FIRE: Fast Incremental Recommendation with Graph Signal Processing

Jiafeng Xia*
School of Computer Science
Fudan University
Shanghai, China
jfxia19@fudan.edu.cn

Dongsheng Li
Microsoft Research Asia
Shanghai, China
dongsli@microsoft.com

Hansu Gu†
Seattle, United States
hansug@acm.org

Jiahao Liu*
School of Computer Science
Fudan University
Shanghai, China
jiahaoliu21@m.fudan.edu.cn

Tun Lu*†
School of Computer Science
Fudan University
Shanghai, China
lutun@fudan.edu.cn

Ning Gu*
School of Computer Science
Fudan University
Shanghai, China
ninggu@fudan.edu.cn

ABSTRACT

Recommender systems are incremental in nature. Recent progresses in incremental recommendation rely on capturing the temporal dynamics of users/items from temporal interaction graphs, so that their user/item embeddings can evolve together with the graph structures. However, these methods are faced with two key challenges: 1) model training and/or updating are time-consuming and 2) new users/items cannot be effectively handled. To this end, we propose the fast incremental recommendation (FIRE) method from a graph signal processing perspective. FIRE is non-parametric which does not suffer from the time-consuming back-propagations as in previous learning-based methods, significantly improving the efficiency of model updating. In addition, we encode user/item temporal information and side information by designing new graph filters in FIRE, which can capture the temporal dynamics of users/items and address the cold-start issue for new users/items, respectively. Experimental studies on four popular datasets demonstrate that FIRE can improve the accuracy by a large margin and improve the model updating efficiency by at least 3X compared with the state-of-the-art incremental recommendation algorithms. The Code is available at https://github.com/Yaveng/FIRE.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Incremental recommendation, graph signal processing

*Also with Shanghai Key Laboratory of Data Science, Fudan University, Shanghai, China.
†Corresponding author.

WWW '22, April 25–29, 2022, Virtual Event, Lyon, France

ACM ISBN 978-1-4503-9096-5/22/04…$15.00
https://doi.org/10.1145/3485447.3512108

ACM Reference Format:
Jiafeng Xia, Dongsheng Li, Hansu Gu, Jiahao Liu, Tun Lu, and Ning Gu. 2022. FIRE: Fast Incremental Recommendation with Graph Signal Processing. In *Proceedings of the ACM Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3485447.3512108

1 INTRODUCTION

Recommender system recommends interesting items to users according to their historical interactions, which is playing an increasingly important role in the era of information overload. Generally, new users, items and user-item interactions are observed continuously over time, which makes the real-world recommender systems incremental in nature. This means that when new data arrives, the recommender systems do not have time to retrain the model, but can only rely on the most updated models to complete the recommendation. For instance, IncCTR [32] jointly learns knowledge from historical data and new incoming data to achieve incremental recommendation. SML [36] employs a neural network-based transfer component to transform the old model to a new model during training to realize fast recommendation without retraining model.

At present, graph neural networks have achieved huge success in the field of recommender systems. Graph neural networks can incorporate additional information by mining from graph structure data, which greatly promotes the research of recommendation algorithms based on the user-item interaction graph [1, 31]. Recent progresses in incremental recommendation also rely on capturing the temporal dynamics of users/items from temporal interaction graphs, so that their user/item embeddings can evolve together with the graph structures. DeepCoevolve [5] and JODIE [17] adopt RNNs to update the node features to tackle the incremental update issue of GNN-based CF methods. TGN [21] proposes a framework for learning continuous-time dynamic graphs, which can accurately capture the changes of nodes' feature over time through memory module and embedding module.

However, existing graph-based incremental recommendation methods are faced with two key challenges. The first challenge is that the model training and/or updating are time-consuming. Due to the large amount of learnable parameters, the deep learning methods can accurately model the user and item features changing

over time, but too many model parameters also limit the efficiency of model training and/or updating. When new interactions arrive, the model needs to update existing parameters or create new parameters to incorporate the new information, the training of which will be very time-consuming. The second challenge is that new users and items cannot be effectively handled. Since most of the existing algorithms rely on learning from user-item interactions, when new users or items appear, the algorithms cannot achieve effective recommendation due to the lack of corresponding interactions.

To this end, this paper proposes the fast incremental recommendation (FIRE) method from a graph signal processing perspective. Specifically, FIRE is a graph signal processing based non-parametric model which does not need complex and time-consuming model training process, but only needs efficient matrix multiplication operations to calculate the recommendation scores, significantly improving the efficiency of model updating. Through a carefully designed filter combined with temporal information, FIRE can mine the time-varying features of users and items in the user-item interaction sequences, and capture the temporal dynamics of users and items, to improve the recommendation accuracy. We also design a filter to combine side information for FIRE, which can alleviate the user/item cold-start issue. This filter is complementary to the filters designed based on user interactions, which can realize the recommendations by only learning from the graph signals of user/item side information, when corresponding user or item interactions are unavailable. Experimental studies on four popular datasets demonstrate that FIRE can improve the accuracy by a large margin and improve the model updating efficiency by at least 3X compared with the state-of-the-art incremental recommendation algorithms.

Our key contributions are summarized as follows. 1) We propose a fast incremental recommendation method based on graph signal processing, which does not need time-consuming model training phase, achieving high model updating efficiency. 2) We design two filters to introduce temporal information and side information into FIRE to capture the temporal dynamics of users/items and address the cold-start issue for new users/items, respectively. 3) We conduct extensive experiments on four real-world datasets, and the results show that FIRE can outperform state-of-the-art incremental recommendation methods in accuracy by a large margin and achieve higher efficiency in the model updating phase.

2 RELATED WORK

A variety of collaborative filtering algorithms have been proposed, from the simplest UserCF [13] that uses similar users' interaction information, to the factorization methods [2, 16, 18, 19] that extract the latent features of users/items from user interaction records, and then to the recent methods based on deep learning [3, 12, 23, 34, 37]. With the increased complexity of the models, especially deep models, the accuracy of the CF algorithm has also been improved.

Side information is a kind of information besides user-item interactions, which has been widely used in recommender systems. Common side information includes the user's age, gender, occupation etc., and the item's category, description, picture, etc. In movie recommendation and music recommendation, this kind of side information is more popular . For example, Meta-Prod2vec [27] leverages historical user interactions with item attributes, which are injected into the model as side information, to compute low-dimensional embeddings of items and achieves good performance. ICLF [25] takes user-category, item-category and category-category interactions into account to complete the recommendation of single category items and multi category items.

Recently, GNN based CF algorithms achieved huge success by incorporating user-item bipartite graph structure information in representation learning [11, 17, 29]. The GCMC method [1] introduced an AutoEncoder into the user-item interaction graph to predict possible ratings. Based on factorization methods, such as matrix factorization [16] and FISM [15], NGCF [31] further extracted high-level features by building high-order connectivities through GNN, thus improved the performance. In addition to the user interaction information, GraphRec [7] takes the users' social relationship into account and models the user, item and social relationship respectively, to improve the recommendation accuracy. DSCF [8] also takes user's social information into account, and it expands the user neighborhood in traditional GNN-based CF from low-order to high-order, and treats the information from different neighbors differently. LightGCN [11] improved over NGCF by removing unnecessary feature transformation and nonlinear activation to improve both efficiency and accuracy. Note that GNN updates node feature by aggregating the information of all its neighboring nodes. With the increase of GNN layers, each node can obtain the information of neighboring nodes with larger distances, which could improve the accuracy if the oversmoothing issue is properly handled [11].

All above GNN-based CF methods suffer from one common issue: a well-trained GNN model may become out-of-date due to the graph structure changes caused by new interactions. To address this issue, DeepCoevolve [5] and JODIE [17] adopt RNNs to update the node features to tackle the incremental GNN update issue. GCMCRNN [6] combine GCMC with RNN to learn both spatial structure information and temporal information of user-item subgraphs in all periods. However, these RNN-based methods may suffer from the common data sparsity issue and cold start issue in recommendation. Incremental learning is another line of works to solve the above problem. SPMF [30] is probabilistic matrix factorization (PMF [22]) based method, which employs a reservoir to maintain historical data and uses the data in the reservoir plus the new observations to update the model. IncCTR [32] uses a data module and a feature module to construct training data and handle features respectively, and uses a model module to fine-tune the model parameters with knowledge distillation. SML [36] employs a neural network-based transfer learning component to transform the old model to a new model during training, and optimizes the accuracy evaluated in the next time period to learn the transfer learning component. TGN [21] proposes a framework for learning continuous-time dynamic graphs, which can accurately capture the changes of nodes' feature over time through memory module and embedding module. However, these methods require a lot of time and computational resources for model training and inference.

3 PRELIMINARIES

3.1 Notations

This section introduces the notations used throughout this paper. Let the user set and item set be $\mathcal{U}$ and $\mathcal{V}$ respectively, $|\mathcal{U}| = n$

and $|\mathcal{V}| = m$, and the interactions between users and items can be defined as R. For implicit feedback dataset, $R \in \{0,1\}^{m \times n}$, and for explicit feedback dataset, $R \in \mathbb{R}^{m \times n}$. The normalized interaction matrix is denoted as $\tilde{R} = D_U^{-\frac{1}{2}} R D_I^{-\frac{1}{2}}$, where $D_U = Diag(R \cdot \mathbf{1})$ and $D_I = Diag(R^T \cdot \mathbf{1})$ are degree matrics.

3.2 Graph Signal Process

3.2.1 Graph Signal and Graph Laplacian Matrix. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ represent the set of nodes and edges, respectively, and $|\mathcal{V}| = n$. The graph can also be represented as an adjacency matrix $A \in \mathbb{R}^{n \times n}$. The graph signal is a vector mapping from node $f : \mathcal{V} \rightarrow \mathbb{R}$, and it can be expressed as $\mathbf{x} = [x_1, x_2, \cdots, x_n]$, where x_i represents the signal strength on node v_i.

Graph laplacian matrix is a core concept to study the structural properties of graphs and it can be defined as $L = D - A$, where D is a diagonal matrix, $D = Diag(A \cdot \mathbf{1})$. In addition, the definition of normalized graph laplacian matrix is $\tilde{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$.

3.2.2 Graph Fourier Transform and Graph Filter. As the graph laplacian matrix is a real and symmetric matrix, it can be decomposed into $L = U \Lambda U^T$, where $\Lambda = Diag(\lambda_1, \cdots, \lambda_n)$ is the diagonal matrix composed of the eigenvalues, $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$, and $U = [\mathbf{u_1}, \cdots, \mathbf{u_n}]$ is the matrix composed of eigenvectors $\mathbf{u_i} \in \mathbb{R}^n$ corresponding to the eigenvalue λ_i.

The matrix U can be seen as the Graph Fourier Transform basis that transforms the graph signal from the spatial domain to the frequency domain. Therefore the Graph Fourier Transform (GFT) and its inverse process can be defined as:

$$\hat{\mathbf{x}} = U^T \mathbf{x} \in \mathbb{R}^n, \mathbf{x} = U \hat{\mathbf{x}} \in \mathbb{R}^n.$$

In the frequency domain, we can use graph filters to filter some undesired frequencies in the graph signal. The graph filter can be defined as:

$$\mathcal{H}(L) = U Diag(h(\lambda_1), \cdots, h(\lambda_n)) U^T,$$

where $h(\cdot)$ is a filter.

Low-pass filter is an important class of filters that filter out the high frequency components that cause the signal to be non-smooth (or noisy) to improve the smoothness of the signal.

We introduce two important low-pass filters here. The first one is a linear filter, which can be defined as $h(\lambda_i) = \sum_{k=0}^{K} \alpha_k \lambda_i^k$, where α_k is the filter's coefficient. The second one is an ideal low-pass filter, which is given by $h(\lambda_i) = \begin{cases} 1, & \text{if } \lambda_i \leq \bar{\lambda} \\ 0, & \text{otherwise} \end{cases}$, where $\bar{\lambda}$ is a cut-off frequency.

3.2.3 Graph Convolution. The graph convolution of a given graph signal $\mathbf{x}$ can be defined as:

$$\mathbf{y} = \mathcal{H}(L)\mathbf{x} = U Diag(h(\lambda_1), \cdots, h(\lambda_n)) U^T \mathbf{x}.$$

The graph convolution process can be explained from the perspective of graph signal processing as follows: the graph signal $\mathbf{x}$ is first transformed from spatial domain to frequency domain through Graph Fourier Transform, then the undesired frequencies are filtered in the signal through filters in frequency domain. Finally, the signal is transformed back to spatial domain through inverse Graph Fourier Transform to complete the signal enhancement.

4 FAST INCREMENTAL RECOMMENDATION

4.1 Overview

Deep learning methods have significantly improved the accuracy of recommendation by encoding additional information, e.g., user/item temporal information. Due to the introduction of the learnable parameters, the recommendation algorithms can accurately and effectively capture the features of users and items changing over time [33]. However, too many learnable parameters also hinder the flexibility of the algorithms. Deep learning based recommendation algorithms usually take a lot of time to learn parameters. When new user interactions arrive, the model needs to update the existing parameters or create new parameters to incorporate the new interactions, the training of which will be very time-consuming.

Recently, a unified graph convolution-based framework for CF (GF-CF) [24] was proposed from the perspective of graph signal processing. Many existing CF methods are special cases of this framework, corresponding to various kinds of graph filters. The low-rank matrix factorization corresponds to the ideal low-pass filter $h(\lambda_i) = \mathbf{1}_{i \leq d}$, the neighborhood-based approaches correspond to a first-order linear filter $h(\lambda_i) = 1 - \lambda_i$, and the LGCN-IDE, which means LightGCN with Infinitely Dimensional Embedding, corresponds to $h(\lambda_i) = \Sigma_{k=0}^{K-1} \beta_k (1 - \lambda_i)^k$. Based on the new framework, GF-CF proposed a stronger graph filter by combining the linear filter and ideal low-pass filter as follows [24]:

$$\mathbf{p}_u = \mathbf{r}_u (\tilde{R}^T \tilde{R} + \alpha D_I^{-\frac{1}{2}} \bar{U} \bar{U}^T D_I^{\frac{1}{2}}), \tag{1}$$

where $\mathbf{r}_u \in \mathbf{R}^n$ is user u's interactions, and $\mathbf{p}_u \in \mathbf{R}^n$ is the predicted interactions for u. $\bar{U}$ is the top-K singular vectors of $\tilde{R}$.

One main advantage of GF-CF is that it is a non-parametric method. It does not need a complex and time-consuming model training process, but only needs efficient matrix multiplication operations to calculate the recommendation scores from a closed form solution, which can significantly improve the efficiency of incremental model update. However, GF-CF faces with two key challenges: **1) incapable of capturing temporal information**. GF-CF cannot capture the features of users and items changing over time, so that very early interactions are as important as recent interactions; and **2) incapable of handling new users/items**. GF-CF cannot provide recommendations for new users/items without historical interactions, which greatly limit the application scope of the algorithm.

To address the above issues, we propose the fast incremental recommendation (FIRE) method by designing the following two filters under the framework of GF-FC:

- **Temporal information filter**, which can capture the dynamics of user preference drifts over time by designing an incremental and time-aware attenuation filter on the interaction matrix;
- **Side information filter**, which can address the cold-start recommendation issue for new users/items by designing a feature-aware filter on user/item side information.

Both the above filters are non-parametric, so that the recommendation scores can also be obtained from closed form solutions after applying them without requiring time-consuming training.

4.2 Temporal Information Filter

GF-CF treats user interactions in different time periods equally, without considering that user interests may change over time and recent interactions are usually more important than very early interactions [5, 17]. For instance, when predicting which movie Alice will see tomorrow, we should pay more attention to movies she has watched recently, and pay less attention to movies she watched last year. If all user interactions are treated equally, the dynamics of user preference drifts will be lost and the recommendation accuracy will be hurt [5, 17].

To incorporate temporal information, FIRE needs to assign unequal weights to different interactions according to the difference between the time of historical interaction and the time of recommendation, reflecting that the interactions in different time periods could have varying impact on the recommendations in the future. In our design, FIRE works in the same way during each time period, so we only describe how FIRE works in a specific time period t without loss of generality.

Assuming that the current time period is t and we want to predict the items that user will interact with at time period $t+1$, we can divide the user historical interactions into two parts - the user interactions before t and the user interactions during t. The former interactions needs to be re-weighted by importance coefficients, indicating the importance of the user historical interaction to the prediction of future interaction, which can be used to capture the dynamics of user preference changing over time. The range of the importance coefficient is (0,1). The importance coefficient of interactions during time period t is 1 by default, i.e., current interactions have the greatest impact on the prediction of user interactions in the next time period. This is because user preferences are continuously changing over time [5, 17], so that interactions in the previous time period imply a lot of information about user interactions in the next time period, useful for predicting user future interactions.

Assume that, before time period t, the interactions between all m users $\mathcal{U}^{(<t)}$ and all n items $\mathcal{V}^{(<t)}$ can be expressed as a quadruple $\mathcal{E}^{(<t)} = \{(u_i, v_j, r_{ij}, t_{ij}) | u_i \in \mathcal{U}^{(<t)}, v_j \in \mathcal{V}^{(<t)}, t_{ij} < t\}$, where t_{ij} is the timestamp of interaction r_{ij}. Thus, we can construct the interaction matrix $R^{(<t)} \in \mathbb{R}^{m \times n}$ and the temporal information matrix $T^{(<t)} \in \mathbb{R}^{m \times n}$ as follows:

$$R_{ij}^{(<t)} = \begin{cases} r_{ij}, (u_i, v_j) \in \mathcal{E}^{(<t)}, \\ 0, (u_i, v_j) \notin \mathcal{E}^{(<t)}. \end{cases} T_{ij}^{(<t)} = \begin{cases} t_{ij}, (u_i, v_j) \in \mathcal{E}^{(<t)}, \\ 0, (u_i, v_j) \notin \mathcal{E}^{(<t)}. \end{cases}$$

From the temporal information matrix $T^{(<t)}$, we can calculate the importance of interactions at different time period before t and form the historical interaction importance matrix $D^{(<t)} \in \mathbb{R}^{m \times n}$ by an attenuation method as follows:

$$D_{ij}^{(<t)} = \mathrm{e}^{k(T_{i,j}^{(<t)} - t)}, \ \textit{(Exponential attenuation)}$$

where $D_{ij}^{(<t)}$ indicates the importance of historical user interaction r_{ij}, k represents the attenuation factor. Note that we also tried other attenuation methods, e.g., linear attenuation, and we choose exponential attenuation due to better empirical performance. After considering the temporal information of interactions, we can get the modified user historical interaction matrix as follows:

$$R^{(\leq t)} = R^{(<t)} \odot D^{(<t)} + R^{(t)},$$

where $R^{(t)}$ represents the interaction matrix of the current time t, and $\odot$ represents the Hadamard product. Therefore, the user interactions of the next time period $t+1$ can be predicted by the following closed form solution:

$$\mathbf{p}_u^{(t+1)} = \mathbf{r}_u^{(\leq t)} (\tilde{R}^{(\leq t)T} \tilde{R}^{(\leq t)} + \alpha D_I^{(\leq t)-\frac{1}{2}} \bar{U}^{(\leq t)} \bar{U}^{(\leq t)T} D_I^{(\leq t)\frac{1}{2}}).$$

$\mathbf{p}_u^{(t+1)}$ is the recommendation scores for user u at time period $t+1$, $\mathbf{r}_u^{(\leq t)} = R_{u,:}^{(\leq t)}$ represents all interactions of user u up to t.

To realize incremental recommendation, we can store interaction matrix $R^{(<t)}$ and time information matrix $T^{(<t)}$. After the prediction of the current period t is completed, the interactions and time information of the current period are used to update the above two matrices for the prediction of subsequent periods as follows:

$$R^{(<t+1)} = R^{(<t)} \oplus R^{(t)}, \quad T^{(<t+1)} = T^{(<t)} \oplus T^{(t)}, \tag{2}$$

where $\oplus$ represents element-wise addition operation.

4.3 Side Information Filter

New users or items are common in real-world recommender systems, but many algorithms cannot provide effective recommendations for them due to the lack of corresponding interactions, also known as the "cold start" issue. Using side information is an effective method to solve the cold start problem [26]. Side information usually refers to the characteristics of users, such as gender, age, occupation, etc., as well as the characteristics of items, such as category, attribute, etc. User interaction data implicitly depicts the characteristics of users and items from the perspective of preference, while side information explicitly depicts the static characteristics of users and items from the essence of them. Therefore, the side information provides the recommendation algorithms with a new perspective to learn about users and items. The benefits of using side information are two-fold. Firstly, it is complementary to user interaction information, so that the recommendation algorithm can further capture the characteristics of users and items through side information and thus improve the recommendation accuracy. Secondly, when new users or items appear, although the recommendation algorithm can not recommend items for users through interactions, it can complete the recommendation with the help of the neighborhood subgraph constructed through the side information of users or items, so as to solve the cold start problem and realize inductive recommendation. Therefore, we equip FIRE with a side information filter to achieve more accurate recommendation and address the cold start issue.

FIRE first needs to encode the side information of users and items into embedding. Different kinds of side information can be encoded by different methods. For example, the attributes of users or items can be encoded in the form of multiple one-hot encoding, and the image or text information can be processed by pretrained models. Suppose that the embeddings of new user and item after encoding are $X^U \in \mathbb{R}^{m \times d}$ and $X^I \in \mathbb{R}^{n \times d}$ respectively, where m and n represent the number of user and item respectively, and d represents the dimension of embedding. Here, for ease of description, the embedding size of user and item are the same. Then, FIRE calculates the similarity for users or items according to their side information and constructs the user or item similarity matrix as:

$$S_{i,j}^U = f(X_i^U, X_j^U), \quad S_{i,j}^I = f(X_i^I, X_j^I),$$

where $S^U_{i,j}$ means the similarity between user u_i and user u_j, X^U_i (X^U_j) represents u_i's (u_j's) side information. $S^I_{i,j}$, X^I_i and X^I_j have the same meaning for item, $f(\cdot,\cdot)$ is a similarity function, and we choose cosine similarity in this paper.

To ensure the numerical stability in the graph convolution process [28], we need to first normalize the two similarity matrices:

$$\tilde{S}^U = D_{S^U}^{-\frac{1}{2}} S^U D_{S^U}^{-\frac{1}{2}}, \quad \tilde{S}^I = D_{S^I}^{-\frac{1}{2}} S^I D_{S^I}^{-\frac{1}{2}},$$

where $\tilde{S}^U$ and $\tilde{S}^I$ represent the normalized user similarity matrix and item similarity matrix, and $D_{S^U} = \text{Diag}(S^U \cdot \mathbf{1})$ and $D_{S^I} = \text{Diag}(S^I \cdot \mathbf{1})$ represent the degree matrix of user similarity matrix and item similarity matrix, respectively. The user and item similarity matrices essentially represent the first-order neighborhood information of users and items. Similar to LightGCN [11], we take high-order neighborhood information into account to achieve more accurate recommendation:

$$\hat{\tilde{S}}^U = \sum_{k_1=0}^{K_1} \beta_{k_1} (\tilde{S}^U)^{k_1}, \quad \hat{\tilde{S}}^I = \sum_{k_2=0}^{K_2} \gamma_{k_2} (\tilde{S}^I)^{k_2},$$

where K_1 and K_2 indicate the neighborhood order of user and item respectively, β_{k_*} and γ_{k_*} are the coefficients to balance low-order information and high-order information. Therefore, the prediction of user interaction at the next time period $t+1$ based on user and item side information can be given by

$$\mathbf{P}_1^{(t+1)} = \mathbf{R}^{(\leq t)^T} \hat{\tilde{S}}^U, \quad \mathbf{P}_2^{(t+1)} = \mathbf{R}^{(\leq t)} \hat{\tilde{S}}^I,$$

where $\mathbf{P}_1^{(t+1)}$ and $\mathbf{P}_2^{(t+1)}$ represent the prediction of user interaction based on user and item side information, respectively. The prediction based on temporal-aware interaction information is:

$$\mathbf{P}_3^{(t+1)} = \mathbf{R}^{(\leq t)} (\tilde{R}^{(\leq t)T} \tilde{R}^{(\leq t)} + \alpha D_I^{(\leq t)-\frac{1}{2}} \bar{U}^{(\leq t)} \bar{U}^{(\leq t)T} D_I^{(\leq t)\frac{1}{2}}).$$

Therefore, the prediction of user interaction $\mathbf{P}^{(t+1)}$ at time period $t+1$ can be expressed as follows:

$$\mathbf{P}^{(t+1)} = \delta_1 \mathbf{P}_1^{(t+1)^T} + \delta_2 \mathbf{P}_2^{(t+1)} + \delta_3 \mathbf{P}_3^{(t+1)}, \tag{3}$$

where δ_1, δ_2 and δ_3 are the coefficients (hyper-parameters) to balance the predictions from three types of information.

4.4 Handling New Users/Items

When new users or items appear, FIRE can quickly adjust and respond. Let us take the emergence of new users as an example to see how FIRE can address the cold start issue. Suppose that there are m users before time t, the interaction matrix, time information matrix and similarity matrix are $R^{(<t)} \in \mathbb{R}^{m \times n}$, $T^{(t)} \in \mathbb{R}^{m \times n}$ and $S^U \in \mathbb{R}^{m \times n}$, respectively. If b users are added at time t, the model only needs to add b lines on $R^{(<t)}$ and $T^{(<t)}$ and fill in the corresponding interactions (if there are corresponding records) or time information (if there are corresponding records). For the similarity matrix S^U, FIRE only needs to calculate the similarities between b users and other $m+b-1$ users and fill them in the corresponding position of S^U. The calculation process of adding new items is similar. These processes require no training at all, so FIRE is very efficient to handle new users or items.

4.5 Model Efficiency Improvement

To further improve the efficiency of FIRE, we try to speedup the running of FIRE without significantly affecting its accuracy. Firstly, since the normalized similarity matrices $\hat{\tilde{S}}^U$ and $\hat{\tilde{S}}^I$ are dense matrices, the calculations involving these two matrices need a lot of matrix multiplication operations, which takes up most of the computations of FIRE. Similar to GraphSAGE[9] and ClusterGCN[4], the graph convolution operation is completed by using subgraphs instead of full graph to improve the efficiency of the algorithm. We propose two methods, the core idea of which is making the matrices sparse, to improve the efficiency of FIRE.

• Thresholding. We set a certain threshold for the similarity scores, so that similarity below the threshold is set to 0 and will not participate in the recommendation scores calculation.

$$\hat{\tilde{S}}^U_{ij} = \begin{cases} \hat{\tilde{S}}^U_{ij}, & \text{if } \hat{\tilde{S}}^U_{ij} \geq \epsilon_1, \\ 0, & otherwise. \end{cases} \quad \hat{\tilde{S}}^I_{ij} = \begin{cases} \hat{\tilde{S}}^I_{ij}, & \text{if } \hat{\tilde{S}}^I_{ij} \geq \epsilon_2, \\ 0, & otherwise. \end{cases}$$

Here, ϵ_1 and ϵ_2 are the thresholds.

• Top-k Neighbors. We can only keep the k most similar users in $\hat{\tilde{S}}^U_{ij}$ for each user or the k most similar items in $\hat{\tilde{S}}^I_{ij}$ for each item, and then use only the k most similar neighbors to predict future user-item interactions.

In this paper, we choose the first method because it is more efficient and similarly accurate. In addition, when we calculate $\mathbf{P}_1^{(t+1)}$, $\mathbf{P}_2^{(t+1)}$ and $\mathbf{P}_3^{(t+1)}$, multi-process parallel computing is adopted, which also significantly improves the computation efficiency.

4.6 Complexity Analysis

4.6.1 The complexity of FIRE. This section analyzes the computational complexity of FIRE. First, to obtain $\mathbf{P}_3^{(t+1)}$, we need to calculate $\tilde{R}^{(\leq t)}$ first, which needs the complexity of $O(mn)$, and $O(an^2) = O(n^2)$ is required to obtain $\tilde{R}^{(\leq t)T} \tilde{R}^{(\leq t)}$ due to the sparse matrix multiplication, where $a \ll n$ is the number of non-zero values per column in $\tilde{R}^{(\leq t)}$. In order to obtain $\bar{U}$, we use the GPM [14] to obtain the Top-K eigenvectors of $\tilde{R}^{(\leq t)T} \tilde{R}^{(\leq t)}$ and the complexity is $O((d\eta + d^3) log(1/\epsilon))$. Therefore, the total complexity is $O(n^2 k + n^2 + mn) = O(n^2 + mn)$ due to $k \ll n$. Similarly, to obtain $\mathbf{P}_1^{(t+1)}$ and $\mathbf{P}_2^{(t+1)}$, the complexities are both $O(nm)$. To sum up, FIRE has the complexity of $O(mn + n^2 + (d\eta + d^3) log(1/\epsilon))$. If there is a constant γ such that $m = \gamma n$, the complexity can also be written as $O(n^2 + (d\eta + d^3) log(1/\epsilon))$.

4.6.2 Incremental effectiveness of FIRE. For exponential operation x^y, suppose that the number of FLOPs M satisfies $1 < M < y-1$, and we assume that there are N_{t-1} records up to time $t-1$, and n_t records are added at time t. The process of calculating $R^{(\leq t)}$ of FIRE without incremental algorithm is $D_{ij}^{(\leq t)} = e^{k(T_{ij}^{(\leq t)} - t)}$, $R^{(\leq t)} = R^{(\leq t)} \odot D^{(\leq t)}$, the number of FLOPs of the former is $M_1 \times (N_{t-1} + n_t)$, and the number of FLOPs of the latter is $N_{t-1} + n_t$, so the total number of FLOPs is $(M_1+1) \times (N_{t-1} + n_t)$. While the process of calculating $R^{(\leq t)}$ of FIRE with incremental algorithm is $D_{ij}^{(<t)} = e^{k(T_{ij}^{(<t)} - t)}$, $R^{(\leq t)} = R^{(<t)} \odot T^{(<t)} + R^{(t)}$, the number of FLOPs of the former is $M_2 \times N_{t-1}$, and the number of FLOPs of the latter is $N_{t-1} + n_t$, so the total

number of FLOPs is $(M_2 + 1) \times N_{t-1} + n_t$. Suppose $M_1 \approx M_2$, obviously, the complexity of FIRE without an incremental algorithm is higher than that of FIRE with an incremental algorithm. In some industrial recommendation scenarios, the frequent interactions of a large number of users will lead to the continuous updating of records. In order to recommend items to users in real time, FIRE with an incremental algorithm is superior to FIRE without an incremental algorithm in efficiency.

5 EXPERIMENTS

5.1 Experimental Setup

5.1.1 Dataset. We adopt four real-world datasets to evaluate the performance of FIRE, including MovieLens 1M (900K ratings with 6,040 users and 3,706 items across 8 months), Douban Movie (1001,970 ratings with 10,000 users and 10,000 items across 12 years), Amazon Book (344,201 ratings with 19,414 users and 19,055 items across 9 months) and Amazon Electrionics (194,494 ratings with 20,000 users and 15,000 items across one year). For the MovieLens 1M dataset, both users and items have side information, while for the other three datasets, only items have side information. More details about the datasets are summarized in the appendix. It should be noted that due to the sparsity of these datasets, we set a large time interval for each dataset. However, our method can handle different time intervals without any modification.

5.1.2 Metrics. We evaluate FIRE and the state-of-the-art recommendation algorithms in top-N recommendation task with three ranking metrics: 1) F1, which balances between precision and recall by harmonic mean; 2) Mean Reciprocal Rank (MRR), which evaluates the performance of ranking according to the harmonic mean of the ranks; and 3) Normalized Discounted Cumulative Gain (NDCG), which accumulate the gains from ranking list with the discounted gains at lower ranks. Note that we use four-star or five-star ratings as positive ones in top-N recommendation.

5.1.3 Compared Methods. FIRE is compared with eleven CF methods. Among them, BPR [20], LightGCN [11], NFM [10], IFM [35] and GF-CF [24] are non-sequential CF methods that treat all user interactions equally regardless of temporal information, while RRN [33], DeepCoevolve [5], JODIE [17], SPMF [30], IncCTR [32] and SML [36] belong to sequential CF methods that take temporal information into account. Note that LightGCN, DeepCoevolve and JODIE are graph based CF methods, which can mine more user/item features on the user-item interaction graph. IFM and NFM can take side information as their input, which is the same as FIRE. More details of the methods are summarized in the appendix.

5.2 Accuracy Comparison

Table 1 shows the performance comparison between FIRE and ten state-of-the-art CF methods in all four datasets. As shown in the table, FIRE substantially outperforms all the compared methods. Besides, we can draw the following observations from the results.
1. Sequential methods can outperform non-sequential methods because the non-sequential methods treat all ratings equally while ignoring the varying importance of interactions in different time periods. On the contrary, sequential methods can better capture the user preference drifts over time and thus achieve better accuracy.
2. GNN-based methods are better than non-graph based methods as shown by the comparison between LightGCN and BPR, JODIE and RRN, and FIRE and SML. The main reason is that GNN-based methods can leverage graph structure information to facilitate recommendation, which can help users/items to enrich their representations from their neighborhoods.
3. FIRE consistently outperforms all the state-of-the-art CF methods in all datasets. The main reason is FIRE can learn from both the temporal information and the side information of users and items. With the help of the proposed temporal information filter, FIRE can mine the time-varying features of users and items from the user-item interaction sequences, and capture the temporal dynamics of users and items, to improve the recommendation accuracy. With the help of the proposed side information filter, FIRE can not only address the cold-start issue for new users/items but also improve the accuracy by auxiliary information. In addition, on two extremely sparse datasets, Amazon Book and Amazon Electronics, the accuracy of FIRE is much higher than traditional incremental recommendation algorithms such as SPMF and SML, and better than graph based incremental recommendation algorithms Jodie and DeepCoevolve. This shows that FIRE can also deal with sparse data by the design of temporal and side information filters.
4. It is interesting that LightGCN is a strong baseline which is non-incremental but outperforms some incremental baselines by always adopting the whole dataset. Incremental methods, e.g., IncCTR and SML, mainly try to improve the efficiency of model updating without sacrificing performance. Temporal methods, e.g., DeepCoevolve and Jodie, achieved comparable performance to LightGCN due to adopting temporal information.

5.3 Ablation Study

We conduct ablation study on Douban Movie and MovieLens 1M to analyze the importance of temporal information and side information to FIRE, and Table 2 shows the results. GF-CF adopts Eq (1) for recommendation, which does not contain any temporal or side information and can be used as a benchmark of FIRE.

- Importance of temporal information filter. Comparing GF-CF with FIRE (2) on Douban Movie dataset, FIRE achieves better performance than GF-CF on all metrics, which shows the importance of temporal information in helping FIRE to improve the recommendation accuracy. Similar observations can also be found in the MovieLens 1M dataset.
- Importance of side information filter. From the results on Douban Movie, we can conclude that item side information ($\delta_2 \neq 0$) is beneficial for FIRE by comparing FIRE (3) against FIRE (1) and FIRE (4) against FIRE (2), respectively. Similar observations can be found in the MovieLens 1M dataset. Note that FIRE only with side information takes the same input as NFM and IFM. FIRE only with side information can outperform NFM and IFM, which demonstrates that the advantages of FIRE are mainly from the algorithm design rather than adopting more information than other CF methods.

5.4 Efficiency Analysis

In this experiment, we compare the efficiency of FIRE against other incremental CF algorithms on the MovieLens 1M dataset. The experiments are all conducted with the same hardware. Figure 1 shows

Table 1: Accuracy comparison between FIRE and the other methods in four datasets. Since there are no training parameters in GF-CF and FIRE, the results are fixed every time, so the standard deviations are not reported in the results.

Models	MovieLens 1M		Douban Movie		Amazon Book		Amazon Electronics	
	F1@5	F1@10	F1@5	F1@10	F1@5	F1@10	F1@5	F1@10
BPR	0.019 ± 0.000	0.027 ± 0.001	0.003 ± 0.000	0.004 ± 0.000	0.007 ± 0.000	0.008 ± 0.000	0.003 ± 0.000	0.003 ± 0.000
LightGCN	0.041 ± 0.001	0.061 ± 0.001	0.004 ± 0.000	0.005 ± 0.000	0.013 ± 0.000	0.013 ± 0.000	0.006 ± 0.000	0.007 ± 0.001
NFM	0.021 ± 0.001	0.032 ± 0.001	0.003 ± 0.000	0.003 ± 0.000	0.001 ± 0.000	0.001 ± 0.000	0.001 ± 0.000	0.001 ± 0.000
IFM	0.014 ± 0.001	0.027 ± 0.001	0.003 ± 0.000	0.003 ± 0.000	0.001 ± 0.000	0.001 ± 0.000	0.001 ± 0.000	0.001 ± 0.000
GF-CF	0.031	0.046	0.007	0.010	0.023	0.024	**0.008**	0.008
RRN	0.045 ± 0.004	0.050 ± 0.008	0.002 ± 0.000	0.003 ± 0.000	0.004 ± 0.000	0.004 ± 0.000	0.007 ± 0.000	0.007 ± 0.000
DeepCoevolve	0.064 ± 0.005	0.091 ± 0.010	0.002 ± 0.000	0.003 ± 0.000	0.004 ± 0.000	0.003 ± 0.000	0.003 ± 0.000	0.004 ± 0.000
JODIE	0.067 ± 0.002	0.092 ± 0.003	0.002 ± 0.000	0.005 ± 0.000	0.008 ± 0.000	0.008 ± 0.001	**0.008 ± 0.001**	0.006 ± 0.001
SPMF	0.023 ± 0.001	0.021 ± 0.000	0.004 ± 0.000	0.005 ± 0.000	0.001 ± 0.000	0.001 ± 0.000	0.003 ± 0.000	0.002 ± 0.000
IncCTR	0.021 ± 0.002	0.032 ± 0.004	0.004 ± 0.001	0.005 ± 0.000	0.002 ± 0.000	0.001 ± 0.000	0.002 ± 0.000	0.002 ± 0.000
SML	0.032 ± 0.002	0.050 ± 0.005	0.005 ± 0.000	0.006 ± 0.001	0.004 ± 0.001	0.005 ± 0.001	0.002 ± 0.000	0.003 ± 0.000
FIRE (Ours)	**0.070**	**0.112**	**0.009**	**0.013**	**0.024**	**0.025**	**0.008**	**0.009**

Models	MovieLens 1M		Douban Movie		Amazon Book		Amazon Electronics	
	MRR@5	MRR@10	MRR@5	MRR@10	MRR@5	MRR@10	MRR@5	MRR@10
BPR	0.115 ± 0.004	0.121 ± 0.005	0.008 ± 0.000	0.009 ± 0.000	0.015 ± 0.000	0.016 ± 0.000	0.006 ± 0.000	0.006 ± 0.000
LightGCN	0.187 ± 0.003	0.184 ± 0.003	0.022 ± 0.001	0.022 ± 0.001	0.027 ± 0.000	0.027 ± 0.001	0.013 ± 0.001	0.015 ± 0.001
NFM	0.169 ± 0.002	0.165 ± 0.003	0.010 ± 0.001	0.010 ± 0.001	0.001 ± 0.000	0.001 ± 0.000	0.001 ± 0.000	0.001 ± 0.000
IFM	0.156 ± 0.007	0.155 ± 0.004	0.010 ± 0.000	0.009 ± 0.000	0.001 ± 0.000	0.001 ± 0.000	0.001 ± 0.000	0.001 ± 0.000
GF-CF	0.141	0.151	0.024	0.027	0.042	0.042	0.015	0.016
RRN	0.088 ± 0.006	0.101 ± 0.022	0.011 ± 0.000	0.013 ± 0.001	0.009 ± 0.001	0.011 ± 0.001	0.013 ± 0.000	0.015 ± 0.000
DeepCoevolve	0.155 ± 0.022	0.169 ± 0.020	0.006 ± 0.000	0.011 ± 0.000	0.016 ± 0.000	0.017 ± 0.000	0.009 ± 0.000	**0.018 ± 0.001**
JODIE	0.177 ± 0.003	0.209 ± 0.003	0.007 ± 0.000	0.010 ± 0.000	0.017 ± 0.001	0.018 ± 0.001	0.012 ± 0.000	0.013 ± 0.001
SPMF	0.105 ± 0.002	0.067 ± 0.001	0.016 ± 0.001	0.016 ± 0.001	0.003 ± 0.000	0.002 ± 0.000	0.004 ± 0.001	0.002 ± 0.000
IncCTR	0.094 ± 0.009	0.100 ± 0.011	0.018 ± 0.001	0.021 ± 0.001	0.002 ± 0.000	0.002 ± 0.000	0.004 ± 0.000	0.004 ± 0.000
SML	0.156 ± 0.004	0.154 ± 0.009	0.023 ± 0.001	0.023 ± 0.001	0.010 ± 0.001	0.010 ± 0.001	0.003 ± 0.000	0.007 ± 0.000
FIRE (Ours)	**0.422**	**0.411**	**0.027**	**0.031**	**0.043**	**0.044**	**0.017**	**0.018**

Models	MovieLens 1M		Douban		Amazon Book		Amazon Electronics	
	NDCG@5	NDCG@10	NDCG@5	NDCG@10	NDCG@5	NDCG@10	NDCG@5	NDCG@10
BPR	0.130 ± 0.004	0.149 ± 0.005	0.011 ± 0.001	0.016 ± 0.001	0.018 ± 0.000	0.024 ± 0.000	0.007 ± 0.000	0.009 ± 0.000
LightGCN	0.218 ± 0.003	0.233 ± 0.001	0.026 ± 0.001	0.031 ± 0.001	0.033 ± 0.001	0.040 ± 0.001	0.016 ± 0.001	0.021 ± 0.001
NFM	0.212 ± 0.008	0.228 ± 0.006	0.012 ± 0.001	0.015 ± 0.002	0.001 ± 0.000	0.001 ± 0.000	0.001 ± 0.000	0.002 ± 0.000
IFM	0.186 ± 0.004	0.211 ± 0.005	0.013 ± 0.000	0.016 ± 0.000	0.001 ± 0.000	0.002 ± 0.000	0.001 ± 0.000	0.001 ± 0.000
GF-CF	0.161	0.196	0.030	0.039	0.052	0.060	0.019	0.023
RRN	0.115 ± 0.004	0.141 ± 0.025	0.012 ± 0.000	0.015 ± 0.001	0.012 ± 0.001	0.015 ± 0.001	0.016 ± 0.000	0.021 ± 0.000
DeepCoevolve	0.198 ± 0.024	0.242 ± 0.024	0.007 ± 0.000	0.012 ± 0.000	0.017 ± 0.000	0.018 ± 0.001	0.010 ± 0.001	0.019 ± 0.001
JODIE	0.238 ± 0.002	0.247 ± 0.005	0.009 ± 0.000	0.019 ± 0.000	0.023 ± 0.001	0.027 ± 0.002	0.016 ± 0.002	0.019 ± 0.002
SPMF	0.145 ± 0.009	0.119 ± 0.004	0.022 ± 0.003	0.025 ± 0.001	0.003 ± 0.000	0.004 ± 0.000	0.006 ± 0.001	0.005 ± 0.001
IncCTR	0.123 ± 0.006	0.149 ± 0.004	0.022 ± 0.001	0.029 ± 0.001	0.003 ± 0.000	0.003 ± 0.000	0.005 ± 0.000	0.006 ± 0.000
SML	0.183 ± 0.005	0.197 ± 0.004	0.029 ± 0.001	0.033 ± 0.002	0.013 ± 0.002	0.015 ± 0.002	0.004 ± 0.000	0.009 ± 0.000
FIRE (Ours)	**0.471**	**0.481**	**0.035**	**0.046**	**0.053**	**0.063**	**0.020**	**0.026**

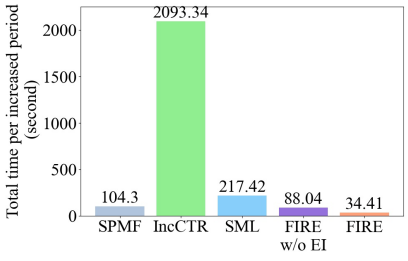

Figure 1: Efficiency of FIRE and other incremental CF methods.

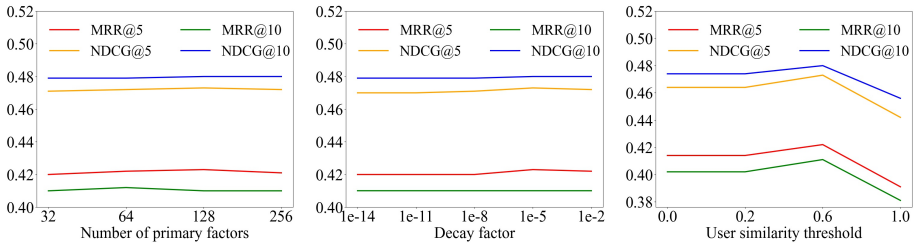

Figure 2: Sensitivity analysis on three hyper-parameters: number of primary factors, decay factor and similarity threshold.

the total time for training and inference of FIRE and the other incremental algorithms for model updating in each new time period. Note that "FIRE w/o EI" means FIRE without efficiency improvement. As we can see from the results, FIRE achieves the highest

Table 2: Ablation study on two datasets. Note that no user side information is available in the Douban Movie dataset.

		Setting	Douban Movie					
			F1@5	MRR@5	NDCG@5	F1@10	MRR@10	NDCG@10
		GF-CF	0.007	0.024	0.030	0.010	0.027	0.039
FIRE	(1)	$\delta_2 = \delta_3 = 0$	0.001	0.005	0.007	0.002	0.005	0.008
	(2)	$\delta_2 = 0, \delta_3 \neq 0$	**0.009**	**0.027**	0.034	**0.013**	0.030	0.045
	(3)	$\delta_2 \neq 0, \delta_3 = 0$	0.003	0.012	0.015	0.005	0.015	0.021
	(4)	$\delta_2 \neq 0, \delta_3 \neq 0$	**0.009**	**0.027**	**0.035**	**0.013**	**0.031**	**0.046**
		Setting	MovieLens 1M					
			F1@5	MRR@5	NDCG@5	F1@10	MRR@10	NDCG@10
		GF-CF	0.031	0.141	0.161	0.046	0.151	0.196
FIRE	(1)	$\delta_1 = \delta_2 = \delta_3 = 0$	0.004	0.074	0.090	0.009	0.079	0.112
	(2)	$\delta_1 = \delta_2 = 0, \delta_3 \neq 0$	0.035	0.149	0.178	0.052	0.148	0.205
	(3)	$\delta_1 \neq 0, \delta_2 = 0, \delta_3 = 0$	0.065	0.420	0.467	0.108	0.407	0.472
	(4)	$\delta_1 = 0, \delta_2 \neq 0, \delta_3 = 0$	0.008	0.074	0.090	0.014	0.081	0.116
	(5)	$\delta_1 \neq 0, \delta_2 \neq 0, \delta_3 = 0$	0.064	0.418	0.464	0.105	0.400	0.467
	(6)	$\delta_1 \neq 0, \delta_2 \neq 0, \delta_3 \neq 0$	**0.070**	**0.422**	**0.471**	**0.112**	**0.411**	**0.481**

Table 3: Performance on new user and item recommendation on two datasets.

MovieLens 1M (new user recommendation)			
Setting	F1@5	MRR@5	NDCG@5
FIRE w/o SI	0.004	0.093	0.118
FIRE w/ SI	**0.070**	**0.578**	**0.633**
Setting	F1@10	MRR@10	NDCG@10
FIRE w/o SI	0.009	0.083	0.127
FIRE w/ SI	**0.116**	**0.545**	**0.627**
Douban Movie (new item recommendation)			
Setting	F1@5	MRR@5	NDCG@5
FIRE w/o SI	0.001	0.010	0.017
FIRE w/ SI	**0.020**	**0.166**	**0.202**
Setting	F1@10	MRR@10	NDCG@10
FIRE w/o SI	0.002	0.011	0.019
FIRE w/ SI	**0.030**	**0.160**	**0.221**

efficiency, i.e., about 3X more efficient than SPMF and 6X more efficient than SML. The reason why FIRE is so efficient is that it does not have a very time-consuming training phase for model parameters. We also compare FIRE with FIRE w/o EI to verify the effectiveness of two efficiency improvement methods proposed in section 4.5. The results show that FIRE is about 2X faster than FIRE w/o EI, although FIRE w/o EI also achieves good efficiency comparing to the other incremental CF methods. Therefore, we conclude that the two efficiency improvement tricks are indeed helpful.

5.5 Hyper-parameter Analysis

We analyze the effects of three important hyper-parameters on the performance of FIRE in Figure 2.

• Number of primary factors l. With the increase of the number of primary factors, MRR@5 and NDCG@5 first increase and then decrease, reaching the peak at l= 128, while the MRR@10 and NDCG@10 fluctuate smoothly. It shows that when the number of primary factors is too small, FIRE cannot fully extract user and item features from user interactions, while when the number of primary factors is too large, no further improvements are observed due to model saturation.

• Decay factor k. With the increase of the decay factor, MRR@5 and NDCG@5 first increase and then decrease, while MRR@10 and NDCG@10 do not change much. When the decay factor is too large, compared with the interactions at the current period, FIRE is not able to extract the useful features of users and items from the historical interactions due to almost complete attenuation of the historical interactions. While when the decay factor is too small, FIRE can not distinguish between the historical interactions and the current interactions, making it unable to learn the temporal features of users and items over time.

• User similarity threshold ϵ_1. All metrics first increase and then decrease with the increase of the user similarity threshold, and reach the peak at $\epsilon_1 = 0.6$. When the user similarity threshold is too low, some dissimilar users also participate in the current user's recommendation, resulting in low recommendation accuracy and affecting the efficiency of the model. When the user similarity threshold is too high, only a few neighbors participate in each user's recommendation. Although it can greatly improve the efficiency, it can not obtain high recommendation accuracy.

5.6 New User/Item Recommendation Issue

We conduct experiments on two datasets to verify that FIRE can solve the new user/item recommendation issue when combined with side information. In the experiment, we use users/items that do not appear in the training phase as new users/items for test. The results are shown in table 3. Note that FIRE w/ SI and FIRE w/o SI represent FIRE combined with side information or not, respectively. We can see that the accuracy of FIRE w/ side information is much higher than that of FIRE w/o side information, indicating that the introduction of side information can effectively solve the new user/item recommendation issue. In fact, side information describes the static characteristics of users. When there is a lack of user interactions, FIRE can learn user preference from side information and recommend items to users. It should be noted that the values of MRR and NDCG in table 3 are much higher than those in table 1, because the rated items (or users) of each new user (or item) are

more than those of users (or items) that appear in the training phase on average, resulting in higher chance of being predicted.

6 CONCLUSION

We propose the fast incremental recommendation method — FIRE, which is a non-parametric method that does not suffer from the time-consuming back-propagations, thus significantly improves the efficiency of model updating. FIRE can capture the temporal dynamics of users/items and address the cold-start issue for new users/items through two carefully designed filters based on temporal information and side information, respectively. Extensive experiments demonstrate that FIRE can outperform the state-of-the-art incremental CF methods in both accuracy and efficiency.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (NSFC) under Grants 61932007 and 62172106.

REFERENCES

[1] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2017. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263* (2017).
[2] Chao Chen, Dongsheng Li, Junchi Yan, Hanchi Huang, and Xiaokang Yang. 2021. Scalable and Explainable 1-Bit Matrix Completion via Graph Signal Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 7011–7019.
[3] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems* (Boston, MA, USA) *(DLRS 2016)*. Association for Computing Machinery, New York, NY, USA, 7–10.
[4] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 2019. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 257–266.
[5] Hanjun Dai, Yichen Wang, Rakshit Trivedi, and Le Song. 2016. Deep coevolutionary network: Embedding user and item features for recommendation. *arXiv preprint arXiv:1609.03675* (2016).
[6] Samuel G Fadel and Ricardo da S Torres. 2018. Link Prediction in Dynamic Graphs for Recommendation. *arXiv preprint arXiv:1811.07174* (2018).
[7] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph Neural Networks for Social Recommendation. In *The World Wide Web Conference* (San Francisco, CA, USA) *(WWW '19)*. Association for Computing Machinery, New York, NY, USA, 417–426. https://doi.org/10.1145/3308558.3313488
[8] Wenqi Fan, Yao Ma, Dawei Yin, Jianping Wang, Jiliang Tang, and Qing Li. 2019. Deep Social Collaborative Filtering. In *Proceedings of the 13th ACM Conference on Recommender Systems* (Copenhagen, Denmark) *(RecSys '19)*. Association for Computing Machinery, New York, NY, USA, 305–313. https://doi.org/10.1145/3298689.3347011
[9] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 1025–1035.
[10] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Shinjuku, Tokyo, Japan) *(SIGIR '17)*. Association for Computing Machinery, New York, NY, USA, 355–364. https://doi.org/10.1145/3077136.3080777
[11] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, YongDong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. Association for Computing Machinery, New York, NY, USA, 639–648.
[12] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web* (Perth, Australia) *(WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 173–182.
[13] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. 1999. An Algorithmic Framework for Performing Collaborative Filtering. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Berkeley, California, USA) *(SIGIR '99)*. Association for Computing Machinery, New York, NY, USA, 230–237.
[14] Michel Journée, Yurii Nesterov, Peter Richtárik, and Rodolphe Sepulchre. 2010. Generalized power method for sparse principal component analysis. *Journal of Machine Learning Research* 11, 2 (2010).
[15] Santosh Kabbur, Xia Ning, and George Karypis. 2013. FISM: Factored Item Similarity Models for Top-N Recommender Systems. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Chicago, Illinois, USA) *(KDD '13)*. Association for Computing Machinery, New York, NY, USA, 659–667.
[16] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37.
[17] Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Anchorage, AK, USA) *(KDD '19)*. Association for Computing Machinery, New York, NY, USA, 1269–1278.
[18] Dongsheng Li, Chao Chen, Tun Lu, Stephen M. Chu, and Ning Gu. 2021. Mixture Matrix Approximation for Collaborative Filtering. *IEEE Transactions on Knowledge and Data Engineering* 33, 6 (2021), 2640–2653.
[19] Dongsheng Li, Chao Chen, Qin Lv, Junchi Yan, Li Shang, and Stephen M. Chu. 2016. Low-Rank Matrix Approximation with Stability. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning* (New York, NY, USA) *(ICML'16)*. JMLR.org, 295–303.
[20] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI '09*. 452–461.
[21] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. 2020. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637* (2020).
[22] Ruslan Salakhutdinov and Andriy Mnih. 2007. Probabilistic Matrix Factorization. In *Proceedings of the 20th International Conference on Neural Information Processing Systems* (Vancouver, British Columbia, Canada) *(NIPS'07)*. Curran Associates Inc., Red Hook, NY, USA, 1257–1264.
[23] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. AutoRec: Autoencoders Meet Collaborative Filtering. In *Proceedings of the 24th International Conference on World Wide Web* (Florence, Italy) *(WWW '15 Companion)*. Association for Computing Machinery, New York, NY, USA, 111–112.
[24] Yifei Shen, Yongji Wu, Yao Zhang, Caihua Shan, Jun Zhang, B. Khaled Letaief, and Dongsheng Li. 2021. How Powerful is Graph Convolution for Recommendation?. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. Association for Computing Machinery, New York, NY, USA, 1619–1629.
[25] Zhu Sun, Guibing Guo, Jie Zhang, and Chi Xu. 2017. A unified latent factor model for effective category-aware recommendation. In *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*. 389–390.
[26] Zhu Sun, Qing Guo, Jie Yang, Hui Fang, Guibing Guo, Jie Zhang, and Robin Burke. 2019. Research commentary on recommendations with side information: A survey and research directions. *Electronic Commerce Research and Applications* 37 (2019), 100879. https://doi.org/10.1016/j.elerap.2019.100879
[27] Flavian Vasile, Elena Smirnova, and Alexis Conneau. 2016. Meta-Prod2Vec: Product Embeddings Using Side-Information for Recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems* (Boston, Massachusetts, USA) *(RecSys '16)*. Association for Computing Machinery, New York, NY, USA, 225–232. https://doi.org/10.1145/2959100.2959160
[28] Saurabh Verma and Zhi-Li Zhang. 2019. Stability and Generalization of Graph Convolutional Neural Networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Anchorage, AK, USA) *(KDD '19)*. Association for Computing Machinery, New York, NY, USA, 1539–1548. https://doi.org/10.1145/3292500.3330956
[29] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. 2019. Knowledge Graph Convolutional Networks for Recommender Systems. In *The World Wide Web Conference* (San Francisco, CA, USA) *(WWW '19)*. Association for Computing Machinery, New York, NY, USA, 3307–3313.
[30] Weiqing Wang, Hongzhi Yin, Zi Huang, Qinyong Wang, Xingzhong Du, and Quoc Viet Hung Nguyen. 2018. Streaming Ranking Based Recommender Systems. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval* (Ann Arbor, MI, USA) *(SIGIR '18)*. Association for Computing Machinery, New York, NY, USA, 525–534.
[31] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Paris, France) *(SIGIR'19)*. Association for Computing Machinery, New York, NY, USA, 165–174.
[32] Yichao Wang, Huifeng Guo, Ruiming Tang, Zhirong Liu, and Xiuqiang He. 2020. A Practical Incremental Method to Train Deep CTR Models. *arXiv preprint arXiv:2009.02147* (2020).

Table 4: The statistics of the four datasets. Here, time span is described by year and month, e.g., 00.05 means May 2000.

	MovieLens 1M	Douban Movie	Amazon Book	Amazon Electronics
# Users	5,972	9,987	19,414	19,955
# Items	3,678	9,999	19,055	14,164
# Records	892,982	1,001,918	344,201	194,494
Training time span	00.05–00.11	08.01–17.12	13.01–13.07	15.01–15.10
Test time span	00.12	18.01 19.12	13.08–13.09	15.11–15.12
# Training records	777,387	785,047	273,785	163,906
# Test records	115,595	216,871	70,416	30,588
Density	0.0399	0.0100	0.0009	0.0007
User side information	age, gender, occupation	–	–	–
Item side information	category	genre, language, region	category	category

[33] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J. Smola, and How Jing. 2017. Recurrent Recommender Networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining* (Cambridge, United Kingdom) *(WSDM '17)*. Association for Computing Machinery, New York, NY, USA, 495–503.
[34] Jiafeng Xia, Dongsheng Li, Hansu Gu, Tun Lu, Peng Zhang, and Ning Gu. 2021. Incremental Graph Convolutional Network for Collaborative Filtering. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. ACM, New York, NY, USA, 2170–2179.
[35] Yantao Yu, Zhen Wang, and Bo Yuan. 2019. An Input-aware Factorization Machine for Sparse Prediction.. In *IJCAI*. 1466–1472.
[36] Yang Zhang, Fuli Feng, Chenxu Wang, Xiangnan He, Meng Wang, Yan Li, and Yongdong Zhang. 2020. How to retrain recommender system? A sequential meta-learning method. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. Association for Computing Machinery, New York, NY, USA, 1479–1488.
[37] Yao Zhang, Yun Xiong, Dongsheng Li, Caihua Shan, Kan Ren, and Yangyong Zhu. 2021. *CoPE: Modeling Continuous Propagation and Evolution on Interaction Graph*. ACM, New York, NY, USA, 2627–2636.

A DETAILS OF DATASETS AND BASELINES

A.1 Datasets

Table 4 describes the detailed statistics of the four datasets used in the experiments, including MovieLens 1M, Douban Movie, Amazon Book and Amazon Electronics. In the MovieLens 1M dataset, we first select a dense 8 months period containing approximately 900K ratings and then use the ratings from the first seven months as training set and the rest as test set. For the Douban Movie dataset, we select the most dense data from 12 years, keep the top 10,000 users and 10,000 items, and we use their ratings in the first ten years as training set and the rest as test set. In the Amazon Book dataset, we first select a 9-month period containing 344,201 ratings with 19,414 users and 19,055 items, then we use the ratings from the first 7 months as the training set and the rest as test set. For the Amazon Electronics dataset, we select the user purchase records from one year as a dense subset and keep top 20,000 users and 15,000 items, then we use ratings from the first ten months as the training set and the rest as test set.

A.2 Baselines

FIRE is compared with eleven state-of-the-art CF methods.

• BPR [20] is a static CF method minimizing a pair-wise loss over positive and unlabeled examples. In BPR, we use MF [16] to initialize user and item embeddings.

• LightGCN [11] is a GCN based static CF algorithm which simplified but outperformed the NGCF method [31] by removing feature transformation and nonlinear activation.

• NFM [10] is a deep learning based static CF method that combines the linearity of FM and non-linearity of neural network to achieve recommendation.

• IFM [35] is a deep learning based static CF method that learns a unique input-aware factor for the same feature in different instances to make accurate recommendation.

• GF-CF [24] is a graph signal processing based non-parametric method that achieves high accuracy recommendation on static interaction graphs.

• RRN [33] is a dynamic CF method that uses two different recurrent neural networks (RNNs) to model the temporal dynamics of users and items, respectively.

• DeepCoevolve [5] is a graph based incremental CF method that uses RNN to model the intensity function in point process to capture temporal dynamics of users and items. We disabled their model updates in test phase for fair comparison.

• JODIE [17] is a graph based incremental CF method that employs two RNNs to learns the embedding trajectories of users and items from interaction graph, respectively. We disabled their model updates in test phase for fair comparison.

• SPMF [30] is an incremental CF method that maintains historical data in a reservoir and combines historical and new observations to update the recommendation.

• IncCTR [32] is an incremental CF method that consists of three decoupled modules to construct data, handle feature and finetune model, respectively.

• SML [36] is a state-of-the-art incremental CF method which can transform the old model to a new model via a neural network-based transfer component.