Graph Convolution Machine for Context-aware Recommender System

Jiancan Wu[1] , **Xiangnan He**[1] , **Xiang Wang**[2] , **Qifan Wang**[3] , **Weijian Chen**[1] , **Jianxun Lian**[4] , **Xing Xie**[4] , **Yongdong Zhang**[1]
[1]University of Science and Technology of China
[2]National University of Singapore
[3]Google Research
[4]Microsoft Research Asia
{wjc1994, naure}@mail.ustc.edu.cn, {hexn, zhyd73}@ustc.edu.cn, xiangwang@u.nus.edu, wqfcr@google.com, jianxun.lian@outlook.com, xing.xie@microsoft.com

Abstract

The latest advance in recommendation shows that better user and item representations can be learned via performing graph convolutions on the user-item interaction graph. However, such finding is mostly restricted to the collaborative filtering (CF) scenario, where the interaction contexts are not available. In this work, we extend the advantages of graph convolutions to context-aware recommender system (CARS, which represents a generic type of models that can handle various side information). We propose *Graph Convolution Machine* (GCM), an end-to-end framework that consists of three components: an encoder, graph convolution (GC) layers, and a decoder. The encoder projects users, items, and contexts into embedding vectors, which are passed to the GC layers that refine user and item embeddings with context-aware graph convolutions on user-item graph. The decoder digests the refined embeddings to output the prediction score by considering the interactions among user, item, and context embeddings. We conduct experiments on three real-world datasets from Yelp, validating the effectiveness of GCM and the benefits of performing graph convolutions for CARS.

1 Introduction

Recommendation has become a pervasive service in today's Web, serving as an important tool to alleviate information overload and improve user experience. The key data source for building a recommendation service is user-item interactions, *e.g.,* clicks and purchases, which spawn wide research efforts on collaborative filtering (CF) [Rendle *et al.*, 2009; He *et al.*, 2017; Wang *et al.*, 2019] that leverage the interaction data only to predict user preference. Recently, inspired by the success of graph neural networks (GNNs) [Kipf and Welling, 2017; Veličković *et al.*, 2018], researchers have attempted to employ GNNs on recommendation in which CF signals are exhibited as high-order connectivity [Wang *et al.*, 2019; Zheng *et al.*, 2018; Wei *et al.*, 2019; Wu *et al.*, 2019b]. While CF provides a universal solution for recommendation, it falls short in utilizing the side information of interaction contexts. In many scenarios, the current contexts could have a strong impact on user choice. For example, in restaurant recommendation, the current time and location can effectively filter out unsuitable candidates; in E-commerce, the click behaviors in recent sessions provide strong signal on user next purchase. As such, it is important to develop context-aware recommender system (CARS) that can effectively integrate contexts (and possibly other side information like user profiles and item attributes) into user preference prediction [Shi *et al.*, 2014].

Inspired by the matrix completion view of CF, early research naturally extended the problem of CARS to tensor completion [Karatzoglou *et al.*, 2010], which however suffers from high complexity. Later on, Rendle proposed factorization machine (FM) [Rendle, 2010], which to the first time addressed CARS from the view of standard supervised learning. Specifically, it converts all information related to an interaction to a feature vector via multi-hot encoding, modeling the second-order feature interactions to predict the interaction label. Due to its generality and effectiveness, FM soon becomes a prevalent solution for CARS and is followed by many work. For example, in the era of deep learning, Wide&Deep [Cheng *et al.*, 2016] and Deep Crossing [Shan *et al.*, 2016] replaced the second-order interaction modeling with a neural network for implicit interaction modeling; recently, Neural FM [He and Chua, 2017], Attentional FM [Xiao *et al.*, 2017], xDeepFM [Lian *et al.*, 2018], and Convolutional FM [Xin *et al.*, 2019] extended FM with various kinds of neural networks to enhance its expressiveness.

Summarizing existing CARS models, we can find a common drawback: they follow the standard supervised learning scheme that ignores the relationship among data instances. This may limit the model's effectiveness in capturing the CF effect, since it needs to consider multiple interactions simultaneously to recognize the CF patterns. An evidence is from the neural graph collaborative filtering (NGCF) work [Wang *et al.*, 2019], which demonstrates that connecting the interactions in the predictive model significantly improves the embedding quality for CF. Since in CARS user-item interactions still play an important role by reflecting user preference, it is reasonable to believe that properly modeling the relationship among interactions can improve the model qual-

ity. Moreover, the recent neural network-based methods like xDeepFM [Lian *et al.*, 2018] and Convolutional FM [Xin *et al.*, 2019] suffer from low efficiency in online serving, since each candidate item needs be scored separately with the deep model architecture that models complex feature interactions, which could be very time-consuming.

In this work, we aim to propose new CARS model by addressing the above-mentioned limitations. Firstly, we cast the data in CARS as an attributed user-item graph, where the side information of users and items are represented as node features, and the contexts are represented as edge features (Figure 1). Secondly, we propose an end-to-end model that consists of three components: an encoder, graph convolution (GC) layers, and a decoder (Figure 2). The encoder projects users, items, and contexts into embedding vectors; the GC layers then exploit the interactions to refine the embeddings via performing graph convolutions; lastly, the decoder models the interactions among embeddings via FM to output the prediction score. After the model is trained, the refined embeddings by GC layers can be pre-computed before serving. As such, the time complexity of online serving is the same as FM, being much more efficient than the recent neural network methods.

We summarize the contributions of this work as follows:

- We highlight the limitation of the mainstream supervised learning schemes and the necessity of exploiting the relationship among data instances in the predictive model of CARS.
- We propose a new model named Graph Convolution Machine (GCM), unifying the strengths of graph convolution network and factorization machine for CARS.
- We conduct extensive experiments on three real-world datasets which demonstrate the effectiveness and efficiency of GCM.

2 Problem Definition

We divide the data used for CARS into four types: users, items, contexts, and interactions. Following [Rendle *et al.*, 2011], we define context as the information that is associated with an interaction, *e.g.,* the current location, time, previous click, etc. Figure 1 illustrates the data in CARS, where the main data is the user-item-context interaction tensor. In the sparse tensor, each nonzero entry (u, i, c) denotes that the user u has interacted with the item i under the context c; we give such entries a label of 1, *i.e.,* $y_{uic} = 1$. Each u, i, c is respectively associated with a multi-hot feature vector $\mathbf{u}$, $\mathbf{i}$, and $\mathbf{c}$, which contain the features that describe the user, item, and context. For example, $\mathbf{u}$ includes static user profiles like gender and interested tags, $\mathbf{i}$ includes static item attributes like category and price, and $\mathbf{c}$ includes dynamic contexts like the current location of the user and the time.

Given such data, we convert it to the form of attributed user-item bipartite graph that has the same representation power. Specifically, each vertex represents a user or an item, and each edge represents the interaction between the connected user and item. Each vertex or edge is associated with a feature vector $\mathbf{u}$, $\mathbf{i}$, or $\mathbf{c}$. Note that there may exist multiple edges between a user-item pair, since a user may interact with the same item multiple times under different contexts. We denote all edges in the graph as the set $\mathcal{Y} = \{(u, i, c)|y_{uic} = 1\}$, the neighbors of the user u as the set $\mathcal{N}_u = \{(i, c)|y_{uic} = 1\}$, and neighbors of the item i as the set $\mathcal{N}_i = \{(u, c)|y_{uic} = 1\}$.

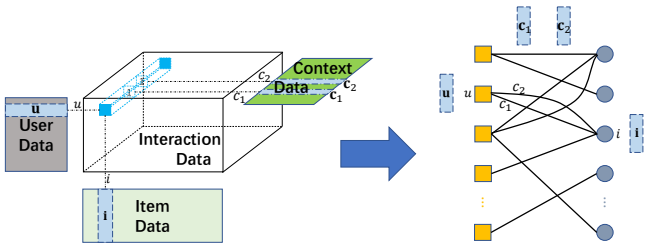


Figure 1: The data used for building a CARS. The mixture data of interaction tensor and user/item/context feature matrices are converted to an attributed user-item bipartite graph without loss of fidelity.

We formulate the problem of CARS as:

Input: User-item-context interactions $\{(u, i, c)|y_{uic} = 1\}$, feature vectors of users $\{\mathbf{u}\}$, items $\{\mathbf{i}\}$, and contexts $\{\mathbf{c}\}$.

Output: Prediction function $f : \mathbf{u}, \mathbf{i}, \mathbf{c} \rightarrow \mathbb{R}$, which takes the feature vector of a user, an item, and a context as the input, and outputs a real value that estimates how likely the user will interact with the item under the context.

3 Graph Convolution Machine (GCM)

We present our method in this section. We first describe the predictive model, followed by the model complexity analyses and optimization details.

3.1 Predictive Model

Figure 2 illustrates the model framework, which consists of three components: an encoder, graph convolution layers, and a decoder. We next describe each component one by one.

Encoder

The input to the encoder has three fields: user-field features $\mathbf{u}$, item-field features $\mathbf{i}$, and the context-field features $\mathbf{c}$. We include the ID feature into the user-field and item-field features, since it helps to differentiate users (items) when their profiles (attributes) are the same[1]. For each nonzero feature, we associate it with an embedding vector, resulting in a set of embeddings to describe the input user, item, and context, respectively. We then pool the set of user (and item) field into a vector, so as to feed the vector into the the following GC layers to refine the user (and item) representations. Specifically, we adopt average pooling, that is,

$$\mathbf{p}_u^{(0)} = \frac{1}{|\mathbf{u}|}\mathbf{P}^T\mathbf{u}, \tag{1}$$

where $|\mathbf{u}|$ denotes the number of nonzero features in $\mathbf{u}$, and $\mathbf{P} \in \mathbb{R}^{U \times D}$ is the embedding matrix for user features, where U denotes the number of total user features and D denotes the embedding size. $\mathbf{p}_u^{(0)}$ denotes the initial representation vector

[1]Note that there is no need to include ID into the context-field features, since a context c and its features $\mathbf{c}$ are one-to-one mapping.

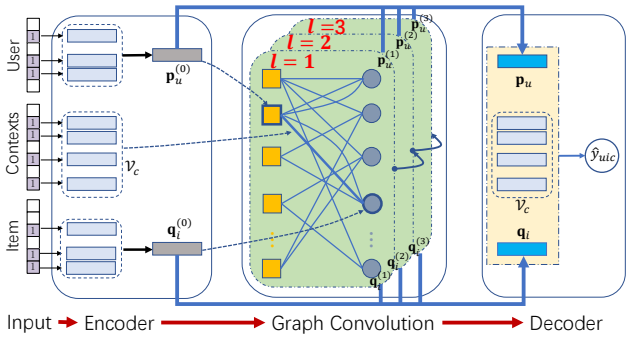


Figure 2: The Graph Convolution Machine model.

for u. Similarly, we get the initial representation vector for item i as $\mathbf{q}_i^{(0)}$.

Note that other pooling mechanisms can be applied here, such as the attention-based pooling [Mei *et al.*, 2018; Xin *et al.*, 2019; Wu *et al.*, 2019a] which learns varying weights for feature embeddings. However, we tried that and find it does not improve the performance. Thus we keep the simplest average pooling and avoid introducing additional parameters. Since we do not update the context representation in the following GC layers, we do not perform pooling on the context field. We denote the set of context-field embeddings as $\mathcal{V}_c = \{\mathbf{v}_s | s \in \mathbf{c}\}$, where $s \in \mathbf{c}$ denotes the nonzero feature in $\mathbf{c}$ and $\mathbf{v}_s$ denotes the embedding vector for context feature s. The encoder outputs $\mathbf{p}_u^{(0)}$, $\mathbf{q}_i^{(0)}$, and $\mathcal{V}_c$, which are fed into the next component of GC layers.

Graph Convolution Layers

This is the core component of GCM, designed to address the limitation of existing supervised learning-based CARS models. It refines $\mathbf{p}_u^0$ and $\mathbf{q}_i^0$ by exploiting holistic user-item interaction data, which can augment the user and item representations with explicit collaborative filtering signal [Wang *et al.*, 2019]. The GC on user-item graph is typically formulated as a message propagation framework:

$$\mathbf{p}_u^{(l+1)} = \sum_{i \in \mathcal{N}_u} g(\mathbf{p}_u^{(l)}, \mathbf{q}_i^{(l)}); \ \mathbf{q}_i^{(l+1)} = \sum_{u \in \mathcal{N}_i} g(\mathbf{q}_i^{(l)}, \mathbf{p}_u^{(l)}), \tag{2}$$

where $\mathbf{p}_u^{(l)}$ and $\mathbf{q}_i^{(l)}$ denote the refined user representation and item representation of the l-th GC layer, respectively, and $g(\cdot)$ is a self-defined function. Recursively conducting such message propagation relates the representation of a user with her high-order neighbors, *e.g.,* first-order for interacted items and second-order for co-interacted users, which is beneficial for collaborative filtering; and the same logic applies to item representation.

However, the standard GC does not consider the features on edges. In our constructed user-item graph, the edges between a user and an item carry the context features, which are important to understand the context-dependent interaction patterns. For example, a user may prefer bars on Friday, and a restaurant is more popular on lunch time. As such, better user and item representations can be obtained if the context features can be properly integrated into the GC.

To this end, we propose a new GC operation that incorporates the edge features of contexts:

$$\begin{aligned} \mathbf{p}_u^{(l+1)} &= \sum_{(i,c) \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}} \left(\mathbf{q}_i^{(l)} + \frac{1}{|\mathcal{V}_c|} \sum_{\mathbf{v}_s \in \mathcal{V}_c} \mathbf{v}_s\right), \\ \mathbf{q}_i^{(l+1)} &= \sum_{(u,c) \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|}} \left(\mathbf{p}_u^{(l)} + \frac{1}{|\mathcal{V}_c|} \sum_{\mathbf{v}_s \in \mathcal{V}_c} \mathbf{v}_s\right). \end{aligned} \tag{3}$$

Next we explain the rationality of the GC of the user side, since the item side can be interpreted in the same way. Here $|\mathcal{N}_u|$ denotes the number of edges connected with the user u, and the coefficient $\frac{1}{\sqrt{|\mathcal{N}_u|}}$ is a normalization term to avoid the scale of embedding values increasing with the GC. We incorporate the context features by averaging their embeddings and adding to the connected user embedding. Through this way, we build the connection between a user with both her interacted item and the interacted context. It is expected to capture the effect that if a user likes to choose an item under a certain context, then the similarity among their representations is similar. Note that we have tried more complicated mechanisms like incorporating the pairwise interactions among $\mathcal{V}_c$ and $\mathbf{q}_i^{(l)}$, and using a MLP to capture high-order interactions. However these ways do not lead to performance improvements. Thus we use this simple average operation, which is easy to interpret and train (no additional parameters are introduced).

By stacking multiple such GC layers, a user (or an item) representation can be refined by its multi-hop neighbors. Since the representation of different layers carry different semantics, we next combine the representations of all layers to form a more comprehensive representation:

$$\mathbf{p}_u = \sum_{l=0}^{L} \alpha_l \mathbf{p}_u^{(l)}; \quad \mathbf{q}_i = \sum_{l=0}^{L} \alpha_l \mathbf{q}_i^{(l)}, \tag{4}$$

where α_l denotes the weight of the l-th layer representation. We treat α_l as hyper-parameters, tuning them via grid search with the constraint that $\alpha_l \geq 0$ and $\sum_{l=0}^{L} \alpha_l = 1$. A possible extension is to learn α_l, *e.g.,* designing attention mechanism or optimizing them on the validation data. We leave this extension as future work, since it is not the focus of this work.

Decoder

The GC layers output refined representation of user $\mathbf{p}_u$ and item $\mathbf{q}_i$, and keep the embeddings of context features unchanged. The role of the decoder is to output the prediction score by taking in the representations. The standard choice of decoder is multi-layer perceptron (MLP), which however falls short here since it only models feature interactions in an implicit way. In CARS, explicitly modeling the interactions between features is known to be important for user preference estimation [He and Chua, 2017]. For example, the classic factorization machine (FM) models the pairwise interactions between feature embeddings and has long been a competitive model for CARS.

Inspired by the simplicity (linear model) and the effectiveness of FM, we adopt it as the decoder of GCM. The idea is

to explicitly model the pairwise interactions between the (refined) representations of user, item, and contexts with inner product. Specifically, let the set of vectors $\mathcal{V}$ be $\mathcal{V}_c \cup \mathbf{p}_u \cup \mathbf{q}_i$, the decoder outputs the prediction score as:

$$\hat{y}_{uic} = \frac{1}{2}\Big(\sum_{\mathbf{v}_s \in \mathcal{V}} \sum_{\mathbf{v}_t \in \mathcal{V}} \mathbf{v}_s^T \mathbf{v}_t - \sum_{\mathbf{v}_s \in \mathcal{V}} \mathbf{v}_s^T \mathbf{v}_s \Big). \tag{5}$$

Here the self-interactions $\mathbf{v}_s^T \mathbf{v}_s$ are excluded since they are useless for the prediction. The bias terms for each user, item, and context feature are omitted for clarity.

Note that our FM-based decoder slightly differs from the vanilla FM, which models the interactions between the embeddings of all input features. Here we project each user (item) into a vector, rather than retaining the embeddings of her (its) features. An advantage is that this way abandons the internal interactions of user-field (item-field) features, shedding more light on the interactions between user (item) and context features, which is as expected.

3.2 Model Complexity Analyses

We analyze the complexity of GCM from two aspects: the number of trainable parameters and the time complexity.

All trainable parameters come from the encoder layer, *i.e.,* the embeddings of input features, since the GC layers and the decoder layer introduce no parameters to train. Let the feature number for the user field, item field, and context field as U, I, and C, respectively, and the embedding size be D. Then the embedding layer costs $(U+I+C) \times D$ parameters. This demonstrates the low model complexity of GCM, since the number of trainable parameters is the same as FM — the most simple embedding-based CARS model.

For model training, since the complexity of the encoder plus the decoder is the same as that of FM, we analyze the additional time complexity caused by the GC layers. We implement the training in the batch-wise matrix form, which is omitted due to space limitation. Assume a batch contains all interactions. Then performing one GC layer takes time $O((|\mathcal{Y}| + M + N)D)$, where M and N denote the number of users and items, respectively. This complexity increases linearly with the number of GC layers.

After the model is trained, we perform one pass of GC layers to obtain the refined representations of all users and items, which can be done offline before online serving. As such, during online serving, we only need to execute the decoder, which has the same time complexity of FM. This is much faster than the recently emerging deep neural network-based CARS models like xDeepFM [Lian *et al.*, 2018] and Convolutional FM [Xin *et al.*, 2019]. Table 1 shows the model inference time of evaluating 1000 Yelp-OH users in which each interaction has 10 nonzero features of embedding size 64. The testing platform is GeFore GTX 1080Ti with 16GB memory CPU. As can be seen, GCM takes similar time as FM, being 68 and 400 times faster than xDeepFM and Convolutional FM, respectively.

3.3 Optimization

To optimize model parameters, we opt for the pointwise log loss, which is a common choice in recommender system [He *et al.*, 2017; Lian *et al.*, 2018]. In each training epoch, we randomly sample 4 non-observed interactions for each instance in $\mathcal{Y}$, forming the negative set $\mathcal{Y}^-$. Then we minimize the following objective function:

$$L = -\sum_{(u,i,c) \in \mathcal{Y}} \log \sigma(\hat{y}_{uic}) - \sum_{(u,i,c) \in \mathcal{Y}^-} \log\left(1 - \sigma(\hat{y}_{uic})\right) + \lambda \left\|\Theta\right\|_2^2 \tag{6}$$

where $\sigma(\cdot)$ is the sigmoid function, λ controls the L_2 regularization to prevent over-fitting. The optimization is done by mini-batch Adam [Kingma and Ba, 2015].

Table 1: Model inference time of evaluating 1,000 Yelp-OH users (14 million interactions and 10 nonzero features per interaction).

Model	FM	GCM	xDeepFM	Convolutional FM
Time/s	6.7	6.1*	414.2	2436.7

Table 2: Statistics of the datasets.

Dataset	Yelp-NC	Yelp-OH	Yelp-NV
#User	29,115	23,637	181,410
#Item	14,042	14,002	212,132
#Instance	268,917	212,132	1,530,967
#User Feature	24	24	24
#Item Feature	68	213	64
#Context Feature	14,926	15,047	36,514

4 Experiments

We evaluate experiments on three benchmark datasets, aiming to answer the following research questions:

- **RQ1:** Compared with the state-of-the-art models, how does GCM perform *w.r.t.* top-k recommendation?
- **RQ2:** How do different settings (*e.g.,* depth of layer, modeling of context features, design of decoder) affect GCM?

4.1 Experimental Settings

Dataset Description

To demonstrate the effectiveness of GCM, we conduct experiments on three datasets from Yelp , which records users' reviews on local businesses like bars and restaurants. In particular, we extract records happened in three different areas of USA — North Carolina, Ohio, Nevada States — to construct datasets, termed Yelp-NC, Yelp-OH, and Yelp-NV, respectively. In what follows, we briefly introduce the features of users, items, and contexts. Specifically, each user profile includes *yelping_since_year*[2] and *average_stars*, while the pre-existing features of items are composed of three attributes: *city*, *stars* and *is_open*. We treat each review record as an observed instance, and collect *city*[3], *month*, *hour*, *day_of_week* and *last_purchase* as its context feature. Moreover, the three-core setting is adopted to ensure data quality, *i.e.,* retaining users with at least three interactions. We summarize the statistics of datasets in Table 2.

[2]We only keep the *year* of the *yelping_since* field which indicates the time the user joined Yelp.

[3] The context feature *city* means which city does the interaction happen on. It is set as the city of the interacted item.

Table 3: Overall Performance Comparison.

	Yelp-NC				Yelp-OH				Yelp-NV			
	HR		NDCG		HR		NDCG		HR		NDCG	
	@10	@50	@10	@50	@10	@50	@10	@50	@10	@50	@10	@50
MF	0.0457	0.1291	0.0228	0.0406	0.0476	0.1379	0.0232	0.0427	0.0433	0.1202	0.0219	0.0385
NGCF	0.0548	0.1549	0.0281	0.0496	0.0612	0.1650	0.0307	0.0530	0.0539	0.1474	0.0276	0.0477
NFM	0.1094	0.2588	0.0578	0.0902	0.2667	0.5362	0.1480	0.2071	0.0593	0.1589	0.0311	0.0526
xDeepFM	0.0945	0.2302	0.0504	0.0797	0.2433	0.5141	0.1318	0.1913	0.0512	0.1379	0.0267	0.0454
GCM	**0.1143**	**0.2613**	**0.0616**	**0.0935**	**0.2740**	**0.5481**	**0.1519**	**0.2121**	**0.0650**	**0.1674**	**0.0343**	**0.0563**

For each user, we select the last interaction record to constitute the test set, while the remains are served as the training set. To emphasize model capability in recommending novel items for a user, we further filter the training set if the user-item pairs have appeared in the test set.

Evaluation Metrics

In the evaluation phase, for each user in the test set, we view all items that she has not consumed before as recommendation candidates. Each method outputs a ranking list over the candidates. We then adopt two widely-used protocols to evaluate the quality of ranking lists: Hit Ratio (HR) and Normalized Discounted Cumulative Gain (NDCG). In particular, HR@K measures whether the test item is in the top-K positions of the recommended list, whereas NDCG@K assigns higher scores to the top-ranked items. In our experiments, we report the results of $K = 10$ and $K = 50$.

Baselines

We compare our GCM with several methods as follows:

- **MF** [Koren *et al.*, 2009]: This exploits the user-item interactions only to learn user and item embeddings, while forgoing the context features.
- **NGCF** [Wang *et al.*, 2019]: Such model is the state-of-the-art GNN-based CF recommender, which incorporates high-order connectivity in user-item interaction graph into embeddings, while neglecting context features.
- **NFM** [He and Chua, 2017]: This model leverages a MLP to capture nonlinear and high-order interaction among user, item, and context features.
- **xDeepFM** [Lian *et al.*, 2018]: This is a recent neural FM model which combines explicit and implicit high-order feature interactions.

We are aware of a recent work [Li *et al.*, 2019] on click-through rate prediction with graph neural network, which is highly relevant with our work. It differs from GCM in graph construction — it builds a feature graph for each interaction, rather than the user-item graph. As a graph needs be built for each interaction to obtain its prediction, the method is very slow in evaluation since all recommendation candidates need be scored. As such, this method is not suitable for our all-ranking CARS evaluation, and we do not further compare with it. The Convolutional FM is not compared for the same reason (see Table 1 for model inference time).

Parameter Settings

We implement our GCM model in Tensorflow and will release our code upon acceptance. We apply the mini-batch Adam to optimize all models. A grid search is conducted for confirming optimal hyperparameters: the learning rates are tuned in $\{0.001, 0.0001\}$, and the coefficient of L_2 regularization term is searched in $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$. Moreover, we set the embedding size to 64 and use vanilla FM as the pre-trained model of NFM, xDeepFM, and GCM to speed up the model training.

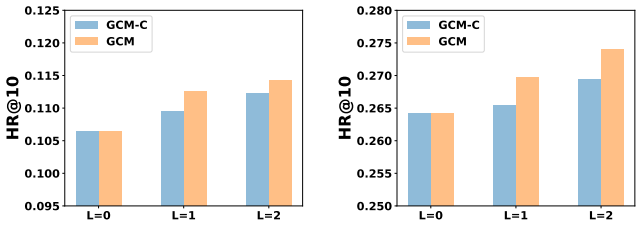
4.2 Performance Comparison (RQ1)

We report the empirical results of all models in Table 3 and have the following observations:

- Clearly, MF achieves the worst performance on three datasets, indicating that modeling user-item pairs as isolated instances limits the representation ability severely. NGCF obtains consistent improvements over MF. We attribute such improvements to the modeling of user-item connectivity. However, neither MF nor NGCF takes the context features into consideration, ignoring important factors and being insufficient for CARS.
- NFM consistently outperforms MF and NGCF across all cases, especially in Yelp-NC and Yelp-OH. This is reasonable since 1) NFM incorporates context features into the representation learning, so as to achieve better expressiveness and help to solve the data sparsity issue; and 2) NFM employs MLP on user, item, and context features to capture their nonlinear and complex interactions. This verifies that simply linear functions (*e.g.,* inner product adopted by MF and NGCF) might limit the representation learning and interaction modeling.
- Surprisingly, xDeepFM slightly underperforms NFM in most cases. One possible reason is that its complex architecture with a large number of model parameters is hard to train and easily leads to overfitting, especially in the sparser datasets, Yelp-NC and Yelp-OH.
- GCM consistently outperforms all baselines *w.r.t.* all measures. In particular, GCM achieves noticeable improvements over the strongest baselines *w.r.t.* HR@10 by 4.48%, 2.74%, and 9.61%, in Yelp-NC, Yelp-OH, and Yelp-NV, respectively. We attribute such improvements to 1) GCM employs the embedding propagation over the attributed graph to distill useful information from neighbors and improve the representation ability; and 2) Having established the refined representations, GCM further adopts FM to explicitly model the feature interactions.

4.3 Study of GCM (RQ2)

We next report ablation studies to verify the rationality of some designs in GCM, *i.e.,* analyzing the influence of model



(a) Yelp-NC (b) Yelp-OH

Figure 3: The impact of depth and context modeling in GC.

depth, context modeling, and decoder.

Impact of Model Depth

As GC is the core of GCM and stacking more GC layers is expected to augment the user and item representations with information propagated from multi-hop neighbors, we investigate how the number of GC layers affects the performance. In particular, we search the number of GC layers, L, in the range of $\{0, 1, 2\}$ and report the empirical results in Figure 3.

We use GCM-1 to represent the model with one GC layer, and similar notations for others. We have several findings:

- GCM-0 disables the embedding propagation over user-item attributed graph and downgrades to a FM-like linear model, thereby achieving poor performance. This again justifies the importance of GC layers.
- Obviously, increasing the number of GC layers consistently results in better performance across all cases. In particular, GCM-2 performs better than GCM-1. It is reasonable since the signals passing from multi-hop neighbors (*e.g.,* the second-order connectivity between behaviorally similar users or co-purchased items) are encoded into user and item representations of GCM-2, while GCM-1 only exploits personal history to enrich representations. This observation is consistent with that in NGCF [Wang *et al.*, 2019]. We also tried to stack more GC layers (*i.e.,* GCM-3), finding no improvements and over-smoothing issue. This suggests that GCM benefits from the first- and second-order neighbors most, but may suffer from degradation when higher-order neighbors are involved.

Impact of Context Modeling

One major contribution of GCM is to organize the context features as edges in the attributed user-item graph. We hence perform ablation study, to demonstrate the rationality and effectiveness of this design. In particular, we build the variant GCM-C by removing the context features from the attributed graph and keeping only the vanilla user-item interaction graph. We show the comparison between GCM and GCM-C in Figure 3 and have the following observations.

- Modeling context features as the edges endows GCM with better generalization ability. In particular, GCM is superior to GCM-C consistently. This again demonstrates the rationality of our context modeling.
- Jointly analyzing Table 3 and Figure 3, we find that GCM-C without considering contexts also outperforms other baselines. This empirically suggests that propagating embeddings over interaction graphs is of importance to generate high-quality representations.

Table 4: The variants of GCM (1 layer) with different decoders

	Yelp-NC		Yelp-OH	
	HR@10	NDCG@10	HR@10	NDCG@10
GCM	0.1126	0.0600	0.2697	0.1496
GCM-MLP	0.1050	0.0536	0.2554	0.1399
GCM-MF	0.0495	0.0247	0.0531	0.0266

Impact of Decoder

Having applied GC layers, we equip GCM with a decoder to model the pairwise interactions between the refined representations of users, items, and contexts. Here we investigate the role of such decoder. Towards this end, we compare GCM with two variants, GCM-MLP and GCM-MF, which separately replace the decoder with MLP and inner product on user and item representations. Table 4 shows the comparison of results. There are several observations:

- Clearly, modeling feature interactions in the decoder enhances the predictive results. In particular, GCM-1 and GCM-MLP perform consistently better than GCM-MF, which relies only on the inner product of user and item representations.
- While having encoded context features into user and item representations via GC layer, GCM-1 highlights their influence in an explicit fashion, while GCM-MLP models the feature interactions in a rather implicit way. The better performances of GCM-1 again verfy the rationality and effectiveness of FM-based decoder.

5 Conclusion and Future Work

In this work, we emphasize the importance of exploiting multiple interactions in CARS. Towards this end, we first convert the features of users, items, and contexts into an attributed graph involving the contexts as edges between user and item nodes. We then develop a new model, GCM, which captures the interactions among multiple user behaviors via graph neural networks, and then models the interactions among features of individual behavior via factorization machine. To demonstrate the effectiveness of GCM, we test it on three public datasets, and it shows significant improvements over the state-of-the-art CF and CARS baselines. Extensive experiments also are conducted to verify the rationality of attributed graph and offer insights into how the representations benefit from such graph learning.

Organizing user behaviors with contextual information in graphs is a promising direction to build an effective context-aware recommender. It helps build strong representations for users and items. However, GCM simply unifies all context features as an edge, neglecting dynamic characteristics of some contexts (*e.g.,* time) and hardly capturing dynamic preference of users [Beutel *et al.*, 2018]. In future, we plan to build dynamic graphs based on contextual information, instead of one static graph, and devise a dynamic graph neural network. Furthermore, rich side information is beneficial for explaining diverse intents behind user behaviors [Wang *et al.*, 2018]. We hence plan to model user-item relationships at a granular level of user intents to generate disentangled representations [Ma *et al.*, 2019].

References

[Beutel *et al.*, 2018] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed H Chi. Latent cross: Making use of context in recurrent recommender systems. In *WSDM*, pages 46–54, 2018.

[Cheng *et al.*, 2016] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *DLRS*, pages 7–10. ACM, 2016.

[He and Chua, 2017] Xiangnan He and Tat-Seng Chua. Neural factorization machines for sparse predictive analytics. In *SIGIR*, pages 355–364, 2017.

[He *et al.*, 2017] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *WWW*, pages 173–182, 2017.

[Karatzoglou *et al.*, 2010] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Recsys*, pages 79–86. ACM, 2010.

[Kingma and Ba, 2015] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[Koren *et al.*, 2009] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *IEEE, Computer Journal*, 42(8):30–37, 2009.

[Li *et al.*, 2019] Zekun Li, Zeyu Cui, Shu Wu, Xiaoyu Zhang, and Liang Wang. Fi-gnn: Modeling feature interactions via graph neural networks for ctr prediction. In *CIKM*, pages 539–548, 2019.

[Lian *et al.*, 2018] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *SIGKDD*, pages 1754–1763, 2018.

[Ma *et al.*, 2019] Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, and Wenwu Zhu. Disentangled graph convolutional networks. In *ICML*, pages 4212–4221, 2019.

[Mei *et al.*, 2018] Lei Mei, Pengjie Ren, Zhumin Chen, Liqiang Nie, Jun Ma, and Jian-Yun Nie. An attentive interaction network for context-aware recommendations. In *CIKM*, pages 157–166, 2018.

[Rendle *et al.*, 2009] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461, 2009.

[Rendle *et al.*, 2011] Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In *SIGIR*, pages 635–644, 2011.

[Rendle, 2010] Steffen Rendle. Factorization machines. In *ICDM*, pages 995–1000, 2010.

[Shan *et al.*, 2016] Ying Shan, T Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and JC Mao. Deep crossing: Web-scale modeling without manually crafted combinatorial features. In *SIGKDD*, pages 255–262, 2016.

[Shi *et al.*, 2014] Yue Shi, Martha Larson, and Alan Hanjalic. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Computing Surveys (CSUR)*, 47(1):1–45, 2014.

[Veličković *et al.*, 2018] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *ICLR*, 2018. accepted as poster.

[Wang *et al.*, 2018] Xiang Wang, Xiangnan He, Fuli Feng, Liqiang Nie, and Tat-Seng Chua. Tem: Tree-enhanced embedding model for explainable recommendation. In *WWW*, page 1543–1552, 2018.

[Wang *et al.*, 2019] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. Neural graph collaborative filtering. In *SIGIR*, pages 165–174, 2019.

[Wei *et al.*, 2019] Yinwei Wei, Xiang Wang, Liqiang Nie, Xiangnan He, Richang Hong, and Tat-Seng Chua. MMGCN: multi-modal graph convolution network for personalized recommendation of micro-video. In *MM*, pages 1437–1445, 2019.

[Wu *et al.*, 2019a] Chuhan Wu, Fangzhao Wu, Tao Qi, Suyu Ge, Yongfeng Huang, and Xing Xie. Reviews meet graphs: Enhancing user and item representations for recommendation with hierarchical attentive graph neural network. In *EMNLP-IJCNLP*, pages 4886–4895, 2019.

[Wu *et al.*, 2019b] Le Wu, Peijie Sun, Yanjie Fu, Richang Hong, Xiting Wang, and Meng Wang. A neural influence diffusion model for social recommendation. In *SIGIR*, pages 235–244, 2019.

[Xiao *et al.*, 2017] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. Attentional factorization machines: Learning the weight of feature interactions via attention networks. In *IJCAI*, pages 3119–3125, 2017.

[Xin *et al.*, 2019] Xin Xin, Bo Chen, Xiangnan He, Dong Wang, Yue Ding, and Joemon Jose. CFM: convolutional factorization machines for context-aware recommendation. In *IJCAI*, pages 3926–3932, 2019.

[Zheng *et al.*, 2018] Lei Zheng, Chun-Ta Lu, Fei Jiang, Jiawei Zhang, and Philip S Yu. Spectral collaborative filtering. In *Recsys*, pages 311–319, 2018.