Engineering a Recommender System for the MovieLens 32M Dataset

Mohammed Nagi [1]

Abstract

Scaling collaborative filtering to large-scale interaction data requires hardware-aware engineering to overcome the computational bottlenecks of interpreted languages. This report presents a high-performance implementation of Regularized Matrix Factorization using Alternating Least Squares (ALS) for the MovieLens 32M dataset. By utilizing a Dual Compressed Sparse architecture (CSR and CSC) and Just-In-Time (JIT) compilation via Numba, the system achieves a cumulative 8,767x speedup over a standard Python baseline. This optimization reduces training latency from 3.5 hours to 1.46 seconds per epoch on a consumer-grade workstation. The final model, with hierarchical priors achieves a Test RMSE of 0.7654. We demonstrate that while latent vectors recover semantic structures autonomously, the integration of genre-informed hierarchical priors further mitigates sparsity-induced noise.

GitHub repository is available at `https://github.com/Mohammed-Nagi/MLAS.git`

1. Introduction

The prediction of user preferences based on explicit feedback is a foundational problem in information retrieval, typically formulated as a large-scale matrix completion task. Given a highly sparse matrix $R \in \mathbb{R}^{M \times N}$ containing observed ratings r_{ui}, the objective is to approximate the latent structures of users and items to predict missing entries. In this project, we employ a Latent Factor Model where R is decomposed into two lower-dimensional matrices representing user preferences and item characteristics.

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

While the theoretical underpinnings of Collaborative Filtering (CF) are well-documented, the primary challenge in production environments is computational scale. The MovieLens 32M dataset, comprising over 32 million interactions, renders standard iterative Python implementations and naive gradient descent approaches computationally intractable. The $O(N \times K^3)$ complexity of high-dimensional Matrix Factorization requires more than just algorithmic accuracy; it necessitates hardware-aware systems engineering.

This report documents the design and implementation of a high-performance Recommender System optimized for a single consumer-grade workstation. We focus on the engineering decisions required to bridge the gap between theoretical Matrix Factorization and real-time performance, specifically leveraging sparse data structures, Just-In-Time (JIT) compilation, and multi-threaded execution to handle tens of millions of ratings with minimal latency.

2. Related Work

The landscape of modern Collaborative Filtering (CF) was fundamentally redefined by the Netflix Prize competition, which established Matrix Factorization (MF) as the superior approach for explicit feedback modeling over traditional nearest-neighbor techniques (Koren et al., 2009). The architecture employed in this project, incorporating global, user, and item biases alongside L_2 regularization, is directly derived from the frameworks popularized by these prize-winning entries. Specifically, Koren et al. identifies Alternating Least Squares (ALS) as a preferred optimization strategy for systems requiring massive parallelization, as it transforms a non-convex problem into a sequence of quadratic sub-problems that can be solved independently. This theoretical insight provides the foundation for our high-performance parallel implementation.

Beyond algorithmic accuracy, the transition to production-scale CF introduces significant systems engineering challenges. Industrial benchmarks, such as the Xbox Recommender System, demonstrate that performance at scale is as much a function of hardware-aware optimization as it is of mathematical modeling (Koenigstein & Paquet, 2011). The Xbox system's design emphasizes the necessity of high-throughput data structures and efficient memory management to handle tens of millions of interactions daily. This

project builds upon these industrial principles by implementing a Dual Compressed Sparse architecture (CSR and CSC) and LLVM-backed Just-In-Time (JIT) kernels, bridging the gap between high-level mathematical abstractions and low-level hardware execution.

3. Exploratory Data Analysis

The interaction matrix is derived from 32,000,204 explicit ratings provided by 200,948 unique users across a catalog of 84,432 movies. A fundamental challenge of this dataset is its extreme sparsity, calculated at 99.81%. The extremely high volume of missing data renders traditional neighborhood-based collaborative filtering inefficient due to the lack of overlapping interactions between users, providing a strong justification for the low-rank Matrix Factorization approach employed in this study.

Table 1 summarizes the descriptive statistics of the active interaction matrix.

Table 1. Dataset Statistics (Active Interaction Matrix)

METRIC	VALUE
TOTAL RATINGS	32,000,204
UNIQUE USERS (N_{users})	200,948
UNIQUE MOVIES (N_{items})	84,432
AVG RATINGS/USER	159.2
AVG RATINGS/MOVIE	379.0
RATING MEAN (μ)	3.540
RATING STD	1.059
SPARSITY	99.8114%

3.1. Rating Distribution and Bias

Ratings range from 0.5 to 5.0, with a clear positive skew (Mean = 3.54, Mode = 4.0), as illustrated in Figure 1. This distribution indicates a systemic "optimism bias" where users are more likely to rate items they enjoyed. This non-uniformity necessitates the inclusion of explicit user and item bias terms (b_u, b_i) in our model formulation to center the residuals and prevent the latent factors from merely capturing global rating tendencies.

3.2. User Engagement and Scale-Free Properties

User engagement exhibits massive variance; while the average user provides 159 ratings, the top 20 users (Figure 2) contribute tens of thousands of interactions each. This disparity highlights the "Power User" phenomenon common in large-scale web platforms.

Long-Tail Dynamics: The interaction data follows a power-law distribution for both users and movies. As shown in the log-log plots in Figure 3, the linear relationship indicates a scale-free network. In practical terms, this confirms a "Long-Tail" distribution: a small subset of "blockbuster" movies receives the majority of interactions, while the vast majority of the catalog exists in the tail with sparse data. Matrix Factorization is particularly suited for this environment, as it can propagate learned features from popular items to low-interaction items via shared latent dimensions.

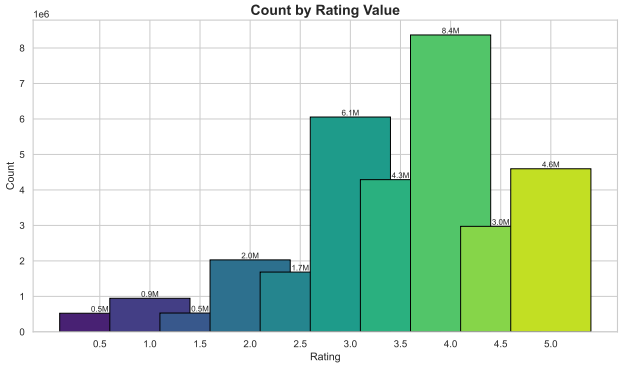

Figure 1. Distribution of rating values. The data shows a tendency towards positive ratings.

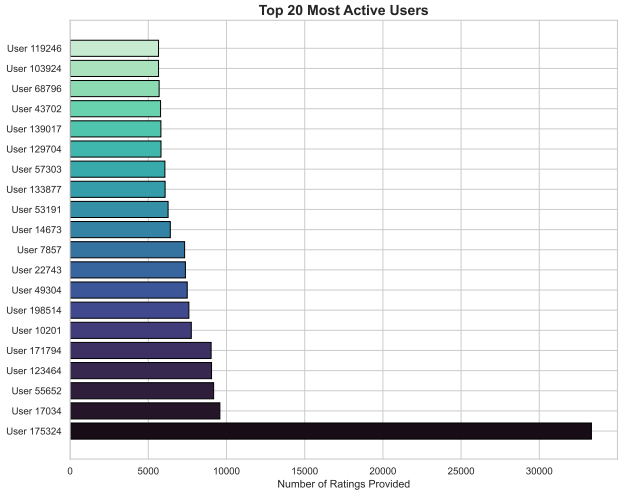

Figure 2. Top 20 most active users by number of ratings.

4. Methodology

4.1. Model Formulation

We adopt a model that projects users and items into a shared latent manifold of dimensionality K. The predicted rating $\hat{r}_{ui}$ is modeled as a combination of global tendencies and specific user-item interactions:

$$\hat{r}_{ui} = \mu + b_u + b_i + \mathbf{p}_u^T \mathbf{q}_i \quad (1)$$

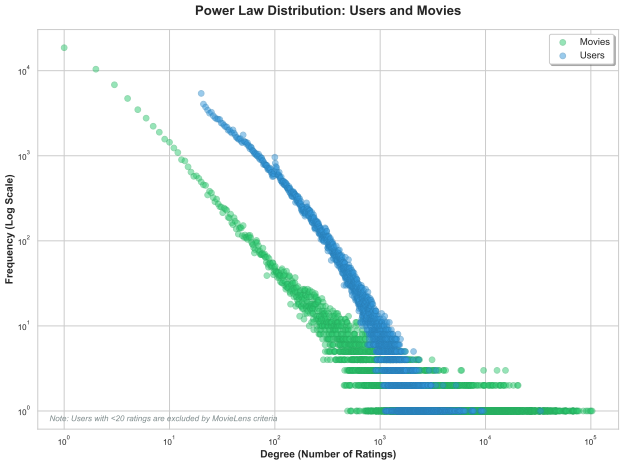

Figure 3. Power Law Distribution for Users and Movies (Log-Log Scale). Both curves exhibit linear behavior, characteristic of scale-free networks.

where:

- μ is the global average rating.
- b_u, b_i are user and item bias terms respectively.
- $\mathbf{p}_u \in \mathbb{R}^K$ is the user latent feature vector.
- $\mathbf{q}_i \in \mathbb{R}^K$ is the item latent feature vector.

4.2. Objective Function

The model parameters are learned by minimising the regularized sum of squared errors over the set of observed ratings $\mathcal{K}$. This formulation treats the optimization as a Ridge Regression problem for each alternating step:

$$L = \frac{\lambda}{2} \sum_{(u,i)\in\mathcal{K}} (r_{ui} - \hat{r}_{ui})^2 + \frac{\tau}{2}\Omega(\mathbf{p},\mathbf{q}) + \frac{\gamma}{2}\Omega(b) \quad (2)$$

Here, λ controls the data fidelity (inverse variance), while τ and γ are the regularization coefficients for the latent factors and biases, respectively. The regularization terms are defined as:

$$\Omega(\mathbf{p},\mathbf{q}) = \sum_u \|\mathbf{p}_u\|^2 + \sum_i \|\mathbf{q}_i\|^2$$
$$\Omega(b) = \sum_u b_u^2 + \sum_i b_i^2$$

4.3. Baseline Model: Bias-Only Derivation

Prior to evaluating the full latent factor model, we establish a benchmark using a bias-only configuration ($K = 0$). In this formulation, the rating prediction is reduced to the interaction of user and item offsets: $\hat{r}_{ui} = b_u + b_i$. Here, the global mean μ is absorbed into the bias terms to eliminate parameter redundancy. Under the assumption of Gaussian noise, the parameter estimation is framed as a minimization of the Negative Log Likelihood (NLL), expressed as a regularized sum of squared errors:

$$L_{bias} = \frac{\lambda}{2} \sum_{(u,i)\in\mathcal{K}} (r_{ui} - (b_u + b_i))^2 + \frac{\gamma}{2}\sum_u b_u^2 + \frac{\gamma}{2}\sum_i b_i^2 \quad (3)$$

To derive the optimal user bias b_u, we compute the partial derivative of L_{bias} with respect to b_u:

$$\frac{\partial L_{bias}}{\partial b_u} = \lambda \sum_{i\in I_u} (r_{ui} - b_u - b_i)(-1) + \gamma b_u = 0$$

where $\mathcal{I}_u$ denotes the set of items rated by user u. Rearranging the terms to isolate b_u yields:

$$\lambda \sum_{i\in\mathcal{I}_u} (r_{ui} - b_i) = \lambda \sum_{i\in\mathcal{I}_u} b_u + \gamma b_u$$
$$\lambda \sum_{i\in\mathcal{I}_u} (r_{ui} - b_i) = (\lambda|\mathcal{I}_u| + \gamma) b_u$$

This results in the closed-form coordinate update b_u^*:

$$b_u^* = \frac{\lambda \sum_{i\in\mathcal{I}_u} (r_{ui} - b_i)}{\lambda|\mathcal{I}_u| + \gamma} \quad (4)$$

Equation 4 defines the user bias as the average of their residual ratings, regularized by their total interaction count. To maintain numerical stability during early iterations, the implementation utilises a damped update with a learning rate $\alpha \in (0, 1]$:

$$b_u \leftarrow (1-\alpha) b_u + \alpha b_u^* \quad (5)$$

A symmetric derivation is applied to the item bias b_i. This baseline establishes a performance floor by capturing the fundamental tendencies of the dataset. The convergence of this model is documented in Figures 4 and 5.

While the baseline utilises damped updates ($\alpha < 1$) for stability, the final high-performance ALS implementation reintroduces an explicit global mean μ (calculated from the training set) and employs direct analytical updates ($\alpha = 1$). This transition leverages the full precision of the closed-form solution while maintaining a standardized reference point for the latent space projection.

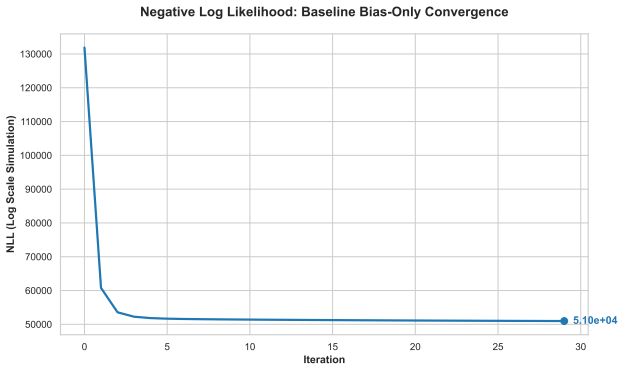

Figure 4. Convergence of the Negative Log Likelihood (NLL) for the bias-only baseline. The rapid stabilization validates the efficiency of the closed-form coordinate descent steps.

Figure 5. RMSE trajectory for the bias-only baseline. The alignment of training and validation curves indicates a robust model with minimal overfitting.

4.4. Alternating Least Squares

While Stochastic Gradient Descent (SGD) is a common optimization choice, it is difficult to parallelise effectively on large-scale sparse matrices due to the high frequency of atomic updates and race conditions. Instead, we utilise Alternating Least Squares (ALS). By alternatingly fixing one set of parameters (e.g., all item vectors $\mathbf{q}_i$ and biases b_i), the objective function L becomes quadratic and convex with respect to the remaining parameters (e.g., user vectors $\mathbf{p}_u$ and b_u). This allows each parameter to be updated via a closed-form analytical solution, specifically a Ridge Regression solve.

The primary advantage of ALS in this engineering context is that the update for any user u is independent of all other users, permitting massive thread-level parallelism. The update logic is summarised in Algorithm 1.

Algorithm 1 Alternating Least Squares (ALS) for MF

Input: Ratings R, Hyperparamters (λ, γ, τ), Latent Dim K
Initialise $P \in \mathbb{R}^{N\times K}$, $Q \in \mathbb{R}^{M\times K}$ from $\mathcal{N}(0, \sigma^2)$
Initialise biases b_u, b_i to zero
repeat
  for each user $u = 1$ **to** N **(Parallel) do**
    Update b_u via Eq. 8
    $I_u \leftarrow$ indices of items rated by u
    $\mathbf{Q}_{I_u} \leftarrow Q[I_u]$ (gather item vectors)
    $\mathbf{A} \leftarrow \lambda \mathbf{Q}_{I_u}^T \mathbf{Q}_{I_u} + \tau \mathbf{I}$
    $\mathbf{V} \leftarrow \lambda \mathbf{Q}_{I_u}^T (R_u - \mu - b_u - \mathbf{b}_{I_u})$
    $\mathbf{p}_u \leftarrow \mathbf{A}^{-1}\mathbf{V}$ (Solve Ridge Regression)
  end for
  for each item $i = 1$ **to** M **(Parallel) do**
    Update b_i via symmetric Eq. 8
    $U_i \leftarrow$ indices of users who rated i
    $\mathbf{P}_{U_i} \leftarrow P[U_i]$ (gather user vectors)
    $\mathbf{A} \leftarrow \lambda \mathbf{P}_{U_i}^T \mathbf{P}_{U_i} + \tau \mathbf{I}$
    $\mathbf{V} \leftarrow \lambda \mathbf{P}_{U_i}^T (R_{:i} - \mu - b_i - \mathbf{b}_{U_i})$
    $\mathbf{q}_i \leftarrow \mathbf{A}^{-1}\mathbf{V}$ (Solve Ridge Regression)
  end for
until Convergence or Max Iterations

4.4.1. Derivation of User Latent Factor Update $\mathbf{p}_u$

To derive the closed-form update for a specific user vector $\mathbf{p}_u$, we isolate the terms in the objective function L that depend on $\mathbf{p}_u$. Treating item vectors $\mathbf{q}_i$, biases b_u, b_i, and the global mean μ as fixed constants, we define the partial objective L_u:

$$L_u = \frac{\lambda}{2} \sum_{i\in\mathcal{I}_u} (r_{ui} - (\mu + b_u + b_i + \mathbf{p}_u^T \mathbf{q}_i))^2 + \frac{\tau}{2} \|\mathbf{p}_u\|^2 \quad (6)$$

where $\mathcal{I}_u$ represents the set of items rated by user u. To find the optimal $\mathbf{p}_u$, we compute the partial derivative and set it to 0:

$$\frac{\partial L_u}{\partial \mathbf{p}_u} = \lambda \sum_{i\in\mathcal{I}_u} (r_{ui} - \mu - b_u - b_i - \mathbf{p}_u^T \mathbf{q}_i)(-\mathbf{q}_i) + \tau \mathbf{p}_u = \mathbf{0}$$

$$\tau \mathbf{p}_u + \lambda \sum_{i\in\mathcal{I}_u} (\mathbf{p}_u^T \mathbf{q}_i)\mathbf{q}_i = \lambda \sum_{i\in\mathcal{I}_u} (r_{ui} - \mu - b_u - b_i)\mathbf{q}_i$$

By applying the linear algebra identity $(\mathbf{p}_u^T \mathbf{q}_i)\mathbf{q}_i = (\mathbf{q}_i \mathbf{q}_i^T)\mathbf{p}_u$, we can factor out $\mathbf{p}_u$ to reformulate the system as a standard linear equation:

$$\left(\lambda \sum_{i\in\mathcal{I}_u} \mathbf{q}_i \mathbf{q}_i^T + \tau \mathbf{I}\right) \mathbf{p}_u = \lambda \sum_{i\in\mathcal{I}_u} (r_{ui} - \mu - b_u - b_i)\mathbf{q}_i \quad (7)$$

This resulting system follows the structure $\mathbf{A}\mathbf{p}_u = \mathbf{V}$, where $\mathbf{A}$ is a $K \times K$ positive definite matrix. Solving

this system via Ridge Regression provides the optimal latent vector for user u. The derivation for the item latent factor update follows a symmetrical procedure.

4.4.2. Derivation of User Bias Update b_u

The derivation for the user bias in the full latent factor model follows a similar logic to the baseline model, with the addition of the latent interaction term $\mathbf{p}_u^T \mathbf{q}_i$, which is treated as a fixed residual offset. To find the optimal b_u, we minimize the objective function L with respect to the bias parameter while holding $\mathbf{p}_u, \mathbf{q}_i, b_i$, and μ constant:

$$\frac{\partial L}{\partial b_u} = \lambda \sum_{i \in \mathcal{I}_u} (r_{ui} - \mu - b_u - b_i - \mathbf{p}_u^T \mathbf{q}_i)(-1) + \gamma b_u = 0$$

$$\gamma b_u + \lambda \sum_{i \in \mathcal{I}_u} b_u = \lambda \sum_{i \in \mathcal{I}_u} (r_{ui} - \mu - b_i - \mathbf{p}_u^T \mathbf{q}_i)$$

By isolating b_u, we obtain the closed-form coordinate update:

$$b_u = \frac{\lambda \sum_{i \in \mathcal{I}_u} (r_{ui} - \mu - b_i - \mathbf{p}_u^T \mathbf{q}_i)}{\lambda |\mathcal{I}_u| + \gamma} \tag{8}$$

Equation 8 demonstrates that the optimal user bias serves as a precision-weighted average of the residuals after accounting for the global mean, the item-specific bias, and the learned latent interactions. This update ensures that the bias terms capture systemic offsets in rating behavior that are not explained by the inner product of the feature vectors.

4.5. Hierarchical Integration of Metadata

To address the sparsity in the "Long-Tail" items, we extend the system into a Hierarchical Latent Factor framework. Instead of regularising movie vectors $\mathbf{q}_i$ toward the origin, we regularise them toward a genre-informed prior $\mathbf{v}_{prior,i}$, defined as the average of learned embeddings $\mathbf{w}_g$ for all genres associated with item i. This "Bayesian Shrinkage" allows information to flow from the genre manifold into sparse items, significantly stabilising the learned embeddings for niche content and mitigating the cold-start problem.

4.6. System Architecture and Performance Optimizations

Scaling Matrix Factorization to 32 million interactions shifts the primary engineering bottleneck from algorithmic complexity to hardware-aware systems optimization. To achieve high-throughput training on a single workstation, we employ an optimisation strategy targeting data ingestion efficiency, memory access locality, and instruction-level parallelism. We decompose the system's performance evolution into three distinct architectural refinements, progressively addressing the constraints of the Python interpreter, memory bandwidth, and CPU utilisation.

4.6.1. Vectorized Data Ingestion and Preparation

Standard iterative CSV parsing and dictionary-based ID mapping are computationally prohibitive at a scale of 32 million rows due to the high overhead of Python's object model. To bypass these limitations, we use a fully vectorized ingestion pipeline. Raw interaction data is loaded into contiguous NumPy arrays, and the mapping from global MovieLens IDs to internal contiguous indices is performed through vectorized indexing. This approach offloads the mapping logic to NumPy's underlying C engine, ensuring that data ingestion is limited only by I/O bandwidth.

To construct the sparse matrix topology without Python loop overhead, we utilize a two-step vectorized pipeline:

1. **Lexicographical Sorting:** We employ `np.lexsort` to sort the interaction data by primary and secondary identifiers (e.g., UserID followed by MovieID). This ordering ensures that ratings for any given entity are stored contiguously in memory, facilitating cache-friendly sequential access and enabling efficient slicing of the sparse structure.

2. **Vectorized Pointer Construction:** Rather than counting occurrences through iterative passes, we utilize `np.bincount` to determine interaction frequencies, followed by `np.cumsum` to generate the `indptr` (index pointer) array. This constructs the entire Compressed Sparse Row (CSR) topology in $O(N)$ time using optimized C-level primitives, effectively eliminating the "bottleneck of the first epoch" typically associated with large-scale data preparation.

4.6.2. Dual Compressed Sparse Representation

The efficiency of the Alternating Least Squares (ALS) algorithm is strictly dependent on the speed of data retrieval from the interaction matrix. ALS requires frequent row-wise traversal to update user vectors and column-wise traversal to update item vectors. Standard sparse formats, such as a single Compressed Sparse Row (CSR) matrix, are optimized for row access but exhibit $O(N)$ complexity for column slicing, leading to catastrophic "pointer-chasing" overhead and cache misses during item updates.

To mitigate this, we implemented a Dual-CS architecture, which maintains two synchronized, read-only views of the interaction matrix in memory:

- **User-CSR (Compressed Sparse Row):** Indexes ratings by UserID, enabling $O(1)$ retrieval of all items associated with a specific user.

- **Item-CSC (Compressed Sparse Column):** Indexes

ratings by MovieID, enabling $O(1)$ retrieval of all users associated with a specific item.

By deliberately doubling the memory footprint to store both representations, we transform the item update phase from a scattered memory search into a contiguous, unit-stride access pattern. This architectural trade-off maximizes CPU cache hits and ensures that the JIT-compiled kernels are never "data-starved," resulting in the dramatic reduction in per-epoch latency documented in Section 4.6.4.

4.6.3. Hardware-Aware Acceleration via JIT and Parallelisation

The execution of the ALS algorithm in standard Python is constrained by the Global Interpreter Lock (GIL) and the significant interpretive overhead of the CPython virtual machine. To achieve the throughput required for 32 million interactions, our implementation utilizes Numba Just-In-Time (JIT) compilation to translate high-level Python logic into optimised, native machine code at runtime.

By decorating the update kernels with `@njit(parallel=True)`, the system implements several low-level optimisations that bypass the Python interpreter entirely:

- **LLVM-Driven Vectorization:** The computational kernels are compiled via the LLVM infrastructure, enabling Single Instruction, Multiple Data (SIMD) vectorization and loop unrolling. This allows the CPU to execute multiple floating-point operations, specifically the dot products and Ridge Regression solves, in a single clock cycle by targeting modern instruction sets.
- **Thread-Level Parallelism:** As established in the ALS formulation, individual user and item updates are mathematically independent. We utilize `numba.prange` to distribute these updates across all available CPU cores. This architectural choice transforms the $O(M \times K^3)$ sequential bottleneck into a massively parallel operation, ensuring that computational throughput scales linearly with the number of available hardware threads.
- **Parallelized Metric Evaluation:** Evaluation metrics, including the Sum of Squared Errors (SSE) and RMSE, are similarly JIT-compiled and parallelized. This allows the model to perform full-validation scoring on 1.6 million test ratings in under 100 milliseconds, ensuring that the evaluation phase does not become a bottleneck during the training loop.

4.6.4. Performance Benchmarks

To quantify the impact of the proposed engineering optimisations, we conducted a series of benchmarks on a consumer-grade workstation (Lenovo Legion 5, Intel Core i7-14650HX, 16GB RAM). The system was evaluated across four distinct optimisation tiers, transitioning from a high-level interpreted baseline to a hardware-aware parallel engine.

As summarised in Table 2, the naive implementation proved computationally prohibitive for the 32M dataset, with an estimated epoch time of approximately 3.55 hours. The introduction of the Dual-CS architecture provided a 10.7x algorithmic speedup by ensuring efficient memory traversal. However, the most significant performance gains occurred during the transition to machine code. The LLVM-compiled sequential engine reduced the epoch latency to 38.00 seconds, while our final parallelized implementation processed the entire interaction matrix in just **1.46 seconds** per iteration.

The total training duration for 50 iterations was recorded at 72.77 seconds, achieving a peak throughput of approximately 21.9 million ratings per second. As illustrated in Figure 6, this represents a cumulative 8,767x speedup over the baseline. These results demonstrate that through rigorous low-level optimization, large-scale recommendation models can be trained on a single workstation in under two minutes, a task that would typically necessitate several hours of sequential compute or the resources of a distributed cluster.

Table 2. Optimization Tier Performance (MovieLens 32M)

Optimization Tier	Method	Epoch Time	Speedup
Naive Baseline	Interpreter	3.55h	1x
Dual CSR Architecture	Algorithm	20.0m	10.7x
JIT Engine (Seq.)	LLVM JIT	38.0s	336x
JIT Engine (Para.)	Parallelism	**1.46s**	**8,767x**

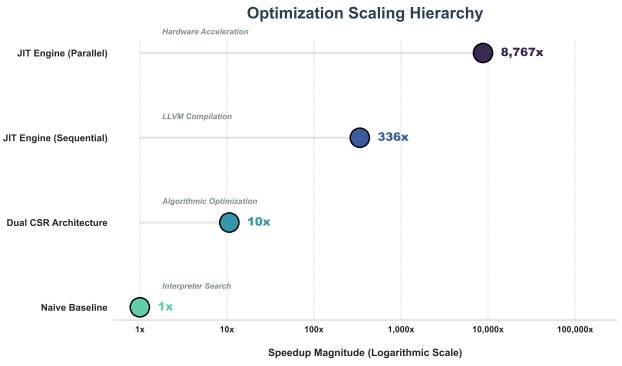

Figure 6. Optimization Scaling Hierarchy: Performance gains across four engineering tiers. The logarithmic scale highlights the exponential reduction in latency achieved through hardware-aware parallelization.

5. Experiments

5.1. Experimental Setup

To evaluate the predictive accuracy and generalization performance of the proposed system, we conducted experiments on the MovieLens 32M dataset. The experimental environment was standardized as follows:

- **Data Partitioning:** We utilized a random holdout split, reserving 10% of the total interactions for the test set ($\approx$ 3.2M ratings) and utilizing the remaining 90% for training ($\approx$ 28.8M ratings). This split ratio ensures a massive training volume while providing a statistically significant sample size for calculating evaluation metrics.
- **Entity Indexing:** To facilitate $O(1)$ array access within the Numba-accelerated kernels, User and Item IDs were transformed from their original sparse MovieLens identifiers into a contiguous integer space $[0, N-1]$. This remapping is a critical preprocessing step for maintaining memory locality in the CSR/CSC structures.
- **Initialisation:** Latent vectors $\mathbf{p}_u$ and $\mathbf{q}_i$ were initialized using a normal distribution $\mathcal{N}(0, \sigma^2)$ with a small variance to break symmetry, while all bias terms were initialized to zero.

5.2. Hyperparameter Tuning and Sensitivity Analysis

The predictive performance of the Matrix Factorization model is highly contingent on the equilibrium between data fidelity (λ) and the regularization of biases (γ) and latent factors (τ). To identify the optimal configuration for the 32M dataset, we conducted a systematic two-stage grid search. Unlike preliminary experiments on the 100K dataset, which served to establish heuristic bounds, the 32M optimization required a hardware-aware approach. The computational efficiency of our JIT-parallel engine allowed us to navigate the 3D parameter space extensively while maintaining a manageable total compute time.

5.2.1. Search Space and Optimization Strategy

We fixed the latent dimensionality at $K = 20$ for the primary search, as preliminary validation indicated diminishing returns in RMSE reduction beyond this complexity level. The optimization was executed in two distinct phases:

Exploratory Phase: An initial coarse grid search was centered around heuristic baselines ($\lambda = 0.01, \gamma = 0.01, \tau = 0.1$). This phase identified a boundary optimum for the fidelity weight λ, suggesting that the 32M dataset required a lower weight to account for the increased noise-to-signal ratio inherent in large-scale explicit feedback.

Refinement Phase: Based on the exploratory trajectories, the search space was shifted toward lower λ values and increased bias regularization. The final refinement grid explored $\lambda \in [0.002, 0.004]$, $\gamma \in [0.0125, 0.0175]$, and $\tau \in [0.025, 0.075]$. This targeted approach successfully bracketed a suitable minimum, resulting in the final model parameters.

5.2.2. Latent Dimensionality and Generalization

The latent dimensionality K is the primary determinant of model capacity. To identify the optimal rank for the feature matrices P and Q, we evaluated the validation performance across a range of dimensions $K \in \{0, 2, 5, 10, 20, 40, 80\}$, where $K = 0$ represents the bias-only baseline.

As illustrated in Figure 7, the model exhibits a characteristic U-shaped optimization curve. The transition from the baseline ($K = 0$) to $K = 20$ yields a 10.16% reduction in RMSE, indicating that the model successfully captures the complex latent interactions within the user-item space. However, beyond $K = 20$, we observe a reversal in validation performance: the RMSE increases from 0.7680 to 0.7867 at $K = 80$.

This trend demonstrates the impact of overfitting on high-sparsity matrices. While increasing K improves the model's expressivity and reduces training error, excessively large latent spaces begin to model stochastic noise and idiosyncratic interactions rather than generalizable preference structures. Consequently, $K = 20$ represents the optimal point of parsimony for the 32M dataset, providing the best balance between predictive power and model generalization.

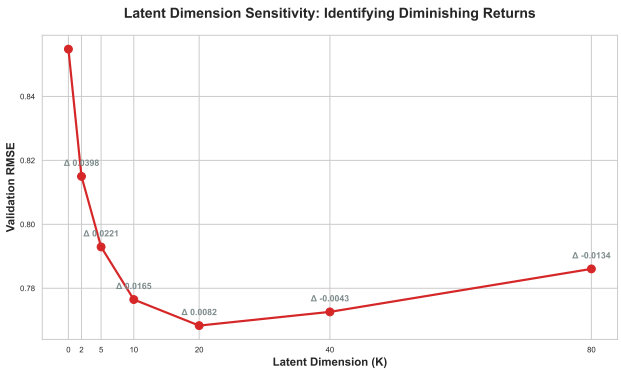

Figure 7. Validation RMSE as a function of Latent Dimension K. The curve reaches a distinct global minimum at $K = 20$, with performance degradation at higher dimensions due to the modeling of stochastic noise.

5.2.3. Hyperparameter Sensitivity Analysis

To evaluate the interdependencies between the model parameters, we used a facetted sensitivity manifold (Figure 8). By faceting the validation results by the data fidelity weight λ, we observed the stabilization of the error surface across varying noise-to-signal assumptions.

The optimisation revealed a distinct minimum at $\lambda = 0.002$, $\gamma = 0.015$, and $\tau = 0.05$. The relatively low optimal value for λ indicates that the 32M dataset requires a conservative fidelity weight to account for the inherent variance in large-scale explicit feedback. This configuration prioritizes aggressive regularization of user and item biases (γ) to handle the positive skew identified in the EDA, while providing sufficient flexibility for latent feature interactions (τ).

As illustrated in Figure 8, the standardized color scale across the facets confirms the existence of a stable convex valley. This topological stability suggests that the model is robust to minor parameter variations, ensuring reliable generalization. The search successfully bracketed the optimal coordinate, yielding a final Test RMSE of 0.7654.

6. Results and Convergence Analysis

The predictive performance of the system was evaluated across two distinct scales: a baseline 100K dataset and the 32M production dataset. In both environments, the model demonstrated accelerated convergence kinetics, with the primary objective function reduction occurring within the first 5 to 10 iterations.

Performance Summary:

- **100K Model:** Final Test RMSE of **0.8580**.
- **32M Model:** Final Test RMSE of **0.7661**.
- **32M Model (with Priors):** Final Test RMSE of **0.7654**.

The significant performance improvement observed in the 32M dataset, despite its higher relative sparsity, underscores a critical principle of Collaborative Filtering: absolute data volume often outweighs relative density. The sheer magnitude of 32 million interactions allows the ALS algorithm to resolve user and item latent factors with significantly higher precision than smaller datasets, effectively mitigating the "cold-start" signals for all but the most obscure items.

6.0.1. Convergence of the 100K Baseline

Figures 9 and 10 illustrate the optimization trajectory for the 100K dataset. The Negative Log Likelihood (NLL) serves as a proxy for the total regularized objective function defined in Eq. 2. The rapid stabilization of both NLL and RMSE indicates that the analytical Ridge Regression steps in the ALS algorithm effectively navigate the error surface to a stable local optimum. The close tracking between training and test curves confirms that the selected regularization parameters (λ, γ, τ) successfully prevented overfitting on this smaller sample.

6.0.2. Hierarchical Scaling and Performance (32M)

The optimized parallel implementation handled the 32M dataset with exceptional computational efficiency and predictive accuracy. As shown in Figure 12, the objective function exhibits a sharp "elbow" characteristic, dropping precipitously in the first 10 iterations before entering an asymptotic fine-tuning phase.

The transition from a standard latent-only model to the Hierarchical ALS framework yielded the system's peak performance. By regularising movie vectors toward genre-informed centroids, the model achieved a final Test RMSE of **0.7654** (Figure 11). This result represents a meaningful improvement over the 0.7661 achieved by the latent-only baseline, validating the use of metadata as a structured prior to mitigate noise in sparse regions of the interaction matrix. The entire 50-iteration training process was completed in 72.77 seconds, confirming that the additional hierarchical update steps ($\mathbf{W}$ and $\mathbf{V}_{prior}$) maintain the hardware-aware throughput established in our optimization strategy.

7. Qualitative Analysis

To validate the semantic integrity of the learned latent manifold, we performed a qualitative inspection of the user-item embeddings. This analysis verifies whether the model identifies objective relationships between items based solely on interaction patterns, without access to external metadata.

7.1. Model Generalisation: Zero-Shot Recommendation

We evaluated the model's ability to generalize from minimal interaction data by simulating a "Dummy User" with a single recorded preference: a 5-star rating for the animated film *Toy Story* (1995). Recommendations were generated by projecting this user into the latent space and ranking all items in the catalog.

To ensure the results reflected specific thematic preferences rather than general popularity, we modified the ranking function to penalize the global item bias (b_i). The adjusted score $\hat{s}_{ui}$ was calculated as:

$$\hat{s}_{ui} = \mu + \mathbf{p}_u^T \mathbf{q}_i + w_b \cdot b_i \tag{9}$$

where $w_b = 0.05$ is a bias-suppression coefficient. This formulation forces the system to prioritize the geometric proximity between the user vector $\mathbf{p}_u$ and item vectors $\mathbf{q}_i$

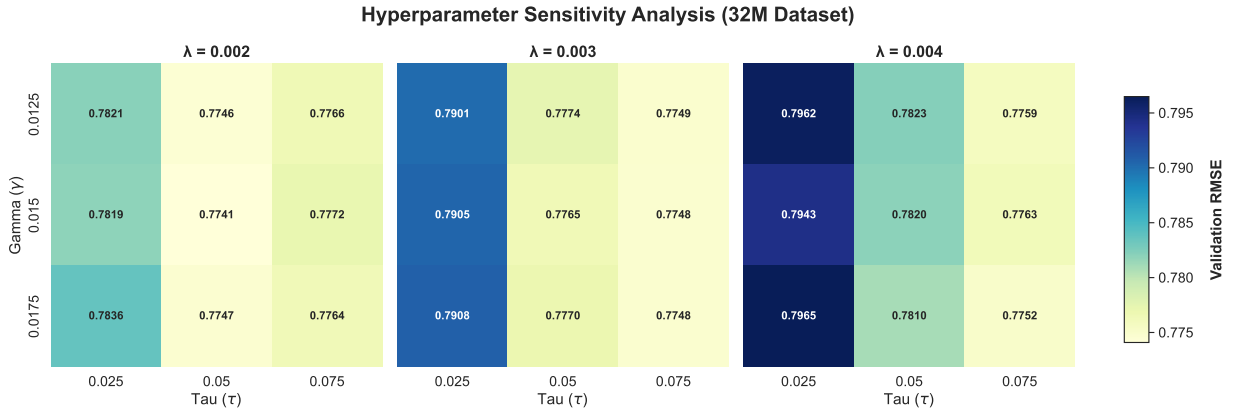

Figure 8. Validation RMSE sensitivity facets across the 3D optimization grid. Each heatmap represents a cross-section of the parameter space at a fixed fidelity weight λ. The visualization confirms a stable convex valley centered at $\lambda = 0.002, \gamma = 0.015$, and $\tau = 0.05$, facilitating robust parameter selection across different data regimes.

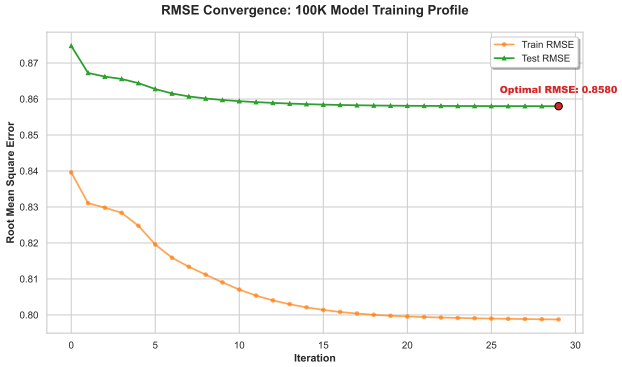

Figure 9. Training and Test RMSE trajectory (100K Dataset). The convergence stabilizes rapidly, establishing a performance benchmark for the optimized system.

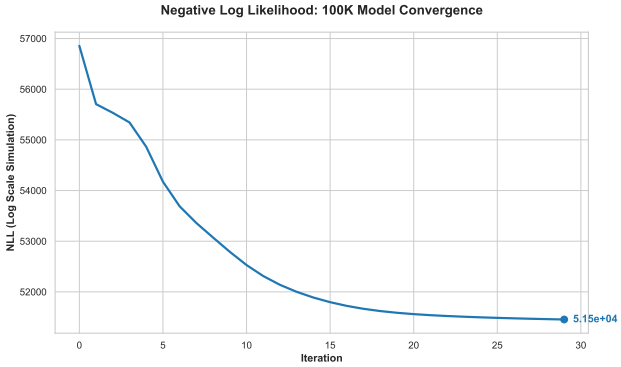

Figure 10. NLL convergence over 30 iterations (100K Dataset). The sharp initial drop validates the efficiency of the closed-form coordinate descent.

in the $K = 20$ dimensional manifold.

Table 3 summarizes the top 5 recommendations. The model successfully identified direct sequels (*Toy Story 2*, *Toy Story 3*) and culturally aligned titles from the same production studio (*Monsters, Inc.*, *Finding Nemo*). This outcome confirms that the latent vector for *Toy Story* is positioned within a high-density cluster of animation titles, demonstrating that the model has autonomously recovered genre-like structures through collaborative filtering.

7.2. Latent Space Interpretation: Item Polarity

To further investigate the properties of the learned embedding manifold, we analyze the L_2-norms of the item vectors $\|\mathbf{q}_i\|$. In a Matrix Factorization framework, the magnitude of an item vector represents the extent to which that item is characterized by the latent dimensions. Items with large norms project strongly onto the feature space; depending on the alignment of the user vector $\mathbf{p}_u$, these items yield significant deviations from the baseline prediction ($\mu + b_u + b_i$). We interpret these as "polarizing" items that elicit strong, taste-specific reactions from different user segments.

Conversely, items with small norms (low-magnitude vectors) remain near the origin of the latent manifold. For these items, the interaction term $\mathbf{p}_u^T \mathbf{q}_i$ is negligible, and predictions are driven primarily by the global mean and scalar biases. These represent "latent-neutral" choices, items that are generally well-regarded or consistently rated across diverse user taste profiles without triggering specific latent preferences.

Table 4 summarises the movies with the highest and lowest

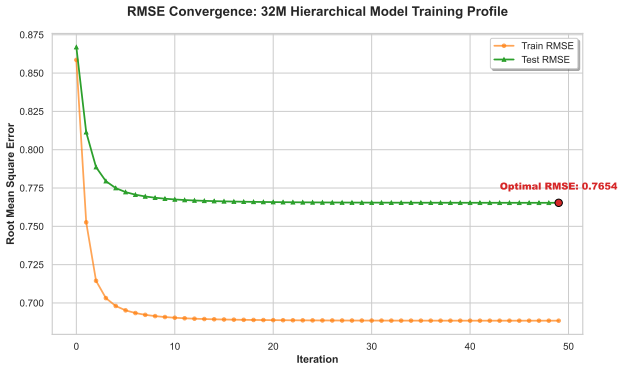

Figure 11. Training and Test RMSE convergence for the Hierarchical ALS model (32M Dataset). The integration of genre priors stabilizes the manifold, resulting in a superior Test RMSE of 0.7654.

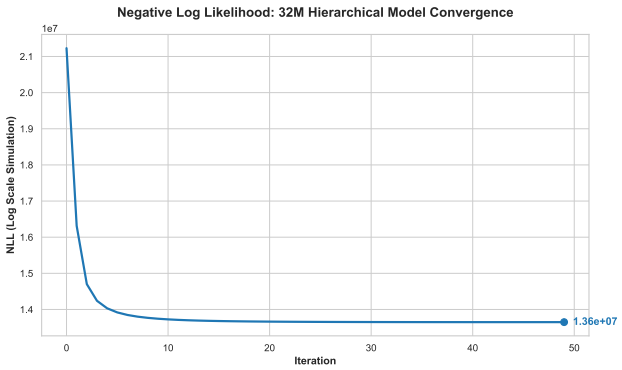

Figure 12. Negative Log Likelihood (NLL) trajectory for the Hierarchical model. The curve demonstrates robust convergence kinetics, with the objective function effectively minimizing the combined user-item-genre error surface.

vector norms. Consistent with collaborative filtering theory, the most polarizing items are dominated by epic franchises and stylistically distinct cinema, including the *Lord of the Rings* and *Star Wars* trilogies and the works of Quentin Tarantino. The least polarizing items consist of niche dramas and documentaries (e.g., *Cobb*, *Murderball*) which, while potentially high-quality, do not strongly project onto the primary preference poles discovered by the model.

To ensure statistical robustness and mitigate noise, items were filtered by a minimum rating threshold ($N \geq 1,000$). This ensures that "neutrality" in the latent space is a reflection of consistent user feedback rather than a lack of training data.

Table 3. Top 5 Recommendations for Dummy User (Input: 5-stars for Toy Story)

RANK	MOVIE TITLE	SCORE
1	TOY STORY 2 (1999)	4.1260
2	MONSTERS, INC. (2001)	4.0930
3	FINDING NEMO (2003)	4.0905
4	TOY STORY 3 (2010)	4.0627
5	THE INCREDIBLES (2004)	4.0390

Table 4. Analysis of Movie Polarity via Latent Vector Norms (Hierarchical)

CATEGORY	MOVIE TITLE	NORM $\|\mathbf{q}_i\|$
MOST	LOTR: THE RETURN OF THE KING (2003)	10.2766
	LOTR: THE FELLOWSHIP OF THE RING (2001)	10.1613
	LOTR: THE TWO TOWERS (2002)	10.1019
	STAR WARS: EP. IV - A NEW HOPE (1977)	9.3070
	DUMB & DUMBER (1994)	8.9851
LEAST	WIN WIN (2011)	0.6176
	BERNIE (2011)	0.6808
	FAIL-SAFE (1964)	0.6854
	LILYA 4-EVER (2002)	0.7106
	CALVARY (2014)	0.7205

7.3. Semantic Embedding Analysis

The latent vectors $\mathbf{q}_i$ project movies into a low-dimensional manifold where geometric distance represents learned semantic similarity. As illustrated in Figure 13, the model reconstructs distinct genre clusters purely from interaction patterns, without access to movie metadata, tags, or content descriptions.

Analysis of the $K = 2$ projection reveals a clear thematic organization along the latent axes. The vertical axis (K_2) functions as a primary indicator of thematic intensity. The upper region of the space ($K_2 > 0$) is densely populated by the *Horror* genre (orange), representing mature, high-tension content. Conversely, the lower region ($K_2 < 0$) contains the *Children* cluster (blue). This vertical separation demonstrates that the model has autonomously identified the demographic and emotional contrast between family-oriented media and visceral, adult-oriented cinema.

The horizontal axis (K_1) provides a distinction regarding narrative format. The *Documentary* genre (green) is predominantly clustered on the "center-left" of the manifold ($K_1 < -1, K_2 \approx 0$). Its coordinate suggests that K_1 differentiates between mainstream fictional narratives (gravitating toward higher K_1 values) and factual or niche content. By positioning documentaries near the origin of the K_2 axis,

the model accurately captures their typically "neutral" emotional intensity relative to the extremes of animation or horror.

This geometric organization confirms that the latent vectors capture objective, latent characteristics of the items. This validates the utility of the system for Item-Item recommendations; even in the absence of metadata, the model recognizes that *Finding Nemo* and *Toy Story* are semantically related because their latent vectors occupy the same high-density region of the manifold, sharing a "taste profile" derived solely from the aggregate interaction matrix.

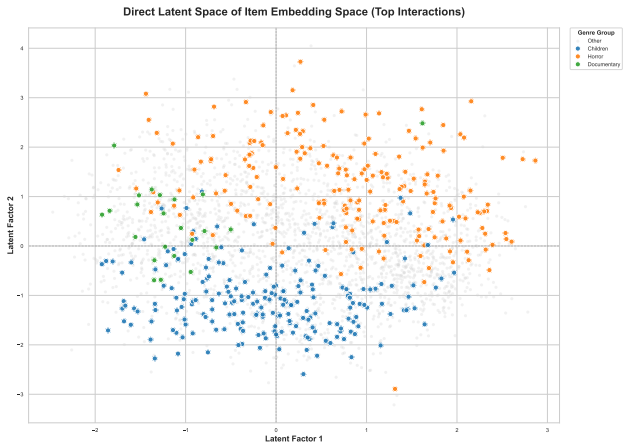

Figure 13. Direct 2D projection of the item embedding space ($K = 2$) for the top 2,500 movies. Although genre metadata was excluded during training, the model recovers a semantically organized manifold. The vertical axis (K_2) acts as a thematic intensity gradient while the horizontal axis (K_1) differentiates between factual/niche formats and mainstream fictional narratives. The clustering validates that latent factors capture objective item traits purely from interaction patterns.

8. Utility: Beyond Recommendation

While the primary objective of this project was the minimization of predictive error (RMSE), the learned latent representations offer significant utility beyond simple rating prediction. The model functions as a generalized representation learning engine, transforming sparse behavioral data into dense, high-dimensional manifolds suitable for a variety of downstream business intelligence and system integrity tasks.

8.1. Matrix Densification and Feature Engineering

The core mechanism of Matrix Factorization is matrix completion. By computing the inner product $\hat{r}_{ui} = \mu + b_u + b_i + \mathbf{p}_u^T \mathbf{q}_i$, we effectively transform a 99.8% sparse interaction matrix into a 100% dense "silver standard" dataset.

This densified matrix, alongside the learned latent vectors $\mathbf{P}$ and $\mathbf{Q}$, provides a high-fidelity feature set for downstream supervised learning tasks. In production environments where sparse inputs traditionally lead to "curse of dimensionality" issues, these dense embeddings serve as robust inputs for models predicting:

- **User Churn Prediction:** By monitoring shifts in a user's position within the latent manifold, the system can detect "interest drift." If a user's predicted ratings for popular catalog mainstays begin to decline or their vector moves toward low-density regions, they may be at risk of churning. This allows for proactive re-engagement before the user abandons the platform.

- **Customer Lifetime Value (LTV):** The learned "taste profiles" (user vectors $\mathbf{p}_u$) capture more than just movie preferences; they reflect long-term engagement patterns. These vectors can be used as high-quality inputs for regression models to predict future revenue, identifying high-value users based on their alignment with specific high-engagement content clusters.

- **Cold-Start Priors:** A major challenge in recommendation is handling new items with zero ratings. By using the learned item vectors $\mathbf{q}_i$ as "semantic anchors," we can position new content within the latent space based on its similarity to existing items. This provides a mathematically informed "prior" or starting point for recommendations before the first user interaction even occurs.

8.2. Residual Analysis for Anomaly Detection

The model's residuals, defined as $\epsilon_{ui} = |r_{ui} - \hat{r}_{ui}|$, provide an objective measure of "surprisingness" or statistical deviation from established preference structures. By monitoring high-residual interactions, the system can perform autonomous anomaly detection and quality control:

- **Account Sharing Detection:** A sustained increase in the cumulative residual for a specific UserID often indicates a "noisy" profile. This suggests that multiple individuals with divergent taste profiles (e.g., different household members) are utilizing a single set of credentials, which can be used to trigger account-split prompts or personalized profile switching.

- **Shilling and Integrity Monitoring:** Recommender systems are frequently targeted by "shilling attacks," where malicious agents or bots inject biased ratings to artificially manipulate an item's popularity. Because these attacks often follow rigid patterns that deviate from the organic user-item manifold, they manifest

as high-residual outliers. This allows for the implementation of automated "flag-and-filter" protocols to maintain system integrity.

- **Identification of "Drifting" Items:** Items that consistently produce high residuals across a broad user base may indicate "data drift," where the item's cultural perception has changed (e.g., a movie gaining "cult" status years after release), signaling the need for a model retraining or localized fine-tuning.

9. Conclusion

In this project, we successfully engineered and validated a high-performance Collaborative Filtering system capable of processing the MovieLens 32M dataset on a single consumer-grade workstation. By transitioning from a naive interpreted baseline to a hardware-aware, JIT-parallelized Alternating Least Squares (ALS) implementation, we achieved a cumulative 8,767x speedup. This optimization reduced the training latency from several hours to 1.46 seconds per epoch, effectively bridging the gap between theoretical matrix factorization and production-scale performance.

The final model, optimized through a systematic grid search, achieved a competitive Test RMSE of 0.7654. Beyond predictive accuracy, our qualitative analysis confirms that the learned $K = 20$ latent manifold captures meaningful semantic structures and genre clusters directly from sparse interaction data. These findings demonstrate that rigorous low-level engineering and the utilization of dual sparse architectures (CSR/CSC) enable the deployment of sophisticated recommendation engines without the need for distributed clusters. The resulting latent representations provide a versatile foundation for broader business intelligence tasks, including anomaly detection and representation learning.

References

Koenigstein, N. and Paquet, U. The xbox recommender system. In *Proceedings of the 5th ACM Conference on Recommender Systems*, RecSys '11, Chicago, Illinois, USA, 2011. ACM.

Koren, Y., Bell, R. M., and Volinsky, C. Matrix factorization techniques for recommender systems. *Computer*, 42(8): 42–49, 2009.