Structure Aware Incremental Learning with Personalized Imitation Weights for Recommender Systems

Yuening Wang[1], Yingxue Zhang[1], Antonios Valkanas[2*], Ruiming Tang[1], Chen Ma[3], Jianye Hao[1, 4], Mark Coates[2]

[1] Huawei Noah's Ark Lab
[2] McGill University
[3] City University of Hong Kong
[4] Tianjin University

yuening.wang@huawei.com, yingxue.zhang@huawei.com, antonios.valkanas@mail.mcgill.ca, tangruiming@huawei.com, chenma@cityu.edu.hk, haojianye@huawei.com, mark.coates@mcgill.ca

Abstract

Recommender systems now consume large-scale data and play a significant role in improving user experience. Graph Neural Networks (GNNs) have emerged as one of the most effective recommender system models because they model the rich relational information. The ever-growing volume of data can make training GNNs prohibitively expensive. To address this, previous attempts propose to train the GNN models incrementally as new data blocks arrive. Feature and structure knowledge distillation techniques have been explored to allow the GNN model to train in a fast incremental fashion while alleviating the catastrophic forgetting problem. However, preserving the same amount of the historical information for all users is sub-optimal since it fails to take into account the dynamics of each user's change of preferences. For the users whose interests shift substantially, retaining too much of the old knowledge can overly constrain the model, preventing it from quickly adapting to the users' novel interests. In contrast, for users who have static preferences, model performance can benefit greatly from preserving as much of the user's long-term preferences as possible. In this work, we propose a novel training strategy that adaptively learns personalized imitation weights for each user to balance the contribution from the recent data and the amount of knowledge to be distilled from previous time periods. We demonstrate the effectiveness of learning imitation weights via a comparison on five diverse datasets for three state-of-art structure distillation based recommender systems. The performance shows consistent improvement over competitive incremental learning techniques.

Introduction

The growth of online services has rendered recommender systems a vital part of providing personalized recommendations to users. Making highly relevant recommendations improves user experience and increases the service provider's revenue. Deep learning models are becoming more prevalent in all aspects of recommender system design due to their superiority in constructing high-quality user and item representations in an end-to-end fashion (Covington, Adams, and Sargin 2016; Guo et al. 2017; Cheng et al. 2016). There is a recent trend to formulate the recommendation problem as a learning task on graphs because of the rich relational information that graphs can model. Much of the data from recommender systems can naturally be expressed using graph structures (van den Berg, Kipf, and Welling 2018; Wang et al. 2019b, 2021; Sun et al. 2019). For example, we can use the user-item bipartite interaction graph, an item similarity graph, a user-user graph derived from social network exchanges, and an additional knowledge graph to improve the representation learning process. Graph Neural Network (GNN) based recommender systems have emerged as one of the most effective models because the message-passing paradigm allows sufficient modeling of the relational information in the data. However, training GNNs on large-scale graphs can be prohibitively expensive (Ying et al. 2018; Zou et al. 2019; Chiang et al. 2019; Zeng et al. 2020; Qiu et al. 2020; Xu et al. 2020), which makes deploying models with GNN backbone networks extremely challenging on large-scale recommender systems.

One approach to address the computation issue is to train the deep learning models incrementally as new data blocks arrive (Kirkpatrick et al. 2017; Shmelkov, Schmid, and Alahari 2017; Castro et al. 2018; Rebuffi et al. 2017; Mallya and Lazebnik 2018; Xu and Zhu 2018; Qiu et al. 2020). However, directly using the data from the incremental block to fine-tune a model can lead to catastrophic forgetting (Kirkpatrick et al. 2017; Shmelkov, Schmid, and Alahari 2017). Because of their superiority in terms of efficiency and performance, knowledge distillation approaches (Castro et al. 2018; Kirkpatrick et al. 2017; Xu et al. 2020; Wang, Zhang, and Coates 2021) are preferable for models with a GNN backbone architecture. Benefiting from the knowledge distillation paradigm, key information from the historical data is preserved and transferred to the student model trained using only newly arrived data. This is achieved by regularizing the distance between the representations of the teacher and the student models. Both feature and structure knowledge distillation techniques have been explored; these allow the GNN model to better preserve both the node feature and structure information in the previous training data while enjoying the fast incremental training process (Yang et al.

*Work done as an intern at Huawei Noah's Ark Lab

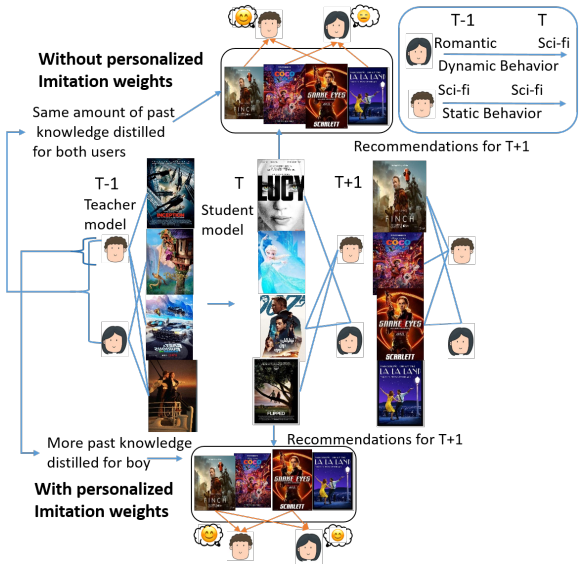

Figure 1: Motivation of the designed framework where the boy has a constant preference from time $t-1$ to t, while the girl's interests change more dynamically. Capturing the interest shift difference between users via a personalized incremental learning scheme will be beneficial.

2020; Xu et al. 2020; Wang, Zhang, and Coates 2021).

However, preserving the same amount of historical information for all users without any distinction between them might be sub-optimal since it fails to take into account the dynamics of each user's potential change of preferences. For users whose interests shift substantially, retaining too much of the old knowledge from the past via the knowledge distillation process might prevent the model from quickly adapting to the users' latest interests. In contrast, for users who have more static preferences, model performance can benefit greatly from preserving as much of the user's long-term preference as possible. We illustrate this with an example in Figure 1.

Thus, how to design an incremental learning training scheme that can model the dynamics of users' personalized preference change to determine how much knowledge to preserve from the past is an intriguing and important research question. We address this research question by targeting an important hyper-parameter in knowledge distillation objective functions, named the *imitation weight*, which is used to balance the contributions to the overall loss from the new data and from distillation.

In this work, following the above intuition, we propose a novel end-to-end training strategy that adaptively learns *personalized imitation weights* for each user to better balance the contributions from the recent data and the amount of knowledge to be distilled from previous time windows. Specifically, we first model each user's preference as a distribution over the distance to item cluster centers, with the clusters being obtained by a deep structural clustering method applied to the user-item bipartite graph. The cluster learning process is integrated into the overall training procedure. Then we construct, for each user, a state vector that encodes the distance between two preference distributions associated with that user, which are derived from consecutive training blocks. This state vector is passed as the input to a weight generator parameterized by a neural network to produce a user-specific imitation weight. This personalized imitation weight determines how much information pertinent to the user is inherited from the teacher (historical) model. Our proposed approach is not restricted to a specific backbone architecture or incremental learning procedure. It can easily be integrated with multiple existing state-of-the-art methods.

To summarize, the main contributions of the paper are:

1. We demonstrate that explicitly assessing the user interest shift between consecutive training blocks and using this signal to learn a user-specific imitation weight is an important modeling factor. It can significantly impact the performance of knowledge distillation-based incremental learning techniques. To the best of our knowledge, this is the first incremental learning training scheme that explicitly models user change of preferences.
2. We propose a novel training strategy that adaptively learns personalized imitation weights for each user to balance the contribution from recent data and the amount of knowledge that is distilled from previous periods.
3. We demonstrate the effectiveness of learning imitation weights via a thorough comparison on five diverse datasets. Our best-performing model improves the SOTA method by 2.30%. We integrate our proposed training procedure with three recent SOTA incremental learning techniques for recommender systems. We show consistent improvement over the non-adaptive counterparts.

Related Work

Incremental Learning

Incremental learning is a branch of machine learning that aims to develop models which are updated continuously with new data. However, naively training on new batches of data as they arrive leads to the problem of *catastrophic forgetting* that the model forgets previously learned information and is overly biased to new data (Kirkpatrick et al. 2017; Shmelkov, Schmid, and Alahari 2017; Castro et al. 2018).

There are two main groups of approaches to combat this issue:(i) regularization-based knowledge distillation (Castro et al. 2018; Kirkpatrick et al. 2017; Xu et al. 2020; Wang, Zhang, and Coates 2021) and (ii) experience replay, also referred to as reservoir sampling (Prabhu, Torr, and Dokania 2020; Ahrabian et al. 2021). Reservoir methods sample a data reservoir containing the most representative historical data and replay it while learning new tasks to alleviate forgetting. Some key reservoir sampling works such as iCarl (Rebuffi et al. 2017) and GDumb (Prabhu, Torr, and Dokania 2020) focus on optimizing the reservoir construction either via direct optimization or via greedy heuristics. Recent work on graph recommender systems expands on the GDumb heuristic and proposes inverse degree sampling of nodes for reservoir construction (Ahrabian et al. 2021).

Regularization techniques typically introduce penalty parameters in the loss function to prevent the model weights

from "drifting" too far from their tuned values from historical data blocks, thus preventing forgetting (Yang et al. 2019; Xu et al. 2020; Wang, Zhang, and Coates 2021). Knowledge distillation is one of the most common regularization approaches. Knowledge distillation refers to the process of transferring knowledge from a large and complex teacher model to a smaller student model without significant loss in performance (Hinton, Vinyals, and Dean 2015). In incremental learning, the *teacher model* is trained on the historical data and the *student model* is trained on the new data.

Although both incremental learning and sequential learning take advantage of historical information, sequential recommendation learning is different from incremental learning in several important aspects. First, incremental learning aims to substantially reduce the training sample number by inheriting the knowledge from the previously trained model with knowledge distillation or experience reply. In contrast, sequential recommender systems focus on better characterizing the user's long-term or short-term interaction sequences through memory units (Hidasi and Karatzoglou 2018), Recurrent Neural Networks (RNNs)or attention design (Kang and McAuley 2018; Fan et al. 2021). The former is a training strategy in the scenario that requires incremental updates and the latter is a specific model architecture to handle the given sequence of data. Second, incremental learning in the context of recommender systems is agnostic to any type of backbone architecture. Both sequential and non-sequential models should be compatible with the incremental training method.

Incremental Learning on Graph Structured Data

Graph representation learning techniques have become a mainstream tool for collaborative filtering and recommender systems (Sun et al. 2019; Ying et al. 2018; Wang et al. 2019b; He et al. 2020). However, they suffer from a computation and memory burden introduced by either the neighborhood sampling process, message passing procedure or the storage of the adjacency matrix, which prevents GNN models from satisfying a strict training time constraint for online systems (Xu et al. 2020). Several incremental recommender system designs have been proposed to tailor the GNN models to better preserve the structural information. GraphSAIL (Xu et al. 2020) employs knowledge distillation at the node level, the node neighborhood, and the global graph level. LSP_s (Yang et al. 2020) minimizes the distance between structure-related distributions drawn from the model trained at previous time steps and the fine-tuned model. SGCT (Wang, Zhang, and Coates 2021) introduces a contrastive approach to knowledge distillation. The objective of SGCT is to maximize the lower bound of the mutual information between pairs of adjacent node embeddings from the student and the teacher model. LWC-KD (Wang, Zhang, and Coates 2021) improves over SGCT by considering intermediate layer embedding distillation and additional contrastive distillation on the user-user and item-item graph.

Novelty of our work Prior works focus on universally distilling as much information as possible for all users, without distinguishing between them. However, this is not always optimal or desirable as the interests of some users may shift quickly over time and the recommendation models should be able to adapt quickly to the new user preference. Our work proposes an adaptive weight mechanism to learn the amount of knowledge to distill for each user. We show experimentally that personalizing the distillation strength by assessing how rapidly each user's interests are changing can lead to significantly better recommendation performance for state-of-art backbones.

Methodology

In this section, we present the proposed **S**tructure **A**ware **I**ncremental **L**earning with **P**ersonalized **I**mitation **W**eight frameworks, abbreviated to SAIL-PIW, which exploits personalized imitation weights by characterizing the user interest distribution shift between the historical data and the newly arrived incremental data. The personalized imitation weight is used in the knowledge distillation loss to balance the contribution of the recent data and the amount of historical knowledge to be distilled from previous time periods. To better model each user's preference, we learn a distribution over the distance to item cluster centers, with the clusters being obtained by a structural clustering method applied to the user-item bipartite graph. Once we obtain the user interest distribution, we then use the difference of the interest distribution between the incremental learning block and the teacher model as a user interest shift indicator. This shift indicator models the change of user preferences between the most recent training block and the newly arrived incremental block. The shift indicator is then passed to a weight generator, parameterized by a multi-layer perceptron to output a user-specific imitation weight. Our proposed framework is trainable in an end-to-end fashion via back-propagation.

The overall architecture of our model is presented in Figure 2. In the following subsections, we initially describe in detail the individual components of our proposed method. Thus, we delicately separate sections into (i) the proposed personalized distillation loss, (ii) the architecture of the imitation weight generator, (iii) the structure-aware item clustering technique to obtain the cluster center embedding, and finally (iv) the way we construct the metric to characterize the user interest shift. It is important to note that our framework is not limited to a particular GNN model, graph-based recommender system architectureor a specific incremental learning framework. We illustrate promising results of using GraphSAIL, SGCT and LWC-KD as backbones. Thus, our proposed solution is highly appealing in real-world settings as it can be readily applied on top of existing graph-based systems regardless of the base models that these systems already employ.

Personalized Distillation Loss

Knowledge distillation is commonly used for model compression in which a small model (student model) aims to achieve approximately equivalent performance to a larger model (teacher model) by inheriting important knowledge from the larger model (Hinton, Vinyals, and Dean 2015). Typically, when applying knowledge distillation in an incremental learning setting we use a *teacher model* trained on

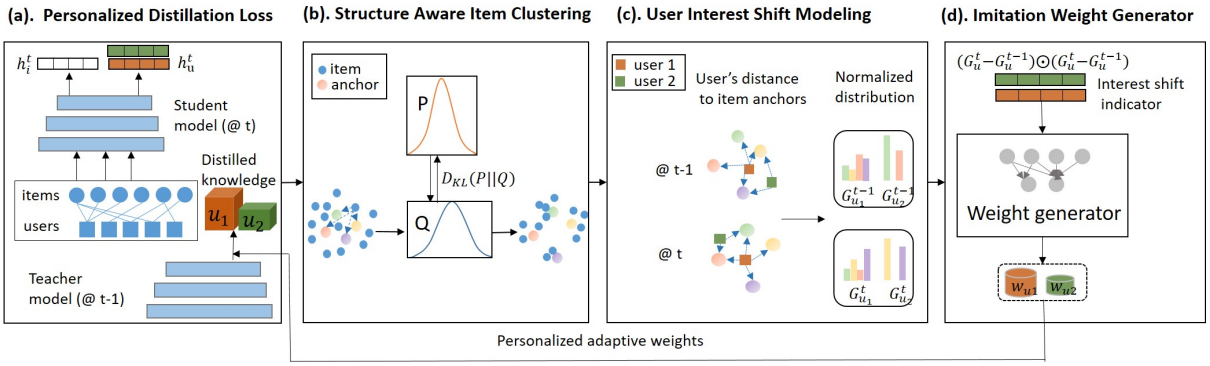

Figure 2: Overall framework of SAIL-PIW. (a). Student model at time t takes a new user-item interaction graph as input to learn user and item embeddings, regularized by knowledge distillation from the teacher model at time point $t-1$. (b). We learn the distance of an item embedding from the GNN model to a learnable item anchor embedding. It minimizes the KL divergence between learned items' distribution to clusters P and target distribution Q. (c). The user's distribution to item anchors is calculated for both time point $t-1$ and time point t. Two types of users are illustrated: user 1 with low interest shift and user 2 with high interest shift. (d). The user interest shift is calculated as the difference of user's normalized distance distribution to item cluster centers. The weight generator produces a personalized imitation weight.

historical data and a *student model* trained on the incremental data block. When training the student model, a distillation term is introduced to the loss (see eq. (1)), in order to retain the knowledge acquired by the teacher model. The objective function for training the incremental learning model can be formulated as:

$$\mathcal{L}_S = \mathcal{L}_{new}(\boldsymbol{y}_S, \tilde{\boldsymbol{y}}_S) + \lambda \mathcal{L}_{KD}(\tilde{\boldsymbol{y}}_T, \tilde{\boldsymbol{y}}_S), \quad (1)$$

where $\mathcal{L}_{new}$ denotes the student model's loss function between ground truth labels $\boldsymbol{y}_S$ and the predicted values $\tilde{\boldsymbol{y}}_S$, and $\mathcal{L}_{KD}$ denotes the knowledge distillation loss between the teacher and the student models. Here, λ denotes a scalar that controls the amount of distillation loss involved during training (Hinton, Vinyals, and Dean 2015).

In the context of recommender systems, $\mathcal{L}_{new}$ is often a Bayesian personalized ranking (BPR) (Rendle et al. 2009) loss. In a recent GNN-based knowledge distillation approach called GraphSAIL (Xu et al. 2020) $\mathcal{L}_{KD}$ consists of self-embedding distillation, global structure distillation, and local structure distillation. Though previous works have achieved promising performance and efficient training processes (Xu et al. 2020; Wang, Zhang, and Coates 2021; Qiu et al. 2020), they preserve previous knowledge using a single imitation weight λ that applies to all users. However, fully preserving the historical information for all users is sub-optimal since it fails to take into account the dynamics of each user's change of preferences. In practice, we observe that user preferences are dynamic and different users can be expected to exhibit different levels of interest change between the past time blocks and the arrival of the incremental data block. For users whose interests shift significantly from previous time periods, we do not want the student model's learning on the incremental data to be overly constrained by the teacher model. Therefore, following this intuition, we aim to distinguish different distillation levels for different users, so we propose to adaptively learn an imitation weight w_u for each user and apply it to the knowledge distillation objective function. We elaborate on the process for learning w_u in the following subsections.

Our personalized knowledge distillation loss is:

$$\mathcal{L}^U_{\text{KD - PIW}} = \sum_{u \in \mathcal{U}} \boldsymbol{w}_u \mathcal{L}^u_{KD}(\tilde{\boldsymbol{y}}_{u,T}, \tilde{\boldsymbol{y}}_{u,S})$$

where $\boldsymbol{w}_u$ is the personalized imitation weight learned for each user to identify the amount of knowledge to retain from the teacher model. $\mathcal{L}^u_{KD}$ refers to the knowledge distillation loss for a user u.

The overall incremental learning training objective is:

$$\mathcal{L}_S = \mathcal{L}_{new}(\boldsymbol{y}_S, \tilde{\boldsymbol{y}}_S) + \lambda \left(\mathcal{L}^U_{\text{KD - PIW}} + \mathcal{L}^I_{\text{KD}}\right) \quad (2)$$

where$\mathcal{L}^I_{\text{KD}} = \sum_{i \in \mathcal{I}} \mathcal{L}^i_{KD}(\tilde{\boldsymbol{y}}_{i,T}, \tilde{\boldsymbol{y}}_{i,S})$ is the knowledge distillation loss among all the items and $\mathcal{L}^i_{KD}$ refers to knowledge distillation loss for a item i. Since item knowledge is usually static by nature, we adopt the original knowledge distillation without imitation weight personalization.

Imitation Weight Generator

Directly learning the personalized imitation weight $\boldsymbol{w}_u$ will introduce a large number of learnable parameters because of the large user number. Instead, we propose to learn the personalized imitation weights using a learnable function parameterized by a neural network. For each training block, we associate with each user a state vector $\boldsymbol{s}_u \in \mathbb{R}^M$ which represents the user interest change between consecutive time blocks. Here M is the total number of clusters of items. We explain how we construct, initialize and update this state vector below. For now, assuming we have such a vector $\boldsymbol{s}_u$, we apply it as input to a weight generator network $f(\boldsymbol{s}_u)$ to learn the imitation weight of each user for distillation. In our realization of the framework, $f(\boldsymbol{s}_u)$ is specified by the following equations:

$$\boldsymbol{z_u} = \text{relu}(\boldsymbol{W}_1 \cdot \boldsymbol{s}_u + b_1), \quad (3)$$

$$w_u = \text{softplus}(\boldsymbol{W}_2 \cdot \boldsymbol{z}_u + b_2), \quad (4)$$

with $\boldsymbol{W}_1 \in \mathbb{R}^{M\times l}$, $\boldsymbol{W}_2 \in \mathbb{R}^{l}$, $\boldsymbol{b}_1 \in \mathbb{R}^{l}$, and $\boldsymbol{b}_2 \in \mathbb{R}$ as the learnable parameters, where l is size of the hidden layer. We note that applying a more advanced network as the weight generator may further improve the performance, but this is not the focus of this work. Thus, we use a simple form where the weight generator is parameterized by a 2-layer multi-layer perceptron (MLP). In the final layer, we adopt a softplus (Zheng et al. 2015) activation function to produce a strictly non-negative imitation weight. An additional normalization is applied across all the user imitation weights in each training mini-batch to better enhance stability during the training process.

Structure Aware Item Clustering

To derive the interest shift state vector, it is first necessary to cluster the items. Inspired by the deep structure clustering method (Wang et al. 2019a; Bo et al. 2020), where both node attributes and higher-order structural information are fully considered during the clustering process, we adopt a similar objective function to learn the item center clusters given the underlying user-item bipartite graph.

We measure the similarity between item embeddings and the item cluster centers' embeddings. Let $q^t_{i,m} \in \mathbb{R}^1$ denote the similarity between the item embedding from the final layers of the GNN backbone encoder $\boldsymbol{h}^t_i \in \mathbb{R}^d$ at time point t and the item cluster center embedding (item anchor) $\boldsymbol{\mu}^t_m \in \mathbb{R}^d$. We measure this distance using a Student's t-distribution to handle differently scaled clusters in a computationally convenient manner (Van der Maaten and Hinton 2008). This can be seen as a soft clustering assignment distribution of each item. The distribution mass of item i at current time block t for item cluster m is calculated as:

$$q^t_{i,m} = \frac{(1 + ||\boldsymbol{h}^t_i - \boldsymbol{\mu}^t_m||^2/\nu)^{-\frac{\nu+1}{2}}}{\sum_{m'\in M}(1 + ||\boldsymbol{h}^t_i - \boldsymbol{\mu}^t_{m'}||^2/\nu)^{-\frac{\nu+1}{2}}}, \quad (5)$$

where M is the total number of item clusters.

The deep structural clustering model we adopt is trained by a self-supervised learning loss using the Kullback–Leibler (KL) divergence (Kullback and Leibler 1951):

$$\mathcal{L}^{kl}_{soft} = D_{KL}(P||Q)^t = \sum_i \sum_m p^t_{i,m} log \frac{p^t_{i,m}}{q^t_{i,m}} \quad (6)$$

where $p^t_{i,m} = \frac{(q^t_{i,m})^2/f^t_m}{\sum_{m'\in M}(q^t_{i,m'})^2/f^t_{m'}}$, $f^t_m = \sum_i q^t_{i,m}$ and $p^t_i \in \mathbb{R}^M$ is the target distribution for item i at time point t which strives to push the representations closer to cluster centers. With the clusters defined, we can now derive $\mathbf{s}_u$.

User Interest Shift Modelling

In this section, we detail the generation and initialization of $\mathbf{s}_u$. We use $\boldsymbol{\mu}^t_m$ as an item cluster anchor embedding. We calculate the user distance to clusters as $\tilde{\boldsymbol{G}}^t_u = [\boldsymbol{\mu}^t_1 \boldsymbol{W}_1 (\boldsymbol{h}^t_u)^T, ..., \boldsymbol{\mu}^t_M \boldsymbol{W}_M (\boldsymbol{h}^t_u)^T]$ and apply softmax:

$$\boldsymbol{G}^t_{u,m} = \frac{e^{\tilde{\boldsymbol{G}}^t_{u,m}}}{\sum_{m'=1}^{M} e^{\tilde{\boldsymbol{G}}^t_{u,m'}}}, \quad (7)$$

where $\boldsymbol{W}_m \in \mathbb{R}^{d\times d}$ is a cluster-specific transformation matrix. Similarly, using the node embeddings from the previous time block we can get $\boldsymbol{G}^{t-1}_u$. We hypothesize that user interest shift is strongly related to the change of the users' distribution of distances to item clusters. Therefore, we design the state vector with the assumption that the importance weights of users are related to their distribution changes between two time blocks. We define the state vector as:

$$\boldsymbol{s}_u = (\boldsymbol{G}^{t-1}_u - \boldsymbol{G}^t_u) \odot (\boldsymbol{G}^{t-1}_u - \boldsymbol{G}^t_u) \quad (8)$$

where $\odot$ denotes the Hadamard product.

Our framework is compatible with any graph-based recommender system incremental learning architecture such as GraphSAIL (Xu et al. 2020), SGCT (Wang, Zhang, and Coates 2021) and LWC-KD (Wang, Zhang, and Coates 2021). These are standard state-of-the-art incremental learning approaches for recommender systems.

The Overall Training Framework

Having illustrated the detailed design of the adaptive knowledge distillation loss as well as the state vector $\mathbf{s}_u$ which characterizes the user interest shift behavior, we now focus on presenting the overall training objective function. Our model is trained in a fully end-to-end fashion where the BPR loss $\mathcal{L}_{\text{BPR}}$, which is applied on the incremental block, the personalized knowledge distillation loss for users $\mathcal{L}^U_{\text{KD - PIW}}$, the distillation loss for items $\mathcal{L}^I_{\text{KD}}$, and the self-supervised loss $\mathcal{L}^{kl}_{soft}$ for item clustering are combined jointly as follows:

$$\mathcal{L} = \mathcal{L}_{\text{BPR}} + \lambda_1 \mathcal{L}^{kl}_{soft} + \lambda_2 \left(\mathcal{L}^U_{\text{KD - PIW}} + \mathcal{L}^I_{\text{KD}}\right). \quad (9)$$

Here λ_1 and λ_2 are the coefficients that balance the loss contributions between the three terms.

Dataset	Gowalla	Yelp	Tbao14	Tbao15	Netflix
# e(M)	0.28	0.94	0.75	1.33	12.40
# users	5992	40863	8844	92605	63691
u deg	46.96	23.06	84.74	14.39	194.73
# items	5639	25338	39103	9842	10271
i deg	49.90	37.19	19.17	135.40	1207.56
% new u	2.67	3.94	1.67	2.67	4.36
% new i	0.67	1.72	2.60	0.22	0.72
# months	19	6	1	5	6

Table 1: Data Statistics. # e(M) represents the number of edges in millions. % new u/i refers to the average percentage of new users/items relative to all users/items in each incremental block. u/i deg refers to average user/item degrees

EXPERIMENTS

Datasets We use a diverse set of datasets consisting of real-world user-item interactions. As shown in Table 1, the datasets vary in the number of edges and number of user and item nodes by up to two orders of magnitude, demonstrating our approach's scalability. The 5 mainstream, publicly available datasets we use are: Gowalla, Yelp, Taobao14, Taobao15 and Netflix.

Gowalla					
Methods	Inc 1	Inc 2	Inc 3	Avg.	Imp %
Fine Tune	0.1412	0.1637	0.2065	0.1705	0.00
LSP_s	0.1512	0.1741	0.2097	0.1783	4.57
Uniform	0.1480	0.1653	0.2051	0.1728	1.34
Inv_degree	0.1483	0.1680	0.2001	0.1738	1.93
GraphSAIL	0.1529	0.1823	0.2195	0.1849	8.44
GraphSAIL-PIW	0.1547	0.1825	0.2253	0.1875	9.97
SGCT	0.1588	0.1815	0.2207	0.1870	9.68
SGCT-PIW	0.1599	0.1892	0.2321	0.1937	13.6
LWC-KD	0.1639	0.1921	0.2368	0.1977	15.9
LWC-KD-PIW	0.1698	0.1978	0.2425	**0.2033**	**19.3**
Yelp					
Fine Tune	0.0705	0.0638	0.0640	0.0661	0.00
LSP_s	0.0722	0.0661	0.0644	0.0676	2.27
Uniform	0.0718	0.0635	0.0610	0.0654	-1.05
Inv_degree	0.0727	0.0699	0.0605	0.0677	2.42
GraphSAIL	0.0674	0.0617	0.0625	0.0639	-3.33
GraphSAIL-PIW	0.0718	0.0638	0.0615	0.0657	-0.66
SGCT	0.0740	0.0656	0.0608	0.0668	1.06
SGCT-PIW	0.0735	0.0655	0.0632	0.0674	1.92
LWC-KD	0.0739	0.0661	0.0637	0.0679	2.72
LWC-KD-PIW	0.0760	0.0690	0.0651	**0.0700**	**5.95**
Taobao14					
Fine Tune	0.0208	0.0112	0.0138	0.0153	0.00
LSP_s	0.0213	0.0106	0.0138	0.0152	-0.65
Uniform	0.0195	0.0127	0.0148	0.0157	2.61
Inv_degree	0.0228	0.0140	0.0159	0.0175	14.63
GraphSAIL	0.0222	0.0105	0.0139	0.0155	1.31
GraphSAIL-PIW	0.0206	0.0103	0.0129	0.0146	-4.58
SGCT	0.0240	0.0092	0.0148	0.0160	1.74
SGCT-PIW	0.0227	0.0104	0.0142	0.0158	3.05
LWC-KD	0.0254	0.0119	0.0156	0.0176	15.3
LWC-KD-PIW	0.0256	0.0118	0.0161	**0.0178**	**16.3**
Taobao15					
Fine Tune	0.0933	0.0952	0.0965	0.0950	0.00
LSP_s	0.0993	0.0952	0.0957	0.0968	1.86
Uniform	0.0988	0.0954	0.1004	0.0982	3.37
Inv_degree	0.0991	0.0977	0.1000	0.0989	4.16
GraphSAIL	0.0959	0.0959	0.0972	0.0963	1.39
GraphSAIL-PIW	0.1024	0.0983	0.1018	0.1008	6.14
SGCT	0.1030	0.0983	0.0984	0.0999	5.16
SGCT-PIW	0.1040	0.0999	0.1027	0.1022	7.58
LWC-KD	0.1039	0.1022	0.1029	0.1030	8.42
LWC-KD-PIW	0.1044	0.1045	0.1052	**0.1047**	**10.2**
Netflix					
Fine Tune	0.1092	0.1041	0.0977	0.1036	0.00
LSP_s	0.1173	0.1136	0.1076	0.1128	8.88
Uniform	0.1018	0.1055	0.0800	0.0957	-7.63
Inv_degree	0.1000	0.1050	0.0820	0.0957	-7.63
GraphSAIL	0.1163	0.1023	0.0968	0.1051	1.45
GraphSAIL-PIW	0.1142	0.1028	0.0986	0.1052	1.54
SGCT	0.1166	0.1161	0.1077	0.1135	9.56
SGCT-PIW	0.1185	0.1144	0.1098	0.1142	10.23
LWC-KD	0.1185	0.1170	0.1071	**0.1142**	**10.23**
LWC-KD-PIW	0.1185	0.1146	0.1087	0.1139	9.97

Table 2: Performance comparison (Recall@20) of all baselines and three recent knowledge distillation algorithms with our proposed personalized adaptive weights design. The improvement ratio is with respect to fine-tune performance.

Base Model We use MGCCF (Sun et al. 2019) as our base model in the incremental learning methods. It is a state-of-art backbone model in the incremental recommendation framework which not only incorporates multiple graphs in the embedding learning process, but also considers the intrinsic difference between user nodes and item nodes when performing graph convolution on the bipartite graph.

Baselines To demonstrate that our model's strength, we compare our algorithm with multiple baselines.
Fine Tune: Fine Tune uses solely the new data of each time block to fine-tune the model that was trained using the previous time blocks.
LSP_s (Yang et al. 2020): LSP is a recent state-art-of approach which applies knowledge distillation on Graph Convolution Network (GCN) models. It preserves local structure from the teacher to student by minimizing the distances between distributions representing local topological semantics.
Uniform: This is a naive reservoir replay method. A subset of old data is sampled and added to the new data.
Inv_degree (Ahrabian et al. 2021): Inv_degree is a state-of-art reservoir replay method. The reservoir is based on graph structure. The approach selects a fixed-size subset of user-item pairs from historical data; each interaction's selection probability is proportional to the inverse degree of its user.
SOTA Graph Rec. Sys. Incremental Learning methods: GraphSail (Xu et al. 2020), SGCT (Wang, Zhang, and Coates 2021) and LWC-KD (Wang, Zhang, and Coates 2021) are state-of-the-art models which we improve upon by integrating our approach. To demonstrate the strength of our method we compare with the base models.

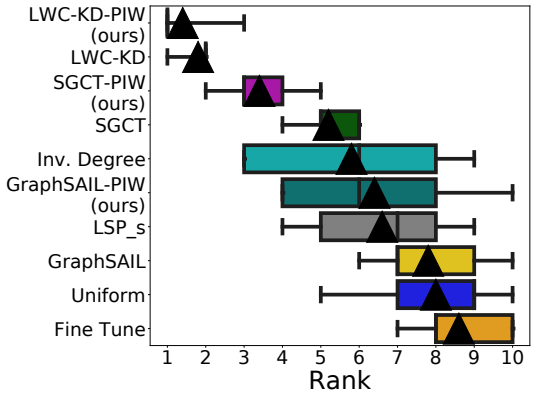

Figure 3: Boxplot of ranks of the algorithms across the 5 datasets. The medians and means of the ranks are shown by the vertical lines and the black triangles respectively; whiskers extend to the minimum and maximum ranks.

Results and Discussion

Table 2 reports the performance of baselines and three distillation algorithms with/without adaptive weights along with standard reservoir replay methods. Please note that all the results reported are an average across three trails with different random seeds. The results across all datasets of Table 2 are summarized in Figure 3. We note that our adaptive weight framework improves method performance in all cases, since

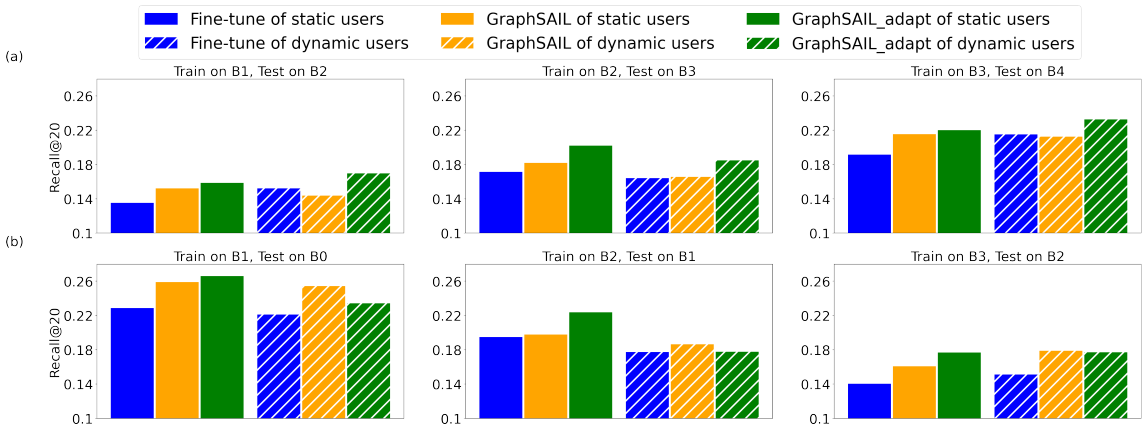

Figure 4: Case study: (a) Model performance for: static users (20% of users whose interests shift the least) and dynamic users (20% of users whose interests shift most). For both groups, GraphSAIL_adapt outperforms fine-tune and GraphSAIL w/o adaptive weights. We evaluate models by training on block t and testing on block t-1 in (b). GraphSAIL_adapt outperforms GraphSAIL for the static user group, while GraphSAIL performs better for the dynamic user group. This indicates that GraphSAIL_adapt preserves more information for users with persistent preferences and forgets more for users with the most dynamic preferences. Therefore, different levels of distillation help to improve the performance of the student model.

the baseline methods for LWC-KD, SGCT and GraphSAIL improve their average and median rank when the adaptive weights are incorporated. Our methods are routinely in the top three algorithms for all datasets and often achieve the best performance (see last column of Table 2). Besides the relative rank, in terms of absolute performance gain, the adaptive weights provide double digit percent performance increase over fine tuning across a variety of datasets. In particular, the strongest performance of adaptive weights is observed in traditional incremental learning datasets such as Gowalla. In datasets such as Netflix most users have a very high number of interactions (more than 100). As a result, the base portion of the dataset provides a good reflection of each user's set of interests. Users are therefore less likely to exhibit drastic changes of interest.

Comparison with full-batch training With Gowalla and Taobao2014, we have trained using all previous blocks and block t, and tested on block t+1 for each incremental block t (i.e. full batch). The average recall@20 is 0.1963 for Gowalla and 0.0191 for Taobao14. Though obtaining better performance in Taobao2014, the full batch method takes three times more time to train compared to LWC-KD-PIW. Therefore, in a live deployment setting for a client-facing recommender system, our system would be able to provide daily updates, whereas full retraining would quickly become computationally infeasible as new data accumulated.

Case Study

We have conducted a case study in order to more closely examine how three models behave for two distinct groups of users. The three models we study are fine-tune, GraphSAIL without adaptive weights and GraphSAIL with adaptive weights. We identify two types of users: (i) *static* users who exhibit minimal interest shift; and *dynamic* users who exhibit dramatic interest shift. Then we test the models on the historical data (i.e., data from previous time block). This evaluation on old data provides us with insight into how much historical information the models preserve for each group of users. We also check how well each group of users performs by testing on the next time block. Therefore, we can assess how preserving different amounts of information for each user group affects the model performance on the task of interest.

We observe that GraphSAIL with adaptive weights performs best for both user groups (Figure 4 (a)). From evaluation on historical data (Figure 4 (b)), we see that GraphSAIL is less affected than fine-tune, indicating that it counters the forgetting problem. GraphSAIL with adaptive weights outperforms GraphSAIL without adaptive weights for static users, while GraphSAIL without adaptive weights performs better with dynamic users. This implies that GraphSAIL with adaptive weights preserves more information for static users and less for dynamic users. Therefore, we conclude that adaptively distilling knowledge can help to improve modelling of future user preferences.

Conclusion

In this paper, we have proposed a novel method for incremental learning in graph-based recommender systems. Our approach hinges on learning adaptive personalization weights to tune the amount of knowledge distilled for each user's preferences during incremental learning. Our proposed method is evaluated on multiple datasets with three different incremental learning backbones and it consistently outperforms standard non-adaptive techniques. Our case study further supports our claim that the use of adaptive weights allows the model to distill more information for users with constant interests and to retain less information for users that are expressing rapid change in interests. This allows the model to adapt more quickly to changes of preferences for users with evolving interests.

References

Ahrabian, K.; Xu, Y.; Zhang, Y.; Wu, J.; Wang, Y.; and Coates, M. 2021. Structure Aware Experience Replay for Incremental Learning in Graph-based Recommender Systems. In *Proc. ACM Int. Conf. Info. & Knowledge Management (CIKM)*, 2832–2836.

Bo, D.; Wang, X.; Shi, C.; Zhu, M.; Lu, E.; and Cui, P. 2020. Structural Deep Clustering Network. In *Proc. World Wide Web Conf. (WWW)*, 1400–1410.

Castro, F. M.; Marín-Jiménez, M. J.; Guil, N.; Schmid, C.; and Alahari, K. 2018. End-to-End Incremental Learning. In *Proc. European Conf. Computer Vision (ECCV)*, 241–257.

Cheng, H.; Koc, L.; Harmsen, J.; Shaked, T.; Chandra, T.; et al. 2016. Wide & Deep Learning for Recommender Systems. In *Proc. ACM Recommender Syst. Conf. - Workshop on Deep Learning for Recommender Syst.*, 7–10.

Chiang, W.-L.; Liu, X.; Si, S.; Li, Y.; Bengio, S.; and Hsieh, C.-J. 2019. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proc. ACM SIGKDD Int. Conf. Knowledge Discovery & Data Mining*, 257–266.

Covington, P.; Adams, J.; and Sargin, E. 2016. Deep neural networks for youtube recommendations. In *Proc. ACM Conf. Recommender Syst.*, 191–198.

Fan, Z.; Liu, Z.; Zhang, J.; Xiong, Y.; Zheng, L.; and Yu, P. S. 2021. Continuous-time sequential recommendation with temporal graph collaborative transformer. In *Proc. ACM Int. Conf. Info. & Knowledge Management (CIKM)*, 433–442.

Guo, H.; Tang, R.; Ye, Y.; Li, Z.; and He, X. 2017. Deepfm: a factorization-machine based neural network for ctr prediction. In *Proc. Int. Joint. Conf. Artificial Intelligence (IJCAI)*, 1725–1731.

He, X.; Deng, K.; Wang, X.; Li, Y.; Zhang, Y.; and Wang, M. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. *Proc. ACM Int. Conf. Research and Development in Info. Retrieval*, 639–648.

Hidasi, B.; and Karatzoglou, A. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *Proc. ACM Int. Conf. Info. & Knowledge Management (CIKM)*, 843–852.

Hinton, G. E.; Vinyals, O.; and Dean, J. 2015. Distilling the Knowledge in a Neural Network. *ArXiv*, abs/1503.02531.

Kang, W.-C.; and McAuley, J. 2018. Self-attentive sequential recommendation. In *Proc. IEEE Int. Conf. Data Mining (ICDM)*, 197–206.

Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A. A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; et al. 2017. Overcoming catastrophic forgetting in neural networks. *National Academy of Sciences*, 114(13): 3521–3526.

Kullback, S.; and Leibler, R. A. 1951. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22: 79 – 86.

Mallya, A.; and Lazebnik, S. 2018. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Conf. Computer Vision and Pattern Recognition (CVPR)*, 7765–7773.

Prabhu, A.; Torr, P. H.; and Dokania, P. K. 2020. GDumb: A simple approach that questions our progress in continual learning. In *European Conf. Computer Vision (ECCV)*, 524–540.

Qiu, R.; Yin, H.; Huang, Z.; and Tong, C. 2020. GAG: Global Attributed Graph Neural Network for Streaming Session-based Recommendation. In *Int. ACM SIGIR Conf. Research and Development in Info. Retrieval*, 669–678.

Rebuffi, S.-A.; Kolesnikov, A.; Sperl, G.; and Lampert, C. H. 2017. iCarl: Incremental classifier and representation learning. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2001–2010.

Rendle, S.; Freudenthaler, C.; Gantner, Z.; and Schmidt-Thieme, L. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proc. Conf. Uncertainty in Artificial Intell. (UAI)*, 452–461.

Shmelkov, K.; Schmid, C.; and Alahari, K. 2017. Incremental Learning of Object Detectors Without Catastrophic Forgetting. In *Proc. Int. Conf. Computer Vision (ICCV)*, 3400–3409.

Sun, J.; Zhang, Y.; Ma, C.; Coates, M.; Guo, H.; Tang, R.; and He, X. 2019. Multi-Graph Convolution Collaborative Filtering. In *Proc. IEEE Int. Conf. Data Mining (ICDM)*, 1306–1311.

van den Berg, R.; Kipf, T. N.; and Welling, M. 2018. Graph Convolutional Matrix Completion. In *ACM SIGKDD Int. Conf. Knowledge Discovery & Data Mining - Deep Learning Day Workshop*.

Van der Maaten, L.; and Hinton, G. 2008. Visualizing data using t-SNE. *J. Machine Learning Research*, 9: 2579–2605.

Wang, C.; Pan, S.; Hu, R.; Long, G.; Jiang, J.; and Zhang, C. 2019a. Attributed Graph Clustering: A Deep Attentional Embedding Approach. In *Proc. Int. Joint. Conf. Artificial Intelligence (IJCAI)*, 3670–3676.

Wang, S.; Hu, L.; Wang, Y.; He, X.; Sheng, Q. Z.; Orgun, M. A.; Cao, L.; Ricci, F.; and Yu, P. S. 2021. Graph learning based recommender systems: A review. In *Proc. Int. Joint. Conf. Artificial Intelligence (IJCAI)*, 4644–4652.

Wang, X.; He, X.; Wang, M.; Feng, F.; and Chua, T.-S. 2019b. Neural Graph Collaborative Filtering. In *Proc. ACM Int. Conf. Research and Development in Info. Retrieval*, 165–174.

Wang, Y.; Zhang, Y.; and Coates, M. 2021. Graph Structure Aware Contrastive Knowledge Distillation for Incremental Learning in Recommender Systems. In *Proc. ACM Int. Conf. Info. & Knowledge Management*, 3518–3522.

Xu, J.; and Zhu, Z. 2018. Reinforced continual learning. In *Proc. Adv. Neural Info. Syst. (NeurIPS)*, 907–916.

Xu, Y.; Zhang, Y.; Guo, W.; Guo, H.; Tang, R.; and Coates, M. 2020. GraphSAIL: Graph Structure Aware Incremental Learning for Recommender Systems. In *Proc. ACM Int. Conf. Info. & Knowledge Management*, 2861–2868.

Yang, Y.; Qiu, J.; Song, M.; Tao, D.; and Wang, X. 2020. Distilling Knowledge from Graph Convolutional Networks. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 7074–7083.

Yang, Y.; Zhou, D.-W.; Zhan, D.-C.; Xiong, H.; and Jiang, Y. 2019. Adaptive Deep Models for Incremental Learning: Considering Capacity Scalability and Sustainability. In *Proc. ACM Conf. Knowledge Discovery & Data Mining*, 74–82.

Ying, R.; He, R.; Chen, K.; Eksombatchai, P.; Hamilton, W. L.; and Leskovec, J. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *Proc. ACM Conf. Knowledge Discovery & Data Mining*, 974–983.

Zeng, H.; Zhou, H.; Srivastava, A.; Kannan, R.; and Prasanna, V. 2020. GraphSAINT: Graph Sampling Based Inductive Learning Method. In *Proc. Int. Conf. Learning Representations (ICLR)*.

Zheng, H.; Yang, Z.; Liu, W.; Liang, J.; and Li, Y. 2015. Improving deep neural networks using softplus units. *Int. Joint Conf. Neural Networks (IJCNN)*, 1–4.

Zou, D.; Hu, Z.; Wang, Y.; Jiang, S.; Sun, Y.; and Gu, Q. 2019. Layer-Dependent Importance Sampling for Training Deep and Large Graph Convolutional Networks. In *Proc. Adv. Neural Info. Syst. (NeurIPS)*, 11247–11256.