Negative Interactions for Improved Collaborative Filtering: Don't go Deeper, go Higher

Harald Steck
Netflix
Los Gatos, California, USA
hsteck@netflix.com

Dawen Liang
Netflix
Los Gatos, California, USA
dliang@netflix.com

ABSTRACT

The recommendation-accuracy of collaborative filtering approaches is typically improved when taking into account *higher-order* interactions [5, 6, 9–11, 16, 18, 24, 25, 28, 31, 34, 36, 41, 42, 44]. While deep nonlinear models are theoretically able to learn higher-order interactions, their capabilities were, however, found to be quite limited in practice [5]. Moreover, the use of low-dimensional embeddings in deep networks may severely limit their expressiveness [8]. This motivated us in this paper to explore a simple extension of linear *full-rank* models that allow for higher-order interactions as additional *explicit* input-features. Interestingly, we observed that this model-class obtained by far the best ranking accuracies on the largest data set in our experiments, while it was still competitive with various state-of-the-art deep-learning models on the smaller data sets. Moreover, our approach can also be interpreted as a simple yet effective improvement of the (linear) HOSLIM [11] model: by simply removing the constraint that the learned higher-order interactions have to be non-negative, we observed that the accuracy-gains due to higher-order interactions more than doubled in our experiments. The reason for this large improvement was that large *positive* higher-order interactions (as used in HOSLIM [11]) are relatively infrequent compared to the number of large *negative* higher-order interactions in the three well-known data-sets used in our experiments. We further characterize the circumstances where the higher-order interactions provide the most significant improvements.

CCS CONCEPTS

• **Information systems → Collaborative and social computing systems and tools**; • **Computing methodologies → Learning linear models**.

KEYWORDS

collaborative filtering, recommender systems, linear models, higher order interactions

ACM Reference Format:
Harald Steck and Dawen Liang. 2021. Negative Interactions for Improved Collaborative Filtering: Don't go Deeper, go Higher. In *Fifteenth ACM Conference on Recommender Systems (RecSys '21), September 27-October 1, 2021, Amsterdam, Netherlands.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3460231.3474273

RecSys '21, September 27-October 1, 2021, Amsterdam, Netherlands

ACM ISBN 978-1-4503-8458-2/21/09.
https://doi.org/10.1145/3460231.3474273

1 INTRODUCTION

Deep learning approaches have led to remarkable improvements in many applications in recent years, including collaborative filtering, e.g., [9, 19–21, 23, 26, 27, 33, 35, 43, 45]. Due to their nonlinear activation functions, deep models are able to learn higher-order interactions *implicitly* among several input-features in theory. This capability may, however, not materialize in practice [5]. To this end, various mechanisms geared to modeling higher-order interactions in deep nonlinear models have been developed, e.g., [5, 9, 10, 16, 18, 24, 25, 28, 34, 36, 41, 42, 44]. In these models, higher-order interactions are typically modelled in the latent embedding space. It was experimentally found [8, 39], however, that a (too) low dimensional embedding-space (due to the bottleneck architecture typically used) can severely degrade the prediction accuracy of the model.

This motivated us in this paper to integrate higher-order interactions into a full-rank model. For this reason, we start out with a simple linear model without a hidden layer, in particular, the Embarrassingly Shallow AutoEncoder (EASE^R) [37], a recent simplification of the SLIM model [29], which has obtained state-of-the-art ranking accuracy in the experiments in [37], even compared to deep nonlinear models. This model learns pairwise relations, i.e., between each item i in the input, and each item j in the output of the autoencoder. This is briefly reviewed in Section 2. We extend this model by *explicitly* adding higher-order relations to its input in Section 3. For instance, given that a user interacted with two items i and k in the input, item j in the output is predicted using a triplet-relation in the proposed model. Given that triplet-relations are more expressive than a sum of pairwise relations, the ranking accuracy of the resulting model significantly improved in our experiments, conducted on three well-known data sets. Compared to various baseline approaches, including several state-of-the-art deep nonlinear models, we observed competitive ranking results on the smaller data sets, while the proposed approach considerably outperformed all baseline models on the larger data set, see Section 5. The simplicity of our approach aids the explainability of the learned model: we find that most third-order relations are *negative* in our experiments, which mainly benefits the less active users. We provide a detailed analysis and characterize the circumstances where the higher-order interactions provide the most significant improvements.

2 REVIEW: PAIRWISE MODEL

This section briefly reviews the Embarrassingly Shallow AutoEncoder (EASE^R) [37], which is then extended by adding higher-order

interactions in Section 3. The EASE^R model is obtained as a simplification of the well-known SLIM model [29] by dropping the L_1-norm regularization and the non-negativity constraint on the learned parameters. In the experiments in [37], it outperformed various models that were more complex, including deep nonlinear models.

Given a set of (training-) users $\mathcal{U}$ and a set of items $\mathcal{I}$ that can be recommended to the users, let the user-item interaction-data be given in terms of the matrix $X \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{I}|}$. While the entries in this matrix may be real-valued in general, X is a binary matrix in our experiments, where 1 indicates that a user played a video/song, and 0 otherwise. The EASE^R model is based on a parameter-matrix $B \in \mathbb{R}^{|\mathcal{I}| \times |\mathcal{I}|}$ that is learned by solving the least-squares problem

$$\begin{aligned} & ||X - X \cdot B||_F^2 + \lambda_B \cdot ||B||_F^2 \\ & \text{s.t.} \quad \text{diag}(B) = 0 \end{aligned} \tag{1}$$

where $||\cdot||_F$ denotes the Frobenius norm, and λ_B is the training hyper-parameter of the L_2-norm regularization, which is tuned via cross-validation to avoid overfitting. The constraint of a zero-diagonal is crucial as to prevent the model from overfitting towards the identity matrix $B = I$, see also [29, 37]. Eq. 1 can be minimized in closed form using the method of Lagrangian multipliers as to account for the equality constraint, resulting in the solution [37]:

$$\hat{B} = I - P \cdot \text{diagMat}(1 \oslash \text{diag}(P)) \qquad \text{where} \quad P = (X^\top X + \lambda_B \cdot I)^{-1} \tag{2}$$

where diagMat$(\cdot)$ denotes a diagonal matrix, diag$(\cdot)$ the diagonal of a matrix, and $\oslash$ the elementwise division. The learned matrix $\hat{B}$ can then be used to predict the personalized scores of all the items $i \in \mathcal{I}$ for a given user $u \in \mathcal{U}$ by computing the dot-product $X_{u,\cdot} \cdot \hat{B}$, where $X_{u,\cdot}$ is the (row-) vector of the user u's past interactions with the items.

3 HIGHER-ORDER MODEL

In this section, we outline a simple yet effective *higher-order* extension of the EASE^R model from the previous section. The resulting model may also be viewed as an improvement of the HOSLIM model [11], which also captures higher-order interactions.

3.1 Data Representation

First, considering the EASE^R model in the previous section, we can see that it is based on a *pairwise* matrix B, where an entry $B_{i,j}$ captures the *pairwise* relation between item i that the user has played and the item j whose score is to be predicted. If the user played two items in the past, say, i and k, then this model predicts the score $B_{i,j} + B_{k,j}$ for item j, which is the sum of two *pairwise* terms $B_{i,j}$ and $B_{k,j}$. A model that is restricted to pairwise relations, like (i, j) and (k, j) in this running example, is obviously less expressive / powerful than a model that is able to *directly* capture higher-order relations, like the triplet-relation (i, k, j) in this example. More specifically, this triplet-relation is a relation between the pair (i, k) played by the user, and the item j whose score is to be predicted. In place of the pair (i, k), in the general case, one may use a set $\mathcal{S}$ of items played by the user for predicting item j, resulting in the higher-order relation $(\mathcal{S}, j)$ of the order $|\mathcal{S}| + 1$. Note that this notation also includes the original pairwise relation as used in EASE^R[37] when the set $\mathcal{S}$ only contains one item i.

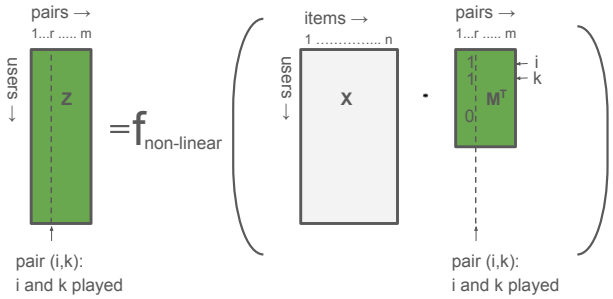

Figure 1: General framework for creating higher-order training-data Z from the given user-item interaction data X (see text for details).

A natural choice for representing higher-order relations are high-dimensional tensors. In this paper, we flatten high-dimensional tensors to matrices. In this process, we retain only the m most 'relevant' higher-order relations (e.g., by the number of occurrences in the data), as to keep the resulting matrices of manageable size for computational reasons. We explored different ways of determining the 'relevant' higher-order relations in our experiments in Section 5.1. Once we have determined a collection of 'relevant' higher-order relations $\mathcal{S}_r$ $(r = 1, ..., m)$ that we want to use, we can now define the model.

First, let us focus on the input of the model, i.e., the collections of distinct sets $\mathcal{S}_r$ $(r = 1, ..., m)$ of the higher-order relations $(\mathcal{S}_r, j)$ (i.e., we ignore items j at this step, given that the scores predicted for items j are the output of the model). To represent all these sets $\mathcal{S}_r$, we define a matrix $M \in \{0, 1\}^{m \times |\mathcal{I}|}$ as follows: row $r \in \{1, ..., m\}$ of matrix M corresponds to a $|\mathcal{S}_r|$-hot encoding of the set $\mathcal{S}_r$, i.e., $M_{r,i} = 1$ if $i \in \mathcal{S}_r$, while all other entries are set to 0. Matrix M can now be used to generate the higher-order training data $Z \in \mathbb{R}^{|\mathcal{U}| \times m}$ from the given user-item interaction-matrix X (as illustrated in Figure 1): first, the matrix multiplication $X \cdot M^\top$ is computed, and then a general nonlinear function $f_{\text{nonlinear}}$ may be applied elementwise. In our running example where set $\mathcal{S}_r$ is comprised of 2 items (i, k), this nonlinear function is simply the thresholding at 2–so that the entry $Z_{u,r}$ is 1 if and only if the user played both items i and k in set $\mathcal{S}_r$, and 0 otherwise. Other nonlinear functions besides thresholding may be used here, but are beyond the scope of this paper. The resulting matrix Z thus captures for each user u, whether the higher-order relation $\mathcal{S}_r$ is present in the user's interaction-history.

3.2 Training Objective

Corresponding to the higher-order training matrix Z, we now introduce the additional parameter-matrix $C \in \mathbb{R}^{m \times |\mathcal{I}|}$ besides the pairwise parameter-matrix B as in EASE^R. The proposed higher-order model consists of the two matrices B and C, both of which have to be learned. This model predicts the score of item j for user u according to $S_{u,j} = X_{u,\cdot} \cdot B_{\cdot,j} + Z_{u,\cdot} \cdot C_{\cdot,j}$, where $X_{u,\cdot}$ denotes row u and $B_{\cdot,j}$ refers to column j. This is illustrated in Figure 2.

For computational efficiency, we use least-squares for learning the two parameter-matrices B and C, i.e., we minimize the squared

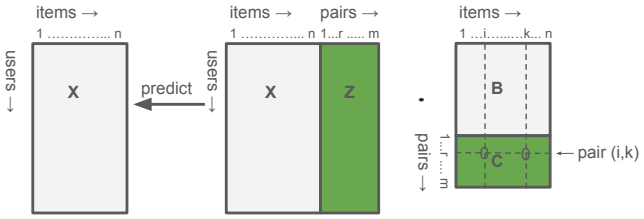

Figure 2: Learning the model-parameters B (pairwise) and C (higher-order) from the given data-matrices X and Z (see text for details).

error:

$$\begin{aligned} & \|X - XB - ZC\|_F^2 + \lambda_B \cdot \|B\|_F^2 + \lambda_C \cdot \|C\|_F^2 \\ \text{s.t.} \quad & \text{diag}(B) = 0 \\ & C \odot M = 0 \end{aligned} \tag{3}$$

where $||\cdot||_F$ denotes the Frobenius norm and $\odot$ denotes the elementwise product. We used two separate L_2-norm regularization terms as to control the 'balance' between the pairwise and higher-order part of the model (e.g., as λ_C becomes extremely large, the learned parameters in C shrink towards zero so that the resulting model essentially becomes the pairwise base-model EASE^R).

The two constraints are essential for avoiding overfitting: given that the entry $C_{r,j}$ captures the higher-order relation of $(\mathcal{S}_r, j)$, the constraint $C \odot M = 0$ on C is crucial when learning C (recall that M is pre-determined based on which higher-order relations are selected). This constraint is analogous to the constraint of a zero diagonal in the pairwise matrix B, as to avoid the trivial identity solution: as illustrated in Figure 2, if row r of C refers to the set $\mathcal{S}_r = \{i, k\}$ in our running example, then $Z_{u,r} = 1$ indicates that user u played both items i and k, and hence we may learn the trivial solution $C_{r,i} = 1$ and $C_{r,k} = 1$ (which predicts the scores of i and k based on $\{i, k\}$ themselves, instead of predicting the scores of i and k from the *other* items $\mathcal{I} \setminus \{i, k\}$). This is prevented by the constraint, as it forces $C_{r,i} = 0$ and $C_{r,k} = 0$ due to the fact that $M_{r,i} = 1$ and $M_{r,k} = 1$ by construction.

3.3 Update Equations for Training

The constrained optimization problem in Eq. 3 may be minimized by using the method of Lagrangian multipliers, together with its generalization, the Alternating Directions Method of Multipliers (ADMM) [7, 13, 15], see also [40]. To this end, we first introduce the additional matrix D and re-write Eq. 3 equivalently as follows:

$$\begin{aligned} & \|X - XB - ZC\|_F^2 + \lambda_B \cdot \|B\|_F^2 + \lambda_C \cdot \|C\|_F^2 \\ \text{s.t.} \quad & \text{diag}(B) = 0 \\ & D \odot M = 0 \\ & D = C \end{aligned} \tag{4}$$

Now we can write down the augmented Lagrangian concerning the equality constraint $C = D$:

$$\begin{aligned} L_\rho(B, C, D, \Gamma) = & \|X - XB - ZC\|_F^2 + \lambda_B \cdot \|B\|_F^2 + \lambda_C \cdot \|C\|_F^2 \\ & + 2 \cdot \rho \cdot \langle \Gamma, C - D \rangle_F + \rho \cdot \|C - D\|_F^2 \\ \text{s.t.} \quad & \text{diag}(B) = 0 \end{aligned} \tag{5}$$

where $\Gamma \in \mathbb{R}^{m \times |\mathcal{I}|}$ denotes the matrix of Lagrangian multipliers associated with the constraint $C - D = 0$. Note that most entries in Γ are actually zero–only when $M_{r,i} = 1$, we have $\Gamma_{r,i} \neq 0$, i.e., only those few entries are constrained to zero in D, while all other entries in D and C are unconstrained. Moreover, $\langle \Gamma, C - D \rangle_F$ denotes the Frobenius inner product of the matrices Γ and $C - D$. Finally, the scalar $\rho > 0$ is the so-called penalty parameter in the augmented Lagrangian, and ρ is an additional training-hyper-parameter (besides the L_2-norm regularization parameters λ_B and λ_C), which may be optimized by cross-validation.

ADMM proceeds with iterative updates, and comes with several convergence guarantees (e.g., see [7]). At iteration $k + 1$, each of the four matrices B, C, D, and Γ are updated as follows:

$$\hat{B}^{(k+1)} = \underset{B}{\text{argmin}}\, L_\rho\left(B, \hat{C}^{(k)}, \hat{D}^{(k)}, \hat{\Gamma}^{(k)}\right) \text{ s.t. } \text{diag}(B) = 0 \tag{6}$$

$$\hat{C}^{(k+1)} = \underset{C}{\text{argmin}}\, L_\rho\left(\hat{B}^{(k+1)}, C, \hat{D}^{(k)}, \hat{\Gamma}^{(k)}\right) \tag{7}$$

$$\hat{D}^{(k+1)} = (1 - M) \odot \hat{C}^{(k+1)} \tag{8}$$

$$\hat{\Gamma}^{(k+1)} = \hat{\Gamma}^{(k)} + \hat{C}^{(k+1)} - \hat{D}^{(k+1)} \tag{9}$$

The closed-form updates for $\hat{B}^{(k+1)}$ and $\hat{C}^{(k+1)}$ are given below. The update of $\hat{D}^{(k+1)}$ is simply the projection of $\hat{C}^{(k+1)}$ onto the permissible domain according to the constraint $D \odot M = 0$; in Eq. 8, the matrix of ones is denoted by 1, and $\odot$ is the elementwise product of the matrices. Note that the update of the Lagrangian multipliers in matrix $\hat{\Gamma}^{(k+1)}$ in Eq. 9 only affects those entries $\hat{\Gamma}_{r,j}^{(k+1)}$ where $M_{r,j} = 1$, as otherwise $\hat{D}_{r,j}^{(k+1)} = \hat{C}_{r,j}^{(k+1)}$.

Update of B: The optimization problem with the equality constraint in Eq. 6 can be solved in closed form using the method of Lagrangian multipliers (similar to [37]): the constraint $\text{diag}(B) = 0$ can be enforced using the vector of Lagrangian multipliers η, and adding the term $\eta^\top \cdot \text{diag}(B)$ to the augmented Lagrangian in Eq. 5. Setting its derivative w.r.t. B to zero, and solving for B, yields:

$$\hat{B}^{(k+1)} = I - P \cdot \left(X^\top Z \hat{C}^{(k)} - \text{diagMat}(\eta)\right) \tag{10}$$

$$\text{where} \quad P = (X^\top X + \lambda_B \cdot I)^{-1} \tag{11}$$

$$\eta = \frac{1 - \text{diag}(P X^\top Z \hat{C}^{(k)})}{\text{diag}(P)} \tag{12}$$

where 1 denotes a vector of ones, and the division of the vectors is elementwise in Eq. 12, yielding the solution for the vector of Lagrangian multipliers. Note that both matrices P and $PX^\top Z$ may be pre-computed.

Update of C: Setting the derivative of the augmented Lagrangian (Eq. 5) w.r.t. C to zero, and solving for C immediately yields the update-equation:

$$\hat{C}^{(k+1)} = \left(Z^\top Z + (\lambda_C + \rho) \cdot I\right)^{-1} \cdot \left(Z^\top X \cdot (I - \hat{B}^{(k+1)}) + \rho \cdot (\hat{D}^{(k)} - \hat{\Gamma}^{(k)})\right) \tag{13}$$

Note that both $(Z^\top Z + (\lambda_C + \rho) \cdot I)^{-1}$ and $Z^\top X$ may be pre-computed.

3.4 Comparison to HOSLIM

As mentioned earlier, our proposed approach can also be interpreted as an improvement to HOSLIM [11] whose training objective is similar to Eq 3:

$$\begin{aligned} & \|X - XB - ZC\|_F^2 + \lambda_B \cdot \|B\|_F^2 + \lambda_C \cdot \|C\|_F^2 + \gamma_B \cdot \|B\|_1 + \gamma_C \cdot \|C\|_1 \\ \text{s.t.} \quad & \text{diag}(B) = 0 \\ & C \odot M = 0 \\ & B_{i,j} \geq 0 \quad \forall i,j \\ & C_{r,j} \geq 0 \quad \forall r,j \end{aligned} \tag{14}$$

The two important differences are as follows:

(1) First, the learned values in B and C are constrained to be non-negative in HOSLIM [11], analogous to the SLIM model [29]. In the three well-known data sets used in our experiments in Section 5, we found the negative higher-order interactions to be actually much more important than the positive ones. Excluding them in the HOSLIM model [11] hence unnecessarily reduces ranking-accuracy by a large amount.

(2) Second, the HOSLIM model [11] also applies L_1-norm regularization to B and C as to obtain a sparse model, again analogous to the SLIM model [29]. Concerning the SLIM model [29], it was found in the various experiments in [40] that L_1-norm regularization as well as the non-negativity constraint improve prediction accuracy only for extremely small data sets (i.e., number of users $\ll$ number of items)–this might have been the reason for using them in the SLIM and HOSLIM models–but when the data sets are of a realistic size (like the three data sets in our experiments), the L_1-norm regularization did not significantly improve prediction accuracy, while the non-negativity constraint actually hurt the ranking accuracy in the experiments in [40].

Apart from that, we use a different algorithm than in HOSLIM for minimizing the training-objective, following the results in [40], where it was shown for the SLIM model that, when the item-item matrix $X^\top X$ fits into memory, the Alternating Directions Method of Multipliers (ADMM) [7, 13, 15] can be more effective in minimizing such a constrained least-squares problem–in term of training time as well as the prediction accuracy of the learned model.

3.5 Possible Extensions

Throughout the section, we use a running example where $\mathcal{S}_r$ consists of two items, which means the model is capable of capturing triplet-relations. A straightforward yet effective extension is to increase the size of $\mathcal{S}_r$ to incorporate even higher-order interactions. Unfortunately this will also increase the difficulty of selecting the 'relevant' relations to construct the matrix M in Section 3.1. Furthermore, it is not unreasonable to assume diminishing returns as we add 4^{th}- or even higher-order interactions.

Alternatively, we can rewrite the matrix notation of the pairwise model in Eq 1 into the so-called *auto-normal* parametrization [3, 4, 38]: The model predicts the score of item j for user u as $S_{u,j} = \sum_{i \neq j} X_{u,i} B_{i,j}$. Similar to Factorization Machines [31, 32], this auto-normal parametrization can be extended to include a (factorized) triplet-relation in the form of a D-dimensional latent factor model ($D \ll |\mathcal{I}|$):

$$S_{u,j} = \sum_{i \neq j} X_{u,i} B_{i,j} + \sum_{\substack{i<k \\ i \neq j, k \neq j}} X_{u,i} X_{u,k} \Big(\sum_{d=1}^{D} \vec{v}_d^{(j)} \vec{\gamma}_d^{(i)} \vec{\gamma}_d^{(k)} \Big)$$

where both $\vec{v}^{(j)} \in \mathbb{R}^D$ and $\vec{\gamma}^{(i)}, \vec{\gamma}^{(k)} \in \mathbb{R}^D$ are latent factors which will be estimated alongside the parameter-matrix B. Note that we distinguish the factors associated with the item j in the output ($\vec{v}^{(j)}$) with the items i, k in the input ($\vec{\gamma}^{(i)}, \vec{\gamma}^{(k)}$). One advantage of this factorized formulation is that we are able to incorporate all possible triple-relations, instead of being constrained with only a pre-selected set of m 'relevant' relations. The triplet interaction term can be efficiently computed in linear time w.r.t. both $|\mathcal{I}|$ and D similar to Factorization Machines. Making use of the recent advance in Polynomial Networks [6], this formulation can be extended beyond triplet-relations with tractable computation. We leave this promising direction as future work.

4 RELATED WORK

It is common practice to add higher-order interaction terms to the linear regression model, which perhaps is the simplest linear model. Depending on the properties of the data set, this can considerably improve prediction accuracy. While the higher-order features are nonlinear functions of the original features in the training-data, the advantage of this approach is that the resulting model remains linear in the parameter space. As a consequence, it is efficient to train and to make predictions.

This idea of adding higher-order interactions has been applied to linear collaborative filtering approaches in the past: for instance, the SLIM model [29] was extended to the HOSLIM model [11], and the Factorization Machine [31, 32] was extended in [6]. For both of these linear model-classes, significant improvements in ranking accuracy were observed when adding higher-order interactions [6, 11].

In this paper, we extended the EASE$^{\text{R}}$ model [37] by adding higher-order relations. Given that EASE$^{\text{R}}$ is a simplified version of SLIM, the proposed approach in this paper can be understood as a simplification of HOSLIM, namely, we removed the non-negativity and sparsity constraints, as outlined in Section 3.4. This not only reduces the computational cost of training this model, but also results in significant improvements of the ranking-accuracy (unless the training data are extremely small) — The interested reader is referred to [40] for a detailed comparison of the pairwise models EASE$^{\text{R}}$ and SLIM and how each of the two constraints impacts the ranking-accuracy. While the removal of the non-negativity constraint was crucial in the pairwise model, we found in our experiments that it is even more important when incorporating higher-order interactions.

Deep-learning approaches, having shown remarkable performances in numerous areas (mostly around computer vision and natural language processing), also entered the field of collaborative filtering in recent years. Many network architectures have been developed to solve various problems, e.g., [9, 19–21, 23, 26, 27, 33, 35, 43, 45]. A commonality of the various approaches for collaborative filtering problems is their relatively shallow architecture (of typically one to three hidden layers), which was empirically found to obtain the best ranking-accuracy, e.g., [19, 23, 26, 27, 33, 35, 45].

Furthermore, dense low-dimensional embeddings are commonly learned, whereas a sparse and relatively high-dimensional representation is used in SW-DAE [23]. Note that the latter is the strongest baseline in our experiments. Our proposed model uses an even higher-dimensional representation for an item i, i.e., column i in the stacked matrices B and C, see Figure 2.

Even though deep non-linear models are able to learn higher-order interactions (as well as any non-linearities) in theory, in practice their capabilities were, however, found to be quite limited [5]. Several different approaches for overcoming this limitation were proposed in recent years. From one perspective, providing explicit interactions in the input layer (e.g., [9, 16, 25]), may be distinguished from learning / modeling interactions in the latent-embedding space (e.g., [5, 41, 42]). From another perspective, one may categorize the approaches by the modeling technique used, like extending the ideas underlying the factorization machine (FM) [31] to deep networks [16, 18, 25, 44], using different variants of the attention mechanism [24, 36, 44], crossing latent embeddings in a multiplicative way [5, 41, 42], or deploying a logarithmic transform layer [10]. A third perspective might be regarding the order of the interactions that can (easily) be modelled: while FM-based approaches are typically restricted to feature-interactions up to second order[1], various other approaches allow for much higher orders. However, [28] argues that third (and higher) order feature-interactions may not be worth the extra computational cost. Note that in our experiments we also did not find statistically significant improvements when adding third (or higher) order feature-interactions. Finally, most of these approaches consider higher-order interactions between additional features and user-item interaction data, while we primarily consider interactions within the items that a user interacted with.

In our proposed approach, the selection of the explicit interactions is automated, hence avoiding the need for feature-engineering. Moreover, the linear nature of our approach allows for easier interpretability / explainability of the learned parameters regarding the interactions. This enables us to obtain deeper insights toward the model's performance.

Apart from that, the use of (too) low dimensional embeddings can result in a significant loss of relevant information that can be propagated through a deep network [8, 39], resulting in reduced ranking-accuracy. This suggests that learning interactions in the low-dimensional embedding space might be less effective than using explicit feature-interactions. This was an additional motivation for us to explore the proposed full-rank model.

Finally, in a meta-analysis [12], it was shown that many of these deep-learning approaches are not easily reproducible and even when they are, they can often be outperformed by carefully-tuned linear methods. We hope that our work provides additional insights into the comparison of deep models vs. simple linear baselines and offers an alternative route to improve the expressive power of these models.

[1]Note that a second-order feature-interaction corresponds to a third-order relation (as used in this paper) among the (two) input features and the target/output-variable.

5 EXPERIMENTS

In this section, we study the performance of our proposed approach both quantitatively and qualitatively. We highlight the following results:

- We demonstrate empirically that a relatively small number of higher-order relations is sufficient to obtain considerable improvements over the pairwise base-model (EASE^R) in our experiments on three well-known data sets. Moreover, the proposed model outperforms the various deep nonlinear models on the larger data set in our experiments, while remains competitive on the smaller data sets. To this end, we introduce the notion of *effective catalog size*, as to understand the performance differences between our approach and deep nonlinear models across the datasets.
- We decompose the predicted scores into the pairwise and higher-order components and observe that the higher-order components are skewed towards negative values. By forcing the higher-order components to be positive (as in HOSLIM), we observe a considerable reduction in ranking-accuracies.
- We break down the performance gains of our proposed model by user activity and to our surprise, the users with relatively low activity have a sizable gain, while the users with many interactions mostly maintain the same performance.

The source code is available at `https://github.com/hasteck/Higher_RecSys_2021`.

5.1 Experimental Protocols

For reproducibility and for a valid comparison to the results in [26], we run the same experiments as in [26], using their code[2] for pre-processing and filtering the data, as well as for evaluating the learned models. Besides Recall@20, Recall@50, and normalized Discounted Cumulative Gain (nDCG@100), which were used in [26], we also report Recall@10 which provides additional insights towards the head of the rankings. We also use the same three data sets as in [26]: MovieLens 20 Million (***ML-20M***) data [17], Netflix Prize (***Netflix***) data [1], and Million Song Data (***MSD***) [2]. For comparison, Table 1 also shows the results of all the various models evaluated in [26], which now serve as baselines:

- Sparse Linear Method (**Slim**) [29]: This is the original pairwise model, which was simplified to the EASE^R model in [37]. We described it in Section 2.
- Weighted Matrix Factorization (**wmf**) [22, 30]: A linear low-dimensional model.
- Collaborative Denoising Autoencoder (**cdae**) [43]: A non-linear model with one hidden layer.
- Denoising Autoencoder (**Mult-dae**) and Variational Autoencoder (**Mult-vae ^{PR}**) [26]: Both are deep non-linear autoencoders. The following architecture was found to result in the best ranking accuracy in [26]: input layer $\rightarrow$ 600-dimensional layer $\rightarrow$ 200-dimensional layer $\rightarrow$ 600-dimensional layer $\rightarrow$ output layer.

In Table 1, we additionally report the results of the following recent improvements of autoencoders, which reported results on the same three data sets as used in our experiments:

[2]`https://github.com/dawenl/vae_cf`

- RecVAE [35]: improves on MULT-VAE^{PR} [26] in several ways, including a user-dependent β in the β-VAE model, a composite prior, and denoising, among others.
- RaCT [27]: an actor-critic based learning-to-rank approach, built on top of the variational autoencoder.
- SW-DAE [23]: a sparse autoencoder, where the connectivity to the (single) hidden layer is learned by clustering the items based on similarity.

Apart from these baselines, we also compare with a baseline similar to HOSLIM [11] by forcing the higher-order parameter matrix C to be non-negative, which is the key difference (see Section 3.4).[3]

In our experiments, we trained the proposed model with ADMM for 40 iterations, which was sufficient for convergence (see also [7]). We optimized the three training hyper-parameters ρ, λ_B, and λ_C in Eq. 5 by grid-search for each model and data-set. Not surprisingly, the optimal λ_B in the higher-order model in Eq. 3 (and equivalently Eq. 5) was essentially identical to the optimal λ_B in the pairwise model in Eq. 1, and took the values of about 500 on *ML-20M*, 1,000 on *Netflix*, and 200 on *MSD* data.

While the pairwise relations are of the form that one item is used to predict the score of the other one, the triplet-relations are such that a *set of two items* is used to predict the score of the third one. To determine such sets, we obtained the best ranking-accuracies when we applied a threshold t to the strictly upper triangular matrix of the item-item matrix $X^\top X$,[4] where X is the given user-item interaction training-data, i.e., the most frequent pairs were selected. By changing the threshold value t, the number of 'relevant' higher-order interactions m can be controlled. The restriction to the strictly upper triangular matrix ensures that we obtain *distinct* sets of two items–in other words, recall the selection matrix M defined in Secion 3.1, if $(X^\top X)_{i,k} > t$ for $i < k$, we have a row r in M where $M_{r,i} = 1$ and $M_{r,k} = 1$.

5.2 Experimental Results

Table 1 shows that the ranking accuracy on the test data is improved beyond the pairwise base-model (EASE^R) as we include m triplet-relations (pairs + triplets). We can see that already a quite small number of 500 triplet-relations leads to improvements over the pairwise model, while 40,000 triplet-relations yield significant improvements. Moreover, when m is increased by a factor of 4-5 from one line to the next in Table 1, we observe diminishing returns in terms of gains in ranking accuracy. Note that even $m = 40,000$ represents only 0.02% of all possible sets of two items on the *ML-20M* data, which suggests that a small number of higher-order relations is sufficient to obtain most of the improvements. Finally, when we constrain the higher-order weights to be non-negative ('pairs + triplets ≥ 0', like in HOSLIM), we can observe a considerable reduction in the gains. We provide a detailed discussion in Section 5.3.

We also experimented with including 4^{th} order relations in addition to the pairwise and triplet relations. To this end, we applied a threshold to the matrix $X^\top Z$, which was already computed when training the model that contains triplet relations. This is obviously a greedy approach for determining higher-order relations, and is hence not guaranteed to find all relevant higher-order interactions, but has the advantage of being computationally tractable. In our experiments, we did, however, not observe any significant improvements when adding 4^{th} order relations on top of the pairwise and triplet relations.

Table 1 also shows that our proposed approach yields state-of-the-art results compared to the various baseline models: it outperforms all models evaluated in [26] on all three data sets. Moreover, it is also competitive compared to the various deep non-linear autoencoders (RecVAE [35], RaCT [27], and SW-DAE [23]): Table 1 shows that the proposed model with 40,000 triplet-relations outperforms all three baselines on the largest data set (*MSD*) by a large margin (5.4% better than the *best* baseline, SW-DAE), while it is only slightly worse on the two small data sets (2.6% on *ML-20M* and 0.7% on *Netflix* compared to the *best* baseline).

Effective Catalog Size: We believe that the differences in the ranking-results on the three data sets are due to two reasons: (1) the catalog size of the *MSD* data is about twice as large as the one of the other two data sets (see data-set properties in Table 1), and (2) the (approximate) power-law distribution of the item-popularities drops off very quickly for the two data sets *ML-20M* and *Netflix*, i.e., they contain a small number of very popular items (head), and a large number of unpopular items (long tail). In contrast, the power-law distribution in the *MSD* data is less extreme. We can capture both effects by the *catalog entropy*:

$$H = -\sum_{i \in I} p_i \log p_i$$

where p_i is the normalized popularity of item i (i.e., the number of users who played it). Given that the entropy may be difficult to understand intuitively, we ask the following question: considering a catalog, where all items have the *same popularity*, how many items does this catalog have to contain so that its entropy is the same as the one of the real-world catalog with a power-law distribution. It can easily be seen that this *effective catalog size* is given by

$$n = \exp(H),$$

where H is the entropy of the real-world catalog from above. As the effective catalog size shrinks, the recommendation problem becomes easier, as the recommender system can rely increasingly on the (unpersonalized) item-popularities. Conversely, as the effective catalog size increases, high ranking accuracies can only be obtained by making highly personalized recommendations, which is a more difficult task. Also note that the effective catalog size is different from other metrics in the literature, like catalog coverage, which is a property of the recommender system (and not of the data), e.g., see [14]. Table 1 shows that the effective catalog size of the *MSD* data is about *seven* times larger than the one of the other two data sets, even though the actual catalog sizes differ by only a factor of

[3] The only difference between this baseline and HOSLIM is that we dropped the two L_1-norm regularization terms from Eq. 14. This reduces the number of (interdependent) hyperparameters that need to be tuned from four to two, and hence greatly saves computation-time, resulting in better scalability.

[4] We also tried other strategies for determining the relevant higher-order relations, like first standardizing the training matrix X (such that each column has zero mean and unit variance), and then thresholding the resulting item-item matrix $X^\top X$, i.e., correlation matrix. This focuses more on similar items, irrespective of their popularities. In our experiments, however, this was inferior to using the original matrix X, which suggests that the effectiveness of the approach also hinges on the fact that the higher-order relations included in the model occur in a large number of users.

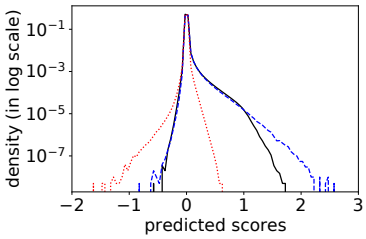

Figure 3: Distribution of the scores predicted by the higher-order model on the test set (black line), and split into the scores' components predicted from pairwise relations (blue dashed line) and triplet-relations (red dotted line): while the pairwise component is skewed to positive values (dashed), the triplet-component is skewed to *negative* values (dotted). We obtained these figures using $m = 40,000$ higher-order relations on the *ML-20M* data, and qualitatively similar figures on the other two data sets. See text for details.

about *two*. Given that the effective catalog size of the *MSD* data is so much larger than the one of the other two data sets, this indicates that the *MSD* data is a considerably more challenging data set than the other two.

This suggests that the model has to be considerably more expressive on the *MSD* data than on the other two datasets to be able to capture all the relevant item-item relations. We suspect that the bottleneck architecture (low-dimensional latent factors) of deep models significantly restricts their ability in the case with a large effective catalog size (see also [8, 39] for relevant discussions). A remedy might be to increase the dimensionality of the latent factors, but this may largely be prohibitive due to the corresponding increase in training time (SW-DAE [23] alleviates this problem by learning a *sparse* high-dimensional bottleneck layer and it outperforms other baselines by a large margin on *MSD*). The bottleneck-architecture may hence constitute a practical limitation of deep models for collaborative filtering problems, and severely limit their ability to learn the relevant relations among an *effectively-large* number of items.

The linear model considered in this paper, on the other hand, avoids this problem, as it is a *full-rank* model, i.e., it explicitly learns each pairwise and (a pre-selected subset of) higher-order relations. For this reason, the linear model has many more parameters than the deep models, which gives it sufficient expressive power to fit the data better (while overfitting can be controlled by L_2-norm regularization). From this perspective, increasing the number of 'relevant' higher-order interactions in the model also increases its number of parameters, further improving the expressive power of the model, which could partially explain the resulting improvements in ranking-accuracy observed in Table 1.

5.3 Exploratory Analysis

In this section, we take a deep dive into the model, and answer the following two questions:

What is captured by the higher-order component of the model? Figure 3 provides deeper insights into the learned higher-order model, in particular, how the predicted scores decompose into their pairwise and higher-order components. Using the test-data $X^{(\text{test})}$ and $Z^{(\text{test})}$, we computed the score-matrix $S^{(\text{test})} = X^{(\text{test})} \cdot \hat{B} + Z^{(\text{test})} \cdot \hat{C}$, which is comprised of all the scores predicted by the learned pairwise and higher-order components of the model for all the test-users regarding all available items. In addition, we also computed its two components separately, the scores $S^{(\text{test})}_{\hat{B}} = X^{(\text{test})} \cdot \hat{B}$ based on the learned pairwise matrix $\hat{B}$ (in the 'pairs + triplets' model), and the scores $S^{(\text{test})}_{\hat{C}} = Z^{(\text{test})} \cdot \hat{C}$ based on the learned higher-order relations. Figure 3 shows the histograms of the scores in these three matrices: $S^{(\text{test})}$, $S^{(\text{test})}_{\hat{B}}$, and $S^{(\text{test})}_{\hat{C}}$. We can see that the scores based on the pairwise component of the model (dashed line) are skewed towards positive values,[5] while the scores based on the triplet-component (dotted line) are skewed towards negative values. This suggests that many of the triplet-relations provide *downward* corrections to the scores that are predicted by summing up the pairwise terms, while only few triplet-relations provide *upward* corrections.

This also makes intuitive sense, which may be seen in the following example. If there exists a group of 'similar' videos, where the predicted scores should ideally be independent of the number of videos that the user has already played from this group, then the higher-order *downward* corrections enable the model to learn this. This is also true if the scores should ideally grow in a sub-linear way with the number of videos that the user has played from this group. Note that summing up the pairwise relations in the model results in an approximate linear increase in the scores within a group. Conversely, if there exists a group of 'similar' videos, where the predicted scores should ideally grow in a super-linear way with the number of videos that the user has already watched from this group, then the higher-order *upward* corrections enable the model to provide such an extra boost to the scores on top of the pairwise relations. Groups that benefit from *downward* corrections are apparently more prevalent in the data sets in our experiments than are the groups that benefit from *upward* corrections, cf. skew of the dotted line in Figure 3. This is also supported by the fact that the scores predicted by the higher-order model (solid line) are less spread out than the scores of its pairwise component (dashed line), where in particular many large positive values are considerably reduced.

This is a fundamental difference to the HOSLIM model [11], where the higher-order weights are constrained to be non-negative, and hence are (unnecessarily) restricted to provide only an *upward* correction to the scores predicted by the pairwise component. When we constrained the higher-order weights to be non-negative in our model (like in HOSLIM), the gains were considerably reduced, see 'pairs + triplets ≥ 0' compared to 'pairs + triplets' in Table 3.

How does adding higher-order interactions help? It seems reasonable to assume that by adding higher-order interactions into the model, the users who interact with many items will benefit more than the users who only interact with a small amount of items, as the former group has more triplet-relations to take advantage of the higher-order term. To verify this hypothesis, we plot the test

[5] Note that this does not contradict the finding in [37] that 60% of the pairwise weights were negative: first, scores and weights are not the same; second, 60% refers to the count (of weights) in [37], while here the skew refers to values (of the scores), i.e., some large positive values.

Table 1: Ranking accuracy on the test set when a different number m of higher-order relations are included in the model, compared to the pairwise model and various baselines (with standard errors $\approx$ 0.002, 0.001, and 0.001 on *ML-20M*, *Netflix*, and *MSD* data, respectively).

	ML-20M				*Netflix*				*MSD*			
models	Recall @10	Recall @20	Recall @50	nDCG @100	Recall @10	Recall @20	Recall @50	nDCG @100	Recall @10	Recall @20	Recall @50	nDCG @100
pairs (EASE^R)	0.336	0.391	0.521	0.420	0.344	0.362	0.445	0.393	0.297	0.333	0.428	0.389
pairs + triplets (m = 500)	0.338	0.394	0.524	0.423	0.345	0.363	0.447	0.395	0.297	0.334	0.429	0.390
pairs + triplets (m = 2k)	0.340	0.396	0.526	0.425	0.347	0.366	0.448	0.397	0.298	0.334	0.429	0.390
pairs + triplets (m = 10k)	0.343	0.400	0.530	0.429	0.349	0.367	0.450	0.399	0.298	0.335	0.430	0.391
pairs + triplets (m = 40k)	**0.344**	0.402	0.534	0.431	**0.351**	**0.369**	0.453	0.401	**0.300**	**0.337**	**0.431**	**0.392**
pairs + triplets ≥ 0 (m = 40k)	0.340	0.396	0.527	0.426	0.346	0.365	0.448	0.396	0.298	0.335	0.429	0.390
reproduced from [26]:												
SLIM	—	0.370	0.495	0.401	—	0.347	0.428	0.379	— did not finish in [26] —			
WMF	—	0.360	0.498	0.386	—	0.316	0.404	0.351	—	0.211	0.312	0.257
CDAE	—	0.391	0.523	0.418	—	0.343	0.428	0.376	—	0.188	0.283	0.237
MULT-VAE ^{PR}	—	0.395	0.537	0.426	—	0.351	0.444	0.386	—	0.266	0.364	0.316
MULT-DAE	—	0.387	0.524	0.419	—	0.344	0.438	0.380	—	0.266	0.363	0.313
further baselines:												
RecVAE [35]	—	**0.414**	**0.553**	**0.442**	—	0.361	0.452	0.394	—	0.276	0.374	0.326
RaCT [27]	—	0.403	0.543	0.434	—	0.357	0.450	0.392	—	0.268	0.364	0.319
SW-DAE [23]	—	0.410	0.549	**0.442**	—	**0.370**	**0.458**	**0.404**	—	0.317	0.416	0.372
data-set properties:												
# interactions			10 mil.				57 mil.				34 mil.	
# users			136,677				463,435				571,355	
catalog size (# items)			20,108				17,769				41,140	
catalog entropy H			7.719				7.968				9.897	
effective catalog size: exp(H)			2,252				2,887				19,875	

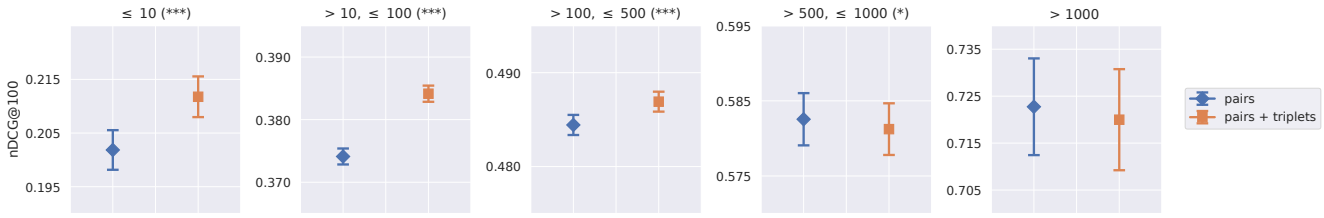

Figure 4: Test nDCG@100 breakdown for users with increasing levels of activity. The title of each subplot indicates the range of items the users in this group interacted with in the test dataset. The error bars represent one standard error. For each subplot, a paired t-test is performed and * indicates statistical significance at α = 0.05 level, ** at α = 0.01 level, and * at α= 0.001 level. The subplot with no * means that a statistically significant result is not obtained at any α level. We can see that by adding higher-order interactions into the model, users who interact with a relatively small amount of items have a sizable gain in ranking-accuracies, while users who interact with many items mostly stay unchanged. We obtained these figures using $m = 40,000$ higher-order relations on the *Netflix* data, and qualitatively similar figures on the other two data sets.**

nDCG@100 for groups of users with different levels of activity in Figure 4, for the pairwise base-model (EASE^R) and the higher-order model (pairs + triplets). The title of each subplot indicates the range of items the users in this group interacted with in the test dataset. Interestingly, we observe that most of the gains in performance come from the low-activity users, while the performance for high-activity users mostly remains unchanged. We also performed a paired t-test on each group (see the caption of Figure 4 for details) and obtained statistically significant results for the groups with mid-to-low activities.

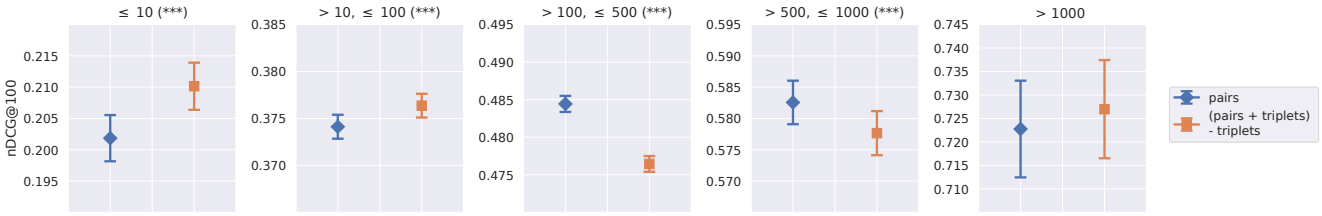

Figure 5: See the caption of Figure 4 for more details. We can see that the pairwise component of the 'pairs + triplets' model is capable of much better modeling the users in the low-activity buckets. On the other hand, the pairwise component alone (i.e., without the triplet-component) is significantly outperformed by the pairwise base-model (EASE^R) for users with high-activity level. This supports our hypothesis that the pairwise component focuses on low-activity users, while the triplet-component focuses on high-activity users.

We hypothesize that the reason for these seemingly counter-intuitive results is the 'explain-away' effect of the higher-order components. From the spread of the histograms in Figure 3 we can see that most of the triplet-relations provide *downward* corrections, which mainly kick in for high-activity users, hence explaining away the higher-order interactions from the data. This enables the pairwise model to better focus on relatively low-activity users without the interference of the higher-order interactions. To validate this hypothesis, we perform an additional comparison on the same dataset as in Figure 4, this time between the pairwise base-model (EASE^R) and the pairwise component of the 'pairs + triplets' model, denoted as '(pairs + triplets) - triplets' in Figure 5. In aggregation, both achieve almost identical ranking accuracy in terms of nDCG@100. However, we can clearly see that the pairwise component of the 'pairs + triplets' model is capable of much better modeling the users in the low-activity buckets. On the other hand, the base-model (EASE^R) is capable of significantly outperforming the pairwise component for users with high-activity level,[6] demonstrating the importance of the higher-order component of the model for the high-activity users.

6 CONCLUSIONS

In this paper, we extended a state-of-the-art linear model EASE^R[37] that learns pairwise relations among the items, by adding higher-order interactions. The resulting model enables efficient optimization via the Alternating Directions Method of Multipliers (ADMM) [7, 13, 15] on realistically-sized datasets commonly used in the literature. Despite the simplicity of the proposed approach, we saw significant gains in ranking-accuracy compared to the pairwise model. Moreover, the proposed approach obtained competitive results compared to the various state-of-the-art baselines in our experiments, which includes several deep nonlinear autoencoders–in particular, we found that the proposed approach outperformed all the baselines by a large margin on the largest of the three data sets in our experiments. To this end, we introduced the notion of *effective catalog size* to understand the performance differences between our approach and deep nonlinear models across the datasets.

Furthermore, we observed that the majority of the learned triplet-relations were negative, while the learned pairwise relations were more positive in our experiments. This suggests that many of the triplet-relations provide a *downward* correction to the predicted scores, which may otherwise be over-predicted when summing up the pairwise terms.

Finally, we broke down the performance gains by user groups of various activity levels and observed that the users with relatively low activity have a more sizable gain, while the users with many interactions mostly remain at the same level of ranking accuracy. We hypothesize that the addition of the triplet-relations 'explained away' the higher-order interactions among the active users, which enables the pairwise component to better focus on relatively low-activity users without the interference of the higher-order interactions.

[6] While this does not hold for the mean for the users with >1000 activities in Figure 5, note that this is not significant due to the large error bars.

REFERENCES

[1] J. Bennet and S. Lanning. 2007. The Netflix Prize. In *Workshop at SIGKDD-07, ACM Conference on Knowledge Discovery and Data Mining.*
[2] T. Bertin-Mahieux, D.P.W. Ellis, B. Whitman, and P. Lamere. 2011. The Million Song Dataset. In *International Society for Music Information Retrieval Conference (ISMIR).*
[3] J. Besag. 1975. Statistical Analysis of Non-Lattice Data. *The Statistician* 24 (1975), 179–95.
[4] J. Besag. 1977. Efficiency of pseudo-likelihood estimation for simple Gaussian fields. *Biometrika* 64 (1977).
[5] A. Beutel, P. Covington, S. Jain, C. Xu, J. Li, V. Gatto, and E.H. Chi. 2018. Latent Cross: Making Use of Context in Recurrent Recommender Systems. In *ACM Conference on Web Search and Data Mining (WSDM).*
[6] M. Blondel, A. Fujino, N. Ueda, and M. Ishihata. 2016. Higher-Order Factorization Machines. In *Advances in Neural Information Processing Systems (NIPS).*
[7] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. 2011. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Found. Trends Mach. Learn.* 3 (2011), 1–122.
[8] T. Chen, J. Lin, T. Lin, S. Han, C. Wang, and D. Zhou. 2018. Adaptive Mixture of Low-Rank Factorizations for Compact Neural Modeling. In *Advances in Neural Information Processing Systems (NIPS).*
[9] H.T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, and H. Shah. 2016. Wide & Deep Learning for Recommender Systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS).* 7–10.
[10] W. Cheng, Y. Shen, and L. Huang. 2020. Adaptive Factorization Network: Learning Adaptive-Order Feature Interactions. arXiv:1909.03276.
[11] E. Christakopoulou and G. Karypis. 2014. HOSLIM: Higher-Order Sparse Linear Method for Top-N Recommender Systems. In *18th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD).*
[12] M. F. Dacrema, P. Cremonesi, and D. Jannach. 2019. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In *ACM Conference on Recommender Systems (RecSys).*
[13] D. Gabay and B. Mercier. 1976. A dual algorithm for the solution of nonlinear variational problems via finite element approximations. *Computers and Mathematics with Applications* 2 (1976), 17–40.

[14] M. Ge, C. Delgado-Battenfeld, and D. Jannach. 2010. Beyond Accuracy: Evaluating Recommender Systems by Coverage and Serendipity. In *ACM Conference on Recommender Systems (RecSys)*.

[15] R. Glowinski and A. Marrocco. 1975. Sur l'approximation, par elements finis d'ordre un, et la resolution, par penalisation–dualite, d'une classe de problems de Dirichlet non lineares. *Revue Francaise d'Automatique, Informatique, et Recherche Operationelle* 9 (1975), 41–76.

[16] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. In *Int. Joint Conf. on Artificial Intelligence (IJCAI)*.

[17] F. M. Harper and J. A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 5 (2015). Issue 4.

[18] X. He and T.-S. Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*.

[19] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua. 2017. Neural Collaborative Filtering. In *International World Wide Web Conference (WWW)*.

[20] B. Hidasi and A. Karatzoglou. 2017. Recurrent Neural Networks with Top-k Gains for Session-based Recommendations. In *International Conference on Information and Knowledge Management (CIKM)*. arXiv:1706.03847.

[21] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk. 2015. Session-based Recommendations with Recurrent Neural Networks. arXiv:1511.06939.

[22] Y. Hu, Y. Koren, and C. Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *IEEE International Conference on Data Mining (ICDM)*.

[23] F. Khawar, L. Poon, and N. L. Zhang. 2020. Learning the Structure of Auto-Encoding Recommenders. In *International World Wide Web Conference (WWW)*.

[24] Z. Li, W. Cheng, Y. Chen, H. Chen, and W. Wang. 2020. Interpretable Click-Through Rate Prediction through Hierarchical Attention. In *ACM Conference on Web Search and Data Mining (WSDM)*.

[25] J. Lian, X. Zhou, F. Zhang, Z. Chen, X. Xie, and G. Sun. 2018. xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems. In *ACM Conference on Knowledge Discovery and Data Mining (KDD)*.

[26] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *International World Wide Web Conference (WWW)*.

[27] S. Lobel, C. Li, J. Gao, and L. Carin. 2020. RaCT: Towards amortized ranking-critical training for collaborative filtering. In *Int. Conference on Learning Representations (ICLR)*.

[28] M. Naumov, D. Mudigere, H.-J.M. Shi, J. Huang, N. Sundaraman, J. Park, X. Wang, U. Gupta, C.-J. Wu, A.G. Azzolini, D. Dzhulgakov, A. Mallevich, I. Cherniavskii, Y. Lu, R. Krishnamoorthi, A. Yu, V. Kondratenko, S. Pereira, X. Chen, W. Chen, V. Rao, B. Jia, L. Xiong, and M. Smelyanskiy. 2019. Deep Learning Recommendation Model for Personalization and Recommendation Systems. arXiv:1906.00091.

[29] X. Ning and G. Karypis. 2011. SLIM: Sparse Linear Methods for Top-N Recommender Systems. In *IEEE International Conference on Data Mining (ICDM)*. 497–506.

[30] R. Pan, Y. Zhou, B. Cao, N. Liu, R. Lukose, M. Scholz, and Q. Yang. 2008. One-Class Collaborative Filtering. In *IEEE International Conference on Data Mining (ICDM)*.

[31] S. Rendle. 2010. Factorization machines. In *IEEE International Conference on Data Mining (ICDM)*. 995–1000.

[32] S. Rendle. 2012. Factorization machines with libFM. *ACM Transactions on Intelligent Systems and Technology (TIST* 3 (2012). Issue 3.

[33] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie. 2015. AutoRec: Autoencoders meet Collaborative Filtering. In *International World Wide Web Conference (WWW)*.

[34] Y. Shan, T.R. Hoens, J. Jiao, H. Wang, D. Yu, and J.C. Mao. 2016. Deep Crossing: Web-Scale Modeling without Manually Crafted Combinatorial Features. In *ACM Conference on Knowledge Discovery and Data Mining (KDD)*.

[35] I. Shenbin, A. Alekseev, E. Tutubalina, V. Malykh, and S. I. Nikolenko. 2020. RecVAE: a New Variational Autoencoder for Top-N Recommendations with Implicit Feedback. In *ACM Conference on Web Search and Data Mining (WSDM)*.

[36] W. Song, C. Shi, Z. Xiao, Z. Duan, Y. Xu, M. Zhang, and J. Tang. 2019. AutoInt: Automatic Feature Interaction Learning via Self-Attentive Neural Networks. In *International Conference on Information and Knowledge Management (CIKM)*.

[37] H. Steck. 2019. Embarrassingly Shallow Autoencoders for Sparse Data. In *International World Wide Web Conference (WWW)*.

[38] H. Steck. 2019. Markov Random Fields for Collaborative Filtering. In *Advances in Neural Information Processing Systems*. 5474–5485.

[39] H. Steck. 2020. Autoencoders that don't overfit towards the identity. In *Advances in Neural Information Processing Systems (NIPS)*.

[40] H. Steck, M. Dimakopoulou, N. Riabov, and T. Jebara. 2020. ADMM SLIM: Sparse Recommendations for Many Users. In *ACM Conference on Web Search and Data Mining (WSDM)*.

[41] R. Wang, B. Fu, G. Fu, and M. Wang. 2017. Deep & Cross Network for Ad Click Predictions. In *Proceedings of the ADKDD*.

[42] R. Wang, R. Shivanna, D.Z. Cheng, S. Jain, D. Lin, L. Hong, and E.H. Chi. 2020. DCN V2: Improved Deep & Cross Network and Practical Lessons for Web-scale Learning to Rank Systems. arXiv:2008.13535.

[43] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester. 2016. Collaborative Denoising Auto-Encoders for top-N Recommender Systems. In *ACM Conference on Web Search and Data Mining (WSDM)*.

[44] J. Xiao, H. Ye, X. He, H. Zhang, F. Wu, and T.-S. Chua. 2017. Attentional Factorization Machines: Learning the Weight of Feature Interactions via Attention Networks. In *Int. Joint Conf. on Artificial Intelligence (IJCAI)*.

[45] Y. Zheng, B. Tang, W. Ding, and H. Zhou. 2016. A Neural Autoregressive Approach to Collaborative Filtering. In *International Conference on Machine Learning (ICML)*.