HyperKDMA: Distilling Recommender Systems via Hypernetwork-based Teacher Assistants

ABSTRACT

Knowledge distillation (KD) is a model compression technique that transfers the knowledge of a well-trained large model (teacher) to a smaller one (student), so that we can obtain a compact model with good performance. Recent works have employed KD for recommender systems and shown that distilling the knowledge of the teacher's predictions or hidden layers considerably enhances the student performance. In this study, however, we demonstrate that this direct teacher-to-student distilling scheme results in bad performance of the student when the sizes of these two models significantly differ. To resolve this issue, we propose HyperKDMA, a KD scheme using multiple hypernetwork-based teacher assistants, i.e., intermediate-sized models between the teacher and the student, to bridge the teacher-student gap. Moreover, we adopt a gate network that automatically coordinates the influence of each TA on the student. Our extensive experiments on five widely used recommendation models and two public data sets show significant performance improvements of HyperKDMA over the state-of-the-art KD methods.

KEYWORDS

knowledge distillation, recommender systemm, teacher assistants

ACM Reference Format:

. 2023. HyperKDMA: Distilling Recommender Systems via Hypernetwork-based Teacher Assistants. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 10 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

In order to capture complex user-item representations and relationships, the scale of recommender systems has been continuously increasing to provide more accurate personalized recommendations. For example, deep learning-based recommenders [3, 23] offer excellent capability to model user-item relationships and have been extensively used in many systems, thanks to their strong performances. These models, however, require substantial computational resources and massive storage, leading to high latency at the inference phase. Therefore, the deployment of deep models in personal or embedded devices with limited computational capacity and memory is pretty challenging.

To alleviate this problem, several efficient training and inference methods [7, 10] have been adopted to allow the recommender systems to serve large-scale use cases. Among these, knowledge distillation (or KD) [4], which is a technique of learning by knowledge transfer from a large model (teacher) to a smaller one (student), has been recently adopted for achieving small but powerful recommenders [6–8, 11, 19] and shown promising improvement.

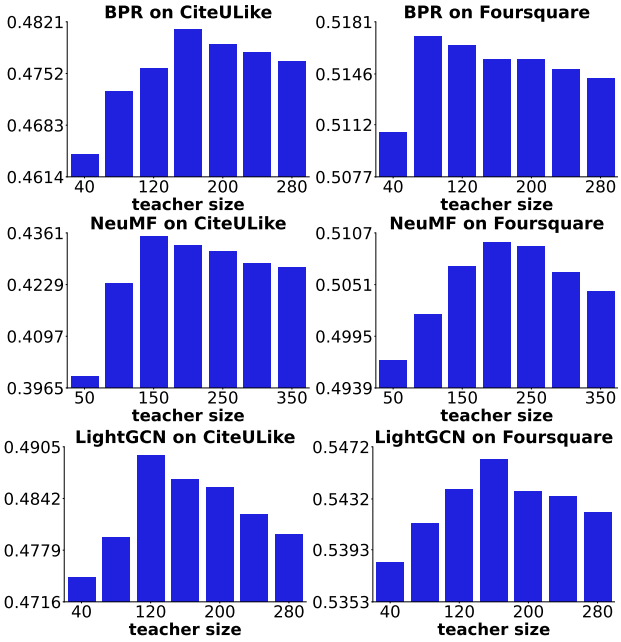


Figure 1: Student $H@5$ with increasing teacher sizes.

Teacher-Student Learning Gap. In KD, one usually assumes that an over-parameterized teacher with higher capability would lead to a better performing student. However, our experiments [1] demonstrate that a larger teacher (more powerful) does not always lead to a better performance of a small student. As illustrated in Figure 1, when we fix the student size and gradually increase the teacher size, the student performance first rises and then peaks at a certain teacher size before dropping. This is because when the teacher becomes more complex, the same student model does not have sufficient capacity to reconstruct the teacher model's hidden representations or output.

Approach. To resolve this issue of a direct teacher-to-student KD scheme, we propose a KD framework for recommender systems using multiple teacher assistant (TA) models, i.e., intermediate-sized models between the teacher and the student. The underlying intuition is that the TAs, which are more capable compared to the student, can help bridge the teacher-student gap, allowing better knowledge transfer from the teacher. We also augment the flexibility of this TA-based KD scheme by utilizing the power of hypernetworks [1] to generate the TA models from the teacher's hidden features. This allows the knowledge transfer process from the teacher to the student to be feature-dependent.

[1] Details about models, data sets and experimental setup are provided in Section 5.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

Contributions. We focus on KD for top-K recommendation with implicit feedback. Our contributions are summarized as follows. *Firstly*, to our knowledge, this is the first framework that deploys hypernetwork and TA models to resolve the learning gap issue in KD for recommender systems. *Secondly*, we propose HyperKDMA, a multiple hypernetwork-based TA KD framework that allows the student to learn from the teacher and the TAs simultaneously, alleviating the shortcoming of the direct teacher-student KD scheme. *Finally*, we conduct extensive experiments on benchmarking data sets to demonstrate the effectiveness of HyperKDMA.

2 RELATED WORK

Knowledge Distillation (KD) is a strategy for enhancing the performance of a compact model (student) by transferring knowledge from a well-trained large model (teacher) [4]. The student trained with KD has comparable performance to that of the teacher, yet has lower inference latency due to its small size. The early work [4] performs KD by matching the softmax distribution of the teacher (aka soft labels) and that of the student, while a subsequent work [17] distills knowledge from intermediate layers. Most KD methods focus on computer vision, while there are just a few studies working on KD for recommender systems.

KD in Recommender Systems. The first approach is to have the student preserve the output of the teacher. The pioneering work Ranking Distillation (RD) [19] executes KD by maximizing the relevance probabilities of the user to items in the teacher's ranked list. The distillation loss for user u is defined as follows:

$$\mathcal{L}_{RD} = -\sum_{\pi_k \in \boldsymbol{\pi}} w_{\pi_k} \log P\left(rel = 1 \mid u, \pi_k\right), \tag{1}$$

where $\boldsymbol{\pi}$ is the teacher's ranked list of top-K unobserved items for user u, $P\left(rel = 1 \mid u, \pi_k\right)$ is the relevance probability of user u to item π_k predicted by the student, and w_{π_k} is the weight that controls the relative importance of the k^{th} item π_k in the top-K list.

RD has a limitation: it only uses top-K teacher-ranked items and ignores other unobserved items in $\mathcal{I}_u^-$, making the KD process have no negative feedback. A subsequent work Collaborative Distillation (CD) [11] overcomes this by introducing a probabilistic rank-aware sampling to sample unobserved items in a more general manner. It proposes a KD loss that makes the student learn from those items:

$$\begin{aligned}\mathcal{L}_{CD} = -\sum_{i \in S(\mathcal{I}_u^-)} & q_{ui} \log(P(r = 1 \mid u, i)) \\ & + (1 - q_{ui}) \log(1 - P(r = 1 \mid u, i)),\end{aligned} \tag{2}$$

where $S\left(\mathcal{I}_u^-\right)$ is an item set sampled from $\mathcal{I}_u^-$ based on the teacher ranked list and q_{ui} is the weight that reflects relative importance among sampled items.

Both RD and CD have an identical drawback: they adopt the point-wise approach to distill from the teacher's predictions, which only considers one item at a time, so it cannot accurately maintain the ranking orders in the teacher's predictions. To directly consider ranking orders among items, the work Relaxed Ranking Distillation (RRD) [6] performs KD by maximizing a permutation probability. For each user, RRD samples K interesting items and L uninteresting items from the teacher's recommendation list. Let $\boldsymbol{\pi}^u$ be the list of sampled items sorted by the original order in the teacher's ranked list. The relaxed permutation probability is formulated as:

$$p\left(\boldsymbol{\pi}_{1:K}^u \mid S\right) = \prod_{k=1}^{K} \frac{\exp S\left(u, \pi_k^u\right)}{\sum_{i=k}^{K} \exp S\left(u, \pi_i^u\right) + \sum_{j=K}^{K+L} \exp S\left(u, \pi_j^u\right)}, \tag{3}$$

where $S(u, i)$ is the ranking score of the user-item interaction (u, i) predicted by the student S. The RRD loss is defined by maximizing the likelihood $p\left(\boldsymbol{\pi}_{1:K}^u \mid S\right)$:

$$\mathcal{L}_{RRD} = -\log p\left(\pi_{1:K}^u \mid S\right), \tag{4}$$

The most recent work item-side ranking regularized distillation (IR) [7] argues that all previous works only distill the user-side ranking information (i.e., ranking orders among items), which can destroy the item-side ranking information (i.e., ranking orders among users) in the student. IR is designed based on the item-side ranking information and can be combined with any user-side KD method based on the teacher's predictions. For each item, IR samples P positive users and N negative users from the teacher-ranked list of users that have not interacted with that item. Then it maximizes the relaxed permutation probability in a similar way to RRD.

Let $\boldsymbol{\pi}^i$ be the list of sampled users sorted by the original order in the teacher's item-side ranking list. The relaxed permutation probability is formulated as:

$$p\left(\pi_{1:P}^i \mid S\right) = \prod_{k=1}^{P} \frac{\exp S\left(\pi_k^i, i\right)}{\sum_{j=k}^{P} \exp S\left(\pi_j^i, i\right) + \sum_{m=P}^{P+N} \exp S\left(\pi_m^i, i\right)}. \tag{5}$$

The second way of KD for recommender systems is to preserve the hidden knowledge from the teacher. Distillation Expert (DE) [6] is the first attempt to distill the teacher's hidden feature by utilizing expert networks to reconstruct the representation of an intermediate layer of the teacher from that of the corresponding intermediate layer of the student. DE utilizes multiple experts, which are small feed-forward networks, and a selection network to select the best expert for distillation.

The most recent work Personalized Hint Regression (PHR) [8] states that it is challenging to deploy DE to a new environment (i.e., data set and base model) since DE is dependent on several hyperparameters that need to be tuned for each data set and base model. PHR resolves this difficulty by adopting a personalization network that provides a personalized mapping function bridging the representation space of the teacher and that of the student. Details about DE and PHR will be presented in Section 3.

KD via Teacher Assistant. In computer vision, the study TAKD [14] introduce a multi-step KD scheme through the TAs to alleviate the large teacher-student gap problem. However, a main shortcoming of TAKD is that it only uses one TA at a step, depriving the student of the chance to get more various knowledge. A later work Densely Guided KD (DGKD) [18] designs stochastic learning with dropout to train the student from multiple TAs. However, the use of dropout is a random process and does not take the importance of each TA into account, posing a limitation to KD procedure. In our paper, we addresses these issues by utilizing a gate network to calculate the influence of every TA on the student, enabling it to effectively acquire knowledge from multiple sources.

Hypernetworks. The pioneering work on hypernetworks [1] raises a limitation in neural network that the same parameters are used for all data points, which degrades the model performance when data points have different characteristics. The hypernetwork addresses this issue by generating the parameters of the main network, thus personalizing the model for every example. In our paper, we utilize the hypernetwork to generate personalized TAs for entities in the recommender system. This ensures that each entity in the student is properly guided by its own created set of TAs.

3 BACKGROUND

Top-K Recommendation. Let $\mathcal{U}$ and $\mathcal{I}$ be the set of users and items, respectively. The implicit user-item interaction information is given as an matrix $\mathbf{R} \in \{0,1\}^{|\mathcal{U}|\times|\mathcal{I}|}$, where entries $r_{ui} = 1$ and $r_{ui} = 0$ indicate user u has and has not interacted with item i, respectively. From this data, a ranking model is created to score how well a user likes an item. Let $\mathcal{I}_u^-$ be the set of items that have not interacted with user u. For user u, the model ranks all items in $\mathcal{I}_u^-$ by their scores, then recommends a ranked list of top-K unobserved items to that user.

KD for Top-K Recommendation. In KD, the student is trained by jointly optimizing a base loss $\mathcal{L}_{Base}$ and a KD loss $\mathcal{L}_{KD}$:

$$\mathcal{L}_S = \mathcal{L}_{Base} + \lambda_{KD}\mathcal{L}_{KD}, \tag{6}$$

where λ_{KD} is a hyperparameter that controls the effect of $\mathcal{L}_{KD}$. $\mathcal{L}_{Base}$ depends on the recommender, e.g. pair-wise ranking loss in BPR [16] or binary cross-entropy in NeuMF [3]; meanwhile, $\mathcal{L}_{KD}$ depends on the KD method. In this paper, we use two KD methods for top-K recommendation, namely DE [6] and PHR [8].

Distillation Experts (DE). Let $\mathbf{h}_T(\cdot)$ and $\mathbf{h}_S(\cdot)$ be the mapping function to the teacher's and the student's representation space, respectively. Specifically, $\mathbf{h}_T(e)$ and $\mathbf{h}_S(e)$ return the teacher embedding and the student embedding of entity e (i.e., user and item), respectively. DE has M experts $E_i, i = 1,\dots,M$, and Q is a selection network. Firstly, the selection network generates an one-hot selection vector $\mathbf{q}^e = \left(q_1^e,\dots,q_M^e\right) = S\left(\mathbf{h}_T(e)\right)$, where an element $q_i^e = 1$ indicates that the i^{th} expert is selected for distillation. The experts are trained by minimizing the following loss:

$$\mathcal{L}_{DE}(e) = \left\| \mathbf{h}_T(e) - \sum_{i=1}^{M} q_i^e \cdot E_i\left(\mathbf{h}_S(e)\right) \right\|_2^2. \tag{7}$$

The final DE loss is computed by distilling the knowledge of users and items in a mini-batch B as follows:

$$\mathcal{L}_{DE} = \sum_{u\in\mathcal{U}_B} \mathcal{L}_{DE}(u) + \sum_{i\in\mathcal{I}_B} \mathcal{L}_{DE}(i), \tag{8}$$

where $\mathcal{U}_B$ and $\mathcal{I}_B$ are the set of users and items in B, respectively.

Personalized Hint Regression (PHR) adopts a personalized mapping function for each entity to reconstruct the teacher's representation from the student's representation. Firstly, PHR obtains enriched representation $\mathbf{v}^e$ from $\mathbf{h}_T(e)$ with neighborhood information of entities that have interacted with entities e as follows:

$$\mathbf{v}^u = \frac{1}{2}\left(\mathbf{h}_T(u) + \frac{1}{|\mathcal{I}_u^-|}\sum_{i\in\mathcal{I}_u^-}\mathbf{h}_T(i)\right), \text{ for } e \text{ is user } u, \tag{9}$$

where $\mathcal{I}_u^-$ is the set of items that have not interacted with user u;

$$\mathbf{v}^i = \frac{1}{2}\left(\mathbf{h}_T(i) + \frac{1}{|\mathcal{U}_i^-|}\sum_{u\in\mathcal{U}_i^-}\mathbf{h}_T(u)\right), \text{ for } e \text{ is item } i, \tag{10}$$

where $\mathcal{U}_i^-$ is the set of users that have not interacted with item i.

Next, the personalization network p generates $\Theta^e = p\left(\mathbf{v}^e\right)$, which is then utilized as the parameters of the mapping function f^e: $\widehat{\mathbf{h}_T(e)} = f^e\left(\mathbf{h}_S(e);\Theta^e\right)$. PHR aims to minize the following loss:

$$\mathcal{L}_{PHR}(e) = \left\|\mathbf{h}_T(e) - \widehat{\mathbf{h}_T(e)}\right\|_2^2. \tag{11}$$

The final PHR loss is computed in a batch B as follows:

$$\mathcal{L}_{PHR} = \sum_{u\in\mathcal{U}_B}\mathcal{L}_{PHR}(u) + \sum_{i\in\mathcal{I}_B}\mathcal{L}_{PHR}(i). \tag{12}$$

4 THE HYPERKDMA FRAMEWORK

We propose *Knowledge Distillation via Multiple Hypernet-based teacher Assistants* (HyperKDMA) for KD in top-K recommendation. Specifically, the intermediate-sized TAs are applied to the hidden knowledge-based KD (HKD) to fill in the teacher-student gap. For clarity, we use d_T, d_S and d_{TA_i} to denote the dimension of entity representations of the teacher, the student and the i^{th} TA, respectively. The student is trained by jointly distilling the knowledge of the teacher and m TAs, as illustrated in Figure 2.

A simple way to obtain the TAs is to have them pretrained by learning from the teacher. However, this presents a drawback that all entities would learn from the same set of TAs regardless of the fact that each entity may have distinct characteristics. Therefore, we design a hypernetwork $P : \mathbb{R}^{d_T} \to \mathbb{R}^{\sum_i d_T \times d_{TA_i}}$ to generate a personalized set of TAs for every entity. For entity e, the hypernetwork takes the teacher representation $\mathbf{h}_T(e)$ and computes:

$$\mathbf{p}^e = P\left(\mathbf{h}_T(e)\right). \tag{13}$$

Next, $\mathbf{p}^e$ is separated into m chunks based on TA sizes, which are reshaped to yield mapping matrices $\mathbf{\Theta}_i,\ i = 1,\dots,m$. The i^{th} TA's embedding is computed by matrix multiplication of $\mathbf{h}_T(e)$ and $\mathbf{\Theta}_i$:

$$\mathbf{h}_{TA_i}(e) = \mathbf{h}_T(e) \times \mathbf{\Theta}_i,\ i = 1,\dots,m. \tag{14}$$

The mapping matrices vary for each entity and provide personalized guidance, ensuring effective KD with various preferences.

For efficient learning, we adopt a gate network to compute the impact of each TA on the student. The gatenet output is a linear transformation of the teacher embedding $\mathbf{h}_T(e)$ with a softmax layer, producing a distribution over m TAs:

$$\mathbf{g}^e = \left(g_1^e,\dots,g_m^e\right) = G\left(\mathbf{h}_T(e)\right) = \text{softmax}\left(\mathbf{\Theta}_G \cdot \mathbf{h}_T(e)\right), \tag{15}$$

where $\mathbf{\Theta}_G \in \mathbb{R}^{m\times d_T}$ is the parameter matrix of the gatenet.

The total TA-based KD loss is a weighted sum of m KD losses:

$$\mathcal{L}_{KD}^{TA\to S}(e) = \sum_{i=1}^{m} g_i^e \mathcal{L}_{KD}^{TA_i\to S}(e), \tag{16}$$

The student learns from all trainers by minimizing the final loss:

$$\mathcal{L}_S = \mathcal{L}_{Base} + \lambda_T\mathcal{L}_{KD}^{T\to S} + \lambda_{TA}\sum_e\sum_{i=1}^{m} g_i^e \cdot \mathcal{L}_{KD}^{TA_i\to S}(e), \tag{17}$$

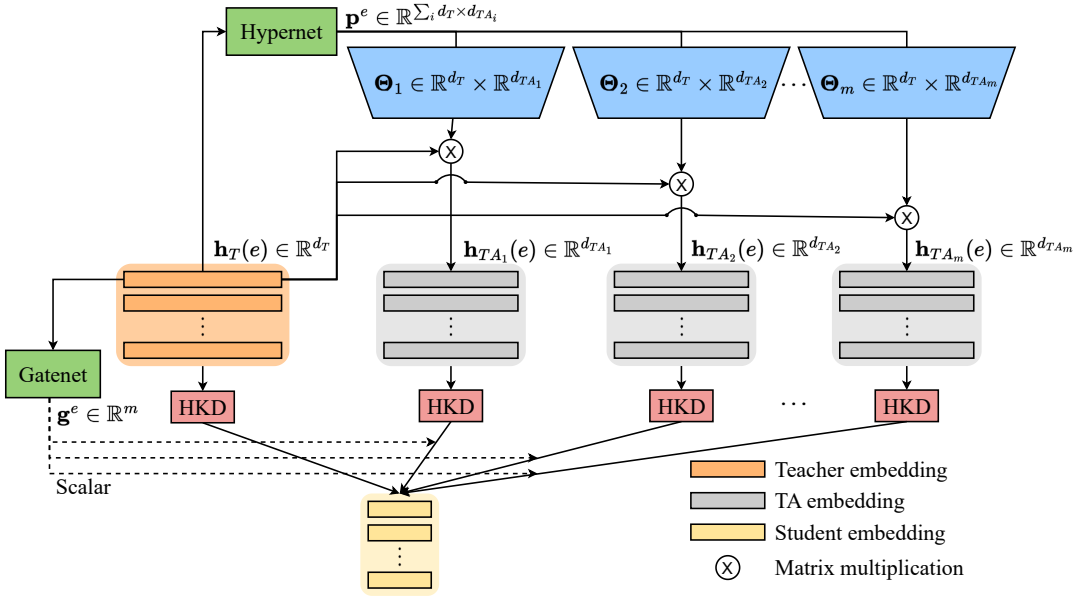


Figure 2: The HyperKDMA framework. HKD denotes hidden knowledge-based KD.

Algorithm 1: HyperKDMA

Data: training set $\mathcal{D}$, a well-trained teacher model $T(\cdot, \mathbf{\Theta}_T)$, learning rate η
Result: student model $S(\cdot, \mathbf{\Theta}_S)$
Initialize: student model $S(\cdot; \mathbf{\Theta}_S)$, hypernet $P(\cdot; \mathbf{\Theta}_P)$, gatenet $G(\cdot; \mathbf{\Theta}_G)$
while *not convergence* **do**
  for *each batch* $\mathcal{B} \in \mathcal{D}$ **do**
    for *each entity* $e \in \mathcal{B}$ **do**
      Initialize TA-based KD loss for a batch: $\mathcal{L}_{KD}^{TA \to S} \leftarrow 0$
      Compute $\mathbf{p}^e = P(e)$
      Separate $\mathbf{p}^e$ into m chunks and reshape them to get m mapping matrices $\mathbf{\Theta}_i, i = 1, \ldots, m$
      Compute $\mathbf{g}^e = G(e)$
      for $i = 1$ **to** m **do**
        $\mathbf{h}_{TA_i}(e) \leftarrow \mathbf{h}_T(e) \times \mathbf{\Theta}_i$
        Compute weighted TA-based KD loss for e: $g_i^e \cdot \mathcal{L}_{KD}^{TA_i \to S}(e)$
        Update $\mathcal{L}_{KD}^{TA \to S} \leftarrow \mathcal{L}_{KD}^{TA \to S} + g_i^e \cdot \mathcal{L}_{KD}^{TA_i \to S}(e)$
      end
    end
    Compute teacher-based KD loss for a batch: $\mathcal{L}_{KD}^{T \to S}$
    Compute base loss for a batch: $\mathcal{L}_{Base}$
    Compute total loss: $\mathcal{L} = \mathcal{L}_{Base} + \lambda_T \mathcal{L}_{KD}^{T \to S} + \lambda_{TA} \mathcal{L}_{KD}^{TA \to S}$
    Update student model: $\mathbf{\Theta}_S \leftarrow \mathbf{\Theta}_S - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{\Theta}_S}$
    Update hypernet: $\mathbf{\Theta}_P \leftarrow \mathbf{\Theta}_P - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{\Theta}_P}$
    Update gatenet: $\mathbf{\Theta}_G \leftarrow \mathbf{\Theta}_G - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{\Theta}_G}$
  end
end

where $\mathcal{L}_{KD}^{T \to S}$ and $\mathcal{L}_{KD}^{TA_i \to S}$ are the KD loss from the teacher and from the i^{th} TA to the student, respectively. The KD loss corresponds to the DE loss in Eq. (8) and the PHR loss in Eq. (12). Details about the procedure of HyperKDMA are provided in Algorithm 1.

It should be noted that other KD methods based on teacher's predictions are *orthogonal development*. They are not our competitors but can be combined with HyperKDMA to augment the knowledge we can transfer, thus further improving the student performance.

4.1 Training Cost Discussion

We discuss the training cost of KD methods. For simplicity, we omit the parameters of entity representation and the bias perceptrons.

For DE, the selection network has the shape $[d_T \to M]$ and each expert has the shape $[d_S \to (d_S + d_T)/2 \to d_T]$ [6], then the number of parameters is:

$$\mathcal{P}_{DE}(d_T, d_S) = M\left(d_T + \frac{(d_S + d_T)^2}{2}\right). \tag{18}$$

Meanwhile, for PHR, the personalization network has the shape $[d_T \to (d_T + d_T d_S)/2 \to d_T d_S]$, so the number of parameters is:

$$\mathcal{P}_{PHR}(d_T, d_S) = \frac{d_T^2 (1 + d_S)^2}{2}. \tag{19}$$

For HyperKDMA, the gatenet and the hypernet have the shape $[d_T \to m]$ and $\left[d_T \to \sum_{i=1}^m d_T d_{TA_i}\right]$, respectively. Thus, the number of parameters required is:

$$d_T\left(m + \sum_{i=1}^m d_T d_{TA_i}\right) + \mathcal{P}_{KD}(d_T, d_S) + \sum_{i=1}^m \mathcal{P}_{KD}(d_{TA_i}, d_S),$$

where $\mathcal{P}_{KD}$ is from Eq. (18) or Eq. (19) based on the KD framework.

HyperKDMA requires a fairly larger number of parameters compared to DE and PHR, and the training cost increases proportionally to the number of TAs. However, we would like to emphasize that the additional parameters are only required in the offline training phase, so there is no extra cost in the inference phase.

Table 1: Data set statistics.

Data set	#Users	#Items	#Interactions	Sparsity
CiteULike	5,219	25,181	115,142	99.91%
Foursquare	24,941	28,593	1,196,248	99.83%

5 EXPERIMENTS AND RESULTS

5.1 Experimental Settings

5.1.1 Data sets. We use CiteULike [20] and Foursquare [13]. We eliminate users with fewer than five ratings for CiteULike, and remove users and items with fewer than ten ratings for Foursquare. Details of the filtered data sets are reported Table 1.

5.1.2 Base models. We implement the following five base models:

- **Bayesian Personalized Ranking (BPR)** [16]: BPR utilizes matrix factorization (MF) to model user-item interactions and assumes that observed items should be ranked higher than unobserved ones.
- **Neural Matrix Factorization (NeuMF)** [3]: NeuMF is a deep model that combines MF and multi-layer perceptron (MLP) to capture non-linear user-item relationships. NeuMF treats the top-K recommendation as a binary classification task and is optimized with the binary cross-entropy loss.
- **Neural Graph Collaborative Filtering (NGCF)** [21]: NGCF is a graph-based recommendation method, which exploits high-order connectivity from user-item interactions by performing embedding propagation.
- **Light Graph Convolution Network (LightGCN)** [2]: LightGCN adopts the light graph convolution layers to capture multi-hop relationships in the user-item bipartite graph.
- **Self-supervised Graph Learning (SGL)** [22]: SGL is a state-of-the-art graph-based recommendation method for the top-K recommendation on implicit feedback. It adopts graph contrastive learning and performs three data augmentation methods: node dropout, edge dropout, and random walk.

5.1.3 Teacher/Student/TAs. We consider the size of a model as the dimension of its entity representation. We follow the recommended setting in [8]. For NeuMF, the teacher size and the student size are set to 250 and 25, respectively; while for other base models, these two sizes are set to 200 and 20.

For the TAs, we will consider the setting of TAs from an empirical perspective. One reasonable setting is to make the TA sizes be evenly spaced between the teacher size and the student size:

$$d_{TA_i} = d_S + i\frac{d_T - d_S}{m+1}, \quad i = 1, \ldots, m. \tag{20}$$

There are various other choices of d_{TA_i}, and in the main experiment of this paper, we provide results for $m = 8$ with the above empirically-set TA sizes. We also investigate the effect of the number of TAs in this paper. Other detailed results with different settings are included in our supplementary material.

5.1.4 Evaluation protocol. We adopt the *leave-one-out* evaluation protocol. For each user, we leave out two interacted items for validation and test. All the remaining items are used for training. To address the time-consuming problem of ranking all the items, we randomly sample 499 unobserved items for the user, and the model is evaluated by how well it ranks the validation/test item higher than these sampled items. We run the experiments on five train, validation, test seeds and report the average results.

5.1.5 Evaluation metrics. We use the following two metrics: 1) Hit Ratio $H@K$ [12], which measures whether the test item is in the top-K list; and 2) Normalized Discounted Cumulative Gain $N@K$ [5], which assigns higher scores to the hits at upper positions. These metrics are defined as follows:

$$H@K = \frac{1}{|\mathcal{U}_{test}|}\sum_{u\in\mathcal{U}_{test}} \delta\left(p_u \le \text{top}\, K\right), \tag{21}$$

$$N@K = \frac{1}{|\mathcal{U}_{test}|}\sum_{u\in\mathcal{U}_{test}} \frac{1}{\log\left(p_u+1\right)}, \tag{22}$$

where $\delta(\cdot)$ is the indicator function, $\mathcal{U}_{test}$ is the set of test users, and p_u is the hit ranking position of the test item for user u.

5.1.6 Implementation Details. We use Pytorch [15] for model implementation. For BPR, we build the model based on its paper [16], while for the other baselines, we follow the public implementations provided by the authors: NeuMF[2], NGCF[3], LightGCN[4], SGL[5]. We set the batch size for mini-batch training to 1024 and the maximum number of epochs to 1000. We adopt the early stopping strategy, which stops training if $H@5$ on the validation set is not improved for 20 consecutive epochs. For all base models, the number of negative samples is set to 1. The weights λ_T and λ_{TA} in the KD loss is chosen from $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$. We use Adam [9] for model optimization. The learning rate and weight decay are chosen from $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$. The DE and PHR framework has additional hyperparameters that need to be tuned. We follow the recommended settings in [6] and [8] for these hyperparameters.

5.2 Experimental Results

5.2.1 Performance comparison. Table 2 compares the performance of different methods in terms of $H@K$ and $N@K$ for $K \in \{5, 10\}$. TAKD, which is originally applied to preserve teacher's softmax output, does not help much in hidden knowledge-based KD. Meanwhile, DGKD improves the student to a larger extent, yet it still suffers from performance degradation due to its random dropout. In contrast, HyperKDMA significantly outperforms other KD methods thanks to the personalized learning mechanism. Figure 3 demonstrates the effectiveness of HyperKDMA when the teacher size increases, where our method allows the student to learn better from big teachers. We report the results on BPR, NeuMF and LightGCN while having similar observations in NGCF and SGL. It should be noted that when the teacher size is 40 for BPR/LightGCN and 50 for NeuMF, we do not adopt any TA, so DE/PHR and HyperKDMA are the same. When the teacher is much larger than the student, the performance gain is smaller, which is justifiable because the student size is fixed and the student gradually approaches its maximum achievable performance.

[2]https://github.com/hexiangnan/neural_collaborative_filtering
[3]https://github.com/huangtinglin/NGCF-PyTorch
[4]https://github.com/gusye1234/LightGCN-PyTorch
[5]https://github.com/wujcan/SGL-Torch

Table 2: Recommendation performance comparison. TAKD and DGKD denote the multi-step KD through TAs [14] and the Densely Guided KD [18], respectively. We conduct the paired t-test with significance level at 0.05. HyperKDMA achieves statistically significant improvements versus the best baseline.

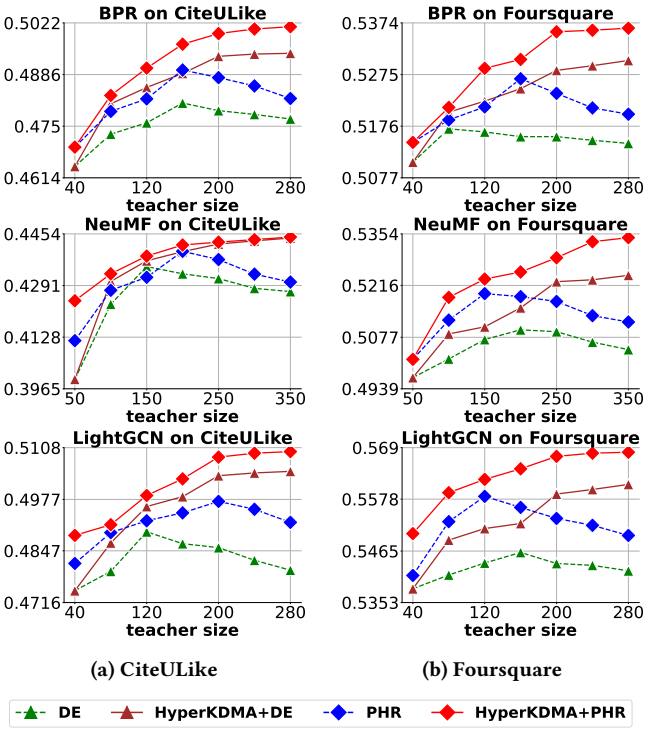
(a) Hit Ratio

Data set	Method	BPR		NeuMF		NGCF		LightGCN		SGL	
		H@5	H@10	H@5	H@10	H@5	H@10	H@5	H@10	H@5	H@10
CiteULike	Teacher	0.5273	0.6248	0.443	0.5426	0.5291	0.6288	0.5316	0.6313	0.5405	0.6424
	Student	0.4432	0.5517	0.3939	0.4976	0.4496	0.5584	0.4554	0.5622	0.4633	0.5697
	DE	0.4791	0.5834	0.4313	0.5298	0.4823	0.5898	0.4855	0.5902	0.4908	0.5983
	TAKD+DE	0.4812	0.5876	0.4335	0.5311	0.4842	0.5904	0.4879	0.5919	0.4931	0.6009
	DGKD+DE	0.4876	0.5913	0.4374	0.5377	0.4896	0.5939	0.4924	0.5986	0.4992	0.6058
	HyperKDMA+DE	**0.4934**	**0.5969**	**0.4422**	**0.5423**	**0.4964**	**0.6005**	**0.5037**	**0.6053**	**0.5086**	**0.6117**
	PHR	0.4878	0.5902	0.4369	0.5396	0.4908	0.5965	0.4972	0.601	0.5016	0.6079
	TAKD+PHR	0.4891	0.5928	0.4382	0.5411	0.4923	0.5989	0.4993	0.6029	0.5031	0.6094
	DGKD+PHR	0.4927	0.5988	0.4406	0.5482	0.4958	0.6041	0.5017	0.6085	0.5075	0.6148
	HyperKDMA+PHR	**0.4994**	**0.6059**	**0.4428**	**0.5441**	**0.5026**	**0.6111**	**0.5084**	**0.6141**	**0.5156**	**0.6193**
Foursquare	Teacher	0.5409	0.6929	0.5405	0.6784	0.5821	0.7234	0.5851	0.7274	0.5921	0.7346
	Student	0.5085	0.6581	0.4804	0.6341	0.5331	0.6814	0.5369	0.6831	0.5412	0.6897
	DE	0.5156	0.6671	0.5092	0.6578	0.5412	0.6873	0.5438	0.6902	0.5479	0.6952
	TAKD+DE	0.5171	0.6692	0.5111	0.6599	0.5443	0.6919	0.5473	0.6968	0.5506	0.6999
	DGKD+DE	0.5218	0.6735	0.5184	0.6643	0.5491	0.6986	0.5522	0.7031	0.5548	0.7067
	HyperKDMA+DE	**0.5283**	**0.6791**	**0.5226**	**0.6693**	**0.5567**	**0.707**	**0.5589**	**0.7092**	**0.5627**	**0.7135**
	PHR	0.5239	0.6717	0.5173	0.6645	0.5503	0.6962	0.5536	0.6991	0.5556	0.7034
	TAKD+PHR	0.5256	0.6745	0.5203	0.6668	0.5549	0.6993	0.5576	0.7021	0.5594	0.7061
	DGKD+PHR	0.5298	0.6811	0.5267	0.6705	0.5584	0.7057	0.5623	0.7088	0.5655	0.7115
	HyperKDMA+PHR	**0.5357**	**0.6872**	**0.529**	**0.6736**	**0.5637**	**0.7097**	**0.5671**	**0.7111**	**0.5688**	**0.7159**

(b) NDCG

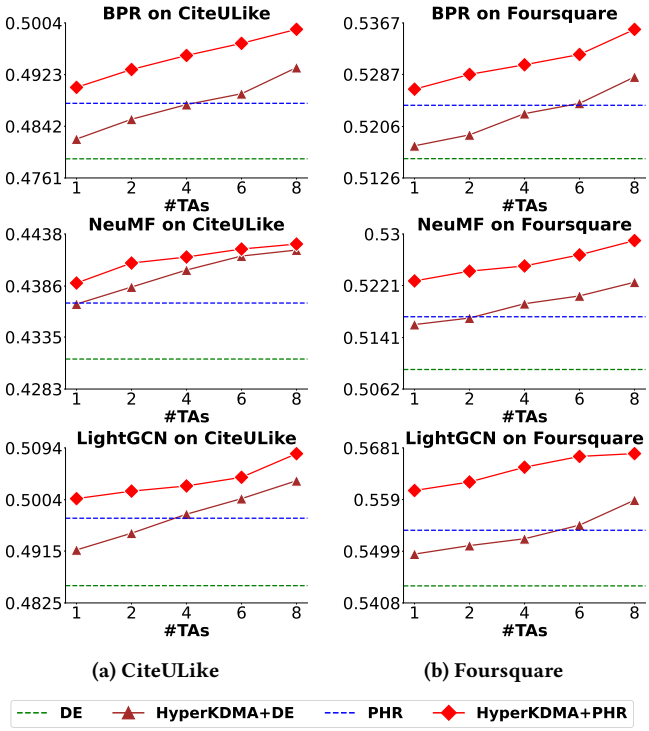
Data set	Method	BPR		NeuMF		NGCF		LightGCN		SGL	
		N@5	N@10	N@5	N@10	N@5	N@10	N@5	N@10	N@5	N@10
CiteULike	Teacher	0.4121	0.4419	0.3431	0.3745	0.4146	0.4478	0.4167	0.4498	0.4217	0.4563
	Student	0.3403	0.3765	0.296	0.3296	0.3441	0.3793	0.3483	0.3837	0.3512	0.3884
	DE	0.3637	0.3998	0.324	0.3524	0.3676	0.4018	0.3684	0.4028	0.3719	0.4091
	TAKD+DE	0.3677	0.4019	0.3266	0.3563	0.3693	0.4041	0.3702	0.4041	0.3734	0.4128
	DGKD+DE	0.3712	0.4041	0.3307	0.3624	0.3731	0.4073	0.3753	0.4085	0.3788	0.4175
	HyperKDMA+DE	**0.3772**	**0.4125**	**0.337**	**0.3687**	**0.3796**	**0.4155**	**0.3821**	**0.4154**	**0.3876**	**0.4243**
	PHR	0.3709	0.4081	0.3314	0.3599	0.3749	0.4098	0.3768	0.4118	0.3803	0.4172
	TAKD+PHR	0.3723	0.4102	0.3336	0.3624	0.3771	0.4119	0.379	0.4139	0.3832	0.4195
	DGKD+PHR	0.3776	0.4147	0.3378	0.3676	0.3826	0.4156	0.3829	0.4193	0.3878	0.4249
	HyperKDMA+PHR	**0.3812**	**0.4198**	**0.3417**	**0.3703**	**0.3867**	**0.4209**	**0.3873**	**0.4256**	**0.3924**	**0.4308**
Foursquare	Teacher	0.3953	0.4444	0.396	0.4405	0.4328	0.4754	0.4341	0.4796	0.4427	0.4836
	Student	0.369	0.4177	0.3396	0.392	0.3897	0.4118	0.3922	0.439	0.3988	0.442
	DE	0.3737	0.4228	0.3635	0.4109	0.3967	0.4427	0.3996	0.4459	0.4075	0.4517
	TAKD+DE	0.3768	0.4254	0.3675	0.4148	0.3994	0.4458	0.4019	0.4496	0.4093	0.4538
	DGKD+DE	0.3803	0.4297	0.3714	0.418	0.405	0.4497	0.4067	0.4535	0.4136	0.4577
	HyperKDMA+DE	**0.3878**	**0.4346**	**0.3788**	**0.4245**	**0.4103**	**0.4579**	**0.4129**	**0.4605**	**0.4193**	**0.4648**
	PHR	0.3794	0.429	0.3717	0.4193	0.4028	0.4507	0.4064	0.4524	0.4155	0.4602
	TAKD+PHR	0.3816	0.4314	0.3746	0.4218	0.4077	0.4543	0.4106	0.4567	0.4183	0.4644
	DGKD+PHR	0.3863	0.4357	0.3792	0.4251	0.4121	0.4596	0.4153	0.461	0.4249	0.4683
	HyperKDMA+PHR	**0.3917**	**0.4398**	**0.3818**	**0.4283**	**0.4188**	**0.4653**	**0.4216**	**0.4676**	**0.4305**	**0.4745**

5.2.2 The effect of the number of TAs. Figure 4 shows the KD performance on different numbers of TAs for BPR, NeuMF and LightGCN; results on NGCF and SGL follow similar trends. We obtain a better student compared to DE/PHR regardless of the number of TAs.



(a) CiteULike (b) Foursquare

Figure 3: Student $H@5$ with increasing teacher size.



(a) CiteULike (b) Foursquare

Figure 4: Student $H@5$ with different numbers of TAs.

Moreover, the student becomes more powerful when the number of TAs increases. However, we argue that applying eight TAs in our experiments is sufficient to figure out the tendency of the student performance, and raising the number of TAs even more will consume much training time and computing resources. According to these observations, we can choose the number of TAs based on the time and computing resources available.

5.2.3 Insights into the effect of HyperKDMA. We further provide quantitative and qualitative analyses on our method. Firstly, we visualize the representation space of the teacher and the students trained with different KD methods in Figure 5. We use BPR as the base model and perform K-means clustering of user embedding vectors in the teacher space, then use the clustering result for visualization. For each data set, we specify the number of clusters by the number of users divided by 500. All KD methods are capable of preserving the similarity information in the teacher space. However, the number of mis-clustered points in HyperKDMA is smaller than that of DE/PHR[6]. This means that HyperKDMA performs similarity preservation better than DE/PHR.

Secondly, we compute the standard deviation of the average performance ($H@5$ and $N@5$) in each cluster and show the results in Table 3. HyperKDMA shows lower standard deviations than other KD methods on all base models and data sets, therefore it can be considered as an ensemble method that distills more robust information, thus decreasing the variance of the student.

[6]Here, we define a misclassified point as the one located in a group that is different from the group in the teacher space.

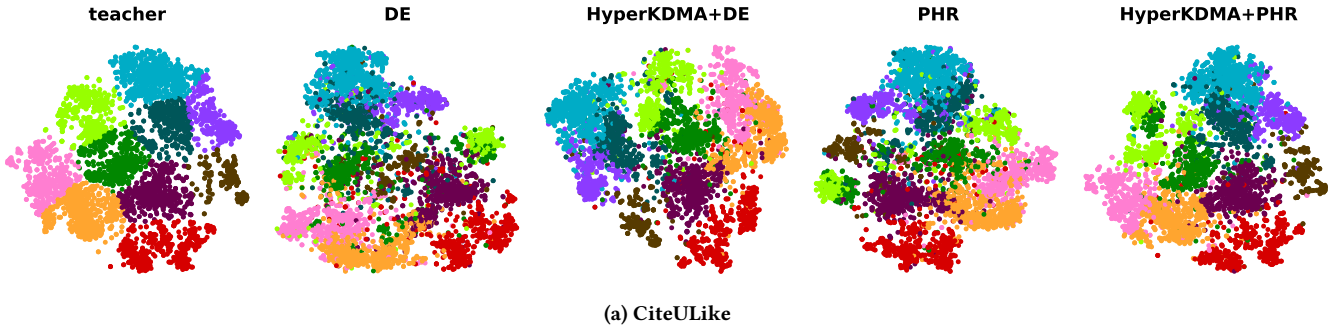
5.3 Ablation Study

To further ascertain the benefit of our proposed method, we provide an in-depth analysis of the ablation of HyperKDMA. The performance comparison results are summarized in Table 4. We consider two choices in which the hypernet or gatenet is removed and observe the effects on the student performance. The method without the hypernet or gatenet still outperforms the baseline method, yet it is inferior to the full version of HyperKDMA. This indicates the use of hypernet and gatenet is indeed effective for personalization. In addition, the model without the hypernet suffers more from performance deterioration compared to that without the gatenet, substantiating the pivotal role of the hypernet in generating personalized TAs.

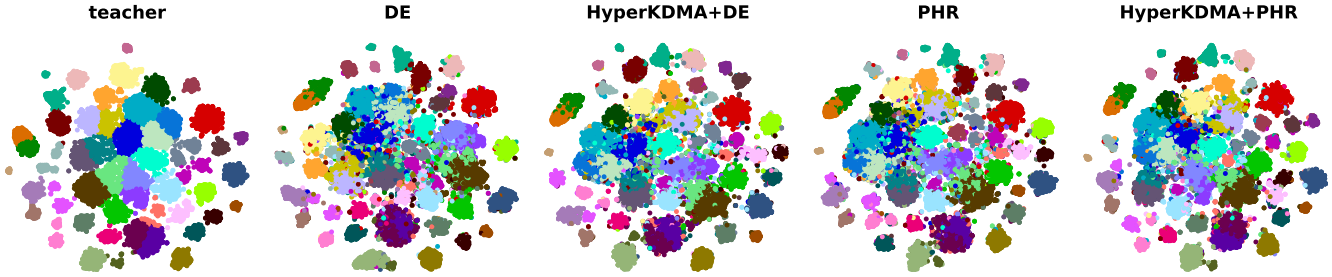
5.4 Hyperparameter Analysis

We perform hyperparameter study to provide guidance for hyperparameter selection of HyperKDMA. We show the effects of two hyperparameters: λ_T and λ_{TA} that control the influence of the teacher and the generated TAs on the student, respectively. The results are summarized in Figure 6. Since KD loss of HyperKDMA is composed of the KD loss from the pretrained teacher model and the KD losses from the generated TAs, it is important to properly balance the total loss by using λ_T and λ_{TA}. We report the results of HyperKDMA-PHR on Foursquare data set while observing similar tendencies on HyperKDMA-DE and CiteULike data set.

For all base models, the best performances are observed when the value of λ_T is around $10^{-3} - 10^{-2}$ and that of λ_{TA} is around $10^{-4} - 10^{-3}$. The optimal values of λ_T is larger than those of λ_{TA},



(a) CiteULike



(b) Foursquare

Figure 5: 2-D tSNE visualization of user embeddings.

Table 3: Standard deviation of average performance in representation clusters.

Data set	Method	BPR		NeuMF		NGCF		LightGCN		SGL	
		H@5	N@5	H@5	N@5	H@5	N@5	H@5	N@5	H@5	N@5
CiteULike	DE	0.0426	0.0355	0.0442	0.0368	0.0404	0.034	0.0378	0.0335	0.0338	0.0318
	HyperKDMA+DE	**0.0377**	**0.0329**	**0.0391**	**0.0336**	**0.0348**	**0.0316**	**0.0321**	**0.0306**	**0.0309**	**0.0285**
	PHR	0.0397	0.0356	0.0409	0.034	0.0374	0.0331	0.0353	0.0322	0.0324	0.0305
	HyperKDMA+PHR	**0.0348**	**0.032**	**0.0366**	**0.0317**	**0.0322**	**0.0309**	**0.0316**	**0.0289**	**0.0287**	**0.0264**
Foursquare	DE	0.1407	0.1208	0.1549	0.1231	0.1367	0.1134	0.1336	0.109	0.1303	0.1059
	HyperKDMA+DE	**0.136**	**0.1166**	**0.1461**	**0.1206**	**0.1312**	**0.1084**	**0.1289**	**0.1033**	**0.1267**	**0.1014**
	PHR	0.1391	0.1195	0.1481	0.1217	0.1359	0.1106	0.1325	0.1077	0.1283	0.1034
	HyperKDMA+PHR	**0.1327**	**0.1062**	**0.1425**	**0.1148**	**0.1293**	**0.1046**	**0.1267**	**0.1012**	**0.1223**	**0.0971**

Table 4: Performance comparison for alternative design choices of HyperKDMA.

Data set	Method	BPR		NeuMF		NGCF		LightGCN		SGL	
		H@5	N@5	H@5	N@5	H@5	N@5	H@5	N@5	H@5	N@5
CiteULike	HyperKDMA+DE	0.4934	0.3772	0.4422	0.337	0.4964	0.3796	0.5037	0.3821	0.5086	0.3876
	w/o HNet	0.4896	0.3733	0.4409	0.3337	0.4916	0.3743	0.4993	0.3783	0.5053	0.3803
	w/o GNet	0.4909	0.374	0.4416	0.3341	0.4939	0.3754	0.5026	0.3802	0.5086	0.3839
	HyperKDMA+PHR	0.4994	0.3812	0.4428	0.3417	0.5026	0.3867	0.5084	0.3873	0.5156	0.3924
	w/o HNet	0.4954	0.3761	0.441	0.3367	0.4973	0.3808	0.5023	0.3812	0.5093	0.3861
	w/o GNet	0.4976	0.3783	0.4423	0.338	0.4992	0.3825	0.5051	0.3841	0.5127	0.3897
Foursquare	HyperKDMA+DE	0.5283	0.3878	0.5226	0.3788	0.5567	0.4103	0.5589	0.4129	0.5627	0.4193
	w/o HNet	0.5226	0.3804	0.517	0.3724	0.5513	0.4044	0.5539	0.4065	0.5571	0.4132
	w/o GNet	0.5253	0.3829	0.5194	0.3733	0.5529	0.4072	0.555	0.4081	0.5596	0.4167
	HyperKDMA+PHR	0.5357	0.3917	0.529	0.3818	0.5637	0.4188	0.5671	0.4216	0.5688	0.4305
	w/o HNet	0.5317	0.3854	0.5221	0.3774	0.5569	0.4082	0.5602	0.4124	0.5631	0.4275
	w/o GNet	0.5323	0.3871	0.5241	0.3796	0.5602	0.4103	0.5633	0.4155	0.5657	0.4296

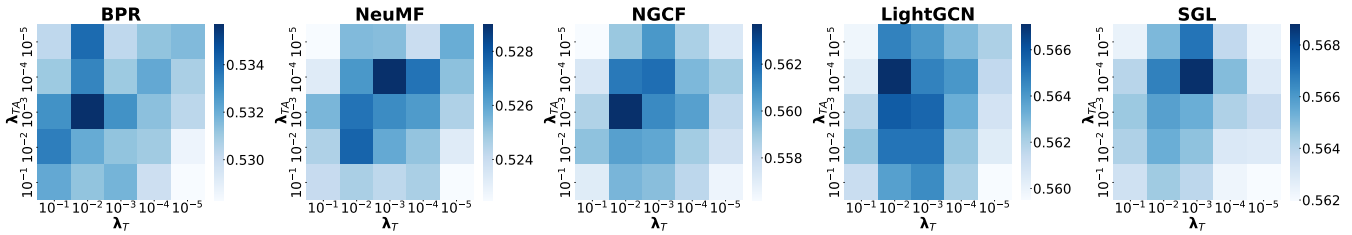


Figure 6: Effects of the hyperparameters ($H@5$).

Table 5: Results on MovieLens-20M data set. We conduct the paired t-test with significance level at 0.05. HyperKDMA achieves statistically significant improvements versus the best baseline.

Method	BPR		NeuMF		NGCF		LightGCN		SGL	
	H@5	N@5	H@5	N@5	H@5	N@5	H@5	N@5	H@5	N@5
Teacher	0.5847	0.4438	0.5856	0.4451	0.6236	0.4827	0.6269	0.4856	0.6327	0.4914
Student	0.5265	0.3846	0.5291	0.3873	0.5667	0.4431	0.5697	0.4472	0.5728	0.4506
DE	0.5357	0.3953	0.5318	0.3926	0.5779	0.4528	0.5788	0.4556	0.5813	0.4598
TAKD+DE	0.5386	0.3979	0.5334	0.3941	0.5802	0.4551	0.5813	0.4582	0.5836	0.4614
DGKD+DE	0.5403	0.4006	0.5369	0.3975	0.5838	0.4582	0.5844	0.4617	0.5874	0.4651
HyperKDMA+DE	**0.5477**	**0.4074**	**0.5443**	**0.3948**	**0.5911**	**0.4655**	**0.5928**	**0.4683**	**0.5949**	**0.4723**
PHR	0.5431	0.404	0.5412	0.4007	0.5856	0.4603	0.5867	0.4622	0.5897	0.4666
TAKD+PHR	0.5466	0.4062	0.5448	0.4029	0.5874	0.4627	0.5891	0.4645	0.5934	0.4692
DGKD+PHR	0.5497	0.4098	0.5495	0.4058	0.5914	0.4658	0.5935	0.4677	0.5968	0.4723
HyperKDMA+PHR	**0.5569**	**0.4164**	**0.5554**	**0.4126**	**0.5988**	**0.4725**	**0.6011**	**0.4747**	**0.6038**	**0.4798**

implying that the influence of the teacher on the student should be stronger than that of the TAs, which agrees with our intuition about the roles of the teacher and the TAs. When the values of these hyperparameters are too small (under 10^{-5}) or too large (over 10^{-1}), the KD performance is degraded.

5.5 Results on Large-scale Applications

Lastly, we compare the performance of the proposed method on a public large-scale data set: MovieLens 20M[7]. This data set has 138,493 users and 27,278 movies with 20,000,263 user–item interactions (sparsity: 99.47%). The size of the teacher and the student are set to 100 and 10, respectively. The results are reported in Table 5. Our proposed method consistently improves the student model and outperforms other KD methods on all base models. We believe that the proposed HyperKDMA can be successfully applied to many real-world applications in which one of the top priorities is low online inference latency.

6 CONCLUSIONS

We propose HyperKDMA, a multiple-TA distillation framework, to tackle the problem of the large teacher-student gap in KD for top-K recommendation. Our method provides entities in the recommender with their personalized TAs generated by a hypernetwork and controlled by a gate network. Our experimental results and analyses prove the effectiveness of HyperKDMA, making the student learn better from large teachers with high performances. Our framework can be applied to any latent knowledge-based KD method and combined with other prediction-based KD methods to further enhance the student performance.

[7] https://grouplens.org/datasets/movielens/

SUPPLEMENTARY MATERIAL

We provide our supplementary material (anonymously): here

REFERENCES

[1] David Ha, Andrew Dai, and Quoc V Le. 2016. Hypernetworks. *arXiv preprint arXiv:1609.09106* (2016).

[2] 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*. 639–648.

[3] 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*. 173–182.

[4] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[5] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)* 20, 4 (2002), 422–446.

[6] SeongKu Kang, Junyoung Hwang, Wonbin Kweon, and Hwanjo Yu. 2020. DE-RRD: A Knowledge Distillation Framework for Recommender System. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 605–614.

[7] SeongKu Kang, Junyoung Hwang, Wonbin Kweon, and Hwanjo Yu. 2021. Item-side ranking regularized distillation for recommender system. *Information Sciences* 580 (2021), 15–34.

[8] SeongKu Kang, Dongha Lee, Wonbin Kweon, and Hwanjo Yu. 2022. Personalized Knowledge Distillation for Recommender System. *Knowledge-Based Systems* 239 (2022), 107958.

[9] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[10] Dung D. Le and Hady W. Lauw. 2021. Efficient Retrieval of Matrix Factorization-Based Top-k Recommendations: A Survey of Recent Approaches. *J. Artif. Intell. Res.* 70 (2021), 1441–1479.

[11] Jae-woong Lee, Minjin Choi, Jongwuk Lee, and Hyunjung Shim. 2019. Collaborative distillation for top-N recommendation. In *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 369–378.

[12] Huayu Li, Richang Hong, Defu Lian, Zhiang Wu, Meng Wang, and Yong Ge. 2016. A Relaxed Ranking-Based Factor Model for Recommender System from Implicit Feedback.. In *IJCAI*. 1683–1689.

[13] Yiding Liu, Tuan-Anh Nguyen Pham, Gao Cong, and Quan Yuan. 2017. An experimental evaluation of point-of-interest recommendation in location-based social networks. *Proceedings of the VLDB Endowment* 10, 10 (2017), 1010–1021.

[14] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. 2020. Improved knowledge distillation via teacher assistant. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 5191–5198.

[15] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

[16] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618* (2012).

[17] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2014. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550* (2014).

[18] Wonchul Son, Jaemin Na, Junyong Choi, and Wonjun Hwang. 2021. Densely guided knowledge distillation using multiple teacher assistants. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 9395–9404.

[19] Jiaxi Tang and Ke Wang. 2018. Ranking distillation: Learning compact ranking models with high performance for recommender system. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2289–2298.

[20] Hao Wang, Binyi Chen, and Wu-Jun Li. 2013. Collaborative topic regression with social regularization for tag recommendation. In *Twenty-Third International Joint Conference on Artificial Intelligence*.

[21] 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*. 165–174.

[22] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised graph learning for recommendation. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*. 726–735.

[23] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep Learning Based Recommender System: A Survey and New Perspectives. *ACM Comput. Surv.* 52, 1, Article 5 (feb 2019), 38 pages. https://doi.org/10.1145/3285029