Deep Sequential Modeling for Recommendation (DISCUSSION PAPER)

Giuseppe Manco[2], Ettore Ritacco[2], Noveen Sachdeva[1], and Massimo Guarascio[2]

[1] International Institute of Information Technology, Hyderabad, India
noveen.sachdeva@research.iiit.ac.in
[2] ICAR-CNR, Via P. Bucci, 8/9c, Rende, Italy
{name.surname}@icar.cnr.it

Abstract. We propose a model which extends variational autoencoders by exploiting the rich information present in the past preference history. We introduce a recurrent version of the VAE, where instead of passing a subset of the whole history regardless of temporal dependencies, we rather pass the consumption sequence subset through a recurrent neural network. At each time-step of the RNN, the sequence is fed through a series of fully-connected layers, the output of which models the probability distribution of the most likely future preferences. We show that handling temporal information is crucial for improving the accuracy of recommendation.

1 Introduction

Learning of accurate models able to suggest interesting items to the user is an important problem for many scientific and industrial applications. To address this problem,

collaborative filtering approaches to recommendation were extensively investigated by the current literature. Among these, latent variable models [5, 14, 17, 6] gained substantial attention, due to their capabilities in modeling the hidden causal relationships that influence user preferences. Recently, however, new approaches based on neural architectures were proposed, achieving competitive performance with respect to the current state of the art. Also, new paradigms based on the combination of deep learning and latent variable modeling [7, 12] were proven quite successful in domains such as computer vision and speech processing. However, their adoption for modeling user preferences is still unexplored, although recently it is starting to gain attention [9, 8].

The aforementioned approaches rely on the "bag-of-word" assumption: when considering a user and her preferences, the order of such preferences can be neglected and all preferences are exchangeable. This assumption works with general

user trends which reflect a long-term behavior. However, it fails to capture the short-term preferences that are specific of several application scenarios, especially in the context of the Web. Sequential data can express causalities and dependencies that require ad-hoc modeling and algorithms. And in fact, efforts to capture this notion of causality have been made, both in the context of latent variable modeling [11, 1, 3] and deep learning [4, 2, 18, 15].

In this paper, we consider the task of sequence recommendation from the perspective of combining deep learning and latent variable modeling. Inspired by the approach in [9], we assume that at a given timestamp the choice of a given item is influenced by a latent factor that models user trends and preferences. However, the latent factor itself can be influenced by user history and modeled to capture both long-term preferences and short-term behavior.

Our contribution can be summarized as follows: (*i*) we extend the framework to the case of sequential recommendation, where user's preferences exhibit temporal dependencies; (*ii*) we evaluate the proposed framework on standard benchmark datasets, by showing that *(a)* approaches not considering temporal dynamics are not totally adequate to model user preferences, and *(b)* the combination of latent variable and temporal dependency modeling produces a substantial gain, even with regard to other approaches that only focus on temporal dependencies through recurrent relationships.

The paper is organized as follows. Sections 2 proposes the modeling of user preferences in a variational setting, by describing how the framework can be adapted to the case of temporally ordered dependencies. The effectiveness of the proposed modeling is illustrated in section 3, and pointers to future developments are discussed in section 4.

2 Variational Autoencoders for User Preferences

The reader is referred to [13] for more details of the approach illustrated here. We shall use the following shared notation: $u \in U = \{1, \ldots, M\}$ indexes a user and $i \in I = \{1, \ldots, N\}$ indexes an item for which the user can express a preference. We model implicit feedback, thus assuming a preference matrix $\boldsymbol{X} \in \{0, 1\}^{N \times M}$, so that $\boldsymbol{x}_u$ represents the (binary) row with all the item preferences for user u. Given $\boldsymbol{x}_u$, we define $I_u = \{i \in I | x_{u,i} = 1\}$ (with $N_u = |I_u|$). Analogously, $U_i = |\{u \in U | x_{u,i} = 1\}|$ and $M_i = |U_i|$.

We also consider a precedence and temporal relationships within $\boldsymbol{X}$. First of all, the preference matrix induces a natural ordering relationship between items: $i \prec_u j$ has the meaning that $x_{u,i} > x_{u,j}$ in the rating matrix. Also, we assume the existence of timing information $\boldsymbol{T} \in \mathbb{R}_+^{M \times N} \cup \{\emptyset\}$, where the term $t_{u,i}$ represents the time when i was chosen by u (with $t_{u,i} = \emptyset$ if $x_{u,i} = 0$). Then, $i <_u j$ denotes that $t_{u,i} < t_{u,j}$, With an abuse of notation, we also introduce a temporal mark in the elements of $\boldsymbol{x}_u$: the term $\boldsymbol{x}_{u(t)}$ (with $1 \leq t \leq N_u$) represents the t-th item in I_u in the sorting induced by $<_u$, whereas $\boldsymbol{x}_{u(1:t)}$ represents the sequence $\boldsymbol{x}_{u(1)}, \ldots, \boldsymbol{x}_{u(t)}$.

The reference model is the Multinomial variational autoencoder (MVAE) proposed in [9]. Within this framework, for a given user u it is possible to devise $\boldsymbol{x}_u$ according to the generative setting:

$$\begin{aligned} \boldsymbol{z}_u \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_K)\,, \qquad \pi(\boldsymbol{z}_u) \sim \exp\{f_\phi(\boldsymbol{z}_u)\}\,, \\ \boldsymbol{x}_u \sim \mathit{Multi}\left(N_u, \pi(\boldsymbol{z}_u)\right)\,. \end{aligned} \tag{2.1}$$

The underlying "decoder" is modeled by $\log P_\phi(\boldsymbol{x}_u|\boldsymbol{z}_u) = \sum_i x_{u,i} \log \pi_i(\boldsymbol{z}_u)$, thus enabling a complete specification of the overall variational framework. Prediction for new items is accomplished by resorting to the learned functions f_ϕ and Q_λ: given a user history $\boldsymbol{x}$, we compute $\boldsymbol{z} = \boldsymbol{\mu}_\lambda(\boldsymbol{x})$ and then devise the probabilities for the whole item set through $\pi(\boldsymbol{z})$. Unseen items can then be ranked according to their associated probabilities.

Ideally, latent variable modeling should be able to express temporal dynamics and hence causalities and dependencies among preferences in a user's history. In the following we elaborate on this intuition. Within a probabilistic framework, we can model temporal dependencies by conditioning each event to the previous events: given a sequence $\boldsymbol{x}_{(1:T)}$, we have

$$P\left(\boldsymbol{x}_{(1:T)}\right) = \prod_{t=0}^{T-1} P\left(\boldsymbol{x}_{(t+1)}|\boldsymbol{x}_{(1:t)}\right)\,.$$

This specification suggests two key aspects:

- There is a recurrent relationship between $\boldsymbol{x}_{(t+1)}$ and $\boldsymbol{x}_{(1:t)}$ devised by $P\left(\boldsymbol{x}_{(t+1)}|\boldsymbol{x}_{(1:t)}\right)$, upon which the modeling can take advantage, and
- each time-step can be handled separately, and in particular it can be modeled through a conditional VAE.

Let us consider the following (simple) generative model

$$\begin{aligned} \boldsymbol{z}_{u(t)} \sim \mathcal{N}\left(\boldsymbol{0}, \boldsymbol{I}_K\right)\,, \qquad \pi\left(\boldsymbol{z}_{u(t)}\right) \sim \exp\left\{f_\phi\left(\boldsymbol{z}_{u(t)}\right)\right\}\,, \\ \boldsymbol{x}_{u(t)} \sim \mathit{Multi}\left(1, \pi\left(\boldsymbol{z}_{u(t)}\right)\right)\,, \end{aligned} \tag{2.2}$$

which results in the joint likelihood

$$P(\boldsymbol{x}_{u(1:T)}, \boldsymbol{z}_{u(1:T)}) = \prod_t P(\boldsymbol{x}_{u(t)}|\boldsymbol{z}_{u(t)}) P(\boldsymbol{z}_{u(t)})\,.$$

Here, we can approximate the posterior $P(\boldsymbol{z}_{u(1:T)}|\boldsymbol{x}_{u(1:T)})$ with the factorized proposal distribution

$$Q_\lambda(\boldsymbol{z}_{u(1:T)}|\boldsymbol{x}_{(1:T)}) = \prod_t q_\lambda(\boldsymbol{z}_{u(t)}|\boldsymbol{x}_{(1:t-1)})\,,$$

where $q_\lambda(\boldsymbol{z}_{u(t)}|\boldsymbol{x}_{(1:t-1)})$ is a gaussian distribution whose parameters $\boldsymbol{\mu}_\lambda(t)$ and $\boldsymbol{\sigma}_\lambda(t)$ depend upon the current history $\boldsymbol{x}_{u(1:t-1)}$, by means of a recurrent layer $\boldsymbol{h}_t$:

$$\begin{aligned} \boldsymbol{\mu}_\lambda(t), \boldsymbol{\sigma}_\lambda(t) = \varphi_\lambda(\boldsymbol{h}_t) \\ \boldsymbol{h}_t = \mathit{RNN}_\lambda\left(\boldsymbol{h}_{t-1}, \boldsymbol{x}_{u(t-1)}\right)\,. \end{aligned} \tag{2.3}$$

The resulting loss function can be formalized as follow:

$$\mathcal{L}(\phi, \lambda, \boldsymbol{X}) = \sum_{u} \sum_{t=1}^{N_u} \left\{ \frac{1}{2} \sum_{k} \left(\sigma_{\lambda,k}(t) - 1 - \log \sigma_{\lambda,k}(t) + \mu_{\lambda,k}(t)^2 \right) \right.$$
$$\left. - \; E_{\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})} \left[\log P_\phi \left(\boldsymbol{x}_{u(t)} | \boldsymbol{z}_\lambda(\boldsymbol{\epsilon}, t) \right) \right] \right\}.$$

The proposal distribution introduces a dependency of the latent variable from a recurrent layer, which allows to recover the information from the previous history. We call this model SVAE.

Notably, the prediction step can be easily accomplished in a similar way as for MVAE: given a user history $\boldsymbol{x}_{u(1:t-1)}$, we can resort to eq. 2.3 and set $\boldsymbol{z} = \boldsymbol{\mu}_\lambda(t)$, upon which we can devise the probability for the $\boldsymbol{x}_{u(t)}$ by means of $\pi(\boldsymbol{z})$.

The generative model of eq. 2.2 only focuses on the next item in the sequence. The base model however is flexible enough to extend its focus on the next k items, regardless of the time: $\boldsymbol{x}_{u(t:t+k-1)} \sim \mathit{Multi}\left(k, \pi\left(\boldsymbol{z}_{u(t)}\right)\right)$. Again, the resulting joint likelihood can be modeled in different ways. The simplest way consists in considering $\boldsymbol{x}$ as a time-ordered multi-set,

$$P(\boldsymbol{x}_{u(1:T)}, \boldsymbol{z}_{u(1:T)}) = \prod_{t} P(\boldsymbol{x}_{u(t:t+k-1)} | \boldsymbol{z}_{u(t)}) P(\boldsymbol{z}_{u(t)}) \,. \tag{2.4}$$

Alternatively, we can consider the probability of an item as a mixture relative to all the time-steps where it is considered:

$$P(\boldsymbol{x}_{u(1:T)}, \boldsymbol{z}_{u(1:T)}) = \prod_{t} P(\boldsymbol{z}_{u(t)}) \sum_{t-k+1 \leq j \leq t} P(\boldsymbol{x}_{u(t)} | \boldsymbol{z}_{u(j)}), \tag{2.5}$$

where $P(\boldsymbol{x}_{u(t)} | \boldsymbol{z}_{u(j)})$ is the probability of observing $\boldsymbol{x}_{u(t)}$ according to $\pi(\boldsymbol{z}_{u(j)})$. In both cases, the variational approximation is modeled exactly as shown above, and the only difference lies in the second component of the loss function, which has to be adapted according to the above equations.

There is an interesting analogy between eq. 2.5 and the attention mechanism [16]. In fact, it can be noticed in the equation that the prediction of $\boldsymbol{x}_{u(t)}$ depends on the latent status of the k previous steps in the sequence. In practice, this enables to capture short-term dependencies and to encapsulate them in the same probabilistic framework by weighting the likelihood based on $\boldsymbol{z}_{u(t-k+1)}, \ldots, \boldsymbol{z}_{u(t)}$.

3 Evaluation

We evaluate SVAE on some benchmark datasets, by comparing with various baselines and the current state-of-the-art competitors, in order to assess its capabilities in modeling preference data. Additionally, we provide a sensitivity analysis relative to the configurations/contour conditions upon which SVAE is better suited.

We evaluate our model along with the competitors on two popular publicly available datasets, namely *Movielens-1M* and *Netflix*. Movielens-1M is a time series dataset containing user-item ratings pairs along with the corresponding timestamp. Since we work on implicit feedback, we binarize the data, by considering only the user-item pairs where the rating provided by the user was strictly greater than 3 on a range 1:5. Netflix has the same data format as Movielens-1M and the same technique is used to binarize the ratings. We use a subset of the full dataset that matches the user-distribution with the full dataset. The subset is built by stratifying users according to their history length, and then sampling a subset of size inversely proportional to the size of the strata.

Since we are considering implicit preferences, the evaluation is done on top-n recommendation, and it relies on the following metrics.

Normalized Discounted Cumulative Gain. Also abbreviated as *NDCG@n*, the metric gives more weight to the relevance of items on top of the recommender list and is defined as

$$NDCG@n = \frac{DCG@n}{IDCG@n}$$

where $DCG@n = \sum_{i=1}^{n} \frac{r_i}{log_2(i+1)}$ and $IDCG@n = \sum_{i=1}^{|R|} \frac{1}{log_2(i+1)}$. Here, r_i is the relevance (either 1 or 0 within the feedback scenario) of the i-th recommended items in the recommendation list, and R is the set of relevant items.

Precision. By defining $Hits = \sum_i r_i$ as the number of items occurring in the recommendation list that were actually preferred by the user, we have

$$Precision@n = \frac{Hits@n}{n}$$

Recall. defined as the percentage of items actually preferred by the user that were present in the recommendation list:

$$Recall@n = \frac{Hits@n}{|R|}$$

In our experiments, we use the above metrics with two values of n, respectively 10 and 100. The evaluation protocol works as follows. We partition users into training, validation and test set. The model is trained using the full histories of the users in the training set. During evaluation, for each user in the validation/test set we split the **time-sorted** user history into two parts, *fold-in* and *fold-out* splits. The *fold-in* split is used as a basis to learn the necessary representations and provide a recommendation list which is then evaluated with the *fold-out* split of the user history using the metrics defined above. We believe that this strategy is more robust when compared to other methodologies wherein the same user can be in both the training as well as testing sets.

It is worth noticing that Liang et. al [9] follow a similar strategy but

they do not consider any sorting for user histories. That is, for the validation/test users, the *fold-in* set doesn't precede the *fold-out* with respect to time.

We compare our model with current state-of-the-art models including recently published neural architectures and we now present a brief summary about our competitors to provide a better understanding of these models.

- **POP** is a simple baseline where the most popular items are recommended.
- **BPR** is a state of the art model based on Matrix Factorization, which ranks items differently for each user [10]. There is a subtle issue concerning BPR: by separating users on training/validation/test as discussed above, the latent representation of users in the validation/test is not meaningful. Then, BPR is only capable of providing meaningful predictions for users that were already exploited in the training phase. To solve this, we extended the training set to include the partial history in *fold-in* for each user in the validation/test. The evaluation takes place on their corresponding *fold-out* sets.
- **FPMC** [11] is a model which clubs both Matrix Factorization and Markov Chains together using personalized transition graphs over underlying Markov chains.
- **CASER** [15] is a convolutional model that uses vertical and horizontal convolutional layers to embed a sequence of recent items thereby learning sequential patterns for next-item recommendation. The authors have shown that this model outperforms other approaches based on recurrent neural network modeling, such as GRU4Rec. We use the implementation provided by the authors and tune the network by keeping the number of horizontal filters to be 16, and vertical filters to be 4.
- **MVAE** from which the SVAE model draws heavily. We use the implementation provided by the authors, with the default hyperparameter settings.

The experiments consider the basic SVAE model and the extensions to the next-k prediction, as discussed in the previous section. The model is trained end-to-end on the full histories of the training users. Model hyperparameters are set using the evaluation metrics obtained on validation users. The SVAE architecture includes an embedding layer of size 256, a recurrent layer realized as a GRU with 200 cells, and two encoding layers (of size 150 and 64) and finally two decoding layers (again, of size 64 and 150). We set the number K of latent factors for the variational autoencoder to be 64. Adam was used to optimize the loss function coupled with a weight decay of 0.01. As for RVAE, the architecture includes user/item embedding layers (of size 128), two encoding layers (size 100 and 64), and a final layer that produces the score $f_\phi(\boldsymbol{z}_{u,i})$. Both SVAE and RVAE were implemented in PyTorch and trained on a single GTX 1080Ti GPU. The source code is available on GitHub.

In a first set of experiments, we compare SVAE with all competitors described above. Table 1 shows the results of the comparison. SVAE outperforms the competitors on both datasets with a significant gain on all the metrics. It is important here to highlight how the temporal *fold-in*/*fold-out* split is crucial for a fair evaluation of the predictive capabilities: MVAE was evaluated both on temporal and random split, exhibiting totally different performances. Our interpretation is that, with random splits, the prediction for an item is easier if the encoding phase is aware of forthcoming items in the same user history. This severely affects the performance and overrates the predictive capabilities

`https://github.com/noveens/svae_cf`.

of a model: the accuracy of MVAE drops substantially when a temporal split is considered.

By contrast, SVAE is trained to capture the actual temporal dependencies and ultimately results in better predictive accuracy. This is also shown in fig. 1, where we show that SVAE outperforms the competitors irrespective of the size of *fold-in*. The only exception is with very short sequences (less than 10 items), where MVAE gets better results with respect to the sequential models. It is also worth noticing how the performance of both sequential models tend to degrade with increasing sequences, but SVAE maintains its advantage over CASER.

We discussed in the previous section how the basic SVAE framework can be extended to focus on predicting the next k items, rather then just the next item. We analyse this capability in fig. 2, where the accuracy for different values of k is considered according to the modeling in 2.4. On Movielens, the best value is achieved for $k = 4$, and acceptable values range within the interval $1 - 10$.

Model	Movielens						Netflix					
	NDCG		Recall		Precision		NDCG		Recall		Precision	
	@10	@100	@10	@100	@10	@100	@10	@100	@10	@100	@10	@100
POP	4.24	11.30	3.41	22.15	3.98	3.14	3.95	7.61	1.29	12.02	4.26	3.95
BPR*	8.51	16.66	5.20	29.38	7.27	4.51	8.94	12.68	2.63	19.45	8.45	5.89
FPMC	5.65	12.06	3.46	23.10	4.77	3.46	11.01	13.55	3.49	20.82	10.12	6.04
CASER	16.32	27.55	11.35	46.01	13.15	6.45	16.46	19.79	6.33	28.33	14.38	7.54
MVAE	11.69	23.01	9.12	41.43	9.02	5.39	16.15	22.19	7.47	32.72	13.94	8.31
SVAE	**17.81**	**29.93**	**12.48**	**49.09**	**14.40**	**6.93**	**24.64**	**26.77**	**8.93**	**35.58**	**21.93**	**10.55**
MVAE*	27.82	39.79	17.46	59.70	23.01	9.59	35.41	37.70	13.50	47.22	31.39	15.20

Table 1. Results of the evaluation (in percentage). MVAE* considers random splits that disregards the temporal order of user history. BPR* relies on including the *fold-in* subsequences in the training phase.

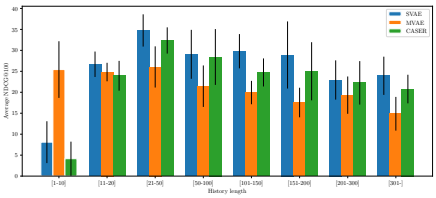

Fig. 1. Average NDCG@100 for MVAE, CASER & SVAE across various history lengths.

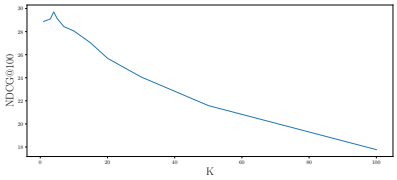

Fig. 2. NDCG@100 values for SVAE on Movielens, across different sizes for the number of items forward in time to predict on.

4 Conclusions and future work

Combining the representation power of latent spaces, provided by variational autoencoders, with the sequence modeling capabilities of recurrent neural networks is an effective strategy to sequence recommendation. To prove this, we

devised SVAE, a simple yet robust mathematical framework capable of modeling temporal dynamics upon different perspectives, within the fundamentals of variational autoencoders. The experimental evaluation highlights the capability of SVAE to consistently outperform state-of-the-art models.

The framework proposed here is worth further extensions that we plan to accomplish in a future work. In particular, from an architectural point of view, the attention mechanism, outlined in section 2.5, requires a better understanding and a more detailed analysis of its possible impact in view of the recent developments in the literature.

However, different architectures (e.g. based on convolution or translation invariance are worth being investigated within a probabilistic variational setting.

References

1. Barbieri, N., Manco, G., Ritacco, E., Carnuccio, M., Bevacqua, A.: Probabilistic topic models for sequence data. Machine Learning pp. 1–25 (2013)
2. Devooght, R., Bersini, H.: Long and short-term recommendations with recurrent neural networks. In: UMAP '17. pp. 13–21 (2017)
3. He, R., Kang, W., McAuley, J.: Translation-based recommendation. In: RecSys '17. pp. 161–169 (2017)
4. Hidasi, B., Karatzoglou, A., Baltrunas, L., Tikk, D.: Session-based recommendations with recurrent neural networks. In: ICLR '16 (2016)
5. Hofmann, T.: Latent semantic models for collaborative filtering. ACM Transactions on Information Systems **22**(1), 89–115 (2004)
6. Kabbur, S., Ning, X., Karypis, G.: Fism: Factored item similarity models for top-n recommender systems. In: KDD '13. pp. 659–667 (2013)
7. Kingma, D., Welling, M.: Auto-encoding variational bayes. In: ICLR '14 (2014)
8. Li, X., She, J.: Collaborative variational autoencoder for recommender systems. In: KK '17. pp. 305–314 (2017)
9. Liang, D., Krishnan, R.G., Hoffman, M., Jebara, T.: Variational autoencoders for collaborative filtering. In: WWW '18. pp. 689–698 (2018)
10. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: Bpr: Bayesian personalized ranking from implicit feedback. In: UAI '09. pp. 452–461 (2009)
11. Rendle, S., Freudenthaler, C., Schmidt-Thieme, L.: Factorizing personalized markov chains for next-basket recommendation. In: WWW '10. pp. 811–820 (2010)
12. Rezende, D.J., Mohamed, S., Wierstra, D.: Stochastic backpropagation and approximate inference in deep generative models. In: ICML '14. pp. 1278–1286 (2014)
13. Sachdeva, N., Manco, G., Ritacco, E., Pudi, V.: Sequential variational autoencoders for collaborative filtering. pp. 600–608. WSDM '19 (2019)
14. Salakhutdinov, R., Mnih, A.: Probabilistic matrix factorization. In: NIPS '08. pp. 1257–1264 (2008)
15. Tang, J., Wang, K.: Personalized top-n sequential recommendation via convolutional sequence embedding. In: WSDM '18. pp. 565–573 (2018)
16. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems 30, pp. 5998–6008 (2017)
17. Wang, C., Blei, D.: Collaborative topic modeling for recommending scientific articles. In: KDD '11. pp. 448–456 (2011)
18. Wu, C., Ahmed, A., Beutel, A., Smola, A.and Jing, H.: Recurrent recommender networks. In: WSDM '17. pp. 495–503 (2017)