DeepLTRS: A deep latent recommender system based on user ratings and reviews

Dingge Liang, Marco Corneli, Charles Bouveyron, Pierre Latouche

DeepLTRS: A deep latent recommender system based on user ratings and reviews

Dingge Liang[a,*], Marco Corneli[a,b], Charles Bouveyron[a], Pierre Latouche[c]

[a] *Université Côte d'Azur, INRIA, CNRS, Laboratoire J.A.Dieudonné, Maasai team, Nice, France*
[b] *Center of Modelling, Simulation and Interactions (MSI), Nice, France*
[c] *Université de Paris, CNRS, Laboratoire MAP5, UMR 8145, Paris, France*

Abstract

We introduce a deep latent recommender system named deepLTRS in order to provide users with high quality recommendations based on observed user ratings *and* texts of product reviews. The underlying motivation is that, when a user scores only a few products, the texts used in the reviews represent a significant source of information, thereby enhancing the predictive ability of the model. Our approach adopts a variational auto-encoder (VAE) architecture as a deep generative latent model for an ordinal matrix encoding ratings and a document-term matrix encoding the reviews. Taking into account both matrices as model inputs, deepLTRS uses a neural network to capture the relationship between latent factors and latent topics. Moreover, a user-majoring encoder and a product-majoring encoder are constructed to jointly capture user and product preferences. Due to the specificity of the model structure, an original row-column alternated mini-batch optimization algorithm is proposed to deal with user-product dependencies and computational burden. Numerical experiments on simulated and real-world data sets demonstrate that deepLTRS outperforms the state-of-the-art, in particular in context of extreme data sparsity.

Keywords: Recommender Systems, Learning Preferences or Rankings, Collaborative Filtering, Topic Modelling

*Corresponding author
Email address: dingge.liang@inria.fr (Dingge Liang)

1. Introduction and related works

1.1. Context and problem

In the current era of information explosion, recommendation systems have become central tools in a wide range of applications ranging from e-commerce [1] to the global positioning of IoT devices [2]. Examples of recommended objects include movies, songs, books, hotels, as well as restaurants to name just a few. At the core of the research on recommendation systems, we point out a widely adopted collaborative filtering (CF) approach [3], which relies on user historical preferences on a set of items. Generally, by converting the list of users and products into a user-item rating matrix, a CF-based recommender system can be considered as completing the rating matrix based on observed entries [4].

While most of users only ranked few products, the rating matrix is usually large and extremely sparse due to massive amount of missing values. Considering that many consumers also use texts to express various opinions along with the scores, reviews can contain crucial information from different aspects about the products, compared to a single rating. Therefore, efforts have been put recently in developing algorithms capable of dealing not only with ratings but also with other sources of information like text reviews to address the matrix completion problem in the case of high data sparseness [5].

1.2. Related works

A long series of techniques have been proposed in the literature for recommendations. On a general point of view, we list approaches depending on the source of information used in the system and the characteristics of the model: *rating-based*, *rating-with-text based* and *deep structure methods*. We briefly review them hereafter.

Rating-based. On the one hand, most algorithms have been proposed on the basis of the sole knowledge of ratings. For instance, HPF [6] assumes that the observed rating matrix is drawn from a Poisson distribution by combining a sparsity model (absence of a rating) with a single response model for rating values. To better capture the relationship between sparsity and response models, HCPF [7] allows to choose the most

appropriate response model from a family of additive exponential dispersion methods. More recently, CCPF [8] was introduced by coupling a hierarchical Poisson factorization with an arbitrary data-generating model among three different methods: mixture models, linear regression and matrix factorization. However, without introducing a modelling framework specific to text reviews, their performances proved to be limited in Section 4.3.

Rating-with-text based. Since the product ratings are usually paired with text reviews, another set of recommender systems exploit both ratings and texts to improve predictions. In this line of methods, CTR [9] and obi-CTR [10] rely on the probabilistic matrix factorization (PMF) [11] and assume that topics are drawn from a Dirichlet distribution. Nonetheless, neither of them formulates the relationship between user latent factors and latent topics. HFT [12] combines latent rating factors with latent review topics by maximizing a penalized log-likelihood where the first term accounts for rating distribution and the second term accounts for the words distribution over latent topics. However, it suffers from the limitation that the number of latent factors should be equal to the number of latent topics. ALFM [13] breaks this limitation by associating latent factors with different aspects and each aspect is represented as a probability distribution of latent topics. The overall rating is computed through a linear combination of all the aspect ratings. Unfortunately, it turns out that its performance are affected by scattered data distribution (*see* Appendix C).

Deep structure methods. Deep neural network (DNN) based approaches have recently shown efficacy on feature representations learning and have been extensively explored in the research of recommender systems [14]. Among state-of-the-art techniques, DeepCoNN [15] uses CNNs to learn representations of users and products from reviews and a regression layer is subsequently introduced for the prediction of ratings. It is limited to the assumption that reviews are only available during the training phase. As an extension of DeepCoNN, TransNet was introduced in [16] with an additional layer that allows the model to also generate approximate comments during test and improves prediction performance. However, in these two models, latent factors are barely obtained by learning reviews, neither of them consider observed ratings during the generative

phase. Although DNNs have achieved great success in representation learning, they require a significant amount of hyper-parameters for training, which result in high computational costs [17]. Thus, in [18], DLFM was proposed following the principle of deep forest [17]. Through sequentially connecting several layers, the output of the precedent layer is taken as the input of the next layer without back propagation. Finally, the best approximation is selected from outputs of all layers. It is worth mentioning that this work is mainly focused on reducing computations, which is beyond the scope of our work.

1.3. Contributions of our work

In order to both improve the robustness to data sparsity and the interpretability of recommendations, we introduce here the deepLTRS, which takes into account both observed ratings and the textual information collected in product reviews as the model input. DeepLTRS extends the PMF [11] by relying on recent auto-encoding extensions [19, 20]. Considering the review part, we use the ProdLDA model introduced by [19] to obtain higher quality topics with respect to a standard LDA [21]. Our goal here is to show that, regardless the parsimony of our model, we are still able to improve performance compared to the state-of-the-art techniques. Our approach has the following key-features:

- a variational auto-encoder (VAE) architecture is used as a generative latent model for both an ordinal matrix encoding ratings and a document-term matrix encoding reviews;
- since the connection between latent factors and the review data is difficult to formulate, we adopt a neural network to capture the relationship between latent representations and latent topics;
- one user-majoring encoder and another product-majoring encoder are constructed to jointly capture user and product preferences. Then, two different decoders are designed for ratings and reviews separately;
- due to the specific model structure, an original strategy of alternating rows and columns in mini-batch optimization is proposed to deal with user-product depen-

dencies and to reduce the computational costs. We further provide a theoretical proof of the unbiasedness of our empirical loss estimator.

2. A rating-and-review based recommender system

2.1. Framework and notations

In this work, we consider data sets involving M users who are scoring and reviewing P products. Such data sets can be encoded by two matrices: an ordinal data matrix Y accounting for the *scores* that users assign to products and a document-term matrix (DTM) W encoding the *reviews* that users write about products. Necessary symbols are summarized in Table 1.

Ordinal data. The ordinal data matrix Y in $\mathbb{N}^{M\times P}$ is such that Y_{ij} corresponds to the score that the i-th user assigns to the j-th product. This matrix is usually extremely sparse in practice (most of its entries are missing) corresponding to users *not* scoring nor reviewing some products. Conversely, when a score is assigned, it takes values in $\{1,\dots,H\}$ with $H>1$. Henceforth, we assume that an ordinal scale is consistently defined. For instance, when customers evaluate products, 1 always means "very poor" and H is always associated with "excellent" reviews. The number of ordered levels H is assumed to be the same for all (not missing) Y_{ij}. If it is not the case, a scale conversion pre-processing algorithm [22] can be employed to normalize the number of levels.

Text data. By considering all the available reviews, it is possible to store all the different vocables employed by the users into a *dictionary* of size V. Thenceforth, we denote by $W^{(i,j)}$ a row vector of size V encoding the review by the i-th user to the j-th product. The v-th entry of $W^{(i,j)}$, denoted by $W_v^{(i,j)}$, is the number of times (possibly zero) that the word v of the dictionary appears into the corresponding review. The document-term matrix W is obtained by concatenation of all the row vectors $W^{(i,j)}$. For the sake of clarity, we assume that the review $W^{(i,j)}$ exists if and only if Y_{ij} is observed. Note that, since each row in W corresponds to one (and only one) not missing entry in Y, the number of rows in the DTM is the same as the number of observed non-missing values in Y.

Table 1: List of all the model parameters

Parameter	**Description**
M, P	number of users and products
i, j	user and product indexes
D	dimension of latent factors
V	number of words in the dictionary considered
Y	an ordinal matrix in $\mathbb{N}^{M \times P}$ encoding ratings
W	a document-term matrix encoding reviews
b^u, b^p	user and product biases
ϵ	a residual term
K	number of latent topics
θ_{ij}	topic proportions for the pair (i, j)
f_γ	a function parametrized by γ to capture the relationship between latent factors and latent topics
L_{ij}	number of words in review $W^{(i,j)}$
β	a matrix in $[0, 1]^{V \times K}$ whose entry β_{vk} is the probability that vocable v occurs in topic k

2.2. Generative model of deepLTRS

It is now assumed that each user i and product j have the latent representations R_i and C_j in a low-dimensional space $\mathbb{R}^D$, with $D \ll \min\{M, P\}$.

Ratings. The following generative model is now considered for the ratings

$$Y_{ij} = \langle R_i, C_j \rangle + b_i^u + b_j^p + \epsilon_{ij}, \quad \forall i = 1, \dots, M, \forall j = 1, \dots, P, \tag{1}$$

where $\langle \cdot, \cdot \rangle$ is the standard scalar product and b_i^u, b_j^p are two unknown real parameters accounting for biases specific to users and products respectively. Finally, the residuals ϵ_{ij} are assumed to be i.i.d. normally distributed random variables: $\epsilon_{ij} \sim \mathcal{N}(0, \eta^2)$.

In the following, R_i and C_j are seen as random vectors with zero mean and the identity matrix of dimension D as the variance, such that

$$\begin{aligned} R_i &\overset{\text{i.i.d}}{\sim} \mathcal{N}(0, I_D), \qquad \forall i \\ C_j &\overset{\text{i.i.d}}{\sim} \mathcal{N}(0, I_D), \qquad \forall j \end{aligned} \tag{2}$$

with R_i and C_j assumed independent. The unbiased version of this model (i.e. with $b_i^u = b_j^p = 0$) is the well known PMF [11]. Note that, due to rotational invariance of PMF, the choice of isotropic prior distributions for R_i and C_j is in no way restrictive (*see* Appendix A).

Reviews. We now extend the generative model outlined in the previous section to account for the document-term matrix W. Following the LDA model, each document $W^{(i,j)}$ is drawn from a mixture distribution over a set of K latent topics. In deepLTRS, topic proportions in the document $W^{(i,j)}$ are denoted by θ_{ij}, a vector lying in the $K-1$ simplex, and we have $\theta_{ij} \in [0,1]^K$, such that $\sum_{k=1}^K \theta_{ij} = 1$. Moreover, we assume that

$$\theta_{ij} = \sigma(f_\gamma(R_i, C_j)), \qquad \forall i, j \tag{3}$$

where $f_\gamma : \mathbb{R}^{2D} \to \mathbb{R}^K$ is a continuous function approximated by a neural network parametrized by γ to capture the relationship between latent factors and latent topics, $\sigma : \mathbb{R}^K \to \mathbb{R}^K$ denotes the softmax function defined by $(\sigma(z))_k = \frac{\exp(z_k)}{\sum_{l=1}^K \exp(z_l)}$, $k \in 1, \dots, K$, where $(\sigma(z))_k$ is the k-th entry of vector $\sigma(z) \in [0,1]^K$ and z denotes here a generic vector in $\mathbb{R}^K$.

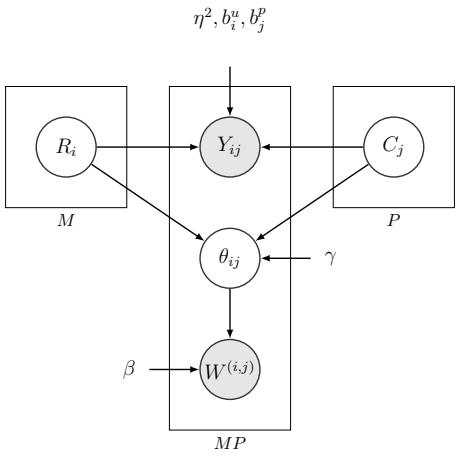

Figure 1: Graphical representation of the generative model for deepLTRS (variational parameters are not included).

As in LDA, each document $W^{(i,j)}$ is seen as a vector in $\mathbb{N}^V$ (we recall that V is the dictionary size) obtained as

$$W^{(i,j)}|\theta_{ij} \sim \text{Multinomial}(L_{ij}, \beta\theta_{ij}), \qquad \forall i, j \tag{4}$$

where L_{ij} is the number of words in the review $W^{(i,j)}$ and $\beta \in [0,1]^{V\times K}$ is a matrix whose entry β_{vk} is the probability that vocable v occurs in topic k. By construction, $\sum_{v=1}^{V} \beta_{vk} = 1, \forall k$. In addition, conditionally to vectors θ_{ij}, all the reviews $\{W^{(i,j)}\}_{i,j}$ are independent random vectors.

A graphical representation of the generative model described so far can be seen in Figure 1.

3. Variational auto-encoding inference

This section now details the auto-encoding variational inference procedure and proposes an original row-column alternate mini-batch strategy to reduce the computational burden.

3.1. Variational lower bound (ELBO)

Let us denote by $\Theta = \{\eta^2, \gamma, \beta, b^u, b^p\}$ the set of the model parameters introduced so far. A natural inference procedure associated with the proposed generative model would consist in looking for $\hat{\Theta}_{ML}$ maximizing the (integrated) log-likelihood of the observed data (Y, W). Unfortunately, this quantity is not directly tractable and we rely on a variational lower bound to approximate it. Let us consider a joint distribution $q(\cdot)$ over the pair (R, C) of all $(R_i)_i$ and $(C_j)_j$. Thanks to the Jensen inequality, it holds that

$$\begin{aligned}
\log p(Y, W|\Theta) \geq & \mathbb{E}_{q(R,C)}\left[\log \frac{p(Y, W, R, C|\Theta)}{q(R, C)}\right] \\
= & \mathbb{E}_{q(R,C)}\left[\log p(W, Y|R, C, \Theta) + \log \frac{p(R, C)}{q(R, C)}\right] \\
= & \mathbb{E}_{q(R,C)}\left[\log p(W|R, C, \beta)\right] \\
& + \mathbb{E}_{q(R,C)}\left[\log p(Y|R, C, \gamma, \eta^2, b_u, b_p)\right] \\
& - D_{KL}(q(R, C)||p(R, C)),
\end{aligned} \tag{5}$$

where D_{KL} denotes the Kullback-Leibler divergence between the variational posterior distribution of the latent row vectors $(R_i)_i, (C_j)_j$ and their prior distributions. The above inequality holds for every joint distribution $q(\cdot)$ over the pair (R, C). In order to deal with a tractable family of distributions, the following *mean-field* assumption is made

$$q(R, C) = q(R)q(C) = \prod_{i=1}^{M}\prod_{j=1}^{P} q(R_i)q(C_j). \tag{6}$$

Moreover, since R_i and C_j follow Gaussian prior distributions (Eq. (2)), $q(\cdot)$ is assumed to be as follows

$$q(R_i) = g(R_i; \mu_i^R := h_{1,\phi}(Y_i, W^{(i,\cdot)}), S_i^R := h_{2,\phi}(Y_i, W^{(i,\cdot)})), \tag{7}$$

and

$$q(C_j) = g(C_j; \mu_j^C := l_{1,\iota}(Y^j, W^{(\cdot,j)}), S_j^C := l_{2,\iota}(Y^j, W^{(\cdot,j)})), \tag{8}$$

where $g(\cdot; \mu, S)$ is the pdf of a Gaussian multivariate distribution with mean μ and variance S. The two matrices S_i^R and S_j^C are assumed to be diagonal matrices with D elements. In addition, Y_i (respectively Y^j) denotes the i-th row (column) of Y, $W^{(i,\cdot)} := \sum_j W^{(i,j)}$ corresponds to a document concatenating all the reviews written by user i and $W^{(\cdot,j)} := \sum_i W^{(i,j)}$ corresponds to all the reviews about the j-th product. The functions $h_{1,\phi}$ and $h_{2,\phi}$ encode elements of $\mathbb{R}^{P+V}$ to elements of $\mathbb{R}^D$. Similarly, $l_{1,\iota}$ and $l_{2,\iota}$ encode elements of $\mathbb{R}^{M+V}$ to elements of $\mathbb{R}^D$. These functions are known as the network *encoders* parametrized by ϕ and ι, respectively.

Thanks to Eqs. (1)-(4)-(6)-(7)-(8) and by computing the KL divergence in Eq. (5), the *evidence lower bound* (ELBO) on the right hand side of Eq. (5) can be further developed as

$$\begin{aligned}
\text{ELBO}(\overline{\Theta}) = &\sum_{i,j} \left(\mathbb{E}_{q(R_i, C_j)} \left[-\frac{1}{2} \left(\frac{(Y_{ij} - (R_i^T C_j + b_u^i + b_p^j))^2}{\eta^2} + \log \eta^2 \right) \right] \right) \\
&+ \sum_{i,j} \left(\mathbb{E}_{q(R_i, C_j)} \left[\left(W^{(i,j)} \right)^T \log \left(\beta \sigma (f_\gamma (R_i, C_j)) \right) \right] \right) \\
&- \sum_i \left[-\frac{1}{2} \left(tr(S_i^R) + (\mu_i^R)^T \mu_i^R - D - \log |S_i^R| \right) \right] \\
&- \sum_j \left[-\frac{1}{2} \left(tr(S_j^C) + (\mu_j^C)^T \mu_j^C - D - \log |S_j^C| \right) \right] + \xi
\end{aligned} \tag{9}$$

where now $\overline{\Theta} := \{\eta^2, \gamma, \beta, b_u, b_p, \phi, \iota\}$ denotes the set of generative model *and* variational parameters, ξ is a constant term that includes all the elements not depending on $\overline{\Theta}$.

VAE structure. The deep view of deepLTRS is shown in Figure 2. We point out that, in the deep learning literature, the term *encoder* denotes a neural network that maps the observed data into a lower dimension space [23]. This is precisely what the functions $h_{1,\phi}, h_{2,\phi}$ and $l_{1,\iota}, l_{2,\iota}$ do, by mapping the observed data from $\mathbb{R}^{P+V}$ and $\mathbb{R}^{M+V}$, respectively, to the variational parameters in $\mathbb{R}^D$. Symmetrically, the term *decoder* denotes a neural network that maps the "compressed" data from the lower dimension space to the original dimension. In deepLTRS, this role is played by

- RC^T, the matrix product of R and C, that maps the lower dimension representations to the "reconstructed" ordinal data matrix $\hat{Y}$;
- β, which maps the topic proportions from $\mathbb{R}^K$ into vectors in $\mathbb{R}^V$ (the "reconstructed" rows of $\hat{W}$).

Unconstrained β. From a practical point of view, when optimizing the ELBO with respect to $\overline{\Theta}$, we remove the constraint on the columns of β, that have no longer to lie on the $V-1$ simplex. This assumption corresponds to the ProdLDA model introduced by [19]. In order to obtain consistent parameters for the multinomial distribution followed by $W^{(i,j)}$, a softmax function $\sigma(\cdot)$ is applied to the product $\beta\theta_{ij}$ instead of θ_{ij} only.

3.2. Monte Carlo EM algorithm and mini-batching

The maximization of $ELBO(\overline{\Theta})$ in Eq. (9) can be performed by means of a Monte Carlo EM algorithm that alternates a sampling step, to numerically approximate the expectations, with a maximization step to update the value of the parameters $\overline{\Theta}$. This algorithm was adopted previously for standard variational auto-encoders (VAEs) in [24, 25]. As in those papers (and in contrast with what happens in simpler latent variable models), there is no close formula for the maximization step and gradient descent algorithms are employed to maximize the (Monte Carlo) lower bound with respect

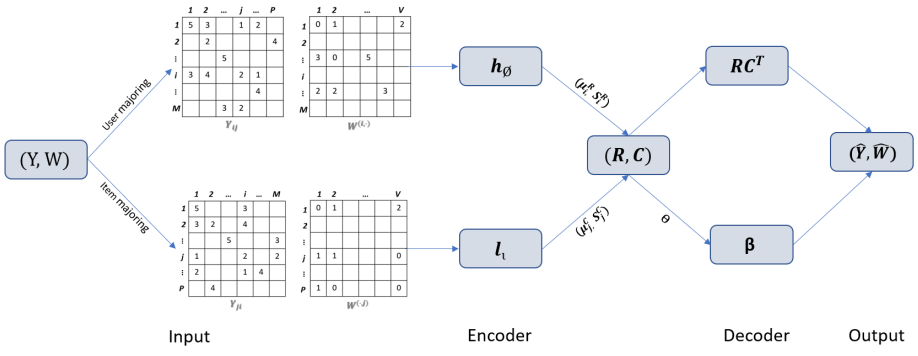

Figure 2: A deep-learning-like model view of DeepLTRS.

to $\overline{\Theta}$[1]. Performing mini-batch optimization paired with stochastic gradient descent algorithms [26] is necessary to reduce the computational burden when working with large data sets. However, there is a substantial difference between the model we adopt and standard VAEs. Whereas in a standard VAE the ELBO can be written as the sum of as many terms as the number of observations, $ELBO(\overline{\Theta})$ in Eq. (9) does not factorize over the number of observations. In more detail, the model sees the pair $(Y_{ij}, W^{(i,j)})$ as one observation. Assuming for simplicity that there is no missing data, the total number of observations is MP. The ELBO in Eq. (9) is unfortunately *not* the sum of MP terms due to the graphical structure of the generative model in Figure 1. Nevertheless, stochastic gradient descent can still be performed in our case thanks to what follows.

Let us define

$$\begin{aligned} z_i := & - D_{KL}(q(R_i) \parallel p(R_i)) \\ & + \mathbb{E}_{q(R_i,C)} \left[\log p(Y_i | R_i, C, \overline{\Theta}) + \log p(W^{(i,\cdot)} | R_i, C, \overline{\Theta}) \right], \end{aligned} \tag{10}$$

for all $i \in \{1, \ldots, M\}$, with Y_i, R_i and $W^{(i,\cdot)}$ previously defined. We now introduce a new random variable Z such that

$$\pi := \mathbb{P}\{Z = z_i\} = \frac{1}{M}. \tag{11}$$

A sample of Z corresponds to a uniformly at random extraction of one row in Y.

The proposition below states that the gradient of MZ with respect to **all parameters but** ι (the product encoder parameter) is an unbiased estimator of the ELBO's gradient with respect to the same parameters.

Proposition 1. *For the random variable Z, whose probability mass function is defined in Eq.* (11)*, it holds that*

$$\mathbb{E}_\pi \left[\nabla_{(\eta^2, \gamma, \beta, \phi, b^u, b^p)} (MZ) \right] = \nabla_{(\eta^2, \gamma, \beta, \phi, b^u, b^p)} \left(\textit{ELBO}(\overline{\Theta}) \right), \tag{12}$$

where $\nabla_x(f)$ denotes the gradient of a function $f(\cdot)$ with respect to the variable(s) x.

[1] We point out that *maximizing* $ELBO(\overline{\Theta})$ with respect to $\overline{\Theta}$ is the same as *minimizing* $-ELBO(\overline{\Theta})$ and this is why we mention gradient *descent* algorithms.

Proof. First, let us notice that, due to the definition of z_i and the assumption in Eq. (6), it holds that:

$$\begin{aligned}\nabla_{(\eta^2,\gamma,\beta,\phi,b^u,b^p)} z_i = \nabla_{(\eta^2,\gamma,\beta,\phi,b^u,b^p)} \Big[& - D_{KL}(q(R_i) \parallel p(R_i)) \\ & + \sum_{j=1}^{P} (\mathbb{E}_{q(R_i,C_j)}[\log p(Y_{ij}|R_i, C_j, \overline{\Theta})]) \\ & + \sum_{j=1}^{P} (\mathbb{E}_{q(R_i,C_j)}[\log p(W^{(i,j)})|R_i, C_j, \overline{\Theta})]) \Big].\end{aligned} \tag{13}$$

Then, Eq. (11) leads to

$$\mathbb{E}_\pi \left(\nabla_{(\eta^2,\gamma,\beta,\phi,b^u,b^p)} MZ \right) = \sum_{i=1}^{M} \nabla_{(\eta^2,\gamma,\beta,\phi,b^u,b^p)} z_i,$$

and replacing Eq. (13) into the above equation we obtain Eq. (12), recalling that $\nabla_{(\eta^2,\gamma,\beta,\phi,b^u,b^p)}(\cdot)$ does not include the partial derivative with respect to ι. Thus

$$\nabla_{(\eta^2,\gamma,\beta,\phi,b^u,b^p)} \left(-D_{KL}(q(C_j) \parallel p(C_j)) = 0. \right.$$

□

Similarly, the quantity

$$\begin{aligned}w_j := & - D_{KL}(q(C_j) \parallel p(C_j)) \\ & + \mathbb{E}_{q(R,C_j)} \left[\log p(Y^j|R, C_j, \overline{\Theta}) + \log p(W^{(\cdot,j)}|R, C_j, \overline{\Theta}) \right],\end{aligned} \tag{14}$$

can be used to introduce an unbiased estimator of $\nabla_{(\eta^2,\gamma,\beta,\iota,b^u,b^p)}(\text{ELBO}(\overline{\Theta}))$, where the partial derivative of the lower bound with respect to ϕ (the user encoder parameter) is now not taken into account.

The above proposition justifies a maximization of the ELBO which alternates rows and columns mini-batching. First, rows of Y are extracted uniformly at random, with re-injection, in such a way that we obtain a collection of random variables $Z_1, Z_2, \ldots, Z_n, \ldots$, being i.i.d. copies of Z. Hence, each Z_n is used to update all the model parameters but ι. Moreover, when performing row mini-batch, the current value of the whole matrix C is used, as well as all columns in Y and all the reviews by products. Conversely, when performing column mini-batch, the whole matrix R is

Algorithm 1 Mini-batch estimation of deepLTRS

procedure ESTIM(Y, W)
  $(R, C) \leftarrow$ INIT($\mathcal{N}(0, 1)$) ▷ Initialize with Gaussian distribution
  Initialization parameters $\overline{\Theta}$
  while $\log p\left(Y, W | R, C, \overline{\Theta}\right)$ increases **do**
    for all i: ▷ Row-majoring mini-batch
      $R_i \sim \mathcal{N}(h_{1,\phi}, h_{2,\phi})$ ▷ Sampling users
      $\overline{\Theta}_{-\iota} = \text{OPTIM}\big(\log p(Y, W | R_i, C, \eta^2, \gamma, \beta, b^u, b^p, \phi)\big)$
    for all j: ▷ Column-majoring mini-batch
      $C_j \sim \mathcal{N}(l_{1,\iota}, l_{2,\iota})$ ▷ Sampling products
      $\overline{\Theta}_{-\phi} = \text{OPTIM}\big(\log p(Y, W | R, C_j, \eta^2, \gamma, \beta, b^u, b^p, \iota)\big)$
  return $(R, C, \overline{\Theta})$

used, as well as all rows of Y and all the reviews by users. In this case, the optimization is performed with respect to all the model parameters but ϕ. The pseudo code in Algorithm 1 summarizes the procedure of estimation detailed so far for deepLTRS.

4. Numerical experiments on simulated data

Before to go further, let us firstly describe the architecture of deepLTRS and the simulation setup that will be used later in this section.

4.1. Architecture and simulation setup

Architecture of deepLTRS. Our architecture involves three neural networks: two *encoders*, and an *internal neural net* working in the low dimensional space. The first encoder (user encoder) models h_ϕ in Eq. (7). It has two hidden layers, with 50 neurons each, equipped with a softplus activation function and followed by a dropout with a ratio of 0.2 and batch normalization. The second encoder (product encoder) models l_ι in Eq. (8). It also has two hidden layers, the number of neurons and subsequent operations are the same as the user encoder. The internal neural net models f_γ in Eq. (3). It has one hidden layer equipped with 80 neurons and a softmax activation function. In deepLTRS, the decoding phase is managed in a simpler way: i) *ratings*: a scalar product between R

Table 2: Simulation of ratings and reviews

(a) Score assignments.

	cluster 1	cluster 2
cluster 1	$\sim \mathcal{N}(2,1)$	$\sim \mathcal{N}(3,1)$
cluster 2	$\sim \mathcal{N}(3,1)$	$\sim \mathcal{N}(2,1)$

(b) Topic assignments.

	cluster 1	cluster 2
cluster 1	A	B
cluster 2	C	D

and C, followed by the intercepts summation decodes the ratings; ii) *reviews*: a linear map, involving the decoding matrix β, is followed by a softmax function in order to produce the probabilities of word occurrences. We stress that the decoding of reviews, from θ to the reconstructed matrix W, can be seen as a simple neural net with no hidden layers.

Simulation setup. An ordinal data matrix Y with $M = 750$ rows and $P = 600$ columns is simulated according to a latent continuous cluster model. The rows and columns of Y are randomly assigned to two latent groups, in equal proportions. Then, for each pair (i, j) corresponding to an entry of Y, the sampling of a Gaussian random variable Z_{ij} is detailed in Table 2a. In addition, the following thresholds $t_0 = -\infty, t_1 = 1.5, t_2 = 2.5, t_3 = 3.5, t_4 = +\infty$ are used to sample the note $Y_{ij} \in \{1, ..., 4\}$ as

$$Y_{ij} = \sum_{k=1}^{4} k \mathbf{1}(Z_{ij})_{]t_{k-1}, t_k[}. \tag{15}$$

Next, regarding the simulation of text reviews, four different texts from the BBC news are used to build a message for each note Y_{ij} according to the scheme summarized in Table 2b. One text is about the birth of Princess Charlotte, the second one is about black holes in astrophysics, the third one focus on British politics and the last one is about cancer diseases in medicine (denoted by A, B, C, D respectively). Thus, when the user i in cluster $X_i^{(R)} = 2$ rates the product j in cluster $X_j^{(C)} = 1$, a random variable $Z_{ij} \sim \mathcal{N}(3, 1)$ is sampled, then Y_{ij} is calculated via Eq. (15) and the review $W^{(i,j)}$ is built by random extraction of words from message C. All the sampled messages have an average length of 100 words. Finally, in order to introduce some noise, only 80% of words are extracted from the main topics, while the remaining 20% are extracted from

the other topics uniformly at random.

The time complexity of deepLTRS is linearly affected by the number of users, products and the vocabulary. For instance, for data with $M = 750, P = 600, V = 1034$, the training time of 20 epochs is $1111.57s$, while with $V = 558$, the execution takes $637.96s$.

4.2. DeepLTRS with and without text data

A first experiment highlights the interest of using the reviews to make more accurate rating predictions. To do so, 10 data sets are simulated according to the above simulation setup, with sparsity rates of Y (i.e. proportion of missing data over MP entries) varying in the interval $[0.5, 0.99]$. The ratios of training set, validation set and test set are 80%, 10% and 10% of the observed (i.e. not missing) simulated data. Two versions of deepLTRS are fitted to the simulated data, the first one accounting for both ordinal and text data (corresponding to the generative model described in Section 2) and the second one not using text, based on the PMF in Eq. (1). For the sake of simplicity, in both versions, as well as in all the experiments shown in this paper, b^u and b^p are fixed (and not estimated) to the average rating by users and products, respectively.

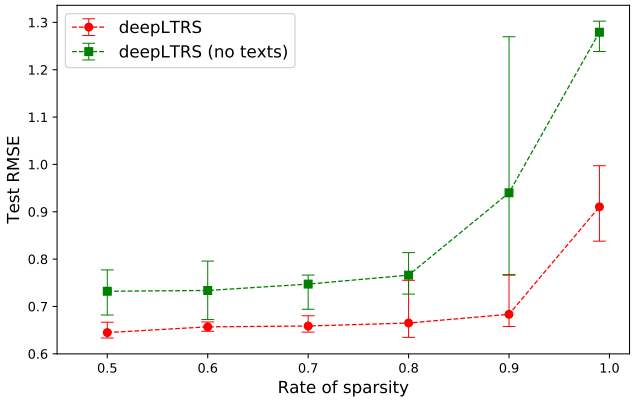

Figure 3: Comparison of deepLTRS with and without text information.

Figure 3 shows the evolution of the test RMSE of deepLTRS, with and without using text data for rating forecasts, versus the data sparsity level. We can observe that, even though both models suffer from the high data sparsity (increasing RMSE), the use of the

text greatly helps deepLTRS to maintain a high prediction accuracy for data sets with many missing values. Furthermore, the use of text reviews tends to reduce the variance of the deepLTRS predictions. More intuitively, the visualization of user and product embeddings of deepLTRS with and without text data is provided in Appendix B.

4.3. Benchmark and effect of data sparsity

In this part, we benchmark deepLTRS by comparing with some state-of-the-art methods, in condition of high data sparsity. The same experimental setup was used to benchmark deepLTRS (from now on, always accounting for text data). Our model is here compared to HFT, HPF [6], CCPF [8] and ALFM. Since for CCPF, many combinations of sparsity and response models exist, we select the pair of models having the best performance as described in [8].

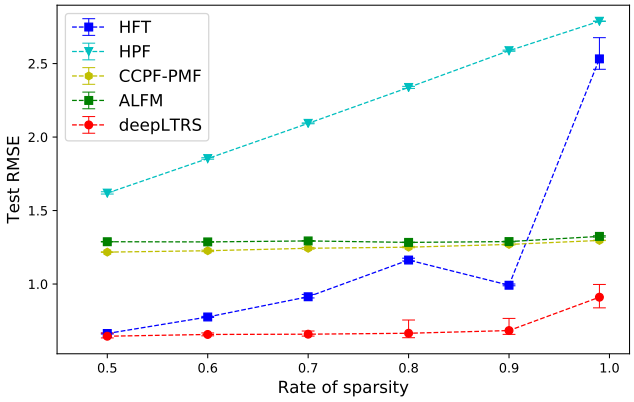

Figure 4: Test RMSE of models with different sparsity level on simulated data.

Figure 4 shows the evolution of the test RMSE for deepLTRS and its competitors. We first remark that, although HFT accounts for the text reviews, it does not perform very well in our simulated scenario and turns out to be very sensitive to the data sparsity. Second, HPF also appears to be quite sensitive to the data sparsity and it always performs worse than CCPF, ALFM and deepLTRS. Finally, although CCPF and ALFM present less sensitivity to the data sparsity, as the changes in the RMSE are small along with the sparsity level, generally HFT performs better than them when the sparsity is less than a certain threshold. In addition, even if the sparsity reaches 0.99, deepLTRS still

outperforms all other models. Let us recall that the simulation setup does not follow the deepLTRS generative model and therefore does not favor any method here.

5. Application on real-world data

We now consider applying deepLTRS to real-world datasets consisting of different product reviews from Amazon[2][3]. The data includes reviews (ratings, text), product metadata (descriptions, category, price, brand and image features) and links. In this section, deepLTRS is compared with previously mentioned models: HFT, HPF, CCPF, ALFM and TransNet.

Data pre-processing. In the pre-processing step, we kept records including user and product information, ratings and a plain-text review. For Amazon Fine Food dataset, we only considered users with more than 20 reviews and products reviewed by more than 50 users to obtain more meaningful information; for other three categories of Amazon product data, we used the reduced 5-core version where each of the remaining users and items have at least 5 reviews. Retained data were processed by removing all punctuations, numbers and stop words, then we deleted the words that appeared less than 3 times in the entire vocabulary for all data sets. The statistics of the processed data sets are given in Table 3.

Table 3: Statistics of evaluation data sets.

Dataset	#users	#items	#reviews	#total_words	#rest_words	sparsity
Fine Foods	1643	1733	32811	5743	3047	98.85%
Musical Instruments	1429	900	10254	15050	5846	99.20%
Patio	1686	962	13258	22441	8746	99.18%
Automotive	2928	1835	20467	20113	7737	99.62%

[2] https://snap.stanford.edu/data/web-FineFoods.html
[3] https://jmcauley.ucsd.edu/data/amazon/

Settings. Five independent runs of the algorithm were performed. For each run, we randomly selected 80% of the data as the training set, 10% samples for validation and the remaining 10% data as the test set. We trained our model for 100 epochs. As a method for stochastic optimization, we adopted an Adam optimizer [27], with a learning rate of $2e^{-3}$. The RMSE is calculated on both the validation and test set. Reported test RMSE is obtained when the RMSE on the validation set was the lowest, as for all methods.

Rating prediction. Table 4 presents the test RMSE for deepLTRS and its competitors on the predicted ratings for Amazon data sets. Since HFT is restricted by the fact that the numbers of latent factors and topics should be equal, we set $D = K = 50$. First of all, HPF and CCPF only considered the user rating information. By replacing the single Poison distribution in HPF with a mixture model, CCPF has made great improvements in RMSE. Next, the remaining four methods all consider both ratings and reviews.

Table 4: Test RMSE on Amazon data sets.

Data sets	HFT	HPF	CCPF-PMF
Fine Food	1.4477 ($\pm$0.0465)	2.9528 ($\pm$0.0144)	1.2913 ($\pm$0.0105)
Musical Instruments	1.3505 ($\pm$0.0061)	4.0926 ($\pm$0.0164)	1.1151 ($\pm$0.0242)
Patio	1.2183 ($\pm$0.0096)	3.8782 ($\pm$0.0051)	1.1353 ($\pm$0.0174)
Automotive	1.0844 ($\pm$0.0084)	4.3252 ($\pm$0.0041)	1.0105 ($\pm$0.0186)
Average	1.2752 ($\pm$0.3729)	3.8122 ($\pm$0.5220)	1.1381 ($\pm$0.1020)

Data sets	ALFM	TransNet	deepLTRS
Fine Food	1.0705 ($\pm$0.0014)	1.3783 ($\pm$0.0012)	**0.9788** ($\pm$0.0215)
Musical Instruments	**0.8929** ($\pm$0.0013)	1.0912 ($\pm$0.0057)	0.9702 ($\pm$0.0143)
Patio	1.0219 ($\pm$0.0027)	1.0589 ($\pm$0.0009)	**0.9855** ($\pm$0.0319)
Automotive	**0.8797** ($\pm$0.0016)	1.0649 ($\pm$0.0012)	0.9299($\pm$0.0511)
Average	0.9663 ($\pm$0.0819)	1.1483($\pm$0.1334)	**0.9661** ($\pm$0.0392)

Among them, TransNet and deepLTRS are deep-learning based models. It can be seen that, in general, ALFM and deepLTRS always have better performance than HFT and TransNet.

It is worth mentioning that deepLTRS outperforms ALFM on two data sets, Fine food and Patio, while ALFM has better performance on the other two data sets since when the data has many positive ratings, ALFM setup benefits from this configuration. Indeed, in the score generation phase, ALFM introduces the average of all ratings to the formula, which leads the predictions to a higher ranking level. When most of the scores of the experimental data are very positive, for example, a lot of scores are equal to 4 in Amazon data, ALFM can achieve very good results thanks to this average bias parameter. Nevertheless, when the score distribution of the data is more scattered, ALFM cannot perform well. Another analysis between the prediction of ALFM and deepLTRS on the simulated data is given in Appendix C.

6. Conclusion and perspectives

We introduced the deepLTRS model for rating recommendation using both the ordinal and text data available. Our approach adopted a VAE architecture as the deep generative latent model for both an ordinal matrix encoding the ratings, and a document-term matrix encoding the reviews. DeepLTRS presents the advantage to jointly learn representations of users and products through the alternated mini-batch optimization and a neural network was introduced to capture the relationship between latent factors and latent topics. Even with a simple topic model for the text part, we are still able to improve performance compared to the state-of-the-art techniques. Numerical experiments on simulated and real-world data sets show that our model outperforms other competitors in the context of high data sparsity.

We finally outline some research perspectives. First, although we mainly focused on the rating matrix, by exploiting the document-term matrix, the further ability of deepLTRS to predict the top words used by reviewers to comment products could be inspected in future works. Furthermore, in order to improve the modelling of text, we might replace LDA by a deep latent generative model, possibly involving RNNs [28]

or BERT [29] for review prediction. Moreover, the inference of the model parameters could be fine-tuned by means of the particle swarm optimization algorithm [30] for self-adaptation of the hyper-parameters and some recent researches [31, 32, 33] focusing on computational efficiency in the context of high-dimensional and sparse matrices in recommender systems could be considered in order to speed-up the learning process.

Acknowledgements

This work has been supported by the French government, through the 3IA Côte d'Azur Investment in the Future, project managed by the National Research Agency (ANR) with the reference numbers ANR-19-P3IA-0002.

References

[1] Z. Huang, D. Zeng, H. Chen, A comparison of collaborative-filtering recommendation algorithms for e-commerce, IEEE Intelligent Systems 22 (5) (2007) 68–78.

[2] H. Gao, Y. Xu, Y. Yin, W. Zhang, R. Li, X. Wang, Context-aware qos prediction with neural collaborative filtering for internet-of-things services, IEEE Internet of Things Journal 7 (5) (2019) 4532–4542.

[3] X. Su, T. M. Khoshgoftaar, A survey of collaborative filtering techniques, Advances in artificial intelligence 2009.

[4] A. Ramlatchan, M. Yang, Q. Liu, M. Li, J. Wang, Y. Li, A survey of matrix completion methods for recommendation systems, Big Data Mining and Analytics 1 (4) (2018) 308–323.

[5] L. Chen, G. Chen, F. Wang, Recommender systems based on user reviews: the state of the art, User Modeling and User-Adapted Interaction 25 (2) (2015) 99–154.

[6] P. Gopalan, J. M. Hofman, D. M. Blei, Scalable recommendation with hierarchical poisson factorization., in: UAI, 2015, pp. 326–335.

[7] M. Basbug, B. Engelhardt, Hierarchical compound poisson factorization, in: International Conference on Machine Learning, 2016, pp. 1795–1803.

[8] M. E. Basbug, B. E. Engelhardt, Coupled compound poisson factorization, arXiv preprint arXiv:1701.02058.

[9] C. Wang, D. M. Blei, Collaborative topic modeling for recommending scientific articles, in: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, 2011, pp. 448–456.

[10] C. Liu, T. Jin, S. C. Hoi, P. Zhao, J. Sun, Collaborative topic regression for online recommender systems: an online and bayesian approach, Machine Learning 106 (5) (2017) 651–670.

[11] A. Mnih, R. R. Salakhutdinov, Probabilistic matrix factorization, in: Advances in neural information processing systems, 2008, pp. 1257–1264.

[12] J. McAuley, J. Leskovec, Hidden factors and hidden topics: understanding rating dimensions with review text, in: Proceedings of the 7th ACM conference on Recommender systems, 2013, pp. 165–172.

[13] Z. Cheng, Y. Ding, L. Zhu, M. Kankanhalli, Aspect-aware latent factor model: Rating prediction with ratings and reviews, in: Proceedings of the 2018 world wide web conference, 2018, pp. 639–648.

[14] Z. Y. Khan, Z. Niu, S. Sandiwarno, R. Prince, Deep learning techniques for rating prediction: a survey of the state-of-the-art, Artificial Intelligence Review 54 (1) (2021) 95–135.

[15] L. Zheng, V. Noroozi, P. S. Yu, Joint deep modeling of users and items using reviews for recommendation (2017). arXiv:1701.04783.

[16] R. Catherine, W. Cohen, Transnets: Learning to transform for recommendation, in: Proceedings of the eleventh ACM conference on recommender systems, 2017, pp. 288–296.

[17] Z.-H. Zhou, J. Feng, Deep forest: towards an alternative to deep neural networks, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence, 2017, pp. 3553–3559.

[18] D. Wu, X. Luo, M. Shang, Y. He, G. Wang, M. Zhou, A deep latent factor model for high-dimensional and sparse matrices in recommender systems, IEEE Transactions on Systems, Man, and Cybernetics: Systems.

[19] A. Srivastava, C. Sutton, Autoencoding variational inference for topic models, arXiv preprint arXiv:1703.01488.

[20] A. B. Dieng, F. J. Ruiz, D. M. Blei, Topic modeling in embedding spaces, arXiv preprint arXiv:1907.04907.

[21] D. M. Blei, A. Y. Ng, M. I. Jordan, Latent dirichlet allocation, the Journal of machine Learning research 3 (2003) 993–1022.

[22] Z. Gilula, R. E. McCulloch, Y. Ritov, O. Urminsky, A study into mechanisms of attitudinal scale conversion: A randomized stochastic ordering approach, Quantitative Marketing and Economics 17 (3) (2019) 325–357.

[23] D. P. Kingma, M. Welling, An introduction to variational autoencoders, arXiv preprint arXiv:1906.02691.

[24] D. P. Kingma, M. Welling, Auto-encoding variational bayes, arXiv preprint arXiv:1312.6114.

[25] D. J. Rezende, S. Mohamed, D. Wierstra, Stochastic backpropagation and approximate inference in deep generative models, in: Proceedings of the 31st International Conference on International Conference on Machine Learning-Volume 32, JMLR. org, 2014, pp. II–1278.

[26] L. Bottou, Large-scale machine learning with stochastic gradient descent, in: Proceedings of COMPSTAT'2010, Springer, 2010, pp. 177–186.

[27] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980.

[28] A. F. Agarap, P. Grafilon, Statistical analysis on e-commerce reviews, with sentiment classification using bidirectional recurrent neural network (rnn), arXiv preprint arXiv:1805.03687.

[29] S. Xu, S. E. Barbosa, D. Hong, Bert feature based model for predicting the helpfulness scores of online customers reviews, in: Future of Information and Communication Conference, Springer, 2020, pp. 270–281.

[30] I. C. Trelea, The particle swarm optimization algorithm: convergence analysis and parameter selection, Information processing letters 85 (6) (2003) 317–325.

[31] X. Luo, Y. Yuan, M. Zhou, Z. Liu, M. Shang, Non-negative latent factor model based on β-divergence for recommender systems, IEEE Transactions on Systems, Man, and Cybernetics: Systems.

[32] X. Luo, M. Zhou, S. Li, D. Wu, Z. Liu, M. Shang, Algorithms of unconstrained non-negative latent factor analysis for recommender systems, IEEE Transactions on Big Data.

[33] X. Luo, W. Qin, A. Dong, K. Sedraoui, M. Zhou, Efficient and high-quality recommendations via momentum-incorporated parallel stochastic gradient descent-based learning, IEEE/CAA Journal of Automatica Sinica 8 (2) (2020) 402–411.

Appendix A. Rotational invariance in PMF

From Eqs.(1)-(2), it easily follows that

$$Y_{ij} \sim \mathcal{N}(b_i^u + b_j^p, D + \eta^2). \tag{A.1}$$

Now, instead of assuming isotropic prior distributions for R_i and C_j, we set

$$R_i \overset{\text{i.i.d}}{\sim} \mathcal{N}(0, \Sigma_R), \qquad \forall i$$
$$C_j \overset{\text{i.i.d}}{\sim} \mathcal{N}(0, \Sigma_C), \qquad \forall j$$

with Σ_R and Σ_C being symmetric positive definite matrices. Then

$$\Sigma_R = Q_R \Lambda_R Q_R^T,$$

with Λ_R being the diagonal matrix with the eigenvalues of Σ_R on the main diagonal and Q_R the orthogonal matrix of the corresponding eigenvectors. Similarly

$$\Sigma_C = Q_C \Lambda_C Q_C^T.$$

Then, the two random vectors $\overline{R}_i := \Lambda_R^{-\frac{1}{2}} Q_R^T R_i$ and $\overline{C}_j := \Lambda_C^{-\frac{1}{2}} Q_C^T C_j$ are independent and follow an isotropic Gaussian distribution each. Thus, if we set

$$Y_{ij} = \langle \overline{R}_i, \overline{C}_j \rangle + b_i^u + b_j^p + \epsilon_{ij},$$

we recover the marginal distribution in Eq. (A.1).

Appendix B. Interpretability on simulated data

This part aims at binging out the role of the latent embeddings of users and products in our model. Figure B.5 and Figure B.6 show the t-SNE representations of R_i and C_j for deepLTRS with and without text data, respectively, with data sparsity of 0.99. We note that the two (row and column) clusters are well separated despite the large degree of sparsity in Figure B.5, which is well representative of the simulation setup. However, without text data, the model does not capture well the structure of simulated data, as shown in Figure B.6. The visualization effects demonstrate that the addition of text information is important and useful for deepLTRS.

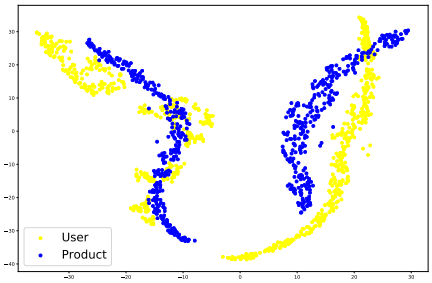

Figure B.5: Visualization of embeddings of deepLTRS with text data.

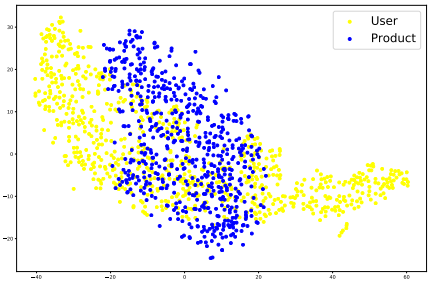

Figure B.6: Visualization of embeddings of deepLTRS without text data.

Appendix C. Analysis of the predictions

Here we analyze the differences between the prediction of ALFM and deepLTRS with the real ratings on the simulated data with the sparsity of 0.9. It is worth mentioning that, here we only compared ALFM with deepLTRS since ALFM has achieved significant performance on the experimental data sets.

Table C.5: Statistics of the predictions

Data set	Models	Min.	Max.	Mean
Simulated data_0.9	ALFM	1.8843	3.1557	2.5075
	deepLTRS	0.6013	4.3340	2.5638

Table C.5 demonstrates the statistics of the predictions. Different from ALFM that all predicted ratings are concentrated in the range of $[2, 3]$, the scores predicted by deepLTRS are well distributed in the interval $[1, 4]$, which is consistent with our initial setup. Moreover, Figure C.7 illustrates that the method used to calculate the predicted ratings in ALFM makes all predictions concentrated near the average. Thus, when the distribution of scores is very scattered (as in the simulated data), the test RMSE will become very large.

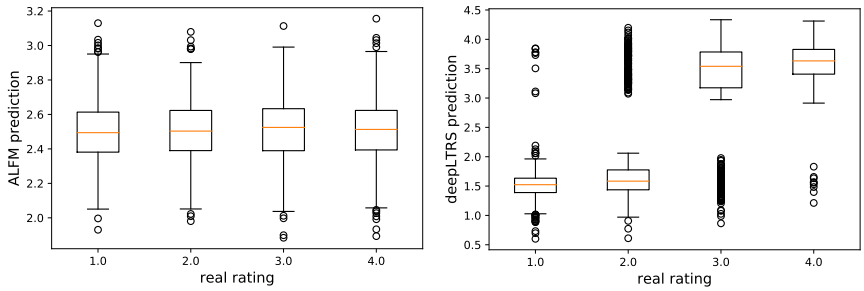

Figure C.7: Comparisons of the predictions of ALFM with actual ratings (Left) and deepLTRS with actual ratings (Right).

Appendix D. Interpretability on Amazon Fine Food

Figure D.8 presents a visualization with t-SNE of the high-dimensional latent representations ($D = K = 50$) of the users and products for the Amazon Fine food data. The overlapping regions of user and product representations correspond to users that are likely to comment on the corresponding products.

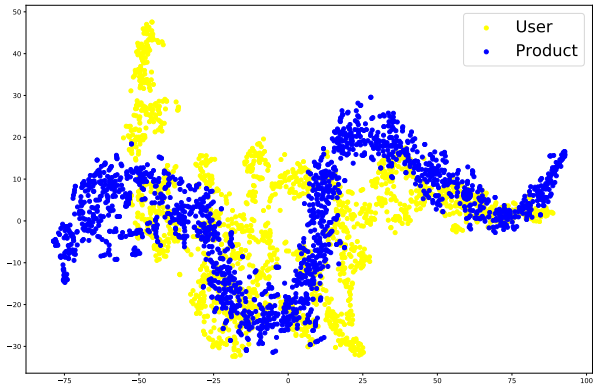

Figure D.8: Projection with t-SNE of user and product latent representations for the Amazon Fine Food data set.

In order to deeper understand the latent representations meaning, we provide in Figure D.9 and D.10 the visualization of user latent positions on two specific latent variables (variable 3 and 11) that can be easily interpreted according to average ratings and numbers of reviews the users give to products. Indeed, it clearly appears that

variable 11 captures the rating scale of Fine food users whereas variable 3 seems to encode the user activity (number of reviews).

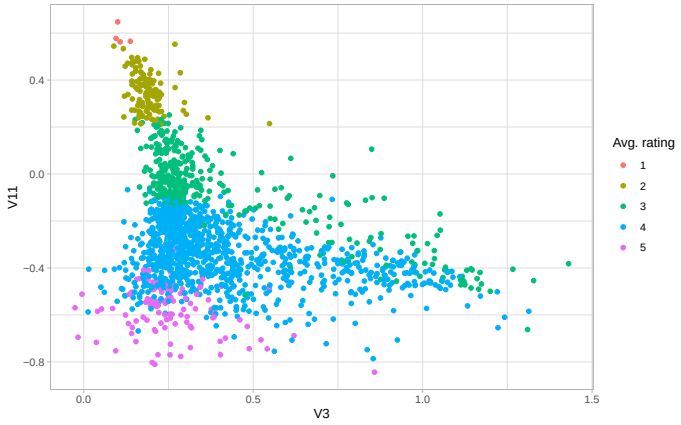

Figure D.9: Latent representation of users on variable 3 according to average ratings.

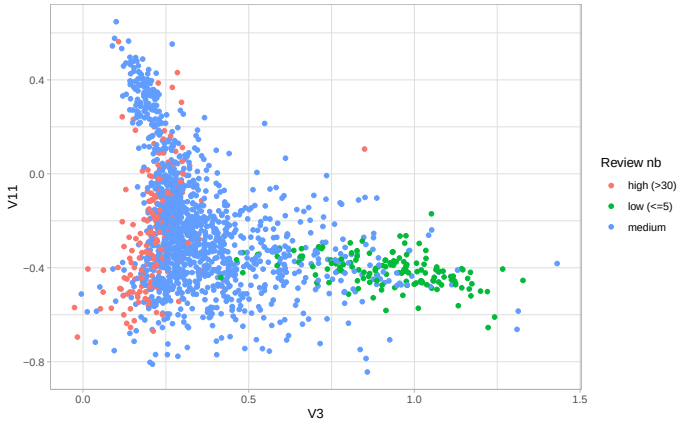

Figure D.10: Latent representation of users on variable 11 according to numbers of reviews they give to products.