Dual-View Preference Learning for Adaptive Recommendation

Zhongzhou Liu, Yuan Fang *Member, IEEE* and Min Wu *Senior Member, IEEE*

Abstract—While recommendation systems have been widely deployed, most existing approaches only capture user preferences in the *macro-view*, i.e., the user's general interest across all kinds of items. However, in real-world scenarios, user preferences could vary with items of different natures, which we call the *micro-view*. Both views are crucial for fully personalized recommendation, where an underpinning macro-view governs a multitude of finer-grained preferences in the micro-view. To model the dual views, in this paper, we propose a novel model called Dual-View Adaptive Recommendation (DVAR). In DVAR, we formulate the micro-view based on item categories, and further integrate it with the macro-view. Moreover, DVAR is designed to be adaptive, which is capable of automatically adapting to the dual-view preferences in response to different input users and item categories. To the best of our knowledge, this is the first attempt to integrate user preferences in macro- and micro- views in an adaptive way, without relying on additional side information such as text reviews. Finally, we conducted extensive quantitative and qualitative evaluations on several real-world datasets. Empirical results not only show that DVAR can significantly outperform other state-of-the-art recommendation systems, but also demonstrate the benefit and interpretability of the dual views.

Index Terms—personalized recommendation systems, dual-view user preferences, adaptive models

1 Introduction

Personalized recommendation systems are playing an important role in a wide range of modern businesses, such as e-commerce [1], streaming services [2] and social media [3]. These businesses often offer thousands if not millions of items, causing a severe information overload—an average user is often interested in and able to consume only a handful of items within a short period of time. An effective recommendation system is thus expected to alleviate the information overload, not only enhancing the user experience but also increasing the conversion rate.

Due to their widespread applications, many recommendation techniques have been proposed, ranging from matrix factorization [4] and factorization machines [5], to more recent deep neural network-based methods [6], [7]. More recently, graph-based recommendation models [8], [9], [10] have been proposed and extensively studied, as the user-item interaction data for recommendation systems can be easily transformed into a graph structure, whereby graph learning methods can learn more complex relations than traditional recommendation models [11]. In particular, heterogeneous information networks (HIN) emerge as a representative form of graph for recommendation, and have been attracting increasing attention in the community [12], [13]. Compared to traditional graphs, HINs contain additional types of node (e.g., category) and edge (e.g., user-user similarity, item-item similarity, item-category association), beyond the standard user-item interaction. Thus, HINs can be regarded as a form of data augmentation capturing much richer semantic information, to overcome data sparsity and noises for more robust recommendations [14].

While existing recommendation approaches capture some form of user preferences in a *macro-view*, an obvious drawback is that they often fail to zoom into the *micro-view*. The macro-view describes a user's general preference across all kinds of items. For example, on a streaming platform featuring movies of different categories (e.g., action and comedy), a user likes Hollywood movies in general across both actions and comedies as shown in Fig. 1(a). In contrast, the micro-view describes and differentiates a user's preference for items of different nature. Beyond a general preference such as Hollywood movies, a user may prefer movies starred by Ford when coming to the action category, while prefers those starred by Pollak in the comedy category. However, a conventional macro-view-only system may only recommend the user popular Hollywood movies while ignoring his/her micro-view preference in cast for different categories.

In particular, the micro view enables finer-grained and differential user preferences in different item categories. For instance, some preferences are only important to certain categories (e.g., cast is important to actions but not documentaries). In other cases, the preference of the same user may disagree in different categories (e.g., while a user may enjoy Ford's action movies, he/she may not like Ford's comedy movies).

To materialize the micro-view w.r.t. different item categories, a naïve solution is to train one model for each category separately. However, this would not only lose some macro-view information, but also ignore the relationships between items in different categories [15], [16] that may mutually benefit each other. Alternatively, some approaches

- *Zhongzhou Liu and Yuan Fang are with the School of Computing and Information Systems, Singapore Management University, Singapore, 188065.*
  E-mail: zzliu.2020@phdcs.smu.edu.sg and yfang@smu.edu.sg
- *Min Wu is with the Institute for Infocomm Research, A*STAR, Singpaore, 138632.*
  Email: wumin@i2r.a-star.edu.sg

Corresponding authors: Yuan Fang and Min Wu.
Manuscript received xxx; revised xxx.

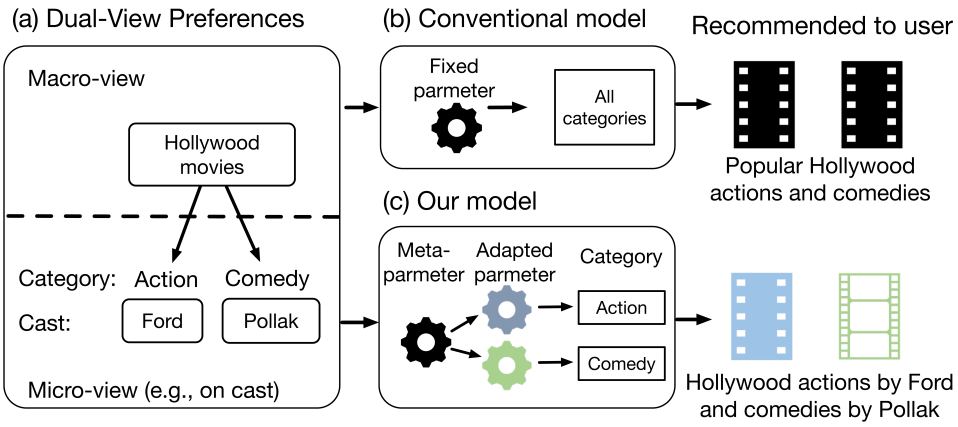


Fig. 1: A toy example on the macro- and micro-views, and the key differences between conventional and our models.

offer a limited micro-view, where the preference of a user toward an item can be drilled down to different aspects extracted from review data [17], [18], [19], or variable influences from other users who also purchased the same item [20]. As a result, for different items, the same user may prefer different aspects or become influenced by different users. However, they heavily rely on the quality of review data or co-purchase users, which may be missing or noisy in many applications, limiting their generalizability.

Toward personalized recommendation with both macro- and micro-views, we need to deal with two major challenges: (1) *How to formulate the micro-view and integrate it with the macro-view?* A micro-view to differentiate individual items can be overly specific and difficult to fit. Furthermore, the two views are fundamentally dependent—the macro-view describes the general preference of a user, which governs the micro-view to specialize into different preferences based on the nature of the item. Thus, a straightforward integration such as a linear combination of the two views, which disregards the dependence, would not perform well. (2) *How to design a model that is adaptive to the dual views?* In conventional approaches, the model is frozen once training is finished as shown in Fig. 1(b). In other words, the same model is applied to all users and items. Typically, users and items are only differentiated via their latent embeddings, but ideally the model itself should be adaptive for personalized preferences in the dual views.

To address the above two challenges, in this paper, we propose a novel model called Dual-View Adaptive Recommendation (DVAR) to learn user preferences in both macro- and micro-views. For the first challenge, we formulate the micro-view in terms of item *categories* (e.g., book, fashion, etc. for e-commerce, and genres for streaming services). Item categories are often more widely available and less noisy than review data or other side information, and strike a balance between the item-wise preferences that are complex and difficult to fit, and the overall macro-view preference that is too coarse. We further integrate the two views in a *dependence*-aware manner, where the micro-view preference is conditioned on both the item category and the general macro-view preference. For the second challenge, unlike the conventional recommendation systems, our model is equipped with a *dual-view adaptive module* as shown in Fig. 1(c), which is able to adaptively self-adjust to suit user preferences in the dual views. This can be achieved through a form of hypernetwork [21], [22], where the parameters of our prediction layer are generated by our adaptive module, a secondary neural network that adjusts a set of meta-parameters to adapt the prediction layer to the changing input (i.e., different users or items). Unlike previous adaptive models, we account for two levels of input conditioning, such that the macro-view is conditioned on the target user, and the micro-view is conditioned on both the macro-view and the item's categories. More concretely, we construct a HIN [14] consisting of users, items and categories, and further implement a base embedding model for the HIN, upon which the dual-view preferences and adaptive module are materialized. Compared to traditional models that only utilized user-item interactions, a HIN-based model can capture richer semantics and thus attain better recommendation performance [12], [13].

In summary, this work makes the following contributions.

- To the best of our knowledge, this is the first attempt to integrate user preferences in macro- and micro-views without relying on review data or other side information.
- We propose DVAR, a novel model that integrate the two views in a dependence-aware manner and is adaptive to the input users and item categories.
- We conduct extensive experiments on four real-world datasets. Both quantitative and qualitative results demonstrate that our dual-view formulation is beneficial to personalized recommendation and achieves state-of-the-art performance.

2 Related Work

In this section, we introduce and discuss related work in three categories: traditional recommendation, micro-view aware recommendation, and adaptive learning.

2.1 Traditional Recommendation Models

Recent years have witnessed numerous recommendation systems following the success of collaborative filtering techniques. Their applications span not only traditional scenarios like e-commerce [23] and social media [24], but also novel domains such as Internet of things [25], [26]. Classical techniques for collaborative filtering, such as matrix factorization [4], factorization machines [5] and Bayesian models [27], aim to learn user preferences based on historical interactions. These methods can also be combined with artificial neural networks to increase the model capacity [6], [7]. Different learning paradigms such as reinforcement learning [28] and active learning [29] also play an important role in the community of preference learning. Besides historical interactions, side information often exists on users or items such as social connections [3] or item categories [30], which has been integrated into a heterogeneous information network (HIN) to exploit its inherent structural and semantic properties [12], [31]. Sometimes, recommendation systems can also act as an auxiliary model to be incorporated into other tasks (e.g., personalized image tag prediction [32]) to improve the personalization of the main model. However, these methods only learn a general preference for each user in the macro view, without adapting to finer-grained micro-view preferences for different categories of items.

2.2 Micro-View Aware Recommendation

Several recent studies have attempted a limited form of the micro-view, which leverage item aspects mined from review texts [17], [33] or influences from users who have interacted with the same item [20]. However, their performance heavily depends on the availability and quality of review or co-interaction user data which can be sparse and noisy especially for less popular items. Item category information has also been used [30], [34], [35], but they merely combine category features as additional side information (e.g., simply adding category features to complement original user/item features [35]) and do not directly tailor user preferences w.r.t. item categories. The name "micro" also coincidentally appears in some other works [36], [37], [38]. However, they refer to the different granularities of interactions such as clicking pictures or chatting with customer service. Hence, their research objective and model design are fundamentally different from our work. Besides, all these methods train a fixed model for all users and items, and thus are not truly adaptive to changing user or item input.

2.3 Adaptive Learning

Our work achieves adaptive dual-view learning through hypernetworks [21], which belong to the paradigm of meta-learning. Different from conventional supervised learning, meta-learning is adaptive in nature by learning a prior that can be easily generalized to similar task. In recommendation, model-agnostic meta-learning (MAML) [39] has been applied to quickly adapt to new users and items in cold-start scenarios [40], [41], [42]. On graph data, hypernetworks such as Feature-wise Linear Modulations (FiLM) [22] have been used in graph neural networks in order to adapt to different nodes and edges [43], [44] or new graphs [45]. These methods do not address the adaptation based on our dual-view preferences. Additionally, MAML-based approaches require additional fine-tuning data for each item or user during testing, which are not needed in our approach.

3 Preliminaries

In this section, we formalize the input data and the recommendation problem, and introduce related concepts. The main notations used are shown in Table 1.

3.1 Input Data

A typical recommendation system works with a set of users $\mathcal{U}$ and items $\mathcal{I}$. Our model also assumes a set of categories $\mathcal{C}$, such that each item is associated with one or more categories. The users, items and categories form a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \mathcal{U} \cup \mathcal{I} \cup \mathcal{C}$ denotes the set of nodes, and $\mathcal{E}$ denotes the set of edges between nodes. The graph is an instance of *heterogeneous information networks* (HIN) [14], as shown in Fig. 2(a). Specifically, there are three types of nodes in $\mathcal{V}$, namely user, item and category, as well as four types edges in $\mathcal{E}$, namely user-item interaction, item-category association, user-user similarity and item-item similarity. Given a user $u \in \mathcal{U}$ and an item $i \in \mathcal{I}$, if there is a historical interaction between them (e.g., purchase in e-commerce, and click in streaming services), we have $(u, i) \in \mathcal{E}$. More succinctly, let $y_{u,i}$ denote the ground-truth interaction status between u and i such that $y_{u,i} = 1$ if $(u, i) \in \mathcal{E}$, and 0 if $(u, i) \notin \mathcal{E}$.

3.2 Recommendation Problem

To incorporate the micro view, our model depends on not only the target user and item, but also the categories of the item. Formally, our goal is to learn a prediction function

$$\hat{y}_{u,i} = \mathcal{F}(u, i, \mathcal{C}_i; \Theta), \tag{1}$$

such that $\hat{y}_{u,i}$ denotes the probability that user u will interact with item i, $\mathcal{C}_i = \{c \in \mathcal{C} | (i, c) \in \mathcal{E}\}$ denotes the set of categories of item i, and Θ denotes the model parameters. Given a set of user-item pairs $\mathcal{D}^{\text{tr}}$ as training data, the model can be learned by minimizing the cross entropy loss between the predictions $\{\hat{y}_{u,i}\}$ and the ground truths $\{y_{u,i}\}$, as follows.

$$\min_{\Theta} - \sum_{(u,i) \in \mathcal{D}^{\text{tr}}} y_{u,i} \log \hat{y}_{u,i} + (1 - y_{u,i}) \log(1 - \hat{y}_{u,i}). \tag{2}$$

TABLE 1: Summary of common notations used.

Symbol	Definition
$\mathcal{U}, u$	Set of users and a user instance
$\mathcal{I}, i$	Set of items and an item instance
$\mathcal{C}_i$	Set of categories for item i
$\mathcal{V}, \mathcal{E}$	Set of nodes and set of edges in a HIN
$\mathcal{M}, m$	Set of meta-paths and a specific meta-path
$p, \mathcal{P}_m$	A meta-path instance and a set of instances of meta-path m
$y_{u,i}, \hat{y}_{u,i}$	Ground truth and predicted interaction between u and i
$\mathbf{h}_m, \mathbf{h}_p$	Embeddings of meta-path m and instance p
$\Phi(\cdot), \phi(\cdot)$	Macro- and micro-view layers

3.3 Meta-paths

Meta-paths [46] have been widely used on HINs for their ability to preserve structural and semantic properties. Specifically, a meta-path is a sequence of node/edge types that captures a particular relationship between the start and end node types. For example, given users (U), items (I) and categories (C) in our graph, U–I–C–I is a meta-path describing that the start user has interacted with an item in the same category as the end item. To model the relationship between users and items, we only consider meta-paths starting from U and ending with I.

Subsequently, between a specific user and item such as u_1 and i_4 in Fig. 2(a), the path (u_1, i_3, c_2, i_4) is an *instance* of the meta-path U–I–C–I, which acts as a bridge between u_1 and i_4 and offers an interpretation on how they are related. In general, given any meta-path m with length ℓ, we can sample a subset of its instances, denoted $\mathcal{P}_m$, by performing ℓ-step random walks on the graph.

4 Proposed Approach

In this section, we introduce our model DVAR. The overall framework is illustrated in Fig. 2, consisting of three main modules: (1) the *base embedding model*, which captures the structural and semantic properties of the input HIN; (2) the *dual-view adaptive module*, which is capable of adaptively self-adjusting w.r.t. changing users or items to suit user preferences in both macro- and micro-views; (3) the *prediction layer*, which predicts the probability of interaction between a user and an item using the adapted parameters. In the following, we elaborate each of the three modules.

4.1 Base Embedding Model

Our base embedding model attempts to learn a set of embeddings to represent the relationships between users and items based on the input HIN, as illustrated in Fig. 2(b).

As HIN representation learning is not a main contribution of this paper, we mainly follow a previous meta-path-based recommendation model [13]. Given a meta-path m, we can construct an embedding $\mathbf{h}_m \in \mathbb{R}^d$ to represent the relationship based on m using a mean pooling over the instances of m. That is, $\mathbf{h}_m = \textsc{Mean}(\{\mathbf{h}_p : p \in \mathcal{P}_m\})$, where $\mathbf{h}_p \in \mathbb{R}^d$ is the embedding of the meta-path instance p. Thus, the meta-path embedding $\mathbf{h}_m$ can represent the overall relationship pattern between users and items in a more robust manner than an individual path, which is subsequently leveraged to predict the probability of interactions between the users and items.

To learn $\mathbf{h}_p$ for an instance $p = (v_1, v_2, \ldots, v_\ell)$, we first construct a feature matrix $X_p = [\mathbf{x}_{v_1}, \mathbf{x}_{v_2}, \cdots, \mathbf{x}_{v_\ell}] \in \mathbb{R}^{d \times \ell}$, where $\mathbf{x}_{v_i} \in \mathbb{R}^d$ indicates the feature vector of node v_i in the instance p. We adopt a self-attention mechanism [13] on path p to encode the pairwise relevance between the nodes in p via an $\ell \times \ell$ similarity matrix

$$S_p = \textsc{Self-Attn}(X_p; \Theta_{\text{attn}}), \tag{3}$$

where Θ_{attn} contains the learnable parameters. The high-level steps of the self-attention are outlined as follows. A typical attention technique computes the compatibility between two sequences, namely, *keys* and *queries*. In self-attention, keys and queries are referring to the same sequence. In our scenario, both keys K_p and queries Q_p are derived from the feature matrix X_p of the same path p. Essentially, we are learning the pair-wise attention weights between different nodes in path p, in the form of a similarity matrix S_p, to see which nodes are more important to a target node in the same path. The attention weights are then used to aggregate the *values* to obtain the overall embedding of path p, where the values are also derived from the path feature matrix X_p.

Specifically, given a path p of length ℓ, the keys, values and queries are computed as follows.

$$K_p = W_K X_p, \tag{4}$$
$$Q_p = W_Q X_p, \tag{5}$$
$$V_p = W_V X_p, \tag{6}$$

where W_K, W_Q and $W_V \in \mathbb{R}^{d \times d}$ are trainable parameters. Subsequently, the attention weights, $S_p \in \mathbb{R}^{\ell \times \ell}$, are given by

$$S_p = \textsc{Softmax}\left(\frac{K_p^\top Q_p}{\sqrt{s}}\right), \tag{7}$$

where $\sqrt{s}$ is a scaling factor to avoid the negative effects of very small gradients when the input data are large. Following previous work [13], we set $s = 128$. The above process of computing the attention weights S_p is absorbed into the $\textsc{Self-Attn}(X_p; \Theta_{\text{attn}})$ function in Eq. (3). The learnable attention parameters are thus $\Theta_{\text{attn}} = \{W_K, W_Q\}$.

Based on the attention weights, $\mathbf{h}_p$ can be obtained by aggregating the values as follows.

$$\mathbf{h}_p = \textsc{Tanh}\left(W^{(\ell)} \textsc{Flatten}(V_p S_p) + \mathbf{b}^{(\ell)}\right), \tag{8}$$

where $\textsc{Tanh}(\cdot)$ is the activation function, $\textsc{Flatten}(\cdot)$ vectorizes a matrix into a vector by stacking its columns, $W^{(\ell)} \in \mathbb{R}^{d \times d\ell}$ is a weight matrix with d-rows and $(d \times \ell)$-columns, and $\mathbf{b}^{(\ell)} \in \mathbb{R}^d$ is a bias vector. Note that for each value of the path length ℓ, there is a corresponding $W^{(\ell)}$ and $\mathbf{b}^{(\ell)}$. In practice, meta-paths are often short (e.g., $\ell < 5$) to prune noises from distant nodes. The learnable aggregation parameters are thus $\Theta_{\text{aggr}} = \{W_V, W^{(\ell)}, b^{(\ell)}\}$.

Overall, $\Theta_{\text{base}} = (\Theta_{\text{attn}}, \Theta_{\text{aggr}})$ forms the collection of all learnable parameters in the base model.

4.2 Dual-View Adaptive Module

Our proposed model DVAR is adaptive to user preferences in the macro- and micro-views. That is, the prediction model can self-adjust its parameters to adapt to different users and items in the two views, unlike conventional models whose parameters are frozen after training.

4.2.1 Dual-view Formulation

On one hand, we formulate the macro view w.r.t. each user, to capture the general preferences of the target user across all items. On the other hand, we formulate the micro view of a user w.r.t. each item's categories. As motivated in Sect. 1, using categories achieves a balance between the coarse-grained macro-view preferences, and the difficult-to-fit item-wise preferences.

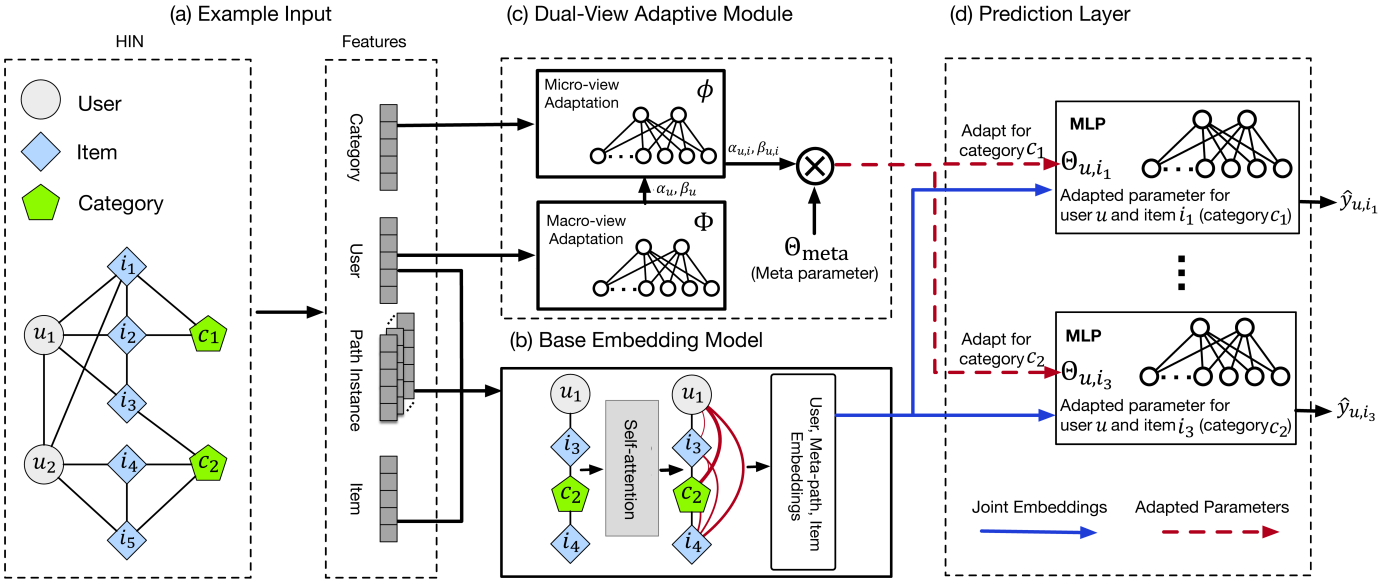


Fig. 2: Overall framework of DVAR.

Moreover, the two views are fundamentally dependent, where the macro view sketches the general preference of a user that governs the finer-grained micro view w.r.t. different categories. As shown in Fig. 2(c), $\Phi(\mathbf{x}_u)$ denotes the macro-view preference of user u, which is a function of u's input features $\mathbf{x}_u$. In contrast, the micro-view preference of u toward item i is denoted by $\phi(\Phi(\mathbf{x}_u), \mathbf{x}_{\mathcal{C}_i})$, which is a function of not only the macro-view preference of u, but also the feature vector of i's category set, $\mathbf{x}_{\mathcal{C}_i}$. Specifically, let $\mathbf{x}_{\mathcal{C}_i} = \textsc{Mean}(\{\mathbf{x}_c : c \in \mathcal{C}_i\})$ where $\mathbf{x}_c = \textsc{Mean}(\{\mathbf{x}_i : (i,c) \in \mathcal{E}\})$. That is, $\mathbf{x}_c$ is the feature vector of a single category c pooled over all items in c, whereas $\mathbf{x}_{\mathcal{C}_i}$ is further pooled over the categories of i as an item may be associated with multiple categories. With the help of mean-pooled category embedding, we are able to comprehensively describe the overall category information of an item, in order to facilitate the modeling of micro-view preference toward each item.

4.2.2 Dual Adaptive Layers

Conventionally, $\Phi(\cdot)$ or $\phi(\cdot)$ is realized as one model that can be applied to all users or item categories to generate the preferences. At the other extreme, we can realize $\Phi(\cdot)$ or $\phi(\cdot)$ as a collection of models, one for each user or category. The former tends to be less adaptive to the changing input, and the latter tends to suffer from overfitting and cannot easily extend to a new user or category.

To address the two problems, we realize the two views as a learnable function to generate a *transformation procedure* specific to the input user or item. The transformations are ultimately used to adapt a set of *meta-parameters* Θ_{meta} for the prediction w.r.t. any input user or item, while the meta-parameters encode the general pattern of the domain, a form of prior shared across all users and items. Given the ability to adapt the meta-parameters, we call $\Phi(\cdot)$ and $\phi(\cdot)$ the macro-view and micro-view adaptive layers, respectively. Each adaptive layer represents neither a single model nor a collection of models. Instead, it outputs different transformations to adjust the meta-parameters in response to the changing input, i.e., different input users or items. In other words, we do not learn any prediction parameter directly, but we learn (1) the meta-parameters Θ_{meta}, and (2) how to transform them into *adapted parameters* for predicting the interaction between the target user and item.

Formally, let $\Theta_{u,i}$ denote the adapted parameters used to predict the probability that user u will interact with item i. Specifically, the adapted parameter $\Theta_{u,i}$ is derived from the shared meta-parameters Θ_{meta} by the dual adaptive layers in a dependence-aware manner, i.e.,

$$\Theta_{u,i} = \mathcal{T}(\Theta_{\text{meta}}, \phi(\Phi(\mathbf{x}_u; \Theta_\Phi), \mathbf{x}_{\mathcal{C}_i}; \Theta_\phi)), \tag{9}$$

where $\mathcal{T}$ transforms Θ_{meta} in accordance with the transformation procedure generated by the dual adaptive layers, and Θ_Φ, Θ_ϕ are the parameters of the two layers, respectively. This is an instance of hypernetwork [21], as the adaptive layers can be deemed a secondary neural network that predicts the adapted parameters $\Theta_{u,i}$ for the primary network responsible for predicting the final recommendations. In the following, we elaborate each layer.

Macro-view Adaptive Layer. We employ scaling and shifting transformations in a form of hypernetwork known as **F**eature-wise **L**inear **M**odulation (FiLM) [22]. The general idea of FiLM is to generate scaling and shifting vectors conditioned on the input feature, which are used to transform the model parameters in order to adapt to different input. In our scenario, given user u, $\Phi(\mathbf{x}_u; \Theta_\Phi)$ outputs a transformation procedure given by

$$\alpha_u = A_\Phi \mathbf{x}_u, \tag{10}$$

$$\beta_u = B_\Phi \mathbf{x}_u, \tag{11}$$

where $\alpha_u, \beta_u \in \mathbb{R}^d$ respectively denote scaling and shifting vectors specific to user u, and $A_\Phi, B_\Phi \in \mathbb{R}^{d \times d}$ are the learnable parameters of the macro-view adaptive layer, i.e., $\Theta_\Phi = (A_\Phi, B_\Phi)$. Subsequently, α_u and β_u will be leveraged in the micro-view adaptive layer to transform the micro-view parameters. In other words, α_u and β_u, which are

generated from the general preference of u in the macro view, govern the micro-view preferences of u.

Micro-view Adaptive Layer. Given an user u and an item i with a category set $\mathcal{C}_i$, $\phi(\Phi(\mathbf{x}_u;\Theta_\Phi), \mathbf{x}_{\mathcal{C}_i};\Theta_\Phi)$ also outputs a transformation procedure given by

$$\alpha_{u,i} = (A_\phi \alpha_u + \beta_u) \odot \mathbf{x}_{\mathcal{C}_i}, \tag{12}$$

$$\beta_{u,i} = (B_\phi \alpha_u + \beta_u) \odot \mathbf{x}_{\mathcal{C}_i}, \tag{13}$$

where $\alpha_{u,i}, \beta_{u,i} \in \mathbb{R}^d$ respectively denote scaling and shifting vectors specific to user u and item i's categories, α_u and β_u are output from the macro-view layer $\Phi(\mathbf{x}_u;\Theta_\Phi)$, and $A_\phi, B_\phi \in \mathbb{R}^{d\times d}$ are the learnable parameters of the micro-view layer, i.e., $\Theta_\phi = (A_\phi, B_\phi)$. Here $\odot$ denotes Hadamard product. Note that α_u and β_u transform the micro-view parameters through scaling and shifting, respectively.

Finally, the output from the micro-view layer will be used to scale and shift the meta-parameters to generate the adapted parameters for the prediction layer in Sect. 4.3. For any user u and item i, the adapted parameters are

$$\Theta_{u,i} = \Theta_{\text{meta}} \odot \alpha_{u,i} + \beta_{u,i}. \tag{14}$$

Algorithm 1: Training of DVAR

Input: Input HIN $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, training data $\mathcal{D}^{\text{tr}}$, meta-paths $\mathcal{M}$;
Output: Optimized model parameters Θ;
Initialize DVAR parameters $\Theta = (\Theta_{\text{base}}, \Theta_{\text{meta}}, \Theta_\Phi, \Theta_\phi)$;
Initialize $\mathbf{x}_u$, $\mathbf{x}_i$ with pre-trained features and compute $\mathbf{x}_c$;
Compute $\mathbf{x}_{\mathcal{C}_i} = \textsc{Mean}(\{\mathbf{x}_c : c \in \mathcal{C}_i\})$;
while *not converged* **do**
  Compute $\mathbf{h}_m$ for each $m \in \mathcal{M}$;
  Compute the macro-view scaling and shifting vectors with Eqs. (10)–(11);
  Compute the micro-view scaling and shifting vectors with Eqs. (12)–(13);
  Compute dual adapted parameters $\Theta_{u,i}$ with Eq. (14);
  Predict $\hat{y}_{u,i}^{(m)}$ with Eq. (16) for each $(u,i) \in \mathcal{D}^{\text{tr}}$ and $m \in \mathcal{M}$;
  Compute the loss given by Eq. (17);
  Update trainable parameters Θ;
end
return *updated parameters* Θ^*.

4.3 Prediction Layer

Consider a set of user-item pairs D^{tr} as training data, and a set of meta-paths $\mathcal{M}$. For any training sample $(u,i) \in D^{\text{tr}}$, we predict a probability of interaction $\hat{y}_{u,i}^{(m)}$ based on each meta-path (i.e., relationship) $m \in \mathcal{M}$. Specifically, we model a triplet (u,i,m) based on the idea of translation [47]. The concept of translation was originally used in knowledge graphs to model relation triplets. A relation triplet (h,r,t) captures the fact that a head entity h is related to a tail entity t via the relationship r. The translation mechanism models the embedding of relation r as the translation from head h to tail t: $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$. In the context of recommendation, we regard a meta-path m that connects user u and item i as a relation [13], [48], which captures different user preferences in items as discussed in Sect. 3.3. To imitate the translation on the triplet (u,m,i), we define a joint embedding

$$\mathbf{h}_{u,i}^{(m)} = \mathbf{x}_u + \mathbf{h}_m - \mathbf{x}_i. \tag{15}$$

which captures the "transition error" of the triplet. During model training, the joint embedding of positive triplets would gradually converge to a certain distribution that is distinguishable from that of negative triplets. Thus, given the joint embedding, we can further predict the probability of interaction between user u and item i, based on only the relationship m, as

$$\hat{y}_{u,i}^{(m)} = \text{MLP}(\mathbf{h}_{u,i}^{(m)}; \Theta_{u,i}), \tag{16}$$

where MLP is a multi-layer perceptron (MLP) parameterized by $\Theta_{u,i}$. Note that the parameters $\Theta_{u,i}$ are not directly learnable; they are generated by the dual adaptive layers conditioned on user u and item i, as shown in Eq. (14).

4.3.1 Training

Overall, our proposed DVAR has a set of learnable parameters $\Theta = (\Theta_{\text{base}}, \Theta_{\text{meta}}, \Theta_\Phi, \Theta_\phi)$ that can be optimized based on the cross entropy loss. Given a set of user-item pairs as training data $\mathcal{D}^{\text{tr}}$, we minimize the following.

$$\min_{\Theta} - \sum_{(u,i)\in\mathcal{D}^{\text{tr}}} \sum_{m\in\mathcal{M}} y_{u,i} \log \hat{y}_{u,i}^{(m)} + (1-y_{u,i}) \log(1-\hat{y}_{u,i}^{(m)}) + \lambda(\|\alpha_{u,i} - \mathbf{1}\|_2^2 + \|\beta_{u,i}\|_2^2), \tag{17}$$

where the last term $\lambda(\|\alpha_{u,i}-\mathbf{1}\|_2^2+\|\beta_{u,i}\|_2^2)$ is a constraint on the transformations generated by the dual adaptive layers, to ensure that the adapted parameters would not be too far away from the prior meta-parameters. Without this constraint the model tends to overfit. Here $\lambda \geq 0$ is a scalar hyperparameter to balance the constraint, and $\mathbf{1}$ denotes a vector of all one's.

The pseudocode of the training process is outlined in Algorithm 1. it mainly involves (1) using the base embedding model to obtain embeddings for users, items, meta-paths and categories, (2) computing dual-view adaptations and the adapted parameters, and (3) computing and backpropagating the recommendation loss.

4.3.2 Testing

During testing, we predict $\hat{y}_{u,i}$, the overall probability that u will interact with i, by pooling the probability w.r.t. different meta-paths. Letting $\textsc{Pool}(\cdot)$ be a pooling function such as max or mean, we have

$$\hat{y}_{u,i} = \textsc{Pool}(\{\hat{y}_{u,i}^{(m)} : m \in \mathcal{M}\}). \tag{18}$$

5 Experiment

We conduct extensive experiments to evaluate and analyze our proposed model DVAR[1]. We further present a case study to provide some insights into the dual views.

[1]We release the code at https://github.com/mediumboat/DVAR

TABLE 2: Statistics of the datasets.

Dataset	#Users	#Items	#Interactions	#Categories	Density
MovieLens	943	1,682	100,000	19	106.0
Last.fm	1,889	10,150	83,763	100	44.3
Serendipity	104,661	49,151	9,997,850	19	95.5
Ali	416,538	256,857	852,346	100	2.0

5.1 Experiment Setup

5.1.1 Datasets

We employ four public datasets, namely, MovieLens [2] and Serendipity [3] for movies, Last.fm [4] for music, and Alibaba (Ali) [5] for e-commerce. Their statistics are shown in Table 2. Note that for Last.fm and Ali, we only retain the top 100 most frequent categories.

On the original datasets, we perform the following preprocessing steps. First, we construct a HIN for each dataset using users, items and categories. The categories of an item is also known as genres on the Movielens and Serendipity datasets, and tags on the Last.fm dataset. An item-category edge exists if the item is in that category, and an item may belong to multiple categories. We also construct user-user and item-item edges based on their similarity. Specifically, we apply Singular Value Decomposition (SVD) on the user-item interaction matrix to generate user and item embeddings. For each user/item, we form links to top 10 users/items ranked by their cosine similarity of the SVD-based embeddings. Second, on the constructed HIN, we obtain the embedding vector $\mathbf{x}_v$ for each node $v \in \mathcal{V}$ using the mp2vec algorithm [49]. For a user or item node, i.e., $v \in \mathcal{U}$ or $v \in \mathcal{I}$, its embedding $\mathbf{x}_v$ is used as the user/item feature vector shown in Fig. 2(a). For a category node, i.e., $v \in \mathcal{C}$, we compute its feature vector by pooling over the feature vectors of all items in the category. Finally, for each meta-path (the choice of meta-paths will be elaborated in Sect. 5.1.5), we sample its instances by performing n random walks starting from each user node as described in Sect. 3.3, and remove repeated path samples. In our case, we set $n = 10,000$. For each sampled path p, we construct a feature matrix X_p by stacking the feature vectors of the nodes in p, as described in Section 4.1.

5.1.2 Data Splitting

We split the users into training/validation/test sets with the ratio of 0.7/0.15/0.15. All user-item interactions of training users are added to $\mathcal{D}^{\text{tr}}$ as training data. For the validation/test users, we add 50% of their interactions to $\mathcal{D}^{\text{tr}}$ in order to learn their representations, whereas the remaining 50% are held out for validation/testing. During evaluation, for each test user u, we construct a ground truth list $\mathcal{D}_u^+$ by adding the held-out interacted items of u. The ground truth list (i.e., the positive examples) is further mixed with a sample of 20 non-interacting items of u (i.e., the negative examples) as the candidate list for recommendation to u.

[2]https://grouplens.org/datasets/movielens/100k/
[3]https://grouplens.org/datasets/serendipity-2018/
[4]https://grouplens.org/datasets/hetrec-2011/
[5]https://tianchi.aliyun.com/dataset/dataDetail?dataId=649

5.1.3 Evaluation Metrics

We choose three widely adopted evaluation metrics, namely, precision, recall and normalized discounted cumulative gain (NDCG), applied to top 10 items recommended to each test user u. Henceforth, we abbreviate them as prec@10, rec@10 and NDCG@10, respectively. Let us denote the ranked list of top 10 items recommended to user u as $\mathcal{D}_u^{\text{rl}}$. Then, these metrics on a given test user u are defined below.

$$\text{prec@10} = \frac{|\mathcal{D}_u^+ \cap \mathcal{D}_u^{\text{rl}}|}{10}, \tag{19}$$

$$\text{rec@10} = \frac{|\mathcal{D}_u^+ \cap \mathcal{D}_u^{\text{rl}}|}{|\mathcal{D}_u^+|}, \tag{20}$$

$$\text{NDCG@10} = \frac{\text{DCG@10}}{\text{iDCG@10}}. \tag{21}$$

Note that NDCG is defined in terms of DCG and iDCG. Specifically, $\text{DCG@10} = \sum_{k=1}^{10} \frac{\text{rel}_k}{\log(k+1)}$, where rel_k represents the relevance of $\mathcal{D}_u^{\text{rl}}(k)$, the k-th item in the ranked list, i.e., $\text{rel}_k = 1$ if $\mathcal{D}_u^{\text{rl}}(k) \in \mathcal{D}_u^+$, and $\text{rel}_k = 0$ otherwise. Furthermore, iDCG is the ideal DCG which is computed as the DCG of the ranked list $\mathcal{D}_u^{\text{rl}}$ where the items in the list are re-sorted by their relevance rel_k in descending order. After computing the metrics for each test user, we report the average prec@10, rec@10 and NDCG@10 over all test users.

5.1.4 Baselines

We consider the following methods as our baselines for comparison.

- **SVD**: The classical singular value decomposition technique for matrix factorization.
- **NeuCF** [7]: a neural collaborative filtering model for recommendation.
- **mp2vec** [49]: a meta-path-based skip-gram model for HIN embedding.
- **HERec** [12]: a HIN-based recommendation model, integrating matrix factorization with fused HIN embedding.
- **MTRec** [13]: a HIN-based recommendation model with a self-attention mechanism on meta-paths.
- **CMN** [20]: a collaborative memory network recommendation model with an attention mechanism to adjust the influences from users who interacted with the same item.
- **BiANE** [50]: a bipartite attributed network embedding model.

5.1.5 Parameters and Settings

We chose the hyper-parameters and settings based on the validation set and guidance from the literature. In particular, we set the embedding dimension for all methods to 128, which is also a common setting in many previous graph representation learning works [13], [49], [50]. For methods requiring meta-paths, we use meta-paths of length 3 or 4 and restrict the two end nodes to *user* and *item*, which include U-I-U-I, U-I-C-I, U-U-U-I, U-I-I-I, U-U-I and U-I-I (U: user, I: item, C: category). For HIN-based methods, the input HIN is the same as that for DVAR; for other methods, the input data are user-item interactions (i.e., a biparatite graph). For BiANE and DVAR, we use mp2vec to generate the initial feature vectors as the default setting. Note

TABLE 3: Comparison between the proposed DVAR and the baselines. The overall best result is bolded, and the best baseline result is underlined.

	MovieLens			Last.fm			Serendipity			Ali		
	Prec@10	Rec@10	NDCG@10	Prec@10	Rec@10	NDCG@10	Prec@10	Rec@10	NDCG@10	Prec@10	Rec@10	NDCG@10
SVD	.639±.011	.149±.008	.806±.004	.448±.013	.220±.001	.701±.010	.402±.014	.258±.015	.698±.019	.072±.038	.482±.213	**.456**±.042
NeuCF	.795±.010	.212±.026	.914±.013	.721±.020	.354±.007	.870±.004	.602±.034	.426±.032	.811±.028	.077±.014	.539±.104	.455±.012
mp2vec	.752±.013	.177±.013	.882±.008	.671±.014	.301±.022	.821±.017	.561±.042	.382±.039	.767±.023	.059±.032	.433±.176	.424±.018
HERec	.812±.017	.214±.021	.917±.015	.703±.021	.338±.009	.874±.015	.587±.021	.389±.017	.781±.019	.051±.013	.411±.119	.419±.022
MTRec	.773±.008	.189±.016	.912±.006	.719±.022	.345±.016	.873±.011	.591±.030	.406±.015	.813±.027	.057±.021	.445±.253	.411±.054
CMN	.771±.019	.152±.006	.834±.015	.599±.028	.246±.036	.740±.026	.545±.031	.361±.024	.721±.010	.041±.004	.522±.103	.390±.044
BiANE	.780±.012	.183±.016	.904±.009	.697±.013	.335±.010	.869±.014	.583±.016	.386±.017	.781±.008	.085±.027	.497±.104	.455±.011
DVAR	**.892**±.010	**.228**±.037	**.968**±.011	**.782**±.022	**.388**±.019	**.897**±.015	**.644**±.035	**.448**±.037	**.853**±.031	**.095**±.024	**.623**±.072	.451±.036

that other methods do not require external node features as input. For DVAR, the default pooling function in the prediction layer in Eq. (18) is max pooling, and the MLP therein has one hidden layer with 128 neurons. For SVD, mp2vec and BiANE, the prediction $\hat{y}_{u,i}$ is computed with cosine similarity on the trained embeddings. Other methods use their respectively proposed prediction function. The effect of other feature initialization methods and pooling functions will be examined in Sect. 5.4. For other standard parameters such as the learning rate and batch size, the main principle is to balance both validation performance and training efficiency. Particularly, for DVAR, we use the Adam optimizer, set the learning rate to 1e-3 and training batch size to 5,000. The constraint scalar λ in Eq. (17) is set to 1e-3 for Movielens and 1e-2 for other datasets, and we further analyze its impact on the model performance in Sect. 5.4.

We implement the proposed DVAR using Tensorflow 2.2 in Python 3.6. All experiments were conducted on a Linux workstation with a 6-core 3.6 GHz CPU, 128 GB DDR4 memory and two RTX 2080 Ti GPUs. For the baseline SVD, we use the SciPy implementations[6]. For mp2vec, we use the DGL port version[7]. For CMN, we use a PyTorch port[8] as reference and port it to TensorFlow. For NeuCF[9], HERec[10] and BiANE[11], we used their respective authors' implementations. For MTRec, we implement the recommendation component as described in its paper with TensorFlow.

5.2 Performance Comparison

Table 3 shows the performance of various models on all the four datasets. To show the robustness the results, for each dataset we run the experiment five times, and each time we split the train/validation/test sets according to the ratio described in Sect. 5.1.2. Finally, we report the corresponding mean and standard deviation for each metric over the five runs. From the reported results, we can make two main observations.

First, DVAR performs consistently well on all datasets, registering an average improvement of about 0.052, 0.039 and 0.027 over the *best baselines* in terms of the three metrics, respectively. There are two reasons why DVAR brings such improvements. On the one hand, DVAR integrates both macro- and micro-views for adaptive recommendation while other models only focus on the macro-view. As a result, compared to the baseline models, DVAR can learn the user's micro-view preference in a more adaptive way while still preserving the macro-view preference. Although there are some approaches also utilizing some form of micro-view preferences as described in Sect. 2.2, they are not compared here as they require additional side information such as text reviews. We will further evaluate the contribution of our dual-view adaptive module through an ablation study in Sect. 5.3. On the other hand, DVAR utilizes a HIN for recommendation. Previous studies have demonstrated that HIN is an effective means of incorporating the structural and semantic properties to improve the performance of a recommendation system [14]. It can also be observed that among the baselines, HERec and MTRec are HIN-based and generally achieve competitive results.

Second, on the Ali dataset, many deep or complex models fail to outperform shallow models (i.e., SVD and NeuCF) in NDCG. This outcome is influenced by two factors. For one, among the three evaluation metrics, only NDCG depends on the item ranking. Hence, NDCG is harder to optimize than the other two metrics, especially when the recommendation model employs a user-item prediction loss instead of a ranking loss [51]. Moreover, the Ali dataset is very sparse, which makes deep or complex models more likely to overfit [52], [53]. Considering the difficulty of NDCG and the sparsity issue, in terms of NDCG our approach cannot easily outperform shallower models like SVD and NeuCF, as well as models employing a pair-wise ranking loss like BiANE. Nevertheless, our design can still achieve a comparable NDCG@10 score, whilst outperforming in Prec@10 and Rec@10.

5.3 Ablation study

We perform model ablation to examine how our dual-view adaptive module contributes to the recommendation. First, we derive two variants with only the base embedding model and prediction layer, without the adaptive module. One of the variants adopts multiple prediction layers, one for each category (named *Multi*). The other uses a single prediction layer for all categories (named *Single*). Next, we combine the micro- and macro-view adaptive layers linearly instead

[6]https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html

[7]https://github.com/dmlc/dgl/tree/master/examples/pytorch/metapath2vec

[8]https://github.com/IamAdiSri/cmn4recosys

[9]https://github.com/hexiangnan/neural_collaborative_filtering

[10]https://github.com/librahu/HERec

[11]https://github.com/fukien/BiANE

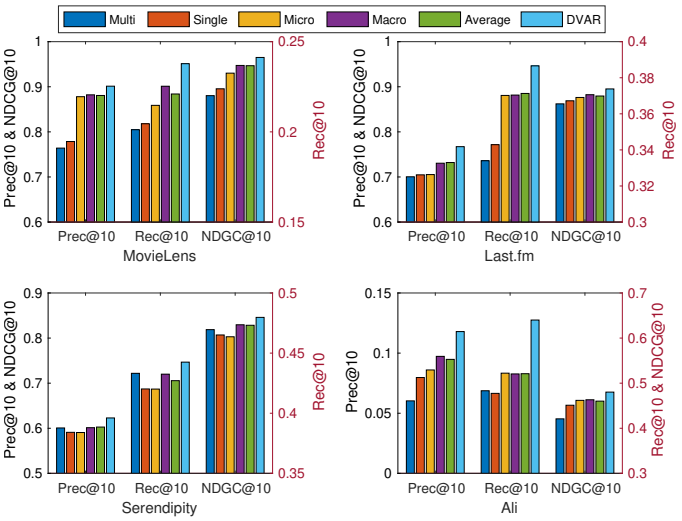


Fig. 3: Ablation study on the dual-view adaptive module.

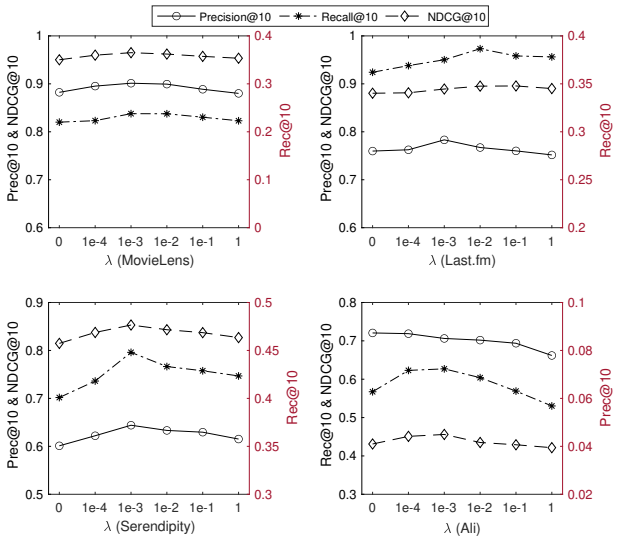


Fig. 4: Impact of λ.

of our dependence-aware manner. Specifically, we replace Eqs. (10)–(13) with the following formulae:

$$\alpha_{u,i} = q(A_\Phi \mathbf{x}_u) + (1-q)(A_\phi \mathbf{x}_{\mathcal{C}_i}), \quad (22)$$

$$\beta_{u,i} = q(B_\Phi \mathbf{x}_u) + (1-q)(B_\phi \mathbf{x}_{\mathcal{C}_i}), \quad (23)$$

where q is a scalar to balance the two views. We obtain three different variants when $q = 0, 0.5, 1$. In particular, $q = 0$ is equivalent to a micro-view only model (named *Micro*), $q = 1$ is equivalent to a macro-view only model (named *Macro*), and $q = 0.5$ is equivalent to a simple averaging of the two views (named *Averaging*). Note that other q values do not achieve better results.

The comparison between DVAR and its five variants is shown in Fig. 3. The absence of dual-view adaptive module in *Multi* and *Single* leads to huge performance decrease in most cases. Furthermore, *Multi* is generally worse than *Single* as it is hard for category-specific prediction layers to capture the whole picture of macro-view preferences. In contrast, by incorporating some adaptive capability, the three variants *Micro*, *Macro* and *Averaging* can achieve much better results. While the *Averaging* method combines both micro- and macro-views, it employs a linear combination that ignores the hierarchical relationship between the two views, limiting its performance to be similar to that of *Micro* or *Macro* only. Lastly, the full model DVAR outperforms all the variants, showing the importance of dual-view adaptive module with a dependence-aware integration.

5.4 Model analyses

We conduct further experiments to analyze our model performance from several aspects.

5.4.1 Impact of Parameters

We investigate the impact of the constraint scalar λ in Eq. (17). The performance of DVAR against λ is plotted in Fig. 4. All four datasets show consistent performance patterns w.r.t. varying λ values. In particular, DVAR is quite robust to different λ values in the range [1e-4, 1e-2]. Nevertheless, we should avoid extreme values which can still hurt the performance significantly. On one hand, $\lambda <$ 1e-4 is a bad choice as it is equivalent to removing the constraint on the adapted parameters, which can be too different from the prior meta-parameters. On the other hand, $\lambda >$ 1e-2 is not ideal either as it forces the adapted parameters to become too similar to the prior meta-parameters, effectively removing the adaptive module.

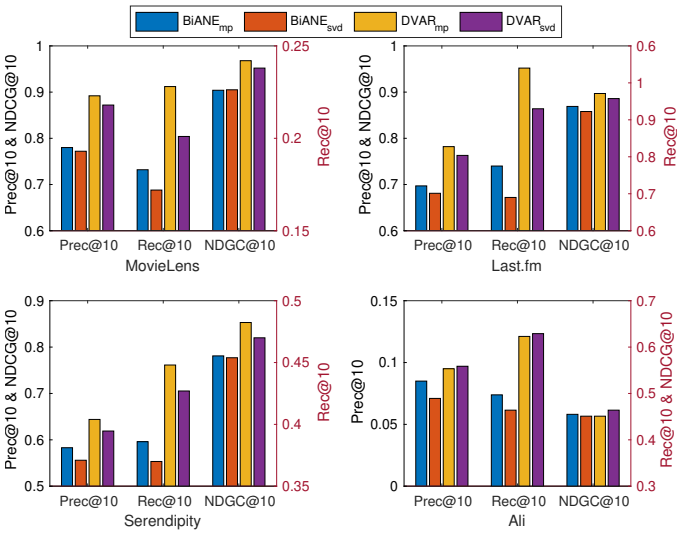


Fig. 5: Impact of feature initialization.

5.4.2 Feature Initialization

To show how different feature initializations can influence the recommendation performance, we compare two initialization methods. The first is mp2vec, which is also the default setting in our experiments. The other is SVD, which is a traditional matrix factorization method widely used in recommendation systems. We apply the two initialization methods to the baseline BiANE and our proposed DVAR. We label the variants of BiANE and DVAR with mp2vec-based initializations as BiANE$_{\text{mp}}$ and DVAR$_{\text{mp}}$, respectively, and those with SVD-based initializations as BiANE$_{\text{svd}}$ and DVAR$_{\text{svd}}$, respectively. The performance is reported in Fig. 5.

TABLE 4: Case study on user 251, with sample movies in training and testing data, and the respective ranking positions of the testing movies in the recommendation lists predicted by two models. The movies matching the user's micro-view preference (i.e., actions starred by Ford and comedies starred by Pollak) appear in bold.

Sample of Interacted Movies in Training Data			Sample of Interacted Movies in Testing Data				
Title	Category	Actor/Actress	Title	Category	Actor/Actress	Rank by DVAR	Rank by Macro
Empire Strikes Back	Action	Harrison Ford	**The Fugtive**	Action	Harrison Ford	2	5
Return of the Jedi	Action	Harrison Ford	**Indiana Jones 3**	Action	Harrison Ford	7	14
Star Wars	Action	Harrison Ford	**That Thing You Do!**	Comedy	Kevin Pollak	10	45
Dragonheart	Action	Harrison Ford	The Rock	Action	Sean Connery	3	2
Grumpier Old Men	Comedy	Kevin Pollak	Twister	Action	Helen Hunt	4	3
Mission: Impossible	Action	Tom Cruise	Men in Black	Action	Tommy L. Jones	8	8
Die Hard	Action	Bruce Willis	Willy Wonka	Comedy	Gene Wilder	5	4
Michael	Comedy	John Travolta	The Birdcage	Comedy	Robin Williams	14	17

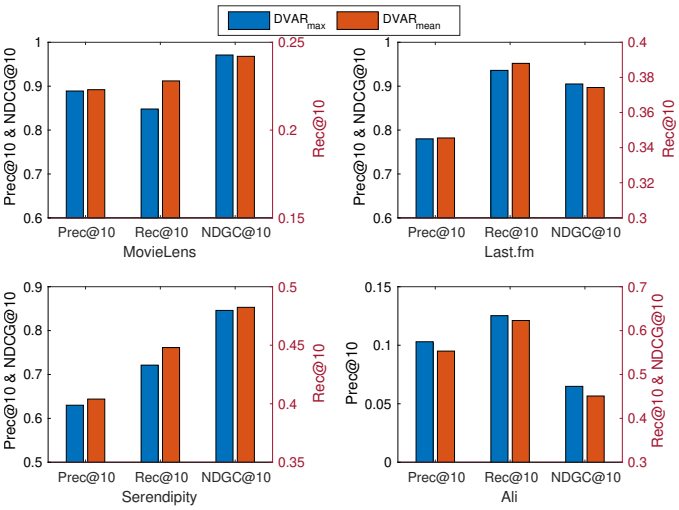


Fig. 6: Impact of pooling in the prediction layer.

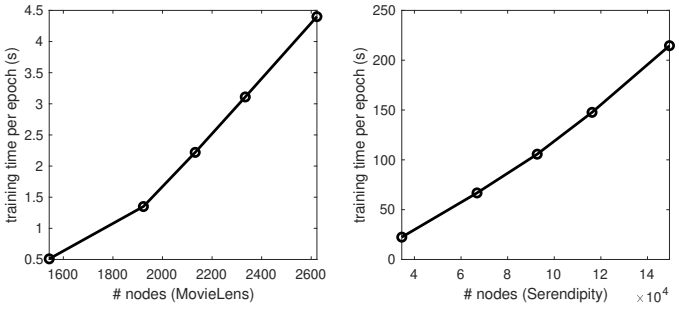


Fig. 7: Training time w.r.t. graph size.

As the results show, on both BiANE and DVAR, mp2vec-based initialization can generally outperform SVD-based initialization. The reason of such improvement is mp2vec can effectively model the rich semantics embodied in the input HIN by way of the meta-paths, whereas SVD only leverages the user-item bipartite graph. This experiment also shows that having a good feature initialization is important to the final model performance.

5.4.3 Prediction Layer

To examine the effect of the pooling function used in the prediction layer in Eq. (18), we compare the performance of DVAR with max and mean pooling. The corresponding variants are labeled DVAR_{max} and $\text{DVAR}_{\text{mean}}$, respectively. The results are shown in Fig. 6. From the results we observe that mean pooling usually fares better than max pooling, since the mean pooling can mitigate the impact of outliers in the predictions given by individual meta-paths. However, the Ali dataset presents an exception, in which the max pooling performs better. The reason is that Ali is much sparser than the other datasets, which means a pair of user-item typically only has one or two meta-paths. In this scenario, mean pooling would be significantly affected by the low prediction scores from infrequent meta-paths, and max pooling would prevail by focusing on the most frequent and meaningful meta-paths. Therefore, adopting mean pooling on denser data and max pooling on sparser data might be a good strategy.

5.4.4 Model Scalability

To evaluate the scalability of DVAR, we investigate the training time per epoch w.r.t. different input graph size. To be more specific, we sample a proportion of users along with their interacted items to construct the input HIN. In total, we obtain five samples consisting of 10%, 30%, 50%, 70% and 100% of the users, respectively. We then train a model on each HIN constructed from the samples, and record the training time per epoch on the MovieLens and Serendipity datasets in Fig. 7, where the x-axis is the total number of nodes in each sampled HIN. As shown, the training time increases linearly with the increased proportion of users on both datasets. The linear growths implies that DVAR is able to scale up to large-scale recommendation scenarios.

5.4.5 Model efficiency

To evaluate the efficiency of DVAR, we investigate the computational overhead of DVAR in comparison to a few representative baselines. Specifically, we compare DVAR with a popular baseline NeuCF, and two competitive baseline BiANE and HERec. We train these models on the two largest datasets, namely, Serendipity and Ali, and report the overall time and training time. The overall time also includes the data pre-processing and I/O time, in addition to the model training time. Note that the running time is also influenced by the implementation (e.g., using TensorFlow or PyTorch), but this experiment can still shed some light on the broad differences in the order of magnitude of the time costs.

We present the experimental results in Fig. 8. On the one hand, compared to the competitive baselines (BiANE

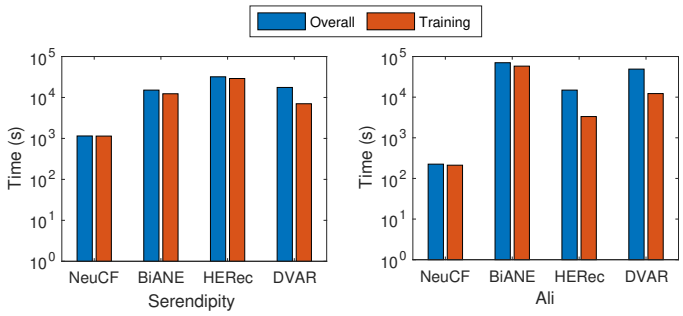


Fig. 8: Efficiency study on Serendipity and Ali dataset.

and HERec), our model DVAR maintains a similar order of magnitude in both the overall and training time cost, without incurring a significant overhead. That being said, the computational overhead of DVAR is somewhat greater than that of HERec on the Ali dataset, while on Serendipity the situation is opposite. A possible reason might be that the sparse and potentially noisy Ali data would make our FiLM-based model more difficult to converge [54], resulting in a longer training time. Nevertheless, BiANE, HERec and DVAR achieve generally similar time efficiency. On the other hand, while the popular baseline NeuCF can be much faster than the others, its recommendation performance can be significantly worse than DVAR in many instances.

5.5 Case Study

We present a case study to demonstrate how the macro- and micro-views interplay and influence the user behavior. The study also sheds light on a fine-grained understanding of the recommendation results. In Table 4, we examine an anonymous user with ID 251 in the MovieLens dataset, who is a 28-year-old male doctor.

5.5.1 Macro-view Preference

User 251 has interacted with a total of 71 movies in the dataset, where 44 of them are Hollywood actions or comedies. A sample of these movies are shown in Table 4. This suggests that his macro-view preference is Hollywood actions or comedies.

5.5.2 Micro-view Preference

As the majority of the user's interacted movies are in the action or comedy category, we focus on analyzing his micro-view preference in the two categories. We notice that among the action movies he has interacted with, 6 are starred by the same actor—Harrison Ford, as shown in bold in Table 4. Among the comedies, it can be observed that the movies "Grumpier Old Men" and "That Thing you Do!" are both starred by Kevin Pollak, which are also shown in bold. Such observations indicate that user 251's micro-view preferences in the two categories are different, i.e., for actions he likes Ford, and for comedies he likes Pollak. Meanwhile, he does not interact with any comedy starred by Ford (e.g., "Sabrina").

5.5.3 Prediction Results

In the testing data, three movies, namely, "The Fugtive", "Indiana Jones 3" in the action category and "That Thing You Do!" in the comedy category match the micro-view preferences of the user 251 identified above based on his preferred cast in each category. Our model DVAR recommends to the user all of the three movies in top 10 (average rank: 6.3), among other action or comedy movies matching his macro-view preference. In contrast, the macro-view model (labeled "Macro") only recommends "The Fugitive" but not the other two in top 10 (average rank: 21.3), and thus does not effectively capture the user's micro-view preferences. Nevertheless, it can still recommend other action or comedy movies generally well based on the user's macro-view preference. Overall, this case study shows preliminary evidence that DVAR can provide finer-grained and more interpretable recommendations, which are consistent with the dual-view assumption.

6 Conclusion

In this paper, we proposed a novel Dual-View Adaptive Recommendation (DVAR) model to fuse both the macro- and micro-views for user preferences learning. Equipped with a dual-view adaptive module, DVAR integrates both dual views in a dependence-aware manner, and becomes fully adaptive—the prediction parameters can flexibly adapt to different input users and item categories. To evaluate the merit of our model, we conducted comprehensive experiments on four real-world datasets in different application domains. The results show that DVAR significantly outperforms a range of state-of-the-art baselines.

Acknowledgement

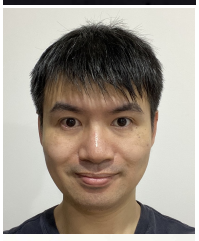
This research / project is supported by the Ministry of Education, Singapore, under its Academic Research Fund Tier 2 (MOE-T2EP20122-0041). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the Ministry of Education, Singapore.

References

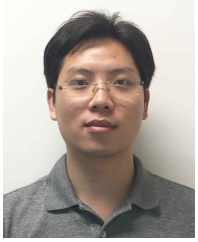
[1] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," *IEEE Internet Comput.*, vol. 7, no. 1, pp. 76–80, 2003.
[2] A. van den Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," in *NeurIPS*, 2013, pp. 2643–2651.
[3] H. Ma, H. Yang, M. R. Lyu, and I. King, "SoRec: Social recommendation using probabilistic matrix factorization," in *CIKM*, 2008, pp. 931–940.
[4] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.
[5] S. Rendle, "Factorization machines," in *ICDM*, 2010, pp. 995–1000.
[6] Z.-H. Deng, L. Huang, C.-D. Wang, J.-H. Lai, and P. S. Yu, "DeepCF: A unified framework of representation learning and matching function learning in recommender system," in *AAAI*, 2019, pp. 61–68.
[7] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *WWW*, 2017, pp. 173–182.
[8] R. van den Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," *arXiv preprint arXiv:1706.02263*, 2017.
[9] A. N. Nikolakopoulos and G. Karypis, "Recwalk: Nearly uncoupled random walks for top-n recommendation," in *WSDM*, 2019, p. 150–158.
[10] C.-M. Chen, C.-J. Wang, M.-F. Tsai, and Y.-H. Yang, "Collaborative similarity embedding for recommender systems," in *WWW*, 2019, pp. 2637–2643.

[11] S. Wang, L. Hu, Y. Wang, X. He, Q. Z. Sheng, M. A. Orgun, L. Cao, F. Ricci, and P. S. Yu, "Graph learning based recommender systems: A review," in *IJCAI*, 8 2021, pp. 4644–4652.
[12] C. Shi, B. Hu, W. X. Zhao, and P. S. Yu, "Heterogeneous information network embedding for recommendation," *IEEE TKDE*, vol. 31, no. 2, pp. 357–370, 2018.
[13] H. Li, Y. Wang, Z. Lyu, and J. Shi, "Multi-task learning for recommendation over heterogeneous information network," *IEEE TKDE*, 2020.
[14] C. Shi, Y. Li, J. Zhang, Y. Sun, and P. S. Yu, "A survey of heterogeneous information network analysis," *IEEE TKDE*, vol. 29, no. 1, pp. 17–37, 2017.
[15] L. Zhao, S. J. Pan, and Q. Yang, "A unified framework of active transfer learning for cross-system recommendation," *Artificial Intelligence*, vol. 245, pp. 38–55, 2017.
[16] P. Li and A. Tuzhilin, "DDTCDR: Deep dual transfer cross domain recommendation," in *WSDM*, 2020, pp. 331–339.
[17] Z. Cheng, Y. Ding, L. Zhu, and M. Kankanhalli, "Aspect-aware latent factor model: Rating prediction with ratings and reviews," in *WWW*, 2018, pp. 639–648.
[18] K. Bauman, B. Liu, and A. Tuzhilin, "Aspect based recommendations: Recommending items with the most valuable aspects based on user reviews," in *KDD*, 2017, pp. 717–725.
[19] J. Y. Chin, K. Zhao, S. Joty, and G. Cong, "ANR: Aspect-based neural recommender," in *CIKM*, 2018, pp. 147–156.
[20] T. Ebesu, B. Shen, and Y. Fang, "Collaborative memory network for recommendation systems," in *SIGIR*, 2018, pp. 515–524.
[21] D. Ha, A. Dai, and Q. V. Le, "Hypernetworks," in *ICLR*, 2017.
[22] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "FiLM: Visual reasoning with a general conditioning layer," in *AAAI*, 2018.
[23] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Analysis of recommendation algorithms for e-commerce," in *ACM EC*, 2000, pp. 158–167.
[24] S. Chen, S. Owusu, and L. Zhou, "Social network based recommendation systems: A short survey," in *IEEE SocialCom*, 2013, pp. 882–885.
[25] X. Zhang, X. Chen, J. K. Liu, and Y. Xiang, "Deeppar and deepdpa: Privacy preserving and asynchronous deep learning for industrial iot," *IEEE Trans. Industr Inform*, vol. 16, no. 3, pp. 2081–2090, 2020.
[26] Y. Xiao, L. Xiao, X. Lu, H. Zhang, S. Yu, and H. V. Poor, "Deep-reinforcement-learning-based user profile perturbation for privacy-aware recommendation," *IEEE Internet Things J.*, vol. 8, no. 6, pp. 4560–4568, 2020.
[27] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian personalized ranking from implicit feedback," in *UAI*, 2009, pp. 452–461.
[28] R. Akrour, M. Schoenauer, and M. Sebag, "April: Active preference learning-based reinforcement learning," ser. ECMLPKDD, 2012, p. 116–131.
[29] N. Houlsby, F. Huszár, Z. Ghahramani, and M. Lengyel, "Bayesian active learning for classification and preference learning," 2011.
[30] N. Koenigstein, G. Dror, and Y. Koren, "Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy," in *RecSys*, 2011, pp. 165–172.
[31] X. Wang, X. He, Y. Cao, M. Liu, and T.-S. Chua, "KGAT: Knowledge graph attention network for recommendation," in *KDD*, 2019, pp. 950–958.
[32] E. Quintanilla, Y. Rawat, A. Sakryukin, M. Shah, and M. Kankanhalli, "Adversarial learning for personalized tag recommendation," *IEEE Trans. Multimedia*, vol. 23, pp. 1083–1094, 2020.
[33] Z. Cheng, Y. Ding, X. He, L. Zhu, X. Song, and M. Kankanhalli, "A^3NCF: An adaptive aspect attention model for rating prediction," in *IJCAI*, 2018, pp. 3748–3754.
[34] Z. Sun, G. Guo, and J. Zhang, "Effective recommendation with category hierarchy," in *UMAP*, 2016, pp. 299–300.
[35] Y. Jhamb and Y. Fang, "A dual-perspective latent factor model for group-aware social event recommendation," *Information Processing & Management*, vol. 53, no. 3, pp. 559–576, 2017.
[36] M. Zhou, Z. Ding, J. Tang, and D. Yin, "Micro behaviors: A new perspective in e-commerce recommender systems," in *WSDM*, 2018, p. 727–735.
[37] Y. Gu, Z. Ding, S. Wang, and D. Yin, "Hierarchical user profiling for e-commerce recommender systems," in *WSDM*, 2020, p. 223–231.
[38] W. Bao, H. Wen, S. Li, X.-Y. Liu, Q. Lin, and K. Yang, "Gmcm: Graph-based micro-behavior conversion model for post-click conversion rate estimation," in *SIGIR*, 2020, p. 2201–2210.
[39] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *ICML*, 2017, p. 1126–1135.
[40] F. Pan, S. Li, X. Ao, P. Tang, and Q. He, "Warm up cold-start advertisements: Improving CTR predictions via learning to learn ID embeddings," in *SIGIR*, 2019, pp. 695–704.
[41] Y. Lu, Y. Fang, and C. Shi, "Meta-learning on heterogeneous information networks for cold-start recommendation," in *KDD*, 2020, pp. 1563–1573.
[42] H. Lee, J. Im, S. Jang, H. Cho, and S. Chung, "MeLU: Meta-learned user preference estimator for cold-start recommendation," in *KDD*, 2019, pp. 1073–1082.
[43] M. Brockschmidt, "GNN-FiLM: Graph neural networks with feature-wise linear modulation," in *ICML*, 2020, pp. 1144–1152.
[44] Z. Liu, Y. Fang, C. Liu, and S. C. Hoi, "Node-wise localization of graph neural networks," in *IJCAI*, 2021.
[45] Z. Wen, Y. Fang, and Z. Liu, "Meta-inductive node classification across graphs," in *SIGIR*, 2021, pp. 1219–1228.
[46] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu, "PathSim: Meta path-based top-k similarity search in heterogeneous information networks," *PVLDB*, vol. 4, no. 11, pp. 992–1003, 2011.
[47] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi- relational data," in *NeurIPS*, 2013, pp. 2787–2795.
[48] H. Li, Y. Liu, N. Mamoulis, and D. S. Rosenblum, "Translation-based sequential recommendation for complex users on sparse data," *IEEE TKDE*, vol. 32, no. 8, pp. 1639–1651, 2020.
[49] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *KDD*, 2017, p. 135–144.
[50] W. Huang, Y. Li, Y. Fang, J. Fan, and H. Yang, "BiANE: Bipartite attributed network embedding," in *SIGIR*, 2020, pp. 149–158.
[51] H. Valizadegan, R. Jin, R. Zhang, and J. Mao, "Learning to rank by optimizing ndcg measure," in *NIPS*, vol. 22, 2009.
[52] S. Lawrence and C. L. Giles, "Overfitting and neural networks: conjugate gradient and backpropagation," in *IJCNN*, vol. 1, 2000, pp. 114–119.
[53] J. Lever, M. Krzywinski, and N. Altman, "Points of significance: model selection and overfitting," *Nature methods*, vol. 13, no. 9, pp. 703–705, 2016.
[54] B. Xu, J. Zhang, R. Wang, K. Xu, Y.-L. Yang, C. Li, and R. Tang, "Adversarial monte carlo denoising with conditioned auxiliary feature modulation," *ACM Trans. Graph.*, vol. 38, no. 6, 2019.

Zhongzhou Liu receives his Master degree in Software Engineering from Wuhan University in 2020. He is now a Ph.D. student in the School of computing and Information Systems, Singapore Management University. His research interets including recommendation system, graph-based machine learning as well as data mining in social network.



Yuan Fang received his Ph.D. degree in Computer Science from University of Illinois at Urbana Champaign in 2014. He is currently an Assistant Professor in the School of Computing and Information Systems, Singapore Management University. His research focuses on graph-based machine learning and data mining, as well as their applications for the Web and social media.



Min Wu is a research scientist in Data Analytics Department, Institute for Infocomm Research. He received Ph.D. degree from Nanyang Technological University, Singapore, in 2011, and received B.S. degree in Computer Science from University of Science and Technology of China, 2006. His research interests include Graph Mining from Large-Scale Networks, Learning from Heterogeneous Data Sources, Ensemble Learning, and Bioinformatics.