Context-aware Dual Representation Learning for Complementary Products Recommendation

Da Xu, Chuanwei Ruan*
Walmart Labs
Sunnyvale, California, USA
{Da.Xu,Chuanwei.Ruan}@walmartlabs.com

Evren Korpeoglu, Sushant Kumar, Kannan Achan
Walmart Labs
Sunnyvale, California, USA
{EKorpeoglu,SKumar4,KAchan}@walmartlabs.com

ABSTRACT

Learning product representations that reflect complementary relationship plays a central role in modern recommender system for e-commerce platforms. A notable challenge is that unlike many simple relationships such as similarity, complementariness is often detected from customer purchase activities, which are highly sparse and noisy. Also, standard usage of representation learning emphasizes on only one set of embedding, which is problematic for modelling the asymmetric property of complementariness.

We propose using context-aware multi-tasking learning with dual product embedding to solve the above challenges. We encode contextual knowledge into product representation by multi-task learning, in order to alleviate the sparsity issue. By explicitly modelling with user bias terms, we take care of the noise induced by customer-specific preferences. Furthermore, we adopt the dual embedding framework to capture the intrinsic properties of complementariness and provide geometric interpretation motivated by the classic separating hyperplane theory. Finally, we propose a Bayesian network structure that unifies all the components, which also concludes several popular models as special cases.

The proposed method compares favourably to state-of-art representation learning and recommendation algorithms for e-commerce, in downstream classification and recommendation tasks. We also develop an implementation that scales efficiently to a dataset with millions of items and customers.

KEYWORDS

Representation Learning, Dual Embedding, Complementary Product, Recommender System, User Modelling

ACM Reference Format:
Da Xu, Chuanwei Ruan and Evren Korpeoglu, Sushant Kumar, Kannan Achan. 2019. Context-aware Dual Representation Learning for Complementary Products Recommendation. In *CIKM '19: ACM International Conference on Information and Knowledge Management, November 03–07, 2019, Beijing*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/1122445.1122456

*Both authors contributed equally to this research.

CIKM '19, November 03–07, 2019, Beijing

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

With tens of millions of products available on modern e-commerce platforms, recommender system aims at providing customers with efficient and personalized recommendation service through the massive volume of information. Many modern recommender systems tend to provide personalized recommendations based on customers' explicit and implicit preferences, where content-based [29] and collaborative filtering systems [14] have been widely applied to e-commerce [17], online media [27] and social network [13]. In recent years representation learning methods have quickly gained popularity in online recommendation literature. Alibaba [33] and Pinterest [35] have deployed large-scale recommender system based on their trained product embeddings. Youtube also use their trained video embeddings as part of the input to a deep neural network [6]. Different from many other use cases, e-commerce platforms often offer a vast variety of products, and nowadays customers shop online for all-around demands from electronics to daily grocery, rather than specific preferences on narrow categories of products. Therefore understanding the intrinsic relationships among products [36] while taking care of individual customer preferences motivates our work. A topic modelling approach was recently proposed to infer complements and substitutes of products as link prediction task using the extracted product relation graphs of Amazon.com [18].

For e-commerce, product complementary relationship is characterized by co-purchase patterns according to customer activity data. For example, many customers who purchase a new *TV* often purchase *HDMI cables* next and then purchase *cable adaptors*. Here *HDMI cables* are complementary to *TV*, and *cable adaptors* are complementary to *HDMI cable*. We use $\rightarrow$ as a shorthand notation for 'complement to' relationship. By this example, we motivate several properties of the complementary relationship:

- **Asymmetric**. *HDMI cable* $\rightarrow$ *TV*, but *TV* $\nrightarrow$ HDMI cable.
- **Non-transitive**. Though *HDMI cables* $\rightarrow$ *TV*, and *cable adaptors* $\rightarrow$ *HDMI cable*, but *cable adaptors* $\nrightarrow$ *TV*.
- **Transductive**. *HDMI cables* are also likely to complement other *TV*s with similar model and brand.
- **Higher-order**. Complementary products for meaningful product combos, such as (*TV*, *HDMI cable*), are often different from their individual complements.

Although the above properties for complementary relationship highlight several components for designing machine learning algorithms, other manipulations are also needed to deal with the **noise** and **sparsity** in customer purchase activity data. Firstly, the low

signal-to-noise ratio in customer purchase sequences causes difficulty in directly extracting complementary product signals from them. Most often, we only observe a few complementary patterns in customer purchase sequences or baskets. We list a customer's single day purchase as a sequence for illustration:

{*Xbox*, *games*, *T-shirt*, *toothbrush*, *pencil*, *notepad*}.

Notice that among the fifteen pairs of possible item combinations, only two paris can be recognized as complementary, i.e *games* $\rightarrow$ *Xbox*, *notepad* $\rightarrow$ *pencil*. The rest purchases are out of the customer's personal interest. As a solution to the noise issue, we introduce a customer-product interaction term to directly take account of the noises, while simultaneously models personalized preferences. We use $\mathcal{U}$ to denote the set of users (customers) and $\mathcal{I}$ to denote the set of items (products). Let $\mathbf{U} \in \mathcal{U}$ be a user categorical random variable and $\{\mathbf{I}_{t-1}, \ldots, \mathbf{I}_{t-k}\}$ be a sequence of item categorical random variables that represents k consecutive purchases before time t. If we estimate the conditional probability $p(\mathbf{I}_{t+1}|\mathbf{U}, \mathbf{I}_t, \ldots, \mathbf{I}_{t-k})$ with *softmax* classifier under score function $S(.)$, then previous arguments suggest that $S(\mathbf{I}_{t+1}, (\mathbf{U}, \mathbf{I}_t, \ldots, \mathbf{I}_{t-k}))$ should consist of a user-item preference term and a item complementary pattern term:

$$S(\mathbf{I}_{t+1}, (\mathbf{U}, \mathbf{I}_t, \ldots, \mathbf{I}_{t-k})) = f_{UI}(\mathbf{I}_{t+1}, \mathbf{U}) + f_I(\mathbf{I}_{t+1}, \mathbf{I}_t, \ldots, \mathbf{I}_{t-k}). \tag{1}$$

Now we have $f_{UI}(.)$ accounting for user-item preference (bias) and $f_I(.)$ characterizing the strength of item complementary pattern. When the complementary pattern of an purchase sequence is weak, i.e $f_I(\mathbf{I}_{t+1}, \mathbf{I}_t, \cdots, \mathbf{I}_{t-k})$ is small, the model will enlarge the user-item preference term, and vice versa.

On the other hand, the sparsity issue is more or less standard for recommender systems and various techniques have been developed for both content-based and collaborative filtering systems [12, 22]. Specifically, it has been shown that modelling with contextual information boosts performances in many cases [1, 11, 19], which also motivates us to develop our context-aware solution.

Representation learning with shallow embedding gives rises to several influential works in natural language processing (NLP) and geometric deep learning. The *skip-gram* (*SG*) and *continuous bag of words* (*CBOW*) models [20] as well as their variants including *GLOVE* [23], *fastText* [5], *Doc2vec* (*paragraph vector*) [16] have been widely applied to learn word-level and sentence-level representations. While classical node embedding methods such as *Laplacian eigenmaps* [3] and *HOPE* [21] arise from deterministic matrix factorization, recent work like *node2vec* [10] explore from the stochastic perspective using random walks. However, we point out that the word and sentence embedding models target at semantic and syntactic similarities while the node embedding models aim at topological closeness. These relationships are all symmetric and mostly transitive, as opposed to the complementary relationship. Furthermore, the **transductive** property of complementariness requires that similar products should have similar complementary products, suggesting that product similarity should also be modelled explicitly or implicitly. We approach this problem by encoding contextual knowledge to product representation such that contextual similarity is preserved.

We propose the novel idea of using context-aware dual embeddings for learning complementary products. While both sets of embedding are used to model complementariness, product similarities are implicitly represented on one of the embeddings by encoding contextual knowledge. Case studies and empirical testing results on the public *Instacart* and a proprietary e-commerce dataset show that our dual product embeddings are capable of capturing the desired properties of complementariness, and achieve cutting edge performance in classification and recommendation tasks. The customer embeddings, which are designed to learn the personal preferences, are also analyzed in downstream supervised and unsupervised learning tasks for user segmentation.

2 CONTRIBUTIONS AND RELATED WORKS

Compared to previously published works on learning product/customer representation, and modelling product relationships for e-commerce, our contributions are summarized below.

Learning higher-order product complementary relationship with customer preferences - Since product complementary patterns are mostly entangled with customer preference, we consider both factors in our work. And by directly modelling whole purchase sequences $f_I(\mathbf{I}_{t+1}, \mathbf{I}_t, \cdots, \mathbf{I}_{t-k})$, we are able to capture **higher-order** complementary relationship. The previous work of inferring complements and substitutes with topic modelling on Amazon data relies on the extracted graph of product relationships and therefore customers are not involved [18]. Alibaba proposes an approach by first constructing the weighted product co-purchase count graph from purchase records and then implementing a node embedding algorithm [33]. However, individual customer information is lost after the aggregation. The same issue occurs in item-based collaborative filtering [26] and product embedding [30]. A recent work on learning grocery complementary relationship for next basket predict models (item, item, user) triplets extracted from purchases sequences, and thus do not account for higher-order complementariness [32].

Use context-aware dual product embedding to model complementariness - Single embedding space may not capture **asymmetric** property of complementariness, especially when they are treated as projections to lower-dimensional real vector space where inner products are symmetric. Although Youtube's work takes both user preference and video co-view patterns into consideration, they do not explore complementary relationship [6]. PinSage, the graph convolutional neural network recommender system of Pinterest, focus on symmetric relations between their pins [35]. The triplet model [32] uses two sets of embedding, but contextual knowledge is not considered. To our best knowledge, we are the first to use context-aware dual embedding in modelling product representations.

Propose a Bayesian network structure to unify all components, and conclude several classic models as special cases - As we show in Table 1, the classic collaborative filtering models for product recommendation [17], sequential item models such as *item2vec* [2] and *metapath2vec* [7] for learning product representation, the recent *triplet2vec* model [32] as well as the *prod2vec* [9] model which considers both purchase sequence and user, can all be viewed as special cases within the Bayesian network representation of our model.

Besides the above major contributions, we also propose a fast inference algorithm to deal with the **cold start** challenge [15, 28], which is crucial for modern e-commerce platforms. Different from the cold-start solutions for collaborative filtering using matrix factorization methods [4, 38], we infer from product contextual features. The strategy is consistent with recent work, which finds contextual features playing an essential role in mitigating the cold-start problem [8, 27]. We also show that cold-start product representations inferred from our algorithm empirically perform better than simple context similarity methods in recommendation tasks.

3 METHOD

In this section, we introduce the technical details of our method. We first list the components of our model, clarify their relationships, and define the embeddings and score functions in Section 3.1. We put together the whole framework for learning the representations under a Bayesian network structure in Section 3.2. We then discuss two ranking criteria for recommendation tasks using embeddings obtained from the proposed approach, in Section 3.3. The geometric interpretation of using dual product embeddings and user bias term is discussed in Section 3.4. Finally, we present our fast inference algorithm for cold-start products in Section 3.5.

3.1 Setup

Let $\mathcal{U}$ be the set of N users and $\mathcal{I}$ be the set of M items. The contextual features for items such as brand, title and description are treated as tokenized words. Product categories and discretized continuous features are also treated as tokens. Without loss of generality, for each item we concatenate all the tokens into a vector denoted by $\mathbf{w}_i$ for item $i \in \mathcal{I}$. We use $\mathcal{W}$ to denote the whole set of tokens which has n_w instances in total. Similarly each user u has a feature vector of tokens $\mathbf{x}_u$ and $\mathcal{X}$ denotes the whole set of n_x user features. A complete observation for user u with the first purchase after time t and previous k (k may vary) consecutive purchases is given by $\{u, i_t, i_{t-1}, \ldots, i_{t-k}, \mathbf{x}_u, \mathbf{w}_{i_t}, \mathbf{w}_{i_{t-1}}, \ldots, \mathbf{w}_{i_{t-k}}\}$.

The representations are optimized for predicting the next purchase according to the user and the most recent purchases. To encode contextual knowledge into item/user representations, we use formulation similar to that of *SG* and *CBOW*:

- $p(\mathbf{I}_t|\mathbf{U}, \mathbf{I}_{t-1}, \ldots, \mathbf{I}_{t-k})$ - Predict next-to-buy item given user and recent k purchases.
- $p(\mathbf{W}_I|\mathbf{I})$ - Predict items' contextual features. We adopt the factorization from *CBOW* such that:

$$p(\mathbf{W}_I|\mathbf{I}) = \prod_{\mathbf{W} \in \mathbf{W}_I} p(\mathbf{W}|\mathbf{I}).$$

- $p(\mathbf{X}_U|\mathbf{U})$ - Predict users' features. We also factorize this term into $\prod_{\mathbf{X} \in \mathbf{X}_U} p(\mathbf{X}|\mathbf{U})$.

In *SG*/*CBOW* models, estimating the conditional probabilities is treated as multi-class classification problem with *softmax* classifier

$$p(i_1|i_2) = \frac{e^{s(i_1, i_2)}}{\sum_j e^{s(i_j, i_2)}},$$

where the score function $s(i, j)$ gives the similarity of the two items according to their embeddings such as $s(i_1, i_2) = \langle \mathbf{z}_1, \mathbf{z}_2 \rangle$. We then define the embedding notations for items, users, words, user features and the score functions.

- Let $Z^I \in \mathbb{R}^{M \times P_I}$ and $\tilde{Z}^I \in \mathbb{R}^{M \times P_I}$ be the dual embeddings for items, such that $s(i_1, i_2) = \langle \tilde{\mathbf{z}}^I_{i_1}, \mathbf{z}^I_{i_2} \rangle$ and $s(i_2, i_1) = \langle \tilde{\mathbf{z}}^I_{i_2}, \mathbf{z}^I_{i_1} \rangle$ measures the complementariness of i_1 given i_2 and i_2 given i_1 respectively. We refer to Z^I as item-in embedding and $\tilde{Z}^I$ as item-out embedding.
- Let $Z^U \in \mathbb{R}^{N \times P_U}$ be the set of embeddings for users and $Z^{IU} \in \mathbb{R}^{M \times P_U}$ be the set of embeddings for item-user context, such that $s(i, u) = \langle \mathbf{z}^{IU}_i, \mathbf{z}^U_u \rangle$ measures the personalized preference of user u on item i.
- Let $Z^W \in \mathbb{R}^{n_w \times P_I}$ be the set of word embeddings, such that $s(w, i) = \langle \mathbf{z}^W_w, \mathbf{z}^I_i \rangle$ measures relatedness between item i and word token w.
- Let $Z^X \in \mathbb{R}^{n_x \times P_x}$ be the set of discretized user feature embeddings, such that $s(x, u) = \langle \mathbf{z}^X_x, \mathbf{z}^U_u \rangle$ measure relatedness between feature x and user u.

By above definitions Z^W and Z^X are explicitly relating contextual knowledge to item and user representations, in a multi-task learning fashion. We then define the score function when input is a sequence of items, i.e $s(i_t, (i_{t-1}, \ldots, i_{t-k}))$, to capture **higher-order** complementariness. Using sequence as input for score function has also been spotted in other embedding-based recommender systems. PinSage has experimented on mean pooling and max pooling as well as using LSTM [35]. Youtube uses mean pooling [6], and another work from Alibaba for learning user interests proposes using the attention mechanism [37]. We choose using simple pooling methods over others for interpretability and speed. Our preliminary experiments show that mean pooling constantly outperforms max pooling, so we settle down to the mean pooling show in (2).

$$s(i_t, i_{t-1:t-k}) \equiv s(i_t, (i_{t-1}, \ldots, i_{t-k})) = \langle \tilde{\mathbf{z}}^I_{i_t}, \sum_{j=1}^{k} \mathbf{z}^I_{i_{t-j}} \rangle. \quad (2)$$

It is now straightforward to see that $s(i, u)$ and $s(i_t, (i_{t-1}, \ldots, i_{t-k}))$ meet the demand of a realization of $f_{UI}(.)$ and $f_I(.)$ in (1).

3.2 Model

To put together all the components we first point out that the joint probability function of the full observation has an equivalent Bayesian network representation [31] (right panel of Table 1). The log probability function now has the factorization (decomposition) shown in (3). Since we do not model the marginals of $p(\mathbf{I})$ and $p(\mathbf{U})$ with embeddings, we treat them as constant terms denoted by C in (3).

$$
\begin{aligned}
&\log p(\mathbf{U}, \mathbf{I}_t, \mathbf{I}_{t-1}, \ldots, \mathbf{I}_{t-k}, \mathbf{X}_U, \mathbf{W}_{I_t}, \mathbf{W}_{I_{t-1}}, \ldots, \mathbf{W}_{I_{t-k}}) \\
&= \log p(\mathbf{I}_t|\mathbf{U}, \mathbf{I}_{t-1}, \ldots, \mathbf{I}_{t-k}) + \log p(\mathbf{X}_U|\mathbf{U}) + \sum_{j=0}^{k} \log p(\mathbf{W}_{I_{t-j}}|\mathbf{I}_{t-j}) + C \\
&= \log p(\mathbf{I}_t|\mathbf{U}, \mathbf{I}_{t-1}, \ldots, \mathbf{I}_{t-k}) \\
&\quad + \underbrace{\sum_{\mathbf{X} \in \mathbf{X}_U} \log p(\mathbf{X}|\mathbf{U}) + \sum_{j=0}^{k} \sum_{\mathbf{W} \in \mathbf{W}_{I_{t-j}}} \log p(\mathbf{W}|\mathbf{I}_{t-j})}_{\text{Contextual knowledge}} + C.
\end{aligned} \tag{3}
$$

As we have pointed out before, several related models can be viewed as special cases of our approach under the Bayesian network structure. Visual presentations of these models are shown in Table 1. Classic item-item collaborative filtering for recommendation works on conditional probability of item pairs $p(I_t|I_{t-1})$. Sequential item models such as *item2vec* work on $p(I_t|I_{t-1}, \cdots, I_{t-k})$. The recently proposed triplet model focus on (item, item, user) triplet distribution: $p(I_t|I_{t-1}, U)$, and *prod2vec* models purchase sequence with user as $p(I_t|I_{t-1}, \cdots, I_{t-k}, U)$.

In analogy to *SG*/*CBOW* model, each term in (3) can be treated as as multi-class classification problem with *softmax* classifier. However, careful scrutiny reveals that such perspective is not suitable under e-commerce setting. To be concrete, using softmax classification model for $p(\mathbf{I_t}|\mathbf{U}, \mathbf{I_{t-1}}, \ldots, \mathbf{I_{t-k}})$ implies that given the user and recent purchases we only predict one next-to-buy complementary item. However, it is very likely that there are several complementary items for the purchase sequence. Same argument holds when modelling item words and user features with multi-class classification.

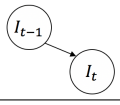
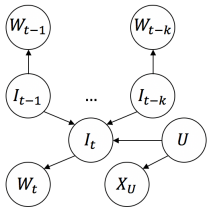
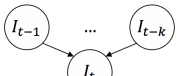
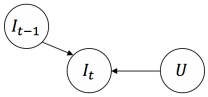
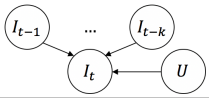
	Special cases	Proposed model
Item-item *CF* models [17]		
Sequential models (*item2vec* [2])		
triplet2vec [32]		
prod2vec [9]		

Table 1: The Bayesian network representation of joint probability distribution of observation, for our proposed approach (right panel) and classic recommendation algorithms (left panels).

Instead, we treat each term in (3) as binary logistic classification problem. Now the semantic for $p(\mathbf{I}_t|\mathbf{U}, \mathbf{I}_{t-1}, \ldots, \mathbf{I}_{t-k})$ becomes that given the user and recent purchases, whether I_t is purchased next. Similarly, $p(\mathbf{W}|\mathbf{I})$ now implies whether or not the item has the context words. So the loss function for each complete observation using binary logistic loss is now given by (4).

$$
\begin{aligned}
\ell = &\log\left(1 + e^{-s(i_t,u)-s(i_t,i_{t-1:t-k})}\right) + \sum_{\tilde{i} \in \mathcal{I}/i_t} \log\left(1 + e^{s(\tilde{i},u)+s(\tilde{i},i_{t-1:t-k})}\right) \\
&+ \sum_{j=0}^{k} \sum_{w \in \mathbf{w}_{i_{t-j}}} \left\{\log\left(1 + e^{-s(w,i_{t-j})}\right) + \sum_{\tilde{w} \in \mathcal{W} \setminus \mathbf{w}_{i_{t-j}}} \log\left(1 + e^{s(\tilde{w},i_{t-j})}\right)\right\} \\
&+ \sum_{x \in \mathbf{x}_u} \left\{\log\left(1 + e^{-s(x,u)}\right) + \sum_{\tilde{x} \in \mathcal{X} \setminus x} \log\left(1 + e^{s(\tilde{x},u)}\right)\right\}
\end{aligned} \tag{4}
$$

Recall that $s(i, u)$ models user preferences and $s(i_1, i_2)$ models item complementary patterns. So the terms in the first line of (4) aim at optimizing the user embedding Z^U and item-user embedding Z^{IU} together with the dual item embeddings Z^I and $\tilde{Z}^I$, according to observed user purchase sequences. Terms in the second line encode contextual knowledge to item-in embedding Z^I such that item with similar word contexts is optimized to be close to each other, and thus preserve the transductive property. This is in analogy to the *CBOW* version of *doc2vec* [16], that if we treat item as documents then $\langle \mathbf{z}^I_{i_1}, \mathbf{z}^I_{i_2} \rangle$ measures item contextual similarity between i_1 and i_2. The same argument applies to the third line, where users with similar features have closer representations in user embedding space. So far, we have unified the different learning tasks into one loss function, which follows naturally from our Bayesian network structure.

We notice that binary classification loss requires summing over all possible negative instances, which is computationally impractical. Like other shallow embedding approaches, we also use negative sampling as an approximation for all binary logistic loss terms, with either frequency-based or hierarchical softmax negative sampling schema proposed in [20]. For instance, the first line in (4) is now approximated by (5) where $Neg(\mathcal{I})$ denotes a set of negative item samples.

$$
\log\left(1+e^{-s(i_t,u)-s(i_t,i_{t-1:t-k})}\right) + \sum_{\tilde{i} \in Neg(\mathcal{I})} \log\left(1+e^{s(\tilde{i},u)+s(\tilde{i},i_{t-1:t-k})}\right) \tag{5}
$$

Implementation details of our optimization algorithm are provided in Appendix.

3.3 Ranking criteria for recommendation

In this section, we briefly discuss how to conduct online recommendation with the optimized embeddings, where top-ranked items are provided for customers based on their recent purchases and/or personal preferences.

The first criteria is based only on the score function $s(i, i_{t:t-k+1})$, which indicates the strength of complementary signal between the previous purchase sequence and target item.

The second criterion accounts for both user preference and complementary signal strength, where target items are ranked according to $s(i,u) + s(i, i_{t:t-k+1})$. The advantage for this criterion is that user preferences are explicitly considered, but this also means the recommendations can be driven away from the most potent complementary item to compensate for user preference. Also, the computation cost is doubled compared with the first criterion. We compare their empirical performances in detail in Section 4.

3.4 Geometric Interpretation

Here we give our geometric interpretation for the additional item-out embedding $\tilde{Z}^I$ and the user bias term. The use of dual item embedding is not often seen in embedding-based recommender system literature, but here it is essential for modelling the intrinsic properties of complementariness. Suppose all embeddings are fixed except $\tilde{Z}^I$, according to classical separating hyperplane theory, the vector $\tilde{\mathbf{z}}_j$ is actually the normal of the separating hyperplane for item j with respect to the embeddings of positive and negative purchase sequences. In other words, the hyperplane tries to identify that for item j as a next-to-buy complementary item, which previous purchase sequences are positive and which are not.

Consider the total loss function $L = \sum_{q=1}^{n} \ell_q$, where n is the number of observed purchase sequences and the subscript q gives the index of the observation. For the loss function in (4) item-out embeddings $\tilde{Z}^I$ only appears in the first two terms. Using the separability of L we can collect all terms that involve the item-out embedding of item j, as we show in (6). For clarity purpose we make $k = 1$ in (6), i.e only the most recent purchase is included.

$$\mathcal{L}_j = \sum_{(i,u)\in\mathcal{P}} \log\left(1 + e^{-b_{j,u} - \langle \tilde{\mathbf{z}}_j^I, \mathbf{z}_i^I \rangle}\right) + \sum_{(\tilde{i},u)\in\mathcal{N}} \log\left(1 + e^{b_{j,u} + \langle \tilde{\mathbf{z}}_j^I, \mathbf{z}_{\tilde{i}}^I \rangle}\right). \tag{6}$$

In (6), $\mathcal{P}$ represents the whole set of item-user (i,u) pairs in the observed two-item purchase sequences $\{j,i\}$ for user u. $\mathcal{N}$ denotes other $(\tilde{i},u)$ pairs where item j is used as one of the negative samples in (5). The scalar $b_{j,u} \equiv s(j,u)$ is the preference of user u on item j. It is then obvious that optimizing $\tilde{\mathbf{z}}_j^I$ in (6) is equivalent to solving a logistic regression with $\tilde{\mathbf{z}}_j^I$ as parameters. The design matrix is constructed from fixed item-in embeddings $\{\mathbf{z}_i^I\}_{(i,u)\in\mathcal{P}\cup\mathcal{N}}$. One difference from ordinary logistic regression is that we have a fixed intercept terms $b_{j,u}$ for each user. Analytically this means that we employ a user's preferences on j, the associated user bias term, as the intercept, when using his/her purchase sequences to model the complementary relationship between j and other items. And under logistic regression, the regression parameter vector $\tilde{\mathbf{z}}_j^I$ gives the normal to the optimized separating hyperplane. We provide a sketch of our concept in Figure 1.

When $k \geq 2$, we replace $\mathbf{z}_i^I$ in (6) with mean pooling $\frac{1}{k}\sum_{q=1}^{k} \mathbf{z}_{i_q}^I$. Geometrically speaking, we now optimize the separating hyperplane with respect to the centroids of the positive and negative purchase sequences in item-in embedding space. The sketch in also provided in Figure 1. Representing sequences by their centroids helps us capturing the **higher-order** complementariness beyond pair-wise setting.

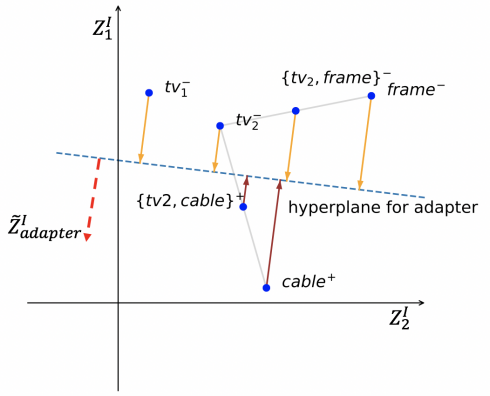


Figure 1: Geometric illustration for the separating hyperplane interpretation with $k = 1, 2$ and $P_I = 2$. For $adaptor$ as complementary item, tv^-, $frame^-$ are negative samples and $cable^+$ is positive sample. Also, $(tv_2, cable)^+$ combo is positive sample and $(tv, frame)^-$ is negative sample, where combo is represented by the centroid of the two items. $\tilde{z}^I_{adaptor}$ is optimized such that positive/negative samples have positive/negative distances to the separating hyperplane characterized by its normal vector $\tilde{z}^I_{adaptor}$ and intercept $b_{adaptor}$ (not shown here).

3.5 Item Cold Start Inference

Another unique advantage of modelling with contextual knowledge is that we are able to infer item-in embedding for cold-start items, only using trained context embedding Z^W. This saves the effort of retraining the whole model, which is often the case for some other methods. The major challenge from cold-start products is that recommender systems may not function properly when users are interested in products with few records or collected data. The coverage of many recommendation algorithms is thus severely limited, especially for large e-commerce platforms where customers frequently explore only a tiny fraction of products. We intend to solve this problem by exploiting the structure of the item-in Z^I embedding space.

Recall that $\langle \tilde{Z}^I, Z^I \rangle$ measures complementary strength and $\langle \tilde{Z}^I, Z^W \rangle$ accounts for contextual relationship. The **transductive** property is preserved on item-in embedding space because items with similar complementary products and contexts are now embedded closer to each other. So when complementary patterns are not available for cold-start products due to lack of data, we can still embed them according to the contextual relationship. For item i_0 with contextual features $\mathbf{w}_{i_0}$, we infer its item-in embedding $\mathbf{z}_{i_0}^I$ according to (7) so that it is embedded closely to contextually similar items in item-in embedding space. Therefore they are also more likely to find similar complementary items in recommendation tasks.

$$\hat{\mathbf{z}} = \arg\max_{\mathbf{z}\in\mathbb{R}^{pI1}} \prod_{w\in\mathbf{w}_{i_0}} p(w|\mathbf{z}), \text{ where } p(w|\mathbf{z}) \approx \frac{e^{s(w,i_0)}}{\sum_{\tilde{w}\in Neg(\mathcal{W})} e^{s(\tilde{w},i_0)}}. \tag{7}$$

Inferring item-out embedding for cold-start item, however, is not of high demand since people seldom recommend cold-start products

to customers. We examine the quality of inferred representations in recommendations tasks in Section 4.

4 EXPERIMENT AND RESULT

We thoroughly evaluate the proposed approach on two tasks: product classification and recommendation, using only the optimized product representations. While classification tasks aim at examining the **meaningfulness** of product representation, recommendation tasks reveal their **usefulness** [32]. We then examine both perspectives for the representations inferred for cold-start products, through the same tasks. Ablation studies are conducted to show the importance of incorporating contextual knowledge or user bias term, and sensitivity analysis are provided for embedding dimensions. We present several case studies to show that our approach captures the desired properties of complementariness (Section 1). Finally, the user embeddings are explored in downstream supervised and unsupervised learning task for various user segmentation (persona) studies in order to demonstrate their **meaningfulness**.

4.1 Datasets

We work on two real-world datasets: the public *Instacart* dataset and a proprietary e-commerce dataset.

- ***Instacart.*** *Instacart.com* provides an online service for same-day grocery delivery in the US. The dataset contains around 50 thousand grocery products with shopping records of ~200 thousand users, with a total of ~3 million orders. All products have contextual information, including name, category, department. No user context is available. Time information is not given, but the order of user activities are presented.
- **Proprietary e-commerce dataset** [1]**.** The proprietary dataset comes from a major e-commerce platform in the US and its online shopping catalogue covers more than 100 million items. It has user add-to-cart and transaction records (with temporal information) over a specific time span, containing ~8 million products, ~20 million customers and a total of ~0.1 billion orders. Product contextual information, including name, category, department and brand are also available. Also, user-segmentation (persona) labels are available for a small fraction of users.

Data preprocessing. We follow the same preprocessing procedure for *Instacart* dataset described in [32]. Items with less than ten transactions are removed. Since time information is not available, we use each user's last order as testing data, the second last orders are validation data and the other orders as training data. For the proprietary dataset we also filter out items with less than ten transactions. Referencing the timestamps, we split the data into training, validation and testing datasets using cutoff times in chronological order.

Purchase sequence construction. Constructing purchase sequences for training and testing is important for the proposed approach since product complementary patterns are revealed from users' sequential activities. Including long-past purchases may not be helpful for learning complementary relationships since they are unlikely to influence users' next purchase. Using same-session purchases may leave out important information since users often come for complementary products of previous purchases. To take account of both factors, we use purchases made in past d_1 days as purchase sequence, where d_1 can be treated as a tuning parameter that varies for different use cases. For the *Instacart* dataset we have to make the compromise of using past k purchases since time is not provided. Here k can be treated as the sliding window size.

[1]The dataset source will be made public at acceptance time.

4.2 Implementation

A primary goal of this work is to provide an efficient implementation for representation learning in large-scale e-commerce dataset. For the proprietary dataset, if all embeddings have dimension 100, the total number of parameters will be over 0.1 billion! This causes computation bottleneck for implementations based on current differentiable programming software such as *Tensorflow* and *PyTorch*. Therefore we design an efficient C++ implementation which fully utilizes the structures of the loss function, as a highly scalable solution.

Separable loss function. We observe that the loss function in (4) is highly separable, which suggests that computations on each observation can be easily parallelized. Therefore asynchronous stochastic gradient descent can be efficiently applied for training.

Sparse loss function. It has been observed in *word2vec* and its variants that despite the total number of parameters is huge for shallow embedding models, the number of parameters being updated during each training step is actually small since each observation is only associated with a few parameter vectors. So in our implementation, we adopt the Hogwild! algorithm, whose efficiency for optimizing sparse separable cost functions has been widely acknowledged.

Finally, our C++ implementation only takes 15 minutes to train on *Instacart* dataset and 11 hours to train on the proprietary dataset, which contains ~2 billion observations from the ~5 million users and ~2 million items after filtering, in a Linux server with 16 CPU threads and 40 GB memory. More implementation details are provided in the Supplement. The code and dataset will be made public at acceptance time.

4.3 Results on *Instacart* dataset

We compare the performance of the proposed approach on product classification and with-in basket recommendation tasks with state of the art product representation learning methods, i.e. *item2vec*, *prod2vec*, *metapath2vec* and *triplet2vec*. We also conduct ablation studies and sensitivity analysis. In ablation studies, we report the performances where item context or user bias term is removed from our model. In the sensitivity analysis, we focus on the dimensions of the item and user embedding. Finally, we evaluate the proposed approach for item cold-start inference. We randomly remove a fraction of items from training dataset as cold-start items and infer their representations after training with the rest data. The inferred representations are then evaluated under the same two tasks.

Tasks and metric. In product classification tasks we use items' department and categories as labels to evaluate the product representations under coarse-grained and fine-grained scenarios, respectively. We apply the one-vs-all linear logistics regression with the learned item embeddings as input features. The label fraction is 0.5

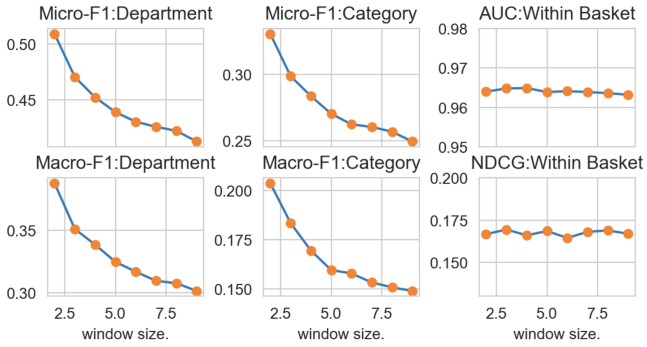


Figure 2: The performances of classification and recommendation tasks under different window size k on *Instacart* dataset without item context.

and l_2 regularization term is selected from 5-fold cross-validation. For this imbalanced multi-class classification problem, we report the average **micro-F1** and **macro-F1** scores. To prevent information leaks, we exclude department and category information and only use product names for the proposed model.

In within-basket recommendation task, we rank candidate products according to their complementariness scores to the current basket, by taking the average item-in embeddings as basket representation. The Area Under the ROC Curve (**AUC**) [24] and Normalized Discounted Cumulative Gain (**NDCG**) designed for evaluating recommendation outcome are used as metrics. While **AUC** examines the average performance without taking account of exact ranking status, **NDCG** is top-biased, i.e. higher ranked recommendations are given more credit. The classification and recommendations performances are reported in Table 2.

First we take a look at how k (sequence length) influence the performances in both tasks (Figure 2). For fair comparisons with the baseline models reported in [32], we also set product embedding dimension to 32. In Figure 2 we see that smaller window size leads to better performances on classification tasks, and for recommendation task, the performance is quite stable under all k. The former result suggests that for grocery shopping on *Instacart.com*, more recent purchases play major roles in fulfilling the **meaningfulness** of product representation. However, the latter result indicates that for the recommendation task, our approach is robust concerning the length of input sequences, even though longer sequences may introduce more noise. Without further notice, we use $k = 2$ in the following experiments and analysis.

From Table 2 we observe that the proposed approach outperforms all baseline models that also learn product representations (**full model**), in both classification and recommendation tasks. This suggests that our model can better capture the **usefulness** and **meaningfulness** of product representations. By taking out the user bias term, we observe a drop in performances (**no user** in Table 2), which indicates the importance of explicitly modelling with user preference. Although removing product context leads to worse performance on product classification (**no context** in Table 2), it gives similar or slightly better results in the recommendation task. We conclude the reasons as follow. Firstly, it is apparent that including context information can boost the **meaningfulness** of product representation since more useful information is encoded. Secondly, as we discussed before, the impact of context information on recommendation can depend on the sparsity of user-item interactions. Product-context interactions are introduced to deal with the sparsity issue by taking advantage of the **transductive** property, i.e. similar products may share complementary products. Therefore context information may not provide much help in the denser *Instacart* dataset which covers only 50 thousand items. We show later that context information is vital for recommendation performances in the sparse proprietary dataset. Finally, the inference algorithm (7) of cold-start products gives reasonable results, as they achieve comparable performances with baseline methods in both tasks. It also outperforms the heuristic method which assigns representations to cold start products according to their most similar product (by Jaccard similarity of contexts).

	Classification				Recom.	
	Department		Category		In-basket	
Method	micro	macro	micro	macro	AUC	NDCG
item2vec	0.377	0.283	0.187	0.075	0.941	0.116
prod2vec	0.330	0.218	0.106	0.030	0.941	0.125
*m.2vec**	0.331	0.221	0.155	0.036	0.944	0.125
triple2vec	0.382	0.294	0.189	0.082	0.960	0.127
no context	0.509	0.470	0.452	0.438	0.964	0.166
no user	0.661	0.545	0.615	0.529	0.954	0.160
full model	0.666	0.553	0.619	0.535	0.965	0.151
-user bias	-	-	-	-	0.858	0.137
cold start	0.301	0.207	0.304	0.214	0.846	0.114
-infer	-	-	-	-	0.801	0.084

Table 2: Performances on multiple tasks on Instacart data. Methods in bold fonts represents our method under several configurations. The left 4 columns provide micro and macro f-1 scores for the product classification tasks under coarse-grained (department) and fine-grained (category) levels. The right two columns gives the AUC and NDCG for within-basket recommendations. The '-user bias' row gives the result when ranking without using user embeddings, and '-infer' row gives cold start recommendation results based only on context Jaccard similarity without using inference algorithm. * *m.2vec* is the shorthand for *metapath2vec*.

4.4 Results on the proprietary dataset

We focus on examining the **usefulness** of our product representation learning method with the proprietary dataset. As we mentioned before, the availability of timestamps and the lack of **confounding** factors such as loyalty make the proprietary dataset ideal for evaluating recommendation performances. This also encourages us to compare with baseline models for recommendation such as collaborative filtering (*CF*). We also include state-of-the-art supervised implicit-feedback recommendation baselines such as factorizing personalized Markov chain (*FPMC*) [25] and Bayesian personalized ranking (*BPR*) [24]. Last but not least, we further compare with the graph-based recommendation method adapted from *node2vec* [33]. On the other hand, without major modifications, the aforementioned product embedding methods do not scale to the size of

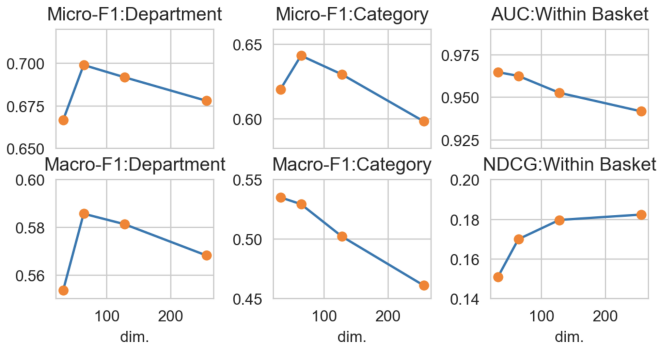


Figure 3: Sensitivity analysis on embedding dimension for product classification and within-basket recommendation tasks on *Instacart* dataset.

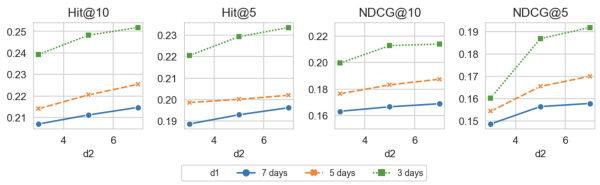


Figure 4: Sensitivity analysis for the $d1$ and $d2$, where purchases up to $d1$ days before the current time are used as input sequences and use the user's purchases in next $d2$ days are used as labels, with $P_I = 100$ and $P_U = 100$.

Model	Hit@10	Hit@5	NDCG@10	NDCG@5
CF	0.0962	0.0716	0.0582	0.0506
FPMC	0.1473	0.1395	0.1386	0.1241
BPR	0.1578	0.1403	0.1392	0.1239
node2vec	0.1103	0.0959	0.0874	0.0790
prod2vec	0.1464	0.1321	0.1235	0.1162
no user	0.1917	0.1726	0.1485	0.1301
no context	0.1778	0.1529	0.1346	0.1198
full model	0.2518	0.2336	0.2139	0.1919
-user bias	0.2514	0.2334	0.2126	0.1917
cold start	0.1066	0.0873	0.0759	0.0562
-infer	0.0734	0.0502	0.0531	0.0428

Table 3: Comparision with baseline models under $P_I = 100$, $P_U = 100$, $d_1 = 3$ and $d_2 = 7$.

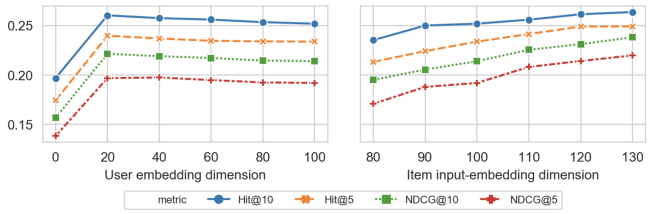


Figure 5: Sensitivity analysis for user and item embeddings dimensions on the proprietary dataset.

the proprietary dataset. Still, we manage to implement a similar version of *prod2vec* based on the *word2vec* implementation for a complete comparison.

Tasks and metric. The general next-purchase recommendations (includes within-basket and next-basket) are crucial for real-world recommender systems and are often evaluated by the top-K hitting rate (*Hit@K*) and top-K normalized discounted cumulative gain (*NDCG@K*) according to customers' purchase in next d_2 days. With timestamps available for the proprietary dataset we can carry out these tasks and metrics.

The performance of the proposed approach under different d_1 and d_2 are first provided in Figure 4, where we observe that larger d_2 and smaller d_1 leads to better outcomes. We notice that different d_1 gives very similar results, which is in accordance with the window size in *Instacart* dataset. They both suggest that our approach is robust concerning the input sequence length. The fact that larger d_2 leads to better results is self-explanatory. Without further notice we use $d_1 = 3$ to construct input sequences and $d_2 = 7$ to extract target products for all models. For fair comparisons with baselines which cannot incorporate user context information, we also exclude user contextual features in our model in all experiments.

The recommendation performance evaluation are provided in Table 3. The proposed approach outperforms all baseline methods by significant margins on all metrics, including standard recommendation algorithms and product embedding methods. Due to the high sparsity and low signal-to-noise ratio of the proprietary dataset, the baseline methods may not be able to capture co-purchase signals without accounting for contextual information and user bias.

As we mentioned before, after removing product context terms the performances drop significantly, which again stresses the importance of designing context-aware models for large sparse datasets. Also, it follows the results of *Instacart* dataset that taking out user bias term deteriorates the overall performance. Also, the representations inferred for cold-start products give also reasonable recommendation performances that are comparable to the outcome of some baselines evaluated on standard products.

4.5 Ranking criteria

The comparisons between the two ranking methods mentioned above (Section 3.3) are shown in Table 2 and 3. In both tables, the **full model** row uses the second ranking criteria, i.e. with both product and user embeddings, while the '-user bias' row uses only product representations. Ranking with only item embeddings leads to worse results on *Instacart* dataset but very similar results on the proprietary dataset. The similar performance on the proprietary dataset is within expectation, as we reasoned in Section 3.3 that both criteria have pros and cons. As for the outcome on *Instacart* dataset, We believe this is again due to the **confounding** factor of customer loyalty, where the user bias term may have implicitly captured some customer loyalty signals.

Generally speaking, the benefits of including user preference in ranking should be discussed case by case. Although user preference

terms are essential for learning product complementary relationships by taking account of noises in the user purchase sequence, they do not necessarily bring significant gains in recommendations.

4.6 Sensitivity analysis

We provide a thorough sensitivity analysis on embedding dimensions on both datasets in Figure 3 and 5. Specifically, for *Instacart* dataset, after increasing product embedding dimension over 80, the proposed model starts to lose classification accuracies. On the other hand, the **NDCG** for recommendation task keeps improving with larger dimensions (within the range considered), though the **AUC** slightly decreases. For the proprietary dataset, we focus on both user and product embedding dimensions. Increasing user embedding dimension over 20 starts to cause worse performances. Moreover, similar to the results on *Instacart* dataset, larger product embedding dimensions give better outcomes.

In fact, recent work reveals the connection between embedding size and the bias-variance trade-off [34]. Embedding dimensions that are too large can overfit training data, and small dimensions can cause underfitting. This phenomenon may explain our results, where a larger product embedding dimension causes overfitting issues in classification tasks on *Instacart* dataset. For the proprietary dataset, increasing user embedding dimension over the threshold may have also overfitted user behaviour in training data.

4.7 Case study

We use several case studies to show that the propose approach captures the desired properties of complementariness, i.e **asymmetric**, **non-transitive** and **higher-order**, provided in Table 4. The **transductive** property has been explicitly modelled by including contextual information, so we do not provide concrete examples here.

By virtue of the examples, we observe the **Asymmetric** property via *TV* $\rightarrow$ *TV mount frame* but *TV mount frame* $\not\rightarrow$ *TV*. The **Non-transitive** is also reflected by *TV* $\rightarrow$ *TV mount frame*, *TV mount frame* $\rightarrow$ *cable cover* but *TV* $\not\rightarrow$ *cable cover*. As for the **Higher-order** property, we first note that when *round cake pan* is combined with *straight spatula*, the recommendations are not a simple mixture of individual recommendations. *Cake carrier* and *cake decorating set*, which complement the combo as a whole, are recommended. Similar higher-order pattern is also observed for *sofa* and *coffee table* combination. They provide evidence that our model captures the **higher-order** complementary patterns.

In the next case study, we demonstrate that taking the average of product representation within a general basket leads to reasonable recommendations. A real-world shopping basket and the top 5 recommendations according to our model are provided in Figure 6. Although the products in the basket are mostly unrelated, we see that the recommendations are essentially a mixture of complements to individual products in the basket.

4.8 User embedding analysis

Besides product embeddings, our model also learns user representations. We demonstrate the **meaningfulness** of user embedding via user segmentation studies. User segmentation plays essential parts in modern e-commerce platform by guiding advertising and marketing strategies. We refer to different user segmentation criteria as the persona - for instance, the persona 'new parents' labels customers into new parents and others. The proprietary dataset contains the scores of persona for a subset of customers.

Here we consider three personas: pet owners, busy family and new parents. For each persona, we first conduct a k-nearest neighbour learning, where customers with high persona score are labelled as positive samples and the rest as negative samples. We use 5-fold cross-validation to tune the hyperparameter k. Around 80% customers are randomly selected as training samples. The testing results on the remaining customers are given in Table 5. The high prediction accuracy for the three personas suggests that users who are close in embedding space have similar persona labels.

We then conduct a KMeans clustering analysis with ten centers, on the ~5 million customers who have user embedding. Cosine distance is used as the similarity measure in clustering. We compute its average persona score of all customers for each cluster (Figure7). We see that the grouping pattern is quite strong, where cluster 1 and 2 have many pet owners, and cluster 3, 4, 5 and 8 consists mostly of busy families and new parents.

Anchor item	Recommended complementary items		
Bath curtain	Bath curtain hooks	Bath curtain rod	Bath mat
Bath mat	Towels	Mops	Detergent
TV	Protection plan	HDMI cable	TV mount frame
TV mount frame	HDMI cable	DVD hold with cable	Cable cover
Round cake pan	Straight spatula	Cupcake pan	Lever tool
Straight spatula	Spatula set	Wood spatula	Steel spatula
Round cake pan + Straight spatula	Cake carrier	Cake decorating set	Icing smoother
Sofa	Sofa pillows	Coffee table	Sofa cover
Sofa + Coffee table	Floor lamp	Area rug	Sofa pillows

Table 4: Case studies of product recommendation.

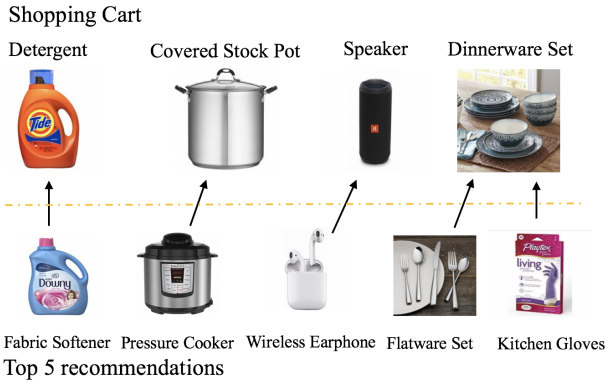


Figure 6: Top 5 recommendations (second row) for a real-world shopping cart (first row) consisting of unrelated items. We mark complementary relationships between items with arrows.

category	precision	recall	accuracy	ap*
Pet Owner	0.7378	0.9925	0.7343	0.7443
Busy Families	0.7378	0.9925	0.7723	0.7797
New Parents	0.8569	0.9976	0.8553	0.8631

Table 5: Metrics of classification tasks for the KNN classifier using user embeddings. *ap means the average precision score.

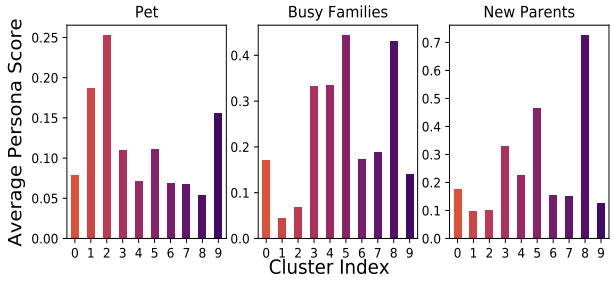


Figure 7: The average persona scores of each cluster.

5 CONCLUSION

In this work, we thoroughly investigate the properties of product complementary relationship-**asymmetric**, **non-transitive**, **transductive** and **higher-order**. We propose a novel representation learning model for complementary products with dual embeddings that are compatible with the above properties. Contextual information is leveraged to deal with the sparsity in customer-product interactions, and user preferences are explicitly modelled to take account of noise in user purchase sequences. A fast inference algorithm is developed for cold-start products. We demonstrate the effectiveness of the product representations obtained from the proposed approach through classification and recommendation tasks on *Instacart* and the proprietary e-commerce dataset.

Future work includes exploring populational contexts such as product popularity and dynamic factors including trend and seasonality. Furthermore, the idea of context-aware representation learning for complex relationships can be extended beyond e-commerce setting to learn the general event representation from user-event interaction sequences.

REFERENCES

[1] Marko Balabanović and Yoav Shoham. 1997. Fab: content-based, collaborative recommendation. *Commun. ACM* 40, 3 (1997), 66–72.

[2] Oren Barkan and Noam Koenigstein. 2016. Item2vec: neural item embedding for collaborative filtering. In *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 1–6.

[3] Mikhail Belkin and Partha Niyogi. 2003. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation* 15, 6 (2003), 1373–1396.

[4] JesúS Bobadilla, Fernando Ortega, Antonio Hernando, and JesúS Bernal. 2012. A collaborative filtering approach to mitigate the new user cold start problem. *Knowledge-Based Systems* 26 (2012), 225–238.

[5] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5 (2017), 135–146.

[6] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 191–198.

[7] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 135–144.

[8] Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, Steffen Rendle, and Lars Schmidt-Thieme. 2010. Learning attribute-to-feature mappings for cold-start recommendations. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*. IEEE, 176–185.

[9] Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, and Doug Sharp. 2015. E-commerce in your inbox: Product recommendations at scale. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1809–1818.

[10] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 855–864.

[11] John Hannon, Mike Bennett, and Barry Smyth. 2010. Recommending twitter users to follow using content and collaborative filtering approaches. In *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 199–206.

[12] Zan Huang, Hsinchun Chen, and Daniel Zeng. 2004. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Transactions on Information Systems (TOIS)* 22, 1 (2004), 116–142.

[13] Ioannis Konstas, Vassilios Stathopoulos, and Joemon M Jose. 2009. On social networks and collaborative recommendation. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. ACM, 195–202.

[14] Yehuda Koren and Robert Bell. 2015. Advances in collaborative filtering. In *Recommender systems handbook*. Springer, 77–118.

[15] Xuan Nhat Lam, Thuc Vu, Trong Duc Le, and Anh Duc Duong. 2008. Addressing cold-start problem in recommendation systems. In *Proceedings of the 2nd international conference on Ubiquitous information management and communication*. ACM, 208–211.

[16] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*. 1188–1196.

[17] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing* 1 (2003), 76–80.

[18] Julian McAuley, Rahul Pandey, and Jure Leskovec. 2015. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 785–794.

[19] Prem Melville, Raymond J Mooney, and Ramadass Nagarajan. 2002. Content-boosted collaborative filtering for improved recommendations. *Aaai/iaai* 23 (2002), 187–192.

[20] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[21] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. 2016. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1105–1114.

[22] Manos Papagelis, Dimitris Plexousakis, and Themistoklis Kutsuras. 2005. Alleviating the sparsity problem of collaborative filtering using trust inferences. In *Trust management*. Springer, 224–239.

[23] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

[24] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 452–461.

[25] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*. ACM, 811–820.

[26] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*. ACM, 285–295.

[27] Martin Saveski and Amin Mantrach. 2014. Item cold-start recommendations: learning local collective embeddings. In *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, 89–96.

[28] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. 2002. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 253–260.

[29] Ian Soboroff and Charles Nicholas. 1999. Combining content and collaboration in text filtering. In *Proceedings of the IJCAI*, Vol. 99. sn, 86–91.

[30] Flavian Vasile, Elena Smirnova, and Alexis Conneau. 2016. Meta-prod2vec: Product embeddings using side-information for recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 225–232.

[31] Martin J Wainwright, Michael I Jordan, et al. 2008. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*

1, 1–2 (2008), 1–305.
[32] Mengting Wan, Di Wang, Jie Liu, Paul Bennett, and Julian McAuley. 2018. Representing and Recommending Shopping Baskets with Complementarity, Compatibility and Loyalty. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 1133–1142.
[33] Jizhe Wang, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao, and Dik Lun Lee. 2018. Billion-scale Commodity Embedding for E-commerce Recommendation in Alibaba. *arXiv preprint arXiv:1803.02349* (2018).
[34] Zi Yin and Yuanyuan Shen. 2018. On the dimensionality of word embedding. In *Advances in Neural Information Processing Systems*. 895–906.
[35] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. *arXiv preprint arXiv:1806.01973* (2018).
[36] Jiaqian Zheng, Xiaoyuan Wu, Junyu Niu, and Alvaro Bolivar. 2009. Substitutes or complements: another step forward in recommendations. In *Proceedings of the 10th ACM conference on Electronic commerce*. ACM, 139–146.
[37] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1059–1068.
[38] Ke Zhou, Shuang-Hong Yang, and Hongyuan Zha. 2011. Functional matrix factorizations for cold-start recommendation. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. ACM, 315–324.

A RESEARCH METHODS

Since we optimize the embeddings with stochastic gradient descent, we first present the gradients for the embeddings with respect to the loss function on single complete observation $\{u, i_t, i_{t-1}, \ldots, i_{t-k}, \mathbf{x}_u, \mathbf{w}_{i_t}, \mathbf{w}_{i_{t-1}}, \ldots, \mathbf{w}_{i_{t-k}}\}$ presented in (4) under negative sampling.

Let $Q_{i,u,i_{t-1:t-k}} = s(i,u) + s(i, i_{t-1:t-k})$, $Q_{w,i} = s(w,i)$ and $Q_{x,u} = s(x,u)$. The gradient for $\mathbf{z}_u^U$ can be computed by

$$\nabla_{\mathbf{z}_u^U} \ell = -\mathbf{z}_{i_t}^I \frac{e^{-Q_{i,u,i_{t-1:t-k}}}}{1+e^{-Q_{i,u,i_{t-1:t-k}}}} + \sum_{\tilde{i}\in Neg(\mathcal{I})} \mathbf{z}_{\tilde{i}}^I \frac{e^{Q_{\tilde{i},u,i_{t-1:t-k}}}}{1+e^{Q_{\tilde{i},u,i_{t-1:t-k}}}} - \sum_{x\in\mathbf{x}_u} \mathbf{z}_x^X \frac{e^{-Q_{x,u}}}{1+e^{-Q_{x,u}}} + \sum_{\tilde{x}\in\mathcal{X}\setminus x} \mathbf{z}_{\tilde{x}}^X \frac{e^{Q_{\tilde{x},u}}}{1+e^{Q_{\tilde{x},u}}}. \tag{8}$$

The gradient for $\tilde{\mathbf{z}}_{i_t}^I$ and $\mathbf{z}_{i_t}^I$ are given by

$$\nabla_{\tilde{\mathbf{z}}_{i_t}^I} \ell = -(\mathbf{z}_u^U - \frac{1}{k}\sum_{j=1}^k \mathbf{z}_{i_{t-j}}^I) \frac{e^{-Q_{i,u,i_{t-1:t-k}}}}{1+e^{-Q_{i,u,i_{t-1:t-k}}}}, \tag{9}$$

and

$$\nabla_{\mathbf{z}_{i_t}^I} \ell = -\sum_{w\in\mathbf{w}_{i_t}} \mathbf{z}_w^W \frac{e^{-Q_{w,i_t}}}{1+e^{-Q_{w,i_t}}} + \sum_{\tilde{w}\in Neg(\mathcal{W})} \mathbf{z}_{\tilde{w}}^W \frac{e^{-Q_{\tilde{w},i_t}}}{1+e^{-Q_{\tilde{w},i_t}}}. \tag{10}$$

For $\mathbf{z}_{i_{t-j}}^I$, $j = 1, \ldots, k$, we compute their gradients as:

$$\nabla_{\mathbf{z}_{i_{t-j}}^I} \ell = -\mathbf{z}_{i_t}^I \frac{e^{-Q_{i,u,i_{t-1:t-k}}}}{k(1+e^{-Q_{i,u,i_{t-1:t-k}}})} + \sum_{\tilde{i}\in Neg(\mathcal{I})} \mathbf{z}_{\tilde{i}}^I \frac{e^{Q_{\tilde{i},u,i_{t-1:t-k}}}}{k(1+e^{Q_{\tilde{i},u,i_{t-1:t-k}}})} - \sum_{w\in\mathbf{w}_{i_{t-j}}} \mathbf{z}_w^W \frac{e^{-Q_{w,i_t}}}{1+e^{-Q_{w,i_{t-j}}}} + \sum_{\tilde{w}\in Neg(\mathcal{W})} \mathbf{z}_{\tilde{w}}^W \frac{e^{-Q_{\tilde{w},i_{t-j}}}}{1+e^{-Q_{\tilde{w},i_{t-j}}}}. \tag{11}$$

As for the embeddings for item contextual feature (word) token w and users feature x that show up in the observation, their gradients can be computed by:

$$\nabla_{\mathbf{z}_w^W} \ell = \sum_{\substack{i:w\in i,\\ i\in\{i_t,\ldots,i_{t-k}\}}} -\mathbf{z}_i^I \frac{e^{-Q_{w,i}}}{1+e^{-Q_{w,i}}} \tag{12}$$

and

$$\nabla_{\mathbf{z}_x^X} \ell = -\mathbf{z}_u^U \frac{e^{-Q_{x,u}}}{1+e^{-Q_{x,u}}}. \tag{13}$$

For the $\tilde{i}$, $\tilde{w}$ and $\tilde{w}$ that are being sampled in (8) to (13), their gradients can be calculated in similar fashion. We do not present them here to avoid unnecessary repetition.

During training, we linearly decay the learning rate after each batch update from the starting value α_0 to 0. This practice is in accordance with other widely-used shallow embedding models such as *word2vec*, *doc2vec*, *fastText*, etc. In our synchronous stochastic gradient descent implementation, each observation constitutes its own batch. Therefore with e epochs and n observations, the adaptive learning rate at step l is given by

$$\alpha_l = \frac{l}{n\times e}\alpha_0. \tag{14}$$

Algorithm 1 Framework for learning embeddings

Input: Training dataset $\mathcal{D}$ with n observations, training epochs $e = 30$;
Output: The embeddings Z^I, $\tilde{Z}^I$, Z^U, Z^W and Z^X;

Shuffle $\mathcal{D}$;
for epoch $= 1, \ldots, e$ **do**
  for $q = 1, \ldots, n$ **do**
    Construct negative sample sets $Neg(\mathcal{I})$, $Neg(\mathcal{W})$ and $Neg(\mathcal{X})$ by (15);
    Update item/user/word/user feature embedding of the q^{th} observation according to (8)-(13) with learning rate $\alpha_{\text{epoch}\times n+q}$ in (14);
    Update embeddings of the instances in the above negative sample sets with same learning rate.
  end for
end for

When implementing frequency-based negative sampling, the items are sampled according to the probability computed by

$$p(i) \propto 1 - \sqrt{\frac{10^{-5}}{f(i)}}, \tag{15}$$

where $f(i)$ is the frequency of item i in training dataset [20]. Negative sampling for words and user features are conducted in the same way. We set the number of negative samples to five under all cases in training. The pseudo-code of our algorithm is given in Algorithm (1).