A Look Inside the Black-Box: Towards the Interpretability of Conditioned Variational Autoencoder for Collaborative Filtering

A Look Inside the Black-Box: Towards the Interpretability of Conditioned Variational Autoencoder for Collaborative Filtering

Tommaso Carraro
tommasocarraro96@gmail.com
University of Padova
Padova, Italy

Mirko Polato
mpolato@math.unipd.it
University of Padova
Padova, Italy

Fabio Aiolli
aiolli@math.unipd.it
University of Padova
Padova, Italy

ABSTRACT

Deep learning-based recommender systems are nowadays defining the state-of-the-art. Unfortunately, their hard interpretability restrains their application in scenarios in which explainability is required/desirable. Many efforts have been devoted to injecting explainable information inside deep models. However, there is still a lot of work that needs to be done to fill this gap. In this paper, we take a step in this direction by providing an intuitive interpretation of the inner representation of a conditioned variational autoencoder (C-VAE) for collaborative filtering. The interpretation is visually performed by plotting the principal components of the latent space learned by the model on MovieLens. We show that in the latent space conditions on correlated genres map users in close clusters. This characteristic enables the model to be used for profiling purposes.

KEYWORDS

recommender systems, explainability, interpretability, variational autoencoder, collaborative filtering

ACM Reference Format:
Tommaso Carraro, Mirko Polato, and Fabio Aiolli. 2020. A Look Inside the Black-Box: Towards the Interpretability of Conditioned Variational Autoencoder for Collaborative Filtering. In *Adjunct Proceedings of the 28th ACM Conference on User Modeling, Adaptation and Personalization (UMAP '20 Adjunct), July 14–17, 2020, Genoa, Italy.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3386392.3399305

1 INTRODUCTION

Since their first appearance in early 2000, the Recommender systems community has always privileged "simple" and intuitive approaches to tackle the recommendation problem. Simplicity helps in understanding the meaning behind a recommendation in a human-like fashion. For many years, similarity-based collaborative filtering (CF) methods have dominated the scene with their effectiveness and simplicity. Afterward, latent factor models (LFM) have shown of being capable of producing new (latent) representations able to capture more sophisticated collaborative nuances. The most famous LFM is Matrix Factorization (MF) [11] that has a really easy interpretation: users and items are projected onto a common (latent) space where a "compatibility" score is computed. The features in this latent space are interpreted as item characteristics (with a certain degree of presence) that users appreciate to some extent. The more the user likes items features the higher the compatibility.

With the rise of deep learning, a constantly increasing number of deep collaborative filtering models have appeared in the literature [6, 13]. However, the black-box nature of these highly non-linear models often creates skepticism in their usage, especially when some sort of explanation is welcomed. For this reason, many efforts have been devoted to making deep models more interpretable/explainable [2, 5]. Most of the proposed approaches try to inject supplementary knowledge (e.g., [2] and [5]) which enables the explainability of the model. Nonetheless, many state-of-the-art deep collaborative models are not meant to be directly interpretable. Examples are generative models like Generative Adversarial Network-based models [4], or Variational Autoencoder-based models [3, 12]. These machine learning models are not explainable/interpretable by design, yet they can be studied and analyzed to understand what happens under the hood [1].

In this work, we take a step in this direction by giving a human-like interpretation of the inner representation of a conditioned variational autoencoder for collaborative filtering [3, 12]. We take this deep black-box model and study its internal latent representation to understand what the network learns. We train a conditioned variational autoencoder (C-VAE [3]) on the MovieLens 20M data set, and we study what happens in the latent space. The latent space exploration is performed using visual aids. We plot the first Principal Components of the latent space and analyze the relationship between users' profiles. Interestingly, the C-VAE network can learn correlations between movie genres that are used to condition the recommendation. It is worth noting that such correlations are not directly fed into the network, but they are learned by the model through the user profiles. Moreover, thanks to the latent space regularization of the VAE-based models, we show that the latent space can also be explored as a mean to profile the user.

2 BACKGROUND

In this section, we briefly review all the background knowledge necessary to grasp the contributions of the paper.

Autoencoder (AE) Autoencoder is an unsupervised machine learning model based on a neural network. It is designed to learn an identity function that reconstructs the original input while compressing the data in the process. This compression capability makes autoencoders useful for performing (non-linear) dimensionality reduction. The high-level design of an AE can be defined abstracting

UMAP '20 Adjunct, July 14–17, 2020, Genoa, Italy

ACM ISBN 978-1-4503-7950-2/20/07…$15.00
https://doi.org/10.1145/3386392.3399305

from its connection to neural nets. Given an input $\mathbf{x} \in \mathcal{X}$, an AE learns a pair of functions (g_ϕ, f_θ) such that $f_\theta(g_\phi(\mathbf{x})) = \tilde{\mathbf{x}} \approx \mathbf{x}$. So, with a perfect learning, $\forall \mathbf{x} \in \mathcal{X}, f_\theta(g_\phi(\mathbf{x})) = \mathbf{x}$, and $g_\phi \circ f_\theta$ is the identity function. The function $g_\phi : \mathcal{X} \longrightarrow \mathcal{Z}$ is called *encoder*, since its purpose is to compress the information, i.e., $dim(\mathcal{Z}) \ll dim(\mathcal{X})$. On the contrary, $f_\theta : \mathcal{Z} \longrightarrow \mathcal{X}$ is called *decoder* and it has to recover all the input information starting from a compressed representation.

Neural networks come into play because they are the most efficient and effective way to learn a (highly non-linear) parametric function. Thus, both g_ϕ and f_θ are (deep) neural networks parametrized by ϕ and θ, respectively. These parameters are learned together in a standard back-propagation fashion via a gradient descent-based procedure. The learning process aims at minimizing a reconstruction loss, i.e., how much the reconstruction $\tilde{\mathbf{x}}$ is distant from $\mathbf{x}$.

Variational Autoencoder (VAE) Variational autoencoder [10] is still an autoencoder but its theoretical backbone is rooted in the methods of Bayesian inference. Broadly speaking, the core difference between VAEs and standard AEs lies in the way inputs are encoded onto the latent space. VAE, instead of mapping an input $\mathbf{x}$ into a fixed vector (i.e., point) $\mathbf{z}$ in the latent space, maps $\mathbf{x}$ into a probability distribution. This implies that each reconstruction is potentially a slight perturbation of the input.

Specifically, AEs map an input $\mathbf{x}$ in a latent vector $\mathbf{z} = g_\phi(\mathbf{x})$ that is fixed once g_ϕ has been learned. VAE, instead, given $\mathbf{x}$ maps it into a probability distribution $q_\phi(\mathbf{z}|\mathbf{x})$, parametrized by ϕ. Thus, the encoder part of the network learns the distribution parameters ϕ. The latent factors distribution is often meant to be normal with diagonal covariance matrix (i.e., latent factors are independent), and this is modeled via the mapping of $\mathbf{x}$ (via g_ϕ) into a mean vector $\boldsymbol{\mu}_\phi(\mathbf{x})$ and a (log) standard deviation vector $\boldsymbol{\sigma}_\phi(\mathbf{x})$. Since the encoding does not produce a single vector but a (bell-shaped) distribution, the decoding part needs to sample from the learned distribution before proceeding with the reconstruction. The sampling operation is performed through an additional input $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ (*reparameterization trick*) that allows to sample a latent representation by computing $\mathbf{z} = \boldsymbol{\mu}_\phi(\mathbf{x}) + \boldsymbol{\epsilon} \odot \boldsymbol{\sigma}_\phi(\mathbf{x})$. Also, the learning process differs from AE. The loss function is composed of a reconstruction part (as of AE) plus a so-called KL (Kullback-Leiber) loss. The KL loss works as a regularizer of the latent space by penalizing spaces that are far from a normal distribution with mean 0 and variance 1. This constraint guarantees very nice properties of the latent space, such as *completeness* and *continuity*.

VAE for collaborative filtering In [12] VAE is successfully applied in the context of top-N recommendation. Starting from the binary rating matrix $\mathbf{R} \in \{0, 1\}^{n \times m}$, with n users and m items, the reconstruction is performed over the users' rating vector $\mathbf{r}_u$ where $r_{ui} = 1$ means that user u interacted with item i. In this work authors employed a variant of VAE, called Mult-VAE, in which: (i) they assume a multinomial prior on the input $\mathbf{r}_u$, (ii) they add noise to the input (i.e., apply a dropout layer), and (iii) they add in the loss a hyper-parameter β (β-VAE [7]) which acts as a trade-off between reconstruction loss and KL loss. The architecture, excluding the orange (condition) part, is shown in Figure 1. Extensive experimental results on diverse benchmark data sets show that Mult-VAE achieves state-of-the-art performance.

Conditioned VAE (C-VAE) Conditioned VAE [3] is a variant of Mult-VAE which allows specifying constraint in the recommendation. For example, in a movie recommendation scenario, a user can indicate a specific genre (or set of genres) that is willing to watch. With a standard Mult-VAE, this cannot be done and the provided recommendation will be always the same. C-VAE addresses this limitation (i) by adding a constraint vector to the input, and (ii) by changing the training algorithm. Figure 1 depicts the full C-VAE's architecture.

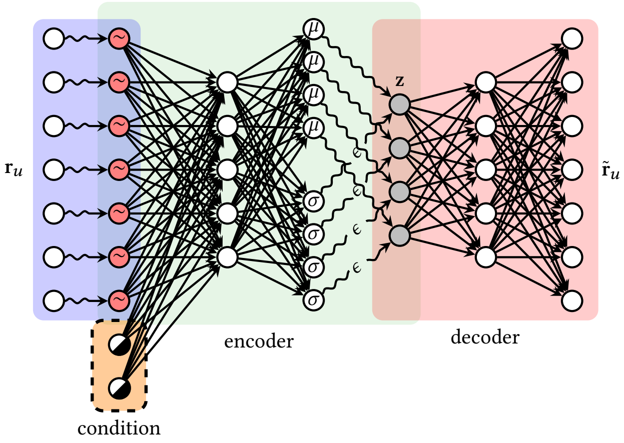

Figure 1: High level illustration of the Conditioned VAE architecture. Red nodes indicate the dropout (noise) layer. The orange (dashed) box highlights the one-hot condition vector.

In C-VAE, the condition vector $\mathbf{c} \in \{0, 1\}^C$, where C is the number of possible conditions, is a one-hot vector in which 1 means that the condition is desirable in the recommendation. So, if a user is a *sci-fi* addicted but it asks for *comedy*, the recommender must push comedy movies on top of the recommended list. The learning process is carried out as in Mult-VAE with a difference in the reconstruction loss: the reconstructed input $\tilde{\mathbf{r}}_u$ must be as close as possible to $\hat{\mathbf{r}}_u$, where $\hat{r}_{ui} = 1$ iff $r_{ui} = 1$ and i satisfies the condition $\mathbf{c}$. In other words, the reconstruction must demote the items that do not satisfy the condition. Conditioning the recommendation can be performed on content features (as we do in this paper) as well as on contextual information. If $\mathbf{c}$ is fixed to $\mathbf{0}$, C-VAE is equivalent to Mult-VAE. Given the just described loss, the training is the same as in Mult-VAE in which a training user appears in the training set conditioned on every watched genre, i.e., she appears $c_u + 1$ times, where c_u is the number of different genres she watched.

3 INTERPRETING C-VAE

In this section, we describe the performed study on the C-VAE latent space. To conduct our experiments we selected the MovieLens 20M [1] dataset, that is one of the most popular datasets for recommendation systems. This dataset is suitable for conditioning the input because it contains user-movie ratings collected from a movie recommendation service. The dataset preprocessing follows the procedure described in [12]. Since we work with implicit feedback,

[1] https://grouplens.org/datasets/movielens/20m/

we binarized the explicit data by keeping ratings of 4 or higher. We only kept users that watched at least five movies and we took the genres of the movies as conditions while training C-VAE. We split the dataset into training, validation and test sets as in [3, 12]. In particular, to validate our model we fed 80% of the users validation ratings to the model and computed the nDCG@100 [8] on the remaining 20% of the users profiles.

As expected, in the data set genres are not evenly distributed. In particular, `Drama` and `Comedy` are the most common genres, while `Film-noir`, `IMAX` and *neutral* (i.e., `no genres listed`) are the least common ones. Neutral movies are movies that do not belong to any specific genre. Since `IMAX` is the type of camera used to film the movie[2], it cannot be considered a movie genre *per se*. So, we decided to understand the distribution of genres when a movie has been filmed with IMAX technology. Interestingly, the majority of movies recorded in IMAX belong to `Action`, `Adventure` and `Sci-fi` genres, while `War` and `Western` movies are the least common. During the training we conditioned users on a single genre to limit epochs' training time. In the training set each user appears conditioned with all the genres of the movies she watched with the addition of the *no condition* (like in Mult-VAE).

We used a C-VAE[3] where the encoder and decoder networks are symmetric. The encoder takes in input a user rating sparse vector together with a one-hot encoded vector that represents the condition. The encoder is composed of one fully connected layer made of 600 neurons and *tanh* as activation function. Also, the encoder output layer is fully connected with 200 neurons for both the mean and standard deviation. A linear activation function is used in this layer. The decoder consists of a fully connected layer made of 600 neurons and *tanh* activated. The last layer of the network is a fully connected layer linearly activated. The output of the model is a vector containing the scores over the entire movie set. Reminding m is the number of movies in the dataset and C the number of genres in the dataset, the architecture of our model is $[m + C \implies 600 \implies 200 \implies 600 \implies m]$.

We initialized the weights of the fully connected layers with Xavier initializer and the biases with the truncated normal with 0 mean and standard deviation 10^{-3}. β has been selected via linear annealing (as described by Liang et al. in [12]). The best performing β has been $8 \cdot 10^{-3}$. We trained our model for 100 epochs and we used early stopping to stop the training if after 10 epochs no improvements were found on the validation metric.

3.1 Latent space exploration

To explore the latent space of the C-VAE model we took 2000 random users from the dataset. 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) [9] and considered only the first 5 principal components (PCs). In the following we discuss only a subset of possible PC combinations. The chosen ones are the most interesting among all the possibilities for the first 5 PCs.

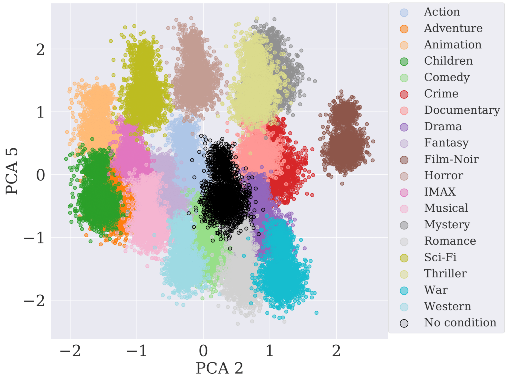

Figure 2: Second and fifth components of PCA performed on selected users latent representations.

The first thing we noticed is the formation of users' clusters (see for example Figure 2 and 3). These clusters correspond to the different input conditions (i.e., movie genres). Thus, C-VAE learns how to cluster the genres, which is not really surprising. However, by plotting the first 3 components (not reported here for space reasons) the (`no genre listed`) genre is placed far away from all the other genres in the first PC. It is reasonable because it is a neutral genre that has nothing to do with the other genres. In fact, every movie with that genre in the data set does not belong to other genres. The fourth principal component (not illustrated here) stretches the clusters on a new dimension underling that even with the same conditioning users still have different tastes.

After the just mentioned considerations, we decided to remove the *neutral* genre and the first principal component. Principal components 2 and 5 (Figure 2) show really interesting features, and the following observation can be done:

- very different genres are placed in very far apart locations. For example, `War` is far away from `Children` and `Animation`, while it is near `Drama` and `Romance`. `Horror` is placed between `Thriller` and `Sci-fi` and is far away from `Western`. `Thriller`, `Mystery` and `Crime` are close to each other;
- popular and common genres (e.g., `Action`, `Comedy`, `Drama`, `Romance`) are placed near the center of the space, while more complex and less popular genres (e.g., `Film-noir`, `Children`, `Animation`) are placed far aside;
- it is interesting that `Film-noir` genre is placed far away from every genre. In fact this genre is difficult to be placed near other genres. Moreover, it is possible to note that the nearest genres are `Crime`, `Drama`, `Thriller` and `Mystery`, that is the closest mix of genres that can be connected to noir movies;
- the not conditioned latent representations (depicted in black) are placed at the center of the space. We think this is due to the fact that when we try to recommend movies without conditioning on a genre, the model computes the unconditioned rank and the most popular genres become more likely. In fact, as previously mentioned, popular genres are near the center of the latent space.

It is worth to notice that all these correlations have been learned by the C-VAE autonomously. Since movies are usually described

[2] https://en.wikipedia.org/wiki/IMAX
[3] We run C-VAE using our `rectorch` python module available at https://github.com/makgyver/rectorch.

by more than one genre, it is not surprising that C-VAE learned that pairs of genres tend to work well together. However, the relative positions in the space and the overall correlation between genres is something that has been learned through the collaborative information provided by the user profiles.

A different but still interesting perspective is provided by the third and fifth PCs (Figure 3). By looking at the way clusters are positioned to each other it is evident that this point of view highly differ from the previous one. We argue (but it needs further investigation) that the third principal component capture the *emotional theme* of the genres. For example, `Mistery` and `Horror` almost completely overlap, and they share many emotional components, such as anxiety, tension and sometimes fear. Very similar considerations can be done for the pairs `Children-Fantasy` and `War-Western`. While `Action`, `Musical` and `Documentary` are a bit offside since they are harder to categorize in a restrict set of emotional states.

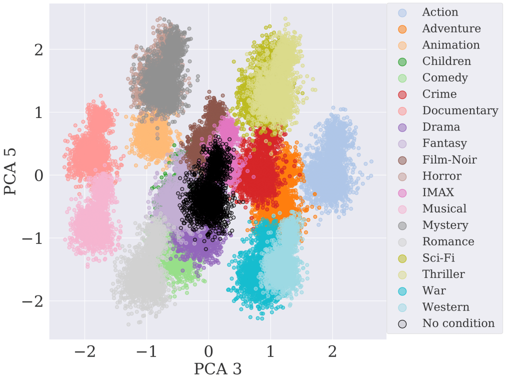

Figure 3: Third and fifth components of PCA performed on selected users latent representations.

3.1.1 *Profiling users.*

As shown in Figure 2 not conditioned users cluster lies in the middle of the latent space. However, it spreads to some extent to any directions and this may suggest user's particular tastes. Users who lie in the bottom of the cluster are more inclined toward light entertainment (`Comedy` and `Romance`). This is also supported by the first three principal components (not shown here for space reasons). In this case, users in the top part of the cluster are mainly interested in more "serious" movies (`Thriller`, `Action` and `Crime`). In this latter cluster we could also include `Documentary`, however on other dimensions this correlation does not hold. This is reasonable since documentaries are generally "serious" but usually less stressful and anxiety-inducing than thriller/crime movies. Hence, given the interpretation of the latent space it is also possible to construct user profiles that can be leveraged to improve recommendation as well as to provide the user a way to check how the system describes her tastes.

4 CONCLUSIONS AND FUTURE WORK

In this paper, we studied Variational Autoencoder and we analyzed a novel VAE for Collaborative Filtering, dubbed C-VAE. We analyzed users' latent representations of a pre-trained model on MovieLens 20M. We performed PCA on a set of random users' latent representations and then we visually analyzed them. We discovered that C-VAE autonomously performs clustering of the inputs in the latent space. In particular, the model creates clusters based on the users' conditions, i.e., movie genres. Moreover, the model has been able to place compatible genres in nearby clusters. We think this is due to the constraint imposed by the model in the training loss.

It is our intent to extend the performed analysis to context-aware scenarios, where the recommendation is based on a context provided in input, such as the day time in which a user interacted with an item. Finally, it will also worth to study methods and algorithms to automatically extract profiling hints from the representation learned by the model.

REFERENCES

[1] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Bolei Zhou, Joshua B. Tenenbaum, William T. Freeman, and Antonio Torralba. 2019. GAN Dissection: Visualizing and Understanding Generative Adversarial Networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019.* OpenReview.net. https://openreview.net/forum?id=Hyg_X2C5FX

[2] Vito Bellini, Angelo Schiavone, Tommaso Di Noia, Azzurra Ragone, and Eugenio Di Sciascio. 2018. Knowledge-Aware Autoencoders for Explainable Recommender Systems. In *Proceedings of the 3rd Workshop on Deep Learning for Recommender Systems (DLRS 2018).* Association for Computing Machinery, New York, NY, USA, 24–31. https://doi.org/10.1145/3270323.3270327

[3] Tommaso Carraro, Mirko Polato, and Fabio Aiolli. 2020. *Conditioned Variational Autoencoder for top-N recommendation.* CoRR abs/2004.11141. https://arxiv.org/abs/2004.11141.

[4] Dong-Kyu Chae, Jin-Soo Kang, Sang-Wook Kim, and Jung-Tae Lee. 2018. CFGAN: A Generic Collaborative Filtering Framework Based on Generative Adversarial Networks. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM '18).* Association for Computing Machinery, New York, NY, USA, 137–146. https://doi.org/10.1145/3269206.3271743

[5] Pegah Sagheb Haghighi, Olurotimi Seton, and Olfa Nasraoui. 2020. An Explainable Autoencoder For Collaborative Filtering Recommendation. *ArXiv* abs/2001.04344 (2020).

[6] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web (WWW '17).* International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 173–182. https://doi.org/10.1145/3038912.3052569

[7] Irina Higgins, Loïc Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings.*

[8] Kalervo Järvelin and Jaana Kekäläinen. 2000. 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 (SIGIR '00).* Association for Computing Machinery, New York, NY, USA, 41–48. https://doi.org/10.1145/345508.345545

[9] Ian T. Jolliffe and Jorge Cadima. 2016. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 374 (2016).

[10] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014*, Yoshua Bengio and Yann LeCun (Eds.).

[11] Y. Koren, R. Bell, and C. Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37.

[12] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *Proceedings of the 2018 World Wide Web Conference (WWW '18).* International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 689–698. https://doi.org/10.1145/3178876.3186150

[13] Shuai Zhang, Lina Yao, Xiwei Xu, Sen Wang, and Liming Zhu. 2017. Hybrid Collaborative Recommendation via Semi-AutoEncoder. In *Neural Information Processing*, Derong Liu, Shengli Xie, Yuanqing Li, Dongbin Zhao, and El-Sayed M. El-Alfy (Eds.). Springer International Publishing, Cham, 185–193.