International Journal of Recent Advances in Multidisciplinary Topics
Volume 5, Issue 5, May 2024
https://www.ijramt.com | ISSN (Online): 2582-7839

Collaborative Filtering Enhanced by Hybrid Variational Autoencoder Framework

Kukunuri Dinesh Kumar*
Student, Department of Computer Science, Mansarovar Global University, Sehore, India

***Abstract*: Personalized recommendations have become crucial in the current era where nearly every industry has an online existence and users engage in online marketplaces. Historically, collaborative filtering had been addressed utilizing Matrix Factorization, which is a linear method. We build on the work described in reference [11] by presenting a hybrid multi-modal approach for collaborative filtering with implicit feedback that uses VAE (Variational Autoencoders). To improve movie recommendation, we combine user ratings from the Movielens 20M dataset with movie embeddings obtained from a related VAE network. We demonstrate how the network of VAE benefits from including movie embeddings through empirical evidence. We cluster the latent representations of movie and user embeddings attained from a VAE and visualize them.**

***Keywords*: collaborative filtering, variational autoencoders, personalization, recommender systems, deep learning, movie embeddings.**

1. Introduction

Recommender systems are crucial in the current environment due to the expansion of social media and online interactions. Individuals often use recommender systems to make decisions about the products they purchase, news articles they read, movies they watch, and songs they listen to. Users have the potential to discover new products in all of these applications. When personalized recommendations are combined with it, it results in higher user engagement, satisfaction, and business profits. Creating customized suggestions has always been and remains difficult. The task involves recommending items to users by considering user context (click-through rate, view history, demographic information) and item context (popularity, genre, description, reviews). Collaborative Filtering (CF) is a highly popular method. Model-based collaborative filtering techniques involve methods like Latent Factor Models, including Matrix Factorization. These approaches are linear, but the interaction among users as well as items appears to be non-linear.

Neural Networks (NNs) have shown significant advancements in digital image processing, natural language processing, autonomous driving, and speech recognition. NNs, specifically deep learning, have been successful due to their capacity to represent complex non-linear data structures. The CF algorithms aim to create a hidden interaction representation among users as well as items. This interaction improved characterization is expected to result in enhanced recommender systems. Promising advancements have been made in utilizing deep learning for collaborative filtering, as evidenced by studies.

Recently, VAEs have been modified for personalized recommendation purposes. Our study is inspired by this research to investigate whether enhancing movie ratings with movie embeddings leads to a more accurate representation of the relationship among users as well as items (movies). We start by utilizing a network of VAE to acquire movie embeddings and after that enhance the ratings of users using these embeddings. The combined representation is inputted into a secondary network of VAE, which is trained using a collaborative filtering model. This new network is referred to as Hybrid-VAE (Figure. 2). We will begin by applying a standard VAE for comparison, as shown in Figure 1. This paper aims to evaluate the execution, suitability, advantages, and additional costs of an H-VAE for collective filtering.

2. Dataset

MovieLens 20M dataset [4]: The dataset comprises 20,000,263 ratings for 27,278 movies provided by 138,493 users. The set of users is divided randomly into the test, training, and validation sets having ten thousands users in the test along with the validation sets, and 118,493 users in the training set. We exclude movies without IMDb information, resulting in an overall of 26,621 movies for analysis.

The ratings are converted into binary form, with a value of 1 assigned to movies rated higher than 3.5 by the user, and 0 to the rest. The threshold of 3.5 is selected to align with the reference [11]. Binarization provides an advanced way to categorize unseen movies as part of class 0 (implicit feedback) in a fair manner. The Variational Autoencoder (VAE) generates a probability distribution for each user's list of movies and function loss minimizes the discrepancy among the generated probability along with the binary user rating. For a movie with a binary rating of 0, the trained model is anticipated to produce a probability that is close to zero, and for a movie with a binary rating of one, the probability is anticipated to be close to one. When using the original ratings on a scale that ranges from 0-5, it is not possible to have a clear relationship among the network input (ratings) as well as the output (probability).

*Corresponding author: kukunooridineshkumar07@gmail.com

3. Evaluation Methodology

We implement 3-fold cross-validations (CVs) on the dataset to make sure the outcomes are reliable. The results presented in this paper are the average of three cross-validations. The standard deviation across coefficients of variation is approximately 10^-3. We utilize evaluation on the basis of rank- metrics such as Recall@50, NDCG@100, and Recall@20. Similarly to reference [11], we compared the movie's predicted rankings with their actual rankings for every "user in the test set. The predicted ranks have been determined by sorting the VAE network's final layer output, which provides a probability distribution for the movies. Recall@R considers all the items in the top R ranks similarly significant, whereas NDCG@R applies a monotonically boosting discount to highlight the significance of greater ranks over low ones. Formally, *w(r)* is termed the item at the rank *r, I* [] represents the function of indicator, and I_u represents the set of held-out items by which the user *u* clicked on. After that, the Recall@R user" *u* has been explained as follows:

$$Recall@R(u,w) = \frac{\sum_{r=1}^{R} I[() \in \quad I_u]}{min(M, |I_u|)} \tag{1}$$

To normalize Recall@R, we select the smaller value between R as well as the number of the items which have been clicked by the user u as the denominator. This normalization is applied when "ranking all essential items within the top R positions. The definition of truncated DCG@R) is provided below. NDCG@R is the DCG@R normalized version, ranging from 0 to 1, achieved by dividing it by the highest possible DCG@R where the highest-ranked held-out items are" all listed.

$$DCG@R(u.w) = \sum_{r=1}^{R} \frac{2^{I[w(r) \in I_u]} - 1}{log(r+1)} \tag{2}$$

Two categories of assessment schemes are employed:

- Eval 1: 10,000 users are used for testing, 10,000 users for validation, and 118,400 users for training in the original scheme. Next, for every test user across all 26,621 movies, the evaluation metrics NDCG as well as Recall have been estimated.
- Eval 2: Eval 1 differs in that every test user's click history is split into an 80/20 split. The movies in the 20% split are assigned a binary rating of 0, while the rating of the movies in the remaining 80percent split remains the same. Recall@k as well as NDCG@k are computed for every test user based on the 20 percent split. This scheme is more severe as well as realistic as it assesses the model's prediction on movies that the user has not seen before.

4. Movie Feature Extraction

The additional data from a secondary source is provided to the main user-rating information and inputted into the initial network of VAE as an item-embedding. Extraction of features involves utilizing 3 sets of information: genome tags, movie genres, and features derived from the movie summaries of IMDb.

A. Movie Genres

This included in the dataset of MovieLens-20M is utilized for this category. The dataset classifies movies into 19 genres, allowing each movie to be assigned to multiple genres. A feature vector is generated for each movie by creating a binary encoding that represents all genres.

B. Genome Tags

The MovieLens-20M dataset includes pre-defined genome tags that cover various aspects of the movie, such as plot characteristics and actors. These genome tags some examples include references to computer animation, a book, and the year 1920. The dataset contains 1128 tags, with each movie being assigned multiple tags along with a relevance score for every movie-tag pair. The paper considers the top 20 tags for every movie as well as creates these tags' binary vectors as the feature vector.

C. IMDb Summary

The Online Movie Database API2 is used to collect data for this category, from which 26,621 films' characteristics are derived. Each movie has an associated language, a certification, an IMDb rating, a viewer review score, and a plot. We employ the subsequent data for the feature extraction purpose:

- Language: Create a 1-hot encoding for the movie language by including "all the languages listed in dataset.
- Certification: This is a 1-hot encoding representing the certification assigned to the movie. Example: R, PG-13, etc.
- IMDb rating: The score provided is a continuous value between zero and ten. This is integrated as is, without any change.
- Plot: Analyzed the plot and extracted several features that describe the text various features.
  - ✓ LIWC (Linguistic Inquiry and Word Count) [7]: LIWC is a lexicon dictionary that links English words to the psychological, sociological, and linguistic processes across 64 categories. Each token in the plot text is tokenized and associated with a binary vector representing the corresponding LIWC category. The entire plot average vector is calculated by averaging the 64-D binary vectors for all the tokens in the plot.
  - ✓ VAD (Valence Arousal and Dominance) [14]: It is a lexicon dictionary that links words with 3-D scores. The plot text is broken down into tokens, and the scores for all words in the movie plot are then averaged.
  - ✓ Word2Vec [13]: The averagedWord2Vec vector of the plot text is utilized as a feature" for capturing the semantic distinctions and

likenesses among movies and their plots. A pre-trained Word2vec model with 300 dimensions is utilized.

Subsequently, all the previously listed characteristics are combined to create a 671-dimensional movie feature vector.

5. Implementation Details

A. *Standard-VAE*

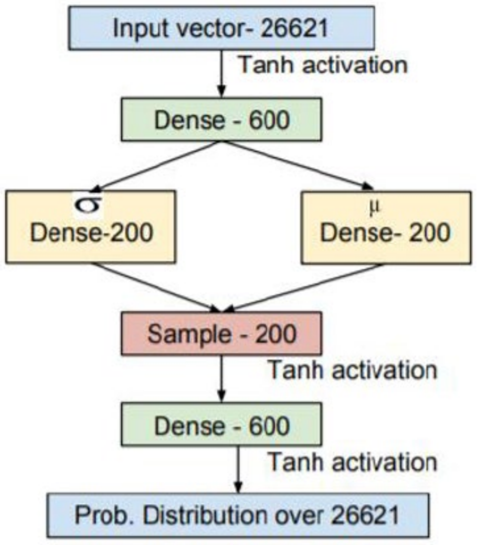

Fig. 1. Standard VAE architecture

The input for "the Standard-VAE examined in this research takes user ratings x_u. The encoder function g_ϕ () (3) in order to determine the standard deviations σ_u, and the mean, m_u of the K-dimensional latent representation. Each user's latent vector, z_u has been sampled by utilizing m_u, σ_u. The decoder function f_θ (4) has been then utilized for decoding the latent vector from the K-dimensions to a probability distribution π_u in the original N-dimension". The probability that user u will watch N movies is given by this distribution.

$$g_\phi(x_u) = m_u, \sigma_u \quad z_u \sim N(m_u, \sigma_u) \tag{3}$$

$$f_\theta(z_u) = \pi_u \tag{4}$$

This paper's standard VAE differs from the typical VAE by not having the final output as the reconstructed input. The output represents a probability distribution across the K-items. The model utilizes the ELBO as the objective function/loss, as specified in equation (5).

$$loss = \log p_\theta(x_m|z_m) + KL(q(z_m)||p(z_m|x_m)) \tag{5}$$

Where "x_m represents the movie feature vector while z_m represents the latent representation. The log-likelihood equation consists for a movie based on its z_m and the KL (Kullback-Leibler) divergence measure. The log-likelihood function" under consideration is provided as follows:

$$\log p_\theta(x_u|z_u) = \sum_i x_{ui} \log \sigma(f_{ui}) + (1 - x_{ui}) \log(1 - \sigma(f_{ui})) \tag{6}$$

where $\sigma(x) = 1/(1 + exp(-x))$ which has been taken over all the items i. The model's latent state, z_u, is used to compute the KL Divergence.

B. *Hybrid-VAE*

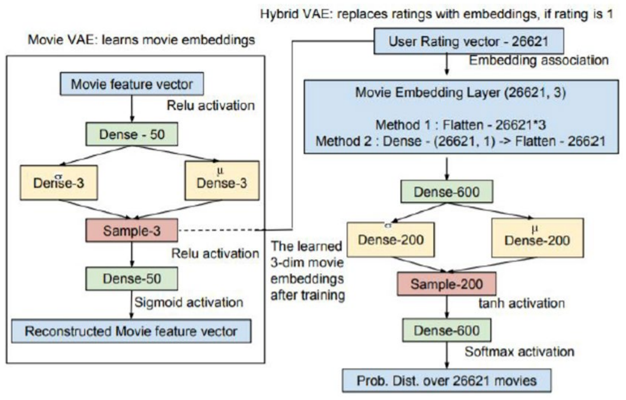
Hybrid VAE: replaces ratings with embeddings, if rating is 1
Movie VAE: learns movie embeddings
User Rating vector - 26621
Embedding association
Movie feature vector
Relu activation
Movie Embedding Layer (26621, 3)
Method 1 : Flatten - 26621*3
Method 2 : Dense - (26621, 1) -> Flatten - 26621
Dense - 50
σ
Dense-3
μ
Dense-3
Dense-600
Sample-3
Relu activation
The learned 3-dim movie embeddings after training
σ
Dense-200
μ
Dense-200
Dense-50
Sigmoid activation
Sample-200
tanh activation
Reconstructed Movie feature vector
Dense-600
Softmax activation
Prob. Dist. over 26621 movies

Fig. 2. Standard VAE architecture

Computational challenges arise when integrating an already high-dimensional user-rating input with a higher-dimensional feature vector for every movie. Movie feature vectors are encoded into a lower-dimensional latent space using a Movie-VAE (M-VAE). The movie features that were taken out of all 26,621 films are used to train the M-VAE. The dimension of the movie embeddings is equivalent to the latent space size. This paper investigates a three-dimensional movie embedding. The movie features are used in the Hybrid-VAE network after being encoded as embeddings.

With an extra layer that combines user ratings for every movie with movie embeddings, the H-VAE is equivalent to the Standard VAE. The M-VAE is the source of the embeddings. To enable embeddings and ratings to line up, movie indices are updated between the M-VAE and H-VAE. Zero-embedding is applied to movies with a user-click history of zero, which is shown as a three-dimensional vector made up of zeros. Figure 2 displays the architecture of the Hybrid-VAE. Given the embedding input x'_u, the user click history x_u, for every movie i, has been provided by,

$$x'_u = < e_1, e_2, e_3, \ldots, e_n >,$$
$$e_i = \begin{cases} Movie\_Embedding(i), \; if \; x_{ui} = 1, \\ Movie\_Embedding(0), \; if \; x_{ui} = 0 \end{cases} \tag{7}$$

The subsequent steps adhere to the same protocol as the Standard-VAE, except that during the encoding process, x'_u is used in place of x_u. Instead of taking into account the embedding input x'_u, the objective function continues to take into account the input user-click history xu. A 3D matrix of dimensions is the output of the embedding layer in the H-VAE (*batch size* x *num of movies* x *movie embedding dimension*). However, the input for the intermediate dense layer needs to be a 2-D vector. The embeddings can be added to the intermediate layer in one of two ways:

- Convert the output of "the 3D embedding layer to a 2D layer: In this instance, a vector with having a length equal to is fed into the intermediate dense layer. *number of movies* X *movie embedding dimension*
- Transform the 3-D embedding into a 2-D embedding by utilizing a Dense layer. The input to a intermediate dense layer in this scenario is a vector with a length equivalent to" number of movies.

The model is tested using the IMDb feature embeddings to identify the superior approach. The findings are recorded in Table 1. The initial method of converting the 3-D vector to a 2-D vector yields superior outcomes for Recall@k, whereas the alternative method produces marginally better results for NDCG@k. Approach 1 yields superior results due to the loss of information that takes place in Approach 2 when converting embeddings of size three to size one. Approach 1 was chosen for all subsequent tasks in the study due to its superior Recall@k performance and marginally lower NDCG@k results.

Table 1
Comparison between the approaches for handling the embedding layer

Measure	Approach 1	Approach 2
Eval 1 : NDCG@100	0.270	**0.271**
Eval 1 : Recall@20	**0.541**	0.539
Eval 1 : Recall@50	**0.573**	0.568
Eval 2 : NDCG@100	0.181	**0.183**
Eval 2 : Recall@20	**0.214**	0.211
Eval 2 : Recall@50	**0.377**	0.369

6. Visualizing Embeddings

Visualizations of user and movie embeddings learned from Variational Autoencoder networks help understand their functionality. The user embedding 200-dimensional latent representation from Standard VAE has been collected into 10 clusters by utilizing k-means clustering for visualization. t-Distributed Stochastic Neighbor Embedding (t-SNE) has been utilized to decrease the dimensionality from 200 - 2 for visualization after obtaining the cluster assignments. Users demonstrate specific patterns in the preferences of movies, which the network of VAE seeks to understand, as illustrated in Fig. 3.

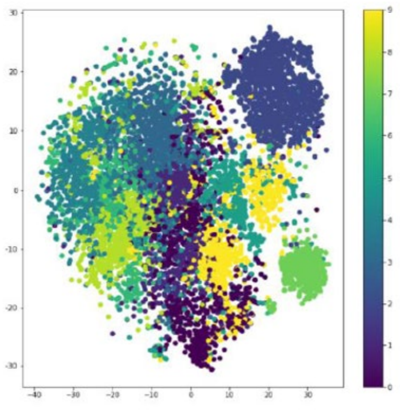

Fig. 3. User embeddings into 10 clusters

Movie embeddings are visualized in a similar manner to user embeddings, as shown in Figure 4. The data was generated by the M-VAE model utilizing genres as the only features and then grouped into the 18 clusters, each representing a different genre. The visualization displays significant clustering.

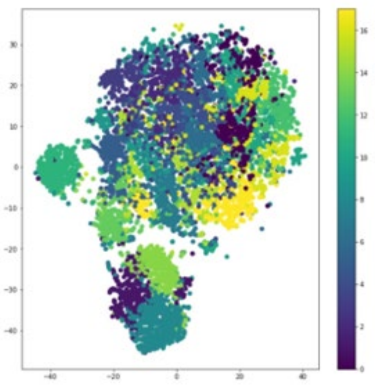

Fig. 4. Movie embedding into 18 clusters learned using genres

7. Results and Analysis

The model is executed on three feature sets as outlined in Section 4, "and the most efficient feature set is identified. The data presented in Table 2 demonstrates that the H-VAE performs better than the Standard-VAE, confirming the feature sets importance. IMDb summaries yield the highest scores for feature extraction, with genome tags following closely behind. Both of these feature sets surpass the feature set of a movie genre. Genres alone are not a potent contextual feature for characterizing movies in recommendation systems. Movies possess nuances which transcend genres, as evidenced by the inferior performance of the H-VAE with genre features like the embeddings which have been comparing with the baseline Standard-VAE.

Table 2
Performance of hybrid-VAE using different feature sets compared with standard-VAE

Measure	Standard-VAE	H-VAE (Random)	H-VAE (Genre)	H-VAE (Genome)	H-VAE (IMDb)
Eval 1 : NDCG@100	0.267	0.171	0.249	0.270	**0.271**
Eval 1 : Recall@20	0.534	0.297	0.501	0.537	**0.541**
Eval 1 : Recall@50	0.562	0.297	0.531	0.566	**0.572**
Eval 2 : NDCG@100	0.155	0.116	0.159	0.180	**0.181**
Eval 2 : Recall@20	0.208	0.127	0.213	0.208	**0.215**
Eval 2 : Recall@50	0.368	0.212	0.368	0.369	**0.377**

The features that have been extracted from the movies may have no effect, and the rise in scores could be entirely due to an additional layer. The model's accuracy was confirmed by training it with having random embeddings". The findings indicate that incorporating a random embedding layer which does not enhance the Standard-VAE model. This confirms the movie embedding utility and significance examined in this study.

It is possible which updates at the time of training cause significant changes in the embeddings from their initial values, makes the feature extraction of the movie meaningless. The IMDb feature embeddings visualized in Figure. 5 indicate that the training process has minimal impact on the embedding space. Therefore, it is logical to retain this information in the movie embedding format.

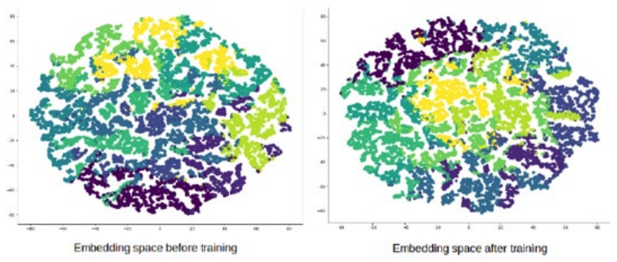

Fig. 5. Comparison of embedding spaces before and after training

The extracted features of IMDb are intricate and include details on the emotion as well as sentiment portrayed in a movie. These embeddings more effectively represent user preferences when combined with user rating data. The genome tags, while comprehensive, fail to convey this sentiment or emotion and are almost as effective as the features of IMDb. The genre feature provides a broad overview of a movie and does not effectively reflect the detailed preferences of users. The genre feature set appears to introduce more interference to the model rather than aiding in the prediction. The IMDb feature set enhances model accuracy by effectively capturing user preferences and providing a user-item interaction with more accurate representation.

8. Conclusion and Future Work

Although it can improve collaborative filtering performance, adding context information to the item set increases model complexity. On the other hand, the proposed method provides a flexible and easy-to-use way to integrate higher-dimensional context data into a VAE network. The outcomes demonstrate the significance and relevance of additional contextual information in automated recommendations. Some improvements are less than 0.01 in the results. However, given that the metrics have been averaged over ten thousand users, even a smaller increase is significant.

In future research, a “statistical significance t-test could be conducted to determine if the performance improvement is important or simply a product of randomness or noise. It is important to conduct qualitative analysis on the acquired movie embeddings to determine if comparable movies share the same embeddings” through the use of a suitable distance metric. Optimizing hyper-parameters shows the potential to enhance the outcome's reliability. We make the codebase available as a public GitHub repository.

References

[1] Mohammed Al-Qizwini, Iman Barjasteh, Hothaifa Al-Qassab, and Hayder Radha. 2017. A deep learning algorithm for autonomous driving using GoogLeNet. In Intelligent Vehicles Symposium (IV), 2017 IEEE. IEEE, 89–96.

[2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. IEEE transactions on pattern analysis and machine intelligence 35, 8 (2013), 1798–1828.

[3] Ronan Collobert, Jason Weston, Leon´ Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. Journal of Machine Learning Research 12, Aug (2011), 2493– 2537.

[4] F. Maxwell Harper and Joseph A. Konstan. 2016. The movie lens datasets: History and context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4 (2016), 19.

[5] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In Proceedings of the 26th Interna- tional Conference on World Wide Web. International World Wide Web Conferences Steering Committee, 173–182.

[6] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, and others. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal Processing Magazine 29, 6 (2012), 82–97.

[7] Ryan L. Boyd James W. Pennebaker, Roger J. Booth and Martha E. Francis. 2015. Linguistic Inquiry and Word Count. Pennebaker Conglomerates, Austin, Texas (2015).

[8] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. Computer 42, 8, 2009.

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems. 1097–1105.

[10] Wonsung Lee, Kyungwoo Song, and Il-Chul Moon. 2017. Augmented Varia- tional Autoencoders for Collaborative Filtering with Auxiliary Information. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. ACM, 1139–1148.

[11] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering, 2018.

[12] Andriy Mnih and Ruslan R Salakhutdinov. 2008. Probabilistic matrix factorization. In Advances in neural information processing systems. 1257–1264.

[13] Kai Chen Greg S. Corrado Tomas Mikolov, Ilya Sutskever and Jeff Dean. 2013. Efficient estimation of word representations in vector space, 2013.

[14] Amy Beth Warriner, Victor Kuperman, and Marc Brysbaert. 2013. Norms of valence, arousal, and dominance for 13,915 English lemmas. Behavior research methods 45, 4 (2013), 1191–1207.

[15] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In Proceedings of the Ninth ACM International Conference on Web Search and Data Mining. ACM, 153–162.

[16] Yin Zheng, Bangsheng Tang, Wenkui Ding, and Hanning Zhou. 2016. A neural autoregressive approach to collaborative filtering, 2016.