Synthesis of Electronic Health Records Using Bayesian Generative Adversarial Networks

Preetom Saha Arko

Abstract

With the increasing demand for diverse datasets in machine learning applications, the generation of synthetic data has emerged as a popular research area. Bayesian Neural Networks introduce a probabilistic framework, enabling modeling of uncertainty and providing a principled mechanism for generating diverse samples. This study explores the synthesis of Electronic Health Records using Bayesian GAN.

1 Introduction

Traditional neural networks use point estimates for their weights, which may not capture uncertainties in the data. In contrast, Bayesian neural networks (BNN) treat the weights as probability distributions, allowing them to model uncertainty and generate synthetic data that reflects this uncertainty. In this study, we will explore an approach to synthetic data generation using Bayesian neural networks.

The increasing digitization of sensitive information has led to heightened concerns regarding privacy and the security of individual-level data. Particularly, where datasets contain private details. The risk of identity and attribute disclosure poses a significant challenge. Addressing these concerns necessitates innovative approaches that balance the preservation of privacy with using data for analytical purposes. Here, the exploration of synthetic data emerges as a promising strategy. Statistical properties of an original dataset are employed to generate artificial samples.

Generating Electronic Health Records (EHR) is a good application of synthetic data generation. EHRs are scarce and have a lot of privacy and confidentiality concerns. That's why there is a pressing need to generate good diverse EHRs from the already available ones, while ensuring that privacy and confidentiality do not get compromised if the synthetic data gets leaked. In section 2, we will discuss on synthesis of electronic health records using Bayesian Generative Adversarial Networks.

2 Synthesis of Electronic Health Records Using Bayesian GAN

2.1 Motivation and Related Works

Generative adversarial networks (GAN) [7] are deep neural network architectures consisting of two components: a generator and a discriminator. Given some unlabeled input data, the generator G tries to generate truly new samples from the input data distribution, therefore implicitly trying to approximate the true data distribution as best as possible. The discriminator D tries to determine as best as possible if any input was part of the real input data or was generated by the generator G. By providing feedback on each other's outputs, the generator and discriminator incrementally improve on their respective tasks.

Due to privacy concerns and patient confidentiality regulations, research on Electronic Health Records (EHR) typically requires synthetic data, and the generator of GAN seems a promising tool for that. For this, we need a good GAN model that can faithfully capture the distribution of actual health data and later its generator can generate synthetic data from which no confidential information can be inferred. However, a GAN exhibits remarkable performance in generating real-valued continuous

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

data, but it has limitations in generating discrete data [14]. The main reason is that a GAN fails to learn the distribution of discrete data in its original form during the gradient update process in training. To overcome this limitation, MEDGAN [5] incorporated an autoencoder with the original GAN to learn the distribution of discrete data and then generated synthetic discrete EHR data. The output of the generator is passed through the decoder of the autoencoder before going to the discriminator. SynthEHR [4] proposed two improved design concepts of the original GAN, namely, Wasserstein GAN with gradient penalty (WGAN-GP) [8] and boundary-seeking GAN (BGAN) [10] as alternatives to the GAN in the MEDGAN framework. The proposed models named MEDBGAN and MEDWGAN both outperformed MEDGAN.
GANs are notoriously hard to train for various reasons. One such reason is that if the generator G finds some convenient way to fool the discriminator D, it can overly exploit this and overfit to a specific part of the distribution. We will then have a generator that does not correctly approximate the distribution of the data, but has instead "collapsed" to one of its modes. The resulting generated data will exhibit very poor diversity. This phenomenon is known as **mode collapse** and this is a very common problem of GANs. To solve this issue, AdaGAN [17], MGAN [11], MADGAN [6] used multiple generators instead of a single one with the intuition that multiple generators can better model multi-modal distributions since each generator only needs to capture a subset of the modes. To entail model aggregation, probabilistic modeling is a natural and principled framework to articulate the aggregation process.

Recently some probabilistic approaches have been tried to address the issue of mode collapse. For example, Bayesian GAN [15] aims to overcome mode collapse by simulating the discriminator and generator from their respective conditional posterior distributions; however, the two conditional posterior distributions are incompatible and can lead to unpredictable behavior [1], especially for minimax-style GAN objective. The authors of ProbGAN [9] showed that Bayesian GAN suffers from bad convergence due to its fixed and weakly informative prior. To solve this issue, ProbGAN imposes an adaptive prior on the generator and updates the prior by successively multiplying the likelihood function at each iteration; consequently, the generator converges to a fixed point instead of a distribution. The most recent approach, Empirical Bayesian GAN (EBGAN) [13] improves over both Bayesian GAN and ProbGAN to solve mode collapse. In this approach, the discriminator converges to a fixed point while the generator converges to a distribution at the Nash equilibrium. Multiple generators are simulated from the posterior distribution conditioned on the discriminator using momentum stochastic gradient Langevin dynamics (MSGLD) algorithm, and the discriminator is updated using stochastic gradient descent along with simulations of the generators. EBGAN has a strong theoretical guarantee to solve the issue of mode collapse.

MEDGAN previously incorporated the minibatch averaging method into their adversarial framework to prevent the problem of mode collapse, but it has no strong theoretical guarantee. As SynthEHR architecture is based on MEDGAN, it also employs the same method. In our study, we will try to find out whether EBGAN (current state-of-the-art method for addressing mode collapse) can outperform MEDGAN or SynthEHR.

2.2 Dataset

We used Medical Information Mart for Intensive Care (MIMIC-III) database [12], a freely available public database comprising de-identified electronic health records associated with approximately 60K patient admissions to the critical care units of the Beth Israel Deaconess Medical Center between 2001 and 2012. *MIMIC-III* contains various types of health-related data, of which we used patients' diagnoses data (DIAGNOSES_ICD) and procedures (PROCEDURES_ICD) data, coded using the International Statistical Classification of Diseases and Related Health Problems (ICD) system. Following the previous approaches [5] [4], we reduced the ICD codes to 3-digit codes. In the *MIMIC-III* dataset, each row corresponds to a patient's admission record of diagnoses and procedures data, represented by ICD codes. A patient likely visits a hospital more than once, so s/he may have multiple records in the dataset. We aggregated each patient's longitudinal record into a single fixed-sized vector of ICD codes. Thus, we represented the dataset as a multidimensional matrix, in which a row corresponds to a patient's record and a column to a specific ICD code (eg, diagnoses code or procedure code). Since ICD codes are aggregated by the patients, they are all count variables. The count variables indicate the number of times a patient was associated with a specific ICD code. In the end, our matrix has 46517 patients and 1358 ICD codes.

2.3 Experimental Setup

Interpreting synthetic data: The raw generated data values from the generator were any continuous nonnegative numbers. We rounded the continuous values of the synthetic data to the nearest integer values.

Source code: We obtained the Pytorch implementation of Bayesian GAN [2], Tensorflow implementation of MEDGAN, MEDBGAN and MEDWGAN [3] and Pytorch implementation of EBGAN [16] from Github. For Bayesian GAN, We had to change some in-place operations to make it work. We had to downgrade our Tensorflow version to 1.14 to train MEDGAN, MEDBGAN and MEDWGAN.

Method for evaluating synthetic EHRs: We performed dimension-wise Kolmogorov–Smirnov test (K–S) test on 2 data samples (synthetic data and real data) to examine whether the 2 data samples originate from the same distribution. In the K–S test, the statistic is calculated by finding the maximum absolute value of the differences between 2 samples' cumulative distribution functions. The null hypothesis is that both samples originate from a population with the same distribution. In our experiment, we rejected the null hypothesis when P-value ≤ 0.05.
We used 80% of our dataset for training and the rest 20% for testing. Specifically, during testing, we generated the same number of samples as the number of samples in the test set using the generator and then performed dimension-wise K-S test between the generated samples and the test samples. We report the average value along all dimensions.

System Information: We used an AMD Ryzen 9 7940HS laptop with NVIDIA RTX 4060 GPU to train the models.

2.4 Experiments and Results

As our study is based on Bayesian Neural Networks and Bayesian GAN is the first Bayesian approach to address mode collapse, we trained Bayesian GAN in its default configuration on CIFAR-10 dataset. Surprisingly, Bayesian GAN performed worse in semi-supervised learning than conventional GAN (Figure 1).

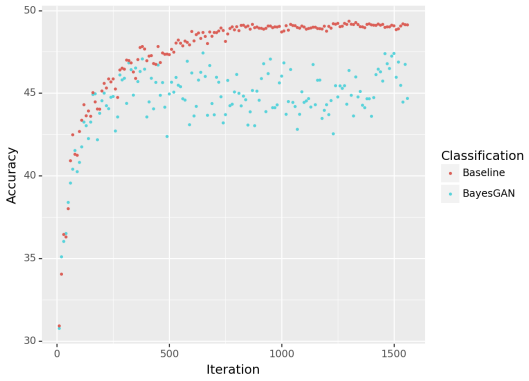

Figure 1: Baeysian GAN vs Conventional GAN in semi-supervised setting on CIFAR-10 dataset

To maintain the good aspects of both MEDGAN and EBGAN, we tried to follow the model architecture of MEDGAN while designing the generator and discriminator in EBGAN. Following the approach of MEDGAN, we added an autoencoder to EBGAN, trained it on our training data and then trained the generator and discriminator of EBGAN. As mentioned before, the output of the generator goes through the decoder before going to the discriminator in this approach. During training, generator loss initially decreased upto 15 epochs, then increased continuously, leading to unstable training. Changing the learning rate had almost no effect on this trend. Probably the autoencoder has somewhat memorized the data and it directly contradicts our approach to find a distribution for the generator. The highest K-S test similarity we got in this setting is only 52.45%.

Then we removed the autoencoder from EBGAN. We had to remove some skip connections in the generator to keep the dimensions suitable for matrix operations in the neural network. Following

Model	K-S test similarity
MEDGAN	87.63%
MEDWGAN	90.06%
MEDBGAN	90.79%
EBGAN	86.97%

Table 1: K-S test similarity shown by various models on our dataset

MEDGAN, we trained EBGAN in this setting for 1000 epochs with a batch size of 1000. We fine-tuned the learning rates of generator and discriminator and found that generator learning rate of $0.2 - 0.3$ and discriminator learning rate of $0.01 - 0.05$ works well for our dataset. The GAN typically converges in 200 epochs and no significant change can be observed afterwards. We got the same K-S test similarity for 200 epoch training and 1000 epoch training.

We have also trained MEDGAN, MEDBGAN and MEDWGAN models on our dataset. MEDWGAN was found to perform the best among these three models on the dataset we used [4]. A comparison of the K-S test similarity between different models on our dataset can be seen in Table 1.

From Table 1, we can see that the data generated by MEDBGAN model resembles the actual data the most. This means the probabilistic approach of EBGAN cannot outperform the minibatch averaging method and boundary-seeking GAN objective of MEDBGAN.

EBGAN also proposes Kullback-Leibler divergence-based prior which enhances the similarity between synthetic data and real data at the density level. But implementing it resulted in CUDA Runtime error.

Our implementation of EBGAN and Bayesian GAN can be found here: `https://github.com/preetom-saha-arko/EHR_Synthesis`

3 Conclusion and Future Work

Though our approach to generate synthetic EHRs could not outperform the current state-of-the-art in terms of K-S test similarity, the disclosure risk of our model may be lower. We plan to investigate it in the future. Adopting KL prior may boost the faithfulness of data generation.

One way to improve the performance of our approach involves using more computational resources. The noticeable computational demands, particularly in datasets with higher dimensions and with probabilistic neural networks, suggest that increasing computing power may noticeably enhance the performance of the model.

References

[1] Barry C Arnold and S James Press. Compatible conditional distributions. *Journal of the American Statistical Association*, 84(405):152–156, 1989.

[2] Ben Athiwaratkun. pytorch-bayesgan. `https://github.com/vasiloglou/mltrain-nips-2017/tree/master/ben_athiwaratkun/pytorch-bayesgan`, 2017. [Online; accessed 6 December 2023].

[3] Mrinal Kanti Baowaly. SynthEHR. `https://github.com/baowaly/SynthEHR`, 2019. [Online; accessed 6 December 2023].

[4] Mrinal Kanti Baowaly, Chia-Ching Lin, Chao-Lin Liu, and Kuan-Ta Chen. Synthesizing electronic health records using improved generative adversarial networks. *Journal of the American Medical Informatics Association*, 26(3):228–241, 2019.

[5] Edward Choi, Siddharth Biswal, Bradley Malin, Jon Duke, Walter F Stewart, and Jimeng Sun. Generating multi-label discrete patient records using generative adversarial networks. In *Machine learning for healthcare conference*, pages 286–305. PMLR, 2017.

[6] Arnab Ghosh, Viveka Kulharia, Vinay P Namboodiri, Philip HS Torr, and Puneet K Dokania. Multi-agent diverse generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8513–8521, 2018.

[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[8] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

[9] Hao He, Hao Wang, Guang-He Lee, and Yonglong Tian. Probgan: Towards probabilistic gan with theoretical guarantees. In *International Conference on Learning Representations*, 2018.

[10] R Devon Hjelm, Athul Paul Jacob, Tong Che, Adam Trischler, Kyunghyun Cho, and Yoshua Bengio. Boundary-seeking generative adversarial networks. *arXiv preprint arXiv:1702.08431*, 2017.

[11] Quan Hoang, Tu Dinh Nguyen, Trung Le, and Dinh Phung. Mgan: Training generative adversarial nets with multiple generators. In *International Conference on Learning Representations 2018*. OpenReview, 2018.

[12] Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9, 2016.

[13] Sehwan Kim, Qifan Song, and Faming Liang. A new paradigm for generative adversarial networks based on randomized decision rules. *arXiv preprint arXiv:2306.13641*, 2023.

[14] Matt J Kusner and José Miguel Hernández-Lobato. Gans for sequences of discrete elements with the gumbel-softmax distribution. *arXiv preprint arXiv:1611.04051*, 2016.

[15] Yunus Saatci and Andrew G Wilson. Bayesian gan. *Advances in neural information processing systems*, 30, 2017.

[16] sehwankimstat. EBGAN. `https://github.com/sehwankimstat/EBGAN/tree/main`, 2023. [Online; accessed 6 December 2023].

[17] Ilya O Tolstikhin, Sylvain Gelly, Olivier Bousquet, Carl-Johann Simon-Gabriel, and Bernhard Schölkopf. Adagan: Boosting generative models. *Advances in neural information processing systems*, 30, 2017.