Exploration of User Privacy Preservation via CTGAN Data Synthesis for Deep Recommenders

Savannah V. McCoy
Department of Computer Science
Stanford University
savmccoy@stanford.edu

Abstract

Modern-day recommender systems use vast amounts of user data. As more ethical concerns emerge about the use of user data by tech companies, it is ever more important to protect user's privacy. This paper describes an experiment that analyzes the effect of using synthetically generated data to train deep learning-based recommender systems. Final results show that there is a 41.18% decrease in Factorized Top-100 Categorical Accuracy when CTGAN-generated synthetic data is used for training in place of real user data (baseline), however a 75/25% split of Real/Synthetic data produces only a 1.33% drop in accuracy from the baseline while maintaining similar loss and Root Mean Square Error (RMSE) metrics.

1 Introduction

One way to try to preserve user privacy is to minimize the amount of data collected and stored by only collecting data that is absolutely necessary for company use. In this project, I analyze the effects of using synthetically generated data to train deep learning-based recommender systems as a proposed privacy preservation technique to approximate how much authentic data is necessary to train useful models. I focus on recommender models in particular because recommenders are a popular machine learning application that adds value to a platform and require vast amounts of user data to be effective. This paper covers a a modern, practical technique for the collection and synthesis of tabular data using using Generative Adversarial Networks (GANs). With this technique, companies could periodically collect small random subsets of real user data and use them to generate large realistic synthetic data sets that can then be used to improve the accuracy of their models, thereby reducing the collection of large amounts of authentic user data and the risk to user privacy.

2 Related Works

Recommendation systems can be broadly categorized into collaborative vs. content based filtering [7, 8, 9]. Collaborative filtering provides recommendations based on the rating predictions for all users collectively, while content based filtering provides recommendations based specifically on the content of the item and the preferences of a given user. In this project, I focus on collabortive filtering as the primary method. In the recommender model detailed later in this paper, collaborative filtering is used to to generate user and product embeddings. These embeddings are then used to obtain recommendations. Figure 1 displays an example architecture of a deep recommder model that makes use of Neural Collaborative Filtering layers.

GANs are well known for their success in generating realistic images, however they can be used to generate realistic tabular data as well. In TGAN: Synthesizing Tabular Data using Generative Adversarial Networks [13] they raise several challenges for generating tabular data. The various data types, data distributions and sparse one-hot-enoding make analyzing and mimicing tabular datasets quite difficult. To solve this issue, in Modeling Tabular Data using Conditional GAN (CTGAN) [12] they apply mode-specific normalization to overcome the non-Gaussian and multimodal distribution. Finally, conditional generator and training-by-sampling are used to handle imbalanced discrete columns. Figure 2 diaplays a workflow diagram of using CTGAN to generate synthesized (ouput) train and target datasets.

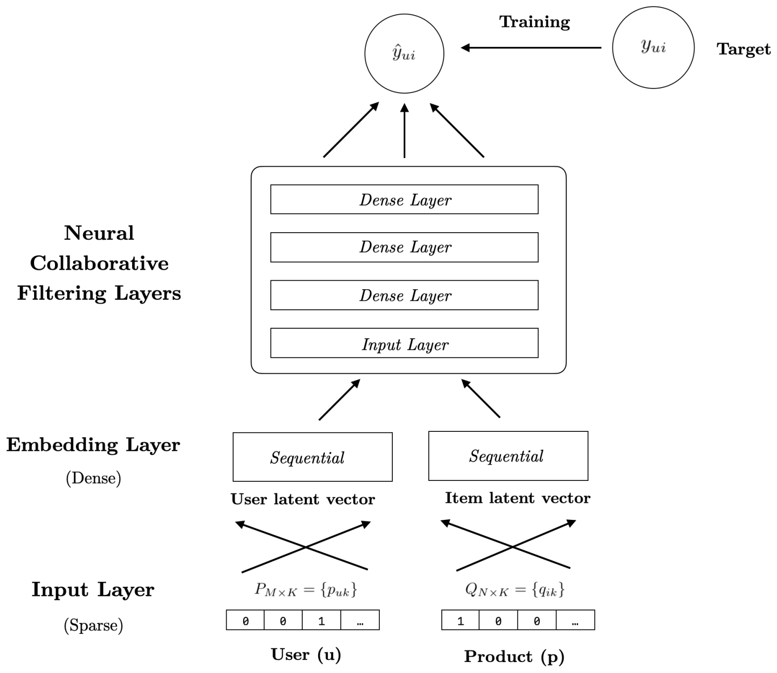


Figure 1: Example Recommender Architecture w/ Neural Collaborative Filtering. Adapted from Neural Collaborative Filtering by Abhishek Sharma

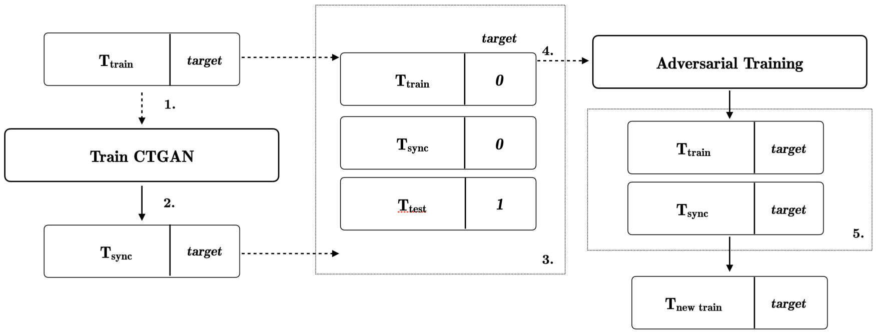


Figure 2: Design and workflow of CTGAN implementation. Adapted from Tabular GANs for uneven distribution by Insaf Ashrapov

3 Data

This project makes use of user-product review from the Amazon Product Reviews dataset by Julian McAuley available at [1] The complete dataset contains 82.83 million unique reviews, from approximately 20 million users as well as product metadata including: item-to-item relationships, timestamps, helpfulness votes, category, and salesRank. There are multiple small subsets of data available for experimentation such as K-cores subsets that have been reduced to extract the k-core, such that each of the remaining users and items have k reviews each. Specifically, I chose the videogame subset of 5-core data because it provides ample reviews to use train both a GAN and Recommender. Notably, prior to synthesizing data with a GAN or training the recommenders, I filtered to data to extract relevant features from this dataset including overall rating, verified status, product price, review date, and number of votes. I included only categorical and numeric features to reduce overall dataset size (450,000+ rows) and total model training time.

4 Method

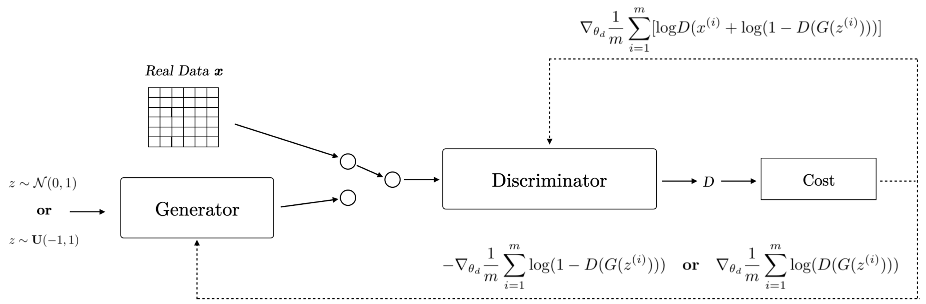

$\nabla_{\theta_d}\frac{1}{m}\sum_{i=1}^{m}[\log D(x^{(i)} + \log(1 - D(G(z^{(i)})))]$
Real Data $\boldsymbol{x}$
$z \sim \mathcal{N}(0,1)$
or
$z \sim \mathbf{U}(-1,1)$
Generator
Discriminator
D
Cost
$-\nabla_{\theta_d}\frac{1}{m}\sum_{i=1}^{m}\log(1 - D(G(z^{(i)})))$ **or** $\nabla_{\theta_d}\frac{1}{m}\sum_{i=1}^{m}\log(D(G(z^{(i)})))$

Figure 3: GAN training pipeline. Adapted from GANs for tabular data by Jonathan Hui

After transforming and filtering the review dataset the next step was to generate synthetic data using a GAN. A Generative Adversarial Network is machine learning framework in which two neural networks contest with each other to generate new data with the same distribution as the training data. The formula below details the backpropagation method of training GANs. Figure 3 displays the GAN training pipeline. For this project, I focused on data generation using a Conditional Tabular GAN or, in other words, CTGAN. The CTGAN methodology was implemented in code by Insaf Ashrapov [18] and converted to a python package [16] which I adapted for this use case. I generated over 100,000 synthetic data points using the `tabgan` pythob library by iterating over random samples of 3% of the real user reviews data which totaled to around 14,000 datapoints each itertion. 3% was arbitrily chosen as it was a small percentage of the total dataset but still provided enough data for effective learning. Notably, I augmented graphical data in which the data points have relationships between them. Hence, I only used categorical features and marked all features as categorical to preserve relationships between elements and emulate realistic relationships when training the CTGAN.

GAN Back Propogation:

$$\nabla_{\theta_d}\frac{1}{m}\sum_{i=1}^{m}[\log D(x^{(i)} + \log(1 - D(G(z^{(i)})))]$$

Once I gathered enough synthetic datapoints, I used them to create hybrid datasets from real and generated data, then used splits to use to train several recommender models for various. I made use of the Tensorflow Recommender System (TFRS) framework to train 5 models on various splits of real and synthetic data. The recommenders had both rating prediction and retrieval tasks each with 50% weight. An Adagrad optimier with learning rate of 0.1 was also used. The formula below details Adagrad optimization. Each model took nearly 3 hours to train over 10 epochs on the Google Colab platform. I trained using 100% real data model to use as the baseline for comparison. 100% synthetic, a 50% /50% split, a 75% /25% split and a 90% /10% split. For the real data model and "hybrid" models trained using synthetic data, I trained all models on the same number of epochs and using an fixed Input length of 100,000 rows.

Adagrad optimization:

$$
\begin{aligned}
&g_0 = 0 \\
&g_1 \leftarrow g_t + \nabla_\theta \mathcal{L}(\theta)^2 \\
&\theta_j \leftarrow \theta_j - \epsilon \frac{\nabla_\theta \mathcal{L}}{\sqrt{g_{t+1} + 1e^{-5}}}
\end{aligned}
$$

5 Results & Analysis

Finally, the results of my experiment are displayed in Table 1. This table list the test metrics of each of the models trained. I compared the metrics of the models and performing qualitative and quantitative evaluation of the results to either support or reject my theory: I expected model performance to worsen with use of synthetic data but perform better than training with a smaller subset of real data. As expected, there was a descrease in performance between the real and fully synthetic-trained models, however the model trained with 25% synthetic data has similar performance to the model trained only on real data, which is an amazing insight. Also, Figure 4 displays two plots of training accuracy over 10 epochs, one for a model trained with 50/50 split data and one for a model trained with 100% synthetic data. The top 100 categorical accuracy predictions were pretty high on both models during training, however the fully synthetic-trained model leveled off in accuracy at around 35% while the model using a 50/50 split leveled off at around 50% during training.

	RMSE	Top 100 Test ACC	Top 10 Test ACC	Total Loss
100% Real	1.7953	0.1481	0.0770	57321.2650
100% Synthetic	1.1509	0.1049	0.0564	46590.9616
50/50% R/S Split	1.2435	0.1314	0.0719	52088.4180
75/25% R/S Split	1.5725	0.1467	0.0793	46480.0462
90/10% R/S Split	2.3282	0.1426	0.0772	76561.0299

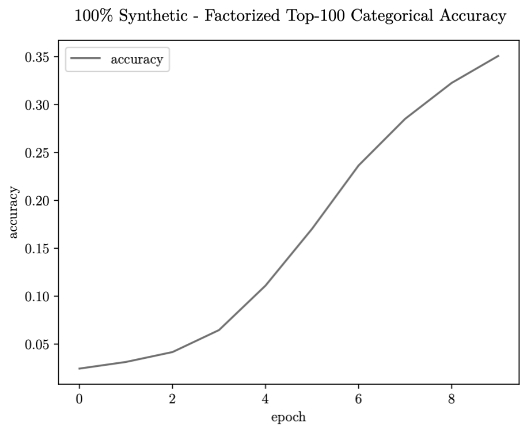
Table 1: Results table displaying Root Mean Square Error (RMSE), Factorized Top-k Test Accuracy (k=10, k=100), and total loss metrics for each model.

To evaluate the performance of the rating rating prediction model, I assessed values for Loss and RMSE. The following lists the formula for Root Mean Square Error (RMSE), a standard way to measure the error of a model in predicting quantitative data:

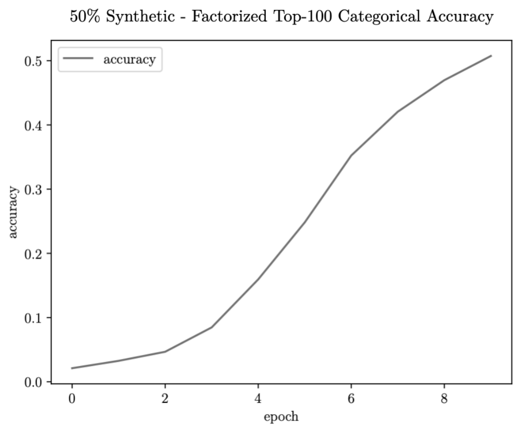
$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \hat{x}_i)^2}{N}}$$

To evaluate the performance of the retreival task, I assessed the values for categorical accuracy. Factorized Categorical top K accuracy is calculated for k = 1, k = 5, k = 10, k = 50, and k = 100. Although only k = 10 and k = 100 are provided in the results table for simplicity. The following lists the general formula used to find Categorical Accuracy:

$$ACC = \frac{\text{true positives} + \text{true negative}}{\text{total observations}}$$



(a)



(b)

Figure 4: Factorized Top-k Categorical training accuracy over 10 epochs for 50/50 and 0/100 splits of Real/Synthetic

6 Conclusion

In this project, I trained 5 recommemder models on large, complex, datasets of real user data. I devoted large amounts of effort to tackle the problems of generating synthetic data and writing code to efficiently build and train a several models from scratch with limited computation resources.

Future work into this project could explore deeper models with additional layer and more featurescontaining various data types such as text and images. Another aspect of future work lies in tryng other methods of data generation, like those described in [5] and [6] or a custom method designed specifically for this use-case.

7 Contributions

Savannah McCoy worked as the individual contributor for this project.

8 Code

The following GitHub repository is being used to host the code for this project:
https://github.com/savannahmccoy/cs230-project

References

[1] Project, U. C. S. D. C. S. E. R. (n.d.). Recommender systems and Personalization datasets. Recommender Systems Datasets. Retrieved October 7, 2021, from https://cseweb.ucsd.edu/j̃mcauley/datasets.html#amazon_reviews.

[2] Kirk, J. (2019, January 22). Getting started with recommender systems and TensorRec. Medium. Retrieved November 6, 2021, from https://towardsdatascience.com/getting-started-with-recommender-systems-and-tensorrec-8f50a9943eef.

[3] Tensorflow recommenders. TensorFlow. (n.d.). Retrieved November 6, 2021,from https://www.tensorflow.org/recommenders.

[4] TensorFlow recommenders: Scalable Retrieval and feature interaction modelling. The TensorFlow Blog. (n.d.). Retrieved November 6, 2021, from https://blog.tensorflow.org/2020/11/tensorflow-recommenders- scalable-retrieval-feature-interaction-modelling.html.

[5] Kong, Kezhi, et al. “FLAG: Adversarial Data Augmentation for Graph Neural Networks.” ArXiv:2010.09891 [Cs, Stat], Oct. 2020. arXiv.org, http://arxiv.org/abs/2010.09891.

[6] Zhao, Tong, et al. “Data Augmentation for Graph Neural Networks.” ArXiv:2006.06830 [Cs, Stat], Dec. 2020. arXiv.org, http://arxiv.org/abs/2006.06830.

[7] P. Melville and V. Sindhwani. Recommender systems. In Encyclopedia of Machine Learning, 2010.

[8] Shuai Zhang, L. Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system. ACM Computing Surveys (CSUR), 52:1 – 38, 2019.

[9] Dhoha Almazro, Ghadeer Shahatah, Lamia Albdulkarim, Mona Kherees, Romy Martinez, and William Nzoukou. A survey paper on recommender systems. ArXiv, abs/1006.5278, 2010.

[10] Jonathan Hui. GAN — What is Generative Adversarial Networks GAN? (2018), medium article.

[12] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Networks (2014). arXiv:1406.2661

[13] Lei Xu LIDS, Kalyan Veeramachaneni. Synthesizing Tabular Data using Generative Adversarial Networks (2018). arXiv: 1811.11264v1 [cs.LG]

[14] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, Kalyan Veeramachaneni. Modeling Tabular Data using Conditional GAN (2019). arXiv:1907.00503v2 [cs.LG]

[15] Denis Vorotyntsev. Benchmarking Categorical Encoders (2019). Medium post

[16] Insaf Ashrapov. GAN-for-tabular-data (2020). Github repository.

[17] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, Timo Aila. Analyzing and Improving the Image Quality of StyleGAN (2019) arXiv:1912.04958v2 [cs.CV]