Conditioned Variational Autoencoder for Top-N Item Recommendation

Conditioned Variational Autoencoder for top-N item recommendation

Tommaso Carraro[1 [0000−0002−3043−1456]], Mirko Polato[1 [0000−0003−4890−5020]], and Fabio Aiolli[1 [0000−0002−5823−7540]]

Department of Mathematics, University of Padova, Padova, Italy

Abstract. In this paper, we propose a Conditioned Variational Autoencoder (C-VAE) for constrained top-N item recommendation where the recommended items must satisfy a given condition. The proposed model architecture is similar to a standard VAE in which the condition vector is fed into the encoder. The constrained ranking is learned during training thanks to a new reconstruction loss that takes the input condition into account. We show that our model generalizes the state-of-the-art Mult-VAE collaborative filtering model. Moreover, we provide insights on what C-VAE learns in the latent space, providing a human-friendly interpretation. Experimental results underline the potential of C-VAE in providing accurate recommendations under constraints. Finally, the performed analyses suggest that C-VAE can be used in other recommendation scenarios, such as context-aware recommendation.

Keywords: recommender systems, collaborative filtering, implicit feedback, variational autoencoder, top-N recommendation

1 Introduction

Recommender system (RS) technologies are nowadays an essential component for e-services. Generally speaking, an RS aims at providing suggestions for items (e.g., movies, songs, news) that are most likely of interest to a particular user [20]. Since the first appearance of RSs, Collaborative Filtering (CF) [23,13] has affirmed of being the *de facto* recommendation approach. CF exploits similarity patterns across users and items to provide personalized recommendations. Latent Factor models, in particular Matrix Factorization (MF), have dominated the CF scene [7,19,16,18] for years, and this has been further emphasized with the deep learning rise [4]. A growing body of work has shown the potential of (deep) neural network approaches to face the recommendation problem. In the last few years, plenty of neural network-based models have raised the bar in terms of recommendation accuracy, such as NeuCF [5], CDAE [26], and EASE [22], to name a few.

Recently, generative approaches have attracted the researchers' attention to the top-N recommendation task. The first generative models that appeared in the RS literature were based on Generative Adversarial Networks [25,11,24,3]. The GAN-based trend has been followed by a series of Variational Autoencoder-based

(VAE, [12]) methods, which have soon gained much success overshadowing GAN-based ones. The seminal Variational approach for CF has been Mult-VAE [15]. After that, other VAE-based models have been proposed, such as [14] and [8] for the content-based recommendation. In particular, the latter work is highly related to ours since it extends the Conditional VAE [27,17] to collaborative filtering (discussed in Section 2.4).

Here, we extend the Mult-VAE model [15] for conditioned recommendations in the top-N setting. Conditions are intended in a generic sense and they can be both content-based or contextual-based [1]. It is important to underline that the way conditions are treated in our training is different from the one proposed in [8]. In our setting, a condition represents a constraint, and thus the provided recommendations must satisfy the constraint to be accepted. For example, in a movie recommendation system a user can ask for movies that belong to a specific genre.

We designed our model as a generalization of Mult-VAE, and hence if trained without the conditions they are equivalent. Treating the conditions as we do allows the model to be versatile making it potentially applicable as a content-based as well as a context-aware recommender. Additionally, thanks to the training process, the latent space shows nice properties that can be exploited to give a human-friendly interpretation of the model. Our experimental analyses show that our method can achieve state-of-the-art performance on different benchmark data sets.

In summary, our main contributions are:

1. we propose a Conditioned VAE for top-N recommendation, dubbed C-VAE, able to manage conditioned recommendations. We define the C-VAE architecture and a new conditioned loss, with which our model is able to learn the relationships between items and conditions;
2. we provide a descriptive as well as a quantitative comparison with state-of-the-art approaches;
3. we provide a in-depth analysis of the properties of the learned latent space giving a human-friendly interpretation.

The remainder of the paper is structured as follows. Section 2 provides the background useful to follow the rest of the paper. Section 3 describes the proposed method, while Section 4 shows the performed evaluation. Finally, Section 5 wraps up the paper and gives some possible future research paths.

2 Background

This section provides the notations (Section 2.1) and background knowledge (Section 2.2) useful to fully understand the rest of the paper.

2.1 Notation

In this section we provide some useful notation used throughout the paper. We refer to the set of users of a RS with $\mathcal{U}$, where $|\mathcal{U}| = n$. Similarly, the set of

items is referred to as $\mathcal{I}$ such that $|\mathcal{I}| = m$. The set of ratings is denoted by $\mathcal{R} \equiv \{(u,i) \mid u \in \mathcal{U} \wedge i \in \mathcal{I}, u \text{ rated } i\}$. Each item i is assumed to belong to a set of categories $C^{(i)} \subseteq \mathcal{C}$, where $\mathcal{C} \equiv \{C_1, C_2, \ldots, C_s\}$ is the set of all possible categories s.t. $|\mathcal{C}| = s$.

Moreover, since we face top-N recommendation tasks, we consider the binary rating matrix with $\mathbf{R} \in \mathbb{R}^{n \times m}$, where users are on the rows and items on the columns, such that $\mathbf{r}_{ui} = 1$ iff $(u,i) \in \mathcal{R}$. Given $\mathbf{R}$, $\mathbf{r}_u\{0,1\}^m$ indicates the column binary vector corresponding to the user u. We add a subscription to both user and item sets to indicate, respectively, the set of items rated by a user u (i.e., $\mathcal{I}_u$) and the set of users who rated the item i (i.e., $\mathcal{U}_i$). Finally, we indicate with $\mathbf{c} = [c_1, ..., c_s]^\top$ the column binary condition vector, where $c_j = 1$ if and only if $\exists i \in \mathcal{I}_u$ such that i belongs to the category C_j.

2.2 Variational Autoencoder

The VAE is a generative model that assumes the input $\mathbf{x}$ is generated according to the following generative process: $\mathbf{z} \sim p_{\theta^*}(\mathbf{z})$ and $\mathbf{x}|\mathbf{z} \sim p_{\theta^*}(\mathbf{x}|\mathbf{z})$, where the dimensionality of $\mathbf{z}$ is (generally) much lower than $\mathbf{x}$. In other words, VAE assumes that the input vector $\mathbf{x}$ is modeled as a function of an unobserved random vector $\mathbf{z}$ of lower dimensionality. VAE aims at estimating the parameters θ^* by maximizing the likelihood of the data (Maximum Likelihood Estimation, MLE), i.e., $\hat{\theta} = \arg\max_{\theta \in \Theta} p_\theta(\mathbf{x})$. Computing the MLE requires solving

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z}) p_\theta(\mathbf{z}) d\mathbf{z}$$

which is often intractable. However, in practice, for most $\mathbf{z}$, $p_\theta(\mathbf{x}|\mathbf{z}) \approx 0$. The key idea behind VAE is to sample values of $\mathbf{z}$ that are likely to have produced $\mathbf{x}$, and compute $p_\theta(\mathbf{x})$ just from those. To do this, we need an approximation $q_\phi(\mathbf{z}|\mathbf{x})$ of the true posterior distribution that returns a distribution over $\mathbf{z}$ that are likely to produce the input. To make this problem tractable, it is assumed that q_ϕ follows a specific family of parametric distributions, usually a normal distribution with 0 mean and unitary variance. The closeness between $q_\phi(\mathbf{z}|\mathbf{x})$ and the assumed posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$ is ensured by the minimization of the Kullback-Liebler divergence (KL), which can be written as:

$$\mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log q_\phi(\mathbf{z}|\mathbf{x}) - \log p_\theta(\mathbf{x}, \mathbf{z})\right] + \log p_\theta(\mathbf{x}). \quad (1)$$

After some rearrangements of Equation (1) it is possible to write the so-called Evidence Lower BOund (ELBO) [12], which naturally defines the objective function that VAE wants to maximize:

$$\begin{aligned} \log p_\theta(\mathbf{x}) &\geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - \mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z})) \\ &= \mathcal{L}(\mathbf{x}; \theta, \phi). \end{aligned}$$

This loss can be interpreted as a reconstruction loss (first term), plus the so-called KL loss which acts as a kind of regularization term.

In practice, p_θ and q_ϕ are parametrized by two (deep) neural networks, i.e., the decoder (f_θ) and the encoder (g_ϕ), respectively. These parameters are optimized using stochastic gradient ascent with the aid of the reparameterization trick [12], that allows to compute the gradient w.r.t ϕ. To this end, the encoder network provides the parameters which define the distributions over each element of $\mathbf{z}$, i.e, mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}$. The sampling over the Gaussian distribution is performed via an additional input $\boldsymbol{\epsilon}$, which allows the reparameterization $\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\epsilon} \odot \boldsymbol{\sigma}$, where $\odot$ is the Hadamard product.

2.3 Variational Autoencoder for collaborative filtering

In [15], Liang et al. propose a VAE for collaborative filtering called Mult-VAE. Mult-VAE takes as input the user-item binary rating matrix and learns a compressed latent representation (the encoder) of the input. These latent representation is then used to reconstruct the input (the decoder) and to impute the missing ratings. The top-N recommendation is computed by taking, for each user, the N items with the highest reconstructed ratings.

Differently from the standard VAE, Mult-VAE uses a multinomial log likelihood rather than the classical Gaussian likelihood. Authors believe that the multinomial distribution is well suited for modeling implicit feedback [15]. Moreover, Mult-VAE employs a β-VAE loss [6] in which the hyper-parameter β is added to the loss as a trade-off parameter between the reconstruction loss and the KL loss.

2.4 Style Conditioned Recommendation (SCR)

In [8] a style conditioned variational autoencoder is proposed. The conditional schema followed by SCR is similar to the standard Conditional VAE [27,17]. The style conditioning is achieved with the addition of a user style profile vector to both the input of the encoder and the decoder. This style vector representation is learned though another network using side information. The rest of the model as well as the training of the network is the same as in a standard VAE, where the input of the encoder and the decoder is the concatenation of their input with the style vector.

3 Conditioned Variational Autoencoder

In this section we present C-VAE for top-N recommendation. We first underline the differences with the state-of-the-art (Section 3.1), and then we define the architecture (Section 3.2) as well as the new loss (Section 3.3) of our model.

3.1 Preliminaries

Once learned the user style profile, the conditioning proposed in [8] is fixed for the user. The only way to force different styles is by acting directly in the latent

space by injecting a specific style via a one-hot encoded style vector. With this trick the decoder network is driven to reproduce inputs with a specific style. However, this is a post-training step, and hence only the decoding is influenced by the injection.

Since our conditions represent constraints, in C-VAE the conditional vector is fed into the encoder network, and the training process guarantees that the learned latent representation depends on the given condition. In this way, different conditions map the user onto (potentially) different regions of the latent space. In Section 4.7 we show that this is a nice property when it comes to interpret the model. C-VAE differs from SCR in the following main aspects:

- C-VAE is architecturally simpler than SCR (Section 3.2);
- C-VAE is more versatile than SCR and it is a natural generalization of the Mult-VAE (Section 2.3) model;
- C-VAE learns the correlations between users and conditions (SCR only learns a specific conditioning for each user).

3.2 Architecture

C-VAE follows the architecture of Mult-VAE with the addition of a conditional vector to the input of the encoder network. The conditional vector is concatenated with the user rating vector after the dropout layer. The dropout layer (implemented in [15] but not reported in the paper) gives to the VAE denoising capabilities, which have shown of being effective in making recommendations [15]. The conditional vector $\mathbf{c} \in \{0,1\}^s$ is defined as a one-hot vector over the s possible conditions. It is noteworthy that the condition, differently from SCR, is not fed into the decoder network.

Figure 1 depicts the network architecture of our model, while Figure 2 provides an overview of the architectural differences between Mult-VAE, SCR, and C-VAE.

3.3 Conditioned loss function

A core difference between our C-VAE and SCR is the way the training works. Since we treat the conditioning as a constraint, the reconstruction must take this into account. This means that, in general, the expected output is a filtered version of the input, where the items that do not satisfy the constraint are dropped. This is achieved by our model via a modified loss function.

The loss function we try to minimize is a conditioned version of the Mult-VAE loss [15]:

$$\mathcal{L}_\beta\left(\mathbf{r}_u, \mathbf{c}; \theta, \phi\right) = \mathbb{E}_{q_\phi(\mathbf{z}_u|\mathbf{r}_u,\mathbf{c})}\left[\log p_\theta\left(\hat{\mathbf{r}}_u|\mathbf{z}_u, \mathbf{c}\right)\right] - \beta \cdot \mathrm{KL}\left(q_\phi\left(\mathbf{z}_u|\mathbf{r}_u, \mathbf{c}\right) \| p\left(\mathbf{z}_u, \mathbf{c}\right)\right)$$

where $\mathbf{z}_u$ is the latent representation of the user u, and $\hat{\mathbf{r}}_u$ is $\mathbf{r}_u$ filtered by the condition $\mathbf{c}$. The filtering is directly embedded in the reconstruction loss as:

$$\log p_\theta\left(\hat{\mathbf{r}}_u|\mathbf{z}_u, \mathbf{c}\right) = \sum_{i \in \mathcal{I}} \langle \mathbf{c}, \mathbf{G}^\top \rangle_i \, \mathbf{r}_{ui} \log \pi_i(f_\theta(\mathbf{z}_u))$$

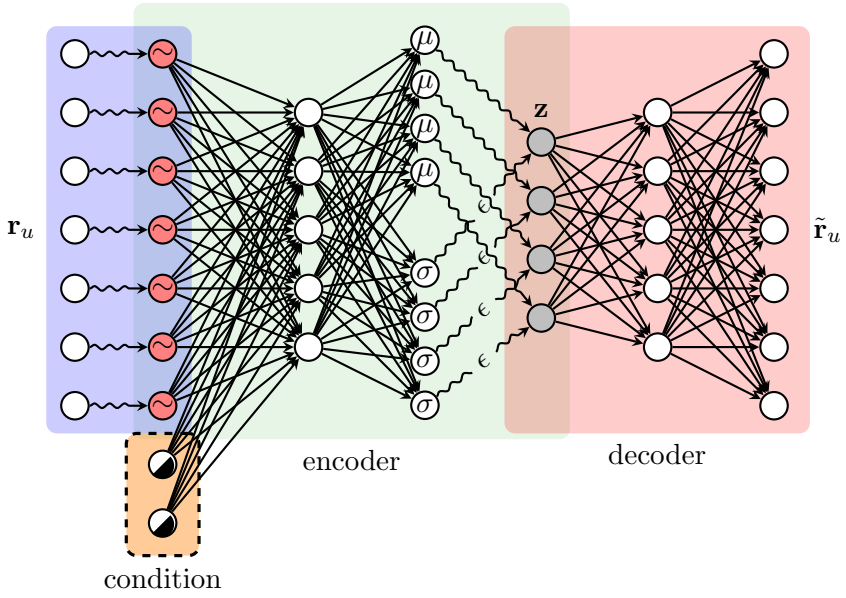


Fig. 1: High level illustration of the Conditioned VAE architecture.

where:

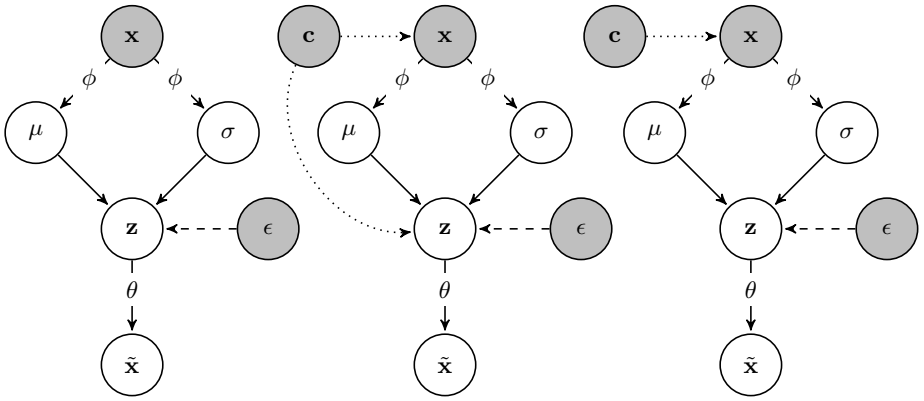
- π is the softmax function;
- $\langle \cdot, \cdot \rangle$ indicates the inner-product operation;
- $\mathbf{G} \in \{0, 1\}^{m \times s}$ is the item-condition matrix, where $g_{ic} = 1$ iff item i satisfies the condition c.

This reconstruction loss is what makes our model able to learn the relationship between items and conditions. When conditions are not used, then all items are assumed of satisfying the empty condition. Implementation-wise, this is achieved by removing the dot product $\langle \mathbf{c}, \mathbf{G}^\top \rangle$, which is indeed always equal to the constant vector $\mathbf{1}$. This leads the C-VAE loss to be the equal to the loss function defined in [15], making C-VAE equivalent to Mult-VAE.

It is worth to underline that the filtering part can also be dependent from both user and item. For example, in the case of context-aware recommendation [1], the conditioning can be defined in terms of the context, which is influenced by both users and items. In this case, the reconstruction loss must be modified accordingly by defining a different condition matrix $\mathbf{G}$. In this paper, we focus on conditioning defined over the items' content.

4 Experiments

In this section we present the experiments performed on C-VAE. We compared C-VAE with Mult-VAE in terms of top-N recommendation accuracy (Section 4.5). We simulate the conditioning on Mult-VAE by filtering its output



(a) Mult-VAE (b) SCR (c) C-VAE

Fig. 2: Overview of the architectural differences between (a) Mult-VAE, (b) SCR, and (c) C-VAE. The dashed arrows denote a sampling operation, while the dotted arrows indicate the conditional input.

according to the given condition. Finally, we analyze the latent space of C-VAE which allowed us to shed some lights on what happens under hood.

4.1 Datasets

We performed our experiments on three real-world data sets, chosen in such a way that they contained items side information to construct the conditions.

`MovieLens 20M`[1] (`ml-20m`): This data set contains user-movie ratings collected from a movie recommendation service. We took the genres of the movies as conditions. We removed the rarest genres (i.e., `IMAX`, `Film-Noir` and the neutral genre `(no genres listed)`) because they were poorly represented in the data set. For the rest of the pre-processing we followed the same procedure as in [15].

`Yelp`[2]: This data set is a subset of Yelp's businesses, reviews, and user data. It was originally put together for the Yelp Data set Challenge. We took the categories of the businesses as conditions. We kept the 20 most popular restaurant categories as described in [21]. Afterwards, since we work on implicit feedback we binarized explicit data by keeping ratings of three or higher. Finally, we only kept users who have reviewed at least four restaurants and restaurants that have been reviewed by at least ten users.

`Netflix Prize`[3]: This is the official data set used in the Netflix Prize competition. As of `ml-20m`, we took the genres of the movies as conditions. Since

[1] `https://grouplens.org/datasets/movielens/20m/`
[2] `https://www.yelp.com/dataset`
[3] `https://www.kaggle.com/netflix-inc/netflix-prize-data`

the dataset does not include information about the genres, we developed a script to fetch these information from the *IMDb*[4] database. We used the title and year of the movies to query the database. Netflix dataset originally contained 17770 movies but only 12279 matched with `IMDb`. The retrieved movies-genres mapping is available at this URL[5]. As in movielens, we removed the rarest genres, i.e., `Talk-Show`, `Film-Noir`, `Short`, `Reality-TV`, `News` and `Game-Show`. For the rest of the pre-processing we followed the same procedure as in [15].

Table 1 summarizes the information about the data sets after the pre-processing presented above.

Table 1: Composition of datasets after pre-processing.

	`ml-20m`	`Yelp`	`Netflix`
# of users	136,466	125,679	459,133
# of items	19,619	22,824	11,844
# of categories	17	20	21
# of interactions	19.3M	2.9M	88.8M
% of interactions	0.7	0.1	1.6
# of held-out users	10,000	9,000	40,000
# training examples	1,728,205	759,955	6,826,774
# validation examples	144,179	47,364	699,901
# test examples	143,965	46,847	700,393

4.2 Conditions computation

Potentially, during the training of C-VAE it might be possible to condition each user with every possible conditions combination. However, the size of the training set would be in the order of $\mathcal{O}(n \cdot s^2)$. We decided to limit its size by conditioning users one category at a time, i.e., $\|\mathbf{c}\|_1 = 1$. If a condition is never satisfied by the user's item set $\mathcal{I}_u$ then the condition is simply not considered in the training. In the training set we also considered the users without any condition (akin Mult-VAE).

The bottom part of Table 1 summarizes the size of the training, validation and test sets after the computation of the conditions.

4.3 Model architecture

An overview of the architecture of our model is presented in Section 3.2. We followed the implementation as in [15], where an L^2 normalization and a dropout

[4] `https://www.imdb.com/`

[5] `https://github.com/bmxitalia/netflix-prize-with-genres`

layer ($p = 0.5$) are applied to the input $\mathbf{r}_u$ before it is fed to the encoder. The encoder network is composed of a fully connected layer made of 600 neurons with *tanh* as activation function. The encoder outputs the mean and the standard deviation of a Gaussian distribution, that are represented with two fully connected layers made of 200 neurons and linearly activated. The decoder network is composed of a fully connected layer made of 600 neurons with *tanh* as activation function. Finally, the decoder linearly outputs the scores over the entire items set.

Recalling that $m = |I|$ and $s = |\mathcal{C}|$, the neural architecture of our model can be summarized as $[m + s \implies 600 \implies 200 \implies 600 \implies m]$. For Mult-VAE we used the same architecture.

4.4 Model training and hyper-parameters tuning

For both Mult-VAE and C-VAE the network weights are initialized with Xavier uniform initializer, while biases are normally initialized with 0 mean and standard deviation 0.001. We used the Adam optimizer with learning rate 0.001. For the tuning of the hyper-parameter β we used the procedure explained in [15]. As a reminder, this is the procedure we followed:

- we trained the model annealing β in such a way to reach 1.0 at the end of the training;
- we selected the β value corresponding to the highest validation score in terms of nDCG@100 [9];
- we re-trained the model annealing β in such a way to reach the selected value at the end of the training.

In our experiments we found that the best values for β are 0.07 for `ml-20m`, 0.35 for `Yelp` and 0.05 for `Netflix`. We used a batch size of 500 for `Yelp` and `ml-20m`, while for `Netflix` a batch size of 1000. We trained the models for 100 epochs on every data set and we kept the model which corresponded to the best validation score. We used early stopping to stop the training if no improvements were found on the validation score for 5 consecutive epochs.

4.5 Experimental results and discussion

In this section we compare C-VAE and Mult-VAE in terms of top-N reocmmendation quality. We used recall@k and nDCG@k as ranking-based metrics. While recall@k considers all items ranked within the first k to be equally important, nDCG@k uses a monotonically increasing discount to emphasize the importance of higher ranked items. We did not have the chance to compare our method with SCR because authors did not provide the implementation of their method. Experiments have been performed using the *rectorch*[6] python library. To quantitatively compare our proposed method with Mult-VAE we used three different types of evaluation:

[6] `https://github.com/makgyver/rectorch`

1. **total**: measures how well the model performs in general, that is when the test set contains both users with conditions and users without conditions;
2. **normal**: measures how well the model performs without conditioning, that is when test set contains only users without conditions;
3. **conditioned**: measures how well the model performs with conditioning, that is when the test set contains only users with conditions.

Since Mult-VAE does not directly handle the conditioning, we filtered its output according to the condition and we computed the ranking only on those items that satisfy the condition. It is worth to notice that this filtering gives Mult-VAE a huge advantage because it greatly narrows down the item set. Clearly, C-VAE instead performs the ranking over the whole item set.

To validate and test the models, for each validation/test user we fed 80% of user ratings to the network and reported metrics on the remaining 20% of the ratings history (for `Yelp` we used 50/50 proportions due to its sparsity). Table 2 reports the obtained results.

Table 2: Comparison between C-VAE and Mult-VAE on selected benchmark data sets. Standard errors are between 0.001 and 0.003. r stands for recall, while n stands for nDCG. Each metric is averaged across all test users.

Dataset	Method	Total			Normal			Conditioned		
		r@20	r@50	n@100	r@20	r@50	n@100	r@20	r@50	n@100
`ml-20m`	C-VAE	0.638	0.786	0.509	0.385	0.527	0.410	0.666	0.816	0.521
	Mult-VAE	**0.645**	**0.792**	**0.517**	**0.394**	**0.537**	**0.420**	**0.674**	**0.822**	**0.529**
`Yelp`	C-VAE	**0.311**	0.459	**0.238**	**0.139**	**0.235**	**0.143**	0.392	0.564	**0.282**
	Mult-VAE	**0.311**	**0.460**	**0.238**	0.134	0.233	**0.143**	**0.394**	**0.567**	**0.282**
`Netflix`	C-VAE	0.590	0.751	0.494	0.335	0.444	0.374	0.613	0.778	0.504
	Mult-VAE	**0.601**	**0.758**	**0.504**	**0.352**	**0.457**	**0.389**	**0.623**	**0.785**	**0.515**

From the table, it is possible to observe that C-VAE obtains state-of-the-art results, even though generally a bit lower than Mult-VAE. We want to emphasize one more time that C-VAE performs the ranking over all items, while Mult-VAE only on the subset of items satisfying the condition. This shows that our method is able to learn the relationships between items and categories, since it is able to push the items that belong to the target category at the top of the ranking.

4.6 Analysis of the C-VAE produced rankings

Since we obtained promising results in terms of ranking accuracy, we decided to analyze the categories distribution on the rankings produced by C-VAE. Thanks to the conditioned loss, C-VAE learns how to filter the items in such a way to focus its attention on the items belonging to the target category. To further validate this argument, we plot the distribution over the rankings produced by C-VAE for the items satisfying the conditions. The plot has been computed on the `ml-20m` training users and is shown in Figure 3. It clearly shows that most of

the items of the target category have been placed in the top positions, showing that C-VAE learns to push the right items at the top.

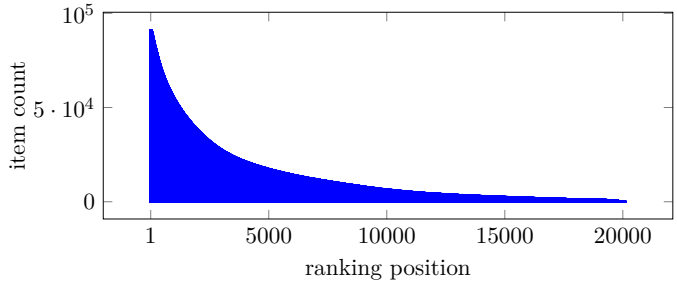


Fig. 3: Number of items of the target category per ranking position.

A further check have shown that in the first 100 positions of the ranking C-VAE is always able to put only items satisfying the input condition.

4.7 Latent space exploration

Given the promising results discussed previously, we decided to further investigate the inner representations of C-VAE. In particular, to explore the latent space of the C-VAE model we took 2000 random users from the original `ml-20m` data set (no genres have been removed). We analyzed their learned latent representations by conditioning all of them on each genre, and also without the condition. We performed Principal Component Analysis (PCA) [10] and considered only the first 5 principal components.

We noticed that the first principal component separates the `(no genre listed)` genre from all the other genres. Thus, we decided to remove this *neutral* genre and the first principal component. We also observed that the fourth principal component (not illustrated here) stretches the clusters on a single dimension, underling that even with the same conditioning users still have different tastes. Principal components 2 and 5 showed really good properties giving an intuitive understanding of the genres correlation. Figure 4 (left hand side) plots these components.

Looking at the figure, the following observations can be done:

- popular and common genres (e.g., `Action`, `Comedy`, `Drama`, `Romance`) are placed close to the center of the latent space, while less popular ones (e.g., `Film-noir`, `Children`, `Animation`) are placed far aside;
- very different genres are placed distant from each other, while similar genres are placed near to each other. For example, `War` is distant from `Children` and `Animation`, while it is close to `Drama` and `Romance`;
- the not conditioned representations (in black) are placed at the center of the space. We argue that when C-VAE recommends movies without the condi-

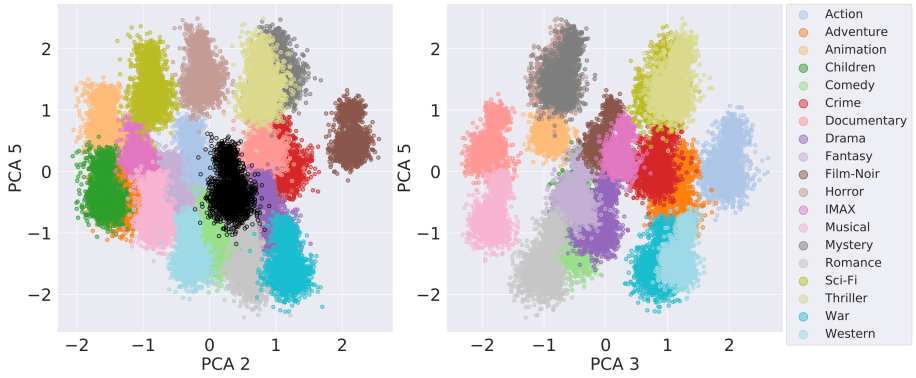


Fig. 4: (left) Second and fifth, (right) third and fifth components of PCA performed on selected users latent representations on `ml-20m`.

tion, it computes the unconditioned ranking and the most popular genres become more likely.

The 2D plot of the third and fifth principal components (right hand side of Figure 4) offers a different perspective with respect to the previous one. We argue that the third principal component captures the *emotional theme* of the genres. For example, `Mistery` and `Horror` have similar emotional components (e.g., anxiety, tension, fear) and they almost completely overlap. Similar considerations can be done for `Children-Fantasy` and `War-Western`.

5 Conclusions and future work

In this paper, we presented a novel method for conditioning the top-N item recommendation process. We developed a conditioned extension of Mult-VAE [15], which relies on a novel loss function that is crucial for the training of our method. We compared our method with the state-of-the-art for top-N recommendation, i.e., Mult-VAE, and we showed that C-VAE reaches similar performance in both the conditioned as well as the non conditioned top-N recommendation tasks. Additionally, we explored the learned latent space of C-VAE and we observed that is able to capture not only the relationships between items and categories, but also the relationships between categories themselves. We also offered an intuitive and human-like interpretation of the latent representation.

As long as we perform content-based recommendations our model is less powerful than the filtered Mult-VAE, but in context-aware scenarios C-VAE can capture patterns between users, items and the interactions between them, while Mult-VAE cannot be applied since the filtering is no longer applicable. In conclusion, it is our intent to extend our evaluation to other open research topics, in particular context-aware and also group recommender systems [2].

References

1. Adomavicius, G., Tuzhilin, A.: Context-Aware Recommender Systems, pp. 191–226. Springer US, Boston, MA (2015). https://doi.org/10.1007/978-1-4899-7637-6_6, https://doi.org/10.1007/978-1-4899-7637-6_6
2. Boratto, L.: Group recommender systems. In: Proceedings of the 10th ACM Conference on Recommender Systems. p. 427428. RecSys 16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2959100.2959197, https://doi.org/10.1145/2959100.2959197
3. Chae, D.K., Kang, J.S., Kim, S.W., Lee, J.T.: Cfgan: A generic collaborative filtering framework based on generative adversarial networks. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management. p. 137146. CIKM 18, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3269206.3271743, https://doi.org/10.1145/3269206.3271743
4. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. The MIT Press (2016)
5. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.S.: Neural collaborative filtering. In: Proceedings of the 26th International Conference on World Wide Web. p. 173182. WWW 17, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE (2017). https://doi.org/10.1145/3038912.3052569, https://doi.org/10.1145/3038912.3052569
6. Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M.M., Mohamed, S., Lerchner, A.: beta-vae: Learning basic visual concepts with a constrained variational framework. In: ICLR (2017)
7. Hu, Y., Koren, Y., Volinsky, C.: Collaborative filtering for implicit feedback datasets. In: 2008 Eighth IEEE International Conference on Data Mining. pp. 263–272 (2008)
8. Iqbal, M., Aryafar, K., Anderton, T.: Style conditioned recommendations. In: Proceedings of the 13th ACM Conference on Recommender Systems. p. 128136. RecSys 19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3298689.3347007, https://doi.org/10.1145/3298689.3347007
9. Järvelin, K., Kekäläinen, J.: Ir evaluation methods for retrieving highly relevant documents. In: Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. p. 4148. SIGIR 00, Association for Computing Machinery, New York, NY, USA (2000). https://doi.org/10.1145/345508.345545, https://doi.org/10.1145/345508.345545
10. Jolliffe, I.T., Cadima, J.: Principal component analysis: a review and recent developments. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences **374** (2016)
11. Kang, W., Fang, C., Wang, Z., McAuley, J.: Visually-aware fashion recommendation and design with generative image models. In: 2017 IEEE International Conference on Data Mining (ICDM). pp. 207–216 (2017)
12. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: Bengio, Y., LeCun, Y. (eds.) 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014 (2014)

13. Koren, Y., Bell, R.: Advances in Collaborative Filtering, pp. 145–186. Springer, Boston, MA (2011). https://doi.org/10.1007/978-0-387-85820-3_5, https://doi.org/10.1007/978-0-387-85820-3_5
14. Li, X., She, J.: Collaborative variational autoencoder for recommender systems. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 305314. KDD 17, Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3097983.3098077, https://doi.org/10.1145/3097983.3098077
15. Liang, D., Krishnan, R.G., Hoffman, M.D., Jebara, T.: Variational autoencoders for collaborative filtering. In: Proceedings of the 2018 World Wide Web Conference. p. 689698. WWW 18, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE (2018). https://doi.org/10.1145/3178876.3186150, https://doi.org/10.1145/3178876.3186150
16. Ning, X., Karypis, G.: Slim: Sparse linear methods for top-n recommender systems. In: 2011 IEEE 11th International Conference on Data Mining. pp. 497–506 (2011)
17. Pagnoni, A., Liu, K., Li, S.: Conditional variational autoencoder for neural machine translation. ArXiv **abs/1812.04405** (2018)
18. Polato, M., Aiolli, F.: Boolean kernels for collaborative filtering in top-n item recommendation. Neurocomput. **286**(C), 214225 (Apr 2018). https://doi.org/10.1016/j.neucom.2018.01.057, https://doi.org/10.1016/j.neucom.2018.01.057
19. Rendle, S.: Factorization machines. In: 2010 IEEE International Conference on Data Mining. pp. 995–1000 (2010)
20. Ricci, F., Rokach, L., Shapira, B.: Recommender Systems Handbook. Springer Publishing Company, Incorporated, 2nd edn. (2015)
21. Spagnuolo, C., Cordasco, G., Szufel, P., Pralat, P., Scarano, V., Kaminski, B., Antelmi, A.: Analyzing, exploring, and visualizing complex networks via hypergraphs using simplehypergraphs.jl. Internet Mathematics (Apr 2020). https://doi.org/10.24166/im.01.2020, http://dx.doi.org/10.24166/im.01.2020
22. Steck, H.: Embarrassingly shallow autoencoders for sparse data. In: The World Wide Web Conference. p. 32513257. WWW 19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3308558.3313710, https://doi.org/10.1145/3308558.3313710
23. Su, X., Khoshgoftaar, T.M.: A survey of collaborative filtering techniques. Adv. in Artif. Intell. **2009** (Jan 2009). https://doi.org/10.1155/2009/421425, https://doi.org/10.1155/2009/421425
24. Wang, H., Wang, J., Wang, J., Zhao, M., Zhang, W., Zhang, F., Xie, X., Guo, M.: Graphgan: Graph representation learning with generative adversarial nets. In: McIlraith, S.A., Weinberger, K.Q. (eds.) Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018. pp. 2508–2515. AAAI Press (2018), https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16611
25. Wang, J., Yu, L., Zhang, W., Gong, Y., Xu, Y., Wang, B., Zhang, P., Zhang, D.: Irgan: A minimax game for unifying generative and discriminative information retrieval models. In: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval. p. 515524. SIGIR 17, Association for Computing Machinery, New York,

NY, USA (2017). https://doi.org/10.1145/3077136.3080786, https://doi.org/10.1145/3077136.3080786

26. Wu, Y., DuBois, C., Zheng, A.X., Ester, M.: Collaborative denoising auto-encoders for top-n recommender systems. In: Proceedings of the Ninth ACM International Conference on Web Search and Data Mining. p. 153162. WSDM 16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2835776.2835837, https://doi.org/10.1145/2835776.2835837
27. Zhang, B., Xiong, D., Su, J., Duan, H., Zhang, M.: Variational neural machine translation. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. pp. 521–530. Association for Computational Linguistics, Austin, Texas (Nov 2016). https://doi.org/10.18653/v1/D16-1050, https://www.aclweb.org/anthology/D16-1050