Incorporating Interpretability into Latent Factor Models via Fast Influence Analysis

Weiyu Cheng
weiyu_cheng@sjtu.edu.cn
Shanghai Jiao Tong University

Yanyan Shen*
shenyy@sjtu.edu.cn
Shanghai Jiao Tong University

Linpeng Huang
lphuang@sjtu.edu.cn
Shanghai Jiao Tong University

Yanmin Zhu
yzhu@sjtu.edu.cn
Shanghai Jiao Tong University

ABSTRACT

Latent factor models (LFMs) such as matrix factorization have achieved the state-of-the-art performance among various collaborative filtering approaches for recommendation. Despite the high recommendation accuracy of LFMs, a critical issue to be resolved is their lack of interpretability. Extensive efforts have been devoted to interpreting the prediction results of LFMs. However, they either rely on auxiliary information which may not be available in practice, or sacrifice recommendation accuracy for interpretability. Influence functions, stemming from robust statistics, have been developed to understand the effect of training points on the predictions of black-box models. Inspired by this, we propose a novel explanation method named FIA (Fast Influence Analysis) to understand the prediction of trained LFMs by tracing back to the training data with influence functions. We present how to employ influence functions to measure the impact of historical user-item interactions on the prediction results of LFMs and provide intuitive neighbor-style explanations based on the most influential interactions. Our proposed FIA exploits the characteristics of two important LFMs, matrix factorization and neural collaborative filtering, and is capable of accelerating the overall influence analysis process. We provide a detailed complexity analysis for FIA over LFMs and conduct extensive experiments to evaluate its performance using real-world datasets. The results demonstrate the effectiveness and efficiency of FIA, and the usefulness of the generated explanations for the recommendation results.

CCS CONCEPTS

• **Information systems → Recommender systems**; *Collaborative filtering*; • **Computing methodologies** → *Factorization methods.*

*Corresponding author

KEYWORDS

Latent factor model; interpretability; influence function; recommender system

ACM Reference Format:
Weiyu Cheng, Yanyan Shen, Linpeng Huang, and Yanmin Zhu. 2019. Incorporating Interpretability into Latent Factor Models via Fast Influence Analysis. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330857

1 INTRODUCTION

Recommender systems play an increasingly significant role in improving user satisfaction and revenue of content providers. Collaborative filtering (CF) methods, aiming at predicting users' personalized preferences against items based on historical user-item interactions, are the primary techniques used in modern recommender systems. Among various CF methods, latent factor models (LFMs) such as matrix factorization (MF), have gained popularity via the Netflix Prize contest [6] and achieved the state-of-the-art performance. Typically, LFMs map users and items into latent vectors in a low-dimensional space, and compute each user-item preference score based on a function of two (i.e., user and item) latent vectors, e.g., performing inner product, or non-linear transformation with neural structures [17].

In spite of the superior performance, a critical issue with LFMs is its lack of interpretability. To be more specific, there is no intuitive meaning for each dimension of the latent vectors, which makes it extremely difficult to understand why LFMs recommend an item out of the others. It is worth noticing that in addition to recommendation accuracy, nowadays recommender systems have devoted great effort to making the recommendation models interpretable, i.e., explain why the model predicts a user-item preference score as it does. Providing an intuitive explanation for each prediction result is inevitably important to increase end-users' trust and acceptance to the recommendation results [7, 31]. The explanation can further enable practitioners to pinpoint the cause of unexpected model behaviors and develop effective strategies for enhancing model performance [20].

In the literature, there are two major categories of approaches that try to explain the prediction results produced by LFMs. The first category is to leverage external data sources to express the semantics of latent dimensions, and provide *feature-based explanations* for the recommendation results [9, 11, 36, 39]. For example, Zhang et al. [39] proposed Explicit Factor Model (EFM), which extracts product features from textual reviews and align each feature with a latent dimension in MF. In this way, EFM provides users with explanations like "*we recommend this item because it owns a particular feature that you may care about.*" The main limitation of such approaches is that the required external sources (e.g. textual reviews) can be expensive to obtain or even not available in practice. The second category focuses on the CF setting and aims to provide *neighbor-style explanations* based on merely historical user-item interactions, such as "*we recommend this item because of the following items that you bought before or the following users who are similar to you*". Existing methods in the second category generate the above explanations by enforcing specific constraints onto LFMs [1–3, 18]. For example, Abdollahi and Nasraoui [1] introduced Explainable Matrix Factorization (EMF), where an "explainability regularizer" was added to the objective function, encouraging MF to recommend items that are prevalent in the target user's neighbors. As for the explanation, EMF associates each recommended item with the ratings of the users who are similar to the target user. However, the additional constraints imposed with LFMs may sacrifice recommendation accuracy. This is mainly because the most explainable recommendation results may not always be the most accurate ones, which is known as the *accuracy-interpretability trade-off* [26].

To avoid the above trade-off, a *post hoc* interpreting method [26] was recently proposed to explain the prediction results of LFMs after the models have been trained. In that work, the authors applied association rules [4] to relate each recommended item (i.e., the output of a trained model) with some historical interactions (i.e., the training data) for a target user, and hence the interpretation and model training are decoupled. Unfortunately, the above method still suffers from two limitations. First, the mined associations among items cannot be used to explain how a trained model produces the particular predictions. In other words, LFMs are still black-box models without interpretability. Second and more importantly, not all the recommended items can be associated with other items based on the available (probably sparse) historical interactions [26], and in such scenarios, no explanation will be provided, which greatly limits the applicability of the proposed method.

In this paper, we aim to incorporate interpretability into LFMs and provide an intuitive explanation for each prediction produced by trained LFMs. Our key insight is that the prediction results from a trained LFM is typically affected by the knowledge learned from model's training data (i.e., historical user-item interactions). Instead of identifying *coincident associations* between training data and recommended items, a more direct question we would like to ask is: how does each training example for LFMs affect the predictions? To be accurate, we try to quantitatively measure the impact of each historical interaction on the prediction made by a trained LFM. As a benefit, given a predicted user-item preference score, the historical interactions with the highest impact for the same user/item naturally become the neighbor-style explanation for the prediction, which is a prevalent and intuitive form of interpretation.

Following our insight, we introduce a novel explanation method named FIA (Fast Influence Analysis) to understand how a trained LFM makes its predictions by tracing back to model's training data. Similar to [22], we formalize the impact of a historical interaction on a particular prediction as the following counterfactual problem: *how would the prediction change if this interaction was not provided?* A straightforward way to measure the prediction change is to remove the interaction and retrain the LFM, which is prohibitively expensive. We notice that influence functions, which stemmed from robust statistics [12], have been used to understand the effect of training points on the predictions of both convex [13] and non-convex models [22]. Therefore, in FIA, we propose to leverage influence functions to quantify the impact of historical interactions on a predicted preference. A key technical challenge of employing influence functions in LFMs for measuring prediction changes is the high computation cost, which is determined by the large number of model parameters and huge training data volume. Fortunately, we observe that for each prediction, only a small subset of parameters in LFMs would be activated to directly affect the results of influence functions. FIA exploits this characteristic of LFMs to accelerate the computation of influence functions, thereby effectively reduce the time cost. We further extend FIA to handle the more general neural LFM setting, i.e., Neural Collaborative Filtering (NCF) [17] and develop an approximation method to perform influence analysis over neural network models efficiently.

To summarize, the major contributions of this paper are the following:

- To the best of our knowledge, this is the first attempt to apply influence functions to LFMs towards interpretable recommendation. We incorporate interpretability into LFMs by tracing each prediction back to models' training data, and further provide intuitive neighbor-style explanations for the predictions. (Section 3.1)
- We propose a novel fast influence analysis method named FIA for performing influence analysis over MF efficiently. FIA significantly reduces the computational cost by exploiting the characteristics of MF methods. (Section 3.2)
- We extend FIA to the neural collaborative filtering setting, which is a general neural network based LFM. We provide the time complexity of FIA on the NCF setting and develop an approximation method to further accelerate influence analysis over NCF. (Section 3.3)
- We conduct extensive experiments over real-world datasets and the results demonstrate the effectiveness of the generated explanations as well as the high efficiency of FIA. Furthermore, the analysis on the results of influence functions leads to better understanding of LFMs, which is valuable for researches on LFM-based recommendation. (Section 4)

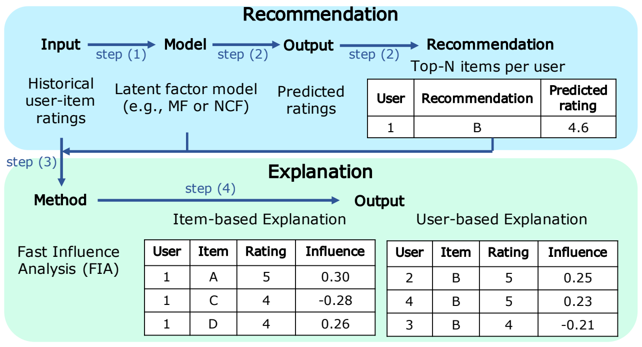
2 PRELIMINARIES

2.1 Latent Factor Models

Matrix Factorization (MF) has become the *de facto* LFM-based approach for recommendation. MF represents each user/item with a real-valued vector of latent features. Let $\mathbf{p}_u$ and $\mathbf{p}_i$ denote the vectors for user u and item i in a joint K-dimensional latent space, respectively. In MF, the predicted rating $\hat{R}_{ui}$ of user u against item i is computed by the inner product of $\mathbf{p}_u$ and $\mathbf{q}_i$, as defined below:

$$\hat{R}_{ui} = \mathbf{p}_u^T \mathbf{q}_i \tag{1}$$

The inner product operation linearly aggregates the pairwise latent feature multiplications, which limits the expressiveness of MF in



User	Recommendation	Predicted rating
1	B	4.6

Item-based Explanation

User	Item	Rating	Influence
1	A	5	0.30
1	C	4	-0.28
1	D	4	0.26

User-based Explanation

User	Item	Rating	Influence
2	B	5	0.25
4	B	5	0.23
3	B	4	-0.21

Figure 1: Interpreting LFMs with FIA (Steps in the figure are corresponding to Section 3.1).

terms of capturing complex user-item interactions. Neural Collaborative Filtering (NCF) [17] is thus proposed to learn a non-linear *interaction function* $f(\cdot)$, which can be considered as a generalization of MF:

$$\hat{R}_{ui} = f(\mathbf{p}_u, \mathbf{q}_i) \tag{2}$$

In the original paper, $f(\cdot)$ is specialized by a multi-layer perceptron (MLP) as follows:

$$\hat{R}_{ui} = \phi_X(...\phi_2(\phi_1(\mathbf{z}_0))...) \tag{3}$$

$$\mathbf{z}_0 = \mathbf{p}_u \oplus \mathbf{q}_i \tag{4}$$

$$\phi_l(\mathbf{z}_{l-1}) = \delta_l(\mathbf{W}_l \mathbf{z}_{l-1} + \mathbf{b}_l),\ l \in [1, X] \tag{5}$$

where X is the number of fully-connected hidden layers in the neural network, $\oplus$ is the concatenation of vectors, $\mathbf{W}_l$, $\mathbf{b}_l$ and δ_l are the weight matrix, bias vector and non-linear activation function for the l-th layer, respectively. Several recent attempts [10, 16] replace MLP with more complex operations (e.g., convolutions), but they still belong to the general NCF framework. In this paper, we focus on MF and the original NCF method, but our proposed algorithms are also applicable to all the instantiated models under the general NCF framework.

2.2 Influence Functions

Consider a general prediction problem from an input domain $\mathcal{X}$ to an output domain $\mathcal{Y}$. $Z = \{z_1, z_2, ..., z_n\}$ is the training set, where $z_i = (x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$. Let $L(z, \theta)$ denote the loss of a training point $z \in Z$ given model parameters $\theta \in \Theta$. The objective function $R(\theta)$ is defined as $\frac{1}{n}\sum_{i=1}^{n} L(z_i, \theta)$, and the optimal model parameters on convergence is defined as $\hat{\theta} \overset{\text{def}}{=} \arg\min_\theta \frac{1}{n}\sum_{i=1}^{n} L(z_i, \theta)$.

The goal of using influence functions is to estimate the influence of training points on a model's predictions. A simple solution to achieve this goal is to remove a training point, retrain the model with the remaining points from scratch, and compute the new prediction results. However, this process involves high computation cost. Influence functions provide an efficient way to estimate the model's prediction change without retraining. We first study the change of model parameters $\hat{\theta}_\epsilon - \hat{\theta}$ when a training point z is upweighted by an infinitesimal step ϵ, where the new parameters are:

$$\hat{\theta}_\epsilon \overset{\text{def}}{=} \arg\min_\theta \frac{1}{n}\sum_{i=1}^{n} L(z_i, \theta) + \epsilon L(z, \theta) \tag{6}$$

According to [13], we have the following closed-form expression:

$$\left.\frac{\mathrm{d}\hat{\theta}_\epsilon}{\mathrm{d}\epsilon}\right|_{\epsilon=0} = -H_{\hat{\theta}}^{-1} \nabla_\theta L(z, \hat{\theta}) \tag{7}$$

where $H_{\hat{\theta}} \overset{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta^2 L(z_i, \hat{\theta})$ is the Hessian matrix, and $\nabla_\theta L(z, \hat{\theta})$ is the derivate of the loss at z with respect to θ. Equation (7) is achieved by a quadratic approximation to $R(\theta)$ around $\hat{\theta}$, and the detailed derivation can be found in [22]. $H_{\hat{\theta}}$ is invertible when $R(\theta)$ is strictly convex in θ. Recent work [22] showed that even for non-convex objective functions that are widely used in neural networks, a damping term can be added into the Hessian to alleviate negative eigenvalues and make this equation approximately work.

Now that we have obtained the change in model parameters when upweighting z by the step ϵ, we can use the chain rule to study the change of model's prediction at a test point x_t:

$$\begin{aligned} \left.\frac{\mathrm{d}\hat{y}(x_t, \hat{\theta}_\epsilon)}{\mathrm{d}\epsilon}\right|_{\epsilon=0} &= \nabla_\theta \hat{y}(x_t, \hat{\theta}_\epsilon)^\top \left.\frac{\mathrm{d}\hat{\theta}_\epsilon}{\mathrm{d}\epsilon}\right|_{\epsilon=0} \\ &= -\nabla_\theta \hat{y}(x_t, \hat{\theta})^\top H_{\hat{\theta}}^{-1} \nabla_\theta L(z, \hat{\theta}) \end{aligned} \tag{8}$$

where $\hat{y}(x_t, \hat{\theta})$ is the model's prediction on the test point x_t with parameters $\hat{\theta}$. By setting ϵ to $-\frac{1}{n}$ (i.e., upweighting z by $-\frac{1}{n}$ is equivalent to removing z), we can approximate the influence of removing z from the training set on the prediction at x_t as follows:

$$\begin{aligned} \hat{y}(x_t, \hat{\theta}_{-z}) &\approx \hat{y}(x_t, \hat{\theta}) - \frac{1}{n}\left.\frac{\mathrm{d}\hat{y}(x_t, \hat{\theta}_\epsilon)}{\mathrm{d}\epsilon}\right|_{\epsilon=0} \\ &= \hat{y}(x_t, \hat{\theta}) + \frac{1}{n}\nabla_\theta \hat{y}(x_t, \hat{\theta})^\top H_{\hat{\theta}}^{-1} \nabla_\theta L(z, \hat{\theta}) \end{aligned} \tag{9}$$

where $\hat{\theta}_{-z}$ is the optimal model parameters after removing z.

Definition 2.1 (Influence). The *influence* of a training point z on the model's prediction at x_t is defined as:

$$INFL(z, x_t) \overset{\text{def}}{=} \hat{y}(x_t, \hat{\theta}) - \hat{y}(x_t, \hat{\theta}_{-z}) \tag{10}$$

Combining Equation (9) and (10), we have:

$$INFL(z, x_t) = -\frac{1}{n}\nabla_\theta \hat{y}(x_t, \hat{\theta})^\top H_{\hat{\theta}}^{-1} \nabla_\theta L(z, \hat{\theta}) \tag{11}$$

The right-hand side can be calculated based on the trained model parameters $\hat{\theta}$, the training point z, and the test point x_t, thus we can approximately measure the influence of a training point on model's prediction according to Equation (11) without retraining.

3 METHODOLOGY

In this section, we first describe how to apply influence functions to explain the recommendation results of LFMs. We then introduce our fast influence analysis method FIA for MF that significantly reduces the computational cost. Finally, we extend FIA to neural settings and propose an approximation algorithm to further improve analysis efficiency over NCF.

3.1 Interpreting LFMs with Influence Functions

Figure 1 depicts the overall process of explaining predictions made by LFMs with influence functions, which consists of the following four major steps: the first two steps are for LFM-based recommendation and the remaining two steps are for explanation.

(1) **Training LFMs.** The first step is to train LFMs under the conventional CF setting. The training data is historical user-item interactions, which can be explicit ratings or implicit interactions (e.g., user's buying records). Here we use explicit ratings for illustration, whereas the subsequent explanation method is independent of the type of training data.
(2) **Generating recommendations.** With the trained LFM from step (1), we can predict unobserved ratings in the user-item rating matrix. After completing the rating matrix, for each user, we select N items with the highest predicted ratings of the user as top-N recommendation results. The results will be supplied to the explanation method as input.
(3) **Calculating influence.** Given the trained LFM, training data (i.e., historical ratings) and recommendation results, we aim to estimate the *influence* of historical ratings on the predicted ratings of the recommendation results. The influence of a historical rating is measured by the difference in model predictions caused by removing the rating from the training set. According to Equation (11), we can estimate the influence with the trained model's parameters, thus avoiding expensive retraining.
(4) **Generating explanations.** The predominant form of interpretation in the context of recommendation is *neighbor-style explanation*, which is formally defined as follows.

    Definition 3.1 (Neighbor-style Explanation). Given a recommended item for a target user, the k most influential historical ratings of the user (resp. item) to the model's predicted rating form its item-based (resp. user-based) neighbor-style explanation.

    In step (3), we have measured the influence of historical ratings on the predicted rating of each recommendation. We then sort the historical ratings for the target user/item according to their influence to the recommendation, and select top-k ratings as explanation. As for the ranking function, if we want to report the most influential historical ratings to practitioners for model optimization, we can sort ratings in an ascending order of their absolute influence values. Otherwise, we may retain ratings with positive influence values and report the ones with highest scores to the target user.

Example. We provide an example of our explanation method in Figure 1, where the LFM predicts User 1's rating on Item B to be 4.6 and recommend B. We estimate the influence of historical ratings of User 1 and Item B on this prediction, and then sort these ratings according to their influence. The top-3 ratings of the user (resp. item) then become an item-based (resp. user-based) neighbor-style explanation for the recommendation. From the figure, User 1 has rated 5 on Item A, and this rating has an influence of 0.3. This means the predicted rating 4.6 would be decreased to 4.3 if the point (User 1, Item A, 5) was not included in the model's training data.

3.2 Fast Influence Analysis for MF

We now present how to estimate the influence of historical ratings in the training set $\mathcal{R}$ on a rating predicted by MF model. Let $\hat{y}(u_t, i_t, \hat{\theta})$ be the model's predicted rating for a target user u_t on an item i_t given parameters $\hat{\theta}$. Formally, we need to compute $\{INFL(z, u_t, i_t) \mid z \in \mathcal{R}_t\}$, where z is a training point in the form of (u, i, y), and $\mathcal{R}_t \subseteq \mathcal{R}$ is the set of historical ratings of u_t and i_t, to support both item-based and user-based neighbor-style explanations. According to Equation (11), we have

$$INFL(z, u_t, i_t) = -\frac{1}{n}\nabla_\theta \hat{y}(u_t, i_t, \hat{\theta})^\top H_{\hat{\theta}}^{-1} \nabla_\theta L(z, \hat{\theta}) \tag{12}$$

The main challenge to compute the right-hand side of the above equation is caused by $H_{\hat{\theta}}^{-1}$, where $H_{\hat{\theta}} \overset{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta^2 L(z_i, \hat{\theta}), z_i \in \mathcal{R}$. Given the training set $\mathcal{R}$ with n training points and an MF model with p parameters (i.e., the latent factors of all users and items), computing $H_{\hat{\theta}}$ needs $O(np^2)$ operations, and reverting $H_{\hat{\theta}}$ needs $O(p^3)$ operations. Thus the time complexity of computing influence for a single prediction is $O(np^2 + p^3)$, which is expensive for modern recommender systems with millions of parameters.

Basic influence analysis approach. Instead of explicitly computing $H_{\hat{\theta}}^{-1}$, a more efficient way is to employ second-order optimization techniques [22]. Noticing that $H_{\hat{\theta}}^{-1}$ is symmetric, we first rewrite Equation (12) as follows:

$$INFL(z, u_t, i_t) = -\frac{1}{n}\nabla_\theta L(z, \hat{\theta})^\top H_{\hat{\theta}}^{-1} \nabla_\theta \hat{y}(u_t, i_t, \hat{\theta}) \tag{13}$$

We then divide the right-hand side into two parts: $H_{\hat{\theta}}^{-1}\nabla_\theta \hat{y}(u_t, i_t, \hat{\theta})$, $\nabla_\theta L(z, \hat{\theta})$, and calculate the influence with the three steps below:

S1. Computing $H_{\hat{\theta}}^{-1}\nabla_\theta \hat{y}(u_t, i_t, \hat{\theta})$. This is a Hessian-vector product, which can be transformed to an optimization problem:

$$H_{\hat{\theta}}^{-1}\nabla_\theta \hat{y}(u_t, i_t, \hat{\theta}) = \arg\min_t \{\frac{1}{2} t^\top H_{\hat{\theta}} t - \nabla_\theta \hat{y}(u_t, i_t, \hat{\theta})^\top t\} \tag{14}$$

The above optimization problem can be solved using conjugate gradients methods, and the complexity of each iteration is determined by the evaluation of $H_{\hat{\theta}} t$, which needs $O(np)$ operations without computing $H_{\hat{\theta}}$ explicitly [27]. Since the optimization empirically converges within a few iterations [25], the time complexity of this step is $O(np)$.

S2. Computing $\nabla_\theta L(z, \hat{\theta})$. For a single training point z, computing $\nabla_\theta L(z, \hat{\theta})$ requires $O(p)$ operations. As we aim to estimate the influence of all the training points in $\mathcal{R}_t$, the complexity of this step is $O(p|\mathcal{R}_t|)$.

S3. Computing $INFL(z, u_t, i_t)$. We can compute the influence by combining the results from the previous two steps using Equation (13). For each training point $z \in \mathcal{R}_t$, this step is an inner product with $O(p)$ operations. Hence, the total time complexity of this step is $O(p|\mathcal{R}_t|)$.

Let $n' = |\mathcal{R}_t|$. Since $n \gg n'$ in practice, the overall complexity of computing the influence of historical ratings on a predicted rating (i.e., $\{INFL(z, u_t, i_t) \mid z \in \mathcal{R}_t\}$) is $O(np)$.

Fast influence analysis (FIA). The basic influence analysis approach with the complexity of $O(np)$ is significantly more efficient than explicitly calculating $H_{\hat{\theta}}^{-1}$. However, it still incurs high computation cost, since both n and p can be very large in practice. For example, when being employed to the Movielens 1M dataset [14], the basic approach can take up to an hour to estimate the influence of historical ratings on a single prediction, which is still too expensive to be applied in real recommender systems.

To accelerate the influence analysis process, we propose a Fast Influence Analysis algorithm (FIA) for MF. We observe that for a given test point (u_t, i_t), the predicted rating $\hat{y}(u_t, i_t)$ by MF is only determined by a small fraction of model parameters, i.e., the latent factors of u_t and i_t, which is denoted by $\theta_t = \{\mathbf{p}_{u_t}, \mathbf{q}_{i_t}\}$. Based on this observation, we can modify the formulation of influence functions for MF as follows:

(1) We first study the change of latent factors $\hat{\theta}_{t,\epsilon} - \hat{\theta}_t$ when a training point z is upweighted by an infinitesimal step ϵ, giving us the new parameters:

$$\hat{\theta}_{t,\epsilon} \overset{\text{def}}{=} \arg\min_{\theta_t} \frac{1}{n'} \sum_{j=1}^{n'} L(z_j, \theta_t) + \epsilon L(z, \theta_t) \tag{15}$$

where $z_j \in \mathcal{R}_t$, $n' = |\mathcal{R}_t|$. This is because θ_t is only optimized by training points in $\mathcal{R}_t$. The above equation can be considered as a subproblem from the original one in Equation (6). According to Equation (7), we have:

$$\frac{\mathrm{d}\hat{\theta}_{t,\epsilon}}{\mathrm{d}\epsilon}\Big|_{\epsilon=0} = -H_{\hat{\theta}_t}^{-1} \nabla_{\theta_t} L(z, \hat{\theta}) \tag{16}$$

where $H_{\hat{\theta}_t} \overset{\text{def}}{=} \frac{1}{n'} \sum_{j=1}^{n'} \nabla^2_{\theta_t} L(z_j, \hat{\theta}), z_j \in \mathcal{R}_t$.

(2) We then apply the chain rule to study the change of model's prediction at the test point (u_t, i_t), which only relies on θ_t:

$$\begin{aligned}\frac{\mathrm{d}\hat{y}(u_t, i_t, \hat{\theta}_{t,\epsilon})}{\mathrm{d}\epsilon}\Big|_{\epsilon=0} &= \nabla_{\theta_t} \hat{y}(u_t, i_t, \hat{\theta}_{t,\epsilon})^\top \frac{\mathrm{d}\hat{\theta}_{t,\epsilon}}{\mathrm{d}\epsilon}\Big|_{\epsilon=0} \\ &= -\nabla_{\theta_t} \hat{y}(u_t, i_t, \hat{\theta})^\top H_{\hat{\theta}_t}^{-1} \nabla_{\theta_t} L(z, \hat{\theta})\end{aligned} \tag{17}$$

By setting ϵ to $-\frac{1}{n'}$ (i.e., upweighting z by $-\frac{1}{n'}$ is equivalent to removing the point), we can approximate the influence of removing z from the training set on the prediction at (u_t, i_t):

$$INFL(z, u_t, i_t) = -\frac{1}{n'} \nabla_{\theta_t} \hat{y}(u_t, i_t, \hat{\theta})^\top H_{\hat{\theta}_t}^{-1} \nabla_{\theta_t} L(z, \hat{\theta}) \tag{18}$$

Time complexity analysis. The computation of Equation (18) is similar to the basic influence analysis. We first rewrite it as:

$$INFL(z, u_t, i_t) = -\frac{1}{n'} \nabla_{\theta_t} L(z, \hat{\theta})^\top H_{\hat{\theta}_t}^{-1} \nabla_{\theta_t} \hat{y}(u_t, i_t, \hat{\theta}) \tag{19}$$

We then analyze the time complexity of the three steps below.

S1'. Computing $H_{\hat{\theta}_t}^{-1} \nabla_{\theta_t} \hat{y}(u_t, i_t, \hat{\theta})$. According to Equation (14), the time cost of this step relies on the evaluation of $H_{\hat{\theta}_t} t$. Since the modified Hessian matrix $H_{\hat{\theta}_t}$ only focuses on n' training points, and the size of parameters $|\hat{\theta}_t|$ is $2K$ (K is the dimension of latent factors), the total time complexity of this step is reduced from $O(np)$ to $O(n'K)$.

S2'. Computing $\nabla_{\theta_t} L(z, \hat{\theta})$. For a single training point z, computing $\nabla_{\theta_t} L(z, \hat{\theta})$ needs $O(K)$ operations. We need to traverse all the training points in $\mathcal{R}_t$, thus the complexity of this step is $O(n'K)$.

S3'. Computing $INFL(z, u_t, i_t)$. We perform an inner product over $H_{\hat{\theta}_t}^{-1} \nabla_{\theta_t} \hat{y}(u_t, i_t, \hat{\theta})$ and $\nabla_{\theta_t} L(z, \hat{\theta})$ to obtain the final influence. In FIA, the dimension of the above vectors is reduced from p to K, hence the computation needs $O(K)$ operations for each training point $z \in \mathcal{R}_t$. The complexity of this step is $O(n'K)$.

To sum up, the overall complexity of FIA for MF is $O(n'K)$, which is greatly reduced compared with the $O(np)$ cost of the basic approach, since $n' \ll n$ and $K \ll p$. It is also worth mentioning that both n' and K are typically small and independent of the size of the training set, while n and p are proportional to the size of training data for recommendation models. This indicates that FIA enables us to perform efficient influence analysis for MF over large datasets.

3.3 Fast Influence Analysis for NCF

Extending FIA to NCF settings. The NCF methods are based on the latent factors (i.e., *embeddings*) of users and items, which aim to learn a complex interaction function (e.g., MLP) over embeddings from training data, unlike performing inner product in MF. To adapt FIA to the NCF setting, we divide the parameters involved in NCF into two parts: θ_e and θ_n, where θ_e is the embeddings of users and items, and θ_n is the parameters of the interaction function (e.g., weight matrices in MLP). Given a test point (u_t, i_t), the predicted rating $\hat{y}(u_t, i_t)$ from NCF is determined by both θ_e and θ_n, but the two parts of parameters are affected by training points in different ways. That is, θ_e is directly optimized by the training points in $\mathcal{R}_t$, which is similar to θ_t in MF; θ_n is learned using the complete set of $\mathcal{R}$. By applying the Taylor expansion over $\hat{y}(u_t, i_t, \theta_e, \theta_n)$ at $(\hat{\theta}_e, \hat{\theta}_n)$ on convergence, we have:

$$\begin{aligned}\hat{y}(u_t, i_t, \theta_e, \theta_n) = \hat{y}(u_t, i_t, \hat{\theta}_e, \hat{\theta}_n) + \nabla_{\theta_e} \hat{y} \Delta\theta_e + \nabla_{\theta_n} \hat{y} \Delta\theta_n \\ + o(||\Delta\theta||)\end{aligned} \tag{20}$$

After dropping the $o(||\Delta\theta||)$ term, we can estimate the influence of a training point by dividing it into two parts:

$$INFL(z, u_t, i_t) \approx INFL(z, u_t, i_t | \Delta\theta_e) + INFL(z, u_t, i_t | \Delta\theta_n) \tag{21}$$

Here we use $INFL(z, u_t, i_t | \Delta\theta_e)$ and $INFL(z, u_t, i_t | \Delta\theta_n)$ to denote the influence of z on the test point through the change of parameters in θ_e and θ_n, respectively. The computation of $INFL(z, u_t, i_t | \Delta\theta_e)$ is similar to Equation (18), the details of which is thus omitted:

$$INFL(z, u_t, i_t | \Delta\theta_e) = -\frac{1}{n'} \nabla_{\theta_e} \hat{y}(u_t, i_t, \hat{\theta})^\top H_{\hat{\theta}_e}^{-1} \nabla_{\theta_e} L(z, \hat{\theta}) \tag{22}$$

where $H_{\hat{\theta}_e} \overset{\text{def}}{=} \frac{1}{n'} \sum_{j=1}^{n'} \nabla^2_{\theta_e} L(z_j, \hat{\theta}), z_j \in \mathcal{R}_t$. Note that the embedding parameters are only optimized by the training points in $\mathcal{R}_t$.

As for the parameters θ_n involved in the interaction function, we have:

$$INFL(z, u_t, i_t | \Delta\theta_n) = -\frac{1}{n} \nabla_{\theta_n} \hat{y}(u_t, i_t, \hat{\theta})^\top H_{\hat{\theta}_n}^{-1} \nabla_{\theta_n} L(z, \hat{\theta}) \tag{23}$$

where $H_{\hat{\theta}_n} \overset{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} \nabla^2_{\theta_n} L(z_i, \hat{\theta}), z_i \in \mathcal{R}$, since the parameters in the interaction function are optimized by all the training points in $\mathcal{R}$. Combining Equation (21)–(23), we can compute $INFL(z, u_t, i_t)$ for NCF methods.

Time complexity analysis. The time cost of evaluating Equation (22) and (23) using FIA is similar to that of Equation (18) for MF. The complexity of performing Equation (22) is $O(n'K)$ since the calculation is only based on the embedding parameters and the training points in $\mathcal{R}_t$. For Equation (23), the computation cost is $O(n|\theta_n|)$, because the learning of θ_n relies on all the training points. Hence, the total time complexity of FIA for NCF is $O(n'K + n|\theta_n|)$.

FIA for NCF with approximation. We observe that in practice, $INFL(z, u_t, i_t | \Delta\theta_e) \gg INFL(z, u_t, i_t | \Delta\theta_n)$, due to the fact that the

Table 1: Statistics of the datasets

Dataset	Interaction#	User#	Item#	Sparsity
Yelp	731,671	25,815	25,677	99.89%
Movielens	1,000,209	6,040	3,706	95.53%

coefficient $\frac{1}{n'} \gg \frac{1}{n}$, especially for large datasets. For example, in the Movielens 1M dataset, n' is only a few hundred, while n can be up to one million. This inspires us to compute $INFL(z, u_t, i_t)$ approximately by dropping the second term in Equation (21), thereby we get:

$$INFL(z, u_t, i_t) \approx -\frac{1}{n'} \nabla_{\theta_e} \hat{y}(u_t, i_t, \hat{\theta})^\top H_{\hat{\theta}_e}^{-1} \nabla_{\theta_e} L(z, \hat{\theta}) \tag{24}$$

The intuition behind the above approximation is that the impact of removing a training point z on θ_e is more significant than that on θ_n, because θ_n is trained on the whole training set instead of the smaller subset $\mathcal{R}_t$. With the approximation, the time complexity of FIA for NCF methods can be further reduced to O($n'K$), which is as efficient as FIA for MF.

4 EXPERIMENTS

In this section, we conduct experiments to answer the following research questions:

- **RQ1**: How is the effectiveness of FIA for interpreting LFMs?
- **RQ2**: How is the efficiency of FIA compared with the basic influence analysis approach?
- **RQ3**: What does the generated explanation of FIA in real-world datasets look like?

4.1 Experimental Settings

4.1.1 Datasets. We used two publicly accessible datasets:

- **Yelp**: This is the Yelp Challenge dataset[1], which includes user ratings on different types of business places (e.g., restaurants, shopping malls, etc).
- **Movielens**: This is the widely used Movielens 1M dataset[2], which contains user ratings on movies.

For both datasets, we follow the common practice [28] to filter out users and items with less than 10 interactions. The statistics of the filtered datasets are summarized in Table 1.

4.1.2 Evaluation Protocols. Recall that FIA estimates the influence of historical ratings z on a trained LFM's prediction $\hat{y}(u_t, i_t)$, i.e., $INFL(z, u_t, i_t)$ as defined in Equation (10), and use the estimated influence to generate explanations. We thus evaluate the effectiveness of FIA in terms of the computed influence. A straightforward yet expensive way to measure the influence $INFL(z, u_t, i_t)$ is to remove the historical rating z and retrain the LFM. In our experiments, we compare the estimated influence using FIA with the observed influence derived by leave-one-out retraining to verify the effectiveness of FIA.

Specifically, we first randomly select 100 test points $\{(u_{t_i}, i_{t_i}) | i \in [1, 100]\}$ from the test set. For each test point (u_{t_i}, i_{t_i}), we apply FIA to compute $\{INFL(z_i, u_{t_i}, i_{t_i}) | z_i \in \mathcal{R}_{t_i}\}$, where $\mathcal{R}_{t_i}$ denotes the historical ratings from u_{t_i} and i_{t_i}. Here we only select z_i with the largest absolute influence value and compare it with the observed influence $INFL(z_i, u_{t_i}, i_{t_i})_{obs}$ after retraining. This is to prevent the small influence from being overwhelmed by the randomness in retraining, since LFMs are not strictly convex models. We perform retraining multiple times and use the average observed influence as $INFL(z_i, u_{t_i}, i_{t_i})_{obs}$ to obtain steadier results. We employ Pearson correlation coefficient (Pearson's R for brevity) to measure the linear correlation between the estimated influence and the observed one. A large Pearson's R indicates the effectiveness of FIA. In addition to the quantitative evaluation of effectiveness, we also evaluate the efficiency of FIA and assess the explanation generated by FIA through case study. We summarize the evaluation protocols used in the experiments as follows:

• **Quantitative evaluation of effectiveness.** We evaluate the effectiveness of FIA by measuring the correlation between the computed influence of FIA and the observed influence with leave-one-out retraining.

• **Quantitative evaluation of efficiency.** We evaluate the efficiency of FIA using running time by comparing with the basic influence analysis approach introduced in Section 3.2.

• **Qualitative evaluation of explanation.** We qualitatively evaluate the generated explanations of FIA through case study using real data.

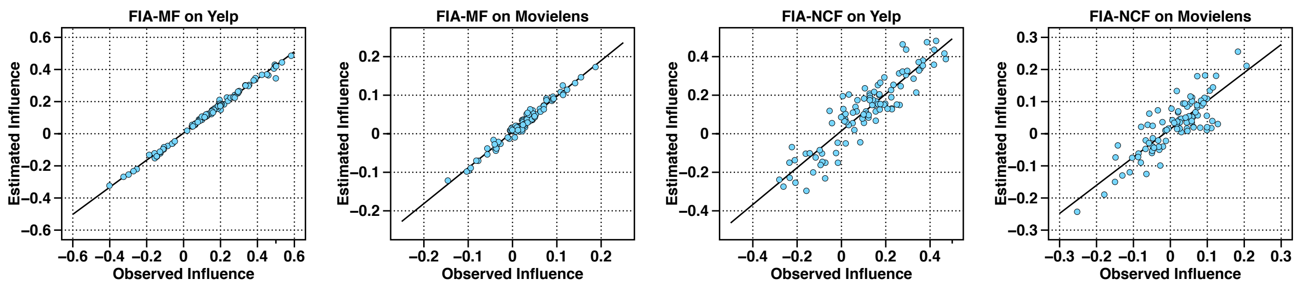
Non-goals. We want to emphasize that the proposed FIA is an explanation method, which aims to provide neighbor-style explanations for the recommendation results of trained LFMs. There are two ways to assess neighbor-style explanations [38]. The first is through online evaluation, which is unfortunately inaccessible to researchers. The second way is to evaluate the proportion of recommended items that can be explained as used in [1, 3, 26]. This proportion is also referred to as Model Fidelity [26], and a higher Model Fidelity indicates better performance of the explanation method. However, it is unnecessary to empirically compare FIA with existing methods over Model Fidelity, because FIA is able to generate explanation for any prediction of LFM, thus undoubtedly outperforming all the existing methods.

4.1.3 Parameter Settings. We implemented FIA using Tensorflow[3]. For each user in the dataset, we randomly held-out one rating to form the test set, and used the remaining data for model training. We adopted Adam [21] as the optimizer, which is a variant of stochastic gradient descent that dynamically tunes the learning rate during training. We set the initial learning rate to 0.001, the batch size to 3000, and the L2 regularization term to 0.001. During the computation of influence functions, we add a damping term of 10^{-6} to avoid negative eigenvalues in Hessian [22]. All the experiments were conducted on a server with 2 Intel Xeon 1.7GHz CPUs and 2 NVIDIA Tesla K80 GPUs. The source code is available at https://github.com/WeiyuCheng/FIA-KDD-19.

4.2 Evaluation of Effectiveness (RQ1)

We conduct experiments for FIA-MF and FIA-NCF on two datasets, where FIA-NCF is the version with approximation. The results are shown in Figure 2. First, we can see that the influence computed by FIA is highly correlated with the observed values obtained by

[1]https://www.yelp.com/dataset/challenge
[2]https://grouplens.org/datasets/movielens/1m/
[3]https://www.tensorflow.org



(a) FIA-MF — Yelp (b) FIA-MF — Movielens (c) FIA-NCF — Yelp (d) FIA-NCF — Movielens

Figure 2: Comparison between the observed influence and those computed by FIA methods for MF and NCF.

Table 2: Running time of FIA and IA for MF

	Yelp		Movielens	
Factors	**FIA-MF**	**IA**	**FIA-MF**	**IA**
8	0.78s	460s	0.96s	291s
16	0.75s	500s	1.21s	292s
32	0.80s	743s	1.59s	456s
64	0.77s	927s	1.26s	371s
128	0.95s	1705s	1.24s	167s
256	0.93s	2242s	1.22s	274s

Table 3: Running time of FIA and IA for NCF

	Yelp		Movielens	
Factors	**FIA-NCF**	**IA**	**FIA-NCF**	**IA**
8	1.17s	653s	0.89s	405s
16	1.01s	705s	1.47s	500s
32	0.97s	1010s	4.01s	601s
64	0.77s	1350s	4.35s	587s
128	1.09s	1633s	4.75s	655s
256	1.38s	2419s	2.41s	712s

retraining, which verifies the effectiveness of FIA methods. Specifically, for FIA-MF, the Pearson's R between the estimated and observed influence are 0.99 and 0.98 on Yelp and Movielens, respectively. For FIA-NCF, the Pearson's R between the estimated and observed influence are 0.92 and 0.84 on Yelp and Movielens, respectively. The strong correlations prove that FIA can effectively approximate the influence of training points without expensive retraining. It is also worth noticing that FIA provides better results for MF than NCF on both datasets. This is because in FIA-NCF, we ignore the effects of a small part of model parameters to improve the computational efficiency. This trade-off sacrifices a tiny fraction of accuracy, but we want to emphasize that the approximation method FIA-NCF can still provide convincing influence analysis according to the results.

4.3 Evaluation of Time Cost (RQ2)

We now empirically study the computational efficiency of FIA. We measure the time cost of FIA-MF and FIA-NCF with IA on the two datasets, where IA denotes the basic influence analysis approach described in Section 3.2. For each dataset, we randomly select a set of test points $\{(u_t, i_t)\}$, and record the average running time for computing the influence of training points on the test points, i.e., $\{INFL(z, u_t, i_t) | z \in \mathcal{R}_t\}$. The running time with different latent factor dimension settings for MF and NCF is provided in Table 2 and Table 3, respectively.

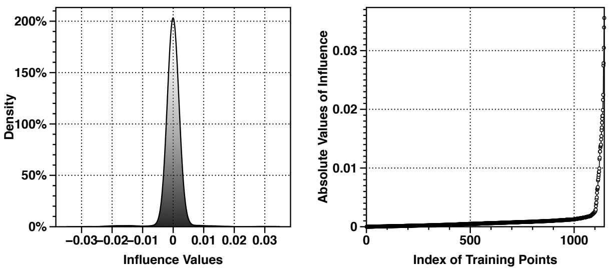
Movie	Movie Type	Your Rating	Influence
A Close Shave	Animation& Comedy	3	-0.036
The Best of Aardman Animation	Animation & Children's	3	-0.034
The Princess Bride	Comedy & Romance	3	-0.030
The Last Days of Disco	Drama & Comedy	5	0.028
My Fair Lady	Musical & Romance	3	-0.027
Dumbo	Animation & Children's	5	0.022

Figure 3: An example of item-based explanation.

From the results, we can see that our FIA methods are consistently much more efficient than IA on both datasets. Specifically, FIA runs 135 to 2411 times faster than IA in MF, and achieves a speedup of 138x to 1752x than IA in NCF. Note that the time cost of FIA is always at a small scale, i.e., less than 5 seconds, regardless of the dimension of latent factors and the employed datasets. This shows the potential of FIA to be applied to real recommendation scenarios. Besides, we can observe that the time cost of both methods generally increases with the dimension of latent factors, but with some exceptions. This is caused by the iterative method that we used to solve Equation (14). That is, the number of iterations until convergence depends on specific model parameters, which results in certain variance in the running time.

4.4 Case Study (RQ3)

We now perform a case study to gain intuitions on the effectiveness of FIA in providing explanations for the predictions of LFMs. We use the Movielens dataset for illustration, in which the items are movies that readers are usually familiar with. We first train an MF model on the dataset until convergence. We then randomly select a test user with 53 historical ratings and predict the user's rating for the movie *The Lion King (1994)* using the trained MF model. In our experiment, the MF method predicts the rating to be 4.04. We then employ FIA to estimate the influence of the user's historical ratings on this prediction, and sort the ratings according to their



(a) Smooth histogram (b) Scatter plot

Figure 4: Distribution of influence.

absolute influence values to generate an item-based explanation for the prediction.

The generated explanation is shown in Figure 3, where we preserve the top-6 influential ratings. We also provide the movie type and the computed influence for clearer illustration. According to the results, we can explain to the user: "we predict your rating for *The Lion King (1994)* to be 4.04, mostly because of your previous ratings on the following 6 items". Since the type of movie *The Lion King (1994)* belongs to Animation & Comedy, the explanation in this example is quite intuitive and convincing, i.e., most of movies in the generated list are similar to *The Lion King (1994)* in terms of the movie type. It is also interesting to see that the user's past ratings of 3 would have negative influence on the prediction, whereas the past ratings of 5 provide positive influence. This is reasonable because the model's predicted rating on the target movie would be increased (resp. decreased) by the user's appreciation (resp. dislike) for the movies of similar types.

Besides, the estimated influence is also helpful for us to better understand LFMs. Here we draw the distribution of the influence of training points on the aforementioned test points, as shown in Figure 4. We focus on the analysis of training points $z \in \mathcal{R}_t$. Figure 4a provides a smooth histogram showing the distribution of influence values, which tells that the influence of training points is usually centered around zero. Figure 4b is a scatter plot showing the sorted absolute values of influence. We observe that only a small fraction of training points contribute significantly to MF's prediction on the test point, which provides some insights on understanding the security risks of LFMs, since several abnormal training points are able to affect the model's prediction markedly.

5 RELATED WORK

Existing works on enhancing recommender systems with interpretability can be classified into different categories based on the type of recommendation models, including latent factor models [1, 3, 23, 26, 39], memory-based models [29, 32], graph-based models [5, 15, 35], deep learning models [8, 24, 33, 34], and tree-based models [37]. In this section, we review the related works on interpreting recommendation results from LFMs, which can be generally divided into two groups: interpreting with only historical user-item interactions and interpreting with external data sources.

5.1 Interpreting with Historical Interactions

The predominant approach to interpreting the predictions of LFMs with historical interactions is to enforce constraints on LFMs [1, 18, 20, 23]. Non-negative Matrix Factorization (NNMF) [23] was proposed to force all the latent factors in MF to be non-negative. In this way, a particular predicted rating can be interpreted as the user's total matching score with the item over latent dimensions. However, NNMF fails to generate explicit explanations for MF. Several methods were proposed to support neighbor-style explanations for the predictions of LFMs. Abdollahi and Nasraoui [1] introduced Explainable Matrix Factorization (EMF), where they augmented the objective function with an explainability regularizer and encouraged MF to recommend items that are prevalent in the target user's neighbors. Later on, the idea was incorporated into the probabilistic method [3] and Restricted Boltzmann Machines [2] for interpretable recommendation under the collaborative filtering setting. More recently, Heckel et al. [18] proposed to identify overlapping co-clusters of users and items with similar patterns, and confine the latent factors to represent users' and items' participation in the clusters. The co-cluster are then used to generate neighbor-style explanations for the recommended items.

One limitation of the above approaches is that the constrained recommendation models typically sacrifice accuracy to improve interpretability, which is known as the accuracy-interpretability trade-off. Moreover, these approaches need model-dependent modifications to the objective function, which lacks flexibility and brings extra switching cost in practice [30]. A recent work [26] focused on performing post-hoc interpretation for LFMs, which decoupled the interpretation process from model training. The explanations for the predictions of LFMs are obtained by learning association rules over the output of a matrix factorization model. As discussed before, while this post-hoc method avoids the accuracy-interpretability trade-off, it still suffers from two drawbacks. First, the generated explanations by mining association rules cannot explain the predictions in terms of the LFMs. In essence, association rule mining aims to find frequent correlations between historical interactions and recommended items, whereas the LFMs are still treated as black-box models. Second, the method has limited applicability as it is not guaranteed to provide explanations for any recommendation result.

Our proposed FIA method is a kind of post-hoc interpretability approach, which is flexible and can be seamlessly incorportated into LFMs. Different from [26], we leverage influence functions [13] to explain all the recommendation results from the perspective of model parameters. Influence functions stemmed from robust statistics [12] and were recently adopted to understand black-box predictions [22]. To the best of our knowledge, our FIA method is the first attempt to employ influence functions for interpreting the predictions of LFMs, under both MF and NCF settings. Furthermore, various optimizations have been developed to accelerate the influence analysis process to make FIA practical.

5.2 Interpreting with External Data Sources

Several methods have been proposed to interpret the predictions of LFMs by assigning semantics to the dimensions of latent vectors based on external data sources. These methods typically provide *feature-based explanations*. Zhang et al. [39] proposed Explicit Factor Models (EFM), which extracts product features from textual reviews and aligns each explicit feature with a latent dimension. EFM has been further extended to sequential settings by modeling dynamic user preferences on item features [40] or performing tensor factorization over multiple categories of items [9]. Similarly, user reviews are also integrated with ratings to build aspect-level LFMs. In [11, 19], aspect-aware topic modeling was performed on the reviews to extract item features from multiple aspects, and the ratings were computed by considering user preference against all the aspects. By doing this, the recommendation results can be interpreted as the degree of user preference on different aspects of items.

The key difference between the above works and our method is that they rely on external data sources to provide insights for interpreting the predictions of LFMs. However, auxiliary information associated with historical interactions can be expensive to obtain or may not be available in practice. Our method leverages merely the historical user-item interactions to enhance LFMs with interpretability, which is more desirable in real recommender systems.

6 CONCLUSION

In this paper, we propose an explanation method FIA by performing influence analysis over LFMs towards interpretable recommendation. We show that influence functions are effective in understanding the prediction results of LFMs by tracing back to historical interactions, and generating intuitive neighbor-style explanations. Our proposed FIA is able to perform influence analysis for MF with high computational efficiency. We then extend FIA to the NCF setting and develop an approximation method to further accelerate influence analysis process. Extensive experiments conducted over real-world datasets demonstrate the effectiveness and efficiency of FIA , as well as the usefulness of the generated explanations for the recommendation results from LFMs.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their insightful reviews. This work is supported by the National Key Research and Development Program of China (No. 2018YFC0831604). Yanyan Shen is also supported by NSFC (No. 61602297). Yanmin Zhu is supported by NSFC (No. 61772341, 61472254) and STSCM (No. 18511103002).

REFERENCES

[1] Behnoush Abdollahi and Olfa Nasraoui. 2016. Explainable Matrix Factorization for Collaborative Filtering. In *WWW'16*. 5–6.
[2] Behnoush Abdollahi and Olfa Nasraoui. 2016. Explainable restricted Boltzmann machines for collaborative filtering. *arXiv preprint arXiv:1606.07129* (2016).
[3] Behnoush Abdollahi and Olfa Nasraoui. 2017. Using Explainability for Constrained Matrix Factorization. In *RecSys'17*. 79–83.
[4] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. 1993. Mining Association Rules between Sets of Items in Large Databases. In *SIGMOD'93*. 207–216.
[5] Qingyao Ai, Vahid Azizi, Xu Chen, and Yongfeng Zhang. 2018. Learning Heterogeneous Knowledge Base Embeddings for Explainable Recommendation. *Algorithms* 11, 9 (2018), 137.
[6] Robert M. Bell and Yehuda Koren. 2007. Lessons from the Netflix prize challenge. *SIGKDD Explorations* 9, 2 (2007), 75–79.
[7] Mustafa Bilgic and Raymond J Mooney. 2005. Explaining recommendations: Satisfaction vs. promotion. In *Beyond Personalization Workshop, IUI*, Vol. 5. 153.
[8] Chong Chen, Min Zhang, Yiqun Liu, and Shaoping Ma. 2018. Neural Attentional Rating Regression with Review-level Explanations. In *WWW'18*. 1583–1592.
[9] Xu Chen, Zheng Qin, Yongfeng Zhang, and Tao Xu. 2016. Learning to Rank Features for Recommendation over Multiple Categories. In *SIGIR'16*. 305–314.
[10] Weiyu Cheng, Yanyan Shen, Yanmin Zhu, and Linpeng Huang. 2018. DELF: A Dual-Embedding based Deep Latent Factor Model for Recommendation. In *IJCAI'18*. 3329–3335.
[11] Zhiyong Cheng, Ying Ding, Lei Zhu, and Mohan S. Kankanhalli. 2018. Aspect-Aware Latent Factor Model: Rating Prediction with Ratings and Reviews. In *WWW'18*. 639–648.
[12] R Dennis Cook and Sanford Weisberg. 1980. Characterizations of an empirical influence function for detecting influential cases in regression. *Technometrics* 22, 4 (1980), 495–508.
[13] R Dennis Cook and Sanford Weisberg. 1982. *Residuals and influence in regression.* New York: Chapman and Hall.
[14] F. Maxwell Harper and Joseph A. Konstan. 2016. The MovieLens Datasets: History and Context. *TiiS* 5, 4 (2016), 19:1–19:19.
[15] Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. 2015. TriRank: Review-aware Explainable Recommendation by Modeling Aspects. In *CIKM'15*. 1661–1670.
[16] Xiangnan He, Xiaoyu Du, Xiang Wang, Feng Tian, Jinhui Tang, and Tat-Seng Chua. 2018. Outer Product-based Neural Collaborative Filtering. In *IJCAI'18*. 2227–2233.
[17] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *WWW'17*. 173–182.
[18] Reinhard Heckel, Michail Vlachos, Thomas P. Parnell, and Celestine Dünner. 2017. Scalable and Interpretable Product Recommendations via Overlapping Co-Clustering. In *ICDE'17*. 1033–1044.
[19] Yunfeng Hou, Ning Yang, Yi Wu, and Philip S. Yu. 2019. Explainable recommendation with fusion of aspect information. *World Wide Web* 22, 1 (2019), 221–240.
[20] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *ICDM'08*. 263–272.
[21] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[22] Pang Wei Koh and Percy Liang. 2017. Understanding Black-box Predictions via Influence Functions. In *ICML'17*. 1885–1894.
[23] Daniel D. Lee and H. Sebastian Seung. 2000. Algorithms for Non-negative Matrix Factorization. In *NIPS'00*. 556–562.
[24] Yichao Lu, Ruihai Dong, and Barry Smyth. 2018. Coevolutionary Recommendation Model: Mutual Learning between Ratings and Reviews. In *WWW'18*. 773–782.
[25] James Martens. 2010. Deep learning via Hessian-free optimization. In *ICML'10*. 735–742.
[26] Georgina Peake and Jun Wang. 2018. Explanation Mining: Post Hoc Interpretability of Latent Factor Models for Recommendation Systems. In *SIGKDD'18*. 2060–2069.
[27] Barak A. Pearlmutter. 1994. Fast Exact Multiplication by the Hessian. *Neural Computation* 6, 1 (1994), 147–160.
[28] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI'09*. 452–461.
[29] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *CSCW'94*. 175–186.
[30] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *SIGKDD'16*. 1135–1144.
[31] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor (Eds.). 2011. *Recommender Systems Handbook.* Springer.
[32] Badrul Munir Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW'01*. 285–295.
[33] Sungyong Seo, Jing Huang, Hao Yang, and Yan Liu. 2017. Interpretable Convolutional Neural Networks with Dual Local and Global Attention for Review Rating Prediction. In *RecSys'17*. 297–305.
[34] Jiaxi Tang and Ke Wang. 2018. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In *WSDM'18*. 565–573.
[35] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. Ripple Network: Propagating User Preferences on the Knowledge Graph for Recommender Systems. *arXiv preprint arXiv:1803.03467* (2018).
[36] Nan Wang, Hongning Wang, Yiling Jia, and Yue Yin. 2018. Explainable Recommendation via Multi-Task Learning in Opinionated Text Data. In *SIGIR'18*. 165–174.
[37] Xiang Wang, Xiangnan He, Fuli Feng, Liqiang Nie, and Tat-Seng Chua. 2018. TEM: Tree-enhanced Embedding Model for Explainable Recommendation. In *WWW'18*. 1543–1552.
[38] Yongfeng Zhang and Xu Chen. 2018. Explainable Recommendation: A Survey and New Perspectives. *arXiv preprint arXiv:1804.11192* (2018).
[39] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. 2014. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *SIGIR'14*. 83–92.
[40] Yongfeng Zhang, Min Zhang, Yi Zhang, Guokun Lai, Yiqun Liu, Honghui Zhang, and Shaoping Ma. 2015. Daily-Aware Personalized Recommendation based on Feature-Level Time Series Analysis. In *WWW'15*. 1373–1383.