IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 25 May 2022, accepted 10 July 2022, date of publication 19 July 2022, date of current version 22 July 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3192427

RESEARCH ARTICLE

Kernel-Based Matrix Factorization With Weighted Regularization for Context-Aware Recommender Systems

VANDANA A. PATIL[1], SANTOSH V. CHAPANERI[2], (Senior Member, IEEE), AND DEEPAK J. JAYASWAL[2]

[1]Department of Information Technology, St. Francis Institute of Technology, Mumbai 400103, India
[2]Department of Electronics and Telecommunication Engineering, St. Francis Institute of Technology, Mumbai 400103, India

Corresponding author: Vandana A. Patil (vandanapatil@sfit.ac.in)

ABSTRACT As an essential task for recommender systems, the rating prediction problem over several contexts has attracted more attention over the recent years. The traditional approaches ignore the contexts and thus fail to predict the ratings for the unseen data in the rating tensor for varying contextual scenarios. Matrix factorization is preferred over decomposing the rating tensor for avoiding the burden of very high computational complexity while learning the interaction of users' and items' latent features. In this work, we propose a novel kernel loss function for optimizing the objective function of matrix factorization in a non-linearly projected rating space under multiple contexts in an optimum manner and also incorporate the implicit feedback of items in the learning process. Further, the optimization is regularized by applying different weights for each regularization term depending on the users' and items' participation. Extensive experimental evaluation on five benchmark context-aware datasets indicates the superiority of the proposed work for capturing the non-linearity and predicting the ratings of unseen items for users under varying contexts over the existing and baseline methods. The proposed kernel loss function is also shown to be resistant against shilling attacks in the recommender system. A detailed ablation study demonstrates the validity of the proposed work and the results are shown to be statistically significantly better with RMSE improvement in the range of 3% to 11% over the baseline methods.

INDEX TERMS Context-aware recommender systems, implicit feedback, matrix factorization, regularization, optimization, shilling attacks.

I. INTRODUCTION

Due to the abundance of smart devices and advancements in technologies for developing recommendation systems, the end-users seek satisfaction via services that learn the ratings of unrated items in the system and recommend the most relevant items by considering the user's preferences. The idea of including contextual factors in recommender systems has gained a lot of significance in recent years and thus there is more exciting research happening in the field of context-aware recommendation systems (CARS). The advantage of using these contextual factors is that the ratings for the unobserved items can be better estimated by the system since users have differing preferences for the items when the context changes. Some examples of contexts include the user and item metadata, time and/or location when the item is consumed by the users, etc. However, modeling the context in the recommendation system is a non-trivial problem since the dimensionality for model representation increases exponentially depending on the number of contextual factors involved.

We focus on the rating prediction problem in this work where the goal is to predict the ratings for the items that have not been rated yet during the user annotation process, by considering varying contextual factors. Once the ratings are predicted, the system may then recommend the top-rated

The associate editor coordinating the review of this manuscript and approving it for publication was Christos Anagnostopoulos.

items to the end-users who have not yet rated those items. For rating estimation under several contexts, three broad approaches exist [1] such as filtering before, filtering after, and modeling of contexts. Since the first two approaches do not explicitly consider the context information, we prefer the modeling approach where the prediction system can give a better prediction by directly using the valuable context information during the optimization phase.

To avoid the curse of dimensionality due to the increasing contextual factors, several approaches are suggested in the literature, most of which solve the rating prediction problem via factorization principles. Since the data will be a tuple made up of <user, item, context #1, context #2, ... >, the most straightforward approach is to use tensor factorization over such tuples. However, this approach is very computationally intensive when the number of contexts increases. Thus, matrix factorization methods are preferred since they are significantly faster compared to the tensor counterpart. But, incorporating the contexts in matrix factorization is challenging for mapping the multi-dimensional data to a two-dimensional space.

In this work, we estimate the ratings for the unobserved data samples by using an improved matrix factorization method by including the contexts and propose several enhancements over the existing methods. We first learn the implicit feedback of both users as well as items that can improve the prediction performance substantially as shown by the experiments. While the concept of user implicit feedback is not new, the impact of item feedback is ignored in the existing work. Learning such implicit feedback gives us clues on the user preferences for items and item appropriateness for users. Conventionally, the factors of the rating tensor or matrix are learned by optimizing the least-squares error loss function. But we demonstrate in this work that such a function has several drawbacks and we propose a novel kernel loss function that projects the original ratings of items by users under varying contexts to a high dimensional manifold where the learning capability is improved. To avoid over-fitting during the training phase, some regularization terms are usually added; however, most existing works use the same hyper-parameter for such terms or perform a careful fine-tuning to get the optimal hyper-parameter. Under this setting, it is assumed that all items and all users are given equal importance, which is not a valid assumption in practice. Thus, we propose weighted regularization that gives different weights to users and items depending on their contribution to the system. An advantage of this step is that the heavy users who contribute a lot to the rating system and popular items that have lots of ratings will not be overly penalized and more penalty is offered to less active users and items having a long tail, thus the resulting optimization has fewer chances of over-fitting.

Our contributions are five-fold:

(a) Besides the user implicit feedback, the item implicit feedback is also learned during the optimization process to improve the system performance.
(b) A novel convex kernel loss function is proposed with nice properties that can be utilized to reduce the impact of malicious users and which results in less error compared to the traditional least-squares loss function.
(c) The proposed kernel context-aware matrix factorization (KCAMF) method is robust to shilling attacks, which is validated theoretically as well as experimentally on benchmark datasets.
(d) Instead of using the same regularization hyper-parameters in the optimization process, we propose weights for such hyper-parameters so that every user and every item is treated differently during the learning phase.
(e) Over five benchmark datasets, the proposed KCAMF method performs better than the existing and recent state-of-the-art methods for solving the rating prediction problem.

The rest of this paper is structured as follows: Section II reviews the existing work in literature for solving the problem of rating prediction in a context-aware recommender system. Section III summarizes the baseline methods used in this work for a comparative evaluation, and Section IV describes the proposed work with details of each of the above-mentioned contributions. Section V covers the experimental results on benchmark datasets and presents an ablation study to evaluate the impact of each contribution. Finally, the work is concluded in Section VI by summarizing the contributions and providing scope for further work in this area.

II. RELATED WORK

A brief overview of the methods for solving the problem of rating prediction is given in [2] which discusses the collaborative filtering, content-based, and hybrid-based techniques and also sheds light on deep learning methods such as autoencoders and Boltzmann machines for predicting the ratings. For incorporating the contexts, factorization machines are used in [3] for modeling the second-order interaction terms in a non-linear way using convolution operations, but this method requires computations to be performed via GPU given its high computational complexity. Auto-encoders and multi-layer perceptrons were used in [4] to learn the interactions between items and users but the authors concluded that the training takes a much longer time with an increase in the depth of the network. Deep sparse autoencoder with particle swarm optimization was used in [5] to learn the latent representation of ratings and trust information to calculate the unknown ratings. A time-aware recommender system was proposed in [6] based on temporal reliability and confidence scores to take into account the varying user's preferences over time, but ignores the context under which the user may have given the ratings. In [7], the authors learned the latent contexts from the datasets instead of using the original contexts with an auto-encoder. However, as noted in [2], such systems suffer from high complexity due to deeper architectures, hence we resort to the simpler mechanisms of factorization for predicting the ratings.

The rating tensor comprising of users, items, and several context dimensions was decomposed using tensor factorization in [8] by also modeling the trusts between users. The Tucker decomposition was applied in [9] with K-means clustering for obtaining similar users, but this method cannot perform well if the dataset is highly sparse (which is the typical case in recommender systems). The tensor factorization was learned via a neural network in [10] to obtain the projected user and item embeddings. However, the methods of tensor factorization have a higher complexity of computations when the tensor is sparse and large compared to the simpler matrix factorization (MF) as noted in [11]; also, the matrix factorization methods have the advantage of obtaining higher accuracy and they are easily scalable to large-scale datasets [12]. The regular MF [13] learns the item and user latent features compared to the biased MF that only learns their corresponding biases and the model parameters are optimized using Stochastic Gradient Descent (SGD). A matrix factorization model was presented in [12] to also learn the temporal and social features for modeling the change of user preferences over time as well as trust among users, respectively. Under different scenarios, each user may have varying preferences for items and to capture this aspect, dual preferences based MF was proposed in [14]. For solving the data sparsity problem, especially for cold-start items, a transfer learning method for MF was proposed in [15] for sharing the knowledge from domains that are relatively rich in information. An attention-based convolutional MF was proposed in [16] to model the side information of items and users. But most of these techniques ignore the contextual factors when predicting the rating, and are applicable only for non-context-based recommender systems. A kernel context-aware recommender system was proposed in [17] to model the different contexts under a non-linear mapping of feature vector spaces. Context similarity using a similarity kernel was determined in [18] for both users and items to determine the predicted ratings under varying contexts. A simplified version of differential context weighting (DCW) based on context similarity was analyzed in [19]. An implicit hierarchical latent context representation was learned in [20] using autoencoder and hierarchical clustering. The context-aware MF (CAMF) method was proposed in [21] to model the multi-dimensional contexts using the simpler matrix factorization principles. Due to its simplicity, we adopt CAMF and its variants as the baseline in this work and propose several enhancements for improving the system performance.

Since the rating tensor is usually highly sparse, besides the explicit rating information available, the implicit feedback is also modeled for each user to get an understanding of the user's affinities towards different items in an indirect manner [22]. In [23], an implicit user feedback matrix was learned incrementally using the matrix co-factorization approach where the users, items, and feedback features are jointly decomposed in the shared latent space. An extensive empirical study was performed in [24] to determine the efficacy of implicit feedback for enhanced user engagement. The implicit feedback was modeled in [25] using soft target enhancements instead of using the softmax activations in neural networks. The implicit feedback information is projected to a feature space of low dimensions in [26] and only the primary learned features are retained using the concept of embedding. In [27], the explicit and implicit feedback matrices are learned in the shared feature space simultaneously and are jointly optimized using standard gradient descent algorithms.

To avoid over-fitting that can occur when learning the model parameters, regularization terms are often included to constrain the parameter values. In [28], the weights of users and items are learned and applied directly to the optimization function; these weights attempt to reduce the impact of malicious users who tend to give bad or incorrect ratings. Several variations of MF were proposed in [29] for regularizing the optimization of matrix-completion based on the frequency of users and items contribution. Dynamic regularization was proposed in [30] for modeling the dynamic changes (e.g. user preferences drift over time, seasonal factors, etc.) in online collaborative filtering. In [31], it was proposed to use different hyper-parameters for the regularization terms based on the popularity of users and items. The explicit feedback of users' latent representations is applied in [32] for the regularization term that is shown to perform better than the standard regularization method. In [33], differentiating regularizing functions (linear, logarithmic) are proposed to adjust each hyper-parameter of the regularization term for solving the problem of cold-start in recommendation systems. In line with this, we propose to apply weighted regularization for different terms of the objective function depending on the importance of items and the effort of users.

III. BASELINE METHODS

In this section, the CAMF model and its improved version ICAMF is discussed to serve as the baseline for the proposed work. Table 1 shows the notations used in this paper for quick reference. The sparse rating dataset is denoted as $\mathcal{K}$ which includes few known ratings and several unknown ratings. The dataset of observed ratings is denoted by $\mathcal{D} \subset \mathcal{K}$, which is split in two sets: a training set $\mathcal{D}_{tr}$ and a testing set $\mathcal{D}_{te}$. $\mathcal{K}'$ represents the set of tuples for whom the rating is to be predicted. The context vector is denoted as $\mathbf{c} = \langle \mathcal{C}_1, \ldots, \mathcal{C}_L \rangle$, where L is the number of context dimensions. Each context dimension has the context conditions denoted by $\mathcal{C}_n = \{0, 1, \ldots, \mathcal{Z}_n\}$, where 0 refers to unknown condition (or N/A), and the remaining values refer to specific conditions. The total number of context conditions are thus given by $\mathcal{Z}_1 + \ldots + \mathcal{Z}_L$.

The block diagram for the baseline CAMF and ICAMF methods is shown in Fig. 1. Both CAMF and ICAMF methods are trained using $\mathcal{D}_{tr}$ and evaluated using $\mathcal{D}_{te}$. The rating prediction functions for CAMF and ICAMF are slightly different and both use the least square loss function as the optimization function for training using Stochastic Gradient Descent (SGD) to optimize the learnable model parameters.

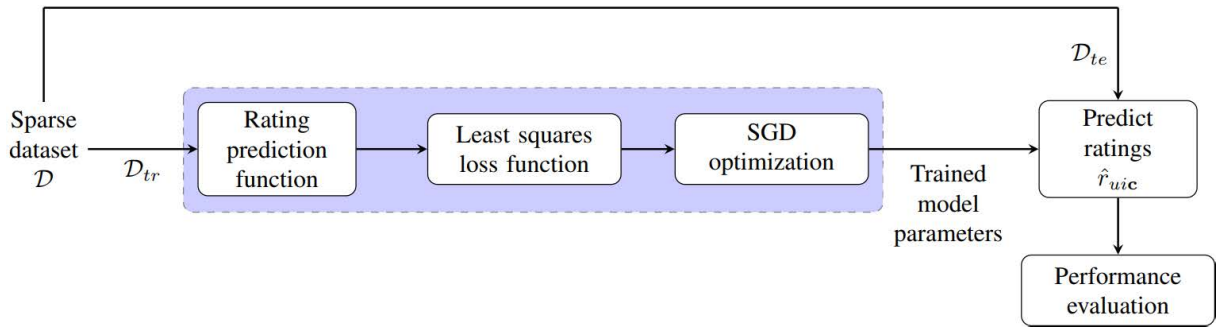


FIGURE 1. Block diagram of CAMF and ICAMF.

TABLE 1. Notations used in this work.

Notation	Description
$\mathcal{K}$	sparse rating dataset
$\mathcal{D}$	set of known ratings ($\mathcal{D} \subset \mathcal{K}$)
$\mathcal{D}_{tr}$	train dataset ($\mathcal{D}_{tr} \subset \mathcal{D}$)
$\mathcal{D}_{te}$	test dataset ($\mathcal{D}_{te} \subset \mathcal{D}$)
$\mathcal{K}^{'}$	unknown ratings to be predicted ($\mathcal{K}^{'} \subset \mathcal{K}$)
$\mathbf{c}$	context vector $\langle \mathcal{C}_1, \ldots, \mathcal{C}_L \rangle$ where L is the number of context dimensions
$\mathcal{C}_n$	n^{th} contextual dimension ($n = 1 \ldots L$)
$r_{ui\mathbf{c}}$	actual rating for $(u, i, \mathbf{c})$ tuple
$\hat{r}_{ui\mathbf{c}}$	predicted rating for $(u, i, \mathbf{c})$ tuple
μ	global mean
F	dimensionality of latent feature vectors
$\mathbf{p}_u, \mathbf{g}_v$	latent features $\in \mathbb{R}^F$ for users u, v
$\mathbf{q}_i, \mathbf{y}_j$	latent features $\in \mathbb{R}^F$ for items i, j
$\alpha_{u\mathcal{C}_n}$	user bias under $\mathcal{C}_n$
$\beta_{i\mathcal{C}_n}$	item bias under $\mathcal{C}_n$
$\mathbf{h}_{\mathcal{C}_n}$	latent features $\in \mathbb{R}^F$ for $\mathcal{C}_n$
$N_u^{(I)}$	set of items rated by u^{th} user
$N_i^{(U)}$	set of users who gave rating for i^{th} item
γ	kernel hyper-parameter
δ	learning rate
λ	regularization hyper-parameter
$\mathcal{A}_F$, $\mathcal{A}_T$	set of filler & target items for shilling attacks

Using the trained parameters, the ratings are predicted using the prediction function for the test set, and the performance is measured using Root Mean Square Error (RMSE) as well as Mean Absolute Error (MAE).

A. CAMF

Context-aware matrix factorization (CAMF) is an extension of the standard Matrix Factorization (MF) by considering the contextual factors in the function for rating prediction. The CAMF rating prediction function is given by Eq. (1), where μ is the global average of the ratings in $\mathcal{D}_{tr}$, $\mathbf{p}_u \in \mathbb{R}^F$ is the latent feature vector of u^{th} user, $\mathbf{q}_i \in \mathbb{R}^F$ is the latent feature vector of i^{th} item, F is the dimensionality of latent feature vectors, α is the user-context interaction bias term, and β is the item-context interaction bias term. The optimization function of CAMF is given by Eq. (2) using the least-squares loss function and regularization terms for the trainable parameters $\boldsymbol{\theta} = \{\mathbf{p}_u, \mathbf{q}_i, \alpha_{u\mathcal{C}_n}, \beta_{i\mathcal{C}_n}\} \forall\, u, i, \mathcal{C}_n \in \mathcal{D}_{tr}$ to avoid overfitting. Typically, same hyper-parameter is chosen for each regularization term, i.e. $\lambda_p = \lambda_q = \lambda_\alpha = \lambda_\beta = \lambda$ for simplicity.

$$\hat{r}_{ui\mathbf{c}} = \mu + \sum_{n=1}^{L} \alpha_{u\mathcal{C}_n} + \sum_{n=1}^{L} \beta_{i\mathcal{C}_n} + \mathbf{p}_u^T \mathbf{q}_i \tag{1}$$

$$\min_{\boldsymbol{\theta}} \sum_{u,i,\mathbf{c} \in \mathcal{D}_{tr}} (\hat{r}_{ui\mathbf{c}} - r_{ui\mathbf{c}})^2 + \frac{\lambda_p}{2} \|\mathbf{p}_u\|^2 + \frac{\lambda_q}{2} \|\mathbf{q}_i\|^2 + \frac{\lambda_\alpha}{2} \sum_{n=1}^{L} \alpha_{u\mathcal{C}_n}^2 + \frac{\lambda_\beta}{2} \sum_{n=1}^{L} \beta_{i\mathcal{C}_n}^2 \tag{2}$$

B. IMPROVED CAMF

An improved version of CAMF called ICAMF was introduced in [34] to also include the latent feature vectors for each context. While CAMF models only the user-context and item-context bias terms, ICAMF also models the user-context and item-context latent feature interaction. Accordingly, the ICAMF rating prediction function is given by Eq. (1), where $\mathbf{h}_{\mathcal{C}_n}$ denotes the latent feature vector of n^{th} contextual condition. The optimization function of ICAMF is given by Eq. (4).

$$\hat{r}_{ui\mathbf{c}} = \mu + \sum_{n=1}^{L} \alpha_{u\mathcal{C}_n} + \sum_{n=1}^{L} \beta_{i\mathcal{C}_n} + \mathbf{p}_u^T \mathbf{q}_i + \sum_{n=1}^{L} \mathbf{p}_u^T \mathbf{h}_{\mathcal{C}_n} + \sum_{n=1}^{L} \mathbf{q}_i^T \mathbf{h}_{\mathcal{C}_n} \tag{3}$$

$$\min_{\boldsymbol{\theta}} \sum_{u,i,\mathbf{c} \in \mathcal{D}_{tr}} (\hat{r}_{ui\mathbf{c}} - r_{ui\mathbf{c}})^2 + \frac{\lambda_p}{2} \|\mathbf{p}_u\|^2 + \frac{\lambda_q}{2} \|\mathbf{q}_i\|^2 + \frac{\lambda_h}{2} \|\mathbf{h}_{\mathcal{C}_n}\|^2 + \frac{\lambda_\alpha}{2} \sum_{n=1}^{L} \alpha_{u\mathcal{C}_n}^2 + \frac{\lambda_\beta}{2} \sum_{n=1}^{L} \beta_{i\mathcal{C}_n}^2 \tag{4}$$

IV. PROPOSED WORK

In this work, we propose a rating prediction framework called Kernel CAMF (KCAMF) which modifies the ICAMF model

by incorporating user and item implicit feedback. We also use a kernel loss function instead of the conventional least square error loss function and apply weighted regularization for the hyper-parameters for optimizing using SGD.

Fig. 2 shows the working of the proposed method, where the input is the sparse dataset $\mathcal{K}$ that is split into $\mathcal{D}$, which is the set of observed ratings and $\mathcal{K}'$, which is the set of unobserved ratings. The set of observed ratings $\mathcal{D}$ is further split into training and test sets $\mathcal{D}_{tr}$ and $\mathcal{D}_{te}$, respectively. The rating prediction function is modeled using $\mathcal{D}_{tr}$, and the optimization function is kernel loss function which is optimized using SGD with weights applied on the regularization terms. The trained model parameters are used for predicting the ratings of the test set $\mathcal{D}_{te}$; these predicted ratings are then compared with the known test ratings and the performance is measured using RMSE and MAE. Further, the trained model is used to compute the ratings for the set of tuples $\mathcal{K}'$ to solve the rating prediction problem resulting in the dense dataset, which can be used for recommending items to the end-user.

A. KCAMF RATING PREDICTION

The KCAMF rating prediction function is given by Eq. (5), which includes the user-context bias, item-context bias, user-item latent feature interaction, user-context latent feature interaction, item-context latent feature interaction, as well as the user and item implicit feedback terms.

The implicit feedback is incorporated to model the rating preferences for users and items since we can determine implicitly which user has rated which item irrespective of the rating values. The advantage of including implicit feedback is that it normalizes the estimated rating by encouraging more deviations from the baseline ratings for those heavy users (i.e. users who have rated several items). On the contrary, for users who have rated fewer items, it is desirable that the predicted rating stays closer to the baseline. This phenomenon is modeled for both users as well as items (heavy-rated items v/s items that received fewer ratings).

To include the user implicit feedback, a binary matrix $\mathbf{B} \in \mathbb{R}^{|U|\times|I|}$ is constructed where each element determines whether the u^{th} user has rated the i^{th} item, regardless of the rating value. The matrix $\mathbf{Y} \in \mathbb{R}^{|I|\times F}$ represents the implicit item affinities $\mathbf{y}$ for each user. The user latent feature matrix $\mathbf{P}$ is thus updated to $\mathbf{P}+\mathbf{BY}$, and each row of $\mathbf{B}$ is unit-normalized by $|N_u^{(I)}|^{-\frac{1}{2}}$, where $N_u^{(I)}$ is the set of items rated by the u^{th} user. Similarly, to include the item implicit feedback, a binary matrix $\mathbf{A} \in \mathbb{R}^{|I|\times|U|}$ is constructed where each element determines whether the item i rated by the user u, irrespective of the rating value. The matrix $\mathbf{G} \in \mathbb{R}^{|U|\times F}$ represents the implicit user affinities $\mathbf{q}$ for each item. The item latent feature matrix $\mathbf{Q}$ is thus updated to $\mathbf{Q}+\mathbf{AG}$, and each row of $\mathbf{A}$ is unit-normalized by $|N_i^{(U)}|^{-\frac{1}{2}}$, where $N_i^{(U)}$ is the set of users who have rated the i^{th} item.

The KCAMF rating prediction function is thus given by Eq. (5), where the updated user and item latent feature vectors are given by (6) and Eq. (7), respectively.

$$\hat{r}_{ui\mathbf{c}} = \mu + \sum_{n=1}^{L} \alpha_{u\mathcal{C}_n} + \sum_{n=1}^{L} \beta_{i\mathcal{C}_n} + \mathbf{p}_u'^T \mathbf{q}_i' + \sum_{n=1}^{L} \mathbf{p}_u'^T \mathbf{h}_{\mathcal{C}_n} + \sum_{n=1}^{L} \mathbf{q}_i'^T \mathbf{h}_{\mathcal{C}_n} \tag{5}$$

$$\mathbf{p}_u' = \mathbf{p}_u + |N_u^{(I)}|^{-\frac{1}{2}} \sum_{j\in N_u^{(I)}} \mathbf{y}_j \tag{6}$$

$$\mathbf{q}_i' = \mathbf{q}_i + |N_i^{(U)}|^{-\frac{1}{2}} \sum_{v\in N_i^{(U)}} \mathbf{g}_v \tag{7}$$

B. KERNEL LOSS FUNCTION

The loss function is typically computed as the least squares error estimate given by Eq. (8), where the residual error e is the difference between the predicted and the true observed rating in the training dataset.

$$J_{LS}(e_{ui\mathbf{c}}) = e_{ui\mathbf{c}}^2 = (\hat{r}_{ui\mathbf{c}} - r_{ui\mathbf{c}})^2 \tag{8}$$

However, the derivative of the least-squares loss function is unbounded since it is given by $J'_{LS}(e) = 2e$, which grows linearly with the residual error. Also, its second derivative $J''_{LS}(e) = 2$ is a constant scalar, i.e. the average rate of change of the gradient of the least-squares loss function does not vary, as illustrated in Fig. 3.

Also, the least-squares error estimate assumes that the ratings can be projected along a low-dimensional linear manifold across users, items, and context dimensions with the matrix factorization approach. However, this assumption is violated for non-linear interaction between ratings and the (user, item, context) tuples as illustrated in Fig. 4.

Thus, we transform the ratings into a non-linear kernel space as $\mathcal{T} \rightarrow \Psi(\mathcal{T})$ for the actual rating tensor and $\hat{\mathcal{T}} \rightarrow \Psi(\hat{\mathcal{T}})$ for the predicted rating tensor, where $\Psi(\cdot)$ is any non-linear function. The residual error in the non-linear space can be calculated as

$$\begin{aligned} d(\hat{\mathcal{T}}, \mathcal{T}) &= \|\Psi(\hat{\mathcal{T}}) - \Psi(\mathcal{T})\|^2 \\ &= \langle \Psi(\hat{\mathcal{T}}), \Psi(\hat{\mathcal{T}})\rangle + \langle \Psi(\mathcal{T}), \Psi(\mathcal{T})\rangle - 2\langle \Psi(\hat{\mathcal{T}}), \Psi(\mathcal{T})\rangle \\ &= \varkappa(\hat{\mathcal{T}}, \hat{\mathcal{T}}) + \varkappa(\mathcal{T}, \mathcal{T}) - 2\varkappa(\hat{\mathcal{T}}, \mathcal{T}) \\ &= 2\left[1 - \varkappa(\hat{\mathcal{T}}, \mathcal{T})\right] \end{aligned} \tag{9}$$

We use the Gaussian kernel given by $\varkappa(\hat{r}, r) = \exp\left(-\gamma(\hat{r} - r)^2\right)$, thus the proposed kernel loss function is given by Eq. (10) for a specific $(u, i, \mathbf{c})$ tuple.

$$J_K(e_{ui\mathbf{c}}) = 2\left[1 - \exp\left(-\gamma(\hat{r}_{ui\mathbf{c}} - r_{ui\mathbf{c}})^2\right)\right] \tag{10}$$

The proposed kernel loss function has the following properties:

(a) *The kernel loss function has a unique optimal solution.* Proof: The first derivative of the kernel loss function is given by $J'_K(e) = 4\gamma e \exp(-\gamma e^2)$ and its second

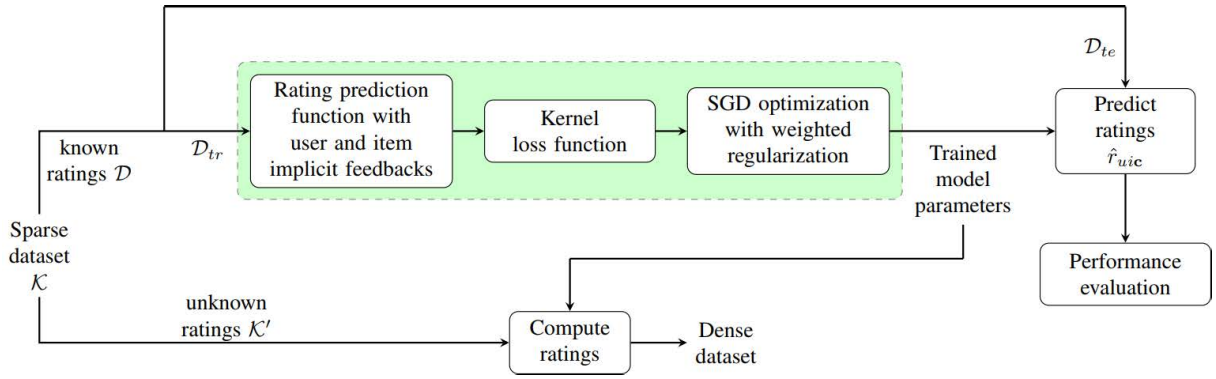


FIGURE 2. Block diagram of KCAMF for rating prediction and obtaining the dense rating tensor.

derivative, using the chain rule, is given by $J''_K(e) = 4\gamma\left[1-2\gamma e^2\right]\exp(-\gamma e^2)$, as shown in Fig. 5. For any arbitrary residual error e, we have $\exp(-\gamma e^2) > 0$. For the second derivative to be positive, we need $\left[1-2\gamma e^2\right] > 0$, which can be satisfied for any $\gamma < \frac{1}{2e^2}$. If the rating scale of the dataset is $r \in [0, 5]$, then the range of residual error will be $e \in [-5, 5]$, and thus the maximum squared error will be 25, which leads to $\gamma < \frac{1}{50}$. Let $\mathcal{G} = \text{dom}(J_K)$ refer to the domain of the kernel loss function. Then, it can be shown that $J_K(e)$ is a strictly convex function for a convex $\mathcal{G}$ and $J''_K(e) > 0\ \forall e \in \mathcal{G}$. Since the kernel loss function is twice differentiable for all $f, h \in \mathcal{G}$, consider $f \neq h$ with $f < h$. According to Taylor's theorem, for any $\zeta \in [f, h]$, we have

$$J_K(h) = J_K(f) + J'_K(f)(h-f) + \frac{J''_K(\zeta)}{2!}(h-f)^2$$

Since $J''_K(\zeta) > 0$ for any $\gamma < \frac{1}{50}$, this reduces to

$$J_K(h) > J_K(f) + J'_K(f)(h-f)$$

Consider $0 < \nu < 1$ and $z = \nu f + (1-\nu)h$. Then, we have

$$J_K(f) > J_K(z) + J'_K(z)(f-z) \tag{11}$$
$$J_K(h) > J_K(z) + J'_K(z)(h-z) \tag{12}$$

Multiplying Eq. (11) with ν and Eq. (12) with $1-\nu$ and adding them, we obtain

$$\nu J_K(f) + (1-\nu)J_K(h) > J_K(z) = J_K(\nu f + (1-\nu)h)$$

This proves that the kernel loss function is strictly convex, and thus it has a unique optimal solution.

1) *The kernel loss function is robust to attacks by malicious users.*
Proof: The least-squares loss function is highly sensitive to outliers. For any malicious user, the residual error e will be a large value and thus the impact of such error on $J_{LS}(e) = e^2$ will be significantly larger, which impacts the learning of model parameters (biases and latent feature vectors), and thus the model solution can be sub-optimal. However, for the kernel loss function $J_K(e) = 2[1-\exp(-\gamma e^2)]$, the impact of large residual errors is small. This is because the kernel loss function is bounded as shown in Fig. 5, since the absolute value of its first derivative is $|J'_K(e)| = 4\gamma|e|\exp(-\gamma e^2)$, where $e \in [-c, c]$ in general with c as the maximum rating scale range, and thus $|J'_K(e)| \leq 4\gamma|e|$. The robustness of the kernel loss function against malicious user attacks is demonstrated experimentally in Sec. V-D3.
2) *The kernel loss function results in a loss smaller than the least-squares loss function.*
Proof: Consider the difference between the kernel loss estimate and the least-squares error estimate as

$$f(e) = J_K(e) - J_{LS}(e) = 2[1-\exp(-\gamma e^2)] - e^2$$

Its first derivative is given by

$$\begin{aligned} f'(e) &= -2(-2\gamma e)\exp(-\gamma e^2) - 2e \\ &= 2\,e\left[2\gamma\exp(-\gamma e^2) - 1\right] \end{aligned}$$

For any $\gamma < \frac{1}{50}$, $\left[2\gamma\exp(-\gamma e^2) - 1\right] < 0$, where $e \in [-5, 5]$ and the difference function is continuous in $[-5, 5]$. When $e \in [-5, 0)$, $f'(e) > 0$ which implies that the difference function is monotonically increasing. When $e \in (0, 5]$, $f'(e) < 0$ impliying that the difference function is monotonically decreasing. Thus, the difference function attains the maximum value at $e = 0$. Thus, $f(e) < 0$ and hence $J_K(e) < J_{LS}(e)$ as illustrated in Fig. 6.

C. OPTIMIZATION WITH WEIGHTED REGULARIZATION

The loss function is typically optimized using Stochastic Gradient Descent (SGD) or its variants, namely RMSProp, ADAM, etc. In this work, the kernel loss function is minimized using SGD due to its simplicity. To avoid over-fitting, regularization terms are added to the kernel loss function. In most existing relevant work, a common regularization hyper-parameter λ is used for user, item latent feature vectors,

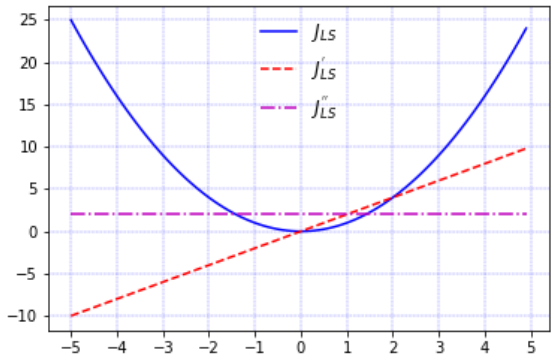


FIGURE 3. Least-squares loss function.

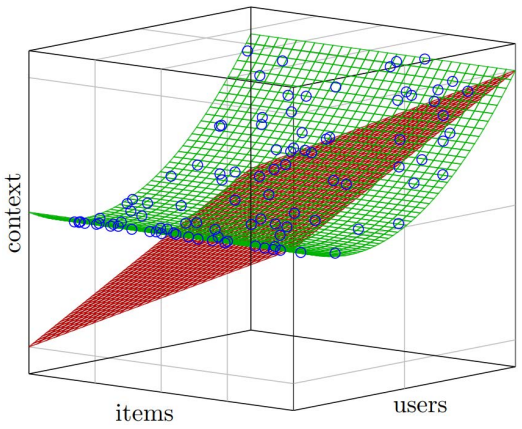


FIGURE 4. Ratings of users for items for a specific context; the linear hyperplane cannot capture all available ratings whereas the kernel function is appropriate to model the non-linearity.

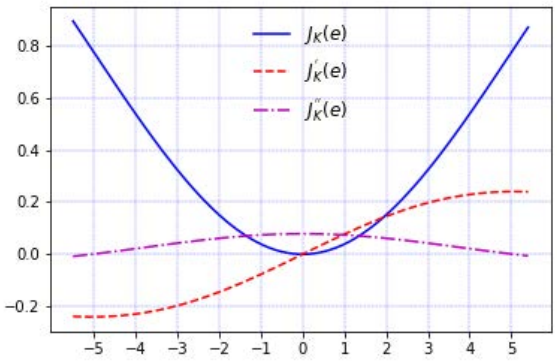


FIGURE 5. Kernel loss function.

and the bias interaction terms. However, this strategy is too naive since we should handle the heavy users and popular items and the less active users and long-tail items separately. For popular items, since many ratings are available, the corresponding item latent features could give more clues, whereas, for long-tail items, little information is available, and thus the corresponding item latent features could be noisy. Likewise, if a user interacted more with the system, i.e. provided more ratings, then the corresponding user latent features can better

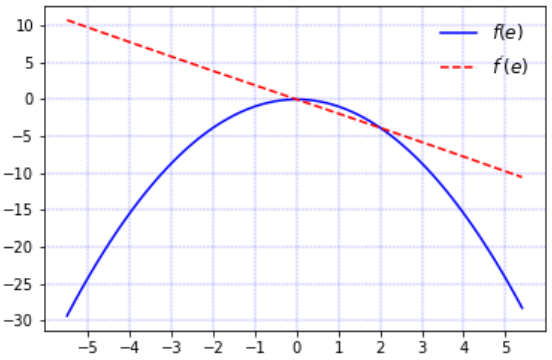


FIGURE 6. Difference between J_K and J_{LS}.

represent this user's preferences and for less active users, the corresponding user latent features could be noisy due to limited information.

To solve this problem, we apply weights for each regularization term to avoid over-fitting for new users and new items. For heavy users (i.e. users who rated more items) and popular items (i.e. items for which several ratings are available), the over-fitting would be typically less and thus the penalty for such users should be subsequently less. On the contrary, for users who are not active as well as for items in the long-tail, the penalty should be more since there is a high chance of over-fitting. Thus, the penalty can be adjusted based on the number of items rated by each user as well as the number of users who rated each item. As a result, the weight for each regularization term is inversely proportional to the corresponding cardinality of either $N_u^{(I)}$ or $N_i^{(U)}$.

The optimization function of KCAMF is $\min_{\boldsymbol{\theta}} \mathcal{S}(\boldsymbol{\theta})$ where $\mathcal{S}(\boldsymbol{\theta})$ is given by Eq. (14), as shown at the bottom of the next page, $N_u^{(I)}$ is the set of items rated by user u, $N_i^{(U)}$ is the set of users who rated item i, and $N_j^{(U)}$ is the set of users who rated item j. The KCAMF learnable model parameters are given by Eq. (13). For each parameter, the SGD update can be obtained using the general form $\theta_t \leftarrow \theta_{t-1} - \delta \frac{\partial \mathcal{S}}{\partial \theta_{t-1}}$, where t refers to the iteration index, and δ is the learning rate.

$$\begin{aligned}\boldsymbol{\theta} = \big[&\mathbf{p}_1, \ldots, \mathbf{p}_{|U|}, \mathbf{q}_1, \ldots, \mathbf{q}_{|I|}, \mathbf{y}_1, \ldots, \mathbf{y}_{|I|},\\ &\mathbf{g}_1, \ldots, \mathbf{g}_{|I|}, \mathbf{h}_{\mathcal{C}_1}, \ldots, \mathbf{h}_{\mathcal{C}_L},\\ &\alpha_{1\mathcal{C}_1}, \ldots, \alpha_{|U|\mathcal{C}_L}, \beta_{1\mathcal{C}_1}, \ldots, \beta_{|I|\mathcal{C}_L}\big] \end{aligned} \tag{13}$$

The update equations for the model parameters via SGD are given by Eq. (15) to Eq. (21), as shown at the bottom of the next page.

D. ALGORITHMS

Algorithm 1 outlines the steps for training the KCAMF model parameters $\boldsymbol{\theta}$ using SGD. The input for training is the training set $\mathcal{D}_{tr}$ and the hyper-parameters. In line 1, the model parameters are initialized randomly using a standard normal distribution $\mathcal{N}(0, 1)$. From lines 2 to 11, the process is iteratively repeated till convergence, i.e. until the kernel loss becomes

less than a threshold $|\mathcal{S}_t - \mathcal{S}_{t-1}| < 10^{-4}$. For each tuple of user, item, and context in the training set, the user and item latent features are determined in line 4 using implicit feedback terms following which the rating $\hat{r}_{ui\mathbf{c}}$ is predicted using the KCAMF rating prediction function. In line 6, the error between the true rating available from the training dataset and the predicted rating is computed. The model parameters are then updated using the error e with stochastic gradient descent updates in lines 7 to 9.

Algorithm 2 outlines the steps for testing the KCAMF model as well as computing the ratings of the tuples of the dataset for whom ratings are not available. In line 1, $\mathcal{K} \setminus \mathcal{D}_{tr}$ results in the set of tuples of the test set $\mathcal{D}_{te}$ as well as $\mathcal{K}'$, i.e. the set of tuples for which the ratings are unavailable. In lines 2 to 4, the ratings $\hat{r}_{ui\mathbf{c}}$ are predicted for each tuple in the test set and the performance is measured using RMSE and MAE in line 5. Further, the ratings are predicted for each tuple in $\mathcal{K}'$ resulting in the dense rating tensor.

Algorithm 1: KCAMF Training

Input : Training dataset $\mathcal{D}_{tr}$, λ, δ, γ, F
Output: Trained parameters: $\boldsymbol{\theta}$
1 **Initialize** parameters $\boldsymbol{\theta}$ randomly
2 **Repeat**
3 **for** $(u, i, \mathbf{c}) \in \mathcal{D}_{tr}$ **do**
4 **Compute**: updated user and item latent feature vectors with implicit feedback p'_u and q'_i respectively using Eq. (6) and Eq. (7)
5 **Compute**: predict rating $\hat{r}_{ui\mathbf{c}}$ using Eq. (5)
6 **Compute**: error $e = \hat{r}_{ui\mathbf{c}} - r_{ui\mathbf{c}}$
7 **Update parameters**: $p_{uf}, q_{if}, h_{\mathcal{C}_n f}, \alpha_{u\mathcal{C}_n}, \beta_{i\mathcal{C}_n}$ using Eq. (15) to Eq. (19) , $\forall\ n = 1 \ldots L$
8 $\forall\ j \in N_u^{(I)}$, update y_{jf} using Eq. (20)
9 $\forall\ v \in N_i^{(U)}$, update g_{vf} using Eq. (21)
10 **end for**
11 **Until convergence**

E. COMPUTATIONAL COMPLEXITY

We compare the model complexity and time complexity of the proposed method with the baseline CAMF method. The model complexity depends on the number of parameters to be learned during training. For CAMF, the learnable parameters are $\mathbf{P} \in \mathbb{R}^{|U|\times F}$, $\mathbf{Q} \in \mathbb{R}^{|I|\times F}$, $\boldsymbol{\alpha} \in \mathbb{R}^{|U|\times L}$, and $\boldsymbol{\beta} \in \mathbb{R}^{|I|\times L}$. Thus, the total model complexity of CAMF is $\Big((|U| + |I|) \times (F + L)\Big)$ where, F and L are number of features and number of context variables, respectively and $|U|$ and $|I|$ are the number of users and number of items in the training dataset, respectively. For the proposed method of KCAMF, there are seven learnable parameters $\mathbf{P} \in \mathbb{R}^{|U|\times F}$, $\mathbf{Q} \in \mathbb{R}^{|I|\times F}$, $\boldsymbol{\alpha} \in \mathbb{R}^{|U|\times L}$, $\boldsymbol{\beta} \in \mathbb{R}^{|I|\times L}$, $\mathbf{H} \in \mathbb{R}^{L\times F}$, $\mathbf{Y} \in \mathbb{R}^{|I|\times F}$ and $\mathbf{G} \in \mathbb{R}^{|U|\times F}$. Thus, the total model complexity of KCAMF is $\Big((|U| + |I|) \times (2F + L) + LF\Big)$, which is slightly higher compared to CAMF (note that, typically $L \ll F$).

$$\mathcal{S}(\boldsymbol{\theta}) = \sum_{u,i,\mathbf{c}\in\mathcal{D}_{tr}} 2\Big[1 - \exp(-\gamma(\hat{r}_{ui\mathbf{c}} - r_{ui\mathbf{c}})^2)\Big] + \frac{\lambda}{2}|N_u^{(I)}|^{-\frac{1}{2}}\|\mathbf{p}_u\|^2 + \frac{\lambda}{2}|N_i^{(U)}|^{-\frac{1}{2}}\|\mathbf{q}_i\|^2 + \frac{\lambda}{2}\sum_{j\in N_u^{(I)}}|N_j^{(U)}|^{-\frac{1}{2}}\|\mathbf{y}_j\|^2 + \frac{\lambda}{2}\sum_{v\in N_i^{(U)}}|N_v^{(I)}|^{-\frac{1}{2}}\|\mathbf{g}_v\|^2 + \frac{\lambda}{2}|N_u^{(I)}|^{-\frac{1}{2}}\sum_{n=1}^{L}\alpha_{u\mathcal{C}_n}^2 + \frac{\lambda}{2}|N_i^{(U)}|^{-\frac{1}{2}}\sum_{n=1}^{L}\beta_{i\mathcal{C}_n}^2 + \frac{\lambda}{2}\sum_{n=1}^{L}\|\mathbf{h}_{\mathcal{C}_n}\|^2 \tag{14}$$

$$p_{uf} \leftarrow p_{uf} - \eta\left[4\gamma e\exp(-\gamma e^2)\left(q'_{if} + \sum_{n=1}^{L}h_{\mathcal{C}_n f}\right) + \lambda|N_u^{(I)}|^{-\frac{1}{2}}p_{uf}\right] \tag{15}$$

$$q_{if} \leftarrow q_{if} - \eta\left[4\gamma e\exp(-\gamma e^2)\left(p'_{uf} + \sum_{n=1}^{L}h_{\mathcal{C}_n f}\right) + \lambda|N_i^{(U)}|^{-\frac{1}{2}}q_{if}\right] \tag{16}$$

$$h_{\mathcal{C}_n f} \leftarrow h_{\mathcal{C}_n f} - \eta\left[4\gamma e\exp(-\gamma e^2)\left(p'_{uf} + q'_{if}\right) + \lambda h_{\mathcal{C}_n f}\right] \tag{17}$$

$$\alpha_{u\mathcal{C}_n} \leftarrow \alpha_{u\mathcal{C}_n} - \eta\left[4\gamma e\exp(-\gamma e^2) + \lambda|N_u^{(I)}|^{-\frac{1}{2}}\alpha_{u\mathcal{C}_n}\right] \tag{18}$$

$$\beta_{i\mathcal{C}_n} \leftarrow \beta_{i\mathcal{C}_n} - \eta\left[4\gamma e\exp(-\gamma e^2) + \lambda|N_i^{(U)}|^{-\frac{1}{2}}\beta_{i\mathcal{C}_n}\right] \tag{19}$$

$$y_{jf} \leftarrow y_{jf} - \eta\left[4\gamma e\exp(-\gamma e^2)|N_u^{(I)}|^{-\frac{1}{2}}\left(q'_{if} + \sum_{n=1}^{L}h_{\mathcal{C}_n f}\right) + \lambda|N_u^{(I)}|^{-\frac{1}{2}}y_{jf}\right] \tag{20}$$

$$g_{vf} \leftarrow g_{vf} - \eta\left[4\gamma e\exp(-\gamma e^2)|N_i^{(U)}|^{-\frac{1}{2}}\left(p'_{uf} + \sum_{n=1}^{L}h_{\mathcal{C}_n f}\right) + \lambda|N_i^{(U)}|^{-\frac{1}{2}}g_{vf}\right] \tag{21}$$

Algorithm 2: Rating Prediction

Input : $\mathcal{K}'$ with unknown ratings, test set $\mathcal{D}_{te}$, trained model parameters $\boldsymbol{\theta}$
Output: Dense rating tensor, RMSE, MAE
Obtain: set of tuples $\mathcal{K} \setminus \mathcal{D}_{tr}$ to predict ratings; this includes $\mathcal{K}'$ and $\mathcal{D}_{te}$
for $(u, i, \mathbf{c}) \in \mathcal{D}_{te}$ **do**
  Predict rating $\hat{r}_{ui\mathbf{c}}$ using Eq. (5)
end for
Evaluate RMSE and MAE on test dataset $\mathcal{D}_{te}$
Predict ratings for tuples in $\mathcal{K}'$ using Eq. (5)

The time complexity of CAMF for computing the loss $\mathcal{S}$ over one epoch is $\mathcal{O}\Big(|\mathcal{D}_{tr}| \times (F + L)\Big)$ where, $|\mathcal{D}_{tr}|$ is the number of tuples in the training dataset. For the proposed method of KCAMF, during each epoch, the time complexity for computing the gradients $\frac{\partial \mathcal{S}}{\partial p_{uf}}$, $\frac{\partial \mathcal{S}}{\partial q_{if}}$, $\frac{\partial \mathcal{S}}{\partial h_{\mathcal{C}nf}}$, $\frac{\partial \mathcal{S}}{\partial \alpha_{u\mathcal{C}n}}$, $\frac{\partial \mathcal{S}}{\partial \beta_{i\mathcal{C}n}}$, $\frac{\partial \mathcal{S}}{\partial y_{jf}}$, and $\frac{\partial \mathcal{S}}{\partial g_{yf}}$ is $\mathcal{O}\Big(|\mathcal{D}_{tr}| \times (F + L)\Big)$, $\mathcal{O}\Big(|\mathcal{D}_{tr}| \times (F + L)\Big)$, $\mathcal{O}\Big(|\mathcal{D}_{tr}| \times (F + L)\Big)$, $\mathcal{O}\Big(|\mathcal{D}_{tr}| \times L\Big)$, $\mathcal{O}\Big(|\mathcal{D}_{tr}| \times L\Big)$, $\mathcal{O}\Big(|\mathcal{D}_{tr}| \times (F + L) \times \bar{j}\Big)$, and $\mathcal{O}\Big(|\mathcal{D}_{tr}| \times (F + L) \times \bar{v}\Big)$, respectively, where $\bar{j}$ is the average number of items rated by users and $\bar{v}$ is the average number of users who rated items.

The overall time complexity of KCAMF for each epoch is thus $\mathcal{O}\Big(|\mathcal{D}_{tr}| \times (F + L) \times \max(\bar{j}, \bar{v})\Big)$. Since typically, $L \ll F$, the time complexity of KCAMF for T epochs is $\mathcal{O}\Big(T \times |\mathcal{D}_{tr}| \times F \times \max(\bar{j}, \bar{v})\Big)$, which is linear relative to the size of the training dataset. Compared to the time complexity of CAMF for T epochs of $\mathcal{O}\Big(T \times |\mathcal{D}_{tr}| \times F\Big)$, the overall time complexity of KCAMF is only marginally higher since $\bar{j} \ll |\mathcal{D}_{tr}|$ and $\bar{v} \ll |\mathcal{D}_{tr}|$.

V. EXPERIMENTAL RESULTS

We use five benchmark context-aware datasets to evaluate the performance of the proposed KCAMF model and provide a comparison with existing state-of-the-art methods for solving the rating prediction problem.

A. DATASETS

The context-aware datasets used in this work are InCarMusic [35] and TripAdvisor [36], DePaulMovie [37], LDos Comoda [38] and Frappe [39] having varying number of context factors and conditions. All these datasets belong to varied domains such as music, travel, movie rating, and mobile app usage, and can be categorized as small, medium and large datasets. The InCarMusic dataset developed for music recommendations contains ratings of vehicle drivers for songs of 10 genres based on various contexts of driving and traffic scenarios. Even though the InCarMusic dataset has a low number of 4012 ratings, 42 users, and 139 items (songs), it has more context factors including driving style, sleepiness, road type, landscape, mood, traffic conditions, natural phenomena, and weather; the total number of context conditions for this dataset is 34. The TripAdvisor dataset consists of 14,175 ratings given by 2,371 users for 2,269 hotels based on one context factor, i.e. trip type with five context conditions including business, solo, family, couples, and friends. The DePaulMovie dataset contains 5,043 ratings of 79 movies given by 97 students based on three context factors of time, location, and companion with a total of 10 context conditions.

The LDos Comoda dataset is context-rich with 2,294 ratings given by 121 users on 1232 items under 12 different contextual factors, namely time, daytime, season, location, weather, social, endEmo, dominantEmo, mood, physical, decision, and interaction resulting in 61 contextual conditions. The Frappe dataset monitors the usage of mobile applications over a duration of two months by 957 users for 4,082 different apps. As per [40], we apply the log transformation on the frequency of usage to convert it to the rating scale of 0 to 4.46, and use 22 contextual conditions obtained from three contexts of daytime, weekday and weather.

A summary of these five datasets is given in Table 2.

TABLE 2. Context-aware datasets used in this work.

Dataset	#Users	#Items	#Ratings	Rating scale	Contextual factors	Contextual Conditions
InCarMusic	42	139	4,012	1-5	8	34
TripAdvisor	2,371	2,269	14,175	1-5	1	5
DePaulMovie	97	79	5,043	1-5	3	10
LDos Comoda	121	1232	2,294	1-5	12	61
Frappe	957	4,082	96,203	0-4.46	3	22

B. METHODS FOR COMPARISON

The proposed work is compared with the following conventional and modern methods:

(a) Traditional: The Biased matrix factorization method [22] models only the global mean, user bias and item bias for predicting the ratings as $\hat{r}_{ui\mathbf{c}} = \mu + \alpha_u + \beta_i$ and ignores the contexts. The SVD++ method [41] models user latent features $\mathbf{p}_u$, item latent features $\mathbf{q}_i$ and adds user implicit feedback for updating the user latent feature vector to $\mathbf{p}'_u$ by incorporating the implicit preferences of users. However, it ignores the item implicit feedback which we have incorporated in KCAMF.

(b) PMF: The Probabilistic Matrix Factorization (PMF) method [42] models rating as a conditional distribution using Gaussian random variables with priors on the user and item latent feature vectors. PMF maximizes the log-posterior which is equivalent to the minimization of the least square error loss function by including the regularization terms. However, PMF is computationally expensive and also ignores the contexts.

(c) NMF: The Non-negative Matrix Factorization (NMF) method [43] imposes the constraint that the user and

item latent features should not be negative, which is achieved by re-scaling the learning rate to discard the negative components while updating each learnable parameter. Like PMF, the NMF method has a high computational cost and also ignores the contexts.

(d) CPTF: The Candecomp/Parafac (CP) tensor factorization method [44], which is the tensor equivalent form of SVD, factorizes the entire tensor to a sum of rank-one tensors using outer products. CPTF models the contexts but it is computationally quite expensive compared to matrix factorization.

(e) KMR/KCR: The Kernel Mapping Recommender (KMR) method [45] determines a non-linear mapping between the vector spaces representing the rated items and a probability density for user's preferences. The KMR method was extended in [17] to utilize the context information resulting in Kernel Context Recommender (KCR) system, which makes use of various context kernels depending on the dataset and combine them with the rating prediction kernel.

(f) CUBCF/CIBCF: In [18], the similarity between contexts with respect to users was computed to obtain the CUBCF (context-similarity user-based collaborative filtering) system where the similarity was measured using the Chi-square similarity kernel. A similar strategy was adopted to obtain the similarity between contexts with respect to items to obtain the item-based CIBCF system.

(g) IHSR: An extension of matrix factorization using hierarchical user and item structures was proposed in [20]. The latent context representation is learned in an unstructured manner using auto-encoder and in a structured manner using agglomerative hierarchical clustering. The Hybrid IHSR model integrates both latent context representations to obtain the predicted ratings.

(h) ND_s-DCW: An empirical comparison of various context similarity approaches was done in [19] and it was found that the non-dominated simplified differential context weighting method performed better than the other methods and hence we use this method for comparison.

(i) CAMF and ICAMF: These are the context-aware baseline methods based on matrix factorization as discussed in detail in Sec. III.

C. EVALUATION PROTOCOL

The sparse rating tensor $\mathcal{K}$ is split into the set of known ratings $\mathcal{D}$ and the set of unknown ratings $\mathcal{K}'$. The set $\mathcal{D}$ is split in the ratio of 80% : 20% to obtain the training set $\mathcal{D}_{tr}$ and the test set $\mathcal{D}_{te}$. For each existing and the proposed method, the training dataset is split into train and validation sets. The best hyper-parameter λ is obtained through a grid-search using the values [0.001, 0.01, 0.1, 1, 10, 100, 1000]. The models are trained using each possible value of λ and the RMSE scores are computed on the validation set. The best λ is chosen as the one resulting in the least RMSE score and this value is used to again train the model with the entire training dataset $\mathcal{D}_{tr}$ and the model performance is reported for the test dataset $\mathcal{D}_{te}$.

The learning rate δ is initialised to 0.01 and as the training progresses, the learning rate is adaptively updated using Eq. (22), where δ_t, $\mathcal{S}_t$ and δ_{t-1}, $\mathcal{S}_{t-1}$ refers to the learning rate and loss in t^{th} iteration and $(t-1)^{\text{th}}$ iteration, respectively. If the previous loss is more than the current loss (i.e. the error has reduced), then the learning rate is increased by 5%, otherwise the learning rate is drastically reduced by 50%.

$$\delta_t = \begin{cases} \delta_{t-1} * 1.05 & \text{if } |\mathcal{S}_{t-1}| > |\mathcal{S}_t| \\ \delta_{t-1} * 0.5 & \text{otherwise} \end{cases} \tag{22}$$

The prediction performance of each model is quantified using the metrics of Root Mean Square Error (*RMSE*), and Mean Absolute Error (*MAE*) on the test dataset, given by Eq. (23). Both metrics measure the difference between the actual ratings $r_{ui\mathbf{c}}$ and the predicted ratings $\hat{r}_{ui\mathbf{c}}$ for all user, item, context tuples in $\mathcal{D}_{te}$.

$$\begin{aligned} RMSE &= \sqrt{\frac{1}{|\mathcal{D}_{te}|} \sum_{u,i,\mathbf{c} \in \mathcal{D}_{te}} (r_{ui\mathbf{c}} - \hat{r}_{ui\mathbf{c}})^2} \\ MAE &= \frac{1}{|\mathcal{D}_{te}|} \sum_{u,i,\mathbf{c} \in \mathcal{D}_{te}} |(r_{ui\mathbf{c}} - \hat{r}_{ui\mathbf{c}})| \end{aligned} \tag{23}$$

To evaluate the robustness of algorithms against malicious attacks, the prediction shift given by Eq. (24) is computed for all user, item, context tuples in the test dataset, where $\hat{r}'_{ui\mathbf{c}}$ is the post-attack predicted rating and $\hat{r}_{ui\mathbf{c}}$ is the pre-attack predicted rating. Lower prediction shift implies that the method is resistant to various malicious/shilling attacks on the recommender system [46].

$$PS = \frac{1}{|\mathcal{D}_{te}|} \sum_{u,i,\mathbf{c} \in \mathcal{D}_{te}} |\hat{r}'_{ui\mathbf{c}} - \hat{r}_{ui\mathbf{c}}| \tag{24}$$

D. RESULTS AND DISCUSSION

1) COMPARATIVE ANALYSIS

The proposed KCAMF method for solving the rating prediction problem is evaluated on five benchmark datasets mentioned in Sec. V-A and compared with several comparative methods detailed in Sec. V-B using the evaluation protocol mentioned in Sec. V-C.

Table 3 shows the comparative results of KCAMF with existing methods using RMSE and MAE metrics with the latent feature dimensionality $F = 80$. It can be observed that SVD++ has an advantage of improving the scores over Biased MF, PMF, NMF, and CPTF due to the inclusion of user implicit feedback. The results reported in the existing work of KMR, KCR, CUBCF, CIBCF, IHSR and NDs-DCW for specific datasets are mentioned. KMR and KCR gives a better performance than the standard baseline methods for the movie datasets. The CUBCF and CIBCF methods result in a slightly higher RMSE since it is based only on context similarity and does not capture the latent user, item and

TABLE 3. Performance evaluation of proposed method and comparison with existing methods.

	InCarMusic		TripAdvisor		DePaulMovie		LDos Comoda		Frappe	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
Biased MF [22]	1.0507	1.2746	0.8667	1.0759	0.9685	1.1778	0.8509	1.0501	0.5773	0.7313
SVD++ [41]	0.6443	0.9702	0.8003	1.0157	0.7158	1.0591	0.8468	1.0571	0.5231	0.6562
PMF [42]	0.6554	1.0226	2.0322	2.1934	0.7261	1.0753	2.5532	2.7918	0.5384	0.6789
NMF [43]	0.6569	1.0211	0.9489	1.2426	0.7132	1.0574	0.9453	1.2451	0.5189	0.6536
CPTF [44]	1.4837	1.7934	1.0259	1.3534	1.2457	1.5987	0.8580	-	0.3920	-
KMR [45]	-	-	-	-	-	1.1028	-	0.9402	-	-
KCR [17]	-	-	-	-	-	1.0579	-	**0.8712**	-	-
CUBCF [18]	1.0580	1.2878	-	-	1.3915	1.9457	-	-	-	-
CIBCF [18]	1.2494	1.4285	-	-	1.1675	1.4556	-	-	-	-
Hybrid-IHSR [20]	-	-	-	-	-	-	0.9700	1.2300	0.5100	0.6860
ND_s-DCW [19]	-	1.0480	-	-	-	-	0.7260	-	0.3790	-
CAMF [21]	0.6399	0.9628	0.8038	1.0191	0.6821	0.9431	0.7844	0.9933	0.4248	0.5471
ICAMF [34]	0.6305	0.9591	0.7528	0.9372	0.6783	0.9284	0.7781	0.9859	0.4189	0.5403
Proposed (KCAMF)	**0.5893**	**0.9114**	**0.6701**	**0.9037**	**0.6573**	**0.9002**	**0.7177**	0.9136	**0.3743**	**0.5179**
% imp. over CAMF	7.9	5.3	16.6	11.3	3.6	4.5	8.5	8.1	11.8	5.3
% imp. over ICAMF	6.5	4.9	10.9	3.6	3.1	3.0	7.7	7.3	10.6	4.1

context interactions. The IHSR method performs better than the PMF and NMF methods due to its learning of latent context features but results in higher RMSE compared to other recent methods. The method of ND_s-DCW obtains a lower MAE relative to all other existing methods for three datasets, but it does not capture the latent interactions of users, items and contexts. The scores are improved with CAMF and ICAMF since these methods model the context interactions for predicting the rating. KCAMF method outperforms all existing works and has a significant improvement in both RMSE and MAE metrics over CAMF, ICAMF, and other existing methods. Since the TripAdvisor dataset has only one context (trip type), the improvement due to KCAMF is significantly higher than for other datasets having more context factors.

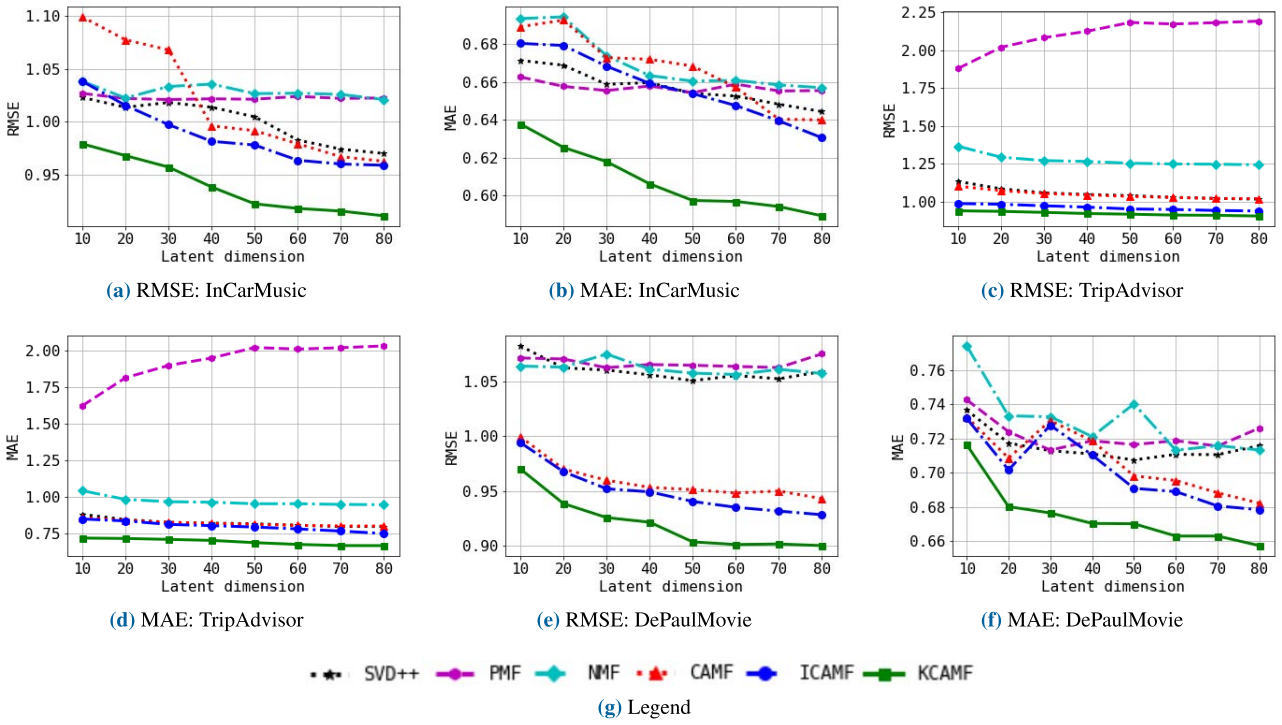
Fig. 7 compares the RMSE and MAE performance of the existing methods and KCAMF for varying latent feature dimensionality F from 10 to 80. It can be observed that for almost all methods, the scores converge after $F = 60$. CAMF and ICAMF have better performance compared to Biased MF, SVD++, PMF, and NMF even at lower F due to explicitly incorporating the contexts. KCAMF outperforms all existing methods for all values of F and the performance improves with increasing F. The advantage of KCAMF is observed for the InCarMusic dataset which has more context factors than other datasets, since KCAMF models user, item, and context interactions effectively using both user and item implicit feedbacks with the kernel loss function.

2) ABLATION STUDY

We further conduct an ablation study with $F = 80$ to determine the impact of each component of KCAMF as shown in Fig. 2. The components of KCAMF analyzed are the kernel loss function (KLF), user/item implicit feedbacks (IFU / IFI), both user and item implicit feedbacks (IFUI), and weighted regularization (WR). Table 4 shows the results of the ablation study of KCAMF for $F = 80$ and the comparison with both CAMF and ICAMF over all five context-aware datasets viz. InCarMusic, TripAdvisor, DePaulMovie, LDos Comoda, and Frappe dataset. With only the kernel loss function (3^{rd} row), KCAMF improves the rating prediction performance on all five datasets. By using only weighted regularization (4^{th} row) without implicit feedback and using the traditional least square loss function, the performance is better than both CAMF and ICAMF, since the regularization terms for the latter methods use the same hyper-parameter λ, whereas KCAMF applies different weights on each regularization term. Further, using the kernel loss function and weighted regularization (5^{th} row) without the implicit feedback, the performance still improves primarily due to the properties of the kernel loss function as discussed in IV-B. By modeling the user and item implicit feedbacks, we observe that there is a marginal improvement when learning the item implicit feedback over the user implicit feedback. Finally, using all the components of KCAMF, the rating prediction performance improves significantly over both CAMF and ICAMF methods. This indicates that combining the kernel loss function with user and item implicit feedbacks and weighted regularization improves the overall performance relative to existing work.

3) ROBUSTNESS TO ATTACKS

To determine the robustness of the proposed KCAMF method, shilling attacks are inserted in the dataset and the



(a) RMSE: InCarMusic (b) MAE: InCarMusic (c) RMSE: TripAdvisor

(d) MAE: TripAdvisor (e) RMSE: DePaulMovie (f) MAE: DePaulMovie

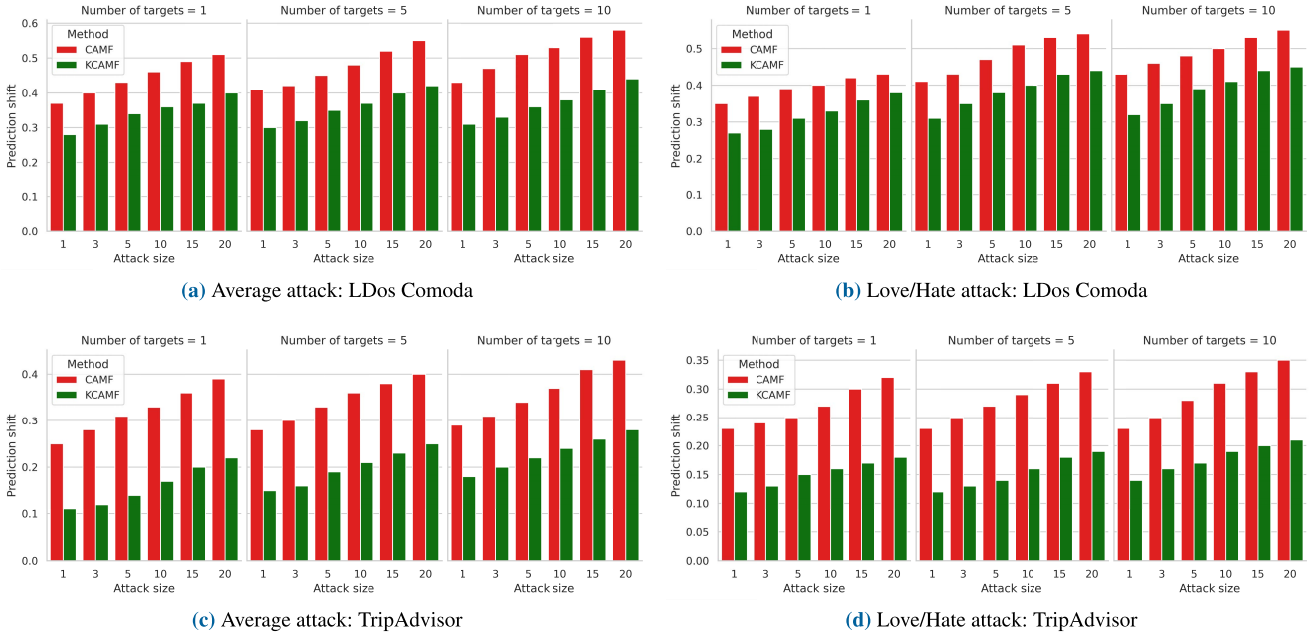
(g) Legend

FIGURE 7. Performance of existing methods and KCAMF as a function of latent feature dimensionality. The KCAMF results are statistically significant as measured with the *paired t-test* at the significance level of 5%.

TABLE 4. Results of ablation study for $F = 80$. Here, KLF: kernel loss function, IF: implicit feedback, IFU: user implicit feedback, IFI: item implicit feedback, WR: weighted regularization.

	InCarMusic		TripAdvisor		DePaulMovie		LDos Comoda		Frappe	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
CAMF	0.6399	0.9628	0.8038	1.0191	0.6821	0.9431	0.7844	0.9933	0.4248	0.5471
ICAMF	0.6305	0.9591	0.7528	0.9372	0.6783	0.9284	0.7781	0.9859	0.4189	0.5403
KCAMF with KLF+, IF-, WR-	0.6196	0.9527	0.6973	0.9304	0.6772	0.9116	0.7629	0.9732	0.4105	0.5386
KCAMF with KLF-, IF-, WR+	0.6075	0.9405	0.6915	0.9202	0.6659	0.9084	0.7613	0.9651	0.4037	0.5392
KCAMF with KLF+, IF-, WR+	0.6005	0.9261	0.6794	0.9101	0.6614	0.9013	0.7552	0.9473	0.3962	0.5329
KCAMF with KLF+, IFU+, WR+	0.5955	0.9208	0.6748	0.9093	0.6603	0.9011	0.7381	0.9389	0.3908	0.5271
KCAMF with KLF+, IFI+, WR+	0.5904	0.9184	0.6713	0.9058	0.6591	0.9007	0.7294	0.9216	0.3847	0.5195
KCAMF with KLF+, IFUI+, WR+	**0.5893**	**0.9114**	**0.6701**	**0.9037**	**0.6573**	**0.9002**	**0.7177**	**0.9136**	**0.3743**	**0.5179**

performance is evaluated using the prediction shift. Various shilling attacks are suggested in the literature for targeting specific items by attackers [46]. Here, we implement two shilling attacks, namely, Average attack for pushing the rating of target items to the maximum scale and Love/Hate attack for nuking the rating of target items to the minimum scale. These attacks are intended to disrupt the learning behavior of the recommender systems so that the target item either gets promoted or demoted for end-users. However, to avoid easier detection of such attacks, profile injection is done in the dataset, where the profile consists of fake users (malicious attackers), the items in the target set $\mathcal{A}_T$, and the items in the filler set $\mathcal{A}_F$. The rationale for using the filler set is to trick the learning model to consider the attackers as genuine users. While the profile injection for shilling attacks also involve the set of selected items $\mathcal{A}_S$, this is ignored in this work since this set is not required for the Average and Love/Hate attacks. The shilling attacks are implemented on LDos Comoda dataset, which is medium-sized but context-rich, and TripAdvisor dataset, which is large-sized but has only one context dimension. We consider $|\mathcal{A}_F| = 1\%$, i.e. the size of filler item set is 12 for LDos Comoda dataset and 22 for TripAdvisor dataset. For both datasets, we use three sizes of target items as $|\mathcal{A}_T| = 1, 5, 10$, where all attackers attack the same target item(s). Further, we vary the attack size as $1\%, 3\%, 5\%, 10\%, 15\%, 20\%$ of the total number of actual



(a) Average attack: LDos Comoda

(b) Love/Hate attack: LDos Comoda

(c) Average attack: TripAdvisor

(d) Love/Hate attack: TripAdvisor

FIGURE 8. **Prediction shift of learning models against malicious user attacks.**

TABLE 5. **Paired t-test for RMSE comparison of CAMF with KCAMF.**

InCarMusic		TripAdvisor		DePaulMovie		LDos Comoda		Frappe	
CAMF	KCAMF	CAMF	KCAMF	CAMF	KCAMF	CAMF	KCAMF	CAMF	KCAMF
0.9628	0.9114	1.1091	0.9037	0.9431	0.9002	0.9933	0.9136	0.5471	0.5179
0.9562	0.9216	1.1146	0.9092	0.9489	0.9018	0.9826	0.9261	0.5503	0.5038
0.9672	0.9167	1.1832	0.9113	0.9421	0.9105	0.9774	0.9158	0.5492	0.5127
0.9663	0.9158	1.0057	0.9008	0.9436	0.9011	0.9872	0.9027	0.5529	0.5081
0.9529	0.9195	1.1929	0.9123	0.9503	0.9009	0.9905	0.9274	0.5448	0.5192
$p = 4.3 \times 10^{-4}$		$p = 2.4 \times 10^{-3}$		$p = 1.5 \times 10^{-4}$		$p = 2.3 \times 10^{-4}$		$p = 9.1 \times 10^{-4}$	
Reject H_0		Reject H_0		Reject H_0		Reject H_0		Reject H_0	

users in the dataset. The items in $\mathcal{A}_F$ are chosen randomly for both type of attacks and their rating value is average rating of specific items for Average attack and maximum rating for Love/Hate attack. The items in $\mathcal{A}_T$ are chosen such that their rating value is below the global mean for Average attack and above the global mean for Love/Hate attack. The rating value of the items in $\mathcal{A}_T$ is maximum for Average (push) attack and minimum for Love/Hate (nuke) attack. Fig. 8 illustrates the prediction shift obtained by CAMF as well as the proposed KCAMF methods for both type of shilling attacks and for various attack sizes with varying number of target items. We observe that KCAMF is relatively resistant to shilling attacks due to lower prediction shift, i.e. the rating prediction value does not change drastically even after the attack profiles are injected in the datasets.

4) STATISTICAL SIGNIFICANCE TESTING

Next, we conducted a statistical significance testing using paired t-test at the significance level of 5% to verify if the results obtained by KCAMF are statistically significant over other methods. The null hypothesis is H_0: there is no statistically significant difference between KCAMF and the existing method under consideration, and the alternative hypothesis is H_1: there is indeed a statistically significant difference. The paired t-test is conducted between KCAMF and CAMF as well as between KCAMF and ICAMF using the RMSE metric. Each experiment is repeated five times and the resulting mean RMSE values of the two methods are measured as M_{A_1} and M_{A_2} and the t value is calculated using Eq. (25), σ is given by Eq. (26) where σ^2 denotes the variance of the two scores, n_{A_1} and n_{A_2} are the number of samples (here, 5).

$$t = \frac{M_{A_1} - M_{A_2}}{\sigma_{M_{A_1} - M_{A_2}}} \tag{25}$$

$$\sigma_{M_{A_1} - M_{A_2}} = \sqrt{\frac{\sigma_{A_1}^2}{n_{A_1}} + \frac{\sigma_{A_2}^2}{n_{A_2}}} \tag{26}$$

To compute the t value, the degree of freedom for the paired t-test is determined using Eq. (27). Using the *dof*, the

TABLE 6. Paired t-test for RMSE comparison of ICAMF with KCAMF.

InCarMusic		TripAdvisor		DePaulMovie		LDos Comoda		Frappe	
ICAMF	KCAMF	ICAMF	KCAMF	ICAMF	KCAMF	ICAMF	KCAMF	ICAMF	KCAMF
0.9591	0.9114	0.9372	0.9037	0.9284	0.9002	0.9859	0.9136	0.5403	0.5179
0.9604	0.9216	0.9467	0.9092	0.9307	0.9018	0.9806	0.9261	0.5471	0.5038
0.9518	0.9167	0.9336	0.9113	0.9291	0.9105	0.9935	0.9158	0.5506	0.5127
0.9569	0.9158	0.9289	0.9008	0.9269	0.9011	0.9861	0.9027	0.5368	0.5081
0.9603	0.9195	0.9391	0.9123	0.9205	0.9009	0.9782	0.9274	0.5417	0.5192
$p = 3.8 \times 10^{-5}$		$p = 3.6 \times 10^{-4}$		$p = 3.5 \times 10^{-4}$		$p = 4.6 \times 10^{-4}$		$p = 1.8 \times 10^{-3}$	
Reject H_0		Reject H_0		Reject H_0		Reject H_0		Reject H_0	

t value is obtained using the standard t-test table from which the corresponding p-value can be found. If the determined p-value is less than 0.05, then the null hypothesis can be rejected indicating that the difference between the two methods A_1 and A_2 is not due to chance and there is indeed a statistically significant difference between the two; otherwise, the null hypothesis has to be accepted.

$$dof = \frac{(\sigma_{A_1}^2/n_{A_1} + \sigma_{A_2}^2/n_{A_2})^2}{(\sigma_{A_1}^2/n_{A_1})^2/(n_{A_1}-1) + (\sigma_{A_2}^2/n_{A_2})^2/(n_{A_2}-1)} \tag{27}$$

Tables 5 and 6 shows the results of paired t-test between A_1 = CAMF, A_2 = KCAMF, and A_1 = ICAMF, A_2 = KCAMF, respectively. In both cases, the proposed method of KCAMF performs better since the p-value obtained is less than 0.05 for all five datasets, thus H_0 can be rejected.

VI. CONCLUSION

In this work, we proposed a novel method for rating prediction of context-aware recommender systems as an improvement over the existing techniques. The item implicit feedback is also learned besides the conventional user implicit feedback resulting in better predictions as shown in the ablation studies. The kernel loss objective function is proposed and its properties are exploited to obtain fewer errors during the optimization compared to the least-squares loss function. The kernel loss function is shown to be robust against various shilling attacks. Also, weighted regularization is proposed to reduce the impact of malicious users who may intentionally give incorrect ratings and also tackle the cold-start problem in recommender systems. The experimental evaluation validated the performance of the proposed method over the baseline and existing methods, and a detailed ablation study demonstrated the impact of each enhancement of the proposed method. For further work, we will extend this method for solving the recommendation problem by applying suitable ranking techniques and generating a list of recommended items personalized to each user.

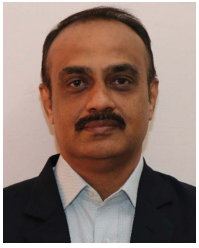
REFERENCES

[1] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin, "Incorporating contextual information in recommender systems using a multidimensional approach," *ACM Trans. Inf. Syst.*, vol. 23, no. 1, pp. 103–145, Jan. 2005.
[2] Z. Y. Khan, Z. Niu, S. Sandiwarno, and R. Prince, "Deep learning techniques for rating prediction: A survey of the state-of-the-art," *Artif. Intell. Rev.*, vol. 54, no. 1, pp. 95–135, Jan. 2021.
[3] X. Xin, B. Chen, X. He, D. Wang, Y. Ding, and J. Jose, "CFM: Convolutional factorization machines for context-aware recommendation," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 3926–3932.
[4] Q. Wang, B. Peng, X. Shi, T. Shang, and M. Shang, "DCCR: Deep collaborative conjunctive recommender for rating prediction," *IEEE Access*, vol. 7, pp. 60186–60198, 2019.
[5] M. Ahmadian, M. Ahmadi, S. Ahmadian, S. M. J. Jalali, A. Khosravi, and S. Nahavandi, "Integration of deep sparse autoencoder and particle swarm optimization to develop a recommender system," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2021, pp. 2524–2530.
[6] S. Ahmadian, N. Joorabloo, M. Jalili, and M. Ahmadian, "Alleviating data sparsity problem in time-aware recommender systems using a reliable rating profile enrichment approach," *Expert Syst. Appl.*, vol. 187, Jan. 2022, Art. no. 115849.
[7] M. Unger, A. Bar, B. Shapira, and L. Rokach, "Towards latent context-aware recommendation systems," *Knowl.-Based Syst.*, vol. 104, pp. 165–178, Jul. 2016.
[8] J. Zhao, W. Wang, Z. Zhang, Q. Sun, H. Huo, L. Qu, and S. Zheng, "TrustTF: A tensor factorization model using user trust and implicit feedback for context-aware recommender systems," *Knowl.-Based Syst.*, vol. 209, Dec. 2020, Art. no. 106434.
[9] S. Shukla, I. Kalsi, A. Jain, and A. Verma, "A tensor decomposition based approach for context-aware recommender systems (CARS)," in *Proc. 13th Int. Conf. Contemp. Comput. (IC3)*, Aug. 2021, pp. 438–442.
[10] X. Wu, B. Shi, Y. Dong, C. Huang, and N. V. Chawla, "Neural tensor factorization for temporal interaction learning," in *Proc. 12th ACM Int. Conf. Web Search Data Mining*, Jan. 2019, pp. 537–545.
[11] J. Fan, "Multi-mode deep matrix and tensor factorization," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–25.
[12] H. Tahmasbi, M. Jalali, and H. Shakeri, "TSCMF: Temporal and social collective matrix factorization model for recommender systems," *J. Intell. Inf. Syst.*, vol. 56, no. 1, pp. 169–187, Feb. 2021.
[13] K. Davagdorj, K. Park, and K. Ryu, "A collaborative filtering recommendation system for rating prediction," in *Advances in Intelligent Information Hiding and Multimedia Signal Processing*. Singapore: Springer, 2020, pp. 265–271.
[14] Y. Li and K. Mu, "Matrix factorization model with dual preferences for rating prediction," in *Proc. IEEE 43rd Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, vol. 1, Jul. 2019, pp. 364–372.
[15] Q. Zhang, J. Lu, D. Wu, and G. Zhang, "A cross-domain recommender system with kernel-induced knowledge transfer for overlapping entities," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 7, pp. 1998–2012, Jul. 2019.
[16] B. Zeng, Q. Shang, X. Han, F. Zeng, and M. Zhang, "RACMF: Robust attention convolutional matrix factorization for rating prediction," *Pattern Anal. Appl.*, vol. 22, no. 4, pp. 1655–1666, Nov. 2019.
[17] M. Iqbal, M. A. Ghazanfar, A. Sattar, M. Maqsood, S. Khan, I. Mehmood, and S. W. Baik, "Kernel context recommender system (KCR): A scalable context-aware recommender system algorithm," *IEEE Access*, vol. 7, pp. 24719–24737, 2019.
[18] H. X. Huynh, N. Q. Phan, N. M. Pham, V.-H. Pham, L. H. Son, M. Abdel-Basset, and M. Ismail, "Context-similarity collaborative filtering recommendation," *IEEE Access*, vol. 8, pp. 33342–33351, 2020.
[19] Y. Zheng, "Context-aware collaborative filtering using context similarity: An empirical comparison," *Information*, vol. 13, no. 1, pp. 1–18, 2022.

[20] M. Unger and A. Tuzhilin, "Hierarchical latent context representation for context-aware recommendations," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 7, pp. 3322–3334, Jul. 2022.
[21] L. Baltrunas, B. Ludwig, and F. Ricci, "Matrix factorization techniques for context aware recommendation," in *Proc. 5th ACM Conf. Recommender Syst. (RecSys)*, 2011, pp. 301–304.
[22] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *IEEE Comput.*, vol. 42, no. 8, pp. 30–37, Aug. 2009.
[23] S. Anyosa, J. Vinagre, and A. Jorge, "Incremental matrix co-factorization for recommender systems with implicit feedback," in *Proc. Companion Web Conf.*, 2018, pp. 1413–1418.
[24] Q. Zhao, F. Harper, G. Adomavicius, and J. Konstan, "Explicit or implicit feedback? Engagement or satisfaction? A field experiment on machine-learning-based recommender systems," in *Proc. 33rd Annu. ACM Symp. Appl. Comput.*, 2018, pp. 1331–1340.
[25] M. Cheng, F. Yuan, Q. Liu, S. Ge, Z. Li, R. Yu, D. Lian, S. Yuan, and E. Chen, "Learning recommender systems with implicit feedback via soft target enhancement," in *Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2021, pp. 575–584.
[26] B. Yi, X. Shen, H. Liu, Z. Zhang, W. Zhang, S. Liu, and N. Xiong, "Deep matrix factorization with implicit feedback embedding for recommendation system," *IEEE Trans. Ind. Informat.*, vol. 15, no. 8, pp. 4591–4601, Aug. 2019.
[27] S. Chen and Y. Peng, "Matrix factorization for recommendation with explicit and implicit feedback," *Knowl.-Based Syst.*, vol. 158, pp. 109–117, Oct. 2018.
[28] Y. Gu, X. Yang, M. Peng, and G. Lin, "Robust weighted SVD-type latent factor models for rating prediction," *Expert Syst. Appl.*, vol. 141, Mar. 2020, Art. no. 112885.
[29] M. Sharma and G. Karypis, "Adaptive matrix completion for the users and the items in tail," in *Proc. World Wide Web Conf. (WWW)*, 2019, pp. 3223–3229.
[30] K. Li, X. Zhou, F. Lin, W. Zeng, B. Wang, and G. Alterovitz, "Sparse online collaborative filtering with dynamic regularization," *Inf. Sci.*, vol. 505, pp. 535–548, Dec. 2019.
[31] W. Shi, L. Wang, and J. Qin, "User embedding for rating prediction in SVD++-based collaborative filtering," *Symmetry*, vol. 12, no. 1, pp. 1–14, 2020.
[32] H. Zhang, I. Ganchev, N. S. Nikolov, and M. Stevenson, "UserReg: A simple but strong model for rating prediction," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2021, pp. 3595–3599.
[33] H.-H. Chen and P. Chen, "Differentiating regularization weights—A simple mechanism to alleviate cold start in recommender systems," *ACM Trans. Knowl. Discovery Data*, vol. 13, no. 1, pp. 1–22, Feb. 2019.
[34] J. Li, P. Feng, and J. Lv, "ICAMF: Improved context-aware matrix factorization for collaborative filtering," in *Proc. IEEE 25th Int. Conf. Tools Artif. Intell.*, Nov. 2013, pp. 63–67.
[35] L. Baltrunas, M. Kaminskas, B. Ludwig, O. Moling, F. Ricci, A. Aydin, K. Lüke, and R. Schwaiger, "InCarMusic: Context-aware music recommendations in a car," in *E-Commerce and Web Technologies*. Berlin, Germany: Springer, 2011, pp. 89–100.
[36] Y. Zheng, B. Mobasher, and R. Burke, "Context recommendation using multi-label classification," in *Proc. IEEE/WIC/ACM Int. Joint Conf. Web Intell. (WI) Intell. Agent Technol. (IAT)*, vol. 2, Aug. 2014, pp. 288–295.
[37] Y. Zheng, B. Mobasher, and R. Burke, "CARSKit: A Java-based context-aware recommendation engine," in *Proc. IEEE Int. Conf. Data Mining Workshop (ICDMW)*, Nov. 2015, pp. 1668–1671.
[38] A. Kosir, A. Odic, M. Kunaver, M. Tkalcic, and J. F. Tasic, "Database for contextual personalization," *Elektrotehniski Vestnik*, vol. 78, no. 5, pp. 270–274, 2011.
[39] L. Baltrunas, K. Church, A. Karatzoglou, and N. Oliver, "Frappé: Understanding the usage and perception of mobile app recommendations in-the-wild," 2015, *arXiv:1505.03014*.
[40] M. Unger, A. Tuzhilin, and A. Livne, "Context-aware recommendations based on deep learning frameworks," *ACM Trans. Manage. Inf. Syst.*, vol. 11, no. 2, pp. 1–15, Jun. 2020.
[41] Y. Koren, "Factorization meets the neighborhood: A multifaceted collaborative filtering model," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2008, pp. 426–434.
[42] R. Salakhutdinov and A. Mnih, "Bayesian probabilistic matrix factorization using Markov chain Monte Carlo," in *Proc. 25th Int. Conf. Mach. Learn. (ICML)*, 2008, pp. 880–887.
[43] X. Luo, M. Zhou, Y. Xia, and Q. Zhu, "An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1273–1284, May 2014.
[44] Y. Kolda, G. Tamara, X. Bader, and W. Brett, "Tensor decomposition and applications," *SIAM Rev.*, vol. 51, no. 3, pp. 455–500, 2009.
[45] M. A. Ghazanfar, A. Prügel-Bennett, and S. Szedmak, "Kernel-mapping recommender system algorithms," *Inf. Sci.*, vol. 208, pp. 81–104, Nov. 2012.
[46] A. P. Sundar, F. Li, X. Zou, T. Gao, and E. D. Russomanno, "Understanding shilling attacks and their detection traits: A comprehensive survey," *IEEE Access*, vol. 8, pp. 171703–171715, 2020.

VANDANA A. PATIL received the B.E. degree in computer engineering from Savitribai Phule Pune University, India, in 2004, and the M.E. degree in computer engineering from the University of Mumbai, India, in 2012. Currently, she is an Assistant Professor at the St. Francis Institute of Technology, University of Mumbai. Her research interests include machine learning and intelligent systems, specifically focusing on recommender systems.

SANTOSH V. CHAPANERI (Senior Member, IEEE) received the B.E. degree in electronics and telecommunication engineering from the University of Mumbai, Mumbai, India, in 2001, and the M.S. degree in electrical and computer engineering from The University of Arizona, USA, in 2008. He has worked as a Software Developer at Patni Computer Systems Ltd., Mumbai, and at Microsoft Corporation, Seattle, USA. Currently, he is an Assistant Professor at the St. Francis Institute of Technology, University of Mumbai. His research interests include machine learning and signal processing. He is a Reviewer of IEEE Access, *IET Communications*, *IET Computer Vision*, *IET Electronics Letters*, *IET Signal Processing*, and *IET Transactions on Image Processing*.



DEEPAK J. JAYASWAL received the B.E. degree in electronics engineering from Shivaji University, Kolhapur, India, in 1991, the M.Tech. degree in communication engineering from IIT Bombay, in 2002, and the Ph.D. degree in computer engineering from the National Institute of Technology, Surat, India, in 2010. He is currently a Professor and the Dean of the Post-Graduate Program, St. Francis Institute of Technology, University of Mumbai, India. His research interests include image and video processing and machine learning. He is a Reviewer of *IETE Journal of Education*, *Evolutionary Intelligence* (Springer), and IEEE Transactions on Industrial Informatics.