DEEPLTRS: A DEEP LATENT RECOMMENDER SYSTEM BASED ON USER RATINGS AND REVIEWS

Anonymous authors
Paper under double-blind review

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. The addition of review information can alleviate data sparsity, thereby enhancing the predictive ability of the model. Our approach adopts a variational auto-encoder architecture as a generative deep latent variable model for both an ordinal matrix encoding users scores about products, and a document-term matrix encoding the reviews. Moreover, different from unique user-based or item-based models, deepLTRS assumes latent representations for both users and products. An alternated user/product mini-batching optimization structure is proposed to jointly capture user and product preferences. Numerical experiments on simulated and real-world data sets demonstrate that deepLTRS outperforms the state-of-the-art, in particular in contexts of extreme data sparsity.

1 INTRODUCTION AND RELATED WORKS

In recent research on recommendation systems, the collaborative and content-based filtering methods are widely used for completing a matrix of user ratings about products. Many applications fall in this context, ranging from e-commerce to the global positioning of IoT devices.

However, problems arise since these user/product matrices are extremely sparse in practice, which makes the inference of the non-observed entries challenging. A long series of approaches have been proposed to tackle this issue.

Introduced by Gopalan et al. (2015), hierarchical Poisson factorization (HPF) assumes that the user ratings follow Poisson distributions with latent user preferences and latent item attributes as parameters. Compared to HPF which is made of a sparsity model (absence of a rating) along with a response model (rating values), hierarchical compound Poisson factorization (HCPF, Basbug & Engelhardt, 2016) allows to choose the most appropriate response model from a family of additive exponential dispersion models, and better captures the relationship between sparsity and response models. More recently, coupled compound Poisson factorization (CCPF, Basbug & Engelhardt, 2017) was introduced as a more general framework capable of selecting an arbitrary data-generating model among different mixture models, matrix factorization models and linear regression models.

Unfortunately, although the aforementioned models can account for side information additionally to the user ratings, they do not introduce a modelling framework specific to the product reviews. However in practice, the product ratings are often paired with reviews that might contain crucial information about users preferences. Thus, in McAuley & Leskovec (2013), the hidden factors and hidden topics (HFT) model combines latent rating factors with latent review topics by defining a transformation. The collaborative topic regression model (CTR, Wang & Blei, 2011) assumes review documents are generated by a topic model and combines a rating matrix for recommendation. Nonetheless, when the auxiliary information is very sparse, the performance of CTR is usually limited. Worse more, both of them suffer from the limitation that the number of latent factors learned from the score should be equal to the number of latent topics learned from the reviews.

Other deep learning-based models help capture more complex patterns in the data. For instance, DeepCoNN (Zheng et al., 2017) uses CNNs to learn representations of users and products from reviews and then a regression layer is introduced for the prediction of ratings. However, it assumes that reviews are available only in the training phase. As an extension of DeepCoNN, TransNet (Catherine & Cohen, 2017) proposes an additional layer that allows the model to also generate approximate comments during testing.

We introduce here the deep latent recommender system (deepLTRS, Sections 2-3) for the completion of rating matrices, accounting for both observed ratings and the textual information collected in the product reviews. DeepLTRS extends the probabilistic matrix factorization (Mnih & Salakhutdinov, 2008) by relying on the idea of latent Dirichlet allocation (LDA, Blei et al., 2003) and its recent auto-encoding extensions (Srivastava & Sutton, 2017; Dieng et al., 2019). Therefore, our approach has the following advantages:

- we adopt a variational auto-encoder architecture as a generative deep latent variable model for both an ordinal matrix encoding the user/product scores, and a document-term matrix encoding the reviews. The latter helps at automatically capturing preferences based on additional reviews;
- the presence of text information alleviates data sparsity when few ratings are actually available in the recommender system;
- the numbers of latent factors and latent topics in deepLTRS are no longer limited. In particular, they can have different values;
- different from unique user-based or item-based models, an alternated row-column mini batch strategy is proposed to jointly optimize users and products preferences in the model;
- two decoding functions are introduced in our model, one aims at predicting missing ratings and the other is used to obtain probabilities of word occurrences, which can be further developed for the prediction of top words in reviews.

The framework of deepLTRS is shown in Figure 1. First, the latent representations of users and products are produced by two encoders separately: the input of the user encoder is obtained by concatenating a user-majoring score matrix with a corresponding review matrix; similarly, the entry of the product encoder is a concatenation of a item-majoring score matrix and a review matrix. Next, two decoding functions are introduced respectively for ratings and reviews. Finally, through this network, we obtained a completed score matrix and a word occurrence probability matrix. All parameters are optimized with a specific alternated row-column mini batch (Section 3.2). Our approach is tested on simulated and real world data sets (Section 4) and compared with other state-of-the-art approaches in contexts of extreme data sparsity.

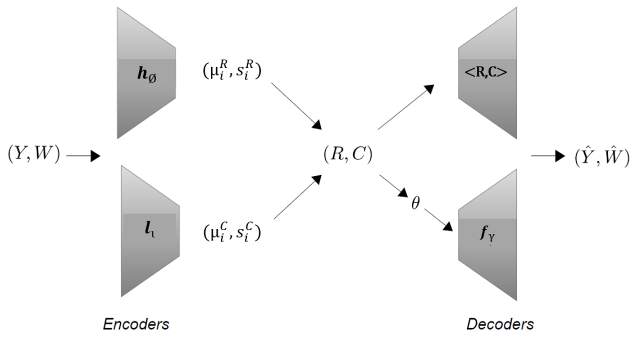

Figure 1: Summary of the deepLTRS model.

2 A RATING-AND-REVIEW BASED RECOMMENDER SYSTEM

2.1 PRELIMINARIES

In this work, we consider data sets involving M users scoring and reviewing P products. Such data sets can be encoded by two matrices: an ordinal data matrix $\boldsymbol{Y}$ accounting for the *scores* that users assign to products and a document-term matrix (DTM) $\boldsymbol{W}$ encoding the *reviews* that users comment on products.

Ordinal data. The ordinal data matrix $\boldsymbol{Y}$ in $\mathbb{N}^{M\times P}$ is such that Y_{ij} is the score that the i-th user assigns to the j-th product. This matrix can be very sparse in practice (most of its entries are missing) corresponding to users *not* scoring/reviewing some products. Conversely, when a score is assigned it takes values in $\{1,\ldots,H\}$ 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 (Gilula et al., 2018) 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 $\boldsymbol{W}$** is obtained by row 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 $\boldsymbol{W}$ corresponds to one (and only one) not missing entry in $\boldsymbol{Y}$, the number of rows in the DTM is the same as the number of observed values in $\boldsymbol{Y}$.

2.2 Generative models

It is now assumed that both users and products have latent representations in a low-dimensional space $\mathbb{R}^D$, with $D \ll \min\{M,P\}$. In the following, R_i denotes the latent representation of the i-th user, similarly C_j is the latent representation of the j-th product.

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

$$Y_{ij} = \langle R_i, C_j\rangle + \epsilon_{ij}, \forall i = 1,...,M, \forall j = 1,...,P, \tag{1}$$

where $\langle\cdot,\cdot\rangle$ is the standard scalar product and the residuals ϵ_{ij} are assumed to be i.i.d. and normally distributed random variables, with zero mean and unknown variance η^2:

$$\epsilon_{ij} \sim \mathcal{N}(0,\eta^2).$$

In the following, R_i and C_j are seen as random vectors, such that

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

with $R_i \perp\!\!\!\perp C_j$. This model is known as probabilistic matrix factorization (PMF, Mnih & Salakhutdinov, 2008). 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 (*cf.* Appendix A).

Reviews. We now extend PMF by also relying on R_i and C_j to characterize the document-term matrix $\boldsymbol{W}$. Following the generative model of LDA in Blei et al. (2003), each document $W^{(i,j)}$ is drawn from a mixture distribution over a set of K latent topics. The topic proportions in the document $W^{(i,j)}$ are denoted by θ_{ij}, a vector lying in the $K-1$ simplex.

In deepLTRS, each topic vector of proportions is now assumed to be sampled by decoding layers as follows

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

where $f_\gamma : \mathbb{R}^{2D} \to \mathbb{R}^K$ is a continuous function approximated by a neural network parametrized by γ and $\sigma(\cdot)$ denotes the softmax function. We emphasize that all the θ_{ij} are no longer independent contrary to the traditional LDA model.

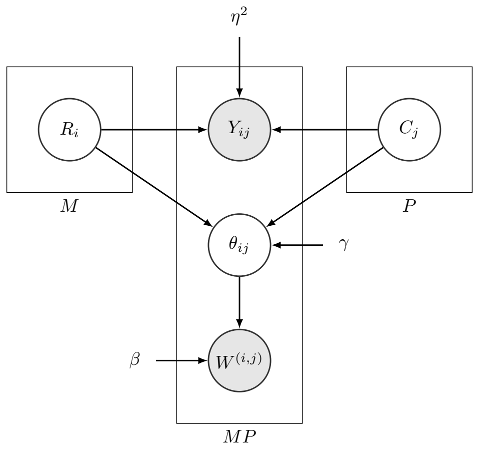

Figure 2: Graphical representation of the generative model (variational parameters are not included).

Furthermore, the multinomial PCA formulation of LDA is considered as

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

where L_{ij} is the number of words in the review $W^{(i,j)}$ and $\beta \in \mathbb{R}^{V \times K}$ is the 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 the vectors θ_{ij}, all the reviews $\{W^{(i,j)}\}$ are independent random vectors.

Finally, we emphasize that given the pair (R_i, C_j), Y_{ij} and $W^{(i,j)}$ are *not* assumed to be independent. Instead, we described a framework in which the dependence between them is completely captured by the latent embedding vectors R_i and C_j. A graphical representation of the generative model described so far can be seen in Figure 2.

3 Variational auto-encoding inference

3.1 Proposed inference

A natural inference procedure associated with the generative model proposed would consist in looking for estimates (η^2, γ, β) 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 Jensen inequality, it holds that

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

where the D_{KL} term denotes the Kullback-Leibler divergence between the variational posterior distribution of the latent row vectors $(R_i)_i, (C_j)_j$ and their prior distribution. The above inequality holds for every joint distribution $q(\cdot)$ over the pair (R, C). In order to work 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. Moreover, 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 follows:

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

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

Unconstrained β. From a practical point of view, when optimizing the ELBO with respect to Θ, 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 Srivastava & Sutton (2017) and which allows to obtain higher quality topics with respect to a standard LDA. 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 Alternated row-column mini-batch optimization

Due to the special architecture of deelLTRS, we introduced an alternated row-column mini-batch strategy to jointly optimize preferences of users and products.

Performing mini-batch optimization (paired with stochastic gradient descent algorithms) is necessary to reduce the computational burden when working with large data sets. However, the ELBO 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 2. Note that this point marks a substantial difference between the model we adopt and standard deep latent variable models (Kingma & Welling, 2013; Rezende et al., 2014) where the ELBO can be written as the sum of as many terms as the number of observations.

However, stochastic gradient descent can be performed thanks to the following property. Let us define

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

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

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

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

Proposition 1. *If re-injection is assumed, so that $Z_1, Z_2, \dots$ are i.i.d. replicates of Z, then:*

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

*where $\nabla_x(f)$ denotes the gradient of a function $f(\cdot)$ with respect to the variable(s) x (*cf *Appendix B for proof).*

The above proposition states that the gradient of MZ_i with respect to **all parameters but** ι (the column autoencoder's parameters) is an unbiased estimator of the ELBO's gradient with respect to the same parameters. Similarly, the quantity

$$w_j = -D_{KL}(q(C_j) \parallel p(C_j)) + \mathbb{E}_{q(R,C_j)} \left[\log p(Y^j|R, C_j, \Theta) + \log p(W^{(\cdot,j)}|R, C_j, \Theta)\right], \tag{13}$$

can be used to introduce an unbiased estimator of $\nabla_{(\eta^2,\gamma,\beta,\iota)}(\text{ELBO}(\Theta))$, where ϕ is now left out.

The above equations justify a maximization of the ELBO which alternates rows and columns mini-batching. When performing row mini-batching the whole matrix C is considered (and in general all columns in Y and all the reviews by product). The optimization is performed with respect to all the model parameters except for ι. When performing column mini-batching the whole matrix R is considered (and in general all rows of Y and all the reviews by user). The optimization is performed with respect to all the model parameters except for ϕ.

4 Numerical experiments

4.1 Simulated data and effect of the data sparsity

This section aims at highlighting the main features of deepLTRS on simulated data and to compare it with some state-of-the-art methods, in condition of high data sparsity.

Simulation setup. An ordinal data matrix $\boldsymbol{Y}$ 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, a Gaussian random variable Z_{ij} is sampled as Table 1. In addition, the following thresholds $t_0 = -\infty, \quad t_1 = 1.5, \quad t_2 = 2.5, \quad t_3 = 3.5, \quad t_4 = +\infty$ are used to sample the note $Y_{ij} \in \{1, ..., 4\}$ as

$$Y_{ij} = \sum_{k=1}^{4} k\mathbf{1}(Z_{ij})_{]t_{k-1},t_k[} \tag{14}$$

Next, four different texts from the BBC news (denoted by A, B, C, D) are used to build a message associated to the note Y_{ij} according to the scheme summarized in Table 2.

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, Y_{ij} is obtained via Eq.14 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 and in order to introduce some noise, only 80% of words are extracted from the main topics, while the remaining 20% is extracted from the other topics uniformly at random.

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

Table 1: Score assignments for simulated data.

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

Table 2: Topic assignments for simulated data.

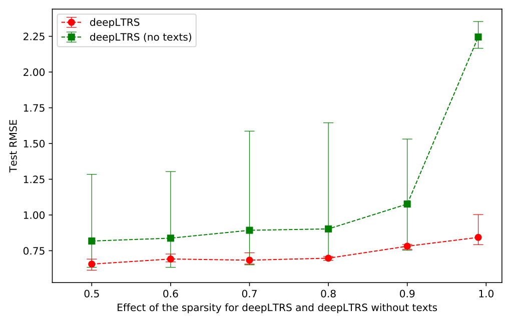

Figure 3: Comparison of deepLTRS with and without text information.

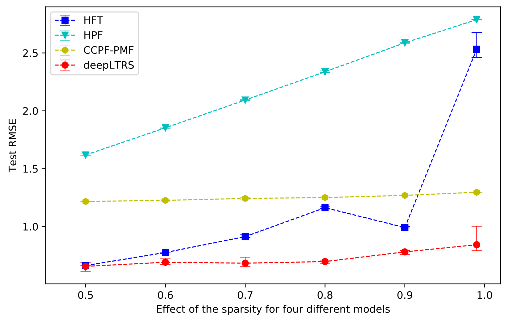

Figure 4: Test RMSE of models with different sparsity level on simulated data.

DeepLTRS with and without text data. We first run a simulation to highlight the interest in using the reviews to make more accurate predictions of the ratings. To do so, 10 data sets are simulated according to the above simulation setup, with sparsity rates varying in the interval $[0.5, 0.99]$. The ratio of the training, validation and test set is divided into 80%, 10% and 10%. Figure 3 shows the evolution of the test RMSE of deepLTRS (with $D = K = 50$), with and without using text data, versus the data sparsity level. One can observe that, even though both models suffer the high data sparsity (increasing RMSE), the use of the text greatly helps deepLTRS in maintaining a high prediction accuracy for data sets with many missing values. Furthermore, the use of text reviews greatly reduce the variance of predictions.

Benchmark. The same experimental setup was used to benchmark deepLTRS by comparing it to some state-of-the-art methods: HFT (McAuley & Leskovec, 2013), HPF (Gopalan et al., 2015) and CCPF (Basbug & Engelhardt, 2017). Since CCPF has many choices of combination between sparsity and response models, we chose one example with better performance. Figure 4 shows the evolution of the test RMSE of deepLTRS (with texts and $D = K = 50$) and its competitors. Although HFT accounts for the text reviews, it does not respond well to our simulated scenario and turns out to be very sensitive to the data sparsity. HPF also appears to be quite sensitive to the data sparsity and it always performs worse than CCPF and deepLTRS. CCPF and deepLTRS present less sensitivity to the sparsity as the changes in the RMSE are small along with the sparsity level and deepLTRS outperforms CCPF here. Let us recall that the simulation setup does not follow the deepLTRS generative model and therefore does not favour any method here.

4.2 Amazon Fine Food data

We now consider applying deepLTRS to a real-world data set consisting of fine food reviews from Amazon. The data sets can be downloaded freely on the dedicated website[1]. Here, deepLTRS is compared with previously mentioned models: HFT, HPF, CCPF and TransNet.

Data and pre-processing. This data set spans over a period of more than 10 years, including all $568,464$ reviews up to October 2012. All records include product and user information, ratings, time of the review and a plain-text review. In the data pre-processing step, we only considered users with more than 20 reviews and products reviewed by more than 50 users to obtain more meaningful information. Retained data were processed by removing all punctuations and numbers. The final data set has $M = 1643$ users, $P = 1733$ products, a vocabulary with $V = 5733$ unique words and 32811 text reviews in total. The data sparsity is here of 0.989%.

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 adopt an Adam optimizer, with a learning rate $lr = 2e^{-3}$. The RMSE is calculated on both the validation and test set. Detailed architecture of our network is described in the supplementary

[1]Amazon Fine Food reviews `https://snap.stanford.edu/data/web-FineFoods.html`

Table 3: Test RMSE on Amazon Fine Food data.

Model	Run 1	Run 2	Run 3	Runt 4	Run 5	Average
HFT	1.4241	1.5327	1.4737	1.4228	1.3850	1.4477 (±0.0465)
HPF	2.9486	2.9682	2.9311	2.9428	2.9734	2.9528 (±0.0144)
CCPF-PMF	1.2695	1.2964	1.3035	1.2923	1.2950	1.2913 (±0.0105)
TransNet	1.3772	1.3806	1.3781	1.3776	1.3782	1.3783 (±0.0012)
deepLTRS	1.1364	1.2595	1.2445	1.1710	1.2475	**1.2118** (±0.0489)

material (*cf* Appendix D). Reported test RMSE is obtained when the RMSE on the validation set was the lowest.

Rating prediction. Table 3 presents the test RMSE for deepLTRS (with $D = K = 50$) and its competitors on the predicted ratings for the Amazon fine food data. Once again, deepLTRS has better performance than other models, with an average test RMSE equal to 1.2118. Since HFT is restricted by the numbers of latent factors and topics should be the same value, we set $D = K = 50$, even a larger K value can enable us to obtain better results. Results presented demonstrate the advantage of our model which jointly takes into account both users and products preferences.

Interpretability. In order to deeper understand the meaning of latent representations, we provide in Figure 5 the visualisation of users latent positions on two specific latent variables (var. 3 and 11) that can be easily interpreted according to average ratings (left) and numbers of reviews (right) the users give to products. Indeed, it clearly appears that var. 11 well captures the rating scales of Amazon users whereas var. 3 seems to encode the users activities (number of reviews). A similar analysis were done on product latent representations (*cf* Figure 6 in the supplementary material).

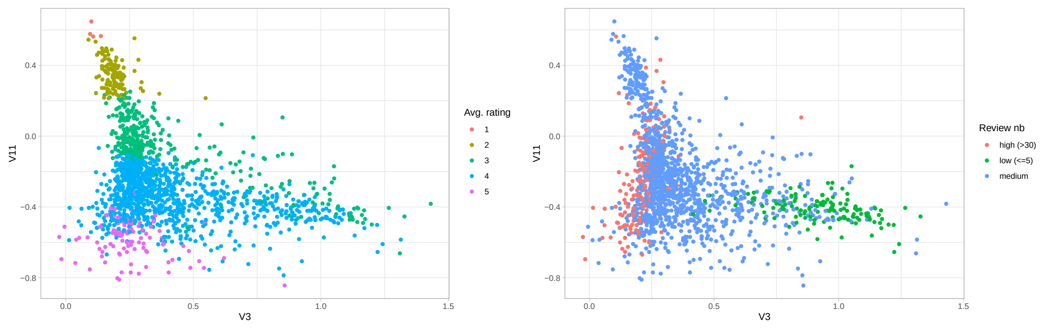

Figure 5: Latent representation of users on var. 3 and 11, according to average ratings (left) and numbers of reviews (right) given to products.

5 Conclusion

We introduced the deepLTRS model for recommendation using both the ordinal and text data available. Our approach adopted a variational auto-encoder architecture as the generative deep latent variable model for both an ordinal matrix encoding the user/product ratings, and a document-term matrix encoding the reviews. DeepLTRS presented the advantage in jointly learning representations of users and products through the alternated mini-batching optimization. Numerical experiments on simulated and real-world data sets showed that our model outperforms competitors in the context of high data sparsity. The further ability of deepLTRS to predict the top words used by reviewers to comment products will be inspected in future works.

References

Mehmet Basbug and Barbara Engelhardt. Hierarchical compound poisson factorization. In *International Conference on Machine Learning*, pp. 1795–1803, 2016.

Mehmet E Basbug and Barbara E Engelhardt. Coupled compound poisson factorization. *arXiv preprint arXiv:1701.02058*, 2017.

Yoshua Bengio and Yann LeCun. Scaling learning algorithms towards AI. In *Large Scale Kernel Machines*. MIT Press, 2007.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

Rose Catherine and William Cohen. Transnets: Learning to transform for recommendation. In *Proceedings of the eleventh ACM conference on recommender systems*, pp. 288–296, 2017.

Adji B Dieng, Francisco JR Ruiz, and David M Blei. Topic modeling in embedding spaces. *arXiv preprint arXiv:1907.04907*, 2019.

Zvi Gilula, Robert McCulloch, Ya?acov Ritov, and Oleg Urminsky. A study into mechanisms of attitudinal scale conversion: A stochastic ordering approach. 2018.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016.

Prem Gopalan, Jake M Hofman, and David M Blei. Scalable recommendation with hierarchical poisson factorization. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, pp. 326–335, 2015.

Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pp. 165–172, 2013.

Andriy Mnih and Russ R Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pp. 1257–1264, 2008.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan 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*, pp. II–1278. JMLR. org, 2014.

Akash Srivastava and Charles Sutton. Autoencoding variational inference for topic models. *arXiv preprint arXiv:1703.01488*, 2017.

Chong Wang and David M. Blei. Collaborative topic modeling for recommending scientific articles. KDD '11, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450308137. doi: 10.1145/2020408.2020480. URL `https://doi.org/10.1145/2020408.2020480`.

Lei Zheng, Vahid Noroozi, and Philip S. Yu. Joint deep modeling of users and items using reviews for recommendation, 2017.

A Rotational invariance in PMF

From Eqs.1-2 it easily follows that

$$Y_{ij} \sim \mathcal{N}(0, D + \eta^2). \tag{15}$$

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

$$Y_{ij} = \overline{R}_i^T \overline{C}_j + \epsilon_{ij},$$

we recover the marginal distribution in Eq. 15.

B Proof of Proposition 1

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

$$\begin{aligned}\nabla_{(\eta^2,\gamma,\beta,\phi)} z_i = \nabla_{(\eta^2,\gamma,\beta,\phi)} \Bigg[&-D_{KL}(q(R_i) \parallel p(R_i)) + \sum_{j=1}^{P} \left(\mathbb{E}_{q(R_i,C_j)}[\log p(Y_{ij}|R_i, C_j, \Theta)] \right). \\ &+ \sum_{j=1}^{P} (\mathbb{E}_{q(R_i,C_j)}[\log p(W^{(i,j)})|R_i, C_j, \Theta)]) \Bigg]. \end{aligned} \tag{16}$$

Then, Eq. 11 leads to

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

and replacing Eq. 16 into the above equation we obtain Eq. 12, recalling that $\nabla_{(\eta^2,\gamma,\beta,\phi)}(\cdot)$ does not include the partial derivative with respect to ι. Thus

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

$\square$

C Additional figures for Amazon data

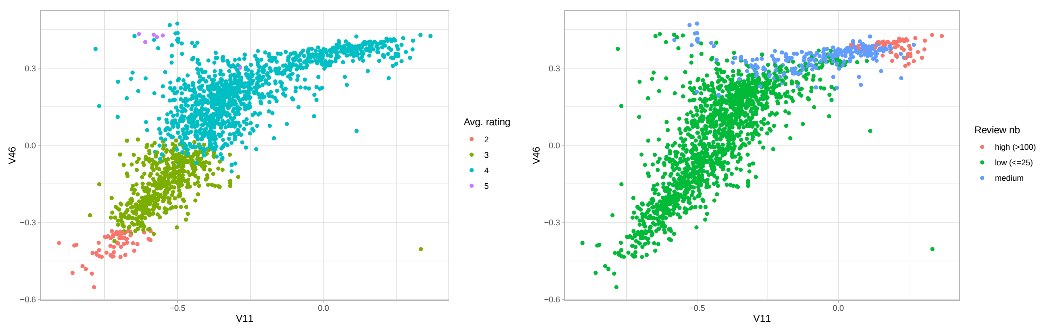

Figure 6: Latent representation of products on var. 11 and 46, according to average rating (left) and number of reviews (right) they receive from users.

D Architecture of deepLTRS

We detailed here the deepLTRS architecture as following.

Firstly, in the user encoder, the first hidden layer has $init\_dim\_R = (P + V)$ neurons, where P is equal to the number of products, and V is the number of words in the text vocabulary; the second hidden layer has $mid\_dim = 50$ neurons. In the product encoder, the first hidden layer has $init\_dim\_C = (M + V)$ neurons, where M is equal to the number of users; the second hidden layer has the same neurons and operations as in the user encoder. $Softplus$ activation function are applied in each layer, followed by a dropout with a ratio of 0.2 and batch normalization.

Secondly, for the decoding phase, the first two layers have $2 \times int\_dim$ and 80 neurons separately, where $int\_dim = init\_dim\_R$ when decoding for users and $int\_dim = init\_dim\_C$ for products. The number of neurons in the third layer depends on the number of topics, here we used $nb\_of\_topics = 50$. Two decoding functions are introduced, one is used to decode the rating matrix, and the other is applied on review matrix. In addition, $Relu$ activation function, a dropout with a ratio of 0.2 and batch normalization are applied. In order to obtain the probability of word occurrence, a $Softmax$ function is used in the end.