ISBN: 978-981-18-7950-0
2023 the 13th International Workshop on Computer Science and Engineering (WCSE 2023)
doi: 10.18178/wcse.2023.06.004

Generating GAN-based fake embedding samples for Recommender Systems

Jesús Bobadilla and Abraham Gutiérrez [+]

Technical University of Madrid, ETSISI, Ctra. de Valencia Km. 7, Madrid, Spain

Abstract. This paper proposes the use of embedding-based architectures to generate collaborative filtering synthetic datasets in the Recommender Systems area; these datasets are useful both for research and commercial purposes. Since raw collaborative filtering data is extraordinarily sparse, it makes sense to embed them before generative processing. This compression stage is expected to improve GAN performance and accuracy, but it also involves measuring the quality of the results before and after the necessary decompression stage. In between embedding and unembedding, it is convenient to test the 'intermediate' sample representations to ensure that they keep some convenient distributions, such as the ratings proportions: one to five stars, and their expected latent space layouts. Experiments have been run using two representative datasets and two generative models: regular and Wasserstein ones; results show an appropriate distribution of the generated fake profiles, opening the door to future works where the short, dense, and continuous generated embeddings will be translated to the long, sparse, and discrete fake samples that will fill the generated synthetic datasets.

Keywords: synthetic datasets, generative adversarial networks, recommender systems, collaborative filtering, embedding

1. Introduction

Recommender Systems (RS) is a relevant IA field where services and products can be personalized. It usually operates by making explicit recommendations of products and services. Relevant companies that use RS are Amazon, Spotify, YouTube Music, TripAdvisor, and TikTok. There are several approaches to filter predictions and then to obtain recommendations: content-based filtering [1], where recommended products are those whose descriptions or images are similar to the ones the active user has consumed; social filtering [2], based on trusted and followed users; context-aware filtering, that relies on geographical information such as GPS coordinates, demographic filtering [3], where recommendations are made attending to the preferences of users with similar demographic features to the active one (age, sex, country, etc.); Collaborative Filtering (CF) [4], based on the preferences of similar users to the active one, and ensembles of the preceding approaches. The most accurate and relevant filtering approach is CF, where a dataset of consumed items (products or services) is available, usually including samples following the format <user, item, rating>.

CF was historically addressed using the KNN algorithm [4]: it is simple and directly implements the CF concept; unfortunately, it is memory-based, its accuracy is not good enough, and its performance is low. Model-based Matrix Factorization (MF) [5] has dominated the RS CF scenario, due to its high performance and accuracy. MF codes the user behaviours in hidden latent factors (same for items), and then combines both the user and the item hidden factors by means of a dot product that returns predictions. Currently, neural architectures improve the MF results, as they can better catch the complex nonlinear patterns of the RS CF data. Remarkable neural CF models are DeepMF and NCF. DeepMF [6] directly implements the MF concept; it incorporates an embedding layer to code users, and another embedding layer to code items. These embedding layers make the MF hidden latent factors task: to compress user and items in the same latent space; same as MF, the DeepMF model uses a dot layer to combine the user and the item embeddings. Since the dot product does not incorporate a non-linear activation function, the NCF [7] architecture improves DeepMF by replacing

[+] Corresponding author.
E-mail address: abraham.gutierrez@upm.es

the dot product with a Multilayer Perceptron (MLP), which can combine the latent space embedding activation maps in a richer and accurate way.

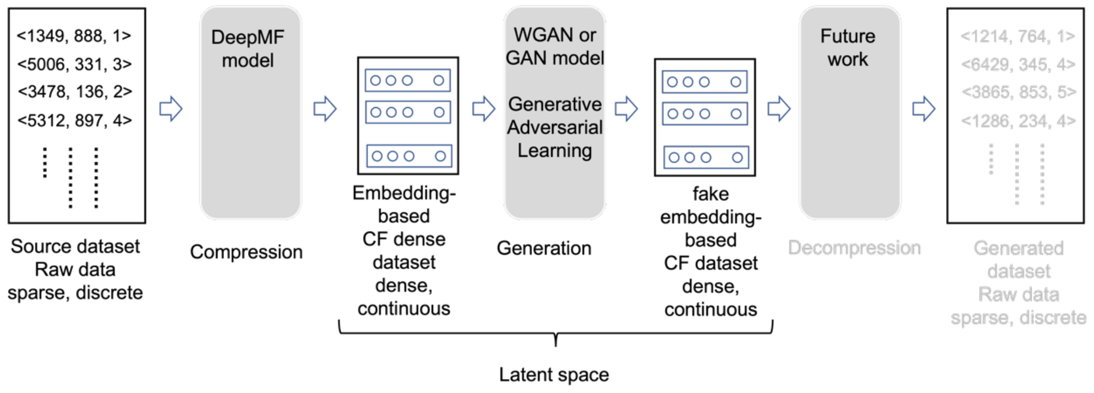
The RS CF field has a growing need of suitable datasets for research purposes, but existing datasets are restricted to fixed sizes, number of users, number of items, and internal distributions. Although there are some 'families' of datasets, such as MovieLens 100K, 1M, 10M, etc., that can help to perform comparative tasks in research, there is a lack of customizable data to make experiments where the parameter values can range in intervals: e.g., where the number of items grows from 1000 to 6000, step 500 (the same for the number of users). Customizable datasets have been generated using statistical methods, but they cannot conveniently catch the CF data patterns. Currently, generative deep learning is being used in the CF field: Generative Adversarial Networks (GAN) [8] can create fake samples from a dataset of real ones; the most known application of GANs is to generate fake faces from a set of real ones, but this model can also be applied to the CF field, by creating fake user or item profiles. The CFGAN [9] uses augmentation to GAN generatively enrich a dataset of purchase profiles. The RecGAN [10] incorporates a Recurrent Neural Network (RNN) to process temporal patterns, whereas the reinforcement learning is added to a regular GAN in the DCFGAN architecture [11] that processes session logs. A Conditional GAN (CGAN) has been used to make a conditional creation of samples [12]. Unbalanced datasets are processed using a Wasserstein GAN (WGAN), and negative sample information is incorporated into the CF processing to improve accuracy in the IPGAN model [13]. Finally, the NCGAN [14] uses a GAN that is fed with the intermediate values provided by a neural network.

GAN models usually take raw data as a source and, also, generate raw data as a target: e.g., MNIST images as a source to generate fake MNIST images as a target. However, CF data are extraordinarily sparse (98% sparsity is a typical value); it means that users, on average, consume a tiny proportion of the hundreds or thousands of available products or services. In this scenario, it seems more rational to work with compressed data instead of the sparse one. Using neural networks, we can use embedding layers to code both users and items in the latent space, converting them to short size vectors of real numbers; the hypothesis of this paper is that GAN models will perform better working on short, dense, and continuous latent space vectors (embeddings) that acting on very large, sparse, and discrete non compressed vectors: e.g., a vector containing the votes rated for a user in a dataset where 3000 items exists. This vector will have size 3000, where approximately the 98% of the elements contain the number zero (meaning non voted item). The objectives of this research are a) to obtain embedding representations of representative CF datasets, b) to create fake embedding versions of them, and c) to test and compare both the real and the generated embeddings. Generating target (synthetic) raw datasets, not the embedding representations, is beyond the scope of this paper and is proposed as future work, partially done in [15]. The rest of the paper is structured as follows: Section 2 details the proposed method; Section 3 introduces the experimental design, shows the obtained results, and explain them; Section 4 contains the conclusions and future work; and finally, Section 5 lists current references related to this research.

2. Method

We basically combine two existing neural networks models to generate both embedding real samples and embedding fake samples; then, both the real and fake embedding samples are tested and compared to determine their adequacy to be taken as a source to generate raw and sparse datasets. Figure 1 shows the main architecture; its black portion is implemented in the paper, whereas the grey section is not. To obtain the real user and item embeddings from the real raw samples, we choose the DeepMF model (left side in Figure 1), since it directly implements the functionality. Once the DeepMF is trained, we can feedforward all the user IDs and the item IDs in the dataset to obtain their corresponding user embedding vectors and item embedding vectors ("Embedding-based CF dense dataset" in Figure 1). Our second neural network stage feeds a GAN or WGAN [16] architecture with the embedding-based CF dense dataset to train the model. Once the GAN is trained, we can generate as many fake samples as desired to obtain the 'Fake embedding-based CF dataset' (centre of Fig. 1). In this paper we test and compare both the real and the fake embedding sets to determine if the second one has suitable layouts and distributions to address the synthetic dataset generation.

In the DeepMF model, both the user and the item embedding layers are represented; they, respectively, code the input user IDs and item IDs. In the training stage, the dot layer obtains predictions that are compared to real ratings to train the model. In the feedforward stage, we provide each set of user IDs and item IDs to get their activation maps (embedding representations of real samples). The GAN or WGAN architecture is made up of its discriminator (or critic) model and its generator model. The generator is fed with random noise vectors from stochastic Gaussian distributions and provides fake samples to the discriminator. The discriminator (or critic in Wasserstein GANs) makes a binary classification 'fake vs. real', taking some batches of the real embeddings (output of the feedforward DeepMF model) and some other batches of the generator output (fake embeddings). When this adversarial training has finished, the generator is able to create as many fake embedding samples as random noise vectors are provided.



<1349, 888, 1>
<5006, 331, 3>
<3478, 136, 2>
<5312, 897, 4>
Source dataset Raw data sparse, discrete
DeepMF model
Compression
Embedding-based CF dense dataset dense, continuous
WGAN or GAN model
Generative Adversarial Learning
Generation
fake embedding-based CF dataset dense, continuous
Future work
Decompression
<1214, 764, 1>
<6429, 345, 4>
<3865, 853, 5>
<1286, 234, 4>
Generated dataset Raw data sparse, discrete
Latent space

Fig. 1: Main architecture to generate both the real and the fake embeddings.

The above architecture, applied to the field of CF, has been formalized in its training stages. The result is provided below. Equations from (1) to (3) define the CF data; Equations (4) to (7) formalize the DeepMF training stage, and, finally, Equations (8) and (9) define the discriminator, generator, and loss function of the Wasserstein GAN.

- Let U be the set of a CF users, I the set of items available, and V the range of allowed votes, usually from 1 to 5 stars (1)
- Let S be the set of the CF samples; $S = \{<u,i,v>_1, <u,i,v>_2, \ldots, <u,i,v>_N\}$; where each $u \in \{1, \ldots, |U|\}$, each $i \in \{1, \ldots, |I|\}$, and each $v \in \{1, \ldots, |V|\}$ (2)
- Let E be the size of the user embedding and the item embedding. (3)
- Let $f(u) = [e_0^u, e_1^u, \ldots, e_E^u]$, the embedding of the users, where $u \in \{1, \ldots, |U|\}$ (4)
- Let $f'(i) = [e_0^i, e_1^i, \ldots, e_E^i]$, the embedding of the items, where $i \in \{1, \ldots, |I|\}$. (5)
- $\hat{y}_j = f(u) \cdot f'(i) = [e_0^u, e_1^u, \ldots, e_E^u] \cdot [e_0^i, e_1^i, \ldots, e_E^i]$, where $\hat{y}_j$ are the rating predictions (6)
- The output error used in the DeepMF neural network is $\varepsilon = (y_j - \hat{y}_j)^2/2$, where the neural weights are iteratively improved from the δ_j values: $\Delta w_{ji} = \alpha y_j f'(Net_i) \sum_k w_{ik}\delta_k$, when k is a hidden layer, and $\Delta w_{ji} = \alpha y_j f'(Net_i)(y_k - \hat{y}_k)^2/2$, if k is the output layer. i, j, and k are successive sequential layers (7)
- Let f^D be the GAN discriminator D model, f^G be the GAN generator G model, and f^{GD} be the GAN optimization function (8)
- $f^{GD} = Min_G Max_D f(D,G) = E_R[f_w(R)] + E_z[f_w(G(z))]$, (9)

where E_R is the expected value for real samples, z is the random noise that feeds the generator G, and E_z is the expected value for the generated fake profiles G(z). f_w is the Wasserstein function based in the earth mover's distance; it satisfies the 1 – Lipschitz constraint: $|f(x_1) - f(x_2)| \leq |x_1 - x_2|, \forall x_1, x_2$. Note that R refers to embedding dataset of real samples.

3. Experiments and Results

Experiments have been carried out on two representative open datasets in the CF area: the MovieLens 1M [17], and a subset of Netflix (Netflix*) [5]. Table 1 shows their main parameter values. We have obtained 12 "Real embedding-based CF datasets" (Fig. 1), by combining both source datasets with two GAN models, regular GAN, and Wasserstein GAN, and then setting the embedding size to 2 and 3. Table 2 summarizes the generation process. To perform a comparative test of the Table 2 datasets containing real and fake embedding samples, their layouts are obtained in two and three dimensions and the rating distributions are shown. For the sake of brevity, this section only shows and explains the MovieLens source dataset results. The Netflix* ones follow the same patterns, and they are accessible through [18].

Table 1: Main parameter values of the datasets used in the experiments.

Dataset	# users	# items	# ratings	scores	sparsity
MovieLens 1M	6,040	3,706	911,031	1 to 5	95,94
Netflix*	23,012	1,750	535,421	1 to 5	98,68

Table 2: Real and synthetic embedding datasets involved in the experiments.

	DeepMF model		Generative model			
			GAN		Wasserstein GAN	
	Embedding size		Embedding size		Embedding size	
Dataset	2	3	2	3	2	3
MovieLens	ML_real_2D	ML_real_3D	ML_fake_2D	ML_fake_3D	ML_W_fake_2D	ML_W_fake_3D
Netflix*	NF_real_2D	NF_real_3D	NF_fake_2D	NF_fake_3D	NF_W_fake_2D	NF_W_fake_3D

Fig. 2 shows the Table 2 results using its labels (‘ML_real_2D’, etc.). Colors represent ratings (from 1 to 5 stars); hot colors code relevant votes, whereas cold colors represent nonrelevant votes. The top row in Fig. 2 shows the 2D and 3D representations of the real embeddings (of sizes 2 and 3, respectively), which we will use as baseline to test the GAN generated fake samples (center row in Fig. 1) and the Wasserstein GAN generated fake samples (bottom row in Fig. 1). Overall, fake samples (both 2D and 3D) conveniently mimic the rating separation existing in the real samples. As expected, fake samples emphasize the Normal distribution due to the stochastic Gaussian distribution used to create the random noise vectors that feed the GAN and the WGAN generators. The fake ratings are also better grouped in the latent space, and it suggest that the generated raw datasets will provide more accurate predictions. The Wasserstein-GAN and the GAN results are similar; it means that the regular GAN does not suffer from the mode collapse in this scenario; it is relevant since it suggests that the use of Wasserstein GAN could not be necessary using the proposed architecture.

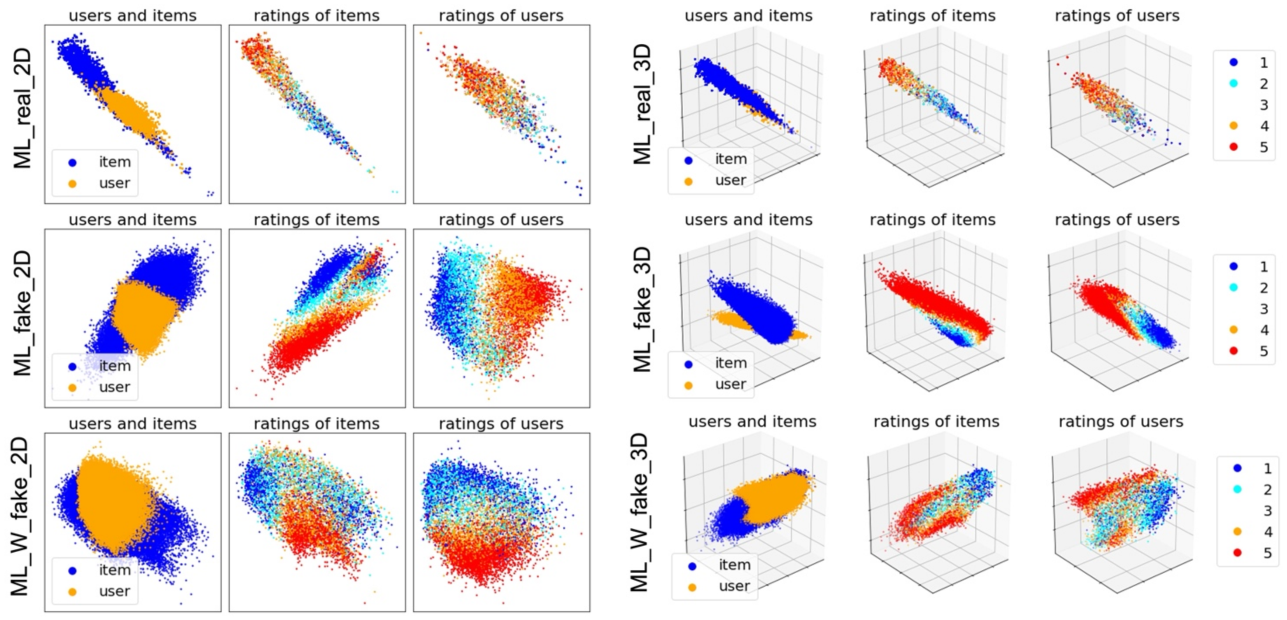


Fig. 2: Embedding layouts obtained from MovieLens 1M: left graphs using embedding vectors of size 2, and right graphs using embedding vectors of size 3. The upper row shows real samples; the center row shows fake samples obtained using a GAN model; the lower row shows fake samples from a Wasserstein GAN model. Colors code rating values.

Fig. 3 shows a comparison of the rating distribution (1 to 5 stars) obtained from the DeepMF embeddings (real embeddings), labelled 'ML', and the ones obtained in the GAN and in the Wasserstein GAN generative processes (fake samples), labelled, respectively, 'gan' and ‘wgan’. The results show that all of them follow a

Gaussian distribution, where the real embedding one is more scored to relevant votes, whereas both fake distributions are centered on the mean rating. However, the fake rating distributions adequately fit the real one.

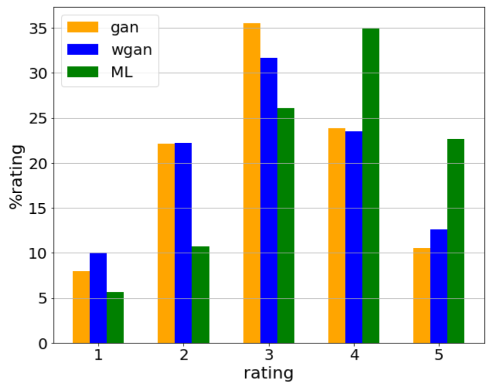


Fig. 3: Distribution of ratings (from 1 to 5 stars) from the embedding vectors of real samples (ML), from the embedding vectors of fake samples obtained using a GAN (gan) and using a Wasserstein GAN (wgan).

4. Conclusions

Generative learning can be applied to the creation of collaborative filtering synthetic datasets in the Recommender Systems field; this will help both the research community and the companies. The generative adversarial models usually take raw data as input and generate raw data as target. However, the source collaborative filtering datasets are extraordinarily large and sparse, and it suggests making use of intermediate compressed representations to feed the GAN. This paper uses a DeepMF model to code user and item IDs in embedding values, and after that generatively obtains fake samples with the same format: embedding vectors. Embedding sizes 2 and 3 have been chosen to draw and compare bi-dimensional and tri-dimensional graphs. The results obtained using a Wasserstein GAN and a regular GAN have also been compared; they show that the set of fake samples follows similar distributions and layouts as those obtained with real embedding samples. Both the 2D and 3D graphs show adequate groups of ratings. The fake results better spread the samples in a Gaussian distribution, that is, the expected behavior due to the stochastic Gaussian random noise existing in the vectors that feed the GAN generator model. It has also been seen that the GAN results do not show any significant mode collapse, and, finally, that the fake ratings distributions adequately mimic the real one. Overall, the intermediate proposed embedding representation seems to be accurate enough to propose, as future work, the generation of collaborative filtering synthetic raw datasets from the obtained fake embedding samples. These synthetic datasets can contain a fixed number of users and items, or more challenging: a parameterized behavior, setting the number of users, number of items, standard deviation of the rating distributions, and the dataset size.

5. Acknowledgements

This work was partially supported by the Ministerio de Ciencia e Innovación of Spain under the project PID2019-106493RB-I00 (DL-CEMG) and the Comunidad de Madrid under the Convenio Plurianual with the Universidad Politécnica de Madrid in the actuation line of the Programa de Excelencia para el Profesorado Universitario.

6. References

[1] Deldjoo, Y., Schedl, M., Cremonesi, P., Pasi, G. Recommender systems leveraging multimedia content. In: *ACM Computing Surveys (CSUR),* 2020, vol. 53(5), pp. 1-38.

[2] B. Zhu, F. Ortega, J. Bobadilla, A. Gutiérrez. Assigning reliability values to recommendations using matrix factorization. In: *Journal of computational science,* 2018, vol. *26*, pp. 165-177.

[3] Bobadilla J, González-Prieto A, Ortega F, Lara-Cabrera R. Deep learning feature selection to unhide demographic recommender systems factors. In: *Neural Computing and Applications*, 2021, vol. *33(12),* pp. 7291-7308.

[4] Zhu, B., Hurtado, R., Bobadilla, J., Ortega, F. An efficient recommender system method based on the numerical relevances and the non-numerical structures of the ratings. In: *IEEE Access*, 2018, vol. 6, pp. 49935-49954.

[5] F. Ortega, B. Zhu, J. Bobadilla, A. Hernando. CF4J: Collaborative filtering for Java. In: *Knowledge-Based*

Systems, 2018, vol. 152, pp. 94-99.

[6] Hong-Jian Xue, Xinyu Dai, Jianbing, Zhang, Shujian Huang, Jiajun Chen. Deep Matrix Factorization Models for Recommender Systems. In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, 2017, pp. 3203-3209.

[7] Xiangnan He, Lizi Liao, Hanwang Zhang. Neural Collaborative Filtering. In: *International World Wide Web Conference Committee (IW3C2)*, 2017.

[8] Min Gao, Junwei Zhang, Junliang Yu, Jundong Li, Junhao Wen, Qingyu Xiong. Recommender systems based on generative adversarial networks: A problem-driven perspective. In: *Information Sciences*, 2021, vol. 546, pp. 1166-1185.

[9] Chae, J. Kang, S. Kim, J. Lee. 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 2018*, 2018, pp. 137–146.

[10] H. Bharadhwaj, H. Park, B.Y. Lim. Recgan: recurrent generative adversarial networks for recommendation systems. In: *Proceedings of the 12th ACM, Conference on Recommender Systems, RecSys 2018*, 2018, pp. 372–376.

[11] Zhao, Jianli; Li, Hao; Qu, Lijun; Zhang, Qinzhi; Sun, Qiuxia; Huo, Huan; Gong, Maoguo. DCFGAN: An adversarial deep reinforcement learning framework with improved negative sampling for session-based recommender systems. In: *Information sciences*, 2022, vol. 596, pp. 222-235.

[12] Wen, Jing; Zhu, Xi-Ran; Wang, Chang-Dong; Tian, Zhihong. A framework for personalized recommendation with conditional generative adversarial networks. In: *Knowledge and information systems*, 2022, vol. 64 (10), pp. 2637-2660.

[13] G. Guo, H. Zhou, B. Chen, et al. IPGAN: Generating informative item pairs by adversarial sampling. In: *IEEE Transactions on Neural Networks and Learning Systems*, 2022, vol. 33(2), pp. 694-706.

[14] Sun, Jinyang; Liu, Baisong; Ren, Hao; Huang, Weiming. NCGAN: A neural adversarial collaborative filtering for recommender system. In: *Journal of intelligent & fuzzy systems*, 2022, vol. 42 (4), pp. 2915-2923.

[15] J. Bobadilla, A. Gutiérrez, R. Yera, L. Martínez. Creating Synthetic Datasets for Collaborative Filtering Recommender Systems using Generative Adversarial Networks, 2023, arXiv:2023.01297v1

[16] Wang, Qingxian; Huang, Qing; Ma, Kangkang; Zhang, Xuerui. A Recommender System Based on Model Regularization Wasserstein Generative Adversarial Network. In: *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2021, pp. 2043-2048.

[17] F.M. Harper, J.A. Konstan. The movielens datasets: History and context. In: *ACM Transactions on Interactive Intelligent Systems*, 2015, vol. 5(4), pp. 1-19.

[18] https://drive.upm.es/s/tkHkuHQpyrYDIdk