Recurrent Latent Variable Networks for Session-Based Recommendation

Sotirios P. Chatzis
Department of Electrical Eng., Computer Eng., and Informatics
Cyprus University of Technology
Limassol, Cyprus
sotirios.chatzis@cut.ac.cy

Panayiotis Christodoulou
Department of Electrical Eng., Computer Eng., and Informatics
Cyprus University of Technology
Limassol, Cyprus
paa.christodoulou@edu.cut.ac.cy

Andreas S. Andreou
Department of Electrical Eng., Computer Eng., and Informatics
Cyprus University of Technology
Limassol, Cyprus
andreas.andreou@cut.ac.cy

ABSTRACT

In this work, we attempt to ameliorate the impact of data sparsity in the context of session-based recommendation. Specifically, we seek to devise a machine learning mechanism capable of extracting subtle and complex underlying temporal dynamics in the observed session data, so as to inform the recommendation algorithm. To this end, we improve upon systems that utilize deep learning techniques with recurrently connected units; we do so by adopting concepts from the field of Bayesian statistics, namely variational inference. Our proposed approach consists in treating the network recurrent units as stochastic latent variables with a prior distribution imposed over them. On this basis, we proceed to infer corresponding posteriors; these can be used for prediction and recommendation generation, in a way that accounts for the uncertainty in the available sparse training data. To allow for our approach to easily scale to large real-world datasets, we perform inference under an approximate amortized variational inference (AVI) setup, whereby the learned posteriors are parameterized via (conventional) neural networks. We perform an extensive experimental evaluation of our approach using challenging benchmark datasets, and illustrate its superiority over existing state-of-the-art techniques.

CCS CONCEPTS

• **Computing methodologies → Neural networks**; **Latent variable models**; **Learning latent representations**;

KEYWORDS

Session-based recommendation, latent variable model, recurrent neural network, amortized variational inference, data sparsity.

ACM Reference Format:
Sotirios P. Chatzis, Panayiotis Christodoulou, and Andreas S. Andreou. 2017. Recurrent Latent Variable Networks for Session-Based Recommendation. In *Proceedings of DLRS 2017, Como, Italy, August 27, 2017,* 8 pages.
https://doi.org/10.1145/3125486.3125493

1 INTRODUCTION

Recommender Systems (RS) constitute a key part of modern e-commerce websites [29]; their aim is to enhance the user experience, by providing personalized product recommendations. Recent study on RS has mainly focused on matrix factorization (MF) methods and neighborhood search-type models. Such approaches work well in practice when a rich user profile can be built from the available data. Unfortunately, though, rich user profiles seldom are readily available to real-world systems.

Session-based recommendation is a characteristic challenge that cannot be properly addressed by conventional methodologies employed in the context of RS. Specifically, under a session-based setup, recommendation is based only on the actions of a user during a specific browsing session [29]. Indeed, this type of recommendation generation approach is based on tracking user actions during an active session. Based on the captured and inferred session-based user behavioral patterns, it tries to predict the following user actions during that session, and proactively recommend items/actions to them.

From this description, it becomes apparent that session-based recommendation engines attempt to generate effective recommendations with the availability of user-specific data being extremely limited. Consequently, under this setting, conventional algorithmic approaches towards RS are confronted with hard challenges that stem from the unavailability of rich user profiles (data sparsity) [19]. Hence, in order to obtain effective session-based RS, it is imperative that novel methodological approaches be devised. Such methods must be capable of more effectively inferring and leveraging subtle session patterns, with the ultimate goal of enriching the available user profiles so as to properly address the challenges associated with data sparsity.

Indeed, user session data constitute action sequences potentially entailing rich, complex, and subtle temporal dynamics. Thus, enabling effective extraction of these underlying dynamics, and utilizing them in the context of a preference inference mechanism, may result in novel session-based RS with considerably improved recommendation quality performance compared to the alternatives. Markov chain models (e.g., [30]) constitute the most typical type of machine learning methods used to achieve this goal. However, recent breakthroughs in the field of Deep Learning (DL) [20] have lately come into close scrutiny, as a potential alternative means of addressing these challenges. Specifically, the introduction of novel treatments of Recurrent Neural Networks (RNNs) [4], with compelling performance in as challenging and diverse tasks as image recognition, natural language understanding, and video captioning, has motivated their application to session-based RS. Contrary to simple deep network formulations that comprise only feedforward connections, RNNs also entail recurrent connections that allow for them to construct an internal representation of their observations history [25]. This representation, which is encoded in the form of a high-dimensional vector of hidden unit activations, can then be utilized to address challenging learning tasks dealing with sequential data.

In this context, the work recently presented in [13] constitutes the most characteristic development. The RNNs employed therein are presented with data regarding the items a user selects and clicks on during a given session. On this basis, recommendation relies on the history of previous actions (clicks on items) during that session, and the inferred behavioral patterns. As shown therein, this method yields state-of-the-art session-based recommendation performance in several challenging benchmark problems.

Motivated from these advances, in this paper we seek derivation of a solid inferential framework that allows for increasing the capability of RNN-driven session-based RS to ameliorate the negative effects of data sparsity. To this end, we draw inspiration from recent RS developments which rely on the utilization of Bayesian inference techniques (e.g., [5, 6, 12, 22, 27]). Bayesian inference in the context of RS can be performed by considering that the postulated model variables pertaining to the system users and items are stochastic latent variables with some prior distribution imposed over them. This inferential framework allows for the developed recommendation engine to account for the uncertainty in the available (sparse) training data. Thus, it is expected to allow for much improved predictive performance outcomes compared to the alternatives.

Under this rationale, our proposed approach is founded upon the fundamental assumption that the hidden units of the postulated RNNs constitute latent variables of stochastic nature, imposed some appropriate prior distribution. On this basis, we proceed to infer their corresponding posteriors, using the available training data. Specifically, to allow for our model to scale to real-world datasets, comprising millions of examples, we perform inference by resorting to the amortized variational inference (AVI) paradigm [17, 18]. This is an approximate inference approach, which consists in: (i) parameterizing the inferred posterior distributions by means of conventional neural networks (inference networks); and (ii) casting the inference problem as an optimization problem, by making use of ideas from variational calculus.

We evaluate the efficacy of the so-obtained approach, dubbed Recurrent Latent Variable Network for Session-Based Recommendation (ReLaVaR), considering a challenging publicly available benchmark. We compare the obtained predictive performance of ReLaVaR with state-of-the-art techniques; we show that our approach completely outperforms the competition, without presenting any limitations in terms of computational efficiency and scalability.

The remainder of this paper is organized as follows: In the following Section, we provide a brief overview of the related work. In Section 3, we introduce our approach; specifically, we elaborate on its motivation, formally define our proposed model, and derive its training and prediction generation formulae. In Section 4, we perform the experimental evaluation of our approach, and illustrate its merits over the current state-of-the-art. Finally, in the concluding Section of this paper, we summarize our contribution and discuss our results.

2 RELATED WORK

The continuous explosion in the availability of content through the Internet renders design of RS a significant challenge for both the academic and industrial research communities. Ratings-based collaborative filtering (CF) systems have served as an effective approach to address this challenge [7]. The intuitive idea behind their design consists in considering that the preferences of a user can be inferred by exploiting past ratings of that user as well as users with related behavioral patterns. This thriving subfield of machine learning started becoming popular in the late 1990s with the development of online services such as Amazon, Yahoo! Music, MovieLens, Netflix, and CiteULike.

Existing CF methods can be classified intro three main categories: memory-based methods, model-based methods, and hybrid methods which attempt to combine ideas from the former two paradigms. Memory-based systems generate predictions under a neighborhood-driven rationale: Item rating prediction for a target user comprises determination of other users with similar ratings (target user neighbors), and computation of a weighted average of the ratings of each item provided by the target user neighbors [1, 15]. A drawback of this paradigm is that, given the high sparsity of the ratings matrix, the neighborhood of a target user may contain only few, if any, ratings for a given item. Moreover, such approaches require keeping the whole ratings matrix in memory to perform prediction in real-time. This might be computationally prohibitive when dealing with large real-world systems; thus, scalability is limited.

Model-based CF methods attempt to ameliorate these issues by using the available ratings data to train a machine learning model which expresses the rating decision function of the users. Given the trained model, prediction generation becomes extremely efficient, thus affording scalable real-time operation. MF-based methods are perhaps the most popular class of model-based CF approaches [9, 21, 26, 27]. These methods assume that the registered users and items are related to sets of variables that lie in some low-dimensional latent space; prediction is performed based on these latent features inferred for each user and item.

Recently, several authors have considered introducing elaborate statistical assumptions into MF, that allow for performing full Bayesian inference (e.g., [5, 6, 12, 22, 27]). Under this approach, it is considered that the user and item variables constitute stochastic latent variables, over which an appropriate prior distribution is imposed, and a corresponding posterior is inferred from the data. Broad empirical evidence has shown that, under such a Bayesian inference-driven setup, real-world RS can yield a noticeable predictive accuracy improvement without comprises in computational scalability. Indeed, this outcome is well-expected from a theoretical point of view; this is due to the fact that a Bayesian inference treatment allows for better accounting for the uncertainty in the (training) data, which is prevalent in RS due to the sparsity of the available ratings matrices.

On the other hand, in the last years the field of machine learning has witnessed a new wave of innovation, due to the DL breakthrough [20]. Unsurprisingly, the significant advances accomplished in the context of DL have had a noteworthy impact on the ongoing research on RS. Indeed, several researchers working on model-based CF methods have recently proposed novel CF algorithms that employ DL-based models as an alternative to conventional MF-driven approaches.

In this vein, the work of [28] constitutes one of the earliest ones that adopted ideas from the field of DL to effect the CF task. Specifically, they employed Restricted Boltzmann Machines (RBMs) to learn the user and item latent vectors, and showed that their approach outperforms various popular alternatives in the Netflix challenge dataset. More recently, the method in [32] presented a hierarchical Bayesian model called collaborative deep learning (CDL) for RS. This approach attempts to resolve the cold-start problem by augmenting the MF algorithm with appropriate side information related to item content. This side information is obtained, in turn, from a DL model; this learns to extract useful, high-level representations from the raw item content, so as to inform the MF algorithm.

Despite this extensive research effort devoted to RS, session-based recommendation is a field that remained unappreciated for quite long, and has only recently attracted significant attention from the research community. Indeed, most of news and media sites, as well as many e-commerce sites (especially of small retailers) track the users that visit their sites only for short periods of time. Besides, the use of cookies or browser fingerprinting does not allow for obtaining reliable user data over long periods, spanning multiple sessions. Finally, it is very often the case that the behavior of users exhibits session-based traits.

These facts bring to the fore the need of developing effective session-based RS, that can satisfy the following desiderata: (i) each session of the same user must be treated independently of their previous ones; (ii) the used algorithms must be capable of extracting subtle temporal behavioral patterns from the available user profiles, e.g. item-to-item similarity, co-occurrence, and transition probabilities; and (iii) this inferential procedure must be effectively carried out over long temporal horizons, as opposed to unrealistic low-order (e.g., one-step) temporal dependence models, that take only the last click or selection of the user into account (and ignore the information of past clicks in the same session).

To address these issues, [33] introduced a novel framework based on traditional RNNs, and evaluated it using the click-through logs of a large scale commercial engine; their results showed significant improvements on the click-prediction accuracy compared to sequence-independent approaches. In a similar vein, [13] presented an RNN-type machine learning model capable of learning subtle temporal patterns in user session data obtained from large e-commerce websites. Specifically, to allow for effectively extracting high-order temporal dynamics, they utilized Gated Recurrent Unit (GRU) networks [8]. Such networks entail a more elaborate model of an RNN unit, that aims at dealing with the vanishing/exploding gradient problem; this is a problem that plagues training of conventional RNNs, often rendering it completely infeasible [14]. Their method was shown to outperform state-of-the-art alternatives in two large-scale tasks, including the challenging RecSys Challenge 2015 benchmark [3]. Finally, [31] proposed two extensions of the breakthrough work of [13], namely: (i) data augmentation via sequence preprocessing; and (ii) a simple model pre-training technique, to account for temporal shifts in the data distribution. As they showed, their proposed extensions yield an improvement over the method in [13] by more than 10%.

3 PROPOSED APPROACH

As we have already discussed, the main contribution of our work consists in devising a machine learning model that is capable of extracting subtle and informative temporal dynamics from sparse user session data, with a special focus on accounting for the entailed uncertainty; we use this information to drive the developed recommendation algorithm. We posit that such a capacity will allow for a significant increase in the eventually obtained predictive performance.

Under this rationale, ReLaVaR frames the session-based recommendation problem as a sequence-based prediction problem. Specifically, let us denote as $\{\boldsymbol{x}_i\}_{i=1}^n$ a user session; here, $\boldsymbol{x}_i$ is the ith clicked item, which constitutes a selection among m alternatives, and is encoded in the form of an 1-hot representation. Then, we formulate session-based recommendation as the problem of predicting the score vector $\boldsymbol{y}_{i+1} = [y_{i+1,j}]_{j=1}^m$ of the available items with respect to the following user action, where $y_{i+1,j} \in \mathbb{R}$ is the predicted score of the jth item. Typically, we are interested in recommending more than one items for the considered user to choose from; hence, at each time point we select the top-k items (as ranked via $\boldsymbol{y}$) to recommend to the user. Thus, the goal of this work is to devise a machine learning model capable of more accurately predicting the vectors $\boldsymbol{y}_{i+1}$, given the observed subsequences $\{\boldsymbol{x}_s\}_{s=1}^i$, $\forall i$.

3.1 Methodological Background

To achieve our goals, we draw inspiration from state-of-the-art, RNN-based approaches, such as [13]. Thus, we begin by postulating an RNN structure comprising GRU units. At each time point, i, the postulated network is presented with the current user action (selected item), $\boldsymbol{x}_i$, and is expected to generate a prediction for the score vector $\boldsymbol{y}_{i+1}$ pertaining to the $(i+1)$th user selection. Formally, the recurrent units activation vector, $\boldsymbol{h}$, of a postulated GRU-based network are updated at time i according to the following expression:

$$\boldsymbol{h}_i = (1 - \boldsymbol{z}_i) \cdot \boldsymbol{h}_{i-1} + \boldsymbol{z}_i \cdot \hat{\boldsymbol{h}}_i \tag{1}$$

where $\cdot$ denotes the elementwise product between two vectors, $\boldsymbol{h}_{i-1}$ is the recurrent units activation vector at the previous time point, and z_i is the *update gate* output. This gate essentially learns to control when and by how much to update the hidden state of the recurrent units; it holds

$$z_i = \tau(\boldsymbol{W}_z \boldsymbol{x}_i + \boldsymbol{U}_z \boldsymbol{h}_{i-1}) \tag{2}$$

where $\tau()$ is the logistic sigmoid function, and the $\boldsymbol{W}_z$ and $\boldsymbol{U}_z$ are trainable network parameters. On the other hand, in Eq. (1), $\hat{\boldsymbol{h}}_i$ is the *candidate activations vector* of the GRU units at time i; its expression is a standard recurrent unit update expression with trainable parameters $\boldsymbol{W}$ and $\boldsymbol{U}$, yielding

$$\hat{\boldsymbol{h}}_i = \tanh(\boldsymbol{W}\boldsymbol{x}_i + \boldsymbol{U}(\boldsymbol{r}_i \cdot \boldsymbol{h}_{i-1})) \tag{3}$$

Here, $\boldsymbol{r}_i$ denotes the output of the *reset gate* of the GRU network. This gate essentially learns to decide when the internal memory of the GRU units must be reset, with the ultimate goal of preventing the gradients of the model objective function from exploding to infinity or vanishing to zero during model training; it reads

$$\boldsymbol{r}_i = \tau(\boldsymbol{W}_r \boldsymbol{x}_i + \boldsymbol{U}_r \boldsymbol{h}_{i-1}) \tag{4}$$

with the $\boldsymbol{W}_r$ and $\boldsymbol{U}_r$ being trainable network parameters.

3.2 Model Formulation

ReLaVaR extends upon the model design principles discussed in the previous section. It does so by introducing a novel formulation that renders the developed GRU-based recommendation model amenable to Bayesian inference. To effect our modeling goals, we consider that the component recurrent unit activations are stochastic latent variables. Specifically, we start by imposing a simple prior distribution over them, which reads

$$p(\boldsymbol{h}_i) = \mathcal{N}(\boldsymbol{h}_i | \boldsymbol{0}, \boldsymbol{I}) \tag{5}$$

where $\mathcal{N}(\xi | \boldsymbol{\mu}, \Sigma)$ is a multivariate Gaussian density with mean $\boldsymbol{\mu}$ and covariance matrix Σ, and $\boldsymbol{I}$ is the identity matrix.

On this basis, we seek to devise an efficient means of inferring the corresponding posterior distributions, given the available training data. To this end, we draw inspiration from the AVI paradigm [17]; specifically, we postulate that the sought posteriors, $q(\boldsymbol{h})$, approximately take the form of Gaussians with means and isotropic covariance matrices parameterized via GRU networks. We have:

$$q(\boldsymbol{h}_i; \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{h}_i | \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \sigma^2_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \boldsymbol{I}) \tag{6}$$

In this expression, the mean vectors, $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$, as well as the variance functions, $\sigma^2_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$, are outputs of a postulated GRU network, with parameters set $\boldsymbol{\theta}$. In other words, we have

$$[\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \log \sigma^2_{\boldsymbol{\theta}}(\boldsymbol{x}_i)] = (1 - z_i) \cdot [\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_{i-1}), \log \sigma^2_{\boldsymbol{\theta}}(\boldsymbol{x}_{i-1})] + z_i \cdot \hat{\boldsymbol{h}}_i \tag{7}$$

where

$$z_i = \tau(\boldsymbol{W}_z \boldsymbol{x}_i + \boldsymbol{U}_z [\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_{i-1}), \log \sigma^2_{\boldsymbol{\theta}}(\boldsymbol{x}_{i-1})]) \tag{8}$$

$$\hat{\boldsymbol{h}}_i = \tanh(\boldsymbol{W} \boldsymbol{x}_i + \boldsymbol{U}(\boldsymbol{r}_i \cdot [\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_{i-1}), \log \sigma^2_{\boldsymbol{\theta}}(\boldsymbol{x}_{i-1})])) \tag{9}$$

and

$$\boldsymbol{r}_i = \tau(\boldsymbol{W}_r \boldsymbol{x}_i + \boldsymbol{U}_r [\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_{i-1}), \log \sigma^2_{\boldsymbol{\theta}}(\boldsymbol{x}_{i-1})]) \tag{10}$$

while $[\xi, \zeta]$ denotes the concatenation of vectors ξ and ζ. On this basis, the values of the latent variables (stochastic unit activations) $\boldsymbol{h}_i$ can be computed by drawing (posterior) samples from the inferred posterior density (6).

Finally, let us turn to the output layer of the proposed model. This is presented with the (drawn samples of the) activation vectors $\boldsymbol{h}_i$ of our model, and generates a vector of predicted score values $\boldsymbol{y}_{i+1}$ pertaining to the following user action. On this basis, we need to appropriately impose a suitable distribution over these generated output variables of our model, $\boldsymbol{y}_{i+1}$, conditional on the corresponding latent vectors, $\boldsymbol{h}_i$. Indeed, by reviewing the related literature, one may discover a number of possible alternatives for the conditional likelihood function of a ranking prediction model with the kind of probabilistic formulation that ReLaVaR adopts. Each one of these alternatives essentially gives rise to a different rationale in terms of quantifying the ranking accuracy of the trained model.

In general, item ranking can be pointwise, pairwise or listwise. Pointwise ranking estimates the score of items independently of each other; then, the goal of model training is to ensure that relevant items receive a high score. Pairwise ranking compares the score of pairs of a positive and a negative item; then, model training aims at enforcing the score of the positive item to be higher than that of the negative one, for all the available pairs. Such a construction allows for one to limit score computation for the purposes of model training to a select subset of the available items. On the downside, such a formulation may undermine the eventually obtained accuracy of the recommendation algorithm. On the other hand, pointwise approaches require score computation for the whole set of available items. This is certainly more computationally demanding than pairwise approaches. However, this extra computational complexity does not necessarily translate into reduced scalability to real-world systems. This is especially the case with DL algorithms, which can be easily parallelized at a large scale by using cheap GPU hardware. Finally, listwise ranking uses the scores of all items and compares them to the perfect ordering. This entails item sorting, which can be computationally prohibitive in cases of large-scale systems.

Motivated from this discussion, in this work we resort to the most straightforward conditional likelihood selection for our model, namely a simple Multinoulli distribution; that is

$$p(y_{i+1,j} = 1 | \boldsymbol{h}_i) \propto \tau(\boldsymbol{w}_y^j \cdot \boldsymbol{h}_i) \tag{11}$$

where $\boldsymbol{W}_y = [\boldsymbol{w}_y^j]_{j=1}^m$ are trainable parameters of the output layer of the model.

This selection can be viewed as giving rise to a pointwise ranking criterion, with the associated ranking loss function, L_s, being equal to the negative conditional log-likelihood expression that stems from (11). We have:

$$\begin{aligned} L_s &= -\sum_{i=1}^{n} \log p(\boldsymbol{y}_{i+1} | \boldsymbol{h}_i) \\ &= -\sum_{i,j=1}^{n,m} \big\{ y_{i+1,j} \log p(y_{i+1,j} = 1 | \boldsymbol{h}_i) \\ &\quad + (1 - y_{i+1,j}) \log (1 - p(y_{i+1,j} = 1 | \boldsymbol{h}_i)) \big\} \end{aligned} \tag{12}$$

It is easy to notice that this loss function expression, L_s, essentially constitutes the familiar (binary) Cross-Entropy function, widely used in DL literature.

3.3 Training Algorithm

Let us consider a training dataset $\mathcal{D}$, which comprises a number of click sequences (sessions), pertaining to a multitude of users. Variational inference for the developed ReLaVaR model consists in maximization of a lower-bound to the log-marginal likelihood of the model (evidence lower-bound, ELBO) w.r.t. the model parameters [16]. Based on the previous model definition, the ELBO expression of ReLaVaR yields:

$$\log p(\mathcal{D}) \geq \sum_i \left\{ -\mathrm{KL}\big[q(\boldsymbol{h}_i; \boldsymbol{\theta}) || p(\boldsymbol{h}_i)\big] - \mathbb{E}[L_s] \right\} \tag{13}$$

Here, $\mathrm{KL}\big[q||p\big]$ is the KL divergence between the distribution $q(\cdot)$ and the distribution $p(\cdot)$ [its analytical expression can be found in the Appendix].

Unfortunately, the posterior expectation $\mathbb{E}[L_s]$ cannot be computed analytically; hence, its gradient becomes intractable. This is due to the nonconjugate formulation of ReLaVaR, which stems from its nonlinear assumptions. As a consequence, training the entailed parameter sets $\boldsymbol{\theta}$ is infeasible.

One could argue that this problem might be resolved by approximating this expectation using a set of Γ Monte-Carlo (MC) samples, $\{\boldsymbol{h}_i^{\gamma}\}_{\gamma=1}^{\Gamma}$, drawn from the inferred posteriors. However, it is well-known that such an approximation would result in estimators with unacceptably high variance. AVI resolves these issues by means of a smart reparameterization of the MC samples of a Gaussian posterior density; specifically, we have [17, 18]:

$$\boldsymbol{h}^{\gamma} = \boldsymbol{\mu}_{\boldsymbol{\theta}}(\cdot) + \sigma_{\boldsymbol{\theta}}(\cdot)\,\boldsymbol{\epsilon}^{\gamma} \tag{14}$$

where $\boldsymbol{\epsilon}^{\gamma}$ is white random noise with unitary variance, i.e. $\boldsymbol{\epsilon}^{\gamma} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. By adopting this reparameterization, the MC samples drawn from the posterior density (6) can be now expressed as differentiable functions of the sought parameter sets, $\boldsymbol{\theta}$, and some random noise variable with low (unitary) variance, $\boldsymbol{\epsilon}$. Consequently, the problematic posterior expectation $\mathbb{E}[L_s]$, originally defined over the latent activations, reduces to a much more attractive posterior expectation w.r.t. a low variance (random noise) variable.

Then, by taking the gradient of the so-reparameterized ELBO (13) in the context of any stochastic optimization algorithm, one can yield low variance estimators for the parameter sets, under some mild conditions [17]. To this end, [17] suggest utilization of Adagrad; this constitutes a stochastic gradient algorithm with adaptive step-size [10], and fast and proven convergence to a local optimum. We follow this advice in this work, selecting Adagrad as the stochastic optimizer of choice for training the ReLaVaR model.

3.4 Prediction Generation

Having trained a ReLaVaR model, given some dataset $\mathcal{D}$, recommendation generation in the context of a user session can be performed by computing the predicted ratings $\boldsymbol{y}_{i+1}$, and selecting the top-k of them to recommend to the user. To effect this procedure, we sample the latent variables $\boldsymbol{h}_i$ from the corresponding variational posterior distributions. Indeed, to allow for obtaining reliable estimators, we draw a set comprising Γ samples from the posteriors (6); eventually, this yields a set of scoring function samples, $\{\boldsymbol{y}_{i+1}^{\gamma}\}_{\gamma=1}^{\Gamma}$. Then, recommendation is performed on the basis of the mean of these samples; that is, we employ a standard MC-type rationale.

4 EXPERIMENTAL EVALUATION

To provide strong empirical evidence of the merits of our approach, in this Section we extensively evaluate it in challenging experimental scenarios. To this end, we exploit the benchmark dataset released in the context of the RecSys Challenge 2015 [3]; this comprises click-stream data pertaining to user sessions with an e-commerce website.

Unfortunately, the test set of the aforementioned benchmark does not provide groundtruth information that can be used for recommendation quality evaluation. To resolve this issue, we adopt the solution suggested in [13]; we split the originally available training data into one set comprising 7,966,257 sessions (with a total of 31,637,239 click actions), and another one comprising the remainder 5,324 sessions (with a total of 71, 222 click actions); we use the former for model training and the latter for evaluation purposes. Both sets entail a total of 37,483 items that a user may select to click on. Thus, we are dealing with a very sparse dataset, where the need of inferring more subtle and informative patterns comes to the fore with increased complexity.

To obtain some comparative results, apart from our method we also cite the performance of recently proposed state-of-the-art alternatives in the same benchmark, namely the GRU-based method presented in [13], and the M2 and M4 approach introduced in [31]. In addition, we also run on the same dataset a standard baseline method in the field of matrix factorization, namely BPR-MF [24] and capture its accuracy. Since BPR-MF is designed for processing single item feature vectors, as oppposed to *sequences*, to apply it in the context of session-based recommendation we average the feature vectors of the items occurring thus far in a given session.

Our source codes have been developed in Python, and made use of the Theano library[1] [2]. We run our experiments on an Intel Xeon 2.5GHz Quad-Core server with 64GB RAM and an NVIDIA Tesla K40 GPU accelerator.

4.1 ReLaVaR Model Configuration

In the following, we experiment with a diverse set of selections for the size of the latent variable space (number of recurrent latent variables). We report the best performing model configuration in Section 4.3; we provide deep insights on how model performance varies with this selection in Section 4.4. In all cases, parameter initialization for our model is performed by resorting to the Glorot Normal initialization scheme [11]; dropout with a rate equal to 0.5 is employed for regularization purposes.

To perform model training, Adagrad is carried out by utilizing *session-parallel mini-batches*, following the suggestions in [13]. Let us consider we adopt a mini-batch size equal to β. Then, session-parallel mini-batches can be obtained by using the first event of the first β sessions to form the input data of the first mini-batch (the desired output is the second events of our active sessions); we use the second events to form the second mini-batch, and so forth. When a session ends, we put the next available session in its place. In the occasion of such a switch taking place, we reset the appropriate hidden state of the model, since we assume that the training sessions constitute independent and identically distributed (sequential) data.

To facilitate convergence, Nesterov momentum [23] is additionally applied during training. In all cases, Adagrad's global stepsize is chosen from the set $\{0.005, 0.01, 0.05, 0.1\}$, while momentum strength is chosen from the set $\{0, 0.1, 0.2, 0.3, 0.4\}$, both on the basis of network performance on the training set in the first few training algorithm iterations. Contrary to [31], we do not perform any tedious data augmentation procedure or model pretraining. Thus, our approach is not *directly comparable* to [31], since application of the pre-processing steps proposed therein should be well-expected to also increase the performance of ReLaVaR. However, we do cite the performance results reported in [31] for completeness sake.

[1] http://deeplearning.net/software/theano/

Table 1: Best performance results of the evaluated methods.

Method	Model Size	Recall@20	MRR@20
BPR-MF	-	0.2574	0.0618
GRU w/ BPR Loss [13]	1000	0.6322	0.2467
GRU w/ TOP1 Loss [13]	1000	0.6206	0.2693
M2 [31]	100	0.7129	0.3091
M4 [31]	1000	0.6676	0.2847
ReLaVaR	1500	0.6507	0.3527

4.2 Performance Metrics

To quantitatively assess the performance of our approach, we employ two commonly used evaluation metrics, namely Recall@20 and Mean Reciprocal Rank (MRR)@20. The former metric expresses the frequency at which the desired (groundtruth) item in the test data makes it to the 20 highest ranked items suggested by the evaluated approach. Hence, this metric allows for modeling and assessing certain practical scenarios where there is no highlighting of recommendations; what matters is the desired item being included in a short list of recommendations, rather than the absolute order that these items are presented to the user. On the other hand, MRR@20 describes the average predicted score of the desired items in the test data, with the score values set to zero if the desired item does not make it to the top-20 list of ranked items. Thus, MRR@20 models scenarios where absolute item ordering does matter; for instance, it allows for better algorithm evaluation in cases where the lower ranked items are visible only after scrolling.

4.3 Empirical Performance

We commence the presentation of our experimental results by reporting on the best-performing configuration of our model (i.e., selection of the number of latent variables that maximizes empirical performance on the test set); our findings are summarized in Table 1. In the same Table, we also illustrate how these empirical findings compare to the considered competitors of our method, that is BPR-MF, M2, M4, and the GRU-driven approach of [13]. We report two different performance results for the GRU-based method, which correspond to two different loss functions considered in [13], namely BPR and TOP1; the former selection yields better Recall@20 outcomes for that method, while the latter yields a better MRR@20 value. Moreover, regarding the methods proposed in [31] it can be observed from Table 1 that as the number of hidden units increases the accuracy of the proposed method declines.

As we observe, our approach outperforms all previously reported state-of-the-art results in terms of the MRR@20 metric, while yielding the second-best reported performance in terms of the Recall@20 metric[2]. The number of component latent variables for our method is equal to 1500; this represents a larger network than the one that obtains best performance for the considered alternatives. This fact constitutes further supporting evidence of the capacity of our approach to extract subtler temporal dynamics from the available data without getting prone to overfitting.

[2]We obtained this outcome with the mini-batch size set equal to 50, Adagrad step size set equal to 0.05, and momentum strength set equal to 0.

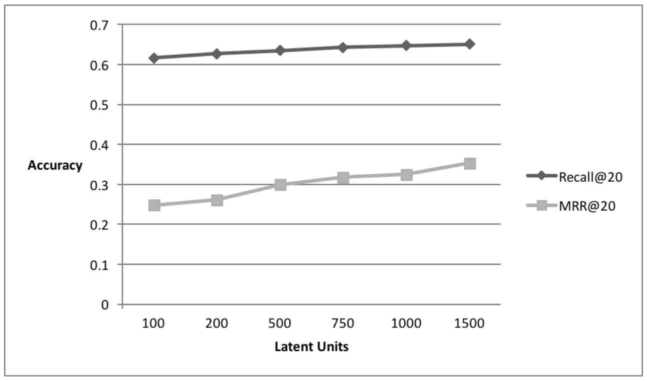

Figure 1: ReLaVaR performance fluctuation with the number of latent variables.

4.4 Further Investigation

4.4.1 Varying the size of the latent variable space. It is well-understood that the number of latent units bears significant impact on the obtained empirical performance. To allow for examining the extent of this effect, in Fig. 1 we show how the considered performance metrics vary when adjusting the number of latent units of a trained ReLaVaR model. These results have been obtained with the training algorithm hyperparameters remaining the same as described in section 4.3.

As it can be seen from Fig. 1, ReLaVaR accuracy, as measured via both the considered metrics, grows as we add more latent units. The increase is more prominent when the network size is small, and tends to be lower for larger networks.

4.4.2 Considering alternative loss functions. Further, it is interesting to examine how the performance of our model compares to the competition in case we adopt a different type of loss function, L_s. To this end, we consider a pairwise ranking loss, namely the TOP1 loss function that was introduced in [13].

The obtained results (for best model configuration) are provided in Table 2. As we observe, appropriate selection of the employed loss function, L_s, is a crucial factor that determines the success of our approach in modeling the considered dataset. Indeed, replacement of the Cross-Entropy loss function with TOP1 yields a notable performance deterioration, especially in terms of the obtained MRR@20 values.

To provide some further insights, in Fig. 2 we illustrate the corresponding results regarding performance fluctuation with the size of the latent variable space. We observe that model size continues to have a significant effect on ReLaVaR predictive performance when using this alternative loss function.

4.5 Computational Complexity

Apart from predictive accuracy, another aspect of machine learning models that is of utmost importance when dealing with real-world applications concerns computational efficiency. This aspect entails examination of both model scalability to large datasets, as well as of the imposed computational overheads when it comes to prediction generation.

Table 2: ReLaVaR model performance for different selections of the employed loss function, L_s (results correspond to best network configuration).

Loss Function	TOP1	Cross-Entropy
# Latent Units	1000	1500
Step Size	0.1	0.05
Momentum Weight	0	0
Recall@20	0.6250	0.6507
MRR@20	0.2727	0.3527

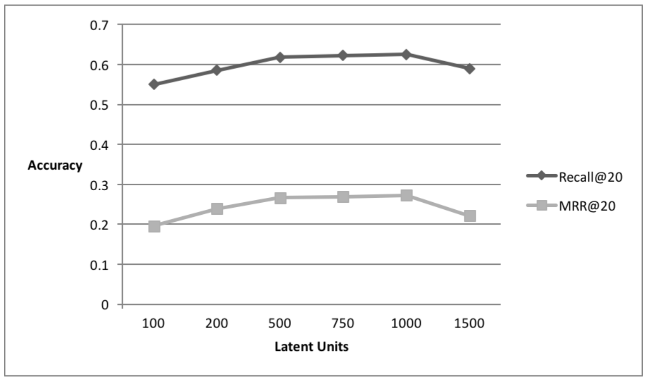

Figure 2: ReLaVaR performance fluctuation with the number of latent variables: Use of the TOP1 loss function.

To investigate these aspects of the proposed approach, in Table 3 we perform a comparison of wall-clock times between our method and the current state-of-the-art. The measurements reported therein concern alternative selections of the employed loss function, L_s; in all cases, they pertain to network sizes yielding the best empirical accuracy of the evaluated methods.

From this exhibition, it becomes apparent that not only our approach yields competitive accuracy, but it does so while remaining competitive in terms of computational costs. Specifically, notice that the slight difference in computational costs between the original formulation of ReLaVaR (i.e., using the Cross-Entropy loss function) and the competition is merely due to the larger size of the trained network. Note also that all the compared approaches allow for real-time prediction generation. Thus, we can soundly argue that our method constitutes an attractive solution for building real-world session-based RS.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we attacked the problem of session-based recommendation. Specifically, our work was motivated from the sequential nature of the addressed predictive setup, and the associated sparsity of the available data. Our expectation was that, by better addressing these issues, one may be able to obtain a noticeable improvement in the quality of the generated recommendations.

To this end, we introduced a way of improving the modeling capacity of RS that utilize deep learning techniques with recurrently connected units. Specifically, we effected this goal by adopting concepts from the field of Bayesian statistics, namely variational inference. The proposed approach, dubbed ReLaVaR, constitutes a hierarchical latent variable model, where the inferred posterior distributions are parameterized via GRU networks. Such a Bayesian inferential setup: (i) retains the prowess of existing GRU-based networks in terms of extracting and analyzing salient temporal patterns in the available user sessions data; and (ii) allows for accounting for the uncertainty in the available (sparse) data when performing prediction and recommendation generation; enabling this capability is well-known to yield a noticeable performance improvement in real-world data modeling scenarios.

We performed an extensive experimental evaluation of our approach using a challenging benchmark dataset. We provided thorough insights into the predictive behavior of our approach under different setups of the employed training algorithm, as well as under different selections of the postulated network configuration. As we showed, our approach is capable of outperforming existing state-of-the-art alternatives in terms of two popular performance metrics. We also illustrated that our proposed approach achieves this accuracy improvement without undermining computational efficiency, both in training time and in prediction generation time.

One research direction that we have not considered in this work concerns the possibility of stacking multiple network layers, one on top of the other, to create a more potent sequential data modeling pipeline. In such a formulation, the input of the bottom GRU layer is the observed data sequence, $\{\boldsymbol{x}_i\}_{i=1}^n$; on the other hand, each one of the subsequent GRU layers is presented with the sequence of activation vectors, $\{\boldsymbol{h}_i\}_{i=1}^n$, of the layer that immediately precedes it in the hierarchy; the model output layer is driven from the recurrent unit activations vector of the topmost GRU layer.

Stacking multiple layers of GRU networks allows for performing inference and analysis of temporal patterns in multiple time-scales. Thus, one might theoretically expect that such an architecture should be capable of yielding improved performance in real-world session-based RS. However, related findings in the recent literature, e.g. [13], have shown this not to be the case; for instance, [13] showed that (stacked) multilayer architectures impose significant extra computational burden without yielding any benefit in terms of predictive accuracy. Indeed, this behavior can be attributed to the short typical length of user sessions, whence temporal pattern analysis on multiple time-scales becomes less relevant.

It was these results that motivated us not to examine stacked multilayer variants of ReLaVaR in the context of this work. However, we do believe that stacked multilayer variants of ReLaVaR might possess favorable performance characteristics in the context of different sorts of systems dealing with session data. Thus, investigation of such possible opportunities remains among our plans for future research pursuits.

APPENDIX

Using (5)-(6), we obtain:

$$\begin{aligned}\mathrm{KL}\big[q(\boldsymbol{h}_i;\boldsymbol{\theta})||p(\boldsymbol{h}_i)\big] =& \frac{1}{2}\sum_{d=1}^{D}\big[\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)^2\big]_d \\ &- \frac{D}{2}\big[1+\log\sigma_{\boldsymbol{\theta}}(\boldsymbol{x}_i)^2-\sigma_{\boldsymbol{\theta}}(\boldsymbol{x}_i)^2\big]\end{aligned} \tag{15}$$

Table 3: Comparison of computational times (in seconds), for various selections of the employed loss function. Results pertain to network configurations yielding best accuracy.

Method	Network Size	Training time (total)	Prediction time per click event (average)
GRU w/ BPR Loss	1000 Units	48692.48	0.683
GRU w/ TOP1 Loss	1000 Units	44716.60	0.627
ReLaVaR w/ TOP1 Loss	1000 Units	42357.84	0.595
ReLaVaR w/ Cross-Entropy Loss	1500 Units	60109.86	0.844

where $[\cdot]_d$ is the dth element of a vector, and D is the dimensionality of the postulated latent space.

ACKNOWLEDGMENT

We gratefully acknowledge the support of NVIDIA Corporation with the donation of one Tesla K40 GPU used for this research.

REFERENCES

[1] G. Adomavicius and A. Tuzhilin. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering* 17, 6 (2005), 734–749.

[2] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. 2012. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop. (2012).

[3] David Ben-Shimon, Alexander Tsikinovsky, Michael Friedmann, Bracha Shapira, Lior Rokach, and Johannes Hoerle. 2015. RecSys Challenge 2015 and the YOOCHOOSE Dataset. In *Proceedings of the 9th ACM Conference on Recommender Systems (RecSys '15)*. ACM, New York, NY, USA, 357–358. https://doi.org/10.1145/2792838.2798723

[4] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu. 2013. Advances in optimizing recurrent networks. In *Proc. ICASSP*. 8624–8628.

[5] Sotirios P. Chatzis. 2012. A Coupled Indian Buffet Process Model for Collaborative Filtering. In *Journal of Machine Learning Research: Workshop and Conference Proceedings*, Vol. 25: ACML 2012. 65–79.

[6] Sotirios P. Chatzis. 2013. Nonparametric Bayesian Multitask Collaborative Filtering. In *Proc. CIKM'13*.

[7] Wen Y. Chen, Jon C. Chu, Junyi Luan, Hongjie Bai, Yi Wang, and Edward Y. Chang. 2009. Collaborative filtering for Orkut communities: discovery of user latent behavior. In *Proc. WWW'09*. 681–690.

[8] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *Proc. Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8)*. ACL, 103–111.

[9] D. DeCoste. 2006. Collaborative prediction using ensembles of maximum margin matrix factorizations. In *Proc. ICML'06*. Pittsburgh, Pennsylvania, USA, 249–256.

[10] John Duchi, Elad Hazan, and Yoram Singer. 2010. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR* 12 (2010), 2121– 2159.

[11] X. Glorot and Y. Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proc. AISTATS*.

[12] Morgan Harvey, Mark J. Carman, Ian Ruthven, and Fabio Crestani. 2011. Bayesian Latent Variable Models for Collaborative Item Rating Prediction. In *Proc. CIKM '11*. 699–708.

[13] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk. 2016. Session-based recommendations with recurrent neural networks. In *Proc. ICLR'16*.

[14] S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[15] T. Hofmann. 2004. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.* 22, 1 (January 2004), 89–115.

[16] M.I. Jordan, Z. Ghahramani, T.S. Jaakkola, and L.K. Saul. 1998. An introduction to variational methods for graphical models. In *Learning in Graphical Models*, M.I. Jordan (Ed.). Kluwer, Dordrecht, 105–162.

[17] D. Kingma and M. Welling. 2014. Auto-Encoding Variational Bayes. In *Proc. ICLR'14*.

[18] D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling. 2014. Semi-Supervised Learning with Deep Generative Models. In *Proc. NIPS'14*.

[19] Y. Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proc.14th ACM SIGKDD*. Las Vegas, Nevada, USA.

[20] Y. LeCun, Y. Bengio, and G. Hinton. 2015. Deep learning. *Nature* 512 (2015), 436–444.

[21] Shinichi Nakajima and Masashi Sugiyama. 2011. Theoretical Analysis of Bayesian Matrix Factorization. *J. Machine Learning Research* 12 (2011), 2583–2648.

[22] Ian Porteous, Arthur Asuncion, and Max Welling. 2010. Bayesian Matrix Factorization with Side Information and Dirichlet Process Mixtures. In *Proc. AAAI-10*.

[23] N. Qian. 1999. On the momentum term in gradient descent learning algorithms. *Neural Networks* 12, 1 (1999), 145–151.

[24] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proc. UAI'09*. 452–461.

[25] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. 1986. Learning internal representations by error propagation. In *Parallel Dist. Proc.* MIT Press, 318–362.

[26] Ruslan Salakhutdinov and Andriy Mnih. 2007. Probabilistic Matrix Factorization. In *Proc. NIPS'07*.

[27] Ruslan Salakhutdinov and Andriy Mnih. 2011. Bayesian Probabilistic Matrix Factorization using Markov Chain Monte Carlo. In *Proc. ICML'11*.

[28] R. Salakhutdinov, A. Mnih, and G. Hinton. 2007. Restricted Boltzmann machines for collaborative filtering. In *Proc. ICML'07*. 791–798.

[29] B. Schafer, J. Konstan, and J. Riedl. 1999. Recommender systems in e-commerce. In *Proc. 1st ACM Conference on Electronic Commerce, EC '99*. ACM, New York, NY, USA, 158–166.

[30] Guy Shani, David Heckerman, and Ronen I. Brafman. 2005. An MDP-Based Recommender System. *J. Machine Learning Research* 6 (2005), 1265–1295.

[31] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved Recurrent Neural Networks for Session-based Recommendations. In *Proc. DLRS '16*. ACM.

[32] H. Wang, N. Wang, and D.-Y. Yeung. 2015. Collaborative deep learning for recommender systems. In *Proc. 21st ACM SIGKDD*. ACM, New York, NY, USA, 1235–1244.

[33] Y. Zhang, H. Dai, C. Xu, J. Feng, T. Wang, J. Bian, B. Wang, and T.-Y. Liu. 2014. Sequential click prediction for sponsored search with recurrent neural networks. In *Proc. AAAI-14*. 1369–1375.