Multimedia Tools and Applications (2024) 83:23599–23634
https://doi.org/10.1007/s11042-023-15208-0

KFDBN: Kernelized Finetuned Deep Belief Network for recommendation

Nouhaila Idrissi[1] · **Ahmed Zellou**[2] · **Zohra Bakkoury**[1]

Received: 13 June 2022 / Revised: 1 March 2023 / Accepted: 30 March 2023 /
Published online: 17 August 2023

Abstract

In today's technologically evolved world, users have become accustomed to personalized tools that provide accurate and precise recommendations that consider their needs and interests, leading to a perceived reduction in information overload in a fiercely competitive marketplace. However, sparsity issues render the effective prediction infeasible and hamper the performance of the recommendation, as users typically only rate a small proportion of the offered items. To solve these problems, we propose, in this paper, a novel hybrid model named Kernelized Finetuned Deep Belief Network (KFDBN) for accurate rating prediction and efficient Top-k recommendation. First, we learn a deep generative model using a Deep Belief Network (DBN) to capture higher-level latent feature representations of users and items to effectively predict missing entries in the rating matrix. Next, we present the practical KFDBN approach to seamlessly incorporate features extracted by the generative model into kernel-based feature extraction. Subsequently, KFDBN is leveraged to learn reliable hidden features that improve the performance of sparse recommendations. Finally, we introduce the KFDBN-based imputation technique to create a denser user-item interaction matrix for a relevant item ranking in the Top-k recommendation. Experimental evaluations on six datasets in various domains reveal that the proposed generative model enhances the accuracy of rating prediction and outperforms the state-of-the-art methods by 21.32% and 16.35% in RMSE and MAE, respectively, averaging on all used datasets. Moreover, the KFDBN exceeds baseline approaches and enhances the Top-k recommendation performance by 5.2% and 8.1% in HR@10 and NDCG@10, respectively, averaging on the six datasets.

Ahmed Zellou and Zohra Bakkoury contributed equally to this work.

✉ Nouhaila Idrissi
nouhaila_idrissi@um5.ac.ma

Ahmed Zellou
ahmed.zellou@um5.ac.ma

Zohra Bakkoury
z.bakoury@um5.ac.ma

[1] AMIPS Research Team, EMI, Mohammed V University, Rabat, Morocco

[2] SPM Research Team, ENSIAS, Mohammed V University, Rabat, Morocco

Keywords Recommender systems · Collaborative filtering · Sparsity · Deep Belief Network · Kernel methods · Matrix factorization

1 Introduction

The expanding demand for personalization has paved the way for the emergence of Recommender Systems (RSs) to defeat the tyranny of prevailing data overwhelm by predicting users' intentions. The efficiency of such a system is generally determined by how effectively diverse users' interests are identified [1, 17], how accurately users' transactions on items are represented [17, 32], and how significant user and item features are extracted [17]. Collaborative filtering (CF) is a fast-evolving technique [18, 36] that relies on feedback scores to infer similar users and items. The Netflix prize competition, which began in 2006, has led to several recent advances in this area and pioneered the research community's access to movie rating datasets, which drew the interest and attention of many researchers and scientists [4]. However, comprehending precisely varied users' preferences demands complex models requiring massive information, which runs counter to the sparsity limitation. The latter refers to the situation where available data of users' interactions with items is sparse [17], making it challenging for the RS to infer accurate predictions.

The continuous upsurge of better prediction accuracy has led to the use of Deep Learning (DL), a subfield of Machine learning (ML), to propose robust and powerful RSs. The current growth of interest in such deep models in the recommendation field is due to novel methods that showed practical training, thus outperforming shallow techniques, especially in sparse conditions [17, 23, 24, 44]. On the other hand, kernel methods have also gained a keen interest in RSs, due to their meaningful feature extraction and efficient structure discovery in data. Recent approaches based on kernel techniques leverage the kernel trick to explore relevant structures in non-linear empirical data in different contexts, thereby capturing reliable features for better prediction [7, 30], or classification [31, 44], as long as the kernel function is selected accurately.

Several state-of-the-art papers integrate side information into DL-based models as an inevitable choice to propose hybrid solutions for missing rating prediction [22, 24]. The additional information could be items properties [24], textual reviews [43], or simply demographics gathered from users' profiles [37]. Nevertheless, accurate personalization is not about names, ages, or places, as it does not track users' volatile tastes and behaviors [17]. Not to mention that such data is hard to collect due to privacy aspects. On the other hand, to generate Top-k recommendations, consisting of ranked and personalized lists of items to users, some papers employ auxiliary information in Matrix Factorization (MF) techniques as regularizations to learn latent features [33, 48]. However, such feature vectors represent linearly user and item interactions, which is notably not useful for heterogeneous real-world data, especially if the user-item matrix is scarce and additional information is unavailable [17].

To address these problems, we introduce, in this paper, the Kernelized Finetuned Deep Belief Network (KFDBN) model to accurately predict missing ratings and generate efficient Top-k recommendations even if only a few ratings of users on items are available. We first learn a deep generative model using a Deep Belief Network (DBN) for missing rating prediction. The model captures higher-level hidden feature representations of users and items by allowing reliable layer-by-layer learning of weights, resulting in better generalization. The deep structure is performed by stacking multiple Restricted Boltzmann Machines

(RBMs) to learn the visible layer's distribution probabilistically, allowing each successive RBM to apprehend meaningful features in the sparse rating matrix. We use the Contrastive Divergence (CD) algorithm [13] to initialize the deep architecture of the generative model in an unsupervised way. Such initialization avoids the typical problems of gradient-based approaches that are time-consuming and have to deal with many layers and parameters throughout the learning process, which might lead to local minima [23]. Next, the compressed data representation and the restricted parameter space establish a starting point for supervised fine-tuning. This optimization ensures further effective generative reconstruction of user-item interactions yielding accurate predictions and personalized recommendations.

We also present the efficient KFDBN classifier that emphasizes extracting powerful hidden features of users and items, unlike classification-based CF approaches, which merely employ shallow feature extraction. We demonstrate that KFDBN can be successfully applied to enhance classification performance for different datasets by using the deep generative approach as a base pre-model that produces, along with Kernel Principal Component Analysis (KPCA) [35], meaningful latent feature vectors to conduct an efficient two-step kernel-based discriminative semi-supervised classification.

Another crucial contribution of this paper is to develop a novel imputation technique to solve sparsity issues in MF-based recommendations. We use KFDBN as a hybrid imputation technique to infer missing entries in the original sparse rating matrix. The positive predictions emanated from the generative model are merged with ratings of relevant items resulting from the KFDBN classifier component to create a denser user-item rating matrix for the Top-k recommendation. This leads to solving the optimization issues that occur when MF-based methods are applied directly to sparse data since the error function considers only available inputs. On the other hand, the proposed unified model discovers the underlying local and global non-linear correlations on sparse data to infer missing user-item entries, which significantly differs from state-of-the-art MF-based recommendation approaches [11, 34] that treat unobserved ratings as negative inputs compensating for the skewed distribution of known interactions.

The contributions of this paper are summarized as follows:

- We propose a novel hybrid model named Kernelized Finetuned Deep Belief Network (KFDBN) that combines the practical training of the DBN with kernel methods' relevant exploration of data structures to overcome sparsity issues in RSs. We first present the effective fine-tuned (FDBN) model for accurate missing rating prediction.
- We show that going deeper helps to capture reliable features by comparing the performance of RBM with DBN.
- We demonstrate that supervised fine-tuning of the generative model yields better prediction results.
- We present a practical hybrid approach to seamlessly incorporate features extracted by the generative model into kernel-based feature extraction using KPCA to create user profiles. The unified KFDBN feature extraction helps to apprehend important information among items by discovering correlation features among co-rated items from users' perspectives, capturing thus precise and relevant items that satisfy users' preferences. The proposed user profile expansion mechanism significantly impacts the performance enhancement of the sparse recommendation.
- We prove that output features extracted using the hybrid KFDBN lead to higher Support Vector Classification (SVC) outcomes, demonstrating that KFDBN can be successfully extended to effective semi-supervised classification.

- We develop a novel imputation technique that infers missing ratings in the original sparse user-item interaction matrix to solve sparsity issues in MF-based recommendations.
- Finally, extensive experimentation on six sparse datasets in diverse domains shows that the FDBN generative model significantly enhances the missing rating prediction and outperforms the state-of-the-art methods by 21.32% and 16.35% in terms of RMSE and MAE, respectively, averaging on all used datasets. We also empirically prove that the KFDBN leads to higher performance for the Top-k recommendation and yields 5.2% and 8.1% improvement in $HR@10$ and $NDCG@10$, respectively, averaging on the six datasets.

The remainder of this paper is divided into five sections. Section 2 provides an overview of related work in DL-based feature learning and kernel-based methods applied in RSs. The foundations and preliminaries of RBM, DBN, and Kernel methods are presented in Section 3. Section 4 is dedicated to thoroughly describing the proposed KFDBN and discussing the different components. Section 5 reports experimental results, answers the research questions addressed in this paper and discusses the findings. Finally, the paper conclusions and future direction of the work are presented in Section 6.

2 Related work

The related work of the paper covers two chief points: DL-based models for enhanced feature extraction and kernel-based technique applied in RSs.

2.1 Enhanced-feature extraction based on deep networks

Autoencoders (AEs) are widely employed in RSs for their appropriate latent feature representation. Bathla et al. [3] uses a shared layer in an AE to incorporate indirect social trust with ratings and learn accurately from linked representations. Hybrid AE-based models include users' side information to address data scarcity [46]. However, users' latent information is typically diverse and multimodal and may contain improper aspects for prediction. As a result, emphasizing such features by the AE will cause overfitting. Not to mention that reconstruction outputs of AE-based approaches are sometimes directly leveraged as predictions of unknown rating values [23], which may impair the effectiveness of the recommendation.

CapsMF [22] leverages Bi-directional Recurrent Neural networks (RNN) with Capsule Networks for textual analysis of reviews to generate recommendations, thus enhancing the capabilities of Convolutional Neural Networks (CNN) and RNN-based models in text modeling and representation. On the other hand, Capsule networks need to be fine-tuned since they take time to be trained, especially for large-scale recommendations.

Generative Adversarial Networks (GAN) are employed to learn user and item representations [9, 29] and perform rating augmentation [6] to alleviate the sparsity in CF. However, GAN-based models tend to predict high-value ratings for most missing entries in the user-item matrix [6]. Hence, if proposed approaches adopt such a biased augmented matrix for generating recommendations, they are likely to result in poor performance as they fail to apprehend users' actual interests. Furthermore, such models' discriminator sometimes leads

to a high percentage of contradicting labels for the same items due to early and misleading convergence [29]. Additionally, inverting in a wholly trained GAN-based system is a tedious task.

2.2 Kernel-based methods for recommendation

Though linear methods are simpler to interpret and have shown promising prediction results [38], they may not be proper to train the non-linearly separable and heterogeneous real-world data [17], which promoted the application of kernel methods in RSs. Chen et al. [7] proposes kernel-based matrix completion for a neighborhood-based recommendation. Approximation for missing ratings is enhanced by integrating the Gaussian kernel function with the similarity measure. The technique combines various characteristics in non-linear latent feature spaces by employing a multi-kernel framework, adjusting the kernels' weights to generate relevant recommendations.

The Kernel Context Recommender System (KCR) employs the kernel trick to build a model based on a context rating matrix for prediction using contextual information [21]. The research shows that variation in kernel functions leads to higher performance than standard context-aware RSs. However, Principal Component Analysis (PCA) could be further applied to filter relevant context features to enhance the efficiency of the kernel mapping framework. Besides, investigating a DL-based model can learn from diverse representations from varied contexts, thus improving the quality of recommendations.

To transcend the idea of incorporating mapped features of similar users directly into classifiers to generate recommendations, [31] used the Multiple Kernel Learning for feature separation; then incorporated output results in the Adaptive Neuro-Fuzzy Inference System (ANFIS) for classification. The proposed model has shown relevant sensitivity and specificity. However, it has not been evaluated in scarce conditions. Furthermore, it is relevant solely for heart disease recommendations. Another paper [44] combined the Scale-Invariant Feature Transform (SIFT) with the Faster R-CNN algorithm and Support vector machine (SVM) classification for music recommendation based on DL and IoT architecture. First, SIFT is used to locally extract features based on the Gaussian convolution kernel. Then, Fast-RCNN is utilized to extract underlying multi-scale features of the scene images. Finally, middle-layer characteristics with spatial information are acquired using SVM classification to develop the background music system. Otherwise, Singular Value Decomposition (SVD) may be leveraged to infer low-dimensional vectors of users' data [30]; after that, a kernel function can discover relevant latent features between users before predicting ratings using neural-based models such as Multilayer perceptron (MLP) [30].

3 Background preliminaries

3.1 Restricted Boltzmann Machine

A Restricted Boltzmann Machine is a stochastic generative Neural Network (NN) that learns a probability distribution over its set of inputs [13]. The weighted bipartite graph of the RBM, $\theta = (W_{ij}, v_{bias_i}, h_{bias_j})$ is composed of two layers (the visible layer V and the hidden layer H) connected through a set of weights W_{ij}, and biases (v_{bias_i}, h_{bias_j}). RBMs can be used for regression, classification, dimensionality reduction, CF, and so forth [19]. Depending on the task, they can be trained in a supervised or unsupervised way. A Restricted Boltzmann Machine is illustrated by Fig. 1.

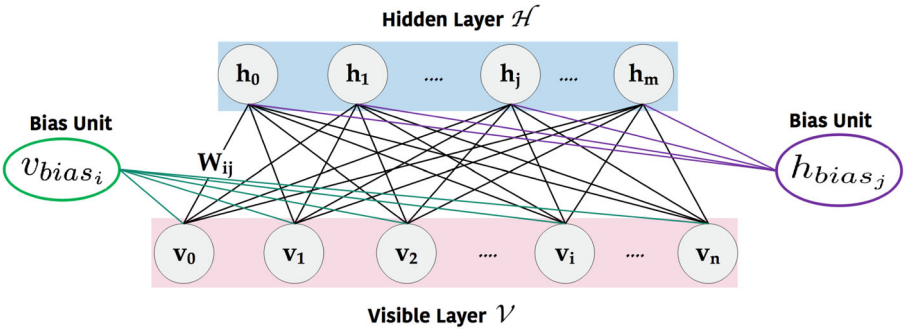


Fig. 1 Restricted Boltzmann Machine

The binary units of the RBM are connected to form a joint configuration (v, h) whose overall energy is defined by (1)

$$E(v, h; \theta) = \sum_{i}^{n}\sum_{j}^{m} W_{ij} v_i h_j - \sum_{i} v_{bias_i} v_i - \sum_{j} h_{bias_j} h_j \tag{1}$$

In a Boltzmann machine (BM), the only restrictions are that all connections are symmetrical, and no unit has a connection to itself. However, due to their high complexity, such networks are much less utilized compared to RBMs, which are BMs in which there are no visible-visible, and hidden-hidden connections [13]. The joint probability distribution is then calculated using (2).

$$p(v, h; \theta) = \frac{e^{-E(v,h;\theta)}}{\eta(\theta)} \tag{2}$$

Where $\eta(\theta) = \sum_v \sum_h e^{-E(v,h;\theta)}$ defines the appropriate normalization factor.
The probability that a model assigns states of hidden units to the vector h is computed by marginalizing out the visible vector v as presented in (3).

$$p(h; \theta) = \sum_{v} p(v, h; \theta) = \frac{1}{\eta(\theta)} \sum_{v} e^{-E(v,h;\theta)} \tag{3}$$

To get a sample generated by the network according to the probability $p(v)$, the Gibbs sampling, which is a Markov chain Monte Carlo (MCMC) technique, is used [13]. Yet, to obtain samples that are faithful to the learned distribution, it is theoretically necessary to iterate between the states of the Markov chain until convergence.

Figure 2 depicts the sampling process of hidden units $h^{(t)} \leftarrow p(h = 1 \mid v^{(t)})$ followed by visible units sampling $v^{(t+1)} \leftarrow p(v = 1 \mid h^{(t)})$.

Training the RBM is thus performed by maximizing the log probability, which consists simply in minimizing the Kullback-Leibler divergence [13] between two distributions that are the target distribution $\langle v_i h_j \rangle^0$ of the learning base, namely the input, and the distribution $\langle v_i h_j \rangle^\infty$ modeled by the RBM. The gradient of the log probability with respect to a specific weight is, therefore, computed according to (4).

$$\begin{aligned} \frac{\partial log(p(v))}{\partial W_{ij}} &= \frac{\partial log(\frac{1}{\eta(\theta)} \sum_h e^{-E(v,h;\theta)})}{\partial W_{ij}} \\ \frac{\partial log(p(v))}{\partial W_{ij}} &= \langle v_i h_j \rangle^0 - < v_i h_j \rangle^\infty \end{aligned} \tag{4}$$

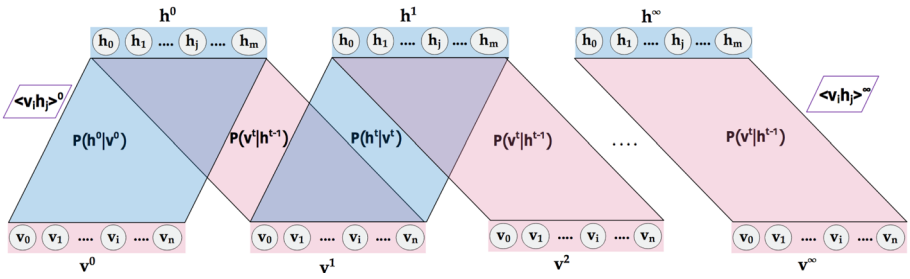


Fig. 2 Gibbs sampling iterative process

3.2 Deep belief network

The Deep belief Network is a probabilistic model mainly composed of stacked RBMs. The training of the DBN is performed through a greedy layer-by-layer learning algorithm [14]. A first unsupervised pre-training phase is achieved successively in each layer before finally employing a supervised optimization on the initialized parameters resulting from the first stage. When trained to a set of unsupervised examples, the stacked RBMs act as feature detectors to probabilistically reconstruct input data. The chief purpose of this unsupervised step is to match the inputs and outputs of successive hidden layers. This reconstruction criterion thus ensures that most of the information is preserved after training all the layers. The model parameters at layer Z, along with conditional probabilities of the hidden units, are used as necessary generated data to train the model's parameters at layer $Z + 1$. The model can be further optimized with supervision to perform fine-tuning.

3.3 Kernel-based methods

The use of kernel techniques is driven by the possibility of projecting data values to another space of a higher dimension where the linear separation is achievable [7]. The performance of such approaches crucially depends on the appropriate choice of the kernel function. The latter is a symmetric, continuous, and positive function corresponding to a scalar product in a higher dimension space. Formally, for each kernel K, there exists a function $\phi : \mathcal{X} \subset \mathbb{R}^n \rightarrow \mathcal{H}$ such that:

$$K(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}} \quad \forall x, x' \in \mathcal{X} \tag{5}$$

with $\mathcal{X}$ the original space and $\mathcal{H}$ the Hilbert space of dimension greater than $\mathcal{X}$. The kernel functions examined in this work are presented in Table 1. Along with the cost C, each kernel optimizes a specific parameter (i.e., γ, β, and P), thus allowing an efficient performance, making it possible to benefit from simple and rigorously guaranteed techniques while dealing with non-linear problems.

In contrast to the linear kernel, which performs linear boundaries in a higher dimension, the polynomial kernel expands m input features to m^P features with degree-$P > 0$ of polynomials by performing feature conjunctions that are valuable for cases where input data is normalized [16]. This kernel also optimizes β, a real weighting parameter that trades off the impact of lower polynomials. The Radial Basis Function (RBF) is a Gaussian kernel independent of the nature of the observed data, which makes it more suitable for cases when there is no prior information about training data [45]. Here, γ controls the variance of the distribution; hence finding the optimal value of γ yields better performance of the kernel. Unlike other statistical similarity metrics, the cosine kernel measures the similarity

Table 1 Kernel functions

Kernel function	Mathematical formula	Optimization parameter
Linear	$K(x, x') = \langle x, x' \rangle$	C
Polynomial	$K(x, x') = (\gamma \langle x, x' \rangle + \beta)^P$	C, γ, β, P
RBF	$K(x, x') = \exp(-\gamma \parallel x - x' \parallel^2)$	C, γ
Cosine	$K(x, x') = \langle \frac{x}{\|x\|}, \frac{x'}{\|x'\|} \rangle$	C
Sigmoid	$K(x, x') = \tanh(\gamma \langle x, x' \rangle + \beta)$	C, γ, β

as a normalized dot product of two vectors in a high dimensional space by abstracting out their magnitude, which is beneficial in cases where the size of vectors influences the similarity computation, primarily when the Euclidean distance is employed [42]. Otherwise, employing a sigmoid kernel, the learning algorithm becomes similar to a two-layer NN, less prone to a local optimum with great generalization results. A typical value for the slope γ is $\frac{1}{M}$ where M is the dimension of input data [28].

Thanks to their theoretical foundations, SVMs provide a unique approach less prone to overfitting and advantageous compared to an Artificial Neural Network (ANN) that may not allow robust classification of different samples [10]. Kernel methods have been particularly effective for classification based on SVMs, which search for an optimal margin hyperplane that corresponds to the accurate potential separations among the closest data points belonging to different classes while being located with the maximum distance to observations [5]. The mapping of points to a space of a higher dimension and the appropriate choice of optimization parameters of the kernel function not only decreases the computational cost, mainly when dealing with extensive data, but also enhances the generalization performance.

Kernel methods are also leveraged in KPCA [35], which are a non-linear extension of PCA. The latter determines new independent variables, called Principal Components (PC), thus finding relevant projection spaces for the data by maximizing their projected variance. In other words, extracted information using KPCA is non-linearly related to the input data. In fact, by employing kernel functions, KPCA projects the observations into a new space of higher dimension and then carries out an ordinary PCA. Compared to other non-linear extensions of PCA, for instance, NNs, KPCA is known for its stability and a reduced computational cost [45].

4 Proposed model

This section represents the proposed Kernelized Finetuned Deep Belief Network (KFDBN) model, which comprehends three major components:

- KFDBN feature extraction:
  - Fine-tuned DBN (FDBN) for missing rating prediction: We first develop a deep generative model of stacked RBMs by leveraging the unsupervised pre-training as an initializer of the deep architecture instead of randomly initializing the model's parameters. This unsupervised pre-training phase results in a regularization effect that renders the learning process more efficient by establishing a restricted parameter space for further optimization. We then incorporate semi-supervised learning to improve the model's generalization

performance by reaching a lower minimum of the cost function. Once all RBMs have been trained, the initialization phase is complete. The compressed representation of data is used as input to a logistic layer that will be trained with backpropagation (BP) to perform discriminative fine-tuning. Hence, the FDBN allows the extraction of higher-level hidden features of users and items and leads substantially to better prediction results in the reconstructed rating matrix.

– kernel-based feature extraction: This stage provides an application of KPCA to capture further the relevant hidden structure of categorical FDBN output features. Processed latent factors are used to expand Users' and Items' Profiles to discover correlation features among co-rated items from users' perspectives. As a result, each item is characterized by a number of hidden characteristics of users that assessed the item. Then, PCA converts learnt factors into a new set of related unified KFDBN features, retaining only the relevant patterns and meaningful information.

- KFDBN item relevancy classification: In this phase, we investigate a hybrid approach to incorporate the extracted KFDBN features and available feature data as input vectors of a kernel-based discriminative classifier aiming to efficiently group items into three classes referring to items utility: (relevant, neutral, and irrelevant).
- KFDBN-SVD Top-k recommendation: The driving idea behind this phase is to enhance the ranking performance in MF-based recommendation approaches. We use KFDBN results to infer missing ratings in the original sparse user-item interaction matrix. The positive predictions emanated from the FDBN generative model are merged with ratings of relevant items resulting from the KFDBN classifier component to create a denser imputed matrix for SVD-based Top-k recommendation.

The proposed KFDBN architecture is depicted in Fig. 3. The following sub-sections discuss and examine the influence in an in-depth investigation of each component on the entire proposed model.

4.1 KFDBN feature extraction

4.1.1 FDBN for missing rating prediction

Unsupervised pre-training In this phase, we aim for an accurate generative reconstruction of sparse input data. This is achieved as long as the model's parameters are initialized effectively. Therefore, to overcome typical problems of gradient descent-based approaches that can quickly get stuck with local minima [23] and do not allow the efficient training of large dimensions of hidden layers [13]; we join an additional layer of feature detectors to an RBM to learn an effective deep generative model for missing rating prediction. Here, the layer-wise unsupervised training is performed on stacked RBMs to capture higher-level features of input data. To expedite the sampling stage depicted in Fig. 2, the Contrastive Divergence is applied [13]. The primary purpose is to maximize the log-likelihood, i.e., $\partial log(p(v))$ of the data to learn the weight matrix $W = \{W_{ij}\}$, which guarantees the efficient reconstruction of the sparse input user-item matrix. The weights of the network can be updated according to (6).

$$\triangle W_{ij} = \alpha(\langle v_i h_j \rangle^0 - \langle v_i h_j \rangle^\infty) \tag{6}$$

Where $\langle v_i h_j \rangle^0$ is the expectation under the training distribution, $\langle v_i h_j \rangle^\infty$ denotes the expectation under reconstructed sparse data distribution provided by the RBM, and α is the

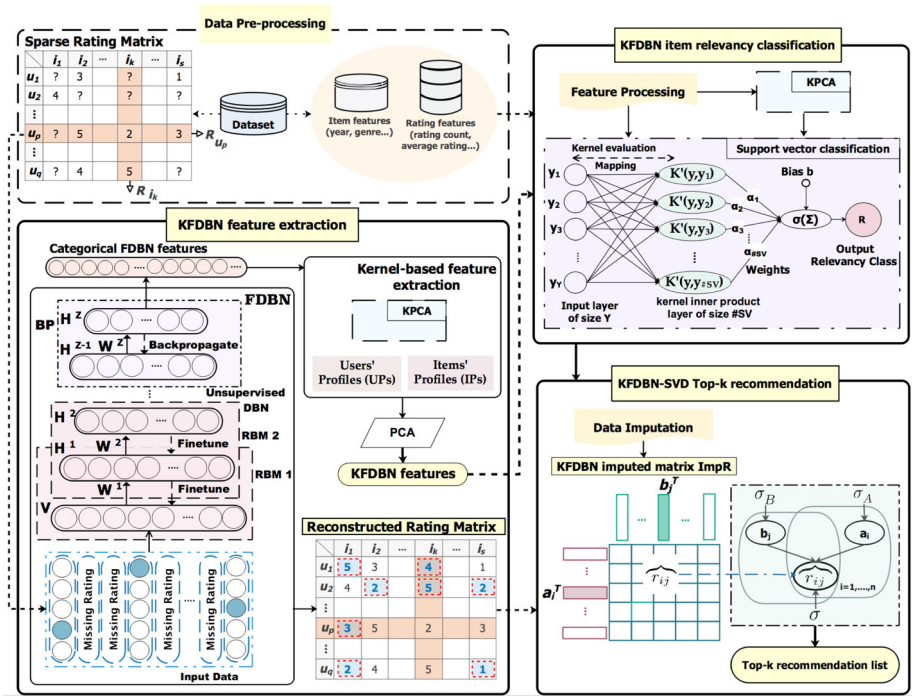


Fig. 3 The architecture of the proposed KFDBN model

learning rate. Since the units of the same layer have no connection, the states in a layer only depend on the states of the units in the other layer, hence $\langle . \rangle^0$ can be easily computed. On the other hand, $\langle . \rangle^\infty$ can be obtained by applying the CD algorithm by incorporating two significant points that are:

- Instead of starting at a random visible point, i.e., $v^{(0)}$, CD uses an input example from training data to initialize the Gibbs procedure, i.e., $v^{(0)} \leftarrow v \in \widehat{T}$, which makes it possible to explore the regions near the training examples to modify the surface of the energy function [13].
- To ensure a regularization effect by only performing G_S iterations of Gibbs sampling [2].

The algorithm of the Contrastive Divergence with G_S steps is described in Algorithm 1. Equation (7) is used to calculate the hidden binary units, and then the visible states are computed using (8). Output visible units are approximate reconstructions of training data.

$$p(h_j = 1 \mid v) = \sigma\left(\sum_i^n w_{ij} v_i + h_{bias_j}\right) \tag{7}$$

Where $\sigma(s) = \frac{1}{1+e^{-s}}$ is the sigmoid logistic function.

$$p(v_i = 1 \mid h) = \sigma\left(\sum_j^m w_{ij} h_j + v_{bias_i}\right) \tag{8}$$

By comparing the computed probability with the predefined threshold κ according to (9), h_j is set to 0 if $p(h_j = 1 \mid v)$ is lower than κ and 1 otherwise. The same principle is applied when it comes to activating or inhibiting the states of visible units.

$$x_k = \begin{cases} 0 & p(x_k = 1 \mid y) \langle \kappa \\ 1 & p(x_k = 1 \mid y) \rangle \kappa \end{cases} \tag{9}$$

Where κ ranges from 0.5 to 1.

Input $\widehat{T}$: Training Data, α: learning rate, W: Weight matrix of n input vector dimension and m hidden layer dimension, v_{bias}: visible bias vector, h_{bias}: hidden bias vector, m_o: momentum, G_S: number of Gibbs sampling steps
Output Updated W, v_{bias}, h_{bias}

Initialize α, G_S, m_o, $W \in R^{n \times m}$, $v_{bias} \in R^n$, $h_{bias} \in R^m$
for each $v \in \widehat{T}$ **do**
  $v^{(0)} \leftarrow v$
  for $t = 0....G_S - 1$ **do**
    for all hidden units j **do**
      Calculate $h_j^{(t)} \leftarrow p(h_j = 1 \mid v^{(t)})$ according to (7)
      Sample $h^{(t)} \in \{0, 1\}$ from $p(h_j = 1 \mid v^{(t)})$ according to (9)
    end for
    for all visible units i **do**
      Calculate $v_i^{(t+1)} \leftarrow p(v_i = 1 \mid h^{(t)})$ according to (8)
      Sample $v^{(t+1)} \in \{0, 1\}$ from $p(v_i = 1 \mid h^{(t)})$ according to (9)
    end for
  end for
  for $i = 1....n$, $j = 1....m$ **do**
    $\triangle W_{ij} \leftarrow \alpha(p(h_j = 1 \mid v^{(0)}).v_i^{(0)} - p(h_j = 1 \mid v_i^{(G_S)}).v_i(G_S))$
    $\triangle v_{bias_i} \leftarrow \alpha(v_i^{(0)} - v_i^{(G_S)})$
    $\triangle h_{bias_j} \leftarrow \alpha(p(h_j = 1 \mid v^{(0)}) - p(h_j = 1 \mid v^{(G_S)}))$
    Update the parameters using output gradients
    $W_{ij} \leftarrow W_{ij} m_o + \triangle W_{ij}$
    $v_{bias_i} \leftarrow v_{bias_i} m_o + \triangle v_{bias_i}$
    $h_{bias_j} \leftarrow h_{bias_j} m_o + \triangle h_{bias_j}$
  end for
end for

Algorithm 1 Contrastive Divergence($\widehat{T}$, α, W, v_{bias}, h_{bias}, m_o, G_S).

When training the RBM with the Contrastive Divergence algorithm, the model's parameters are updated in each training epoch, and the G_S steps of Gibbs sampling are conducted. We divide the training data $\widehat{T}$ into mini-batch samples with a batch size of B_S data points. Updating the summations only needs to be done once in each batch; hence, a batch of input vectors still requires $O(nm)$ complexity, where n and m are input vector and hidden vector dimensions, respectively. Moreover, the CD algorithm avoids the exponential complexity of summing over all of the values of the visible or hidden units by sampling the model distribution with the probability distribution of the input vector. The calculation of conditional

probabilities has a computation complexity of $O(nm)$. Therefore, the overall complexity of the Contrastive Divergence with G_S of Gibbs sampling steps is given by $O(B_S G_S nm)$.

Input $\widehat{T}$: Training Data, N_{ls}: Number of layers, α_{pr}: pre-training learning rate, W: Weight matrix, h_{bias}: hidden bias vector, m_o: momentum, G_S: number of Gibbs sampling steps
Output Unsupervised DBN

Initialize α_{pr}, N_{ls}, G_S, m_o, W, h_{bias}
for $Z = 1...N_{ls}$ **do**
  while stopping criteria is not met **do**
    Assign input sample from the training set
    Sample $H^1 \leftarrow V$ from $\widehat{T}$
    for $z = 1...Z-1$ **do**
      Sample H^z from $p(H^z \mid H^{z-1})$
    end for
    Update W^Z, h^Z_{bias}, h^{Z-1}_{bias}
    Contrastive Divergence(H^{Z-1}, α_{pr}, W^Z, h^{Z-1}_{bias}, h^Z_{bias}, m_o, G_S)
  end while
end for

Algorithm 2 Unsupervised pre-training($\widehat{T}$, N_{ls}, α_{pr}, W, h_{bias}, m_o, G_S).

The first RBM of the unsupervised DBN model is trained using the Contrastive Divergence with the $v^{(0)}$ input sample of training data presented on its visible layer, regarding categorical five-star ratings as five nodes, one for each potential evaluation value. Once trained, for each sample v of the training set, the first RBM associates a configuration h of neurons from the hidden layer. h is constructed by sampling the distributions $p(h_j \mid v^{(t)})$. The configurations h thus obtained are compressed versions of v. Each of the following RBMs learns about the hidden representations of the previous one. Here, the hidden layer of the first RBM acts as a visible layer for the next RBM. Once all RBMs have been trained in an unsupervised fashion, the compressed version of input data presented in the hidden layer of the last RBM, along with the model's inner parameter space, defines an initialization point to a supervised fine-tuning algorithm and thus enabling the effectiveness of semi-supervised strategies in achieving optimal generalization results. The unsupervised pre-training stage of the DBN is detailed in Algorithm 2.

Supervised fine-tuning In the supervised fine-tuning phase, we back-propagate the error to calculate gradients of the loss function to obtain optimal weights and biases that will further enhance the performance of the generative rating prediction. The proposed model is considered fine-tuned if it can find the desired target T associated with the corresponding input data V. This is achieved by minimizing the quadratic cost function MSE defined by (10).

$$\ell = \frac{1}{2N} \sum_{V} \parallel T - H^{(V,N_{ls})} \parallel^2 \tag{10}$$

Where N_{ls} is the number of layers, N is the number of training examples, and H is the vector of output activation states from the NN when using V as input.

Input $\widehat{T}$: Training data with L labels, N_{ls}: Number of layers, α_{fn}: fine-tuning learning rate, ℓ reconstruction loss
Output FDBN

for each training example $V \in \widehat{T}$ **do**
  feedforward propagation
  $H^1 \leftarrow V$
  for $Z \in 2, 3, ..., N_{ls}$ **do**
    $D^{(V,Z)} \leftarrow W^Z H^{Z-1} + h^Z_{bias}$
    $H^{(V,Z)} \leftarrow \sigma(D^{(V,Z)})$
    Compute $out^{(V,N_{ls})} \leftarrow \nabla_H \ell_V \cdot \sigma'(D^{(V,N_{ls})})$ according to (11)
  end for
  for $Z \in N_{ls} - 1, N_{ls} - 2, ...2$ **do**
    Backpropagate the error
    Compute $out^{(V,Z)} \leftarrow ((W^{(Z+1)})^T out^{(V,Z+1)}) \cdot \sigma'(D^{(V,Z)})$
  end for
end for
for $Z \in N_{ls}, N_{ls} - 1, ...2$ **do**
  $W^Z \leftarrow W^Z - \alpha_{fn} \sum_V out^{(V,Z)} (H^{(V,Z-1)})^T$
  $h^Z_{bias} \leftarrow h^Z_{bias} - \alpha_{fn} \sum_V out^{(V,Z)}$
end for

Algorithm 3 Supervised fine-tuning($\widehat{T}$, L, $N_{ls}, \alpha_{fn}, \ell$).

The aim of the fine-turning stage, described in Algorithm 3, is to investigate how weight changes and biases affect the reconstruction loss function ℓ by computing the gradients ($\frac{\partial \ell}{\partial W}$, $\frac{\partial \ell}{\partial h_{bias}}$). Thence, an intermediate output error $out^{N_{ls}}$ in the last layer is computed by taking into account the rate of change of ℓ according to output activations: $\nabla_H \ell = \frac{\partial \ell}{\partial H^{N_{ls}}}$ and σ', the derivative of the activation function. Given that $D^{(Z)} \leftarrow W^Z H^{Z-1} + h^Z_{bias}$, σ' indicates how fast the activation function is changing at $D^{N_{ls}}$. $out^{N_{ls}}$ is measured using the (11). In case of MSE function, $\nabla_H \ell = H^{N_{ls}} - T$.

$$out^{N_{ls}} = \frac{\partial \ell}{\partial H^{N_{ls}}} \sigma'(D^{N_{ls}}) \tag{11}$$

The output error for any layer can be measured by merging (11) with (12). The latter is based on the error of the next layer, which involves a backward propagation of the error through the NN.

$$out^Z = (W^{(Z+1)})^T out^{(Z+1)} \cdot \sigma'(D^Z) \tag{12}$$

To examine the impact of the weights and biases of any layer Z on ℓ the gradient $\frac{\partial \ell}{\partial W^Z}$ and $\frac{\partial \ell}{\partial h^Z_{bias}}$ are calculated using (13) and (14) respectively.

$$\frac{\partial \ell}{\partial W^Z} = H^{Z-1} out^Z \tag{13}$$

$$\frac{\partial \ell}{\partial h^Z_{bias}} = out^Z \tag{14}$$

The gradients are further updated using different batches until training examples are entirely fine-tuned. Resulted weights and biases from Algorithm 3 are then employed to optimize the FDBN model for better prediction.

4.1.2 Kernel-based feature extraction

In this stage, we seamlessly incorporate hidden features learned by the generative FDBN model into the kernel-based feature extraction using KPCA. Since visible units of the FDBN model are not just simple nodes with a single entry but have instead been considered five nodes to represent individual evaluations (1–5 rating scale), we accordingly convert each set of five binary latent feature nodes of the last hidden layer of the FDBN into categorical feature data. Then, KPCA maps these non-linear FDBN features $\{x_1, x_2..., x_X\} \in \mathbb{R}^M$ to a high-dimensional space by transforming them to linearly separable factors F'. The mapping projection function ϕ extends data points into a feature space $\mathbb{F}$ of a higher dimension as follows:

$$\phi : \mathbb{R}^M \rightarrow \mathbb{F}, x \rightarrow \phi(x) \tag{15}$$

Where $\{\phi(x_1), \phi(x_2), ...\phi(x_X)\}$ are assumed to be centered at the origin of $\mathbb{F}$, i.e $\sum_{k=1}^{X} \phi(x_k) = 0$

Given $\overline{C}$, the covariance matrix in the resulting space $\mathbb{F}$, KPCA satisfies the equation $\nu\overline{V} = \overline{C}\overline{V}$ for positive eigenvalues ν and eigenvectors $\overline{V} \in \mathbb{F} \setminus \{0\}$. $\overline{C}$ is described by (16).

$$\overline{C} = \frac{1}{X}\sum_{j=1}^{X} \phi(x_j)\phi(x_j)^T \tag{16}$$

One may note that all $\overline{V}$ with $\nu \neq 0$ lie in the subspace generated by $\{\phi(x_1), \phi(x_2), ...\phi(x_X)\}$. We may thence consider the following equivalent equations for all $i = 1, ..., X$:

$$\phi(x_i)\overline{C}\overline{V} = \nu(\phi(x_i)\overline{V}) \tag{17}$$

and that there exist eigenvector coefficients $e_u (u = 1, ..., X)$ such that:

$$\overline{V} = \sum_{u=1}^{X} e_u \phi(x_u) \tag{18}$$

Given a kernel function K, to get e_u, we define the $X \times X$ kernel matrix $\overline{K}$ according to (19)

$$\overline{K}_{uj} := \langle \phi(x_u), \phi(x_j) \rangle = (K(x_u, x_j))_{uj} \tag{19}$$

Then, substituting $\overline{C}$, and $\overline{V}$ into (17), we arrive at:

$$X\nu\overline{K}e = \overline{K}^2 e \tag{20}$$

Which is equivalent to solving $X\nu e = \overline{K}e$ for nonzero ν [35].

It is noteworthy that, by employing kernel functions $K(x, x')$ defined in Table 1 dot products of projected vectors $\phi(x)$ are computed without explicitly carrying out the map $\phi(.)$. Besides, data points cannot be directly centered in $\mathbb{F}$. Hence, the final definition of the kernel matrix of centered data can be described using the matrix $\overline{K}$ as follows:

$$\widetilde{\overline{K}_{uj}} = \overline{K} - 1_X\overline{K} - \overline{K}1_X + 1_X\overline{K}1_X \tag{21}$$

Where the matrix $(1_X)uj = \frac{1}{X}$, further details can be referred in [35].

To extract Principal Components (PC), corresponding to a kernel function K, we first compute the matrix $\overline{K}$, then calculate and normalize $\overline{V}$ in the feature space $\mathbb{F}$. The mapping of a data point sample x is calculated onto $\overline{V}^i$ in $\mathbb{F}$, which can be expressed by:

$$(K\ PC)_i(x) = \overline{V}^i \phi(x) \tag{22}$$

Therefore, for the test point x, we can describe the $i-th$ feature representation value as follows:

$$F_i' = \overline{V}^i \phi(x) = \sum_{u=1}^{X} e_u^i K(x, x_u) \tag{23}$$

As shown in Fig. 4, we use output F' features to create Users' Profiles (UPs), which in turn generate novel Items' Profiles (IPs). Here, we suppose that all users who have rated a particular item have common characteristics. Hence IPs represent hidden features hc in items that may interest the users. In other words, an item i_k is described by merged UPs of users that rated the item, which is beneficial in cold-start situations where items and users are new to the system, and no initial rating or feedback is available [20]. Therefore, dimensionality reduction using PCA is applied to new UP-based items' profiles to boost computational efficiency and capture important information among items relying on UPs. PCA constructs the transformation matrix M that covert $D = \{hc_1, hc_2..., hc_D\}$, the original combined hidden features of IPs in q-dimensional space to new reduced KFDBN features D' in a t-dimensional space, where $(t \leq q)$. The Eigen Decomposition (ED) decomposes the Covariance Matrix $\overline{C}(D.D^T)$ into three matrices as follows:

$$D.D^T \rightarrow S.L.S^T \tag{24}$$

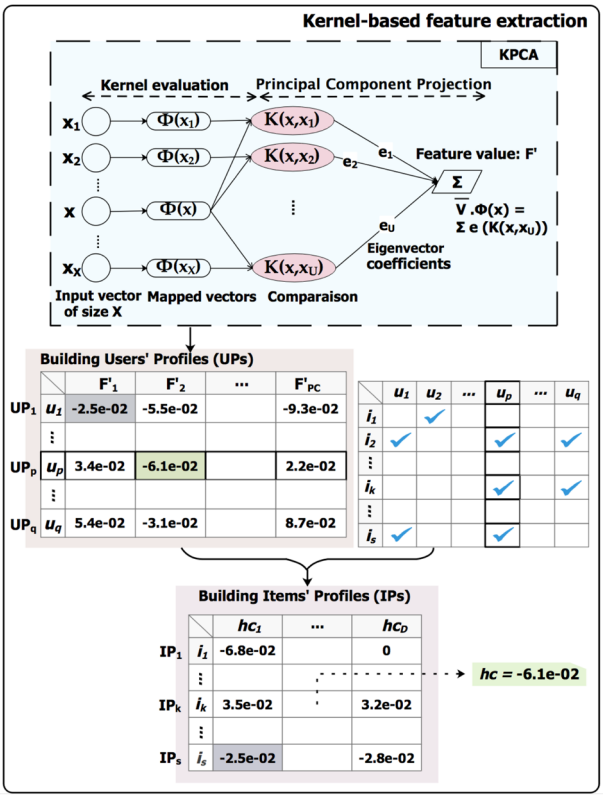


Fig. 4 Kernel-based feature extraction

Where S is the matrix $(q \times q)$ that contains Eigenvectors $\overline{V}$, which represent directions; and L is the diagonal matrix $(q \times q)$ with eigenvalues ν that represent data magnitude.

Input $D \in \mathbb{R}^{n \times q}$
Output $D' \in \mathbb{R}^{n \times t}$
1: Compute $\overline{C}$ $(D.D^T)$
2: Compute ν and $\overline{V}$ using ED
3: Sort in descending order ν to sort $\overline{V}$
4: Construct $M(q \times t)$ with t top $\overline{V}$
5: Use M to transform D to $D' = D.M$

Algorithm 4 PCA(D,q,t).

One primary benefit of employing PCA on UP-based items profiles remains in further discovering correlation features among co-rated items from users' perspectives, capturing thus precise and relevant items that satisfy users' needs. The impact of resulting KFDBN features on item relevancy classification is examined in-depth through the following sections.

4.2 KFDBN item relevancy classification

To perform item relevancy classification, we emphasise powerful deep feature engineering by incorporating the hidden, latent factors extracted by KFDBN in the classification task, unlike traditional classifiers that apply sallow techniques in the learning process [40, 44]. The purpose is to take advantage of the practical training procedure of the FDBN generative approach by using it as a base pre-model that produces input features, along with KPCA, to improve classification results when facing sparsity issues. The KFDBN model is leveraged to create input vectors for kernel-based Support Vector Classification (SVC) in two prominent cases:

- KPCA is applied on features as a pre-processing stage to apprehend other meaningful hidden structures to enhance the item relevancy classification. Let $\{x_1, x_2..., x_X\} \in \mathbb{R}^M$ be the training data with X observations of features resulting from the KFDBN model combined with other feature data related to rating and items in a given space $\mathbb{R}$ of dimension M. KPCA first maps X to a high-dimensional space using Equations defined in Section 4.1.2. Therefore, the projection of training data is used as input to the SVC.
- KFDBN output features are combined with other features and then directly incorporated to perform the kernel-based semi-supervised classification approach.

Let $\{(y_p, R_p)\}_{p=1}^{Y}$ be the training set with input examples $y_p \in Y \subset \mathbb{R}^N$ and corresponding labels $R_p \in \{1, ..., N_c\}_{p=1}^{Y}$ of N-dimensional feature vectors y_p. Let N_c be the number of possible classes; we use the one-versus-one (OvO) approach, also referred to as pairwise classification [25], to introduce $\frac{N_c(N_c-1)}{2}$ classifiers, one for each pair of classes. The following binary optimisation equation is solved to train the classifier for the class pair separating the class of index r from that of index r'.

$$min_{w^{rr'},b^{rr'},\xi^{rr'}} \frac{1}{2} \parallel w^{rr'} \parallel^2 + C \sum_l \xi_l^{rr'} (w^{rr'})^T \tag{25a}$$

$$\text{subject to} \quad (w^{rr'})^T \phi(y_l) + b^{rr'} \geq 1 - \xi_l^{rr'}, \text{ if } R_l = r \tag{25b}$$

$$(w^{rr'})^T \phi(y_l) + b^{rr'} \leq -1 + \xi_l^{rr'}, \text{ if } R_l = r' \tag{25c}$$

$$\xi_l^{rr'} \geq 0 \tag{25d}$$

Where $w^{rr'}$ is a vector of dimension L ($L \geq N$), C is the penalty parameter, ξ_l are the slack variables, and b denotes the bias term.
The decision function for classes r and r' is defined according to (26).

$$\mathscr{F}_{r,r'}(y) = (w^{rr'})^T \phi(y) + b^{rr'} \tag{26}$$

Here, $\phi(y)$ are mapped vectors in a feature space of dimension L and $\mathscr{F}_{r,r'}(y) = -\mathscr{F}_{r',r}(y)$. The voting strategy [15] is used to classify a new sample. Specifically, for the input instance y, we compute

$$\mathscr{V}_r(y) = \sum_{r' \neq r, r'=1}^{N_c} sign(\mathscr{F}_{rr'}(y)) \tag{27}$$

Where

$$sign(t) = \begin{cases} 1, & \text{if } t > 0 \\ 0, & \text{otherwise} \end{cases} \tag{28}$$

Namely, the binary classifier gives 1 as its vote to $\mathscr{V}_r(y)$ when the instance y is allocated to class r rather than class r'. Otherwise, it votes 0. $\mathscr{V}_r(y))$ is hence the sum of all contributed votes obtained from the classifiers determining that y is assigned to the class r.

$$r^*(y) = \underset{r=1,\ldots,N_c}{\arg\max} \mathscr{V}_r(y) \tag{29}$$

Therefore, as described by (29), the sample y is classified into the relevancy class (relevant, neutral, and irrelevant) that receives the highest number of votes from the classifiers.

The employed multi-support vector classification using the OvO technique is beneficial for kernel methods since it solves a two-class binary classification instead of using the entire training data N_c times. The proposed classification approach allows thus robust and efficient classification of both small and large data. Moreover, the KFDBN classifier leverages a quadratic optimization that does not suffer from local optimum and overcomes the overfitting problem by employing regularization principles, which expedites the learning process naturally.

4.3 KFDBN-SVD Top-k recommendation

Now that the KFDBN model has generated missing rating predictions using its FDBN generative model and has classified the relevant items using the classifier component, we exploit the KFDBN results to propose a new imputation technique for MF-based recommendation approaches. Specifically, we merge high FDBN's output predictions with ratings of relevant items to create a denser user-item interaction matrix. Then we apply SVD to the imputed rating matrix to generate Top-k recommendation lists of ranked items for the users.

The regularized SVD decomposes the rating matrix $R \in \mathbb{R}^{n \times m}$ into two low-rank matrices, such that users are associated with feature vectors a_i of $A \in \mathbb{R}^{n \times p}$, and items are described by their latent features b_j of $B \in \mathbb{R}^{p \times m}$, where the rank $p << min(n, m)$.

Hence, the matrix R can be estimated using:

$$R \approx A^T B \tag{30}$$

The rating record r_{ij} is factorized into user and item latent feature vectors by minimizing the following squared loss function:

$$min_{A,B} \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} I_{ij} (r_{ij} - a_i^T b_j)^2 + \frac{\lambda_A}{2} || A||_F^2 + \frac{\lambda_B}{2} || B||_F^2 \tag{31}$$

Since most inputs in R are missing due to the sparsity, the indicator function I_{ij} takes 1 if the user i has evaluated the item j and 0 otherwise. The regularization terms ($\lambda_A = \frac{\sigma_A^2}{\sigma^2}, \lambda_B = \frac{\sigma_B^2}{\sigma^2}$) are incorporated into the objective function to make a trade-off between overfitting and the variance σ in approximating available entries. The optimum minimization with respect to the Frobenius norm $|| \, . \, ||_F$ can be obtained by employing gradient descent in A and B.

Therefore, the low-rank linear approximation of the KFDBN-based imputed rating matrix $ImpR$ can be estimated by solving the following optimization:

$$min_{A,B} \triangle_\omega + \triangle_\psi + \frac{\lambda_A}{2} || A||_F^2 + \frac{\lambda_B}{2} || B||_F^2 \tag{32}$$

Where $\triangle_\omega = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} I_\omega(i, j)(r_{ij} - a_i^T b_j)^2$ and $\triangle_\psi = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} I_\psi(i, j)(r_{ij}^* - a_i^T b_j)^2$. $I_\omega(i, j)$ is the indicator function that takes 1 if the entry r_{ij} is available and 0 otherwise. $I_\psi(i, j)$ is the observed indicator which equals to 1 if the imputed rating r_{ij}^* exists and 0 otherwise. ω and ψ are respectively the sets of known entries and KFDBN-based imputed ratings. Combining $\triangle_\omega$ and $\triangle_\psi$, the aim is to minimize the following equation:

$$min_{A,B} \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} I(i, j)(\widehat{r_{ij}} - a_i^T b_j)^2 + \frac{\lambda_A}{2} || A||_F^2 + \frac{\lambda_B}{2} || B||_F^2 \tag{33}$$

Where $\widehat{r_{ij}} = \begin{cases} r_{ij} & (i, j) \in \omega \\ r_{ij}^* & (i, j) \in \psi \end{cases}$. Updating latent matrices of $ImpR$ for users and items is performed as follows:

$$a_i = a_i + \alpha_A (\iota_{ij} b_j - \lambda_A a_i) \tag{34a}$$

$$b_j = b_j + \alpha_B (\iota_{ij} a_i - \lambda_B b_j) \tag{34b}$$

Where α_A, α_B are the learning rates and $\iota_{ij} = \widehat{r_{ij}} - a_i^T b_j$ is the prediction error.

Finally, based on the extracted correlation matrices, we generate the Top-k item recommendations for users. The proposed KFDBN-based imputation approach is particularly useful in solving the sparsity hurdle in SVD while leveraging its advantages from complexity reduction to performance improvement [30].

5 Experiments and evaluation

To fully evaluate the effectiveness of the proposed KFDBN model, we perform extensive experiments on two main recommendation tasks: missing rating prediction and Top-k ranking recommendation. We first present each task's experimental setup, including evaluation

Table 2 Description of evaluation datasets

Datasets	#Users	#Items	#Ratings	Sparsity
MovieLens 100K	943	1 682	100 000	93.7%
MovieLens 1M	6 040	3 706	1 000 209	95.55%
FilmTrust	1 508	2 071	35 497	98.86%
Amazon Digital Music	5 541	3 568	64 706	99.67%
Amazon Automotive	2 928	1 835	20 473	99.62%
Amazon Instant Video	5 130	1 685	37 126	99.57%

metrics, baseline approaches, and parameter settings. Next, we analyze and discuss the experimental findings, aiming to answer the following Research Questions (RQs):

1. **RQ1.** What is the performance of the RBM model in rating prediction?
2. **RQ2.** Does going deeper using the RBM leads to better accuracy?
3. **RQ3.** Does the FDBN outperform the state-of-the-art approaches in missing rating prediction?
4. **RQ4.** Do KFDBN features yield better accuracy in the KFDBN classifier?
5. **RQ5.** What is the impact of kernel Dimensionality Reduction on KFDBN classifier performance?
6. **RQ6.** Does the KFDBN-based imputation technique outperform the state-of-the-art approaches in the Top-k recommendation?

The experiments are conducted using six real datasets resulting from various domain fields and characterized by different sparsity levels. Table 2 summarizes the statistics of each data set. One may note that experiments are performed both on large-scale and small datasets to examine the effectiveness of the proposed approach in different data sizes. The MovieLens dataset, which was collected by GroupLens,[1] over several periods, describes rating transactions from the MovieLens website. FilmTrust[2] was crawled from a website where users provide feedback ratings on recommended movies. Amazon Digital Music, Automotive, and Instant Video reflect users' interests for items of several categories and industries in the Amazon platform.[3] The sparsity reported in Table 2 is computed using the following formula: $1 - \frac{N_R}{N_U \times N_I}$. Where N_R, N_U, and N_I are the number of ratings, users, and items.

We investigate the sensitivity of the chief parameters and settings of the proposed model and benchmark approaches for each recommendation task. Table 3 presents these hyper-parameters, their definitions, and their impact on the performance of solutions.

5.1 Evaluation for missing rating prediction

5.1.1 Experiment settings

Evaluation metrics We use two widely employed metrics, Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), to calculate the average error in the rating

[1] https://grouplens.org/datasets/movielens/
[2] https://guoguibing.github.io/librec/datasets.html
[3] http://jmcauley.ucsd.edu/data/amazon/

Table 3 Hyper-parameters of the proposed KFDBN and benchmark methods

Symbol	Parameter	Definition	Impact
α	Learning rate	The step size that controls the amount of weight updates.	Small α = slower convergence. High α = Miss the local minima
B_S	Batch size/ mini batch	The number of training samples propagated through the network to calculate the gradient error before updating the parameters.	Has an influence on the convergence speed, the stability, and the generalization performance of the learning process
D	Latent dimension	Number of hidden factors / latent features.	Influences the output and the capacity of the proposed approach. Small D may result in underfitting. High D increases the training time and leads to overfitting
E	Epochs	The number of complete forward and backward passes through the network.	Underfitting occurs when using only one epoch. High number of epochs can lead to overfitting
d	Dropout	Ignoring visible and hidden units at random during the training stage.	A regularization approach that prevents the co-adaptation of neurons, which helps to reduce the overfitting
λ	Regularization coefficient	Controls the extent of regularization to enhance the generalization and avoid the overfitting.	A well-defined λ meaningfully reduces the variance of the model without losing any valuable data. Ill-defined λ causes under-fitting, where significant properties in the data are lost
W_D	Weight decay	A L2 regularisation method that causes the weights to decrease to zero.	A well-chosen size improves the generalization of the network and helps in preventing overfitting
m_o	Momentum	Like α, it also controls the weight updates by increasing the size of the steps.	Small momentum = slows down the training and does not avoid local minima. Large momentum = faster convergence

predictions. MAE and RMSE indicate, as presented in (35) and (36), the extent to which the proposed model can generate relevant predictions close to the ratings provided by users.

$$MAE = \frac{1}{\mid N \mid} \sum_{n_k \in N} \mid \widehat{R}(n_k) - R(n_k) \mid \tag{35}$$

Due to the linearity of MAE, the average variations are weighted equally, whereas the RMSE measure emphasizes the most substantial deviations by squaring the errors before averaging them.

$$RMSE = \sqrt{\frac{1}{|N|} \sum_{n_k \in N} (\widehat{R}(n_k) - R(n_k))^2} \tag{36}$$

Where N, $\widehat{R}(n_k)$, and $R(n_k)$ are respectively the number of assessed items, the predicted rating and the ground truth rating.

Baselines To assess the performance of the proposed generative model in missing rating prediction, we compare it with the following state-the-art methods on different recommendation domains:

- **Non-Negative Matrix factorization (NMF)** [12]: MF method with a non-negative probabilistic constraint to select similar user groups and explain CF recommendations.
- **Neural Collaborative Filtering (NCF)** [11]: A state-of-the-art NN-based approach that models hidden user-item interaction using a standard MLP.
- **SVD++** [26]: SVD extension that integrates implicit records into the recommendation model to uncover latent factors that characterize dense ratings.
- **A hybrid Deep Learning-based Recommender System (DNNREC)** [24]: DNN-based approach that learns non-linear latent features using embedding and auxiliary information about users and items.
- **Joint Representation Learning with Ratings and Reviews (JRLRR)** [43]: A deep hybrid method that fuses rating embedding and textual item features based on Gated Recurrent Unit and attention mechanisms for prediction.
- **Multi-Objective Evolutionary Algorithm (MOEA)** [32]: A multi-objective method to find the appropriate set of similar neighbors for an active user in CF recommendation.

Parameter settings To train the RBM, we use m_o of 0.1 for the six datasets. We set α as 0.0001 for Digital Music (DG), Instant Video (AIV), and automotive (Auto), while α is set as 0.0005 for both MovieLens 100K (ml-100k) and MovieLens 1M (ml-1m), and as 0.09 for Filmtrust data set. The B_S of 32 performed well on all datasets. Several experiments are set up to examine the hyper-parameters validity and select their optimal values for the FDBN model. Based on the validation outcomes, D is set to 2000 latent features for the first hidden layer and 1000 latent features for the second hidden layer, while the number of propagated training samples B_S is set as 32 on all datasets. α is set as 0.9 on all datasets except AIV, where $\alpha = 0.2$ shows better performance. With unsupervised pre-training, m_o is set as 0.1 on all datasets to prevent the model from getting stuck in local minima. For benchmark models, the parameters are determined from the proclaimed ones and by a grid-search, which optimizes the approaches to determine the hyper-parameters that result in the most precise recommendation. The optimal values for each method according to recommendation domains and datasets are reported in Tables 4 and 5.

5.1.2 The performance of the RBM model (RQ1)

Figures 5 and 6 show the sensitivity of the dimension of hidden features on the accuracy of the RBM model. The RMSE results of the RBM model are presented in Fig. 5, while the MAE results are reported in Fig. 6. The best accuracy on all datasets is achieved when setting D to 5000. By increasing the dimension of latent features, RBM leads to better performance, which indicates that the model can extract more reliable information.

Figure 7 depicts the time complexity of all datasets in both the training and testing phases. It has been observed that the use of a smaller batch size leads to better accuracy results but slows down the convergence of the learning process. Likewise, a lower learning rate allows the model to find a more optimal set of weights but takes longer to train.

Table 4 Hyper-parameters of benchmark models in movie recommendation domain: Rating Prediction task

Model	Hyper-parameters	MovieLens 100K	MovieLens 1M	Filmtrust
DNNREC	α	$1e-2, 1e-5$	$1e-2, 1e-5$	$1e-2, 1e-5$
	B_S	65	65	65
	d	0.25	0.25	0.25
	W_D	$1e-4$	$1e-4$	$1e-4$
NMF	α	0.007	0.007	0.007
	D	10	10	2
	E	100	100	100
	λ	0.1	0.1	0.1
SVD++	α	0.007	0.007	0.0014
	D	10	10	2
	E	100	100	100
	d	0.2	0.2	0.2
	λ	0.1	0.1	0.1
NCF	B_S	256	256	256
	D	50	50	40
MOEA	E	500	500	500

Table 5 Hyper-parameters of benchmark models in music, video, and automotive recommendation domains: Rating Prediction task

Model	Hyper-parameters	Digital music	Instant video	Automotive
JRLLR	α	[0.0001, 0.001]		
	B_S	100	100	100
	D	50	50	50
	d	[0 , 0.1, 0.2 , 0.5]		
NMF	α	[0.006, 0.005, 0.004, 0.003, 0.002, 0.001]		
	D	[10, 25, 50, 100, 150, 200]		
	E	60	60	60
	λ	[0.001, 0.01, 0.1, 1.0]		
SVD++	D	16	16	16
	d	0.5	0.5	0.5

5.1.3 The performance of the DBN model (RQ2)

To illustrate the performance of the DBN model before and after fine-tuning, Table 6 reports RMSE and MAE results on the six datasets. We can observe that the RBM model outperforms better than DBN before fine-tuning on four out of the six datasets: ml-1m, Filmtrust, AIV, and Auto. While the DBN before fine-tuning outperforms better than RBM on ml-100k and DG. Nevertheless, after fine-tuning the DBN model, its performance leads to more accurate predictive results.

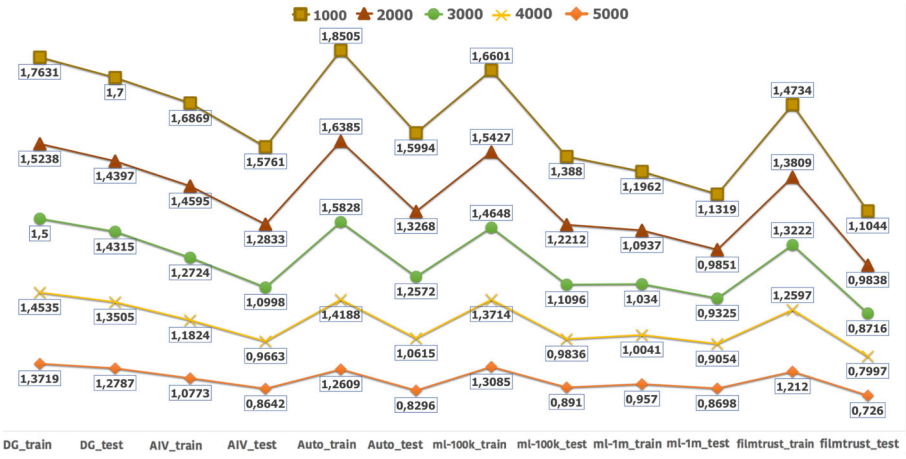


Fig. 5 RMSE Results of the RBM model (the lowest is better) according to the dimension of latent features

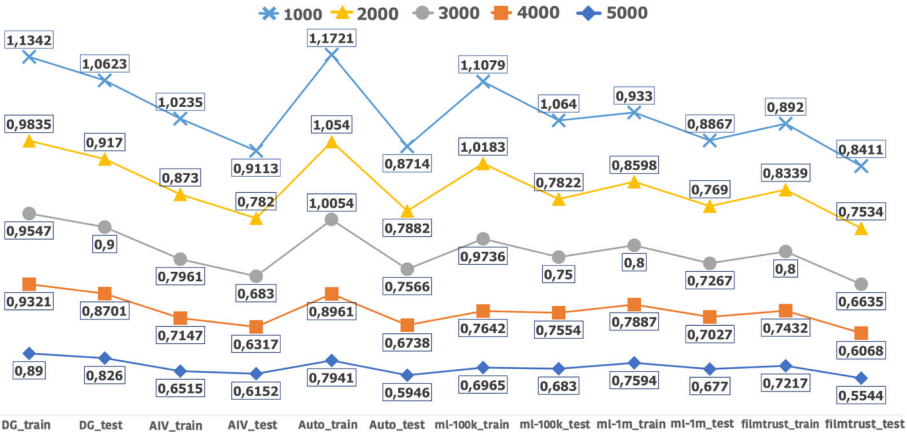


Fig. 6 MAE Results of the RBM model (the lowest is better) according to the dimension of latent features

The best values of RMSE and MAE on all datasets are achieved using the FDBN, which indicates the significance of the supervised fine-tuning phase. Through the BP, we have conducted the fine-tuning procedure to get an optimal model. The goal here is not to discover new latent features but to search locally for relevant parameters using the hidden features. For further accuracy, the optimization method aims at updating the weights, as depicted in Fig. 8. The weights distribution of the RBM-0 is presented using Matplotlib, while the distribution of the weights of the RBM-1 is presented using Tensorboard. These new adjusted weights result from the fine-tuning stage using BP, which leverages the valuable data in the labels. The model's accuracy is hence increased after seeking the optimal values of the weights of the RBMs, which produced the efficient FDBN generative model.

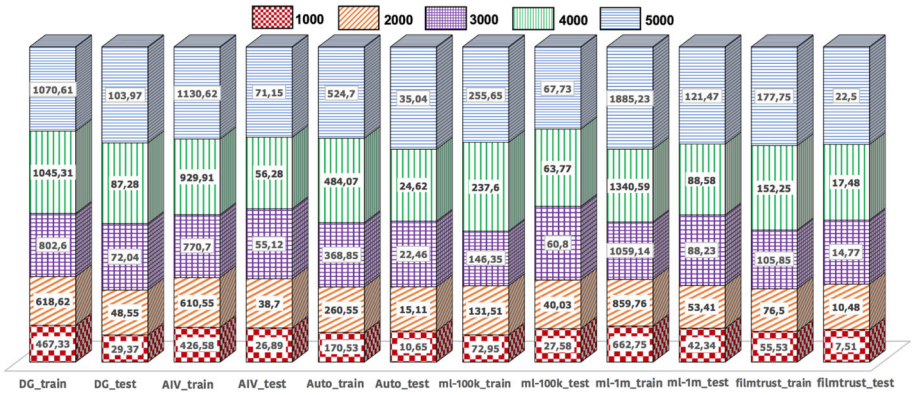


Fig. 7 Time complexity of the RBM model in both training and testing stages in seconds based on the dimension of latent features

Table 6 Comparison of prediction accuracy of the RBM, DBN, and FDBN (fine-tuned DBN)

	RBM		DBN		FDBN	
	RMSE	MAE	RMSE	MAE	RMSE	MAE
MovieLens 100K	0.891	0.683	0.8657	0.667	0.6382	0.4886
MovieLens 1M	0.8698	0.677	0.8848	0.693	0.6266	0.4966
FilmTrust	0.726	0.5544	0.833	0.637	0.6989	0.5301
Amazon Digital Music	1.2787	0.826	0.9045	0.6592	0.6673	0.4916
Amazon Instant Video	0.8642	0.6152	0.8733	0.6343	0.6789	0.5
Amazon Automotive	0.8296	0.5946	0.9027	0.6249	0.681	0.481

5.1.4 Overall performance of the FDBN model in missing rating prediction (RQ3)

We compare the prediction errors of the FDBN and benchmark models using the six datasets. Table 7 presents RMSE and MAE error values in the movie recommendation field, while Table 8 reports RMSEs and MAEs of the methods in the music, video, and automotive recommendation domains.

The FDBN model immensely outperforms all the state-of-the-art approaches using the six real datasets in the various recommendation fields. Specifically, on ml-100k, ml-1m, and Filmtrust, the RMSE values of the FDBN are, respectively, 0.6382, 0.6266, and 0.6989; which are 22.6%, 24.84 %, and 10.61% lower in error compared to the most competitive solution, i.e., DNNREC. The FDBN also outperforms DNNREC in MAE with 17.74%, 19.14 %, and 9.59% on ml-100k, ml-1m, and Filmtrust, respectively. DNNREC employs side information as input to model users' and items' feature vectors, operating different embedding layers. However, the learned latent factors are simply merged and fed to MLP for missing rating prediction. Each feature vector is directly exploited for representation, which can overfit the training data. This fails to ensure an accurate recommendation, especially in cases where the model needs numerous content data to learn significant hidden factors. Nevertheless, the efficient greedy layer-wise training of FDBN not only yields relevant rating predictions without relying on auxiliary or implicit information but also enhances

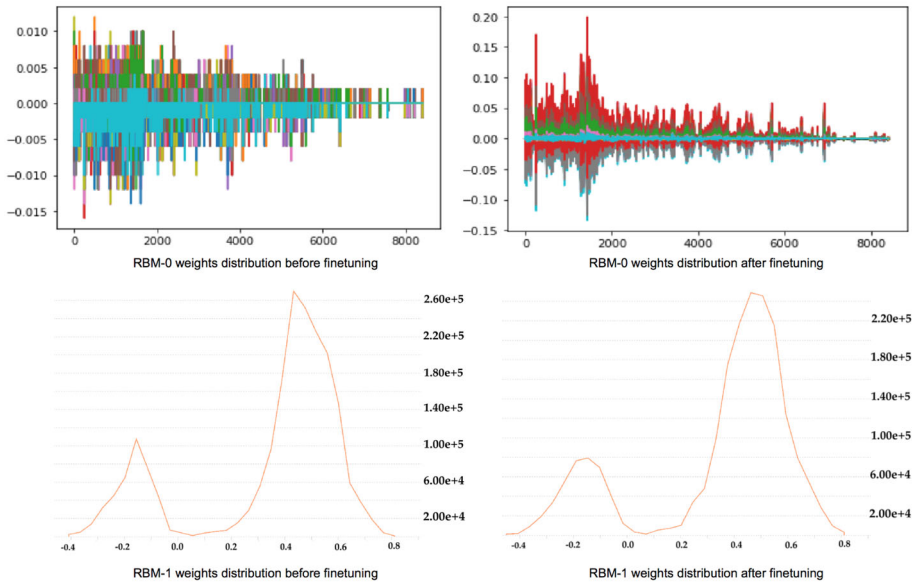


Fig. 8 Weights distribution of the DBN before and after fine-tuning on Movielens 100K

Table 7 FDBN performance compared with benchmark approaches in the movie recommendation field

Model	MovieLens 100K		MovieLens 1M		Filmtrust	
	RMSE	MAE	RMSE	MAE	RMSE	MAE
DNNREC	0.864	0.666	0.875	0.688	0.805	0.626
NMF	0.968	0.752	0.923	0.719	0.8507	0.646
SVD++	0.869	0.671	0.882	0.691	0.825	0.659
NCF	0.9363	0.7437	0.8758	0.6921	0.8743	0.656
MOEA	0.8695	0.6783	0.9254	0.7219	0.8876	0.6924
FDBN	**0.6382**	**0.4886**	**0.6266**	**0.4966**	**0.6989**	**0.5301**
Improvement (%)	$\triangle_{DNNREC}$ 22.6	$\triangle_{DNNREC}$ 17.7	$\triangle_{DNNREC}$ 24.8	$\triangle_{DNNREC}$ 19.1	$\triangle_{DNNREC}$ 10.6	$\triangle_{DNNREC}$ 9.6

the generalization to new data, thereby avoiding the overfitting problem that may occur in DNNREC.

On Amazon datasets, SVD++ and JRLRR show good results but fail to surpass the FDBN in RMSE and MAE. On Amazon DG, the RMSE value of FDBN is 0.6673, which is 22.1% lower than the second-best solution, i.e., JRLRR. The latter relies substantially on auxiliary data, which is not always effortlessly obtained, especially if privacy constraints remain. Nevertheless, the FDBN copes with co-rated items limitation by incorporating all available interactions of users with items in the learning process without relying on side information. On the AIV dataset, the FDBN achieves an RMSE of 0.6789 and outperforms the MF-based SVD++ technique with 26.3% improvement. This demonstrates that the generative FDBN is better than MF-based techniques (NMF, SVD++) that may lose valuable information from

Table 8 FDBN performance compared with benchmark approaches in the music, video, and automotive recommendation fields

Model	Amazon Digital Music		Amazon Instant video		Amazon Automotive	
	RMSE	MAE	RMSE	MAE	RMSE	MAE
JRLRR	0.8883	0.6481	0.9733	0.7188	0.917	0.6319
NMF	0.997	0.7	1.1156	0.8429	1.1002	0.847
SVD++	0.905	0.641	0.9418	0.721	0.896	0.63
FDBN	**0.6673**	**0.4916**	**0.6789**	**0.5**	**0.681**	**0.481**
Improvement	$\triangle_{JRLRR}$	$\triangle_{SVD++}$	$\triangle_{SVD++}$	$\triangle_{JRLRR}$	$\triangle_{SVD++}$	$\triangle_{SVD++}$
(%)	22.1	14.9	26.3	21.9	21.5	14.9

the original matrix. However, the FDBN learns relevant latent factors and valuable hidden features based on probabilistic building blocks, where neural units merely communicate their activation states.

Unlike NCF, which employs simple MLP for feature learning, FDBN leverages a more powerful deep generative model to capture higher-level latent feature interaction in the sparse user-item matrix. The FDBN significantly outperforms neighborhood-based techniques (e.g., MOEA) due to its effective learning process of complex hidden features. MOEA uses Non-dominated Sorting Genetic Algorithm II (NSGA-II) to determine the user's ideal number of similar neighbors. The genetic technique is based on crowded distance comparison that does not consider the distribution of decisions, thus resulting in a biased solution, which tends to fall into a local optimum and poor convergence [39]. However, the FDBN's greedy unsupervised pre-training provides a regularization effect by efficiently initializing chief parameters. This enhances the generalization performance and directs the model's weights toward a good local minimum, indicating why the FDBN vastly exceeds baseline approaches and justifies the best accuracy values in RMSE and MAE errors.

5.2 Evaluation for Top-k ranking recommendation

5.2.1 Experiment settings

Evaluation metrics We use the Hit Rate (HR) and the Normalized Discounted Cumulative Gain (NDCG) metrics to assess the performance of the proposed KFDBN model in providing an effective ranked list of Top-k items. $HR@k$ examines the presence of the test item in the recommended Top-k list, while the $NDCG@k$ evaluates the ranking's quality, taking into account the position of the hits. The HR and the $NDCG$, according to k, the truncated number, are described by (37) and (38).

$$HR@k = \frac{N\_hits@k}{N\_users} \tag{37}$$

where $N\_hits@k$ is the total number of users for which the exact test item appears in the top-k recommendation and N_users denotes the number of users in the test set.

$$NDCG@k = \frac{1}{IDCG@k} \times \sum_{i=1}^{k} \frac{2^{rel(i)} - 1}{log_2(i+1)} \tag{38}$$

NDCG reduces the scores to hits by log_2 at lower positions and increases those at top ranks; $rel(i)$ is the ith rating.

$$IDCG@k = \sum_{i}^{|REL_k|} \frac{2^{rel(i)} - 1}{log_2(i+1)} \tag{39}$$

To provide a normalization factor, the IDCG (Ideal Discounted Cumulative Gain) determines the Discounted Cumulative Gain (DCG) score for the ideal ranking ordered by relevance; $| REL_k |$ is the sorted ideal list of relevant items.

Since the KFDBN-based imputation technique proposed for the Top-k recommendation is a hybrid approach that relies on the results of the KFDBN classifier, we define in (40) the accuracy metric to evaluate the performance of the classification model.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{40}$$

The accuracy measures the proportion of right predictions by the classifier from the total number of individuals in the data set. TP (True Positive) presents an accurate positive prediction, where the prediction is positive, and the real value is indeed positive. TN (True Negative) is the case of a correct negative prediction, where the real value is negative, and the prediction is also negative. While False Positive (FP) indicates an incorrect prediction, where the prediction is positive, but the actual value is negative. FN (False Negative) is the case of a wrong negative prediction, where the prediction is negative, but the true value is positive.

Baselines We compare the KFDBN with the following competitive approaches on the Top-k ranking task:

- **Bayesian Personalised Ranking (BPR)** [34]: MF-based approach that exploits pairwise learning for personalized ranking.
- **Deep Matrix Factorization (DMF)** [47]: A DL-based approach that maps non-linear projections of users and items into a common low-dimensional feature space with a binary-based cross-entropy loss function.
- **Neural Matrix Factorization (NeuMF)** [11]: A model that fuses generalized matrix factorization (GMF) and MLP. GMF allows the application of a linear kernel that learns latent low-rank features, while the MLP employs a non-linear kernel to extract latent feature interactions.
- **Deep Collaborative Filtering (DeepCF)** [8]: It proposes a representation learning based on the fusion of CF models, including the Collaborative Filtering Network (CFNet), for complex matching function modeling while retaining the capability to learn low-rank user-item interactions effectively.
- **Neural Graph Collaborative Filtering (NGCF)** [41]: It exploits the high-order connectivity in the bipartite graph of user-item transactions to inject a collaborative signal into an embedding propagation layer.

Parameter settings For baseline approaches, the optimal values of hyper-parameters on the Top-k recommendation task are reported in Tables 9 and 10.

5.2.2 Impact of KFDBN features on the classification performance (RQ4)

We perform the item relevancy classification using three different types of SVC kernels (RBF, Sigmoid, and Polynomial) with a regularization parameter $C = 1$. To examine the

Table 9 Hyper-parameters of benchmark models in movie recommendation domain: Top-k recommendation task

Model	Hyper-parameters	MovieLens 100K	MovieLens 1M	Filmtrust
BPR	α	[0.5, 0.1, 0.01, 0.001, 0.0001]		
	d	[0, 0.3, 0.5, 0.7]		
NeuMF	B_S	256	256	256
	D	50	50	40
DMF	α	0.0001	0.0001	0.0001
	B_S	256	256	256
	D	128	128	128
DeepCF	α	[0.001, 0.005, 0.0001, 0.0005]		
	B_S	256	256	256
	D	128	128	128
	E	20	20	20

Table 10 Hyper-parameters of benchmark models in music, video, and automotive recommendation domains: Top-k recommendation task

Model	Hyper-parameters	Digital music	Instant video	Automotive
BPR	α	[0.001, 0.005, 0.01, 0.05, 0.1]		
	B_S	[8, 16, 32, 64]		
	D	500	100	200
	λ	[0.00001, 0.0001, 0.001, 0.01]		
NeuMF	D	64	64	64
	E	8	8	8
DMF	D	64	50	40
	λ	[0.0001, 0.001, 0.01, 0.1, 1, 10]		
NGCF	α	[0.025, 0.020, 0.015, 0.010]		
	D	64	64	64
	d	[0.0, 0.1, $\cdots$, 0.8]		

impact of KFDBN output features on the model performance, we conduct the classification on ml-100k and ml-1m datasets using: (1) Rating features such as the average rating and ratings count combined with item features such as the year and genre. (2) Rating features combined with KFDBN features. (3) Rating features combined with item features and KFDBN features. Classification results in the three cases using the different SVC kernels are reported in Fig. 9.

The obtained classification outcomes show that the RBF kernel outperformed other kernel functions in SVC for all datasets. In comparison, the Sigmoid function provided the lowest accuracies. With a classification accuracy of 97.62%, the SVC with RBF kernel on ml-100k rating features combined with KFDBN features achieved the best performance. It is significant to highlight that combining KFDBN features with rating information led to better accuracy using all SVC kernels on ml-100k, followed by cases where KFDBN features are merged with items and rating features for item relevancy classification. On

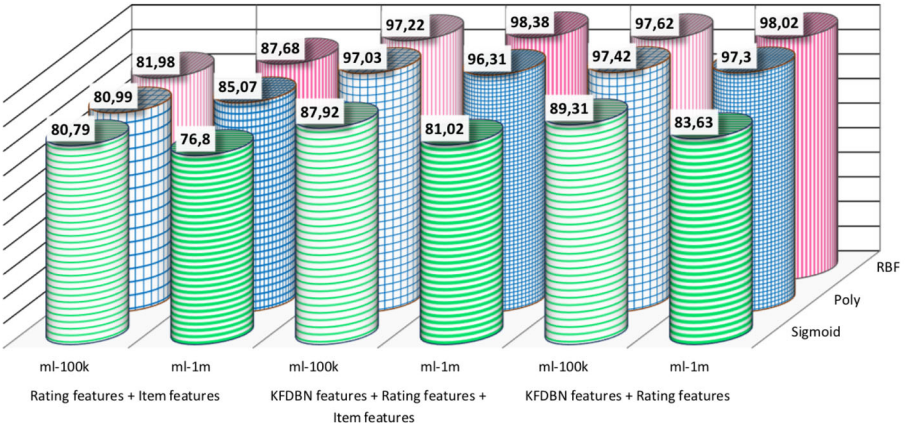


Fig. 9 Classification Accuracy with and without KFDBN features by using different SVC kernels

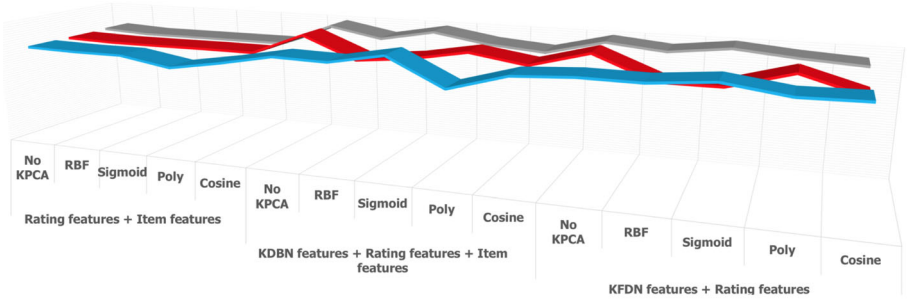
the other hand, on the ml-1m dataset, the SVC with RBF kernel yielded the highest accuracy of 98.38% when applied on KFDBN features merged with ratings and item features. Moreover, KFDBN features improved the classification results of SVC with other kernels, especially when they are incorporated solely with rating information. However, when combining only item features with rating factors, the classification showed inferior results for both datasets. When PCA was not applied to UP-based Items profiles, the classification accuracy decreased by 17% for ml-100k and 16% for ml-1m. This further proves the significance of the KFDBN model in extracting meaningful latent features that can efficiently train the SVC for a classification problem.

5.2.3 Influence of kernel dimensionality reduction in KFDBN classification (RQ5)

We show the impact of the KPCA employed in the KFDBN item relevancy classifier by comparing the classification results after applying the kernel PC with SVC accuracies, reported in Fig. 9. Classification outcomes before and after the application of KPCA on ml-100k and ml-1m datasets are presented in Figs. 10 and 11, respectively. We have applied KPCA on rating features, item characteristics, and KFDBN latent factors used by the SVC model. We use four kernels for the examination (RBF, Sigmoid, Polynomial, and Cosine).

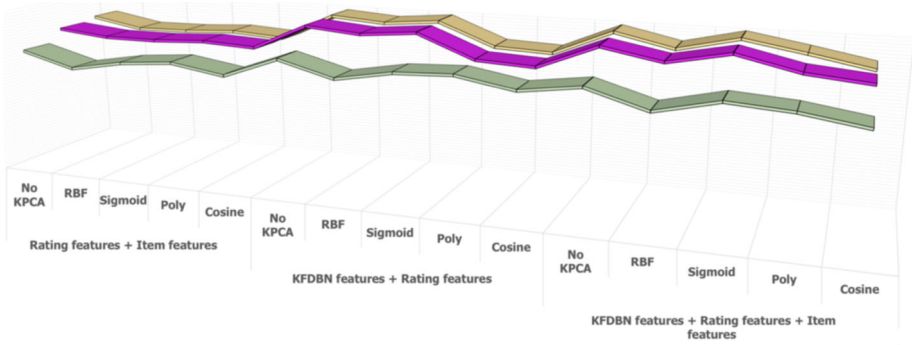
For ml-100k, the best accuracy of 98.61% is obtained when applying Sigmoid PCA on rating features combined with KFDBN latent factors used to train the SVC classifier with RBF kernel. Similarly, for ml-1m, the application of Sigmoid PCA on rating features merged with KFDBN and item factors achieved the highest accuracy of 98, 83%.

Note that, when KFDBN features are incorporated with training factors, using a Sigmoid PCA leads to higher accuracy of SVC with RBF kernel outperforming, thus other classification results. On the contrary, KPCA with other kernels hampers the performance of SVC with all kernels. Otherwise, when only ratings and item factors are included, reducing the dimensionality of features using KPCA leads to a relatively lower accuracy on SVC based on all kernels for all datasets. This proves the efficiency of Sigmoid PCA in extracting relevant PC of latent features obtained by the KFDBN model, thus enhancing the semi-supervised classification results. Nevertheless, leveraging RBF, Polynomial, and



	Rating features + Item features					KDBN features + Rating features + Item features					KFDN features + Rating features				
	No KPCA	RBF	Sigmoid	Poly	Cosine	No KPCA	RBF	Sigmoid	Poly	Cosine	No KPCA	RBF	Sigmoid	Poly	Cosine
Sigmoid SVC	80,79	79,8	79,8	72,87	79,8	87,92	86,93	95,05	73,86	86,73	89,31	88,91	93,07	86,93	87,72
Poly SVC	80,99	79,8	79,8	80	79,8	97,03	79,8	84,16	92,08	85,35	97,42	79,8	79,8	92,87	79,8
RBF SVC	81,98	79,8	79,8	79,8	79,8	97,22	89,31	97,42	90,3	86,73	97,62	93,27	98,61	94,45	91,68

Fig. 10 Classification accuracy of SVC with and without the application of KPCA on trained features: ml-100k dataset



	Rating features + Item features					KFDBN features + Rating features					KFDBN features + Rating features + Item features				
	No KPCA	RBF	Sigmoid	Poly	Cosine	No KPCA	RBF	Sigmoid	Poly	Cosine	No KPCA	RBF	Sigmoid	Poly	Cosine
Sigmoid SVC	76,8	68,07	73,83	76,5	71,31	83,63	74,73	80,75	82,19	76,26	81,02	70,86	78,96	76,52	71,22
Poly SVC	85,07	81,74	81,74	83,36	81,74	97,3	94,33	97,3	83,81	82,1	96,31	89,84	96,94	88,94	85,7
RBF SVC	87,68	81,74	82,1	83,81	81,74	98,02	95,5	98,47	85,07	83,63	98,38	90,74	98,83	92,8	86,15

Fig. 11 Classification accuracy of SVC with and without the application of KPCA on trained features : ml-1m dataset

Cosine kernels in PCA adversely affects the performance of SVC classification by changing features drastically due to the ignorance of some dimensions.

5.2.4 Overall performance of the KFDBN model in the Top-k ranking recommendation (RQ6)

The HR and NDCG values of the proposed KFDBN model and the baseline methods on the six datasets are summarized in Tables 11 and 12. The highest score in each column is bold, and the underlined value is the second-best, while $\triangle$ represents KFDBN's relative improvement over the best competitive solution. The experimental results demonstrate that the hybrid KFDBN model achieves the most promising ranking performance on all datasets, meaningfully outperforming baseline methods, including DMF, by the largest margin of 17% in $NDCG@10$ on ml-100k. We believe this is because DMF learns the latent space to capture users' transactions on items with the inner product directly on the sparse rating

Table 11 KFDBN ranking performance compared with benchmark approaches in the movie recommendation field

Model	MovieLens 100K		MovieLens 1M		Filmtrust	
	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10
BPR	0.6914	0.3933	0.7162	0.4628	0.868	0.7632
NeuMF	0.6886	0.4008	0.7045	0.4201	0.9171	0.8067
DMF	0.6797	0.405	0.6565	0.415	0.9071	0.7896
DeepCF	0.6819	0.3981	0.7253	0.4416	0.9158	0.8074
KFDBN	**0.7808**	**0.5750**	**0.7440**	**0.5939**	**0.9393**	**0.8204**
Improvement	$\triangle_{BPR}$	$\triangle_{DMF}$	$\triangle_{DeepCF}$	$\triangle_{BPR}$	$\triangle_{NeuMF}$	$\triangle_{DeepCF}$
(%)	8.94	17	1.87	13.11	2.22	1.3

Table 12 KFDBN ranking performance compared with benchmark approaches in the music, video, and automotive recommendation fields

Model	Amazon Digital Music		Amazon Instant video		Amazon Automotive	
	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10
BPR	0.4742	0.2045	0.5904	0.3949	0.4607	0.2992
NeuMF	0.5322	0.2643	0.5821	0.3876	0.4501	0.2891
DMF	0.3201	0.1462	0.42	0.325	0.13	0.06
NGCF	0.4971	0.2575	0.375	0.1916	0.2283	0.1191
KFDBN	**0.5625**	**0.3390**	**0.6571**	**0.4458**	**0.5454**	**0.3459**
Improvement	$\triangle_{NeuMF}$	$\triangle_{NeuMF}$	$\triangle_{BPR}$	$\triangle_{BPR}$	$\triangle_{BPR}$	$\triangle_{BPR}$
(%)	3.03	7.47	6.67	5.09	8.47	4.67

matrix, which is inadequate for complex real-world data. In contrast, using a generated denser rating matrix, the KFDBN hybrid model effectively determines the item's relevance for personalized ranking. Not to mention that the cosine measure employed by DMF may not incorporate the varied rating scales assessed by a particular user [17], further hindering its efficiency in the Top-k recommendation.

On the ml-1m dataset, the $HR@10$ value of KFDBN is 0.7440 and surpasses the second-best approach, i.e., DeepCF (0.7253). The latter merges DMF and MLP for enhanced user-item matching score prediction and reaches the closest results to NeuMF. Nevertheless, despite their efficiency, NeuMF and DeepCF models show inferior results compared to the proposed KFDBN model. In fact, two reasons that hinder the performance are the use of dual embedding spaces in NeuMF that may lead to overfitting and the reliance on simple MLP for feature extraction in DeepCF. However, the KFDBN leverages more powerful DL-based feature interaction learning rather than using MLP as a mapping function to capture complex low-rank features. Furthermore, KFDBN employs a greedy unsupervised pre-training phase to effectively initialize the models' parameters, which enhances the regularization effect, addresses the gaps in training state-of-the-art DNN approaches, and makes the proposed model less prone to overfitting.

Otherwise, one major limitation with graph-based models (e.g., NGCF) is the potential of exploring the bipartite graph structure of irrelevant user-item transactions, which encodes a poor collaborative signal in the embedding function. Such a limitation explains why KFDBN largely outperforms NGCF in item ranking performance and demonstrates that imputing a denser rating matrix using KFDBN is more accurate in user interest detection than propagating embeddings on the user-item bipartite graph structure.

Besides, DL-based models do not always outperform non-deep methods. Regarding HR and NDCG, BPR performs well on AIV and Auto datasets. However, on AIV, the $HR@10$ and $NDCG@10$ values of KFDBN are 0.6571 and 0.4458, which are 6.67% and 5.09% higher than the second-best solution, i.e., BPR. On Auto data set, the KFDBN surpasses the competitive solution, BPR, with 8.47% and 4.67% in $HR@10$ and $NDCG@10$, respectively. This demonstrates that KFDBN hybridization exceeds the performance of bayesian ranking in capturing complex correlations of users and items.

Moreover, instead of relying on the standard uniform negative sampler used in BPR to incorporate unavailable interactions as crucial indicators of users' potential negative entries and suppose unobserved items as negative instances, KFDBN aims for reliable imputation of unobserved ratings in the original sparse matrix and then applies MF on the denser generated matrix to learn significant latent features of users and items, yielding highly encouraging results and justifying KFDBN's meaningful outperformance over baselines in the Top-k recommendation.

5.3 Implications of findings

The proposed model fully exploits DBN's advantages to present a deep structure learning architecture for discovering multiple and high-level feature representations for users and items. The unsupervised pre-training tackles traditional random weight initialization, which may draw the model to locally optimal solutions or increase training complexity. The KFDBN results in better generalization by providing a greedy layer-by-layer learning procedure of reliable weights. This stage establishes a starting point for further supervised fine-tuning by placing the model's parameters in an appropriate restricted range. Leveraging such a semi-supervised synergy yields a regularizing impact that renders the learning strategy more efficient, especially in sparse conditions, which is evident in conducted extensive experimentations that proved the significant improvement of FDBN in terms of RMSE and MAE.

Unlike traditional classification-based CF approaches, which merely employ shallow feature extraction [40, 44], the proposed KFDBN generates deep and meaningful latent representations for users and items, leading to higher SVC-based classification outcomes. The unsupervised procedure of KFDBN indirectly affects the classifier's performance by providing a reliable initialization of weights; then, the supervised fine-tuning phase helps further optimize the model by searching locally for optimal parameters. Therefore, the SVC can benefit from these stages by adopting output KFDBN features to conduct an efficient semi-supervised classification. Moreover, experimental results proved that KFDBN features could help enhance the accuracy of supervised classifiers, especially if Sigmoid KPCA is further leveraged. Therefore, the sigmoid extraction of features learned by the KFDBN model is a promising new approach that consistently yields an accurate and practical recommendation based on semi-supervised classification.

Some state-of-the-art imputation-based techniques replace missing ratings with plausible values. The column mean or the row mean of the rating matrix may, for instance, be substituted for an empty entry [40]. Such an imputed value is not derived from users'

interests, which alters the data's covariance structure and thus impeding the system performance. On the other hand, several ranking-based benchmark methods [8, 11, 34] imply a significant modification of the experimental evaluation by treating unobserved ratings as negative instances, thus making up for the skewed distribution of known interaction. Nevertheless, the proposed model exploits a more powerful architecture for imputation to infer unavailable ratings in the original sparse matrix and thus overcome the hindrance of missing values on MF-based ranking performance. We showed that by imploying SVD on a denser matrix imputed using KFDBN, the model captured more significant latent features yielding promising prediction results for the Top-k recommendation. The proposed unified model exploits local and global correlations in the data for imputation. An MF technique can directly be leveraged to make relevant predictions using a reduced rating matrix, making this hybrid approach appropriate for high dimensional and heterogeneous datasets while exhibiting more meaningful accuracy than traditional imputation-based techniques.

One of the significant issues that sparsity is imposing is the inability to generate reliable predictions for long-tail items with only a few ratings [27]. This challenge occurs when the recommendation approach depends on a single latent representation to exploit the ratings concentrated on a few popular items. However, the proposed imputation approach can provide users with personalized and accurate recommendations from the items available in the long tail using hybridization of deep and kernel-based methods to learn latent features of users and items stemming from various aspects. These latent factors comprehensively reflect users' interests and items' characteristics for effective missing rating prediction, resulting in better novelty and diversity [17].

6 Conclusion

The proposed KFDBN first leans a deep generative model by leveraging greedy learning layer-by-layer procedure for efficient feature extraction and missing rating prediction. Then, it creates input vectors for kernel-based discriminative support vector classification, performed in two stages to explore the impact of KFDBN output features on kernel-based classification. This work intensively investigates the influence of kernel-based feature extraction on the system's performance. Experiments prove that further Sigmoid extraction applied to output KFDBN features using Kernel Principal Component Analysis yields accurate and practical recommendations based on semi-supervised classification. Moreover, the positive predictions derived from the KFDBN's generative model are combined with ratings of relevant items resulting from the KFDBN kernel-based classifier to impute a denser user-item interaction matrix for effective Matrix Factorization-based Top-k recommendation. Empirical evaluations on six datasets with varying sparsity levels demonstrate that the KFDBN achieves higher accuracy outcomes and outperforms the state-of-the-art models. Future perspectives include extending the KFDBN to meta-learning by inferring powerful generalization from meta rating interactions to model the preference learning for cold-start users and items.

Data Availability The datasets that support the findings of the current study are available on the repositories named MovieLens, LibRec, and Amazon Review data (URLs are https://grouplens.org/datasets/movielens/, https://guoguibing.github.io/librec/datasets.html, and https://jmcauley.ucsd.edu/data/amazon/), respectively.

Declarations

Conflict of interest The authors declare that they have no conflict of interest/competing interests. The authors did not receive any specific grant from any organization for the submitted work.

References

1. Ahmadian S, Joorabloo N, Jalili M, Ahmadian M (2022) Alleviating data sparsity problem in time-aware recommender systems using a reliable rating profile enrichment approach. Expert Syst Appl 187:115849. https://doi.org/10.1016/j.eswa.2021.115849
2. Amari SI, Murata N, Muller KR, Finke M, Yang HH (1997) Asymptotic statistical theory of overtraining and cross-validation. IEEE Trans Neural Netw 8:985–996
3. Bathla G, Aggarwal H, Rani R (2020) Autotrustrec: recommender system with social trust and deep learning using autoencoder. Multimed Tools Appl 79(29):20845–20860. https://doi.org/10.1007/s11042-020-08932-4
4. Bell RM, Koren Y, Volinsky C (2010) All together now: a perspective on the netflix prize. Chance 23(1):24–29. https://doi.org/10.1080/09332480.2010.10739787
5. Cervantes J, Garcia-Lamont F, Rodríguez-Mazahua L, Lopez A (2020) A comprehensive survey on support vector machine classification: Applications, challenges and trends. Neurocomputing 408(2):189–215. https://doi.org/10.1016/j.neucom.2019.10.118
6. Chae DK, Kang JS, Kim SW, Choi J (2019) Rating augmentation with generative adversarial networks towards accurate collaborative filtering. In: The World Wide Web conference, pp 2616–2622. https://doi.org/10.1145/3308558.3313413
7. Chen Z, Zhao W, Wang S (2021) Kernel meets recommender systems: a multi-kernel interpolation for matrix completion. Expert Syst Appl 168:114436. https://doi.org/10.1016/j.eswa.2020.114436
8. Deng ZH, Huang L, Wang CD, Lai JH, Philip SY (2019) Deepcf: a unified framework of representation learning and matching function learning in recommender system. In: Proceedings of the AAAI conference on artificial intelligence, vol 33, pp 61–68. https://doi.org/10.1609/aaai.v33i01.330161
9. Ding R, Guo G, Yan X, Chen B, Liu Z, He X (2020) Bigan: collaborative filtering with bidirectional generative adversarial networks. In: Proceedings of the 2020 SIAM international conference on data mining, pp 82–90. https://doi.org/10.1137/1.9781611976236.10
10. Fernandez-Delgado M, Cernadas E, Barro S, Amorim D (2014) Do we need hundreds of classifiers to solve real world classification problems. J Mach Learn Res 15:13133–3181
11. He X, Liao L, Zhang H, Nie L, Hu X, Chua TS (2017) Neural collaborative filtering. In: Proceedings of the 26th international conference on World Wide Web, pp 173–182. https://doi.org/10.1145/3038912.3052569
12. Hernando A, Bobadilla J, Ortega F (2016) A non negative matrix factorization for collaborative filtering recommender systems based on a bayesian probabilistic model. Knowl-Based Syst 97:188–202. https://doi.org/10.1016/j.knosys.2015.12.018
13. Hinton GE (2012) A practical guide to training restricted boltzmann machines. In: Montavon G, Orr GB, Müller K (eds) Neural networks: tricks of the trade, 2nd edn. Springer, Berlin, pp 599–619. https://doi.org/10.1007/978-3-642-35289-8_32
14. Hinton GE, Osindero S, Teh YW (2006) A fast learning algorithm for deep belief nets. Neural Comput 18(7):1527–1554. https://doi.org/10.1162/neco.2006.18.7.1527
15. Hsu CW, Lin CJ (2002) A comparison of methods for multiclass support vector machines. IEEE Trans Neural Netw 13(2):415–425. https://doi.org/10.1109/72.991427
16. Huang J, Sun Y, Zhang J (2021) Reduction of computational error by optimizing svr kernel coefficients to simulate concrete compressive strength through the use of a human learning optimization algorithm. Eng Comput 1–18. https://doi.org/10.1007/s00366-021-01305-x
17. Idrissi N, Zellou A (2020) A systematic literature review of sparsity issues in recommender systems. Soc Netw Anal Min 10(15). https://doi.org/10.1007/s13278-020-0626-2
18. Idrissi N, Zellou A, Hourrane O, Bakkoury Z (2019) Addressing cold start challenges in recommender systems: Towards a new hybrid approach. In: International conference on smart applications, communications and networking (smartnets), pp 1–6. https://doi.org/10.1109/SmartNets48225.2019.9069801
19. Idrissi N, Hourrane O, Zellou A (2019) A restricted boltzmann machine-based recommender system for alleviating sparsity issues. In: 2019 1st International conference on smart systems and data science (ICSSD), pp 1–5. https://doi.org/10.1109/ICSSD47982.2019.9003149

20. Idrissi N, Zellou A, Hourrane O, Bakkoury Z, Benlahmar EH (2019) A new hybrid-enhanced recommender system for mitigating cold start issues. In: Proceedings of the 2019 11th international conference on information management and engineering, pp 10–14. https://doi.org/10.1145/3373744.3373746
21. Iqbal M, Ghazanfar MA, Sattar A, Maqsood M, Khan S, Mehmood I, Baik SW (2019) Kernel context recommender system (kcr): a scalable context-aware recommender system algorithm. IEEE Access 7:24719–24737. https://doi.org/10.1109/ACCESS.2019.2897003
22. Katarya R, Arora Y (2020) Capsmf: a novel product recommender system using deep learning based text analysis model. Multimed Tools Appl 79(47):35927–35948. https://doi.org/10.1007/s11042-020-09199-5
23. Khan ZY, Niu Z, Sandiwarno S, Prince R (2021) Deep learning techniques for rating prediction: a survey of the state-of-the-art. Artif Intell Rev 54:95–135. https://doi.org/10.1007/s10462-020-09892-9
24. Kiran R, Kumar P, Bhasker B (2020) Dnnrec: a novel deep learning based hybrid recommender system. Expert Syst Appl 144:113054. https://doi.org/10.1016/j.eswa.2019.113054
25. Knerr S, Personnaz L, Dreyfus G (1990) Single-layer learning revisited: a stepwise procedure for building and training a neural network. Neurocomputing 41–50. https://doi.org/10.1007/978-3-642-76153-9_5
26. Koren Y (2008) Factorization meets the neighborhood: a multifaceted collaborative filtering models. In: Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining, pp 426–434. https://doi.org/10.1145/1401890.1401944
27. Kumar SG, Sridhar SS, Hussain A, Manikanthan SV, Padmapriya T (2022) Personalized web service recommendation through mishmash technique and deep learning model. Multimed Tools Appl 81:9091–9109. https://doi.org/10.1007/s11042-021-11452-4
28. Lin HT, Lin CJ (2003) A study on sigmoid kernels for svm and the training of non-psd kernels by smo-type methods. Neural Comput 3:1–32
29. Liu J, Pan W, Ming Z (2020) Cofigan: collaborative filtering by generative and discriminative training for one-class recommendation. Knowl-Based Syst 191:105255. https://doi.org/10.1016/j.knosys.2019.105255
30. Lu S, Chen H, Zhou X, Wang B, Wang H, Hong Q (2018) Graph-based collaborative filtering with mlp. Math Probl Eng. https://doi.org/10.1155/2018/8314105
31. Manogaran G, Varatharajan R, Priyan MK (2018) Hybrid recommendation system for heart disease diagnosis based on multiple kernel learning with adaptive neuro-fuzzy inference system. Multimed Tools Appl 77(4):4379–4399. https://doi.org/10.1007/s11042-017-5515-y
32. Mohammadpour T, Bidgoli AM, Enayatifar R, Haj Seyyed Javadi H (2023) Efficient recommendations in collaborative filtering recommender system: a multi-objective evolutionary approach based on nsga-ii algorithm. Int J Nonlinear Anal Appl 14(1):785–804. https://doi.org/10.22075/IJNAA.2020.21020.2226
33. Ravanifard R, Buntine W, Mirzaei A (2021) Recommending content using side information. Appl Intell 51(6):3353–3374. https://doi.org/10.1007/s10489-020-01945-4
34. Rendle S, Freudenthaler C, Gantner Z, Schmidt-Thieme L (2009) Bpr: Bayesian personalized ranking from implicit feedback. In: Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence, pp 452–461
35. Schölkopf B, Smola A, Müller KR (1998) Nonlinear component analysis as a kernel eigenvalue problem. Neural Comput 10(5):1299–1319. https://doi.org/10.1162/089976698300017467
36. Sharaf M, Hemdan EED, El-Sayed A, El-Bahnasawy NA (2022) A survey on recommendation systems for financial services. Multimed Tools Appl 81(12):16761–16781. https://doi.org/10.1007/s11042-022-12564-1
37. Tahmasbi H, Jalali M, Shakeri H (2021) Modeling user preference dynamics with coupled tensor factorization for social media recommendation. J Ambient Intell Humaniz Comput 12(10):9693–9712. https://doi.org/10.1007/s12652-020-02714-4
38. Tosh C, Krishnamurthy A, Hsu D (2021) Contrastive learning, multi-view redundancy, and linear models. In: Proceedings of the 32nd international conference on algorithmic learning theory, pp 1179–1206
39. Verma S, Pant M, Snasel V (2021) A comprehensive review on nsga-ii for multi-objective combinatorial optimization problems. IEEE Access 9:57757–57791. https://doi.org/10.1109/ACCESS.2021.3070634
40. Wang X, Zhang J (2012) Svd-based privacy preserving data updating in collaborative filtering. In: Proceedings of the world congress on engineering, pp 377–384
41. Wang X, He X, Wang M, Feng F, Chua TS (2019) Neural graph collaborative filtering. In: Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval, pp 165–174. https://doi.org/10.1145/3331184.3331267
42. Wang J, Lv L, Wu R, Fan T, Lee I (2020) Cosine kernel based density peaks clustering algorithm. Int J Comput Sci Math 12(2):1–20. https://doi.org/10.1504/IJCSM.2020.108790

43. Wang Z, Xia H, Chen S, Chun G (2021) Joint representation learning with ratings and reviews for recommendation. Neurocomputing 425:181–190. https://doi.org/10.1016/j.neucom.2020.04.033
44. Wen X (2021) Using deep learning approach and iot architecture to build the intelligent music recommendation system. Soft Comput 25(4):3087–3096. https://doi.org/10.1007/s00500-020-05364-y
45. Wilson AG, Hu Z, Salakhutdinov R, Xing EP (2016) Deep kernel learning. In: Proceedings of the 19th international conference on artificial intelligence and statistics, pp 370–378
46. Wu Y, Macdonald C, Ounis I (2020) A hybrid conditional variational autoencoder model for personalised top-n recommendation. In: Proceedings of the 2020 ACM SIGIR on international conference on theory of information retrieval, pp 89–96. https://doi.org/10.1145/3409256.3409835
47. Xue HJ, Dai X, Zhang J, Huang S, Chen J (2017) Deep matrix factorization models for recommender systems. In: Proceedings of the 26th international joint conference on artificial intelligence, pp 3203–3209. https://doi.org/10.24963/ijcai.2017/447
48. Zhang H, Ganchev I, Nikolov NS, Ji Z, O'Droma M (2021) Featuremf: an item feature enriched matrix factorization model for item recommendation. IEEE Access 9:65266–65276. https://doi.org/10.1109/ACCESS.2021.3074365