User preference and embedding learning with implicit feedback for recommender systems

Sumit Sidana, Mikhail Trofimov, Oleh Horodnytskyi, Charlotte Laclau, Yury Maximov, Massih-Reza Amini

User Preference and Embedding Learning with Implicit Feedback for Recommender Systems

Sumit Sidana[a], Mikhail Trofimov[b], Oleh Horodnytskyi[c], Charlotte Laclau[a], Yury Maximov[c,d], Massih-Reza Amini[a]

[a] *University Grenoble Alpes CNRS/LIG, France*
[b] *Federal Research Center "Computer Science and Control" of Russian Academy of Sciences*
[c] *Skolkovo Institute of Science and Technology, Russia*
[d] *Theoretical Division T-5/CNLS, Los Alamos National Laboratory, USA*

Abstract

In this paper, we propose a novel ranking framework for collaborative filtering with the overall aim of learning user preferences over items by minimizing a pairwise ranking loss. We show the minimization problem involves dependent random variables and provide a theoretical analysis by proving the consistency of the empirical risk minimization in the worst case where all users choose a minimal number of positive and negative items. We further derive a Neural-Network model that jointly learns a new representation of users and items in an embedded space as well as the preference relation of users over the pairs of items. The learning objective is based on three scenarios of ranking losses that control the ability of the model to maintain the ordering over the items induced from the users' preferences, as well as, the capacity of the dot-product defined in the learned embedded space to produce the ordering. The proposed model is by nature suitable for implicit feedback and involves the estimation of only very few parameters. Through extensive experiments on several real-world benchmarks on implicit data, we show the interest of learning the preference and the embedding simultaneously when compared to learning those separately. We also demonstrate that our approach is very competitive with the best state-of-the-art collaborative filtering techniques proposed for implicit feedback.

1. Introduction

In the recent years, recommender systems (RS) have attracted a lot of interest in both industry and academic research communities, mainly due to new challenges that the design of a decisive and efficient RS presents. Given a set of customers (or users), the goal of RS is to provide a personalized recommendation of products to users which would likely to be of their interest. Typical examples of applications include the recommendation of movies (Netflix, Amazon Prime Video), music (Pandora), videos (YouTube), news content (Outbrain) or advertisements (Google). The development of an efficient RS is critical for both the company and the consumer perspective. On the one hand, users usually face a huge number of options: for instance, Amazon proposes over 20,000 movies in its selection, and it is, therefore, essential to help them to take the best possible decision by narrowing down the choices they have to make. On the other hand, major companies report significant increase of their traffic and sales coming from personalized recommendations: Amazon declares that recommendations generate 35% of its sales, two-thirds of the movies watched on Netflix are recommended, and 28% of ChoiceStream users said that they would buy more music, provided the fact that they meet their tastes and interests [1].

*Corresponding Author: Massih-Reza Amini
Email address: Massih-Reza.Amini@univ-grenoble-alpes.fr (Massih-Reza Amini)

[1] Talk of Xavier Amatriain - Recommender Systems - Machine Learning Summer School 2014 @ CMU.

Two main approaches have been proposed to tackle this problem [1]. The first one, referred to as Content-Based recommendation technique [2] makes use of existing contextual information about the users (e.g. demographic information) or items (e.g. textual description) for recommendation. The second approach referred to as collaborative filtering (CF) and undoubtedly the most popular one, relies on the past interactions and recommends items to users based on the feedback provided by other similar users. Feedback can be *explicit*, in the form of ratings; or *implicit*, which includes clicks, browsing over an item or listening to a song. Such implicit feedback is readily available in abundance but is more challenging to take into account as it does not clearly depict the preference of a user for an item. Explicit feedback, on the other hand, is very hard to get in abundance. The adaptation of CF systems designed for another type of feedback has been shown to be sub-optimal as the basic hypothesis of these systems inherently depends on the nature of the feedback [3]. Further, learning a suitable representation of users and items have been shown to be the bottleneck of these systems [4], mostly in the cases where contextual information over users and items, which allows having a richer representation, is unavailable.

In this paper, we are interested in the learning of user preferences mostly provided in the form of implicit feedback in RS. Our aim is twofold and concerns:

1. the development of a theoretical framework for learning user preference in recommender systems that justifies the learnability of pairwise ranking models proposed until now for this task, and its analysis in the worst case where all users provide a minimum of positive/negative feedback;

2. the design of a new neural-network model based on this framework that jointly learns the preference of users over pairs of items and their representations in an embedded space without requiring any contextual information.

We extensively validate our proposed approach over five publicly available benchmarks with implicit feedback by comparing it to state of the art models.

The remainder of this paper is organized as follows. In Section 2, we provide an overview of existing related methods. Then, Section 3 defines the notations and the proposed framework and analyze its theoretical properties. Section 4 is devoted to numerical experiments on five real-world benchmark datasets including binarized versions of MovieLens and Netflix, and two real data sets on online advertising. We compare different versions of our model with state-of-the-art methods showing the appropriateness of our contribution. Finally, we summarize the study and give possible future research perspectives in Section 5.

2. State-of-the-art

This section provides an overview of the state-of-the-art approaches that are the most similar to ours.

2.1. Neural Language Models

Neural language models have proven themselves to be successful in many natural language processing tasks including speech recognition, information retrieval, and sentiment analysis. These models are based on a distributional hypothesis stating that words, occurring in the same context with the same frequency, are similar. To capture such similarities, these approaches propose to embed the word distribution into a low-dimensional continuous space using Neural Networks, leading to the development of several powerful and highly scalable language models such as the word2vec Skip-Gram (SG) model [5, 6].

The recent work of [7] has shown new opportunities to extend the word representation learning to characterize more complicated pieces of information. This paper established the equivalence between the SG model with a negative sampling and implicitly factorizing point-wise mutual information (PMI) matrix. Further, they demonstrated that word embedding could be applied to different types of data, provided that it is possible to design an appropriate context matrix for them. This idea has been successfully applied to recommendation systems where different approaches attempted to learn representations of items and users in an embedded space to meet the problem of recommendation more efficiently [8, 9, 10].

In [11], the authors used a bag-of-word vector representation of items and users, from which the latent representations of latter are learned through word-2-vec. [9] proposed a model that relies on the intuitive idea that the pairs of items which are scored in the same way by different users are similar. The approach reduces to finding both the latent representations of users and items, with the traditional Matrix Factorization (MF) approach, and simultaneously learning item embeddings using a co-occurrence shifted positive PMI (SPPMI) matrix defined by items and their context. The latter is used as a regularization term in the traditional objective function of MF. Similarly, in [10] the authors proposed Prod2Vec, which embeds items using a Neural-Network language model applied to a time series of user purchases. This model was further extended in [12] who, by defining appropriate context matrices, proposed a new model called Meta-Prod2Vec. Their approach learns a representation of both items and side information available in the system. The embedding of additional information is further used to regularize the item embedding. Inspired by the concept of a sequence of words; the approach proposed by [8] defined the consumption of items by users as trajectories. Then, the embedding of items is learned using the SG model, and the users' embeddings are further used to predict the next item in the trajectory. In these approaches, the learning of item and user representations are employed to predict with predefined or fixed similarity functions (such as dot-products) in the embedded space.

Although learning user and item embeedings seem to be unavoidable in RS, as the interaction between users and items is generally the only available source of information characterizing them, many studies have pointed out that the second ingredient, making RS to be efficient, is the learning of user's preference.

2.2. Learning-to-Rank based Neural Networks for Recommender systems

Motivated by automatically tuning the parameters involved in the combination of different scoring functions, Learning-to-Rank approaches were initially developed for Information Retrieval (IR) tasks and are grouped into three main categories: pointwise, listwise and pairwise [13].

Pointwise approaches [14, 15] assume that each queried document pair has an ordinal score. The ranking is then stated as a regression problem, in which the rank value of each document is estimated as an absolute quantity. In the case where relevance judgments are given as pairwise preferences (rather than relevance degrees), it is usually not straightforward to apply these algorithms for learning. Moreover, pointwise techniques do not consider the inter-dependency among documents, so that the position of documents in the final ranked list is missing in the regression-like loss functions used for parameter tuning. On the other hand, listwise approaches [16, 17, 18] take the entire ranked list of documents for each query as a training instance. As a direct consequence, these approaches are able to differentiate documents from different queries and consider their position in the output ranked list at the training stage. Listwise techniques aim to optimize a ranking measure directly, so they generally face a complex optimization problem dealing with non-convex, non-differentiable and discontinuous functions. Finally, in pairwise approaches, [19, 20, 21, 22] the ranked list is decomposed into a set of document pairs. The ranking is therefore considered as the classification of pairs of documents, such that a classifier is trained by minimizing the number of misorderings in ranking. In the test phase, the classifier assigns a positive or negative class label to a document pair that indicates which of the documents in the pair should be better ranked than the other one.

Perhaps the first Neural Network model for ranking is RankProp, proposed initially by [23]. RankProp is a pointwise approach that alternates between two phases of learning the desired real outputs by minimizing a Mean Squared Error (MSE) objective, and a modification of the desired values themselves to reflect the current ranking given by the net. Later on [24] proposed RankNet, a pairwise approach, that learns a preference function by minimizing a cross entropy cost over the pairs of relevant and irrelevant examples. SortNet proposed by [25, 26] also learns a preference function by minimizing a ranking loss over the pairs of examples that are selected iteratively with the overall aim of maximizing the quality of the ranking. The three approaches above consider the problem of Learning-to-Rank for IR and without learning an embedding.

The architecture of the propsed approach bears similarity with the wide and deep learning model which has also two components [27]. As in our case, the wide part is a linear model and learns dense embeddings of the users and items and is said to take care of memorization. The prediction function is the dot product

of weights learned and user and item embeddings. The deep part is a fully connected deep neural network and takes care of generalization. As in our case, the training takes place jointly.

The key difference, though, lies in the prediction function. In wide and deep learning, the goal is to predict a score (as in pointwise ranking) which is very different from the pairwise learning to rank function that is in our model. In recent years, most focus has been put on the development of pairwise approaches as they have been shown to be more efficient than pointwise approaches for recommender systems.

Perhaps, the closest work to ours is [28], in which authors learn efficient user and item embedding. Then, the respective embeddings are concatenated and goes through a dense layer consisting of fully connected layers. But, no loss focusing on quality of representations is employed. In [29], authors develop deep learning techniques for textual data, particularly reviews and use ratings based feedback to optimize. It is not applicable to implicit feedback, such as clicks. [30] use heterogeneous sources such as review text, item image and rating in order to learn good representation of the item for doing top-n recommendation. While being very effective model, it does require heterogeneous sources for it to be effective. Finally, [31] use wide and deep learning, to combine the benefits of memorization and generalization for recommender systems.

In this study we tackle the consistency of the empirical risk minimization principle used to learn pairwise ranking models for RS and propose a Neural Network model, with a composite objective loss, which jointly influences the learning of the embeddings and the scoring function.

3. Framework and Model

We denote by $\mathcal{U} \subseteq \mathbb{N}$ (resp. $\mathcal{I} \subseteq \mathbb{N}$) the set of indexes over users (resp. the set of indexes over items). Further, for each user $u \in \mathcal{U}$, we consider two subsets of items $\mathcal{I}_u^- \subset \mathcal{I}$ and $\mathcal{I}_u^+ \subset \mathcal{I}$ such that;

i) $\mathcal{I}_u^- \neq \varnothing$ and $\mathcal{I}_u^+ \neq \varnothing$,

ii) for any pair of items $(i, i') \in \mathcal{I}_u^+ \times \mathcal{I}_u^-$; u has a preference, symbolized by $\succ_u$. Hence $i \succ_u i'$ implies that, user u prefers item i over item i'.

From this preference relation, a desired output $y_{i,u,i'} \in \{-1, +1\}$ is defined over each triplet $(i, u, i') \in \mathcal{I}_u^+ \times \mathcal{U} \times \mathcal{I}_u^-$ as:

$$y_{i,u,i'} = \begin{cases} 1 & \text{if } i \succ_u i', \\ -1 & \text{otherwise.} \end{cases} \tag{1}$$

3.1. Learning objective

Following [32], we consider the learning task that consists in finding a scoring function f from a class of functions $\mathcal{F} = \{f \mid f : \mathcal{I} \times \mathcal{U} \times \mathcal{I} \to \mathbb{R}\}$ that minimizes the ranking loss:

$$\mathcal{L}(f) = \mathbb{E}\left[\frac{1}{|\mathcal{I}_u^+||\mathcal{I}_u^-|} \sum_{i \in \mathcal{I}_u^+} \sum_{i' \in \mathcal{I}_u^-} \mathbb{1}_{y_{i,u,i'} f(i,u,i') < 0}\right], \tag{2}$$

where $|.|$ measures the cardinality of sets and $\mathbb{1}_\pi$ is the indicator function which is equal to 1, if the predicate π is true, and 0 otherwise. Many approaches tackled this problem by proposing to learn a mapping function $\Phi : \mathcal{U} \times \mathcal{I} \to \mathcal{X} \subseteq \mathbb{R}^k$ that projects a pair of user and item indices into a feature space of dimension k, and a function $g : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that each function $f \in \mathcal{F}$ can be decomposed as:

$$\forall u \in \mathcal{U}, (i, i') \in \mathcal{I}_u^+ \times \mathcal{I}_u^-, \ f(i, u, i') = g(\Phi(u, i)) - g(\Phi(u, i')).$$

The previous loss (2) is a pairwise ranking loss, and it is related to the Area under the ROC curve [33]. The learning objective is, hence, to find a function f from the class of functions $\mathcal{F}$ with a small expected risk, by minimizing the empirical error over a training set

$$S = \{(\mathbf{z}_{i,u,i'} \doteq (i, u, i'), y_{i,u,i'}) \mid u \in \mathcal{U}, (i, i') \in \mathcal{I}_u^+ \times \mathcal{I}_u^-\},$$

constituted over N users, $\mathcal{U} = \{1, \ldots, N\}$, and their respective preferences over M items, $\mathcal{I} = \{1, \ldots, M\}$ and is given by:

$$\begin{aligned}\hat{\mathcal{L}}(f, S) &= \frac{1}{N} \sum_{u \in \mathcal{U}} \frac{1}{|\mathcal{I}_u^+||\mathcal{I}_u^-|} \sum_{i \in \mathcal{I}_u^+} \sum_{i' \in \mathcal{I}_u^-} \mathbb{1}_{y_{i,u,i'}(f(i,u,i'))<0} \\ &= \frac{1}{N} \sum_{u \in \mathcal{U}} \frac{1}{|\mathcal{I}_u^+||\mathcal{I}_u^-|} \sum_{i \in \mathcal{I}_u^+} \sum_{i' \in \mathcal{I}_u^-} \mathbb{1}_{y_{i,u,i'}(g(\Phi(u,i))-g(\Phi(u,i')))<0}. \end{aligned} \tag{3}$$

The pairwise ranking loss (3) is equivalent to a classification loss over the pairs of examples. By this, the aim of the prediction function is not to predict the score (+1, relevant, or, -1; irrelevant), but rather to preserve the relative order of preferences between two ratings given by the same user. Although different studies have shown the efficiency of jointly learning an adapted users and items representations, as well as the scoring function g, [32]. However this minimization problem involves dependent random variables as for each user u and item i; all comparisons $g(\Phi(u, i)) - g(\Phi(u, i')); i' \in \mathcal{I}_u^-$ involved in the empirical error (3) share the same observation $\Phi(u, i)$.

To the best of our knowledge, there is no study which considered the consistency of the empirical risk minimization principle that is generally used for this task. To tackle this problem, we build on [34] and derive generalization error bounds for bounding the error (2) with respect to (3). The idea is based on graph coloring, introduced by [35], and which consists in dividing a graph $\Omega = (\mathcal{V}, \mathcal{E})$ that links dependent variables represented by its nodes $\mathcal{V}$ into J sets of *independent* variables, called the exact proper fractional cover of Ω and defined as:

Definition 1 (Exact proper fractional cover of Ω, [35])**.** *Let $\Omega = (\mathcal{V}, \mathcal{E})$ be a graph. $\mathcal{C} = \{(\mathcal{M}_j, \omega_j)\}_{j \in \{1,\ldots,J\}}$, for some positive integer J, with $\mathcal{M}_j \subseteq \mathcal{V}$ and $\omega_j \in [0, 1]$ is an exact proper fractional cover of Ω, if: i) it is* proper*: $\forall j$, $\mathcal{M}_j$ is an* independent set*, i.e., there is no connections between vertices in $\mathcal{M}_j$; ii) it is an* exact fractional cover *of Ω: $\forall v \in \mathcal{V}$, $\sum_{j:v \in \mathcal{M}_j} \omega_j = 1$.*

The weight $W(\mathcal{C})$ of $\mathcal{C}$ is given by: $W(\mathcal{C}) \doteq \sum_{j=1}^{J} \omega_j$ and the minimum weight $\chi^*(\Omega) = \min_{\mathcal{C} \in \mathcal{K}(\Omega)} W(\mathcal{C})$ over the set $\mathcal{K}(\Omega)$ of all exact proper fractional covers of Ω is the *fractional chromatic number* of Ω.

Figure 1 depicts an exact proper fractional cover corresponding to the problem we consider for a toy problem with $M = 5$ items and a user u, preferring $|\mathcal{I}_u^+| = 2$ items over $|\mathcal{I}_u^-| = 3$ other ones. In this case, the nodes of the dependency graph correspond to 6 pairs constituted by; pairs of the user and each of the preferred items, with the pairs constituted by the user and each of the no preferred items, involved in the empirical loss (3). Among all the sets containing independent pairs of examples, the one shown in Figure 1,(c) is the exact proper fractional cover of Ω and the fractional chromatic number is, in this case, $\chi^*(\Omega) = |\mathcal{I}_u^-| = 3$.

By mixing the idea of graph coloring with the Laplace transform, Hoeffding like concentration inequalities for the sum of dependent random variables are proposed by [35]. In [36] this result is extended to provide a generalization of the bounded differences inequality of [37] to the case of interdependent random variables. This extension then paved the way for the definition of the *fractional Rademacher complexity* that generalizes the idea of Rademacher complexity and allows one to derive generalization bounds for scenarios where the training data are made of dependent data.

In the worst case scenario where all users provide the lowest interactions over the items, which constitutes the bottleneck of all recommendation systems:

$$\forall u \in S, |\mathcal{I}_u^-| = n_*^- = \min_{u' \in S} |\mathcal{I}_{u'}^-|, \text{ and } \; |\mathcal{I}_u^+| = n_*^+ = \min_{u' \in S} |\mathcal{I}_{u'}^+|,$$

the empirical loss (3) is upper-bounded by:

$$\hat{\mathcal{L}}(f, S) \leq \hat{\mathcal{L}}_*(f, S) = \frac{1}{N} \frac{1}{n_*^- n_*^+} \sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{I}_u^+} \sum_{i' \in \mathcal{I}_u^-} \mathbb{1}_{y_{i,u,i'} f(i,u,i')<0}. \tag{4}$$

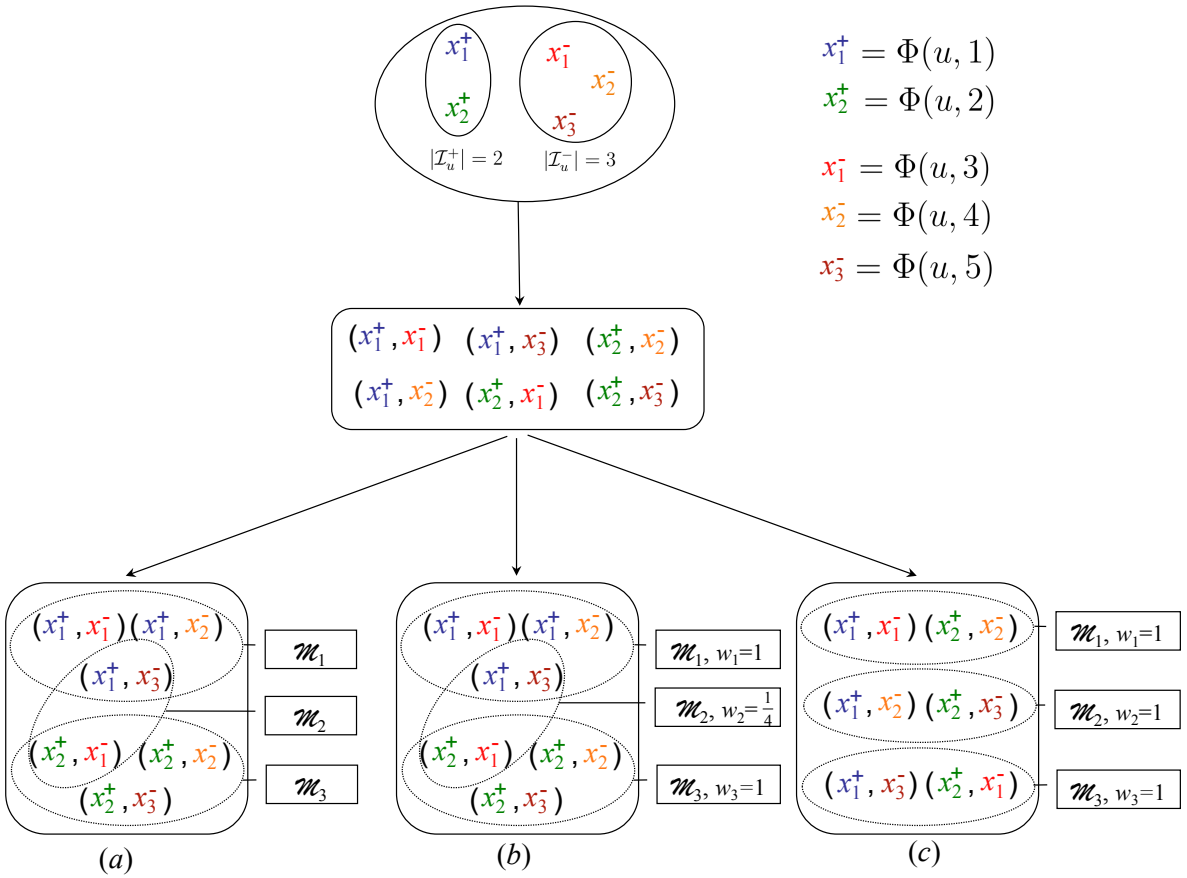


Figure 1: A toy problem with 1 user who prefers $|\mathcal{I}_u^+| = 2$ items over $|\mathcal{I}_u^-| = 3$ other ones (top). The dyadic representation of pairs constituted with the representation of the user and each of the representations of preferred and non-preferred items (middle). A different covering of the dependent set, (a) and (b); as well as the exact proper fractional cover, (c), corresponding to the smallest disjoint sets containing independent pairs.

Following [38, Proposition 4], a generalization error bound can be derived for the second term of the inequality above based on local Rademacher Complexities that implies second-order (i.e. variance) information inducing faster convergence rates.

For sake of presentation and in order to be in line with the learning representations of users and items in an embedded space introduced in Section 3.2, let us consider the kernel-based hypotheses with $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ a *positive semi-definite* (PSD) kernel and $\Phi : \mathcal{U} \times \mathcal{I} \to \mathcal{X}$ its associated feature mapping function. Further we consider linear functions in the feature space with bounded norm:

$$\mathcal{G}_B = \{g_{\boldsymbol{w}} \circ \Phi : (u,i) \in \mathcal{U} \times \mathcal{I} \mapsto \langle \boldsymbol{w}, \Phi(u,i) \rangle \mid ||\boldsymbol{w}|| \leq B\}$$

where $\boldsymbol{w}$ is the weight vector defining the kernel-based hypotheses and $\langle \cdot, \cdot \rangle$ denotes the dot product. We further define the following associated function class:

$$\mathcal{F}_B = \{\mathbf{z}_{i,u,i'} \doteq (i,u,i') \mapsto g_{\boldsymbol{w}}(\Phi(u,i)) - g_{\boldsymbol{w}}(\Phi(u,i')) \mid g_{\boldsymbol{w}} \in \mathcal{G}_B\},$$

and the parameterized family $\mathcal{F}_{B,r}$ which, for $r > 0$, is defined as:

$$\mathcal{F}_{B,r} = \{f : f \in \mathcal{F}_B, \mathbb{V}[f] \doteq \mathbb{V}_{\mathbf{z},y}[\mathbb{1}_{yf(\mathbf{z})}] \leq r\},$$

where $\mathbb{V}[.]$ denotes the variance. The fractional Rademacher complexity introduced in [36] entails our analysis:

$$\mathfrak{R}_S(\mathcal{F}) = \frac{2}{m}\mathbb{E}_\xi \sum_{j=1}^{n_*^-} \mathbb{E}_{\mathcal{M}_j} \sup_{f\in\mathcal{F}} \sum_{\substack{\alpha\in\mathcal{M}_j \\ \mathbf{z}_\alpha\in S}} \xi_\alpha f(\mathbf{z}_\alpha),$$

where $m = N \times n_*^+ \times n_*^-$ is the total number of triplets $\mathbf{z}$ in the training set and $(\xi_i)_{i=1}^m$ is a sequence of independent Rademacher variables verifying $\mathbb{P}(\xi_i = 1) = \mathbb{P}(\xi_i = -1) = \frac{1}{2}$.

Theorem 1. *Let $\mathcal{U}$ be a set of M independent users, such that each user $u \in \mathcal{U}$ prefers n_*^+ items over n_*^- ones in a predefined set of $\mathcal{I}$ items. Let $S = \{(\mathbf{z}_{i,u,i'} \doteq (i,u,i'), y_{i,u,i'}) \mid u \in \mathcal{U}, (i,i') \in \mathcal{I}_u^+ \times \mathcal{I}_u^-\}$ be the associated training set, then for any $1 > \delta > 0$ the following generalization bound holds for all $f \in \mathcal{F}_{B,r}$ with probability at least $1-\delta$:*

$$\mathcal{L}(f) \leq \hat{\mathcal{L}}_*(f,S) + \frac{2B\mathfrak{C}(S)}{Nn_*^+} + \frac{5}{2}\left(\sqrt{\frac{2B\mathfrak{C}(S)}{Nn_*^+}} + \sqrt{\frac{r}{2}}\right)\sqrt{\frac{\log\frac{1}{\delta}}{n_*^+}} + \frac{25}{48}\frac{\log\frac{1}{\delta}}{n_*^+},$$

where $\mathfrak{C}(S) = \sqrt{\frac{1}{n_*^-}\sum_{j=1}^{n_*^-}\mathbb{E}_{\mathcal{M}_j}\left[\sum_{\substack{\alpha\in\mathcal{M}_j \\ \mathbf{z}_\alpha\in S}} d(\mathbf{z}_\alpha,\mathbf{z}_\alpha)\right]}$, $\mathbf{z}_\alpha = (i_\alpha, u_\alpha, i'_\alpha)$ *and also* $d(\mathbf{z}_\alpha,\mathbf{z}_\alpha) = \kappa(\phi,\phi) + \kappa(\phi',\phi') - 2\kappa(\phi,\phi')$ *with* $\phi = \Phi(u_\alpha, i_\alpha)$, $\phi' = \Phi(u_\alpha, i'_\alpha)$.

The proof is given in Appendix.

This result suggests that :

- even though the training set S contains interdependent observations; following [39, theorem 2.1, p. 38], theorem 1 gives insights on the consistency of the empirical risk minimization principle with respect to the minimization of (Eq. 4),
- in the case where the feature space $\mathcal{X} \subseteq \mathbb{R}^k$ is of finite dimension; lower values of k involves lower kernel estimation and hence lower complexity term $\mathfrak{C}(S)$ which implies a tighter generalization bound.

3.2. A Neural Network model to learn user preference

In this section we present a Neural Network, denoted as `NERvE`, to learn the embedding representation jointly, $\Phi(.)$, as well as the scoring function, $f(.)$, defined in the previous section. The input of the network is a triplet (i,u,i') composed by the indexes of an item i, a user u and a second item i'; such that the user u has a preference over the pair of items (i,i') expressed by the desired output $y_{i,u,i'}$, defined with respect to the preference relation $\succ_u$ (Eq. 1). Each index in the triplet is then transformed to a corresponding binary indicator vector $\mathbf{i}, \mathbf{u}$, and $\mathbf{i}'$ having all its characteristics equal to 0 except the one that indicates the position of the user or the items in its respective set, which is equal to 1. Hence, the following one-hot vector corresponds to the binary vector representation of user $u \in \mathcal{U}$:

$$\mathbf{u}^\top = \underset{\downarrow}{(0,} \ \underset{\downarrow}{\ldots,} \ \underset{u-1}{\underset{\downarrow}{0,}} \ \underset{u}{\underset{\downarrow}{1,}} \ \underset{u+1}{\underset{\downarrow}{0,}} \ \underset{\downarrow}{\ldots,} \ \underset{N}{\underset{\downarrow}{0)}}.$$

The network entails then three successive layers, namely *Embedding*, *Mapping* and *Dense* hidden layers depicted in Figure 2.

- The *Embedding* layer transforms the sparse binary representations of the user and each of the items to denser real-valued vectors. We denote by $\mathbf{U}_u$ and $\mathbf{V}_i$ the transformed vectors of user u and item i; and $\mathbb{U} = (\mathbf{U}_u)_{u\in\mathcal{U}}$ and $\mathbb{V} = (\mathbf{V}_i)_{i\in\mathcal{I}}$ the corresponding matrices. Note that as the binary indicator vectors of users and items contain one single non-null characteristic, each entry of the corresponding dense vector in the *Embedding* layer is connected by only one weight to that characteristic.

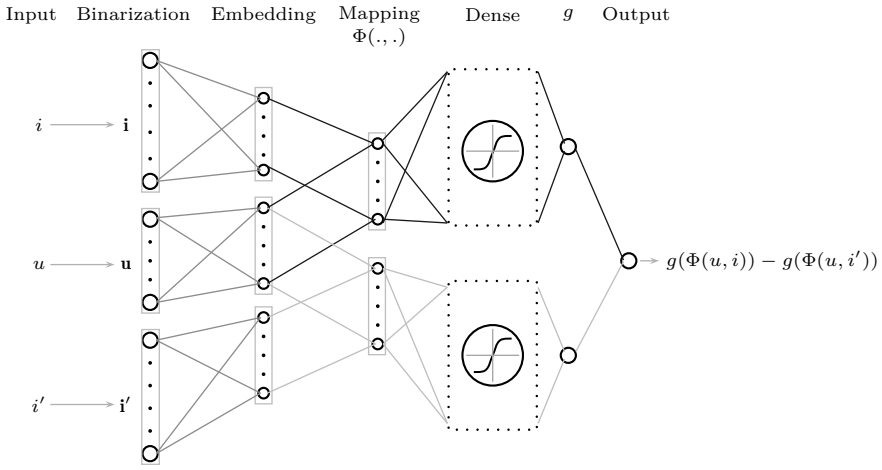


Figure 2: The architecture of `NERvE` trained to reflect the preference of a user u over a pair of items i and i'.

- The *Mapping* layer is composed of two groups of units each being obtained from the element-wise product between the user representation vector $\mathbf{U}_u$ of a user u and a corresponding item representation vector $\mathbf{V}_i$ of an item i inducing the feature representation of the pair $(u, i); \Phi(u, i)$.
- Each of these units is also fully connected to the units of a *Dense* layer composed of successive hidden layers (see Section 4 for more details related to the number of hidden units and the activation function used in this layer).

The model is trained such that the output of each of the dense layers reflects the relationship between the corresponding item and the user and is mathematically defined by a multivariate real-valued function $g(.)$. Hence, for an input (i, u, i'), the output of each of the dense layers is a real-value score that reflects a preference associated to the corresponding pair (u, i) or (u, i') (i.e. $g(\Phi(u, i))$ or $g(\Phi(u, i'))$). Finally the prediction given by `NERvE` for an input (i, u, i') is:

$$f(i, u, i') = g(\Phi(u, i)) - g(\Phi(u, i')). \tag{5}$$

3.3. Algorithmic implementation

The main difference with other approaches which also proposed to learn jointly the user and items embedding and the scoring function g [28, 29], is that here we consider a composite pairwise ranking loss defined as :

$$\mathcal{L}_{c,p}(f, \mathbb{U}, \mathbb{V}, \mathcal{S}) = \alpha \mathcal{L}_c(f, \mathcal{S}) + (1 - \alpha)\mathcal{L}_p(\mathbb{U}, \mathbb{V}, \mathcal{S}), \tag{6}$$

where $\alpha \in [0, 1]$ is a real-valued parameter and the first term reflects the ability of the non-linear transformation of user and item feature representations, $g(\Phi(.,.))$, to respect the relative ordering of items with respect to users' preferences:

$$\mathcal{L}_c(f, \mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{(\mathbf{z}_{i,u,i'}, y_{i,u,i'}) \in \mathcal{S}} \log(1 + e^{y_{i,u,i'}(g(\Phi(u,i')) - g(\Phi(u,i)))}). \tag{7}$$

The second term focuses on the quality of the compact dense vector representations of items and users that have to be found, as measured by the ability of the dot-product in the resulting embedded vector space to respect the relative ordering of preferred items by users:

$$\mathcal{L}_p(\mathbb{U},\mathbb{V},\mathcal{S}) = \frac{1}{|\mathcal{S}|}\sum_{\mathcal{S}}\Big[\log(1+e^{y_{i,u,i'}\mathbf{U}_u^\top(\mathbf{V}_{i'}-\mathbf{V}_i)}) + \lambda(\|\mathbf{U}_u\|_2^2+\|\mathbf{V}_{i'}\|_2^2+\|\mathbf{V}_i\|_2^2)\Big], \tag{8}$$

where λ is a regularization parameter for the user and items norms.

The purpose of conjugating between the two losses is to see the impact of each on the learning of the final scoring function.

Training phase

The empirical minimization of the ranking losses is carried out by back-propagating [40] the error-gradients from the output to both the deep and embedding parts of the model using mini-batch stochastic optimization (Algorithm 1).

During training, the input layer takes a random set $\tilde{\mathcal{S}}_n$ of size n of interactions by building triplets (i,u,i') based on this set and generating a sparse representation from id's vector corresponding to the picked user and the pair of items. The binary vectors of the examples in $\tilde{\mathcal{S}}_n$ are then propagated throughout the network, and the ranking error (Eq. 6) is back-propagated.

Algorithm 1 `NERvE_`: Learning phase

Input:
T: maximal number of epochs
A set of users $\mathcal{U} = \{1,\ldots,N\}$
A set of items $\mathcal{I} = \{1,\ldots,M\}$
for $ep = 1,\ldots,T$
  Randomly sample a mini-batch $\tilde{\mathcal{S}}_n \subseteq \mathcal{S}$ of size n from the original user-item matrix
  for all $((i,u,i'),y_{i,u,i'}) \in \tilde{\mathcal{S}}_n$
    Propagate (i,u,i') from the input to the output.
  Retro-propagate the pairwise ranking error (Eq. (6)) estimated over $\tilde{\mathcal{S}}_n$.
Output: Users and items latent feature matrices $\mathbb{U},\mathbb{V}$ and the model weights.

Model Testing

As for the prediction phase, shown in Algorithm 2, a ranked list $\mathfrak{N}_{u,k}$ of the $k \ll M$ preferred items for each user in the test set is maintained while retrieving the set $\mathcal{I}$. Given the latent representations of the triplets, and the weights learned; the two first items in $\mathcal{I}$ are placed in $\mathfrak{N}_{u,k}$ in a way which ensures that preferred one, i^*, is in the first position. Then, the algorithm retrieves the next item, $i \in \mathcal{I}$ by comparing it to i^*. This step is simply carried out by comparing the model's output over the concatenated binary indicator vectors of (i^*,u,i) and (i,u,i^*). Hence, if $f(i,u,i^*) > f(i^*,u,i)$, which from Equation (5) is equivalent to $g(\Phi(u,i)) > g(\Phi(u,i^*))$, then i is predicted to be preferred over i^*; $i \succ_u i^*$; and it is put at the first place instead of i^* in $\mathfrak{N}_{u,k}$. Here we assume that the predicted preference relation $\succ_u$ is transitive, which then ensures that the predicted order in the list is respected. Otherwise, if i^* is predicted to be preferred over i, then i is compared to the second preferred item in the list, using the model' prediction as before, and so on. The new item, i, is inserted in $\mathfrak{N}_{u,k}$ in the case if it is found to be preferred over another item in $\mathfrak{N}_{u,k}$.

By repeating the process until the end of $\mathcal{I}$, we obtain a ranked list of the k most preferred items for the user u. Algorithm 2 does not require an ordering of the whole set of items, as also in most cases we are just interested in the relevancy of the top-ranked items for assessing the quality of a model. Further, its complexity is at most $O(k \times M)$ which is convenient in the case where $M \gg 1$ and sufficiently small k ($k = 10$ in our experiments). The merits of a similar algorithm have been discussed by [41] but, as pointed out above, the basic assumption for inserting a new item in the ranked list $\mathfrak{N}_{u,k}$ is that the predicted preference relation induced by the model should be transitive, which may not hold in general.

Algorithm 2 NERvE_: Testing phase

Input:
A user $u \in \mathcal{U}$; A set of items $\mathcal{I} = \{1, \ldots, M\}$;
A set containing the k preferred items in $\mathcal{I}$ by u;
$\mathfrak{N}_{u,k} \leftarrow \varnothing$;
f: the output of Algorithm 1;
Apply f to the first two items of $\mathcal{I}$ and, note the preferred one i^* and place it at the top of $\mathfrak{N}_{u,k}$;
for $i = 3, \ldots, M$
  if $g(\Phi(u, i)) > g(\Phi(u, i^*))$ **then**
    Add i to $\mathfrak{N}_{u,k}$ at rank 1
  else
    $j \leftarrow 1$
    while $j \leq k$ AND $g(\Phi(u, i)) < g(\Phi(u, i_g)))$ // where $i_g = \mathfrak{N}_{u,k}(j)$
      $j \leftarrow j + 1$
    if $j \leq k$ **then**
      Insert i in $\mathfrak{N}_{u,k}$ at rank j
Output: $\mathfrak{N}_{u,k}$;

In our experiments, we also tested a more conventional inference algorithm, which for a given user u, consists in the ordering of items in $\mathcal{I}$ with respect to the output provided by the function g, and we did not find any substantial difference in the performance of NERvE_, as presented in the following section.

4. Experimental Results

We conducted several experiments aimed at evaluating how the simultaneous learning of user and item representations, as well as the preferences of users over items, can be efficiently handled with NERvE_. To this end, we considered five real-world benchmarks commonly used for collaborative filtering. We validated our approach concerning different hyper-parameters that impact the accuracy of the model and compare it with competitive state-of-the-art approaches.

We run all experiments on a cluster of five 32 core Intel Xeon @ 2.6Ghz CPU (with 20MB cache per core) systems with 256 Giga RAM running Debian GNU/Linux 8.6 (wheezy) operating system. All subsequently discussed components were implemented in Python3 using the TensorFlow library with version 1.4.0.[2]

4.1. Datasets

The datasets that were used in our experiments are :

- MovieLens[3] 100K (ML-100K), MovieLens 1M (ML-1M) [42] and Netflix[4] which consist of user-movie ratings, on a scale of one to five, collected from a movie recommendation service and the Netflix company. The latter was released to support the Netflix Prize competition[5]. For all three datasets, we only keep users who have rated at least five movies and remove users who gave the same rating for all movies. In addition, for Netflix, we take a subset of the original data and randomly sample 20% of the users and 20% of the items. In the following experiments, as we only compare with approaches developed for the ranking purposes and our model is designed to handle implicit feedback, these three data sets are made binary such that a rating higher or equal to 4 is set to 1 and 0 otherwise.

[2] https://www.tensorflow.org/.
[3] https://movielens.org/
[4] http://academictorrents.com/details/9b13183dc4d60676b773c9e2cd6de5e5542cee9a
[5] B. James and L. Stan, The Netflix Prize (2007).

- The KASANDR[6] dataset contains the interactions and clicks done by the users of Kelkoo, an online advertising platform from Germany. It gathers 17,764,280 interactions from 521,685 users on 2,299,713 offers belonging to 272 categories and spanning across 801 merchants [43]. For KASANDR, we remove users who gave the same rating for all offers, as well as all those who never clicked or always clicked on every offer showed to them.

- The PANDOR[7] collection is another publicly available dataset for online recommendation [44] provided by Purch (`http://www.purch.com/`). The dataset records 2,073,379 clicks generated by 177,366 users of one of the Purch's high-tech website over 9,077 ads they have been shown during one month.

Basic statistics on these collections after pre-processing, as discussed above, are presented in Table 1.q

Table 1: Statistics of various collections used in our experiments after preprocessing.

Dataset	# of users	# of items	# of interactions	Sparsity
ML-100K	943	1,682	100,000	93.685%
ML-1M	6,040	3,706	1,000,209	95.530%
NETFLIX	90,137	3,560	4,188,098	98.700%
KASANDR	25,848	1,513,038	9,489,273	99.976%
PANDOR	5,894,431	14,716	48,754,927	99.873%

4.2. Experimental set-up

Compared baselines. In order to validate the framework defined in the previous section, we propose to compare the following approaches.

- `Co-Factor` [9], developed for implicit feedback, constraints the objective of matrix factorization to use jointly item representations with a factorized shifted positive pointwise mutual information matrix of item co-occurrence counts. The model was found to outperform WMF [45] also proposed for implicit feedback.

- `LightFM` [46] was first proposed to deal with the problem of cold-start using meta information. As with our approach, it relies on learning the embedding of users and items with the Skip-gram model but optimizes a pointwise based likelihood ranking loss depending on the dot product of user and item representations, adjusted by user and item feature biases.

- Neural Collaborative Filtering (NCF)[8][28], that jointly learns the user and items embeddings and the scoring function using Neural Networks, by minimizing a least squared error.

- Wide & Deep (W&D)[9][27]; that jointly learns a linear model component with feature transformation and a neural network component with embeddings.

- `BPR-MF` [32] provides an optimization criterion based on implicit feedback; which is the maximum posterior estimator derived from a Bayesian analysis of the pairwise ranking problem and proposes an algorithm based on Stochastic Gradient Descent to optimize it. The model can further be extended to the explicit feedback case and is close to $\texttt{NERvE}_p$. That main difference between the two is that in the latter there is a dense layer with rectified linear units in the dense layers; between the mapping layer and the estimation of the scoring function g.

[6] `https://archive.ics.uci.edu/ml/datasets/KASANDR`
[7] `https://archive.ics.uci.edu/ml/datasets/PANDOR`
[8] `https://github.com/hexiangnan/neural_collaborative_filtering`
[9] `https://github.com/tensorflow/models/tree/master/official/r1/wide_deep`

- NERvE$_p$[10] focuses on the quality of the latent representation of users and items by learning the preference and the representation through the ranking loss $\mathcal{L}_p$ (Eq. (8)).
- NERvE$_c$ focuses on the accuracy of the score obtained at the output of the framework and therefore learns the preference and the representation through the ranking loss $\mathcal{L}_c$ (Eq. (7)).
- NERvE$_{c,p}$ uses a linear combination of $\mathcal{L}_p$ and $\mathcal{L}_c$ as the objective function, with $\alpha = \frac{1}{2}$.

Evaluation protocol. For each dataset, we sort the interactions according to time and take 80% for training the model and the remaining 20% for testing it. Besides, we remove all users and offers which do not occur during the training phase. We study the real-world scenario, setting of predicting the right order over the set of all items for each user.

All comparisons are made based on common ranking metrics, namely the Mean Average Precision (MAP) and the mean Normalized Discounted Cumulative Gain (NDCG). First, let us recall that the Average Precision (AP@ℓ) is defined over the precision, Pr (fraction of recommended items clicked by the user) of clicked items, at rank ℓ.

$$\text{AP@}\ell = \frac{1}{\ell}\sum_{j=1}^{\ell} r_j Pr(j),$$

and the Normalized Discounted Cumulative Gain (nDCG) is defined as :

$$\text{nDCG@}\ell = \frac{\text{DCG@}\ell}{\text{IDCG@}\ell},$$

where the relevance judgment at rank j, r_j, is binary (i.e. equal to 1 when the j^{th} top ranked item is clicked or preferred, and 0 otherwise), $\text{DCG@}\ell = r_1 + \sum_{j=2}^{\ell} \frac{r_j}{\log_2 j}$ is the discounted cumulative gain at rank ℓ, and IDCG@ℓ is the ideal discounted cumulative gain till position ℓ. Then, the means of these AP's and nDCG's across all users are the MAP and the NDCG. In the following results, we report both measures at ranks $\ell = 1$ and $\ell = 10$.

Hyper-parameters tuning. For all datasets, hyper-parameters tuning is done on a separate validation set among the following sets.

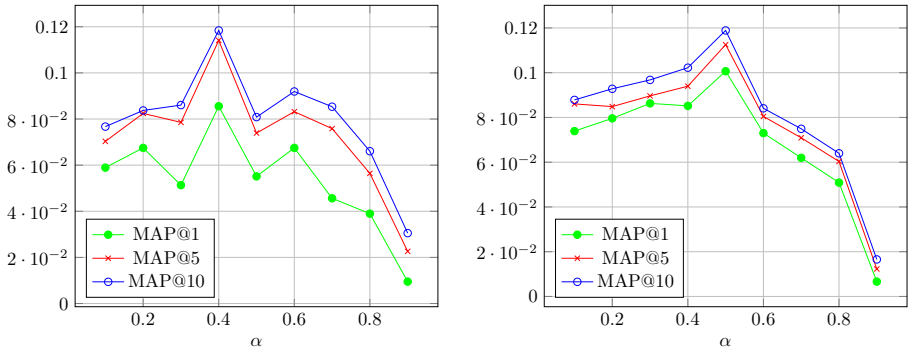
- The size of the embedding is chosen among $k \in \{1, \ldots, 20\}$.
- We use ℓ_2 regularization on the embeddings and choose the hyperparameter λ from the set : $\lambda \in \{10^{-4}, 10^{-3}, 5.10^{-3}, 10^{-2}, 5.10^{-2}\}$.
- We run NERvE with 1 hidden layer with relu activation functions, where the number of hidden units is chosen in $\{16, 32, 64\}$.
- In order to train NERvE, we use ADAM [47] and found the learning rate $\eta = 1e-3$ to be more efficient for all our settings. For other parameters involved in Adam, i.e., the exponential decay rates for the moment estimates, we keep the default values ($\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$).
- Finally, we fix the number of epochs to be $T = 10,000$ in advance and the size of mini-batches to $n = 512$.

Best hyperparameters values for NERvE$_p$, NERvE$_c$ and NERvE$_{c,p}$ with respect to MAP@ℓ are reported in Table 2. It comes out that best results are generally obtained with small sizes of an item and user embedded vector spaces k which support our theoretical analysis where we found that small k induces smaller generalization bounds. This observation on the dimension of embedding is also in agreement with the conclusion of [46], which uses the same technique for representation learning. We exhibit the impact of the value of the hyper-parameter $\alpha \in [0, 1]$ (Eq. (6)) on ML-1M and PANDOR datasets in the learning of model parameters in Figure 3. As expected the conjunction of both ranking losses (Eq. 7) and (Eq. 8) corresponding to situations where $\alpha \neq 0$ and $\alpha \neq 1$ gives always the best results.

[10] https://github.com/sumitsidana/NERvE

Table 2: Best parameters for NERvE$_p$, NERvE$_c$ and NERvE$_{c,p}$; k denotes the dimension of embeddings, λ the regularization parameter. We also report the # of hidden units per layer.

	ML-100K			ML-1M			Netflix			Kasandr			Pandor		
	NERvE$_c$	NERvE$_p$	NERvE$_{c,p}$	NERvE$_c$	NERvE$_p$	NERvE$_{c,p}$	NERvE$_c$	NERvE$_p$	NERvE$_{c,p}$	NERvE$_c$	NERvE$_p$	NERvE$_{c,p}$	NERvE$_c$	NERvE$_p$	NERvE$_{c,p}$
k	15	5	8	2	11	2	3	13	1	4	16	14	19	15	18
λ	10^{-3}	10^{-3}	10^{-3}	5.10^{-2}	10^{-4}	10^{-3}	10^{-4}	10^{-3}	10^{-3}	10^{-3}	10^{-4}	5.10^{-2}	5.10^{-3}	5.10^{-4}	5.10^{-2}
# units	32	16	16	32	64	32	32	64	64	32	64	64	64	64	64



(a) ML-1M (b) Pandor

Figure 3: MAP@1, MAP@5, MAP@10 as a function of the value of α for ML-1M, and Pandor.

4.3. Results

Hereafter, we compare and summarize the performance of NERvE with the baseline methods on various datasets. Since the prediction on all offers is more realistic setting, and predicting on shown offers introduces the bias of the algorithm used to show offers, we compute the results of predicting over all offers of the catalog for all baselines.

Empirically, we observed that the version of NERvE$_{c,p}$ where both $\mathcal{L}_c$ and $\mathcal{L}_p$ have an equal weight while training gives better results on average, and we decided to only report these results later.

Table **??** reports all results. Also, in each case, we statistically compare the performance of each algorithm, and we use boldface to indicate the highest performance, and the symbol $^{\downarrow}$ indicates that performance is significantly worse than the best result, according to a Wilcoxon rank sum test used at a p-value threshold of 0.01 [48].

When the prediction is made over all offers (see Table **??**), we can make two observations. First, all the algorithms encounter an extreme drop in their performance in terms of MAP. Second, NERvE framework significantly outperforms all other algorithms on all datasets, and this difference is all the more important on Kasandr, where for instance NERvE$_{c,p}$ is in average 15 times more efficient.

Comparisons with pointwise ranking approaches. CoFactor, LightFM and NCF are pointwise ranking models and, in the majority of cases, they perform less than pairwise approaches on all datasets regarding both performance measures MAP and NDCG. These results are in line with other studies that compared these two approaches for RS and previously presented in [32, 49]. The common point between these pointwise models is that they all learn users and items' embeddings (using neural networks or not). On Kasandr and Pandor datasets which are larger than the other collections; NN based models (i.e. NCF and W&D) are more efficient than CoFactor and LightFM. These results suggest that with sufficient data, NN are able to learn user and item representations that are more robust regarding the related scoring function in the embedded space for implicit feedback.

Comparisons with BPR-MF. BPR-MF is a pairwise non-neural network approach and fails to capture the non-linearity of user-item interactions. Since NERvE models user-item representations and minimize pairwise ranking loss simultaneously using a dense, fully connected network, it is able to learn non-linear relationships between users and item interactions and hence able to outperform BPR-MF considerably.

Comparison between NERvE versions. One can note that while optimizing ranking losses by Eq. (6) or Eq. (7) or Eq. (8), we simultaneously learn representation and preference function; the main difference is the amount of emphasis we put in learning one or another.

Results presented in table **??** show that, on larger datasets, KASANDR and PANDOR where the number of interactions are higher than in other collections, optimizing the linear combination of the pairwise-ranking loss and the embedding loss ($\texttt{NERvE}_{c,p}$) increases the quality of overall recommendations instead of optimizing standalone losses to learn embeddings and the pairwise preference function.

5. Conclusion

We presented and analyzed a learning to rank framework for recommender systems which consists of learning user preferences over items. We showed that the minimization of pairwise ranking loss over user preferences involves dependent random variables and provided a theoretical analysis by proving the consistency of the empirical risk minimization in the worst case where all users choose a minimal number of positive and negative items. From this analysis, we then proposed NERvE, a new neural-network based model for learning the user preference, where both the user's and item's representations and the function modeling the user's preference over pairs of items are learned simultaneously. The learning phase is guided using a ranking objective that can capture the ranking ability of the prediction function as well as the expressiveness of the learned embedded space, where the preference of users over items is respected by the dot product function defined over that space. The training of NERvE is carried out using the back-propagation algorithm in mini-batches defined over a user-item matrix containing implicit information in the form of subsets of preferred and non-preferred items. The learning capability of the model over both prediction and representation problems show their interconnection and also that the proposed double ranking objective allows to conjugate them well. We assessed and validated the proposed approach through extensive experiments, using five popular collections proposed for the task of recommendation. Furthermore, we propose to study two different settings for the prediction phase and demonstrate that the performance of each approach is strongly impacted by the set of items considered for making the prediction. We believe that our model is a fresh departure from the models which learn pairwise ranking function without the knowledge of embeddings or which learn embeddings without learning any pairwise ranking function.

For future work, we would like to extend NERvE to take into account additional contextual information regarding users and/or items. More specifically, we are interested in the integration of data of different natures, such as text or demographic information as it exists in the PANDOR dataset. We believe that this information can be taken into account without much effort and by doing so, it is possible to improve the performance of our approach and tackle the problem of providing recommendations for new users/items at the same time, also known as the cold-start problem. The second important extension will be the development of an online version of the proposed algorithm to make the approach suitable for real-time applications and online advertising. Finally, we have shown that choosing a suitable α, which controls the trade-off between ranking and embedding loss, greatly impact the performance of the proposed framework, and we believe that an exciting extension will be to learn this hyper-parameter automatically and to make it adaptive during the training phase.

Acknowledgements

This work was partly done under the Calypso project supported by the FEDER program from the Région Auvergne-Rhône-Alpes. The work of Yury Maximov at LANL was carried out under the auspices of the National Nuclear Security Administration of the US Department of Energy under Contract No. DE-AC52-06NA25396 and CNLS/LANL support.

Appendix

Theorem 1. *Let $\mathcal{U}$ be a set of M independent users, such that each user $u \in \mathcal{U}$ prefers n_*^+ items over n_*^- ones in a predefined set of $\mathcal{I}$ items. Let $S = \{(\mathbf{z}_{i,u,i'} \doteq (i,u,i'), y_{i,u,i'}) \mid u \in \mathcal{U}, (i,i') \in \mathcal{I}_u^+ \times \mathcal{I}_u^-\}$ be the*

associated training set, then for any $1 > \delta > 0$ *the following generalization bound holds for all* $f \in \mathcal{F}_{B,r}$ *with probability at least* $1 - \delta$*:*

$$\begin{aligned}\mathcal{L}(f) \leq \; & \hat{\mathcal{L}}_*(f,S) + \frac{2B\mathfrak{C}(S)}{Nn_*^+} + \\ & \frac{5}{2}\left(\sqrt{\frac{2B\mathfrak{C}(S)}{Nn_*^+}} + \sqrt{\frac{r}{2}}\right)\sqrt{\frac{\log\frac{1}{\delta}}{n_*^+}} + \frac{25}{48}\frac{\log\frac{1}{\delta}}{n_*^+},\end{aligned}$$

where $\mathfrak{C}(S) = \sqrt{\frac{1}{n_*^-}\sum_{j=1}^{n_*^-}\mathbb{E}_{\mathcal{M}_j}\left[\sum_{\substack{\alpha\in\mathcal{M}_j \\ \mathbf{z}_\alpha\in S}} d(\mathbf{z}_\alpha,\mathbf{z}_\alpha)\right]}$, $\mathbf{z}_\alpha = (i_\alpha, u_\alpha, i'_\alpha)$ *and*

$$\begin{aligned}d(\mathbf{z}_\alpha,\mathbf{z}_\alpha) = \; & \kappa(\Phi(u_\alpha,i_\alpha),\Phi(u_\alpha,i_\alpha)) \\ & + \kappa(\Phi(u_\alpha,i'_\alpha),\Phi(u_\alpha,i'_\alpha)) - 2\kappa(\Phi(u_\alpha,i_\alpha),\Phi(u_\alpha,i'_\alpha)).\end{aligned}$$

Proof. Let $n_\star^+$ and respectively $n_\star^-$ be the minimum number of preferred and non-preferred items for any user $u \in \mathcal{U}$, then we have :

$$\hat{\mathcal{L}}(f,S) \leq \underbrace{\frac{1}{N}\frac{1}{n_*^-n_*^+}\sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{I}_u^+}\sum_{i'\in\mathcal{I}_u^-}\mathbb{1}_{y_{i,u,i'}f(i,u,i')<0}}_{=\hat{\mathcal{L}}_*(f,S)}. \tag{9}$$

As the set of users $\mathcal{U}$ is supposed to be independent, the exact fractional cover of the dependency graph corresponding to the training set S will be the union of the exact fractional cover associated to each user such that cover sets which do not contain any items in common are joined together.

Following [38, Proposition 4], for any $1 > \delta > 0$ we have with probability at least $1 - \delta$:

$$\begin{aligned}&\mathbb{E}_S[\hat{\mathcal{L}}_*(f,S)] - \hat{\mathcal{L}}_*(f,S) \\ &\qquad \leq \inf_{\beta>0}\left((1+\beta)\mathfrak{R}_S(\mathcal{F}_{B,r}) + \frac{5}{4}\sqrt{\frac{2r\log\frac{1}{\delta}}{n_*^+}} + \frac{25}{16}\left(\frac{1}{3}+\frac{1}{\beta}\right)\frac{\log\frac{1}{\delta}}{n_*^+}\right).\end{aligned}$$

The infimum is reached for $\beta^* = \sqrt{\frac{25}{16}\frac{\log\frac{1}{\delta}}{n_*^+\times\mathfrak{R}_S(\mathcal{F}_{B,r})}}$ which by plugging it back into the upper-bound, and from equation (4), gives:

$$\mathcal{L}(f) \leq \hat{\mathcal{L}}_*(f,S) + \mathfrak{R}_S(\mathcal{F}_{B,r}) + \frac{5}{2}\left(\sqrt{\mathfrak{R}_S(\mathcal{F}_{B,r})} + \sqrt{\frac{r}{2}}\right)\sqrt{\frac{\log\frac{1}{\delta}}{n_*^+}} + \frac{25}{48}\frac{\log\frac{1}{\delta}}{n_*^+}. \tag{10}$$

Now, for all $j \in \{1,\ldots,J\}$ and $\alpha \in \mathcal{M}_j$, let (u_α, i_α) and (u_α, i'_α) be the first and the second pair constructed from $\mathbf{z}_\alpha$, then from the bilinearity of dot product and the Cauchy-Schwartz inequality, $\mathfrak{R}_S(\mathcal{F}_{B,r})$

is upper-bounded by:

$$\frac{2}{m}\mathbb{E}_\xi \sum_{j=1}^{n_*^-} \mathbb{E}_{\mathcal{M}_j} \sup_{f\in\mathcal{F}_{B,r}} \left\langle \boldsymbol{w}, \sum_{\substack{\alpha\in\mathcal{M}_j \\ \mathbf{z}_\alpha\in S}} \xi_\alpha \left(\Phi(u_\alpha, i_\alpha) - \Phi(u_\alpha, i'_\alpha)\right) \right\rangle$$

$$\leq \frac{2B}{m} \sum_{j=1}^{n_*^-} \mathbb{E}_{\mathcal{M}_j} \mathbb{E}_\xi \left\| \sum_{\substack{\alpha\in\mathcal{M}_j \\ \mathbf{z}_\alpha\in S}} \xi_\alpha (\Phi(u_\alpha, i_\alpha) - \Phi(u_\alpha, i'_\alpha)) \right\|$$

$$\leq \frac{2B}{m} \sum_{j=1}^{n_*^-} \left(\mathbb{E}_{\mathcal{M}_j \xi} \left[\sum_{\substack{\alpha,\alpha'\in\mathcal{M}_j \\ \mathbf{z}_\alpha, \mathbf{z}_{\alpha'}\in S}} \xi_\alpha \xi_{\alpha'} d(\mathbf{z}_\alpha, \mathbf{z}_{\alpha'}) \right] \right)^{1/2}, \tag{11}$$

where the last inequality follows from Jensen's inequality and the concavity of the square root, and

$$d(\mathbf{z}_\alpha, \mathbf{z}_{\alpha'}) = \langle \Phi(u_\alpha, i_\alpha) - \Phi(u_\alpha, i'_\alpha), \Phi(u_\alpha, i_\alpha) - \Phi(u_\alpha, i'_\alpha) \rangle .$$

Further, for all $j \in \{1, \ldots, n_*^-\}, \alpha, \alpha' \in \mathcal{M}_j, \alpha \neq \alpha'$; we have $\mathbb{E}_\xi[\xi_\alpha \xi_{\alpha'}] = 0$, [50, p. 91] so:

$$\mathfrak{R}_S(\mathcal{F}_{B,r}) \leq \frac{2B}{m} \sum_{j=1}^{n_*^-} \left(\mathbb{E}_{\mathcal{M}_j} \left[\sum_{\substack{\alpha\in\mathcal{M}_j \\ \mathbf{z}_\alpha\in S}} d(\mathbf{z}_\alpha, \mathbf{z}_\alpha)) \right] \right)^{1/2}$$

$$= \frac{2Bn_*^-}{m} \sum_{j=1}^{n_*^-} \frac{1}{n_*^-} \left(\mathbb{E}_{\mathcal{M}_j} \left[\sum_{\substack{\alpha\in\mathcal{M}_j \\ \mathbf{z}_\alpha\in S}} d(\mathbf{z}_\alpha, \mathbf{z}_\alpha)) \right] \right)^{1/2}.$$

By using Jensen's inequality and the concavity of the square root once again, we finally get

$$\mathfrak{R}_S(\mathcal{F}_{B,r}) \leq \frac{2B}{Nn_*^+} \sqrt{\sum_{j=1}^{n_*^-} \frac{1}{n_*^-} \mathbb{E}_{\mathcal{M}_j} \left[\sum_{\substack{\alpha\in\mathcal{M}_j \\ \mathbf{z}_\alpha\in S}} d(\mathbf{z}_\alpha, \mathbf{z}_\alpha)) \right]}. \tag{12}$$

The result follows from equations (10) and (12). □

6. References

References

[1] F. Ricci, L. Rokach, B. Shapira, P. B. Kantor, Recommender Systems Handbook, 1st Edition, Springer-Verlag New York, Inc., New York, NY, USA, 2010.

[2] P. Lops, M. de Gemmis, G. Semeraro, Content-based recommender systems: State of the art and trends., in: F. Ricci, L. Rokach, B. Shapira, P. B. Kantor (Eds.), Recommender Systems Handbook, Springer, 2011, pp. 73–105.

[3] R. White, J. M. Jose, I. Ruthven, Comparing explicit and implicit feedback techniques for web retrieval: TREC-10 interactive track report, in: Proceedings of Proceedings of the Tenth Text Retrieval Conference (TREC), 2001, pp. 534–538.

[4] H. Wang, N. Wang, D. Yeung, Collaborative deep learning for recommender systems, in: Proceedings of SIGKDD Conference on Knowledge Discovery and Data Mining, 2015, pp. 1235–1244.

[5] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, in: Proceedings of the International Conference on Learning Representations (ICLR), 2013, pp. 1–11.

[6] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Proceedings of International Conference on Neural Information Processing Systems (NIPS), Curran Associates, Inc., 2013, pp. 3111–3119.
[7] O. Levy, Y. Goldberg, Neural word embedding as implicit matrix factorization, in: Proceedings of the International Conference on Neural Information Processing Systems (NIPS), 2014, pp. 2177–2185.
[8] É. Guàrdia-Sebaoun, V. Guigue, P. Gallinari, Latent trajectory modeling: A light and efficient way to introduce time in recommender systems, in: Proceedings of the ACM Recommender Systems Conference (RecSys), 2015, pp. 281–284.
[9] D. Liang, J. Altosaar, L. Charlin, D. M. Blei, Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence, in: Proceedings of the ACM Recommender Systems Conference (RecSys), 2016, pp. 59–66.
[10] M. Grbovic, V. Radosavljevic, N. Djuric, N. Bhamidipati, J. Savla, V. Bhagwan, D. Sharp, E-commerce in your inbox: Product recommendations at scale, in: Proceedings of SIGKDD Conference on Knowledge Discovery and Data Mining, 2015, pp. 1809–1818.
[11] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T. Chua, Neural collaborative filtering, in: Proceedings of the International Conference on World Wide Web Companion (WWW), 2017, pp. 173–182.
[12] F. Vasile, E. Smirnova, A. Conneau, Meta-prod2vec: Product embeddings using side-information for recommendation, in: Proceedings of the ACM Recommender Systems Conference (RecSys), 2016, pp. 225–232.
[13] T. Liu, Learning to rank for information retrieval, Foundations and Trends in Information Retrieval 3 (3) (2009) 225–331.
[14] K. Crammer, Y. Singer, Pranking with ranking, in: Proceedings of the International Conference on Neural Information Processing Systems (NIPS), 2001, pp. 641–647.
[15] P. Li, C. J. C. Burges, Q. Wu, Mcrank: Learning to rank using multiple classification and gradient boosting, in: Proceedings of the International Conference on Neural Information Processing Systems (NIPS), 2007, pp. 897–904.
[16] Y. Shi, M. Larson, A. Hanjalic, List-wise learning to rank with matrix factorization for collaborative filtering, in: Proceedings of the ACM Recommender Systems Conference (RecSys), 2010, pp. 269–272.
[17] J. Xu, H. Li, Adarank: a boosting algorithm for information retrieval, in: Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval, 2007, pp. 391–398.
[18] J. Xu, T. Liu, M. Lu, H. Li, W. Ma, Directly optimizing evaluation measures in learning to rank, in: Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval, 2008, pp. 107–114.
[19] W. W. Cohen, R. E. Schapire, Y. Singer, Learning to order things, J. Artif. Intell. Res. (JAIR) 10 (1999) 243–270.
[20] Y. Freund, R. D. Iyer, R. E. Schapire, Y. Singer, An efficient boosting algorithm for combining preferences, Journal of Machine Learning Research 4 (2003) 933–969.
[21] T. Joachims, Optimizing search engines using clickthrough data, in: Proceedings of SIGKDD Conference on Knowledge Discovery and Data Mining, 2002, pp. 133–142.
[22] J. Pessiot, T. Truong, N. Usunier, M.-R. Amini, P. Gallinari, Learning to rank for collaborative filtering, in: Proceedings of ICEIS, 2007, pp. 145–151.
[23] R. Caruana, S. Baluja, T. M. Mitchell, Using the future to sort out the present: Rankprop and multitask learning for medical risk evaluation, in: Proceedings of the International Conference on Neural Information Processing Systems (NIPS), 1995, pp. 959–965.
[24] C. J. C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, G. N. Hullender, Learning to rank using gradient descent, in: Proceedings of International Conference on Machine Learning (ICML), 2005, pp. 89–96.
[25] L. Rigutini, T. Papini, M. Maggini, M. Bianchini, A neural network approach for learning object ranking, in: Proceedings of International Conference on Artificial Neural Networks (ICANN), 2008, pp. 899–908.
[26] L. Rigutini, T. Papini, M. Maggini, F. Scarselli, Sortnet: Learning to rank by a neural preference function, IEEE Trans. Neural Networks 22 (9) (2011) 1368–1380.
[27] 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, H. Shah, Wide & deep learning for recommender systems, in: Proceedings of the 1st workshop on deep learning for recommender systems, 2016, pp. 7–10.
[28] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T. Chua, Neural collaborative filtering, in: Proceedings of the 26th International Conference on World Wide Web (WWW), 2017, pp. 173–182.
[29] W. Zhang, Q. Yuan, J. Han, J. Wang, Collaborative multi-level embedding learning from reviews for rating prediction, in: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI), 2016, pp. 2986–2992.
[30] Y. Zhang, Q. Ai, X. Chen, W. B. Croft, Joint representation learning for top-n recommendation with heterogeneous information sources, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM '17, 2017, pp. 1449–1458.
[31] H. 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, H. Shah, Wide & deep learning for recommender systems, in: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, 2016, pp. 7–10.
[32] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, BPR: bayesian personalized ranking from implicit feedback, in: Proceedings of the International Conference on Uncertainty in Artificial Intelligence (UAI), 2009, pp. 452–461.
[33] N. Usunier, M.-R. Amini, P. Gallinari, A data-dependent generalisation error bound for the AUC, in: ICML workshop on ROC Analysis in Machine Learning, Bonn, Germany, 2005, pp. 1–9.
[34] M.-R. Amini, N. Usunier, Learning with Partially Labeled and Interdependent Data, Springer, New York, NY, USA, 2015.
[35] S. Janson, Large Deviations for Sums of Partly Dependent Random Variables, Random Structures and Algorithms 24 (3) (2004) 234–248.
[36] N. Usunier, M.-R. Amini, P. Gallinari, Generalization error bounds for classifiers trained with interdependent data, in: Proceedings of the International Conference on Neural Information Processing Systems (NIPS), 2006, pp. 1369–1376.

[37] C. McDiarmid, On the method of bounded differences, Survey in Combinatorics (1989) 148–188.
[38] L. Ralaivola, M.-R. Amini, Entropy-based concentration inequalities for dependent variables, in: Proceedings of Intenational Conference on Machine Learning (ICML), 2015, pp. 2436–2444.
[39] V. Vapnik, The nature of statistical learning theory, Springer Science & Business Media, 2000.
[40] L. Bottou, Stochastic gradient descent tricks, in: Neural Networks: Tricks of the Trade - Second Edition, Curran Associates, Inc., 2012, pp. 421–436.
[41] N. Ailon, M. Mohri, An efficient reduction of ranking to classification, in: Proceedings of Computational Learning Theory (COLT), (2008), pp. 87–98.
[42] F. M. Harper, J. A. Konstan, The movielens datasets: History and context, ACM Trans. Interact. Intell. Syst. 5 (4) (2015) 19:1–19:19.
[43] S. Sidana, C. Laclau, M.-R. Amini, G. Vandelle, A. Bois-Crettez, KASANDR: a large-scale dataset with implicit feedback for recommendation, in: Proceedings the International ACM SIGIR Conference on Research and Development in Information Retrieval, 2017, pp. 1–8.
[44] S. Sidana, C. Laclau, M. R. Amini, Learning to recommend diverse items over implicit feedback on PANDOR, in: Proceedings of the 12^{th} ACM Conference on Recommender Systems (RecSys), 2018, pp. 427–431.
[45] Y. Hu, Y. Koren, C. Volinsky, Collaborative filtering for implicit feedback datasets, in: Proceedings of the International Conference on Data Mining (ICDM), 2008, pp. 263–272.
[46] M. Kula, Metadata embeddings for user and item cold-start recommendations, in: Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems co-located with RecSys., 2015, pp. 14–21.
[47] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Proceedings of the 3rd International Conference on Learning Representations (ICLR), 2014, pp. 1–11.
[48] E. Lehmann, H. D'Abrera, Nonparametrics: statistical methods based on ranks, Springer, 2006.
[49] M. F. Dacrema, P. Cremonesi, D. Jannach, Are we really making much progress? A worrying analysis of recent neural recommendation approaches, in: Proceedings of the 13th ACM Conference on Recommender Systems, RecSys 2019, Copenhagen, Denmark, September 16-20, 2019, 2019, pp. 101–109.
[50] J. Shawe-Taylor, N. Cristianini, Kernel Methods for Pattern Analysis, Cambridge University Press, New York, NY, USA, 2004.