GLocal-K with Temporal Information and Factorization

Enhancing Kernel-Based Recommendation with Matrix Factorization and Temporal Modeling

Link to GitHub

YOAV ZELINGER, Ben-Gurion University of the Negev, Israel
YEHONATAN KIDUSHIM, Ben-Gurion University of the Negev, Israel

1 Abstract

Recommender systems predict user preferences and generate personalized item suggestions, addressing the matrix completion problem where missing ratings must be inferred from sparse user-item interactions. Traditional Matrix Factorization (MF) methods capture linear relationships, while deep learning approaches, such as auto-encoder-based models, extract complex non-linear dependencies. GLocal-K [2] is a kernel-based model utilizing both local and global dependencies, by capturing non-linear relationships, which can impact its predictive accuracy.

In this work, we propose a hybrid model that enhances the GLocal-K framework by incorporating MF at the pre-training stage. Unlike the original GLocal-K model, which relies solely on kernel-based learning, our method integrates latent representations and various biases into the rating matrix reconstruction process. This improves the model's ability to generalize across datasets by capturing both linear and non-linear dependencies.

Additionally, we introduce a session-based bias component within MF to model short-term user preferences. Inspired by [3], we incorporate a dynamic user-session bias term that adjusts ratings based on session-specific interactions. To enhance temporal modeling, we employ a *time-decaying function* [4], ensuring that recent interactions have a stronger influence on predictions while preserving long-term patterns.

Evaluation on MovieLens' (ML) 100K and 1M datasets shows that our hybrid model consistently outperforms the baseline GLocal-K. In ML-100K, it surpasses standalone MF while using fewer latent dimensions, effectively balancing global and local structures. However, in the sparser ML-1M dataset, MF alone remains the best-performing model, suggesting that kernel-based refinements contribute less in highly sparse environments.

These findings emphasize the value of integrating structured factorization techniques within kernel-based models and highlight the critical role of bias modeling in recommendation systems.

2 Introduction

Recommender systems are essential in domains like e-commerce and streaming services, where they predict ratings and suggest relevant items based on user preferences. A key challenge in these systems is rating prediction, which is often approached as a matrix completion problem. The goal is to estimate missing values in a user-item rating matrix, representing user affinities. However, real-world matrices are typically sparse, as users rate only a small fraction of items, making it difficult to capture meaningful patterns and accurately predict missing ratings.

To address the matrix completion problem, various approaches have been developed, each aiming to estimate missing ratings based on different modeling techniques. These approaches can be broadly categorized into three groups: matrix factorization methods, neural network-based models and rank minimization techniques. Matrix factorization (MF) methods approximate the user-item rating matrix by decomposing it into low-rank latent factors that capture user preferences and item characteristics. Neural network-based models leverage deep learning techniques to capture complex, non-linear relationships in user-item interactions. Rank minimization techniques reconstruct the rating matrix by minimizing its rank while ensuring consistency with the observed ratings through an optimization process.

Authors' Contact Information: Yoav Zelinger, Ben-Gurion University of the Negev, Beer-Sheva, Israel, yoavzel@post.bgu.ac.il; Yehonatan Kidushim, Ben-Gurion University of the Negev, Beer-Sheva, Israel, kidushim@post.bgu.ac.il.

Our proposed improvement builds upon the GLocal-K model [2] by incorporating temporal information into the matrix completion process. The original GLocal-K method constructs the rating matrix using global and local kernel-based learning but does not explicitly utilize the timestamps available in the dataset. Since user preferences evolve over time, ignoring temporal information can limit the accuracy of rating predictions. To address this, we propose a hybrid approach that enhances the pre-training process of GLocal-K. Instead of relying solely on the pre-trained representations generated by the local kernelized weight matrix, we integrate a matrix factorization component that incorporates temporal dynamics through user-session bias and weighted decaying factors. This allows the model to capture both linear relationships using matrix factorization and higher-order interactions through the kernel-based learning mechanism. By doing so, our method leverages structured temporal dependencies and both linear and complex non-linear relationships to improve the accuracy of missing rating predictions.

Unlike matrix factorization and neural network-based models, which focus solely on either linear or non-linear relationships, our approach integrates both while also incorporating temporal information, enabling a more comprehensive representation of user-item interactions. Unlike rank minimization techniques, which rely solely on existing ratings and typically exclude additional information, our method leverages temporal patterns to capture a richer structure of user-item interactions.

The paper is structured as follows. We will start with a **background**, providing an overview of matrix completion and relevant methods. Next, we will present **related work**, with studies and techniques that form the foundation of our improvement. Afterwards, we will describe **our method**, detailing our proposed approach and implementation. Following that, we will present the **evaluation process** with the experimental setup and evaluation metrics. Then, we will end with the **results**, **discussion** on the experimental findings and **conclusions** that summarize our insights, contributions, and future directions.

3 Background

A fundamental task in recommender systems is rating prediction, where we aim to estimate the rating a user would give to an item. This task is often approached as a matrix completion problem, in which the objective is to predict missing values in a user-item rating matrix. This matrix represents user affinities for different items, with each entry corresponding to a rating provided by a user for a specific item. Given a partially observed rating matrix, the objective is to estimate the missing values, enabling the system to predict a user's potential rating for unseen items and generate personalized recommendations. However, in real-world scenarios, this matrix is typically highly sparse, as users rate only a small fraction of available items, leaving most entries unobserved. This sparsity significantly complicates the task of predicting missing ratings, as the limited data makes it difficult to capture meaningful patterns and generalize user preferences effectively.

To address these challenges, various matrix completion techniques have been proposed. According to the survey presented in [1], these methods can be broadly classified into the following categories.

3.1 Matrix Factorization Methods

Matrix factorization (MF) is a widely used technique for matrix completion, aiming to approximate the user-item rating matrix as the product of two low-rank matrices that capture latent user and item features. Given a sparse ratings matrix R, which represents a dataset with n users and m items, each entry $R_{u,i}$ denotes the rating given by user u to item i if the user has interacted with the item; otherwise, the entry is missing. Matrix Factorization (MF) methods decompose R into two dense matrices $U \in \mathbb{R}^{n\times d}$ and $V \in \mathbb{R}^{d\times m}$, such that the observed ratings in R are approximated by the product $\hat{R} = UV$, where d is the number of latent factors. Several models have been developed to enhance MF performance in recommender systems. **Non-negative Matrix Factorization (NMF)** is a matrix decomposition technique that factorizes the user-item rating matrix into two non-negative matrices, capturing latent user and item features. The non-negativity constraint ensures interpretability, making it particularly useful for recommender systems, as it provides meaningful and human-interpretable representations of latent factors. **SVD++** extends the standard Singular Value Decomposition (SVD) by incorporating both explicit ratings and implicit feedback, such as user interactions with items. It enhances the traditional matrix factorization model by integrating additional signals from co-rated items and user behaviors, leading to more accurate recommendations, especially for users with sparse rating histories. **SLIM (Sparse Linear Methods)** learns an item-item similarity matrix by solving an optimization problem while enforcing sparsity constraints, ensuring that recommendations are based on directly co-rated items. **FISM (Factored Item Similarity Model)** extends SLIM by incorporating matrix factorization, capturing transitive relationships between items and improving scalability through stochastic gradient descent. **LLORMA (Locally Low-Rank Matrix Approximation)** decomposes the rating matrix into multiple local low-rank approximations rather than assuming a single global low-rank structure. By applying kernel smoothing and leveraging local models, LLORMA enhances accuracy in capturing user-item relationships while enabling parallel computation, making it more efficient for large-scale datasets. **CoFactor** is an extension of traditional matrix factorization that incorporates item co-occurrence data to improve recommendation accuracy. Inspired by word embedding techniques, it learns item embeddings by factorizing both the user-item interaction matrix and an item co-occurrence matrix, ensuring that similar items are embedded closer in the latent space. This approach enhances performance in sparse datasets by leveraging implicit feedback and contextual item relationships.

3.2 Neural Network Models

Neural network-based approaches have been increasingly utilized in matrix completion tasks due to their ability to capture complex, non-linear relationships in user-item interactions. Unlike traditional matrix factorization methods, which assume a linear latent space, neural networks leverage deep representations to enhance prediction accuracy. One of the earliest models in this category is **AutoRec** is an auto-encoder-based model that learns compressed latent representations of users and items by reconstructing the observed ratings. It projects the rating matrix into a lower-dimensional space, capturing meaningful patterns for predicting missing ratings, though its reliance on linear layers may limit feature learning effectiveness. **Collaborative Denoising auto-encoder (CDAE)** extends the standard auto-encoder framework by introducing noise into the input data during training. This approach enhances the model's ability to learn robust latent representations by preventing it from simply memorizing observed ratings. By reconstructing the original user-item interactions from corrupted inputs, CDAE improves generalization and effectively handles sparse recommendation scenarios. **Deep Matrix Factorization (DMF)** leverages deep neural networks to model user-item interactions by replacing traditional matrix factorization with multi-layer perceptrons (MLPs). It learns hierarchical representations of users and items, capturing both explicit and implicit feedback. The model projects users and items into a shared latent space and estimates their similarity using a cosine function, improving the ability to model complex patterns in recommendation systems. **Graph-based methods** utilize the structure of user-item relationships as a bipartite graph, enabling the application of spectral graph convolution techniques. These methods, such as Graph Convolutional Networks (GCNs), capture high-order connectivity patterns, making them effective in handling cold-start problems and sparsity in recommender systems. By leveraging adjacency relationships and spectral filtering, these models improve the accuracy of missing value predictions in interaction matrices.

3.3 Rank Minimization Models

Rank minimization methods address the matrix completion problem by enforcing a low-rank structure on the observed data, leveraging the assumption that user-item interactions lie in a low-dimensional latent space. These approaches aim to recover missing entries by minimizing a rank-related objective function, often incorporating optimization techniques to enhance performance. **Iteratively Reweighted Nuclear Norm (IRNN)** is an optimization method for low-rank matrix completion that replaces the nuclear norm with a reweighted surrogate function. By iteratively updating singular values using a supergradient-based approach, IRNN effectively promotes sparsity in the singular spectrum, leading to better recovery of missing values while maintaining the structural integrity of the data. **Double Nonconvex Nonsmooth Rank (DNNR)** extends IRNN by introducing a weighted singular value function to further refine low-rank approximations. This method applies double nonconvex regularization on singular values, improving accuracy in rank minimization problems. However, the reliance on full SVD computations makes it computationally expensive for large-scale datasets. **Multi-Schatten-p Norm Surrogate (MSS)** is a method for low-rank matrix approximation that decomposes the optimization problem into multiple subproblems, each solved using an efficient block coordinate descent approach. By avoiding full singular value decomposition (SVD) of the entire matrix, MSS significantly improves computational efficiency, making it suitable for large-scale datasets. **Iterative Singular Value Thresholding Algorithm (ISVTA)** refines rank minimization by iteratively updating singular values using a controlled thresholding mechanism. By progressively adjusting singular values, ISVTA improves completion accuracy while ensuring structural preservation of the data. The iterative optimization process reduces computational costs and enhances convergence guarantees, making it an effective alternative to standard nuclear norm minimization. However, despite its improvements, ISVTA still faces scalability challenges when applied to large-scale recommendation systems due to its computational complexity.

3.4 GLocal-K: Global and Local Kernels for Recommender Systems

The algorithm we aim to improve is **GLocal-K** [2], a kernel-based matrix completion framework designed for recommender systems. It enhances rating prediction by leveraging both local and global kernels in a structured learning process. GLocal-K begins its **pre-training** phase with an item-based auto-encoder (AE) that reconstructs missing ratings. The encoder compresses the input rating matrix, while the decoder reconstructs it to estimate the missing values. As part of this pre-training process, a local kernel-based weight matrix is computed using a Radial Basis Function (RBF) kernel. This step strengthens important connections while reducing noise, allowing the model to capture meaningful user-item interactions more effectively. Once the pre-training phase is complete, the model proceeds to the **global kernel-based rating matrix construction** phase. In this phase, a global kernel-based rating matrix is constructed to refine the learned representations. The reconstructed item features are processed through item-based pooling, generating feature-indicative weights that adjust multiple kernel matrices. These adjusted kernels are aggregated into a global kernel, capturing broader structural patterns in the user-item space. The global kernel is then used as a convolution mask for the original data, which serves as input to the final **fine-tuning** stage. The refined representation is fed back into the auto-encoder, where the encoder extracts improved latent features and the decoder reconstructs a more accurate rating matrix. This process integrates both local and global information, resulting in more precise rating predictions.

4 Related Work

4.1 Session-Based Bias

To enhance the GLocal-K model, we rely on insights from *Yahoo! Music Recommendations: Modeling Music Ratings with Temporal Dynamics and Item Taxonomy* [3], which presents a matrix factorization approach designed for large-scale music recommendation systems. Their study, conducted on the Yahoo! Music dataset, provides a detailed analysis of how user preferences evolve over time and how item taxonomy can improve rating predictions. One aspect of their work focuses on item taxonomy as a means to address sparsity. By structuring items hierarchically and organizing them based on meta-data relationships, the model facilitates knowledge transfer between related entities, enabling a more coherent and context-aware recommendation process. While this helps improve predictions for rarely rated items, it does not account for temporal variations in user behavior, which play a critical role in dynamic recommendation environments. A key contribution of their study is the integration of temporal dynamics into matrix factorization. Unlike traditional approaches that assume static user preferences, their method incorporates time-aware latent factors and user biases to model preference shifts over time. This allows the system to adapt to seasonal trends, evolving popularity, and changing user tastes, leading to more accurate and personalized recommendations. Within the scope of user biases, they further introduce **session-based rating biases** to refine short-term user behavior representation. Users tend to consume multiple items in short bursts, where their ratings are influenced by contextual factors such as consumption order, mood, and prior exposure to similar content. Instead of treating these interactions as independent events, their model dynamically adjusts user preferences based on session-aware biases, ensuring a more fine-grained understanding of user intent at different points in time.

Building on these insights, we propose a hybrid approach that enhances GLocal-K by integrating matrix factorization into the local kernel component before the pooling stage. Specifically, we incorporate session-based rating biases into the factorization process, refining the local kernel's output to better capture short-term user preference variations. This integration leverages the complementary strengths of both techniques. Matrix factorization effectively captures linear user-item interactions, identifying direct correlations in the observed data. In contrast, the encoder in GLocal-K learns higher-order and non-linear relationships, extracting complex latent patterns beyond simple co-occurrences. By combining these approaches, our method improves the quality of the local kernel's pre-trained output, providing a stronger foundation for the pooling stage.

4.2 Time Decay Function

To represent the importance of item ordering within a session, we leverage insights from Yehuda Koren's *Collaborative Filtering with Temporal Dynamics* [4]. Koren's work addresses the challenge of modeling time-dependent user preferences in recommender systems by recognizing that user-item interactions evolve over time rather than remaining static. Unlike traditional collaborative filtering approaches that assume fixed user preferences, his study introduces a dynamic factorization model that explicitly accounts for shifting user tastes, item popularity fluctuations. Koren identifies that user preferences change due to multiple factors, including external influences such as seasonal trends and item popularity, as well as intrinsic user preference drift, where individuals gradually modify their tastes over time. Furthermore, he highlights how user biases and preferences evolve over time, influenced by factors such as changing item popularity, seasonal effects, and individual variations in rating behavior. Rather than assuming static user preferences, his study incorporates time-dependent user biases that adapt to these gradual changes. Additionally, the model employs an **exponential decay function** to appropriately weigh past interactions, ensuring that older preferences contribute less to current predictions while maintaining the long-term structure of user behavior. This approach enables the system to refine its recommendations by balancing persistent user tendencies with evolving preferences over time.

To effectively represent the session within our matrix factorization framework, we incorporate the exponential decay function proposed by Koren. This approach allows us to dynamically weight interactions within a session, ensuring that more recent interactions exert a stronger influence while gradually diminishing the impact of older ones. By integrating this temporal modeling technique, we enhance the representation of session-based rating biases, capturing short-term user behavior more accurately within the factorization process.

5 Method

The original architecture of the model (GLocal-K) is shown in Fig 1. The model operates in three main stages: pre-training, global kernel-based rating matrix construction, and fine-tuning. In the pre-training stage, an item-based auto-encoder (AE) reconstructs missing ratings by encoding and decoding the input rating matrix. In the global kernel-based rating matrix construction stage, multiple kernels representations are aggregated to form the overall global kernel, capturing broader structural patterns in the user-item space. The multiple kernels are constructed by average pooling each item and multiplying it with a kernel matrix. In the fine-tuning stage, the initial rating matrix is refined using global kernel-based convolution and used as input for the pre-trained auto-encoder. The auto-encoder is fine-tuned for further optimizing the learned representations and reconstructing the final rating matrix.

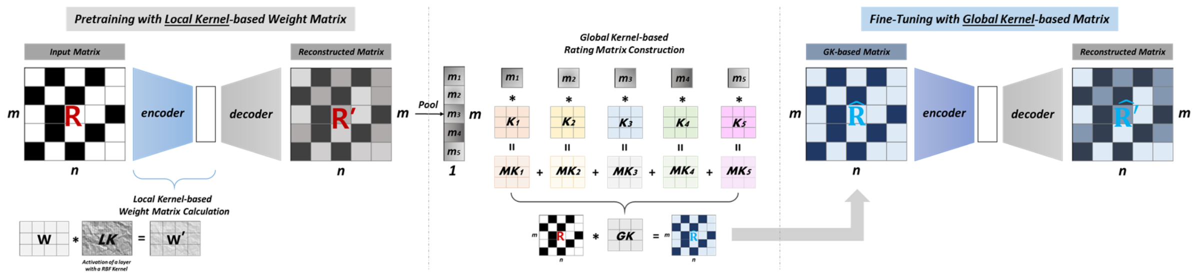
Pretraining with Local Kernel-based Weight Matrix
Input Matrix
Reconstructed Matrix
encoder
decoder
Local Kernel-based Weight Matrix Calculation
Activation of a layer with a RBF Kernel
Pool
Global Kernel-based Rating Matrix Construction
GK
Fine-Tuning with Global Kernel-based Matrix
GK-based Matrix
Reconstructed Matrix

Fig. 1. The original GLocal-K model architecture

The improvement we propose focuses on the transition between the pre-training stage and the global kernel-based rating matrix construction stage. Instead of using only the output of the pre-training stage as input to the pooling process, we introduce a hybrid weighted approach that combines both the pre-trained output and the result of a matrix factorization process applied to the original input matrix.

The hybrid model can be seen in Fig 2. By integrating these two models, our improvement leverages the complementary strengths of both approaches. Matrix factorization effectively captures linear user-item interactions, identifying direct correlations in the observed data. In contrast, the encoder in the pre-training stage models higher-order and non-linear relationships, capturing complex latent patterns beyond simple co-occurrences. In addition, by integrating matrix factorization into the auto-encoder, we can incorporate temporal information into the prediction, such as session-based bias with a decaying weight over time, as described in [3] and [4].

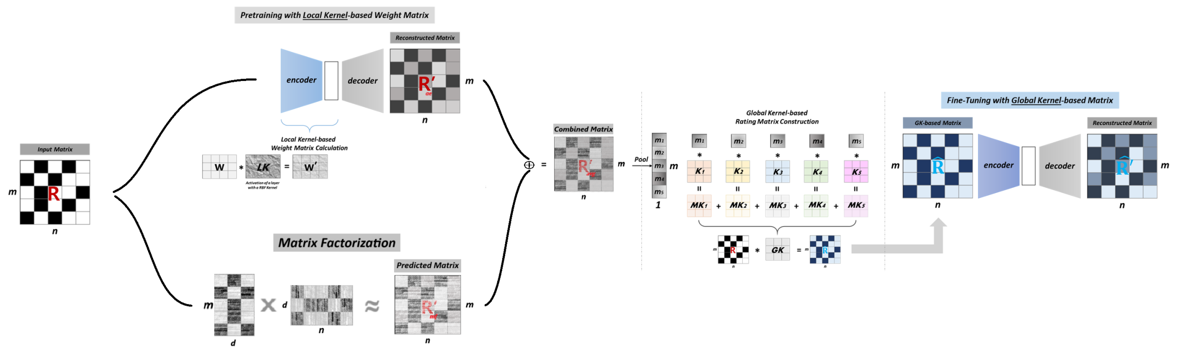
Pretraining with Local Kernel-based Weight Matrix
Input Matrix
Reconstructed Matrix
encoder
decoder
Local Kernel-based Weight Matrix Calculation
Matrix Factorization
Predicted Matrix
Combined Matrix
Pool
Global Kernel-based Rating Matrix Construction
GK
Fine-Tuning with Global Kernel-based Matrix
GK-based Matrix
Reconstructed Matrix

Fig. 2. GLocal-K and matrix factorization hybrid model architecture

Placing matrix factorization between the pre-training stage and the global kernel-based rating matrix construction stage ensures that the hybrid representation influences the global kernel formation before pooling. Rather than relying solely on the pre-trained local kernel output, incorporating structured factorization-based information at this stage enriches the learned representations. This adjustment enhances the stability of the pooling process, ensuring that the final global structure retains both localized refinements and broader relational patterns from the input data.

5.1 Matrix Factorization

The simplest form of Matrix Factorization (MF) predicts a rating as the dot product between a user's latent factor vector and an item's latent factor vector. In this approach, the user-item rating matrix is decomposed into two lower-dimensional matrices, where each row in the user matrix represents a user's latent preferences, and each column in the item matrix represents an item's latent characteristics. Given latent dimension $d \in \mathbb{N}$, user u with factor vector $p_u \in \mathbb{R}^d$ and item i factor vector $q_i \in \mathbb{R}^d$, the predicted rating $\hat{r}_{ui}$ by the user to the item is given by

$$\hat{r}_{ui} = p_u^T q_i \tag{1}$$

This factorization captures hidden structures in the data but assumes that all users and items interact purely based on learned latent factors. However, biases naturally exist in rating behaviors, as some users tend to rate consistently higher or lower than others. Additionally, a user's rating tendencies may be influenced by previous ratings within a certain time frame, while certain items may be inherently rated higher or lower due to their popularity. With this addition, given $b_{ui} \in \mathbb{R}$ as the bias when user u rates item i, the updated predicted rating is presented as

$$\hat{r}_{ui} = p_u^T q_i + b_{ui} \tag{2}$$

Some biases are global and affect the entire system, while others are specific to individual users or items. To account for these effects, the general framework for capturing bias in user-item ratings, described as

$$b_{ui} = \mu + b_i + B_u \tag{3}$$

where μ is the overall mean rating value (a constant), B_u stands for user bias and b_i stands for item specific bias. The term b_i represents the inherent bias of each item, reflecting its general tendency to receive higher or lower ratings regardless of the user.

5.2 User Bias

B_u captures the bias of each user, which is composed of multiple bias factors that collectively influence the user's overall rating tendency. Naturally, users tend to rate items in relation to previously rated items. If the first item a user interacts with is exceptionally good, the ratings of subsequent items are influenced in comparison to that initial experience. To account for such effects, we extend the user bias model by incorporating the session bias introduced at [3]. Users' consecutive ratings are grouped into sessions, where a new session begins after a period of idleness lasting T time with no rating activity. We denote the prefix of the session (until the user rated item i) as $session(u, i)$ and integrate this session-based bias into our user bias model to better capture short-term variations in user rating behavior. The extended user bias, which includes session biases, is formulated as

$$B_u = b_u + b_{u,session(u,i)} \tag{4}$$

where b_u represents the user-specific bias component. The session bias parameter $b_{u,\text{session}(u,i)}$ captures the bias between all ratings given by user u from the start of session until he rated item i.

5.3 Session Bias Presentation

In real-world scenarios, when predicting the rating a user assigns to a specific item, it is not guaranteed that the session in which the rating occurs was encountered during the training phase. Therefore, we aim to embody the session bias within the user bias. To formalize this, we suggest a generalized presentation function f for $session(u,i)$, the prefix of the session in which user u rates item i, as the mean of all predicted ratings in the session until the rating of item i. Let $S = \langle j_1, j_2, \ldots, i\rangle$ be the sequence of items rated by user u in $session(u,i)$. The initial session bias is then formulated as

$$f^{(0)}(session(u,i)) = \frac{1}{|S|}\sum_{j\in S} p_u^T q_j \tag{5}$$

This presentation has several issues. In this approach, two sessions of the same user containing the same items but in a different order will be represented identically. Moreover, all items within the same session are assigned equal weight, although more recent items tend to have a greater influence on the current rating.

To address this, inspired by Koren [4], we extend the session representation by incorporating a decaying factor, ensuring that more recent ratings have a stronger influence on the overall session representation. We adopt Koren's exponential decay function $e^{-\beta_u\cdot\Delta t_{u,i}}$, where $\beta_u > 0$ controls the user-specific decay rate, and $\Delta t_{u,i}$ represents the time gap between the prediction point and the timestamp at which user u rated item i ($t_{u,i}$). The updated session presentation is given as

$$f(session(u,i)) = \frac{1}{|S|}\sum_{j\in S} e^{-\beta_u\cdot\Delta t_{u,j}} p_u^T q_j \tag{6}$$

5.4 The Full Matrix Factorization Model

The simplified final matrix factorization formula is presented as

$$\hat{r}_{ui} = p_u^T q_i + \mu + b_i + b_u + b_{u,session(u,i)} \tag{7}$$

where the representation of $session(u,i)$ is given in Equation (6).

The user and item latent factors, all the biases and the decaying rate (β_u, as presented in Equation (6)) are learned from the data using stochastic gradient descent (SGD), where all learned parameters are L2-regularized. More specifically, for training example (u,i), SGD lowers the squared prediction error $e_{u,i}^2 = (r_{u,i} - \hat{r}_{u,i})^2$. With the regularization, the overall SGD learning rule is defined as

$$\min_{b_*,q_*,p_*,\beta_*} \sum_{(u,i)\in\phi_{train}} \left(r_{ui} - \mu - b_i - b_u - b_{u,session(u,i)} - q_i^T p_u\right)^2 + \lambda\left(b_i^2 + b_u^2 + b_{u,session(u,i)}^2 + \beta_u^2 + \|q_i\|^2 + \|p_u\|^2\right) \tag{8}$$

where ϕ_{train} is all the user-item rating pairs we have in the train data, and λ is the regularization rate.

5.5 The Extended Pre-Trained Stage

Looking back at the hybrid model architecture in Fig. 2, we can observe that the pre-trained auto-encoder and the Matrix Factorization algorithm provide two independent estimations of the full user-item rating matrix: $R'_{ae} \in \mathbb{R}^{m\times n}$, the reconstructed matrix from the auto-encoder, and $R'_{mf} \in \mathbb{R}^{m\times n}$, the predicted matrix from the matrix factorization process. To proceed to the next phase of the pipeline, we aggregate the two matrices into a single combined matrix ($R'_{avg} \in \mathbb{R}^{m\times n}$), which is calculated as the average of the two matrices:

$$R'_{avg} = \frac{1}{2}R'_{ae} + \frac{1}{2}R'_{mf} \tag{9}$$

R'_{avg} is considered the output of the pre-training stage. We use this matrix as input to the global kernel-based rating matrix construction stage. From there, the algorithm continues as in the original model.

6 Evaluation

The original model proposed in [2] was evaluated through experiments conducted on three widely used matrix completion benchmark datasets: MovieLens-100K (ML-100K), MovieLens-1M (ML-1M) and Douban (density 0.0630 / 0.0447 / 0.0152). These datasets comprise of (100 k / 1 m / 136 k) ratings of (1,682 / 3,706 / 3,000) movies by (943 / 6,040 / 3,000) users on a scale of $r \in \{1, 2, 3, 4, 5\}$. Since our improvement relies on temporal data and the Douban dataset provided by Monti et al. [5] lacks timestamps in the rating records, we evaluated our method using only ML-100K and ML-1M.

We followed the same train/test split suggested in [2], using an 80:20 split for ML-100K and a 90:10 split for ML-1M. To avoid data leakage, we performed the split based on timestamps, ensuring that test records are chronologically after the train records. Since our approach does not focus on handling the cold-start situation, after train/test splitting the data we removed test records containing users or items that did not appear in the training phase. This split resulted in a total of (82,863 / 995,911) ratings for ML-100K and ML-1M, respectively, covering (1,616 / 3,678) movies interacted with by (751 / 6,011) users, with densities of (0.0287 / 0.0097).

To evaluate our enhancement, we compare it with the original GLocal-K model. In the original paper [2], the model was evaluated on the datasets using a random train/test splits. This evaluation process leads to data leakage and results in an unreliable overestimation of performance. Thus, after precisely reproducing the claimed results and ensuring that the model was reliably reconstructed, we ran it using our suggested train/test split.

In addition, the baselines presented in the paper were evaluated using a different train/test split than the one we proposed, which does not account for the cold-start situation. As a result, a direct comparison with our results is not possible, and any such comparison may lead to an inaccurate assessment of the model's effectiveness.

The performance of our enhancement was evaluated using Root Mean Squared Error (RMSE), which is calculated as:

$$RMSE = \sqrt{\frac{1}{|\phi_{test}|} \sum_{(u,i)\in\phi_{test}} \left(r_{u,i} - \hat{r}_{u,i}\right)^2} \tag{10}$$

where ϕ_{test} is all the user-item rating pairs we have in the test data.

The experiments were conducted using the same architecture shown in Fig. 2. The input matrix is the rating matrix that contains only the training records. The whole model outputs a reconstructed matrix, which is used to measure performance on the test records using *RMSE*. The hyperparameters for GLocal-K were set to the same values suggested in [2].

For the matrix factorization model, we used 100 epochs, a learning rate of $\gamma = 0.002$, and L2-regularization with $\lambda = 0.05$. For the batch size, we set it to 1, resulting in SGD visiting the training examples one by one and updating the corresponding model parameters after each example. In addition, We tested four different latent dimension sizes: $d \in \{25, 50, 75, 100\}$.

The session bias introduced in [3] was originally tested on a music dataset, where an idleness time of $T = 5_{hours}$ was used. Since our model is applied to a movie dataset, we must select a hyperparameter that aligns with movie consumption patterns. Therefore, we define session boundaries based on an idleness time threshold of one week ($T = 1_{week}$). For the decaying function, we used the same time unit suggested by Koren [4], which is measured in days. In the session presentation, Δt is supposed to measure the time difference between the current time (**the**

time at which we want to predict the ratings) and the time when the session item's rating was recorded. Since we are evaluating the model's performance in predicting ratings that belong to the test data, using the actual current time would introduce bias and underestimate the model's capabilities. Thus, we used the average timestamp of the test records as the current time:

$$t_{\text{current}} = \frac{1}{\phi_{\text{test}}} \sum_{(u,i)\in\phi_{test}} t_{u,i}$$

To promote transparency and reproducibility in research, the complete code for our algorithms and experiments has been made publicly available on GitHub[1]. This repository provides access to all implementation details, allowing other researchers to replicate our results and further explore the effectiveness of our proposed methodologies.

7 Results

In this section we will present results divided to two experiments:

(1) The overall performance, compared to the original model, will help us evaluate whether our enhancement has led to a real improvement.
(2) The effect of latent dimensions on performance for each dataset, will help us identify the optimal latent space for each dataset.

7.1 Comparison with the Baseline Model

Table 1 presents a performance comparison between the isolated MF model, the baseline GLocal-K model, and the new combined architecture (GLocal-K + MF). The results are evaluated using *RMSE*, with the best performance for each dataset highlighted.

Model	ML-100K	ML-1M
GLocal-K (baseline)	1.0457	0.8951
MF	0.9692	**0.8916**
GLocal-K + MF	**0.9691**	0.8943

Table 1. Comparison of different models on the ML-100K and ML-1M datasets. For each model and dataset, the lowest RMSE is presented, with the best result highlighted.

Examining the results presented in Table 1, it is evident that the standalone matrix factorization (MF) model outperforms the baseline GLocal-K model. Furthermore, the hybrid architecture (GLocal-K + MF) achieves comparable performance, demonstrating a slight advantage on the smaller dataset (ML-100K).

These results suggest that the hybrid model (GLocal-K + MF) is particularly effective when the data is less sparse. Its slight advantage on the denser ML-100K dataset indicates that combining GLocal-K with MF better captures local structures, which standalone matrix factorization may not fully exploit. This highlights the hybrid approach's ability to leverage richer user-item interactions in less sparse settings.

7.2 Effect of Latent Dimensions

The results presented in Table 2 show the performance of the MF and GLocal-K + MF models on the ML-100K dataset for varying factor dimensionality (d). To better visualize these trends, we refer to Figure 3, where the

[1]https://github.com/yoavzelinger/Glocal_K

performance of the models is shown graphically, providing a clearer view of how RMSE changes with varying dimensionality.

In the case of GLocal-K + MF, we observe that as the factor dimensionality increases up to 75, the RMSE decreases, indicating improved performance. However, when the dimensionality increases from 75 to 100, there is a decline in performance, as the RMSE increases. On the other hand, for the MF model, we see a decline in performance up to 50 dimensions, after which the performance improves. As the dimensionality grows beyond 50, the RMSE continues to decrease, leading to better prediction accuracy.

Model	$d = 25$	$d = 50$	$d = 75$	$d = 100$
MF	0.9772	0.9810	0.9728	**0.9692**
GLocal-K + MF	0.9984	0.9739	**0.9691**	0.9713

Table 2. Performance of the MF and GLocal-K + MF models on the ML-100K dataset: prediction accuracy is measured by RMSE (lower is better) for varying factor dimensionality (d).

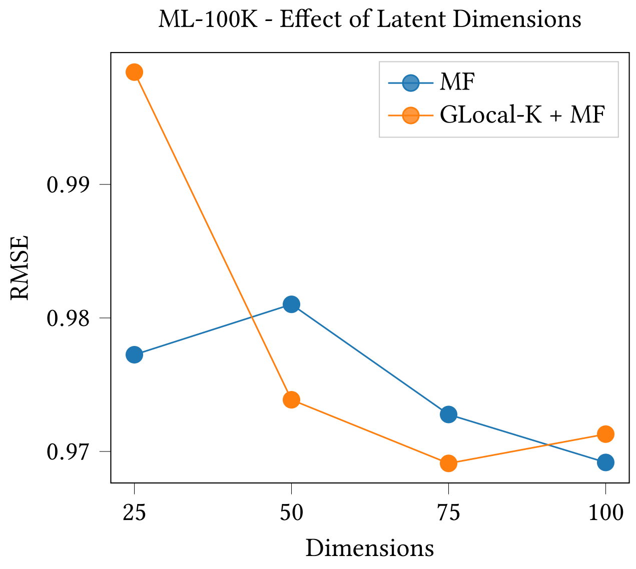

Fig. 3. Prediction accuracy (RMSE) of MF and GLocal-K + MF models on the ML-100K dataset for varying factor dimensionality (d). Lower RMSE indicates better performance.

The results in Table 3 show the performance of the MF and GLocal-K + MF models on the ML-1M dataset for varying factor dimensionality (d). For a clearer view of these trends, see Figure 4, which provides a graphical representation of how RMSE changes with dimensionality.

In the case of GLocal-K + MF, we observe that as the factor dimensionality increases up to 50, the RMSE decreases, indicating improved performance. However, when the dimensionality increases from 50 to 100, there is a decline in performance, as the RMSE increases. For the MF model, we observe the opposite trend compared to the hybrid model: performance declines up to $d = 50$, after which it improves as the dimensionality increases from 50 to 100, with the RMSE continuing to decrease.

Model	d = 25	d = 50	d = 75	d = 100
MF	0.8927	0.8942	0.8922	**0.8916**
GLocal-K + MF	0.8979	**0.8943**	0.8954	0.8954

Table 3. Performance of the MF and GLocal-K + MF models on the ML-1M dataset: prediction accuracy is measured by RMSE (lower is better) for varying factor dimensionality (d).

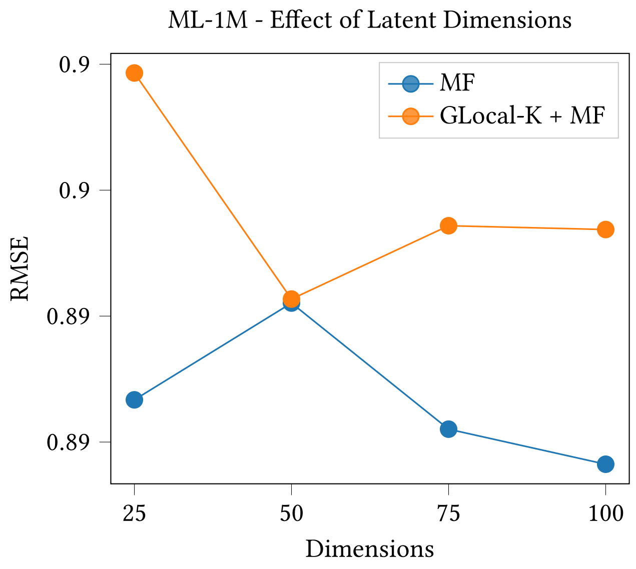

Fig. 4. Prediction accuracy (RMSE) of MF and GLocal-K + MF models on the ML-1M dataset for varying factor dimensionality (d). Lower RMSE indicates better performance.

From the results on both datasets, we observe a similar trend. As shown in Fig. 3 and Fig. 4, the hybrid approach achieves its best performance with fewer latent dimensions compared to the independent MF model. A possible explanation for this could be the specific role of each model in the hybrid architecture. As the latent dimension increases, the MF model captures more local patterns. The combination of the pre-trained auto-encoder (which is designed to capture local patterns within the data) with MF model may lead to overfitting as the dimensionality grows, with the model becoming increasingly focused on the local patterns in the data. In contrast, relying solely on the MF model requires identifying both global and local patterns, which may explain the observed performance difference.

Another notable observation shared by both datasets concerns the independent MF model. In both figures, we see that after reaching a high RMSE peak in the middle, the MF model's performance continues to improve as the number of latent dimensions increases. This finding also suggests that further increasing the latent dimensions beyond $d = 100$ could lead to even better performance.

8 Discussion

The results presented in Table 1 highlight the strength of Matrix Factorization (MF) as a standalone model, achieving lower RMSE compared to the baseline GLocal-K model. This can be attributed to its ability to efficiently capture linear user-item interactions by decomposing the rating matrix into latent factors, allowing it to generalize user preferences effectively, even in sparse datasets. Furthermore, as shown in Figures 3 and 4, increasing the latent dimensionality enhances MF's performance, indicating that a richer latent space improves its ability to capture complex user-item relationships.

Combining MF with GLocal-K (GLocal-K + MF) creates a balanced model that integrates both linear and non-linear relationships while leveraging global and local structures. The results demonstrate that the hybrid model consistently outperforms the baseline GLocal-K across both datasets. However, its advantage over standalone MF varies depending on dataset sparsity. In the ML-100K dataset, where data is less sparse, the hybrid model not only surpasses the baseline but also achieves a lower RMSE with fewer latent dimensions than MF alone. This suggests that GLocal-K effectively captures high-order dependencies, allowing the hybrid model to achieve strong predictive performance without requiring a large latent space (in the MF model). Conversely, in the sparser ML-1M dataset, MF alone achieves the best results, outperforming both GLocal-K and the hybrid model. This indicates that in highly sparse settings, MF's ability to generalize global patterns provides a stronger predictive foundation, while the kernel-based refinements in GLocal-K contribute less significantly. The high performance achieved by MF may be attributed to the extensive use of biases, including the overall data bias (μ), user bias (b_u), item bias (b_i), and user-session bias ($b_{u,session(u,i)}$).

Another factor affecting the performance of the hybrid model is the number of pre-training epochs. In the original GLocal-K implementation, the optimal number of epochs was set to 30 for ML-100K and 20 for ML-1M [2]. These values were determined based on experiments with the standalone GLocal-K model, without considering the hybrid architecture that integrates MF. As a result, the fixed epoch settings may not be optimal for the combined approach. Given that our hybrid model incorporates both GLocal-K and MF, it is likely that a different number of epochs would yield better performance. Since we adhered to the pre-training settings from [2], the hybrid model may not have reached its full potential. Performing a dedicated search for the optimal number of epochs for the hybrid model could lead to improved accuracy, particularly in datasets where the RMSE trends indicate sensitivity to pre-training duration.

Another factor that may have influenced the results is the stage at which MF is integrated into the hybrid model. Currently, MF is aggregated with the reconstructed matrix from the pre-training stage. However, integrating MF at a different stage could yield different results. For instance, incorporating MF at the fine-tuning stage, where the convolutional layer operates directly on the MF-predicted matrix, may refine global patterns more effectively. Similarly, a late-fusion approach, where MF is combined with the final output, could alter the model's ability to balance local and global dependencies. Our findings indicate that increasing latent dimensions in the hybrid model leads to overfitting. However, if MF were integrated at the fine-tuning stage, a higher number of dimensions might instead enhance global structure learning, potentially improving performance.

9 Conclusion

In this work, we introduced a hybrid approach that integrates Matrix Factorization (MF) into the GLocal-K framework, enhancing rating prediction by combining the strengths of both linear and non-linear modeling. Unlike the original GLocal-K model [2], our approach incorporates MF at the pre-training stage, where its latent representations are leveraged to refine the reconstructed rating matrix before applying the global kernel-based rating matrix construction stage. This integration allows the model to capture both linear user-item interactions and non-linear refinements more effectively.

Additionally, inspired by [3], beyond the traditional global, user, and item biases, we incorporate a session-based modeling component within MF to capture short-term user preferences. The user-session bias term dynamically adjusts ratings based on session-specific interactions. Unlike traditional static user biases, our approach models user preferences within the sequential context of interactions, allowing the system to better reflect real-world user behavior. Furthermore, we structure the MF representation to preserve the sequential order within each session. Our approach integrates a time-decaying function, inspired by [4], allowing the model to distinguish between immediate preferences and long-term patterns.

Through evaluation on the ML-100K and ML-1M datasets, we demonstrated that our hybrid model consistently outperforms the baseline GLocal-K. In ML-100K, our approach achieved lower RMSE than MF alone while using fewer latent dimensions, showcasing its efficiency in balancing global structure with local refinements. However, in the sparser ML-1M dataset, MF model alone demonstrated the best-performing results, suggesting that kernel-based refinements contribute less in highly sparse environments.

Future research could explore alternative ways to integrate MF within the GLocal-K architecture. For instance, incorporating MF directly into the fine-tuning stage, where the convolutional layer operates on the MF-predicted matrix instead of the original, may enhance global pattern extraction. Another promising direction is a late-fusion approach, where MF predictions are combined with the final output instead of the pre-training stage. Additionally, rather than relying on a hybrid-weighted approach, a cascading method—where the output of one model serves as the input for the other—could be investigated. Each approach is worth exploring with different hyperparameters, including those from the original GLocal-K, such as the number of epochs at each stage of the algorithm, the optimal selection of weights in weighted hybrid approaches, or MF-specific hyperparameters like the number of latent dimensions, all of which could impact overall performance.

References

[1] Zhaoliang Chen and Shiping Wang. 2022. A review on matrix completion for recommender systems. *Knowledge and Information Systems* 64, 1 (01 Jan 2022), 1–34. doi:10.1007/s10115-021-01629-6

[2] Soyeon Caren Han, Taejun Lim, Siqu Long, Bernd Burgstaller, and Josiah Poon. 2021. GLocal-K: Global and Local Kernels for Recommender Systems *(CIKM '21)*. Association for Computing Machinery, New York, NY, USA, 3063–3067. doi:10.1145/3459637.3482112

[3] Noam Koenigstein, Gideon Dror, and Yehuda Koren. 2011. Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy. In *Proceedings of the Fifth ACM Conference on Recommender Systems* (Chicago, Illinois, USA) *(RecSys '11)*. Association for Computing Machinery, New York, NY, USA, 165–172. doi:10.1145/2043932.2043964

[4] Yehuda Koren. 2009. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Paris, France) *(KDD '09)*. Association for Computing Machinery, New York, NY, USA, 447–456. doi:10.1145/1557019.1557072

[5] Federico Monti, Michael Bronstein, and Xavier Bresson. 2017. Geometric Matrix Completion with Recurrent Multi-Graph Neural Networks. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/2eace51d8f796d04991c831a07059758-Paper.pdf