# Incorporating Price into Recommendation With Graph Convolutional Networks

Yu Zheng, Chen Gao, Xiangnan He, Depeng Jin, *Member, IEEE*, and Yong Li, *Senior Member, IEEE*

**Abstract**—In recent years, much research effort on recommendation has been devoted to mining user behaviors, i.e., collaborative filtering, along with the general information which describes users or items, e.g., textual attributes, categorical demographics, product images, and so on. Price, an important factor in marketing — which determines whether a user will make the final purchase decision on an item — surprisingly, has received relatively little scrutiny. In this work, we aim at developing an effective method to predict user purchase intention with the focus on the price factor in recommender systems. The main difficulties are two-fold: 1) the preference and sensitivity of a user on item price are unknown, which are only implicitly reflected in the items that the user has purchased, and 2) how the item price affects a user's intention depends largely on the product category, that is, the perception and affordability of a user on item price could vary significantly across categories. Towards the first difficulty, we propose to model the transitive relationship between user-to-item and item-to-price, taking the inspiration from the recently developed Graph Convolution Networks (GCN). The key idea is to propagate the influence of price on users with items as the bridge, so as to make the learned user representations be price-aware. For the second difficulty, we further integrate item categories into the propagation progress and model the possible pairwise interactions for predicting user-item interactions. We conduct extensive experiments on two real-world datasets, demonstrating the effectiveness of our GCN-based method in learning the price-aware preference of users. Further analysis reveals that modeling the price awareness is particularly useful for predicting user preference on items of unexplored categories.

**Index Terms**—Price-aware, recommendation, collaborative filtering, price, user preference

## 1 Introduction

Recommendation is attracting increasing attention in both industry and academia, owing to the prevalence and success of recommender systems in many applications [1], [2], [3], [4]. From the perspective of product providers, the aim of building a recommender system is to increase the traffic and revenue, by recommending the items that a user will be likely to consume. As such, the key data source to leverage is the past consumption histories of users, since they provide direct evidence on a user's interest. To this end, much research effort has been devoted to collaborative filtering (CF) [5], [6], [7], which casts the task as completing the user-item consumption matrix, and incorporating side information into CF, such as textual attributes [8], [9], [10], categorical demographics [11], social information [12] and product images [13]. To utilize such diverse data in an unified model, a general class of feature-based recommendation models have been proposed, such as the pioneer work of factorization machines (FM) [14] and several recent developments that augment FM with neural networks [15], [16], [17].

With respect to E-commerce products and restaurants recommendation, where the item comes at an economic cost, not only the inherent interest of the user, but also the item price, plays a critical role in determining whether the user will make the final purchase decision. It has long been acknowledged that price is a significant factor in affecting user behaviors and product sales in marketing research [18], [19]. Nevertheless, and surprisingly, it has received relatively little scrutiny in recommendation.

Different from other item attributes like manufacturer and tags that influence a user's interest, the price of an item instead affects whether the user is willing to pay (WTP) for it. In other words, price and other attributes play orthogonal roles in the user decision making process — in most cases, only when both the item is of interest and its price is acceptable, will the user purchase it. In general, there are two difficulties in effectively integrating item price into recommender systems:

- *Unstated Price Awareness*. A user seldom states her preference and sensitivity on item price explicitly. As such, to build data-driven approaches, we have to infer a user's personalized awareness on item price from her purchase history. More challengingly, we need to consider the CF effect reflected in the histories of similar users to enhance the inference accuracy.
- *Category-Dependent Influence*. A user would have rather different perception and affordability on items

- *Yu Zheng, Chen Gao, Depeng Jin, and Yong Li are with the Beijing National Research Center for Information Science and Technology (BNRist), Department of Electronic Engineering, Tsinghua University, Beijing 100084, China. E-mail: {y-zheng19, gc16}@mails.tsinghua.edu.cn, {jindp, liyong07}@tsinghua.edu.cn.*
- *Xiangnan He is with the School of Information Science and Technology, University of Science and Technology of China, Hefei 101127, China. E-mail: xiangnanhe@gmail.com.*

*Manuscript received 14 April 2020; revised 14 December 2020; accepted 3 June 2021. Date of publication 22 June 2021; date of current version 10 January 2023.*
*This work was supported in part by the National Nature Science Foundation of China under Grants U1936217, 61971267, 61972223, 61941117, and 61861136003.*
*(Corresponding author: Yong Li.)*
*Recommended for acceptance by M. Wang.*
*Digital Object Identifier no. 10.1109/TKDE.2021.3091160*

of different categories. For example, a sport lover would have high tolerance on the price of a sport equipment, but not on alcoholic drinks. As such, it is important to take the item category information into consideration to accurately infer users' price preference.

As a special case of item side information, price could be integrated to generic recommender models like FM as a normalized numerical feature or discretized categorical feature. However, such solutions ignore the unique role of price in affecting user decision — they are not specifically designed to tackle the above-mentioned two challenges, thus whether the price sensitivity is properly captured remains unclear. In this work, we aim to address the two difficulties in price-aware recommendation system. We propose a new solution named ***P**rice-aware **U**ser **P**reference-modeling* (PUP), which employs the recently emerged Graph Convolution Networks (GCN) [20] to learn the price-aware and category-dependent user representations.

To be specific, we discretize the price variable and build a heterogeneous graph consisting of four types of nodes — users, items, prices and categories — where users connect to items, and items connect to prices and categories. We then propagate embeddings from prices to users with items as the bridge, so as to encode the indirect influence of prices on users. This makes the user embedding be related to the price embedding, such that high affinity is conceptually assigned to a user and her frequently purchased prices. To capture the CF effect of collective behaviors, we further propagate the user embeddings back to items and prices. Towards the second challenge of category relevance, we also integrate categories into the propagation process, and employ a pairwise interaction-based decoder to capture the interactions among users, items, prices and categories. Lastly, the overall model is optimized in end-to-end to estimate the consumption behaviors. Through these designs, our PUP method effectively incorporates the important yet complicated price factor into recommender systems.

To summarize, the main contributions of this work are as follows.

- We highlight the significance of the price factor in recommending items with economic cost, and propose a graph-based solution to unify the influence of item price and category to learn user preference.
- We experiment on real-world datasets to evaluate our method. Further analysis justifies the utility of modeling price in cold-start scenarios and recommending the items of unexplored categories for a user.

A conference version of this paper was published in [21]. We would like to state the new contribution of this paper compared with the ICDE version. First, in the proposed PUP method, there are several crucial hyper-parameters largely influencing the recommendation performance. We include hyper-parameter study in Section 4.7 in this journal version, covering most of the tunable hyper-parameters in the proposed method. Second, to provide more thorough understanding of the proposed method, we add an extra user study part in Section 4.8, where we investigate whether the proposed method is capable of recommending items with proper price from both user's sensitive category and non-sensitive category. Finally, there exist some unclear statements and typos in the conference version, and we further polish for clarity and quality of the presentation.

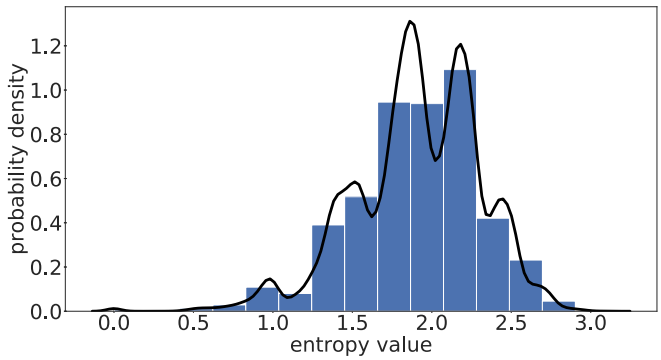



Fig. 1. Histogram of users' CWTP entropy value. High entropy value means users consider price differently in distinct categories.

The remainder of the paper is as follows. First we conduct some preliminary studies on a real-world dataset to analyze users' category-dependent price awareness in Section 2.1. Then we formalize the problem in Section 2.2 and present our proposed method in Section 3. After that we conduct experiments in Section 4 and we review related work in Section 5. Last, we conclude the paper in Section 6.

## 2 Motivation and Problem Formulation

### 2.1 Preliminary Study

In this section, we conduct statistical analyses on a real-world e-commerce dataset (details in Section 4). As stated in Section 1, price sensitivity depends largely on product category. To understand the inconsistent sensitivity across categories, we extend the widely used *willing to pay* (WTP) to *category willing to pay* (CWTP) which takes category into consideration. As an indicator reflecting users' price awareness, WTP is defined as the highest acceptable price of an item at which a user is willing to pay [22]. One step forward, we define CWTP as the highest price a given user is willing to pay for items of a given category. Since users usually interact with multiple categories, they will have multiple CWTP values (one CWTP per category). The distribution of CWTP values describes whether a user is consistent across different categories with respect to price awareness. For example, if a user always purchases expensive items regardless of categories, then her CWTP values will concentrate on large values. While if a user purchases expensive items from some categories and cheap items from other categories, her CWTP values will be much more diverse. Therefore, to measure the diversity of price awareness, we compute the entropy of CWTPs for each user, where a small entropy value implies that the user's price sensitivity is consistent across categories, while a large value means that the user considers price differently for distinct categories. Fig. 1 plots the histogram of the entropy value over all users. Only a few users' CWTP entropy values are 0, while most of the users have relatively large entropy value. In other words, except for a small number of users that only purchase items

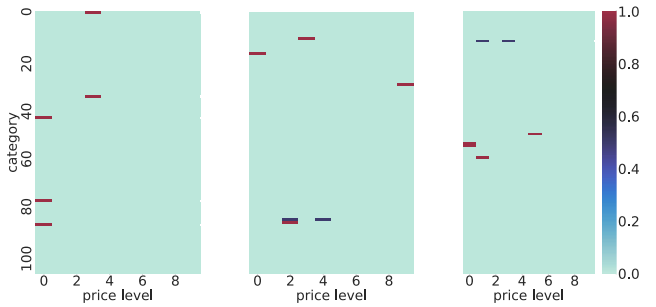



Fig. 2. Price-category purchase heatmap of three randomly selected users.

of the same price level in all categories, most users consider price differently across product categories.

In addition, we randomly sample three users from the dataset, and present their interaction history as a price-category heatmap in Fig. 2. We discretize the price into 10 levels using uniform quantization. A row in the heatmap represents one category and a column means one price level. The heatmap shows that the consumption on one category mostly concentrates on one price level, which confirms that the price sensitivity of a user is closely related to product categories. Furthermore, a user is likely to purchase cheap items in one category, but purchases products of a much higher price in another category.

In summary, through both macro-level statistical analyses and micro-level case studies, we find that the effect of price is relevant to category and such sensitivity is often inconsistent across different categories. The observations verify the category-dependent price awareness, which is the main difficulty when modeling price in recommendation.

### 2.2 Problem Definition

The focus of this work is to leverage item price to improve recommendation accuracy. As discussed above, since price awareness of a user is closely relevant to product category, it is essential to take category into consideration when designing a system of price-aware recommendation. We formulate this recommendation task as follows.

Let $U$ and $I$ denote the sets of users and items, and $R_{M\times N}$ denote the interaction matrix where $M$ and $N$ are the number of users and items. Here, an observed interaction $R_{ui} = 1$ in $R$ means user $u$ once purchased item $i$. We use $\mathbf{p} = \{p_1, p_2, \ldots, p_N\}$ and $\mathbf{c} = \{c_1, c_2, \ldots, c_N\}$ to denote the price and category of items.

For ease of modeling, we consider price as a categorical variable, discretizing a price value into separate levels using uniform quantization.[1] For example, suppose the price range of category *mobile phone* is [200, 3000] and we discretize it to 10 price levels. Then a mobile phone at the price of 1000 will have the price level $\left\lfloor \frac{1000-200}{3000-200} \times 10 \right\rfloor = 2$.

Finally, we formulate the problem of price-aware item recommendation as follows:

*Input:* Interaction matrix $R$, price of items $\mathbf{p}$ and category of items $\mathbf{c}$.

*Output:* The estimated probability of purchasing behavior given a user-item pair $(u, i)$.

## 3 Method

Fig. 3 illustrates our proposed PUP model. Given a user-item pair $(u, i)$ and the item's two attributes $< p_i, c_i >$ as the input, the model aims to predict the likelihood that $u$ will consume item $i$. Our proposed PUP method is featured with the following three special designs.

- Unified heterogeneous graph. To explicitly model user behaviors and item attributes, we discretize the price variable and build a heterogeneous graph with four types of nodes. To tackle the problem of unstated price awareness, we explicitly introduce price as price nodes on the graph instead of input features of item nodes. As for the difficulty of category-dependent influence, we further add category nodes to the graph.
- Graph convolutional encoder. To capture both the CF effect and price awareness, we utilize a graph convolutional network as an encoder to learn semantic representations for users, items, prices and categories. With embeddings propagating on the heterogeneous graph, users' price sensitivity is captured by aggregating price-aware information into user nodes.
- Pairwise-interaction based decoder. Since the heterogeneous graph consists of four types of nodes which are factorized to a shared latent space, inspired by the philosophy of Factorization Machines [14], we employ a pairwise interaction-based decoder to estimate interaction probability.

To capture the complicated price factor in recommendation, we estimate the category-dependent and price-aware user preference using two branches — one branch focuses on a user's interest and models price as a global effect representing a user's overall purchasing power which is unrelated to category, while the other branch focuses on the category-dependent influence of price factor. In this paper, we will call the first branch as the global branch and the second as the category branch. For each branch, we construct a heterogeneous graph and employ a graph convolutional encoder and a pairwise-interaction decoder. For simplicity, we introduce our method in a single branch manner and the two-branch version is similar like mirror image.

### 3.1 Unified Heterogeneous Graph

For the task of price-aware item recommendation, where we have both user-item interaction data and items' price attributes, it is challenging to explicitly capture users' price awareness since user is not directly related to price. In other words, a user's relation with price is build upon the transitive relation of user-to-item and item-to-price. In this way, items play a *bridge* role connecting users and prices. To address this challenge, we discretize the price variable and build a heterogeneous graph consisting of four types of nodes – users, items, prices and categories.

1. In the following of the paper, we will use "price" and "price level" interchangeably to denote the categorical variable.

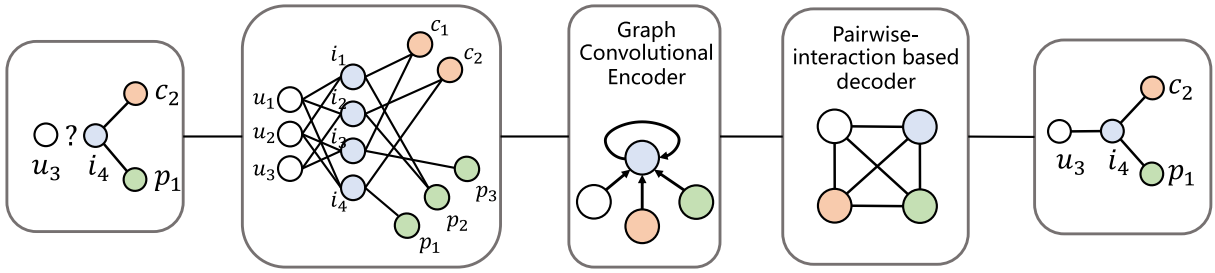



Fig. 3. The overall design of our proposed PUP method which consists of a unified heterogeneous graph, a graph convolutional encoder and a pairwise-interaction based decoder. The constructed unified heterogeneous graph is composed of four types of nodes where user nodes connect to item nodes, and item nodes connect to price nodes and category nodes.

Formally, the input interaction data and attributes (category and price) can be represented by an undirected graph $G = (V, E)$. Node set $V$ consists of user nodes $u \in U$, item nodes $i \in I$, category nodes $c \in \mathbf{c}$ and price nodes $p \in \mathbf{p}$. Edge set $E$ is composed of interaction edges $(u, i)$ with $R_{ui} = 1$, category edges $(i, \mathbf{c}_i)$ and price edges $(i, \mathbf{p}_i)$ with $i \in I$. The second block in Fig. 3 illustrates our constructed graph. In this way, we represent all the entities, features and relations in a unified graph, so as to capture all pairwise relations in an explicit manner.

Notice that we use separate node types for category and price instead of a single node type for the cross feature of (category, price) to avoid redundant parameters. Intuitively, items of the same category with different price shares functionality similarity. Meanwhile, items of the same price from various categories reflect similar price awareness as well. Thus a single type of cross feature lacks connections of the two situations above. By applying distinct node types to category and price, different levels of semantic similarities are captured in the graph.

With respect to GCN for node classification [23], it is common to use high-level feature vectors like word embeddings extracted by word2vec [24] as the input node features. Following the same fashion, it seems reasonable to encode price and category information into the input features for user nodes and item nodes which leads to a rather concise bipartite design. However, in our work, we explicitly squeeze out the two important attributes (price and category) as entity nodes to capture the category-dependent price awareness in a more expressive way.

As prices and categories are captured directly and explicitly by assigning separate nodes to them, the two aforementioned difficulties of price-aware item recommendation are alleviated. Specifically, the unstated price awareness is transformed to high-order neighbor proximity on the heterogeneous graph which could be well captured by graph convolutional networks. And the category-dependent influence issue is alleviated by linking item nodes to both price nodes and category nodes. It is worthwhile to notice that the relative order of price levels is now implicitly captured by the similarity of the learned node embeddings, which is more flexible compared with using scalar price values since users do not always prefer higher or lower price levels.

### 3.2 Graph Convolutional Encoder

Latent factor model (LFM), which tries to encode entities in a low-dimensional latent space, is a widely-used mechanism in recommender systems [25], [26]. For traditional LFM in recommender system, such as Matrix Factorization, an observed $(u, i)$ pair will push $u$ and $i$ to each other in latent space. However, in our built unified heterogeneous graph, there are two more pairs, $(i, p)$ and $(i, c)$ . Besides, there are underlying user-price interaction $(u, p)$ when user $u$ purchases item $i$ with price $p$. In this paper, we extend traditional LFM that only learns representations for users and items, and try to learn representations of four types of entities in the same latent space. Recent research [23], [27], [28], [29] have shown that message passing on the graph could lead to semantic and robust node representations for multiple tasks like node classification and link prediction. A special class of algorithms among them called Graph Neural Networks achieve the state of art in the field of network representation learning. We employ an encoding module consisting of an embedding layer for converting one-hot input to low-dimensional vectors, an embedding propagation layer to capture both CF effect and price awareness, and a neighbor aggregation layer to model neighbor similarity.

*Embedding Layer.* As discussed previously, in our proposed model, since price attributes and category attributes are squeezed out as nodes, ID is the only feature for a node. Thus, we introduce an embedding layer to compress the one-hot ID encoding to a dense real-value vector. That is, we represent each node with a separate embedding $e' \in \mathbb{R}^d$, where $d$ is the embedding size. To be specific, we first allocate a unique ID to each user, item, category and price, by translating the original ID with offset. Then we multiply the one-hot ID feature with a weight matrix to attain the embedding.

*Embedding Propagation Layer.* In GCN, embeddings of nodes propagate to their first order neighbors and further if more than one convolutional layer is applied. In our encoder, the embedding propagation layer captures the message transferred between two directly connected nodes which could be user-item, item-price or item-category. Suppose node $i$ and node $j$ are two connected nodes in our heterogeneous graph. The propagated embedding from node $j$ to node $i$ is formulated as follows:

$$\mathbf{t}_{ji} = \frac{1}{|\mathcal{N}_i|} e'_j, \tag{1}$$

where $\mathcal{N}_i$ denotes the set of neighbors for node $i$, and $e'_j$ is the embedding of node $j$ in the embedding layer. As stated in [30], adding self-loops is of significant importance to GCN since it shrinks the spectrum of the normalized Laplacian, thus we link each node in our heterogeneous graph to itself which makes node $i$ also appear in $\mathcal{N}_i$.

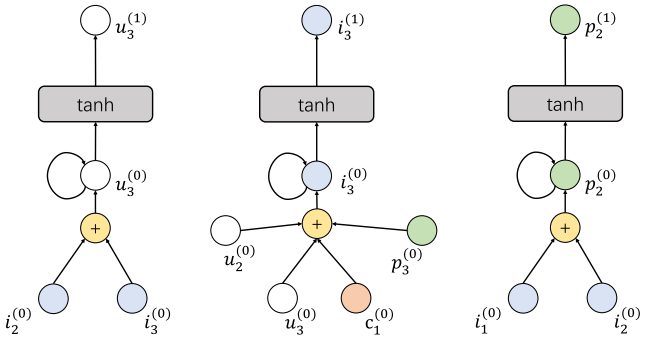



Fig. 4. The updating rule for different types of nodes. Left: user node. Mid: item node. Right: a price node. Category nodes update similarly with price nodes.

*Neighbor Aggregation Layer.* From the perspective of network representation learning, the neighbor relation of two nodes in the graph structure implies that their representations should also be near in the transformed latent space. Inspired by recent advances in graph convolutional networks [27], [28], [31], we update the representation of a node by aggregating its neighbors' representations. Among all the aggregating operations, summation, taking average, and LSTM are the most frequently used approaches [27], [32]. In our proposed encoder, we adopt average pooling and utilized a non-linear activation function to perform message passing on the graph.

Specifically, let $e_u$, $e_i$, $e_c$, and $e_p$ denote the representations for user $u$, item $i$, category $c$ and price $p$. The updating rule can be formulated as follows:

$$\begin{aligned} o_u &= \sum_{j \in \{i \text{ with } R_{ui}=1\} \cup \{u\}} \mathbf{t}_{ju}, \\ o_i &= \sum_{j \in \{u \text{ with } R_{ui}=1\} \cup \{i, \mathbf{c}_i, \mathbf{p}_i\}} \mathbf{t}_{ji}, \\ o_c &= \sum_{j \in \{i \text{ with } \mathbf{c}_i=c\} \cup \{c\}} \mathbf{t}_{jc}, \\ o_p &= \sum_{j \in \{i \text{ with } \mathbf{p}_i=p\} \cup \{p\}} \mathbf{t}_{jp}, \\ e_f &= \tanh(o_f), f \in \{u, i, c, p\}. \end{aligned} \tag{2}$$

Fig. 4 illustrates the updating rule for different types of nodes. Being able to learn high-order proximity is one of the key advantages of GCN. With more convolutional layers stacked together, the learned representations can effectively model the relations between nodes and their high order neighbors, which makes it possible to recommend items that match users' category-dependent price awareness.

Intuitively, items of the same price level are likely to be more similar than items of different price levels. In our constructed heterogeneous graph, a price node links to all the items of that price level, and the graph convolutional encoder guarantees that the output representations for those items will absorb the price embedding into themselves by embedding propagation and neighbor aggregation. Therefore, the encoder generates item representations with price-aware similarities. Category-aware similarities are captured in the same way since a category node is connected to all the item nodes which belong to that category.

Furthermore, a user's price awareness is largely reflected by her interacted items and other users' purchase history through collaborate filtering. Thus it is crucial to leverage items as the bridge between users and price awareness. In our model, a user's representation is aggregated from her interacted items explicitly and the items are directly linked to categories and prices. Thus category nodes and price nodes are high-order neighbors with respect to user nodes, and the price awareness is propagated to the users via intermediate item nodes.

From the perspective of recommendation, the graph convolutional encoder is able to capture the similarity of any two nodes when there exists a path between them. With respect to the classic Matrix Factorization algorithm, collaborative filtering effect is captured implicitly by optimizing to estimate user-item interactions. However, in our graph convolutional encoder, we explicitly incorporate collaborate filtering effect by aggregating a node's neighbors. Specifically, similar users who have interacted with the same item are 2-order neighbors on the graph and this proximity could be captured by the graph convolutional encoder.

*Implementation.* Similar to the semi-supervised approach proposed in [23], the graph convolutional encoder can be implemented effectively utilizing sparse matrix production. We define the rectified adjacency matrix $\hat{A}$ which is sparse as follows:

$$\hat{A} = f(A + M_I), \tag{3}$$

where $A$ is the original adjacency matrix and $M_I$ is the identity matrix. In terms of the original adjacency matrix $A$, each row or column represents an entity, which could be user, item, category and price. And only user-item block, item-category block and item-price block have one's in it, while other blocks are zero blocks. $f(M)$ takes average on each row of matrix $M$. By taking row-wise average, the rectified adjacency matrix encodes the embedding propagation layer, since we normalize the propagated embedding by the number of neighbors of one node. Adding the identity matrix is equivalent to adding self-loops on the graphs which will influence the performance of GCN significantly [30].

We denote the input one-hot encoding feature matrix as $F_{\text{in}}$ where each row represents a node, and use $W$ to represent the learnable embedding matrix. Then the output representation matrix is computed as follows:

$$F_{\text{out}} = \tanh(\hat{A} F_{\text{in}} W). \tag{4}$$

where multiplying $W$ transforms the one-hot feature to embedding feature, and multiplying $\hat{A}$ accomplishes embedding propagation and neighbor aggregation.

### 3.3 Pairwise-Interaction Based Decoder

We adopt a two-branch design to estimate user-item interactions. The global branch models the price effect in a large scope focusing on a user's overall purchasing power. The category branch instead concentrates on a rather "local" level where the category factor influences a user's price sensitivity. For each branch, we employ a pairwise-interaction based decoder to estimate the interaction probability and combine the two predicted scores as the final result.

As we represent users, items, categories and prices as four types of nodes on a unified heterogeneous graph, the learned representations for different types of nodes share the same latent space. Inspired by Factorization Machines [14], we employ a decoder following the FM fashion to learn the complex relationships including user-item interaction, user-price preference, item-price bias and so on. Formally, using the same notation from the previous encoder section, the estimated purchase probability between user $u$ and item $i$ of category $c$ and price $p$ can be formulated as follows:

$$\begin{aligned} s &= s_{\text{global}} + \alpha s_{\text{category}}, \\ s_{\text{global}} &= e_u^T e_i + e_u^T e_p + e_i^T e_p, \\ s_{\text{category}} &= e_u^T e_c + e_u^T e_p + e_c^T e_p, \end{aligned} \tag{5}$$

where the final prediction combines the results from two branches with a hyper-parameter $\alpha$ to balance the two terms. Note that the each branch has its own graph convolutional encoder, thus the embeddings used for computing $s_{\text{global}}$ and $s_{\text{category}}$ are different and independent.

With respect to the global branch, complex relationships of users, items and prices are learned in a 2-way FM manner. In this branch, the three inner products each captures the user's interest, the user's global price effect and the item's price bias respectively. we estimate the interaction probability without category embeddings and thus category nodes only serve as a regularization term on the graph which makes items of the same category near to each other. Since category information is hidden in the decoder process in the global branch, the *local* effect of price which is related to category is pushed out from the learned latent space. And the global price influence, which reflects a user's overall purchasing power and affordability, is reserved in the latent space learned by the powerful graph convolutional encoder.

However, as discussed previously, users' price sensitivity is largely relevent to category and often appears inconsistently across different categories. Thus we add a category branch which serves to capture this subtle price awareness which is related to category. In this branch, we omit item embeddings when estimating interactions and only take users, categories and prices into consideration. Item nodes simply play the *bridge* role transferring price and category information to users. By taking inner products of the three embeddings, users' category-dependent price awareness is captured in the shared latent space.

On top of the learned high-quality representations from graph convolutional encoder, we employ a two-branch pairwise-interaction based decoder to model the complex relationships among users, items, categories and prices, by factorizing all the features in a shared latent space. We then estimates the interaction by taking inner products of every pair of feature vectors. Moreover, our decoder serves great interpretability with respect to price awareness since the two-branch design disentangles the global effect and the category-dependent effect of the price factor.

### 3.4 Model Training

*Semi-Supervised Graph Auto-Encoder.* To train our proposed PUP model, we follow the fashion of semi-supervised graph auto-encoder [28], [33], [34]. That is, during encoding, we utilize a GCN which aims at learning robust representations for all the four types of nodes. While during decoding, we only focus on reconstructing user-item edges on the heterogeneous graph and omit item-price and item-category edges, because predicting user-item interaction is the main task for recommendation.

*Loss Function.* We adopt Bayesian Personalized Ranking (BPR)[6] as our loss function which has been widely used for implicit data [7], [35], [36]. This pairwise object which focuses on the relative preference priority of items instead of absolute interests could be formulated as follows:

$$L = \sum_{(u,i,j)\in\mathcal{O}} -\ln(\sigma(s(u,i)) - \sigma(s(u,j))) + \lambda\|\Theta\|^2, \tag{6}$$

where $\mathcal{O}$ denotes the set of positive-negative sample pairs and $\sigma$ stands for *sigmoid* function. The second term of Equation (6) performs L2 regularization where $\Theta$ stands for model parameters and $\lambda$ controls the penalty strength.

*Dropout.* Dropout is an effective way to prevent models from overfitting [37]. We adopt dropout on the feature level, which means randomly dropping the output representations with probability $p$. $p$ is a hyper-parameter in our method. We perform grid search on this hyper-parameter and report the best performance. With the help of dropout technique, our proposed PUP method can learn more robust node representations on the unified heterogeneous graph.

TABLE 1
Statistics of the Datasets

| Dataset | #Users | #Items | #Cate | #Price | #Interactions |
|---|---|---|---|---|---|
| Yelp | 20637 | 18907 | 89 | 4 | 505785 |
| Beibei | 52767 | 39303 | 110 | 10 | 677065 |
| Amazon | 48424 | 33483 | 5 | 10 | 438355 |

## 4 Experiments

In this section, we first investigate the performance of our proposed PUP method compared with existing baselines and verify the effect of price-aware recommendation system. One step forward, we dive deeper into the role price factor plays in our proposed method. Then we study the effect of our two-branch design to see whether price awareness is carefully modelled in our method. Furthermore, we test the performance on users with different consistency in terms of the price awareness across categories. Moreover, as stated in [38], incorporating the price factor could improve the performance when recommending items of unexplored categories which is a cold-start problem. Thus we also measure our proposed model under this protocol.

### 4.1 Experimental Settings

#### 4.1.1 Dataset and Evaluation Protocol

To evaluate the performance of our proposed PUP method, we utilize three real-world datasets for comparison: Yelp, Beibei and Amazon, which all have abundant category and price information for items. Statistics of these three datasets are summarized in Table 1.

- *Yelp*. We adopt Yelp2018 Open Dataset[2] in which restaurants and shopping malls are regarded as items. We choose all the sub-categories under the top-level category *restaurant*. In this dataset, price of each restaurant is shown as different number of dollar symbols which ranges from 1 to 4. Thus we directly use the number of dollar symbols as price levels in our experiments. Finally, we utilize 10-core settings which means only retaining users and items with at least 10 interactions.
- *Beibei*. This is a dataset collected from one of the largest E-commerce platforms[3] in China. In this dataset, all items are with specific category and price information. Since the price of each item in this dataset is of continuous form, we discretize the continuous price to 10 price levels using uniform quantization and use the 10-core settings to guarantee data quality.
- *Amazon*. To confirm the importance of the price factor, we also conducted experiments on a general dataset which is collected from the reviews on Amazon[39] and the original price information is available. We selected product reviews of 5 categories (*Cell Phones and Accessories*, *Tools and Home Improvements*, *Toys and Games*, *Video Games* and *Beauty*). We utilized the 5-core version provided by [39].

For each dataset, we first rank the records according to timestamps and then select the early 60% as the training set, middle 20% as the validation set, and the last 20% as the test set. For each user, the items that are not interacted by the user are viewed as negative samples. We perform negative sampling to constitute positive-negative sample pairs for training. To evaluate the effectiveness of top-K recommendation, we use the same metrics as in [7] including Recall and NDCG. We set top-K as 50 and 100 to evaluate the general candidate retrieval performance. We report average metrics for all the users in the test set.

#### 4.1.2 Baselines

To show the effectiveness of our proposed PUP method, we compare the performance with the following baselines.

- *ItemPop*. It is a non-personalized method that ranks items just according to their popularity in the training set.
- *BPR-MF [6]*. It is a matrix factorization model optimized by Bayesian Personalized Ranking (BPR) [6] loss.
- *PaDQ [38]*. This method is based on CMF [40] to handle price information. Specifically, it factorizes user-item, user-price and item-price matrix simultaneously with shared latent representations, which regards price as an extra target to predict.
- *FM [14]*. Factorization Machines (FM) is a competitive model which applies a sum of pairwise inner product of user or item features to obtain the prediction score. In our experiments, we integrate price and category into FM by regarding them as item features.
- *DeepFM [15]*. This method is an ensemble model that combines FM and deep neural networks to capture both low- and high- order feature interactions.
- *GC-MC [28]*. It adopts GCN to learn representations on a bipartite user-item graph. We use one-hot ID features as input features for user and item nodes.
- *NGCF [20]*. This method is a GCN based method which explicitly captures the collaborative signal by performing embedding propagation on a bipartite user-item graph. In our experiments, we use a concatenation of one-hot ID feature and one-hot price feature as the input feature for item nodes.

We did not compare our method with FMF proposed by [41] and $\mathrm{SVD}_{\mathrm{util}}$ proposed by [42] since these methods incorporate the *dynamic* of price or net utility which we leave it for future work.

#### 4.1.3 Parameter Settings

We adopt BPR loss for all methods and fix the embedding size as 64 for fair comparison, which is a reasonable choice as suggested by experiments in Section 4.7.1. We use Adam for optimization with the initial learning rate as 1e-2. The batch size is fixed at 1,024 and negative sampling rate is set to 1. For each model, we train for 200 epochs and reduce the learning rate by a factor of 10 twice for convergence.

### 4.2 Performance Comparison

We first compare the results of all the methods on two datasets with respect to: Recall@50, NDCG@50, Recall@100 and NDCG@100. Table 2 presents the overall comparison of our proposed PUP method and other baselines. From the results, we have several important observations.

*Incorporating Price into Recommendation Improves the Accuracy.* Generally, incorporating more attributes and features into recommender systems would increase the overall recommendation performance. Compared to trivial CF methods like MF which only consider users and items, attribute-aware methods are able to capture the relationship between much more features and interactions. In this work, we only focus on two attributes which are category and price. As illustrated in the results, attribute-aware methods outperform other trivial CF methods. With respect to price-awareness modeling, experimental results in both datasets verify that incorporating price into recommendation could attain improvements in accuracy. Specifically, FM and DeepFM outperform BPR-MF, and NGCF outperforms GC-MC in most cases. As stated previously, the input feature for NGCF contains price information while the input of GC-MC is just one-hot ID encoding. However, the performance gain is not significant, which indicates that the two challenges of price-aware recommendation remain unresolved in these attribute-aware methods.

*Price Should be Considered More as an Input Rather Than a Target.* PaDQ and FM, which are two typical methods of attribute-aware recommendations, differ in how they incorporate price into the system. Specifically, PaDQ works in a generative way, which means it regards price as an extra target to predict. However, FM follows a deterministic

2. https://www.yelp.com/dataset
3. https://www.beibei.com

TABLE 2
Top-K Recommendation Performance Comparison on the Yelp and Beibei Datasets (K is Set to 50 and 100)

| | Yelp dataset | | | | Beibei dataset | | | |
|---|---|---|---|---|---|---|---|---|
| method | Recall@50 | NDCG@50 | Recall@100 | NDCG@100 | Recall@50 | NDCG@50 | Recall@100 | NDCG@100 |
| ItemPop | 0.0401 | 0.0182 | 0.0660 | 0.0247 | 0.0087 | 0.0027 | 0.0175 | 0.0046 |
| BPR-MF | 0.1621 | 0.0767 | 0.2538 | 0.1000 | 0.0256 | 0.0103 | 0.0379 | 0.0129 |
| PaDQ | 0.1241 | 0.0572 | 0.2000 | 0.0767 | 0.0131 | 0.0056 | 0.0186 | 0.0068 |
| FM | 0.1635 | **0.0771** | 0.2538 | 0.1001 | **0.0259** | 0.0104 | 0.0384 | 0.0130 |
| DeepFM | 0.1644 | 0.0769 | 0.2545 | 0.0998 | 0.0255 | 0.0090 | **0.0400** | 0.0122 |
| GC-MC | 0.1670 | 0.0770 | **0.2621** | **0.1011** | 0.0231 | 0.0100 | 0.0343 | 0.0124 |
| NGCF | **0.1679** | 0.0769 | 0.2619 | 0.1008 | 0.0256 | **0.0107** | 0.0383 | **0.0134** |
| PUP | **0.1765** | **0.0816** | **0.2715** | **0.1058** | **0.0266** | **0.0113** | **0.0403** | **0.0142** |
| impr.% | 5.12% | 5.84% | 3.59% | 4.65% | 2.70% | 5.61% | 0.75% | 5.97% |

fashion which means regarding price as input features. As the results illustrated, FM substantially surpasses PaDQ and PaDQ is even worse than BPR-MF. The large performance gap indicates that price should be considered more as an input of recommendation rather than a target to predict.

*Neural Based Methods and Graph Based Methods Have an Advantage over Other Methods.* From the results, neural based methods and graph based methods achieve better results than other "shallow" models in most cases. This performance gain is reasonable since neural units increase the capacity of the model and the graph structure is much more expressive than a look-up embedding table. Specifically, DeepFM, GC-MC and NGCF generally attain better results. DeepFM combines DNN and FM to capture low- and high-order interactions simultaneously. GC-MC applies graph convolution on the user-item bipartite to learn more meaningful representations for users and items. NGCF captures the CF effect by performing embedding propagation on the graph. The results show that it is promising to enhance representation learning and interaction modeling in recommendation by introducing neural networks and graph neural networks into the system.

*Our Proposed PUP Method Achieves the Best Performance.* Our proposed PUP consistently achieves the best results on all metrics of both datasets. Following the semi-supervised graph auto-encoder fashion, this two-branch method which is specifically designed for modeling price awareness is proven to be effective. Reuslts of t-tests indicate that the improvements are statistically significant for $p < 0.005$.

To summarize, extensive comparisons on two datasets verify that our proposed PUP method is able to effectively leverage price of items to improve recommendation.

## 4.3 The Effect of Price Factor

In this section, we perform ablation study to verify the importance of incorporating price into recommender systems. Several modified models with the price factor removed are constructed for comparison with our proposed PUP method. Furthermore, in modern e-commerce platforms, the price range is often large and the distribution of price tends to be much complicated. The uniform quantization process of price factor might fail to capture the price preference when price is not uniformly distributed. Therefore, we adapt the quantization process according to the rank of price and compare the two quantization methods. With respect to the discretization process of price factor, the number of price levels has a great influence on the fineness of capturing the price preference which directly impacts the performance of the recommendation system. Thus we conducted experiments on different price levels to investigate how the granularity of price affects the recommendation results.

### 4.3.1 Ablation Study of Price Factor

In our proposed two-branch design, the price factor is incorporated into recommender systems from both global level reflecting users' purchasing power and local level focusing on category-dependent price awareness. We constructed several slim versions of our PUP model to verify the necessity of taking price into consideration. In these slim versions, category factor, or price factor, or both are removed. Results are illustrated in Table 3. From the results we can address that incorporating price into recommender systems brings significant improvements with respect to recommendation accuracy and the price factor is indeed a crucial feature in e-commerce recommendation scenarios. Nevertheless, jointly modeling the price factor and the category factor achieves the best performance which verifies the necessity of taking category into consideration. The large performance gap between PUP and other slim versions confirms that our proposed method is of great effectiveness in capturing users' price awareness.

### 4.3.2 The Quantization Process of Price Factor

In most cases, the distribution of price factor on e-commerce platforms is complicated. Usually there are a few products with extremely high or particularly low price which in turn makes the price range large. However, the majority of the products often lie in the middle of the range. The non-uniform

TABLE 3
Study of Price Factor on Amazon Dataset

| method | Recall@50 | NDCG@50 | Recall@100 | NDCG@100 |
|---|---|---|---|---|
| PUP w/o c,p | 0.0726 | 0.0211 | 0.1155 | 0.0285 |
| PUP w/ c | 0.0633 | 0.0222 | 0.0944 | 0.0276 |
| PUP w/ p | 0.0854 | 0.0277 | 0.1275 | 0.0350 |
| PUP | 0.0890 | 0.0293 | 0.1336 | 0.0370 |

TABLE 4
Results of Different Quantization Process on Amazon Dataset

| Method | Recall@50 | NDCG@50 | Recall@100 | NDCG@100 |
|---|---|---|---|---|
| Uniform | 0.0807 | 0.0264 | 0.1192 | 0.0331 |
| Rank | 0.0885 | 0.0294 | 0.1313 | 0.0368 |

distribution of price factor requires subtle adaptation of the uniform quantization process. We adopt rank-based quantization to alleviate the problem of uniform quantization whose performance is suboptimal when the price distribution is complex. In uniform quantization, we calculate the normalized price by subtracting the minimum price and then divided by the price range of the corresponding category. However, in rank-based quantization, we first rank the products by price within their categories and then transform the rank to percentile form. At last, the discretized price is obtained by multiplying the total number of price levels and taking the integer part. Results of the two different quantization processes are shown in Table 4. It is reasonable that rank-based quantization attains much better results than uniform quantization since the price factor is not uniformly distributed. Rank-based quantization transforms the non-uniform distribution of the absolute price value to the uniformly distributed ranking percentage value. This adaptation of the quantization process of price factor makes our proposed PUP model still feasible under the circumstance of large price range and complicated distribution of price factor.

#### 4.3.3 The Fineness of the Price Factor

Usually price is of continuous form on e-commerce platforms. However, utilizing the price factor directly as an absolute value brings much complexity to the system since nodes on the heterogeneous graph are discrete in essence. Nevertheless, consumers tend to pay attention to other factors like brand or sales when the candidate items cost roughly the same. Therefore, we translate the price to price levels when it is continuous for both simplicity and rationality. The number of price levels is an important option which decides the fineness of the price factor in our proposed method. We experimented on different price levels to study how the granularity of the price factor influences the recommendation performance. Fig. 5 shows the results of different price levels. When the number of price levels is set extremely low such as two which means the model only makes a coarse discrimination between cheap and expensive, the price factor is not accurately incorporated, thus the overall performance is inferior to more elaborate models with finer capture of price factor. While if we set the number of price levels too high like 100, items of near price are allocated to different price levels which could damage the performance as well since the difference in price is not so important under this condition.

### 4.4 The Two-Branch Design

Previously we introduced our two-branch design which aims at disentangling the global and local effect of price awareness in recommendation. The final prediction is made by combining the two branches. Balancing between the two branches is crucial in the proposed method, since the global effect and category-dependent effect play different roles in users' decision-making process. Generally, there are two approaches to accomplish such balance, embedding allocation and balancing weight. In this section, we investigate these two approaches.

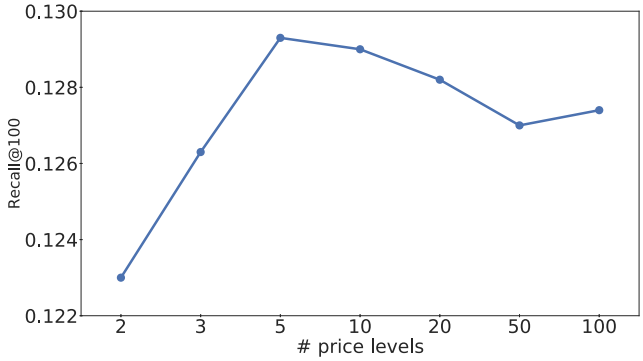



Fig. 5. Performance on amazon at different number of price levels.

#### 4.4.1 Embedding Allocation

If we fix the holistic embedding size, the specific dimension at which we slice the embedding requires careful consideration. With more dimensions allocated to the global branch, the PUP model pays more attention to users' interest and the average influence of the price factor. With the category branch getting more dimensions, the local and category-dependent price awareness plays a more important role in estimating interactions. Intuitively, larger embedding size leads to more capacity and expressive power in that branch.

We slice at different dimension to study how the allocation of embedding size influence price awareness modeling. Results are shown in Table 5. In the table, an allocation of $m/n$ indicates the embedding size is $m$ for the global branch and $n$ for the category branch. It is reasonable that better performance is achieved when the global branch takes the majority since items are of vital importance when estimating user-item interactions while in the category branch item embeddings are hidden. However, further compressing the embedding size of the category branch to extremely low such as 4 or lower will worsen the recommendation accuracy. Extremely low embedding size restricts the capacity of the category branch and thus the category-dependent price awareness could not be well captured which leads to inferior performance.

#### 4.4.2 Balancing Weight

In the pairwise-interaction based decoder, we combine the estimation of two branches together, by performing a weighted sum with an introduced hyper-parameter $\alpha$. With

TABLE 5
Performance Comparison of Different Embedding Size Allocations on Yelp Dataset (Topk=50)

| Allocation | 16/48 | 32/32 | 48/16 | 56/8 | 60/4 |
|---|---|---|---|---|---|
| Recall | 0.1460 | 0.1689 | 0.1757 | **0.1765** | 0.1745 |

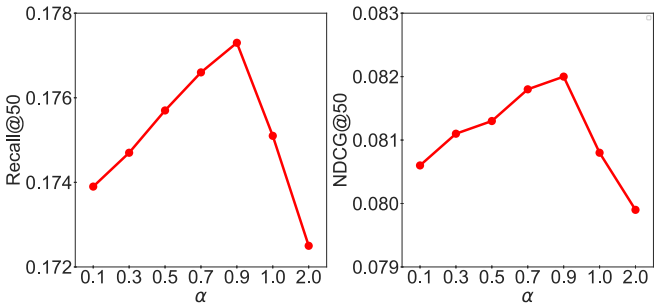



Fig. 6. Performance comparison of different balancing hyper-parameter $\alpha$ on Yelp dataset.

larger $\alpha$, we impose higher importance on the category branch. To study how this hyper-parameter affects the overall performance, we conduct experiments of different $\alpha$ on Yelp dataset, and results are shown in Fig. 6. From Fig. 6 we could observe that PUP achieves the best performance with respect to both Recall@50 and NDCG@50 when $\alpha$ is 0.9, and the performance drops when it turns to a higher value. This indicates that the category-dependent part is a minor aspect, compared with price's global effect, which is reasonable since users' overall purchasing power largely determines users' decision making.

In the propose PUP method, allocating different dimensions to the two branches and setting different values of $\alpha$ serve as two ways to balance between the modeling of global effect and category-dependent effect of price factor. Specifically, embedding allocation controls the expressive power and capacity, while balancing weight controls the importance. Since the CF signal is the most important information, a large proportion of dimensions should be allocated to the global branch. Meanwhile, the allocated dimensions for the category branch should not be too small, otherwise it would greatly limit the effect of capturing category-dependent price awareness. On the other hand, the category-dependent part is a minor aspect, and thus the balancing weight should better not surpass the major global part.

### 4.5 Consistency of Price Awareness Across Categories

Since the effect of price on influencing users' purchase behavior largely depends on item categories, whether the price awareness is consistent across categories is crucial when making recommendations for items of various categories. We divide users into groups according to their entropy value of CWTPs which was defined in Section 2.1. Table 6 shows the performance of our proposed PUP method on different user groups compared to DeepFM. From the results, We have the following two findings:

- Both models perform much better on consistent users than inconsistent users. The reason for the performance gap is that it is more difficult to predict users' interest when they regard price differently over distinct categories.
- Our proposed PUP achieves better results on both consistent users (about 41.76% boost) and inconsistent users (about 1.18% boost). Given the fact that capturing the preference of inconsistent users is more challenging, our proposed PUP method is still able to improve the performance due to its powerful capacity of learning high-quality representations on the constructed unified heterogeneous graph.

TABLE 6
Performance Comparison on Different User Groups on Beibei Dataset (Metrics=NDCG@50)

| user group | DeepFM | PUP | boost |
|---|---|---|---|
| consistent | 0.0091 | 0.0129 | 41.76% |
| inconsistent | 0.0085 | 0.0086 | 1.18% |

The inconsistency of price awareness across categories brings much more difficulties when incorporating price into recommendation. The performance gap between DeepFM and PUP suggests that it is not sufficient to model this inconsistency using a unified model and it requires specific design towards this inconsistency. In our two-branch structure, this inconsistency is exactly what the category branch aims to model. By combining global and local effect of the price factor, our proposed PUP method could improve the recommendation performance on users with diverse consistency of price awareness.

### 4.6 Utilizing Price to Tackle Cold-Start Problems

In e-commerce platforms, a user usually only interacts with a small number of categories compared with the total number of categories on the platform. However, recommending items of unexplored categories is rather important both for diversity and for maximizing the revenue. This task is a cold-start problem since there exists limited data records in the unexplored categories. Users' preference and price awareness are not always consistent across categories. Therefore, what we have learned on explored categories cannot be directly transferred to unexplored ones. [38] first proposed that the price could come to help in this situation.

To evaluate the performance on unexplored categories, we make a few adaptations to our datasets. We find those users who purchase items in the test set from categories that are different from those purchased categories in the training set. Then we filter out those items in the test set belonging to explored categories. We conduct experiments according to two protocols which were utilized in [38]:

- *CIR* (Category item recommendation): For this protocol, the candidate item pool is composed of all the items which belongs to the test positive unexplored categories.
- *UCIR* (Unexplored category item recommendation): For this protocol, the candidate item pool consists of all the items which are not in the train positive categories.

Suppose there are 7 categories, $\{A, B, C, D, E, F, G\}$. A user purchases items of category $A$, $B$ and $C$ in the training set and purchases items of category $E$ in the test set. Then according to CIR protocol, all the items of category $E$ form the candidate item pool. However, in UCIR problem, the candidate item pool is composed of items from unexplored categories which are $\{D, E, F, G\}$.

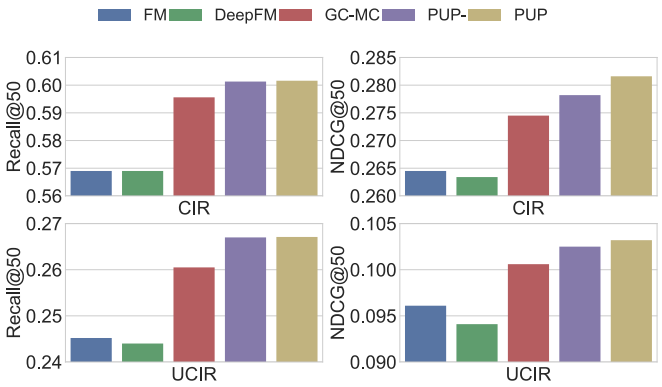



Fig. 7. Performance comparison on unexplored categories of Yelp dataset.

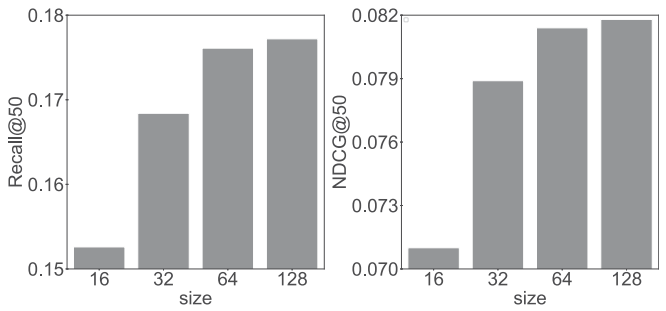



Fig. 8. Performance comparison of different embedding size on Yelp dataset.

Fig. 7 shows the performance comparison on unexplored categories. PUP- stands for a slim version of PUP with category nodes removed. The results verify the positive effect of incorporating price when recommending items of unexplored categories. Specifically, PUP and PUP- outperforms GC-MC in all cases. Without incorporating price and category, GC-MC only captures user-item interactions. And when performing neighbor aggregation, item nodes of unexplored categories can only be reached via intermediate user nodes in GC-MC, while in PUP or PUP- price nodes can also be leveraged which makes it much easier to transfer from explored categories to unexplored categories.

Moreover, the results illustrates that GCN based methods (GC-MC, PUP-, PUP) consistently outperform factorization based methods (FM, DeepFM). The performance gap indicates that a simple look-up embedding table is insufficient to learn high-quality representations under cross category protocols. With GCN, items of unexplored categories could be reached by neighbor aggregation on the graph.

Finally, our proposed PUP model achieves the best performance in both CIR and UCIR problems on two datasets. Due to the powerful graph convolutional encoder, our proposed PUP is able to capture users' preference on unexplored categories. In our contructed unified heterogeneous graph, item nodes linked to unexplored categorie nodes are high order neighbors for the user since those item nodes could be reached via price nodes and user nodes connected to them. With strong collaborative filtering effect, an item of an unexplored category could be reached within 3 hops (user-item-user-item). Moreover, if the user purchases items of enough price levels, the item could also be reached through price nodes (user-item-price-item).

## 4.7 Hyper-Parameter Study

In this section, we conduct experiments to investigate the effect of hyper-parameters in the proposed PUP method. Specifically, we study the impact of embedding size and dropout ratio, which both determine the model capacity and are highly related to overfitting.

### 4.7.1 Embedding Size

In terms of Latent Factor Models (LFM), the number of latent factors usually influences the recommendation accuracy significantly, since it directly determines the model capacity. For embedding based models, the size of embeddings is just such a crucial hyper-parameter. With larger embedding size, the model capacity gets increased, and it is easier to model much complicated interactions.

We conduct experiments of different embedding sizes on Yelp dataset, and Fig. 8 illustrates the results. From the results we can observe that when the embedding size is too small, such as 16, the recommendation performance is much worse than larger embedding sizes, because the model capacity is largely limited by insufficient embedding size. If we increase the embedding size to 32 and 64, the recommendation performance gets promoted significantly, with respect to both Recall and NDCG. However, when we further increase the embedding size to 128, the performance gain is not significant, since much large embedding size also increases the risk of overfitting, which to great extent restricts the performance.

### 4.7.2 Dropout Ratio

As introduced in Section 3.4, we use the dropout technique to prevent the PUP model from overfitting. Specifically, during model training, we randomly drop the output representation with certain probability, i.e., dropout ratio. With the help of this feature-level dropout operation, our proposed PUP model could learn robust node representations for users, items, prices and categories.

Neural models are usually sensitive to the dropout ratio, because large dropout ratio limits the model capacity, while small dropout ratio could not effectively prevent overfitting. We conduct experiments of different dropout ratios on Beibei dataset to investigate how dropout affects our proposed PUP model. Results of different dropout ratios are shown in Fig. 9. From the results, we can observe that our proposed PUP model performs well when dropout ratio is not greater than 0.5, and PUP achieves the best performance when dropout is 0.2 or 0.3, in terms of NDCG@50 and Recall@50 respectively. In addition, if we further increase the dropout ratio to 0.6 or 0.7, the performance drops drastically, since the model capacity is greatly restricted by strong dropout, which is corresponding to our analysis.

## 4.8 User Study

As discussed in previous sections, users' price awareness could only be inferred implicitly from purchasing history,

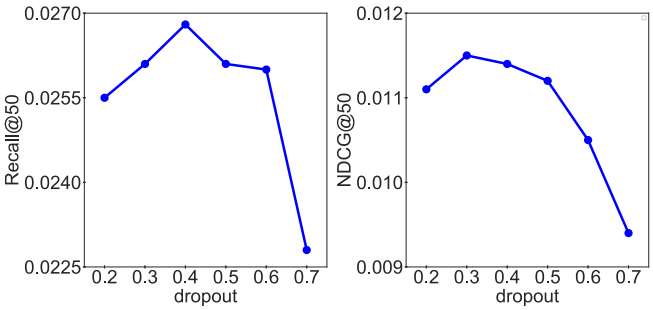



Fig. 9. Performance comparison of different dropout ratio on Beibei dataset.

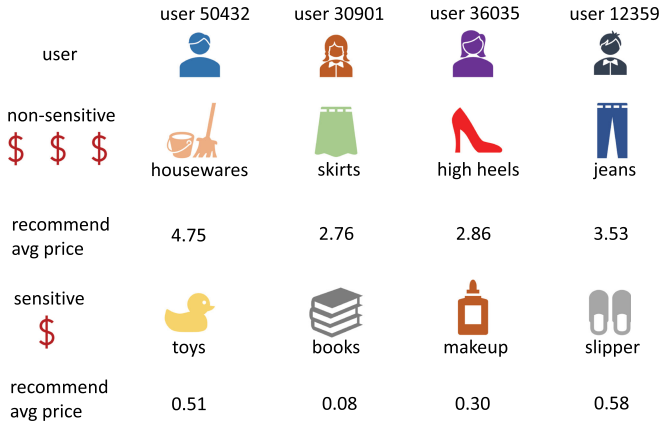



Fig. 10. Average price of recommended items from user-non-sensitive category and user-sensitive category.

and such awareness is often inconsistent across different categories. In our proposed PUP method, we construct a heterogeneous graph consisting of four types of nodes. Particularly, we push out price and category as separate nodes, instead of item features. In this way, item nodes serve as bridges, which makes it possible to propagate price awareness from price nodes and category nodes to user nodes. In addition, we adopt a two-branch design, aiming at disentangling global effect and category-dependent effect of the price factor. Through experimental studies on two real-world datasets, we verify the effectiveness of the proposed method from a macro level. In this section, we dive deeper into particular users, and perform user study to confirm whether users' category-dependent price awareness is well captured by the proposed model.

A user might purchase many expensive products of one category, while also being very sensitive on price in another category. It is significantly important to recommend items with the proper price of the two extreme categories. Therefore, we first explore each user's purchase history, and find out the two extreme purchased categories with the highest and lowest price respectively. Formally, we define the category where the user purchases the highest price level as *user-non-sensitive* category, and the category where user purchases the lowest price level as *user-sensitive* category. Then we investigate how the proposed PUP method performs when recommending items from the two extreme categories. Specifically, we calculate the average price of the recommended top items of the two extreme categories. What we are interested in is whether PUP recommends expensive items of the *user-non-sensitive* category, and recommends cheap items of the *user-sensitive* category.

We conduct experiments on Beibei dataset. We first calculate the average price gap between recommended items of *user-non-sensitive* category and *user-sensitive* category. Results show that the proposed PUP method indeed recommend more expensive items from *user-non-sensitive* category than *user-sensitive* category. To be specific, the price for recommended items from *user-non-sensitive* category is significantly higher than *user-sensitive* category, with a mean price gap as 1.66 price level.

In addition, we take out several users, who are inconsistent with respect to price awareness across different categories. Fig. 10 illustrates the recommended results for four specific users. For example, user 36,035 is highly tolerant on the price of high heels, but usually purchases cheap makeups. With special design on price awareness modeling, the proposed PUP model successfully learns to recommend more expensive high heels with average price level 2.86 than makeups with average price level 0.30. Similarly, user 50,432 tends to buy expensive housewares, while he is sensitive to the price of toys. Our proposed PUP is also capable of providing housewares and toys with proper price, whose average price levels are 4.75 and 0.51 respectively. By investigating the specific recommended items of these typical users who are inconsistent with respect to price awareness across different categories, we could observe that the category-dependent price awareness is well captured by the proposed PUP model. And the two main challenges of incorporating price into recommendation are successfully addressed by the proposed method.

To summarize, we conduct extensive experiments on real-world datasets, results illustrate that incorporating price could improve the recommendation accuracy and our proposed PUP method outperforms other baselines on price-aware recommendation. We perform ablation study on price factor to confirm the critical role of price in e-commerce recommendation. Experiments on the quantization and fineness of the price factor illustrate the capability of our proposed method in capturing price preference. Our two-branch design disentangles the global and local effect of the price factor and serves great recommendation performance. Moreover, we investigate the consistency of price awareness across multiple categories. Furthermore, our proposed PUP shows superior performance when tackling cold-start problems. In our proposed PUP method, several crucial hyper-parameters greatly influence the model performance, so we further conduct experiments to investigate how the proposed PUP model performs under different settings of hyper-parameters. At last, we conduct user study to verify the effectiveness of capturing price awareness for specific users.

## 5 Related Work

*Price-Aware Recommendation.* In e-commerce systems, price serves as a significant role in researches focusing on the task of revenue maximization. Nevertheless, price is seldom

specifically utilized in improving recommendation. Schafer *et al.* [43] first pointed out the potential of including price in e-commerce recommender systems. Wang *et al.* [42] proposed $\text{SVD}_{\text{util}}$ which leveraged price to recommend items with higher net utility. Recently, several studies [38], [41], [44], [45], [46] started to approach price-aware recommendation, of which the aim is to better estimate users' preference with the help of products' price information. Asnat *et al.* [44] leveraged a Gaussian distribution to model users' *acceptable price*, and when predicting user-item interactions, a user-price matching score calculated from the distribution is multiplied to the traditional user-item score to get the final predictions. A major concern is that directly matching user and price ignores the category-dependent influence of the price factor. Wan *et al.* [41] incorporate dynamic pricing of item and price elasticity (how the change of price affects user behaviors) into recommender systems via modeling users' decision making into three stages: category purchase, product choice and purchase quantity. However, in most real scenarios, decision making is always compressed to one stage because a user is directly exposed to item candidates selected by recommendation engines. Chen *et al.* [38] applied collective matrix factorization (CMF) [40] and proposed a model called PaDQ to factorize user-item, user-pattern and item-price matrices simultaneously. Jannach *et al.* [46] use numerical simulation to maximize business benefits for service providers. Umberto [45] explored the role of price in e-commerce recommendation and utilized price to improve business profit based on a multi-dimensional collaborative filtering approach [47]. In summary, these methods are not able to address the two main difficulties in integrating item price into recommender systems which have been discussed in previous sections.

*Attribute-Aware Recommendation.* Attribute-aware recommendation leverages user profiles or item descriptions to enhance recommendation [48]. Rendle *et al.* [14] proposed Factorization Machines (FM) which estimates preferences with weighted sum of pairwise interactions between every two attributes. Juan *et al.* [49] further extended FM by adding field information which could distinguish among different feature interactions. Cheng *et al.* [50] proposed a model named Wide&Deep, which is the first work to utilize neural networks to capture various attributes in recommendation. He *et al.* [17] extended FM via replacing weighted sum with a Bi-Interaction layer and multi-layer perception. Guo *et al.* [15] extended FM with a deep neural network to model high order feature interactions. Lian *et al.* [16] combined compressed interaction network with multi-layer perceptions to learn both low- and high-order feature interactions. However, these models are relatively general models and not specifically designed for price-awareness modeling. Although these models can be adapted to the task of price-aware recommendation by regarding price as an attribute, some intrinsic characteristics such as category-dependent price awareness could not be captured.

## 6 Conclusion and Future Work

In this work, we highlight the significance of incorporating price into recommendation. To address the two difficulties of incorporating price which are unstated price awareness and category-dependent influence, we propose a GCN-based method named PUP and adopt a two-branch structure which is specifically designed to disentangle the global and local effect of the price awareness. We conduct extensive experiments on real-world datasets, demonstrating that our proposed PUP could improve the recommendation performance over existing methods. Further insights are provided on alleviating the cold start issue via capturing price awareness. Hyper-parameter study and user study are also conducted to investigate the performance of the proposed PUP on different hyper-parameters and for specific users.

Although specifically designed for modeling price sensitivity, our proposed model serves great generality with respect to feature engineering where other features can be easily integrated into our proposed method. For example, user profiles can be added as separate nodes linked to user nodes, while item features other than price and category can be integrated similarly. Empirical studies are needed to investigate the performance of PUP when further incorporating large scale features. As there are increasing research focusing on the price factor from the perspective of service providers, how to utilize PUP to maximize the revenue is an interesting and important research question which extends price-aware recommendation to value-aware recommendation.

## References


[1] C. Eksombatchai *et al.*, "Pixie: A system for recommending 3+ billion items to 200+ million users in real-time," in *Proc. World Wide Web Conf.*, 2018, pp. 1775–1784.

[2] G. Zhou *et al.*, "Deep interest network for click-through rate prediction," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 1059–1068.

[3] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *Proc. 10th ACM Conf. Recommender Syst.*, 2016, pp. 191–198.

[4] P. Sun, L. Wu, K. Zhang, Y. Fu, R. Hong, and M. Wang, "Dual learning for explainable recommendation: Towards unifying user preference prediction and review generation," in *Proc. Web Conf.*, 2020, pp. 837–847.

[5] S. Kabbur, X. Ning, and G. Karypis, "FISM: Factored item similarity models for top-n recommender systems," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2013, pp. 659–667.

[6] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme , "BPR: Bayesian personalized ranking from implicit feedback," in *Proc. 25th Conf. Uncertainty Artif. Intell.*, 2009, pp. 452–461.

[7] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 173–182.

[8] M. Wang and J. McAuley, "One-class recommendation with asymmetric textual feedback," in *Proc. IEEE Int. Conf. Data Mining*, 2018, pp. 648–656.

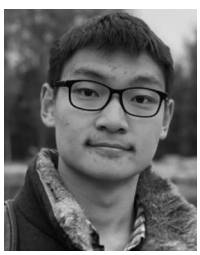
[9] L. Zheng, V. Noroozi, and P. S. Yu, "Joint deep modeling of users and items using reviews for recommendation," in *Proc. 10th ACM Int. Conf. Web Search Data Mining*, 2017, pp. 425–434.

[10] X. Wang, X. He, L. Nie, and T.-S. Chua, "Item silk road: Recommending items from information domains to social users," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2017, pp. 185–194.

[11] J. Chen, H. Zhang, X. He, L. Nie, W. Liu, and T.-S. Chua, "Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2017, pp. 335–344.

[12] P. Sun, L. Wu, and M. Wang, "Attentive recurrent social recommendation," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2018, pp. 185–194.

[13] Z. Cheng, X. Chang, L. Zhu, R. C. Kanjirathinkal, and M. Kankanhalli, "MMALFM: Explainable recommendation by leveraging reviews and images," *ACM Trans. Inf. Syst.*, vol. 37, no. 2, 2019, Art. no. 16.
[14] S. Rendle, "Factorization machines," in *Proc. IEEE Int. Conf. Data Mining*, 2010, pp. 995–1000.
[15] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "DeepFM: A factorization-machine based neural network for CTR prediction," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, 2017, pp. 1725–1731.
[16] J. Lian, X. Zhou, F. Zhang, Z. Chen, X. Xie, and G. Sun, "xDeepFM: Combining explicit and implicit feature interactions for recommender systems," *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 1754–1763.
[17] X. He and T.-S. Chua, "Neural factorization machines for sparse predictive analytics," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2017, pp. 355–364.
[18] S.-F. S. Chen, K. B. Monroe, and Y.-C. Lou, "The effects of framing price promotion messages on consumers' perceptions and purchase intentions," *J. Retailing*, vol. 74, no. 3, pp. 353–372, 1998.
[19] L. Krishnamurthi, T. Mazumdar, and S. Raj, "Asymmetric response to price in consumer brand choice and purchase quantity decisions," *J. Consum. Res.*, vol. 19, no. 3, pp. 387–400, 1992.
[20] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *Proc. 42nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2019, pp. 165–174.
[21] Y. Zheng, C. Gao, X. He, Y. Li, and D. Jin, "Price-aware recommendation with graph convolutional networks," 2020, *arXiv:2003.03975*.
[22] J. K. Horowitz and K. E. McConnell , "A review of WTA/WTP studies," *J. Environ. Econ. Manage.*, vol. 44, no. 3, pp. 426–447, 2002.
[23] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.
[24] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. 26th Int. Conf. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.
[25] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *Proc. IEEE Int. Conf. Data Mining*, 2008, pp. 263–272.
[26] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," in *Proc. 20th Int. Conf. Neural Inf. Process. Syst.*, 2008, pp. 1257–1264.
[27] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1024–1034.
[28] R. van den Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," *Stat*, vol. 1050, 2017, Art. no. 7.
[29] M. Wang, W. Fu, S. Hao, H. Liu, and X. Wu, "Learning on big graph: Label inference and regularization with anchor hierarchy," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 5, pp. 1101–1114, May 2017.
[30] F. Wu, T. Zhang, A. H. de Souza, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," 2019, *arXiv:1902.07153*.
[31] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 974–983.
[32] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," 2018, *arXiv:1810.00826*.
[33] K. Tu, P. Cui, X. Wang, P. S. Yu, and W. Zhu, "Deep recursive network embedding with regular equivalence," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 2357–2366.
[34] A. Bojchevski and S. Günnemann, "Deep Gaussian embedding of graphs: Unsupervised inductive learning via ranking," 2017, *arXiv:1707.03815*.
[35] R. Pasricha and J. McAuley, "Translation-based factorization machines for sequential recommendation," in *Proc. 12th ACM Conf. Recommender Syst.*, 2018, pp. 63–71.
[36] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma, "Collaborative knowledge base embedding for recommender systems," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 353–362.
[37] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.
[38] J. Chen *et al.*, "Does product recommendation meet its waterloo in unexplored categories?: No, price comes to help," in *Proc. 37th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2014, pp. 667–676.
[39] R. He and J. McAuley, "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering," in *Proc. 25th Int. Conf. World Wide Web*, 2016, pp. 507–517.
[40] A. P. Singh and G. J. Gordon, "Relational learning via collective matrix factorization," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2008, pp. 650–658.
[41] M. Wan *et al.*, "Modeling consumer preferences and price sensitivities from large-scale grocery shopping transaction logs," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 1103–1112.
[42] J. Wang and Y. Zhang, "Utilizing marginal net utility for recommendation in e-commerce," in *Proc. 34th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2011, pp. 1003–1012.
[43] J. B. Schafer, J. Konstan, and J. Riedl, "Recommender systems in e-commerce," in *Proc. 1st ACM Conf. Electron. Commerce*, 1999, pp. 158–166.
[44] A. Greenstein-Messica and L. Rokach, "Personal price aware multi-seller recommender system: Evidence from eBay," *Knowl.-Based Syst.*, vol. 150, pp. 14–26, 2018.
[45] P. Umberto, "Developing a price-sensitive recommender system to improve accuracy and business performance of ecommerce applications," *Int. J. Electron. Commerce Stud.*, vol. 6, no. 1, pp. 1–18, 2015.
[46] D. Jannach and G. Adomavicius, "Price and profit awareness in recommender systems," 2017, *arXiv:1707.08029*.
[47] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin, "Incorporating contextual information in recommender systems using a multidimensional approach," *ACM Trans. Inf. Syst.*, vol. 23, no. 1, pp. 103–145, 2005.
[48] M. J. Pazzani and D. Billsus, "Content-based recommendation systems," in *The Adaptive Web*. Berlin, Germany: Springer, 2007, pp. 325–341.
[49] Y. Juan, Y. Zhuang, W.-S. Chin, and C.-J. Lin, "Field-aware factorization machines for CTR prediction," in *Proc. 10th ACM Conf. Recommender Syst.*, 2016, pp. 43–50.
[50] H.-T. Cheng *et al.*, "Wide & deep learning for recommender systems," in *Proc. 1st Workshop Deep Learn. Recommender Syst.*, 2016, pp. 7–10.

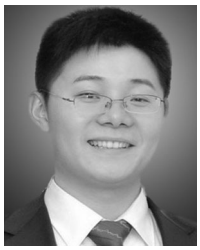


**Yu Zheng** received the BS degree in electronic engineering in 2019 from Tsinghua University, Beijing, China, where he is currently working toward the PhD degree with the Department of Electronic Engineering. His reserach interests include recommender systems, data mining, and digital marketing.

**Chen Gao** received the BS degree in electronic engineering in 2016 from Tsinghua University, Beijing, China, where he is currently working toward the PhD degree with the Department of Electronic Engineering. He has publications appeared in several top conferences, such as WWW, ICDE, KDD, and SIGIR. His research interests include recommender systems, data mining, and data privacy.

**Xiangnan He** received the PhD degree in computer science from the National University of Singapore. He is currently a research fellow with the School of Computing, National University of Singapore. His research interests include recommender system, information retrieval, natural language processing, and multi-media. His work on recommender system was the recipient of the Best Paper Award Honorable Mention in WWW 2018 and SIGIR 2016. He is/was the PC member for top-tier conferences, including SIGIR, WWW, MM, KDD, WSDM, CIKM, AAAI, and ACL, and the invited reviewer for prestigious journals, including TKDE, TOIS, TKDD, TMM, and WWWJ.

**Depeng Jin** (Member, IEEE) received the BS and PhD degrees in electronics engineering from Tsinghua University, Beijing, China, in 1995 and 1999, respectively. He is currently an associate professor with Tsinghua University and the vice chair of the Department of Electronic Engineering. His research interests include telecommunications, high-speed networks, ASIC design, and future internet architecture. He was awarded the National Scientific and Technological Innovation Prize (Second Class) in 2002.



**Yong Li** (Senior Member, IEEE) received the BS degree in electronics and information engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2007, and the PhD degree in electronic engineering from Tsinghua University, Beijing, China, in 2012. He is currently a faculty member of the Department of Electronic Engineering, Tsinghua University. He is/was the general chair, TPC chair, SPC/TPC member for several international workshops and conferences, and he is on the editorial board of two IEEE journals. His papers have total citations more than 6900. Among them, ten are ESI Highly Cited Papers in computer science, and was the recipient of four conference best paper (run-up) awards. He was the recipient of the IEEE 2016 ComSoc Asia-Pacific Outstanding Young Researchers, Young Talent Program of China Association for Science and Technology, and National Youth Talent Support Program.