u}_u^T\mathbf{v}_i)}{\lambda_{\text{reg}}|U_i|+\gamma_{\text{bias}}}$$

These update rules for $b_u$ and $b_i$ balance the aim of fitting the model to the data while controlling for overfitting through the regularization terms.

### A.3. Partial Derivative with respect to User Latent Factor $u_u$

To update the user latent vector $u_u$, differentiate the objective function with respect to $u_u$ and set the derivative to zero to find the update rule. The relevant part of the objective function concerning $u_u$ is:

$$\lambda_{\text{reg}} \sum_{i \in I_u} \left( r_{ui} - b_u - b_i - \mathbf{u}_u^T \mathbf{v}_i \right)^2 + \gamma_{\text{latent}} \|\mathbf{u}_u\|^2$$

Taking the derivative with respect to $\mathbf{u}_u$ and setting it to zero gives:

$$\lambda_{\text{reg}} \sum_{i \in I_u} \left( r_{ui} - b_u - b_i - \mathbf{u}_u^T \mathbf{v}_i \right) (-\mathbf{v}_i) + \gamma_{\text{latent}} \mathbf{u}_u = 0$$

Rearranging and solving for $\mathbf{u}_u$:

$$\mathbf{u}_u = \left( \lambda_{\text{reg}} \sum_{i \in I_u} \mathbf{v}_i \mathbf{v}_i^T + \gamma_{\text{latent}} I \right)^{-1} \lambda_{\text{reg}} \sum_{i \in I_u} (r_{ui} - b_u - b_i) \mathbf{v}_i$$

### A.4. Partial Derivative with respect to Item Latent Factor $\mathbf{v}_i$

Similarly, to update the item latent vector $\mathbf{v}_i$, differentiate the objective function with respect to $\mathbf{v}_i$:

$$\lambda_{\text{reg}} \sum_{u \in U_i} \left( r_{ui} - b_u - b_i - \mathbf{u}_u^T \mathbf{v}_i \right)^2 + \gamma_{\text{latent}} \|\mathbf{v}_i\|^2$$

Taking the derivative with respect to $\mathbf{v}_i$ and setting it to zero gives:

$$\lambda_{\text{reg}} \sum_{u \in U_i} \left( r_{ui} - b_u - b_i - \mathbf{u}_u^T \mathbf{v}_i \right) (-\mathbf{u}_u) + \gamma_{\text{latent}} \mathbf{v}_i = 0$$

Rearranging and solving for $\mathbf{v}_i$:

$$\mathbf{v}_i = \left( \lambda_{\text{reg}} \sum_{u \in U_i} \mathbf{u}_u \mathbf{u}_u^T + \gamma_{\text{latent}} I \right)^{-1} \lambda_{\text{reg}} \sum_{u \in U_i} (r_{ui} - b_u - b_i) \mathbf{u}_u$$