Medical Missing Data Imputation by Stackelberg GAN

Hongyang Zhang
Machine Learning Department
Carnegie Mellon University
hongyanz@cs.cmu.edu

Committee: Maria-Florina Balcan
David P. Woodruff

Tuesday 11th December, 2018

Abstract

We study the problem of imputing medical missing data by Stackelberg GAN. Having complete medical datasets has many applications in disease prevention, diagnose, and control. However, most of medical data that we can access to suffer from missing values due to failure of data collection, damage of lab devices, lost records, and many other reasons. To resolve the issue, traditional methods apply matrix factorization or GAN based methods to impute the missing position. Unfortunately, it is well-known that matrix factorization based methods cannot characterize the non-linear structure of data, while standard GAN based methods suffer from mode collapse and dropping issues, where oftentimes the imputed values tend to be the same. This paper aims at resolving both of these issues by introducing the Stackelberg GAN. The idea is to utilizing multiple generators instead of a single generator as in the standard GAN, so that the imputed values are more diverse. Preliminary experiments on the UCI dataset verify the effectiveness of the proposed method compared with other state-of-the-art imputation approaches.

To solve the real medical problems, we apply the proposed Stackelberg GAN based imputation method to the MIMIC-III dataset. MIMIC-III is a dataset which consists of over 58,000 hospital admissions for 38,645 adults and 7,875 neonates. The data ranges from June 2001 to October 2012. However, the percentage of missing records is as high as 74%, causing potential difficulties of mining and analysis of this dataset. To resolve the issue, we apply our proposed model to impute the dataset. Our algorithm obtains outstanding performance on this dataset, achieving 0.86 F1 score and 95% prediction accuracy after we pre-process/complete the data by our algorithm. Our result reveals the possibility of imputing missing data in the real problems, which might be of independent interest more broadly.

1 Introduction

We are in an era of big data as well as high dimensional data. However, the data may not be complete in most cases. This is typically because of failure of data collection, damage of measuring devices, lost records, and many other reasons. Missing data can cause huge problems in many real-world applications. For example, in the medical domain, failure of obtaining complete data may lead to false diagnose of diseases. In the recommendation system, absence of complete user preference data causes incorrect recommendation of products, resulting in huge economic loss. The focus of this paper is to design computationally efficient methods to impute the medical missing data based on Generative Adversarial Net (GAN). Our study of

medical missing data imputation algorithms help to build better medical auxiliary system for hospitals and potentially save more lives.

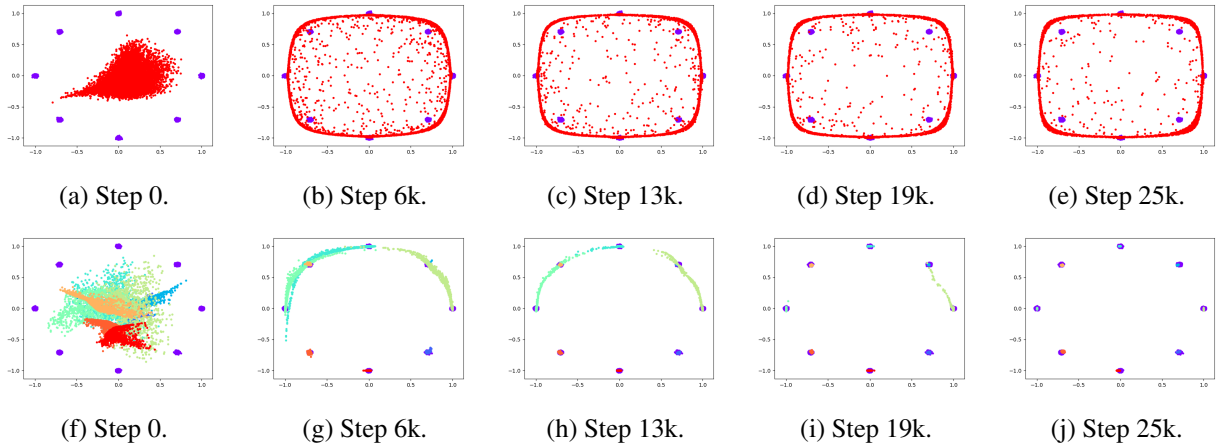
GANs are an emerging object of study in machine learning, computer vision, natural language processing, and many other domains. In machine learning, study of such a framework has led to significant advances in adversarial defenses [XLZ+18, SKC18] and machine security [ACW18]. In computer vision and natural language processing, GAN has resulted in improved performance over standard generative models for images and texts [GPAM+14], such as variational autoencoder [KW13] and deep Boltzman machine [SL10]. A main technique to achieve this goal is to play a minimax two-player game between generator and discriminator under the designs that the generator tries to confuse the discriminator with its generated contents and the discriminator tries to distinguish real images/texts from what the generator creates. As a generative model, we apply GAN to the missing data imputation problems.

Despite a large amount of variants of GAN, many fundamental questions remain unresolved. One of the long-standing challenges is designing *universal, easy-implemented* architectures that alleviate the mode collapse and dropping issues, where often the generated images of GAN are not very diverse (see the first row in Figure 1). This phenomenon is caused by the large discrepancy between the minimax and maximin objective values. With the absence of convexity of discriminators and generators which are parametrized by deep neural networks, gradient-based algorithms may easily get stuck at sub-optimal solutions, resulting in large minimax gap and unstable training. As a result, the imputed values for the missing positions by GAN tend to be the same, leading to meaningless imputation.

To alleviate the issues caused by the large minimax gap, our study is motivated by a so-called Stackelberg competition in the domain of game theory. In the Stackelberg leadership model, the players of this game are one *leader* and multiple *followers*, where the leader firm moves first and then the follower firms move sequentially. It is known that the Stackelberg model can be solved to find a *subgame perfect Nash equilibrium*. We apply this idea of Stackelberg leadership model to the architecture designs of GAN. That is, we design an improved GAN architecture with multiple generators (followers) which team up to play against the unique discriminator (leader), and plug this model into the missing data imputation problems. We therefore term our model *Stackelberg GAN*. Stackelberg GAN enjoys alleviated mode collapse and dropping issues (see the second row in Figure 1).

Our Contributions. This paper tackles the problem of instability during the GAN training procedure and applies the improved GAN to the medical data imputation problems with wide potential applications in the future.

- We propose a Stackelberg GAN framework of having multiple generators in the GAN architecture. Our framework is general that can be applied to all variants of standard GAN. It is built upon the idea of jointly optimizing an ensemble of GAN losses w.r.t. all pairs of discriminator and generator.
- We clarify how to interpolate the idea of Stackelberg GAN into the missing data imputation problem. In particular, we follow the framework of [YJvdS18]. However, there are key difference between the methodology in [YJvdS18] and our methodology: In [YJvdS18], the authors apply the standard GAN as core and there is only one generator in their model. Instead, in our method, we apply the idea of Stackelberg GAN to have multiple generators. The advantage of this design is that we can hope to learn more complicated distribution w.r.t. the missing value; It can be observed that the generated value of standard GAN is not very diverse. We empirically conduct preliminary tests of our algorithm on



(a) Step 0. (b) Step 6k. (c) Step 13k. (d) Step 19k. (e) Step 25k.

(f) Step 0. (g) Step 6k. (h) Step 13k. (i) Step 19k. (j) Step 25k.

Figure 1: Stackelberg GAN stabilizes the training procedure on a toy 2D mixture of 8 Gaussians. **Top Row:** Standard GAN training. It shows that several modes are dropped. **Bottom Row:** Stackelberg GAN training with 8 generator ensembles, each of which is denoted by one color. We can see that each generator exactly learns one mode of the distribution without any mode being dropped.

the UCI dataset. Experiments show that our algorithm significantly outperforms other state-of-the-art methods.

- As a real application of our proposed algorithm to the medical problems, we conduct experiments on the MIMIC-III dataset. MIMIC-III (Medical Information Mart for Intensive Care III, [JPS+16]) is a dataset which consists of over 58,000 hospital admissions for 38,645 adults and 7,875 neonates. The data ranges from June 2001 to October 2012. However, the percentage of missing records is as high as 74%, which causes potential difficulties of mining and analysis of this dataset. To resolve the issue, we apply our proposed model to impute the dataset. Our algorithm obtains outstanding performance on this dataset, achieving 0.86 F1 score and 95% prediction accuracy after we pre-process/complete the data by our algorithm. In contrast, without running our missing data imputation algorithm, to the best of our knowledge, there is no previous work which reports prediction result on this dataset (although they use the dataset for other purpose). Our result reveals the possibility of imputing missing data in the real problems, which might be of independent interest more broadly.

2 Stackelberg GAN

Before proceeding, we define some notations and formalize our model setup in this section.

Notations. We will use bold lower-case letter to represent vector and lower-case letter to represent scalar. Specifically, we denote by $\theta \in \mathbb{R}^t$ the parameter vector of discriminator and $\gamma \in \mathbb{R}^g$ the parameter vector of generator. Let $D_\theta(\mathbf{x})$ be the output probability of discriminator given input $\mathbf{x}$, and let $G_\gamma(\mathbf{z})$ represent the generated vector given random input $\mathbf{z}$. We will use I to represent the number of generators throughout the paper.

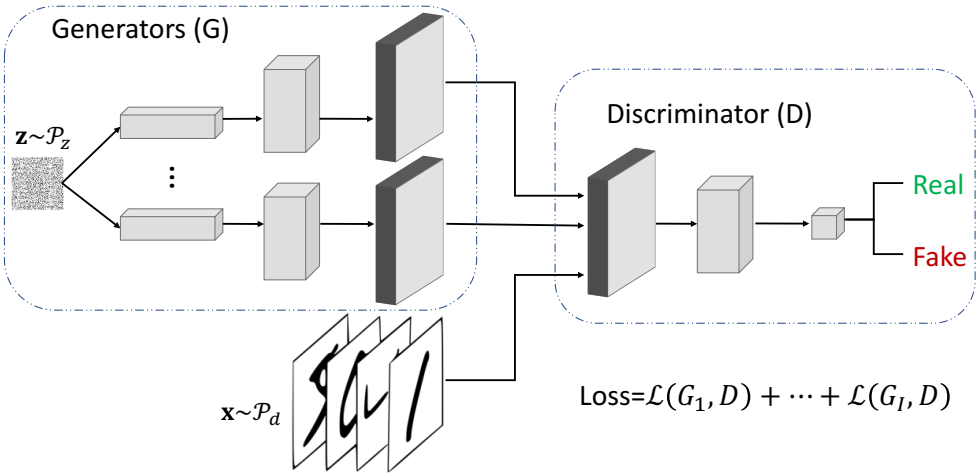


Figure 2: Architecture of Stackelberg GAN.

2.1 Model Setup

Preliminaries. The key ingredient in the standard GAN is to play a *zero-sum two-player* game between a discriminator and a generator — which are often parametrized by deep neural networks in practice — such that the goal of the generator is to map random noise $\mathbf{z}$ to some plausible images/texts $G_\gamma(\mathbf{z})$ and the discriminator $D_\theta(\cdot)$ aims at distinguishing the real images/texts from what the generator creates.
For every pure strategy γ and θ of generator and discriminator, respectively, denote by the payoff value

$$\phi(\gamma;\theta) := \mathbb{E}_{\mathbf{x}\sim\mathcal{P}_d} f(D_\theta(\mathbf{x})) + \mathbb{E}_{\mathbf{z}\sim\mathcal{P}_z} f(1 - D_\theta(G_\gamma(\mathbf{z}))),$$

where $f(\cdot)$ is some concave, increasing function. Hereby, $\mathcal{P}_d$ is the distribution of true images/texts and $\mathcal{P}_z$ is a noise distribution such as Gaussian or uniform distribution. The standard GAN thus solves the following saddle point problems:

$$\inf_{\gamma\in\mathbb{R}^g} \sup_{\theta\in\mathbb{R}^t} \phi(\gamma;\theta), \qquad \text{or} \qquad \sup_{\theta\in\mathbb{R}^t} \inf_{\gamma\in\mathbb{R}^g} \phi(\gamma;\theta). \tag{1}$$

For different choices of function f, problem (1) leads to various variants of GAN. For example, when $f(t) = \log t$, problem (1) is the classic GAN; when $f(t) = t$, it reduces to the Wasserstein GAN. We refer interested readers to the paper [NCT16] for more variants of GAN.

Stackelberg GAN. Our model of Stackelberg GAN is inspired from the Stackelberg competition in the domain of game theory. Instead of playing a two-player game as in the standard GAN, in Stackelberg GAN there are $I + 1$ players with two firms — one discriminator and I generators. One can make an analogy between the discriminator (generators) in the Stackelberg GAN and the leader (followers) in the Stackelberg competition.
Stackelberg GAN is a general framework which can be built on top of all variants of standard GAN. The objective function is simply an ensemble of standard GAN losses w.r.t. all pairs of generators and

discriminator: $\Phi(\gamma_1, ..., \gamma_I; \theta) := \sum_{i=1}^{I} \phi(\gamma_i; \theta)$. Thus it is very easy to implement. The Stackelberg GAN therefore solves the following saddle point problems:

$$w^* := \inf_{\gamma_1,...,\gamma_I \in \mathbb{R}^g} \sup_{\theta \in \mathbb{R}^t} \frac{1}{I} \Phi(\gamma_1, ..., \gamma_I; \theta), \qquad \text{or} \qquad q^* := \sup_{\theta \in \mathbb{R}^t} \inf_{\gamma_1,...,\gamma_I \in \mathbb{R}^g} \frac{1}{I} \Phi(\gamma_1, ..., \gamma_I; \theta).$$

We term $w^* - q^*$ the *minimax duality gap*. We note that there are key differences between the naïve ensembling model and ours. In the naïve ensembling model, one trains multiple GAN models *independently* and averages their outputs. In contrast, our Stackelberg GAN shares a unique discriminator for various generators, thus requires *jointly training*. Figure 2 shows the architecture of our Stackelberg GAN.

3 Missing Data Imputation by Stackelberg GAN

In this section, we apply Stackelberg GAN to the missing data imputation problem. We also complement our new method with preliminary experiments on the UCI dataset to verify the effectiveness of our proposed method.

3.1 Problem Formulation

Consider a d-dimensional space $\mathcal{X} = \mathcal{X}_1 \times ... \times \mathcal{X}_d$. Denote by $\mathbf{X} = (X_1, ..., X_d)$ a random variable taking values in $\mathcal{X}$ with distribution $P(\mathbf{X})$. Let $\mathbf{M} = (M_1, ..., M_d)$ be a random variable taking value in $\{0, 1\}^d$. We name $\mathbf{X}$ the data vector and $\mathbf{M}$ the mask vector. For each $i \in \{1, ..., d\}$, denote by $\widetilde{\mathcal{X}}_i = \mathcal{X}_i \cup \{*\}$, where $*$ is a point which is not in any $\mathcal{X}_i$ and represents an unobserved value. Let $\widetilde{\mathcal{X}} = \widetilde{\mathcal{X}}_1 \times ... \times \widetilde{\mathcal{X}}_d$ and $\widetilde{\mathbf{X}} = (\widetilde{X}_1, ..., \widetilde{X}_d)$, where

$$\widetilde{X}_i = \begin{cases} X_i, & \text{if } M_i = 1, \\ *, & \text{otherwise.} \end{cases} \tag{2}$$

We note that $\mathbf{M}$ indicates which components of $\mathbf{X}$ are observed.
In the imputation setting, our goal is to impute the unobserved values in each $\widetilde{\mathbf{x}}_i$. That is, we want to generate samples according to $P(\mathbf{X}|\widetilde{\mathbf{X}} = \widetilde{\mathbf{x}}^i)$, the conditional distribution of $\mathbf{X}$ given $\widetilde{\mathbf{X}} = \widetilde{\mathbf{x}}^i$, in order to fill in the missing data points. This is in contrast with the *deterministic* matrix completion problem where we assume there exists an underlying deterministic matrix and our goal is to exactly recover that matrix with a high probability [CR09].

3.2 Methodology

We use the idea of [YJvdS18] to combine Stackelberg GAN with the missing data imputation problem.
For a data matrix with missing entries, the basic idea is to use generators to create a value for each missing position so that the discriminator cannot distinguish whether this entry is missing or originally available.
To be more specific, we use the architecture in Figure 3 to do missing data imputation by Stackelberg GAN. We will clarify the function of each component (generator, discriminator, hint matrix, loss function) in the following subsections.

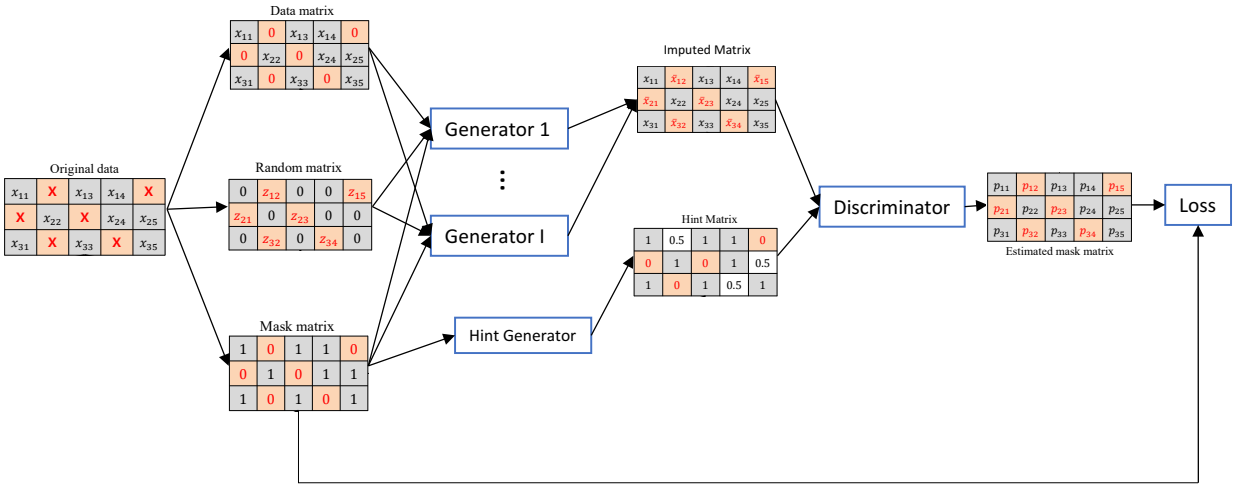


Figure 3: Architecture of matrix imputation by Stackelberg GAN.

3.2.1 Generator

In the Stackelberg GAN, there are multiple generators. We denote by I the number of generators. For each generator, it takes data matrix $\widetilde{\mathbf{X}}$, mask matrix $\mathbf{M}$, and random matrix $\mathbf{Z}$ as inputs. It outputs an imputed matrix $\widehat{\mathbf{X}}$. Let $G_i(\cdot)$ be the mapping associated with the i-th generator. Then we have

$$\widehat{\mathbf{X}} = G_i(\widetilde{\mathbf{X}}, \mathbf{M}, (\mathbf{1} - \mathbf{M}) \odot \mathbf{Z}). \tag{3}$$

Analogous to the standard GAN, hereby the random matrix $\mathbf{Z}$ is analogous to the noise variable $\mathbf{z} \sim \mathcal{P}_z$ in Section 2.1. The $\odot$ denotes the element-wise multiplication. Note that the target distribution $P(\widehat{\mathbf{X}}|\widetilde{\mathbf{X}})$ is essentially of dimension $\|\mathbf{1} - \mathbf{M}\|_1$. Thus the noise that we pass into the i-th generator is $(\mathbf{1} - \mathbf{M}) \odot \mathbf{Z}$ instead of $\mathbf{Z}$. Note that we involve multiple generators here in order to avoid the mode collapse issues and increase the diversity of the generated contents.

3.2.2 Discriminator

We apply the discriminator as an adversary to train the generators. It takes the imputed matrix obtained by the above-mentioned generators as an input. However, there is a key difference between the role of discriminator in the standard GAN and in our model: In the standard GAN the output of the generators is either real or fake so that the discriminator is only required to distinguish whether the whole matrix is real or fake, while in our model the discriminator tries to distinguish whether a given component of the matrix is originally observed or is imputed by the generators so that the distinguishing is element-wise. More formally, we formulate the discriminator as a function $D : \mathcal{X} \to [0, 1]^d$, where the i-th component of $D(\widehat{\mathbf{X}})$ represents the probability that the i-th component of $\widehat{\mathbf{X}}$ was observed. We want to match this probability with the index in the mask matrix $\mathbf{M}$.

3.2.3 Hint Matrix

As can be seen from the theoretical results of [YJvdS18], we have to introduce a hint mechanism, otherwise the algorithm of missing data imputation would fail. [YJvdS18] showed that if we do not provide enough information about $\mathbf{M}$ to the discriminator D, then there are multiple distributions that G can generate which are all optimal w.r.t. D. To this end, we follow the design of hint matrix from [YJvdS18]. In particular, a hint mechanism is a random variable $\mathbf{H} \in \{0, 0.5, 1\}^d$. We allow $\mathbf{H}$ to depend on the mask matrix $\mathbf{M}$. Let the random variable $\mathbf{B} = (B_1, ..., B_d)$ be drawn by first sampling a k uniformly from $\{1, 2, ..., d\}$ and then

$$B_j = \begin{cases} 1, & j \neq k; \\ 0, & j = k. \end{cases} \tag{4}$$

We then have $\mathbf{H}$ defined by

$$\mathbf{H} = \mathbf{B} \odot \mathbf{M} + 0.5(\mathbf{1} - \mathbf{B}). \tag{5}$$

We note that $\mathbf{H}$ satisfies that $H_i = t$ implies $M_i = t$ for $t \in \{0, 1\}$, while $H_i = 0.5$ implies nothing about the mask M_i. That is, $\mathbf{H}$ reveals all but one entry of $\mathbf{M}$ to the discriminator D. However, $\mathbf{H}$ indeed contains certain information about M_i as M_i may not be independent of all other components of $\mathbf{M}$. With the introduction of $\mathbf{H}$, the output of the discriminator becomes $D(\widehat{\mathbf{X}}, \mathbf{H})$.

3.2.4 Loss Function

We train D and G_i's adversarially. The goal of the discriminator is to maximize the probability of successfully predicting $\mathbf{M}$ and the goal of generators is to minimize that probability. Denote by

$$V(D, G) = \mathbb{E}_{\widehat{\mathbf{X}}, \mathbf{M}, \mathbf{H}} \left[\left\langle \mathbf{M}, \log D(\widehat{\mathbf{X}}, \mathbf{H}) \right\rangle + \left\langle \mathbf{1} - \mathbf{M}, \log(\mathbf{1} - D(\widehat{\mathbf{X}}, \mathbf{H})) \right\rangle \right], \tag{6}$$

where $\log$ is the element-wise logarithm and $\widehat{\mathbf{X}}$ depends on G. The imputation model associated with the standard GAN is then

$$\min_G \max_D V(D, G). \tag{7}$$

In our Stackelberg GAN, however, we apply multiple generators instead and the corresponding model is

$$\min_{G_1, ..., G_I} \max_D \sum_{i=1}^{I} V(D, G_i). \tag{8}$$

3.2.5 Missing Value Imputation

We use the output of G to impute the missing data. However, there are multiple generators in our Stackelberg GAN model. In order to generator a single value for each missing entry position, we first draw a uniformly random value i from 1 to I and use the i-th generator to obtain a new sample. Note that this procedure is independent of the training procedure.

Table 1: Description of UCI datasets.

Name	# Training	# Test	# Features	Data Types
bc	489	105	9	Categorical
iris	105	23	4	Continuous
vowel	693	149	9	Continuous
housevote	304	66	16	Categorical
servo	116	26	4	Categorical
boston	354	76	13	Categorical, Ordinal, Continuous

Table 2: Performance of various methods on the UCI datasets with 50% missing values.

	Our method	VAE	MICE	Mean	Zero	Medain	AE	DAE	RAE	MissForest	KNN	PCA	SoftImpute
bc	**2.59**	2.71	2.8	3.24	4.44	2.76	3.25	3.04	3.23	3.12	2.94	3.03	4.5
iris	**0.94**	0.98	1.02	1.05	3.18	1.08	1	1.02	1.04	1.07	1.1	1.05	3.02
vowel	**4.30**	5.19	5.68	5.22	6.34	5.27	5.22	5.19	5.22	5.43	5.59	5.2	8.62
housevote	**2.69**	3.1	3.12	3.19	11.2	3.9	3.19	3.17	3.19	3.25	3.39	3.1	8.8
servo	**0.70**	1.15	1.32	1.22	2.89	1.23	1.26	1.2	1.2	1.32	1.28	1.32	2.67
boston	**3.77**	4.65	5.03	4.34	8.39	4.27	4.74	4.7	4.71	5.42	5.11	4.56	11.7

3.3 Preliminary Experiments on UCI Dataset

To demonstrate the effectiveness of the proposed method, we conduct preliminary experiments on the UCI dataset before we apply it to the real dataset. We choose 6 sub-datasets from UCI dataset for our purpose: bc, iris, vowel, housevote, servo, and boston. The descriptions of these 6 sub-datasets w.r.t the number of training data, the number of test data, feature dimension and data types are in Table 4. In order to make the problem a missing data imputation problem, we randomly remove 50% components of the data matrix and run various algorithms to complete it. These algorithms include our Stackelberg GAN based method with 10 generators, VAE [ZXX18], MICE [BGO10], Mean, Zero, Median, AE [RHW85], DAE [GW17], RAE [TLZJ17], MissForest [SB11], KNN, PCA, and SoftImpute [MHT10].
Table 2 records the performance of various imputation methods on the UCI dataset. Hereby, we report the mean square error between the imputed matrix and the ground-truth. It shows that in all 6 sub-datasets, the Stackelberg GAN based method uniformly outperforms other algorithms.

4 Imputation and Prediction on Real Medical Dataset — MIMIC-III

In this section, we apply our Stackelberg GAN to the real-world problem of imputing missing data in the medical dataset — MIMIC-III.

4.1 Problem Description

MIMIC-III Dataset. MIMIC-III (Medical Information Mart for Intensive Care III, [JPS+16]) is a dataset which consists of over 58,000 hospital admissions for 38,645 adults and 7,875 neonates. The data ranges from

	A	B	C	D	E	F	G	H	I
1	ROW_ID	SUBJECT_ID	HADM_ID	ITEMID	CHARTTIME	VALUE	VALUENUM	VALUEUOM	FLAG
2	441	3	145834	50868	##########	17	17	mEq/L	
3	442	3	145834	50882	##########	25	25	mEq/L	
4	443	3	145834	50893	##########	8.2	8.2	mg/dL	abnormal
5	444	3	145834	50902	##########	99	99	mEq/L	abnormal
6	445	3	145834	50910	##########	48	48	IU/L	
7	446	3	145834	50911	##########	NotDone		ng/mL	
8	447	3	145834	50912	##########	3.2	3.2	mg/dL	abnormal
9	448	3	145834	50931	##########	91	91	mg/dL	
10	449	3	145834	50960	##########	2.4	2.4	mg/dL	
11	450	3	145834	50970	##########	4.8	4.8	mg/dL	abnormal
12	451	3	145834	50971	##########	5.4	5.4	mEq/L	abnormal
13	452	3	145834	50983	##########	136	136	mEq/L	
14	453	3	145834	51002	##########	<0.3		ng/ml	
15	930	3	145834	50863	##########	89	89	IU/L	
16	931	3	145834	50868	##########	13	13	mEq/L	
17	932	3	145834	50878	##########	67	67	IU/L	abnormal
18	933	3	145834	50882	##########	15	15	mEq/L	abnormal
19	934	3	145834	50885	##########	0.8	0.8	mg/dL	
20	935	3	145834	50893	##########	7.6	7.6	mg/dL	abnormal
21	936	3	145834	50902	##########	112	112	mEq/L	
22	937	3	145834	50912	##########	1.9	1.9	mg/dL	abnormal
23	938	3	145834	50931	##########	133	133	mg/dL	abnormal
24	939	3	145834	50960	##########	2.1	2.1	mg/dL	
25	940	3	145834	50970	##########	3.1	3.1	mg/dL	delta
26	941	3	145834	50971	##########	4.3	4.3	mEq/L	
27	942	3	145834	50983	##########	136	136	mEq/L	
28	943	3	145834	51002	##########	GREATER THAN 50		ng/ml	
29	944	3	145834	51006	##########	36	36	mg/dL	abnormal
30	945	3	145834	51137	##########	NORMAL			

Figure 4: Examples of labevents table in the MIMIC-III dataset.

June 2001 to October 2012. Although the dataset is de-identified, it contains significant information about the clinical care of patients. The dataset is provided by the researchers at MIT Laboratory for Computational Physiology.

There are 26 tables in the MIMIC-III dataset. We use the labevents table in our experiments. The labevents table provides information about all laboratory measurements for various patients. There are totally 27,854,055 rows in the table. Table columns include ROW_ID (INT), SUBJECT_ID (INT), HADM_ID (INT), ITEMID (INT), CHARTTIME (TIMESTAMP(0)), VALUE (VARCHAR(200)), CALUENUM (DOUBLE PRECISION), VALUEUOM (VARCHAR(20)), and FLAG (VARCHAR(20)). Here SUBJECT_ID is the identifier which is unique to a patient and HADM_ID is unique to a patient hospital stay. ITEMID is the identifier for a single measurement type in the database. For example, ITEMID 212 corresponds to an instantiation of the heart rate. CHARTTIME records the time at which an obsrevation was charted. VALUE contains the value measured for the test identified by the ITEMID. VALUENUM contains the same data as VALUE if this value is numeric, and is NULL if this value is not numeric. VALUEUOM is used to record the unit of measurement. Finally, FLAG indicates whether the laboratory value is considered abnormal or not. An example of raw labevents table is shown in Table 4. Besides, all HADM_ID is labelled according to whether the patient passes away or not. Identifying whether a given a patent passes away or not, this is a natural two-class classification problem. This problem has huge applications in the medical auxiliary system, as predicting the health of a patient helps doctors to act in advance and save life. For example, when

our system predicts one patient may have potential risk to pass away, the hospital may assign ICU to this patient and have more nurses to take care of him/her. However, the challenge is that there are huge amount of missing values in the dataset.

We preprocess the data by reformulating the table according to HADM_ID. That is, each row of matrix after our reformulation corresponds to various lab tests of one patient. With this preprocessing step, we obtain a huge training data matrix with $37,199$ rows and 77 features, and a test matrix with $11,625$ rows and 77 features, where features correspond to results from various lab tests. We find that there are only 26% entries available for both matrices. This is no ground-truth for these missing values, but we have ground-truth label . Our goal is using the *training data only* to learn the distribution of the missing values and impute the test data. That is, although our method is unsupervised, we do not try to use the test data to learn the distribution of the missing value, as in reality the test data comes row-wise in sequence. What we learn from is the (incomplete) training data and their labels. After both matrices being imputed, we run a classifier to predict whether patients passes away or not in the test data matrix with a high accuracy.

4.2 Experimental Setup

Our experimental setup is in Table 3. Note that here we do not try to use large networks. So our algorithm runs pretty fast in minutes.

Table 3: Architecture and hyperparameters setup in our experiments.

Operation	Input Dim	Output Dim	Activation
Generator $G(\mathbf{z})$: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$		256	
Linear	256	128	ReLU
Linear	128	Dim	ReLU
Discriminator			
Linear	Dim	256	ReLU
Linear	256	128	ReLU
Linear	128	1	Sigmoid
Number of generators	10		
Batch size for real data	64		
Number of iterations	10000		
Learning rate	0.0002		
Optimizer	Adam		

4.3 Experimental Results

We implement and run our Stackelberg GAN algorithm on the MIMIC-III dataset. To alleviate the mode collapse issue where the completed values of GAN are oftentimes exactly the same, we apply $I = 10$ generators in our Stackelberg GAN. In order to generate one value for a missing-value position, we sample a number i uniformly at random from 1 to 10, and use the i-th generator to complete that value.

Our experimental environment is Tensorflow and we can access to one NVIDIA GTX 1080Ti GPU. Specifically, we run our method developed in Section 3 to learn the distribution of missing component and use it to impute the test data. After we obtain an imputed test data, we simply run weighted logistic regression for 100 epochs to predict the label of test data. We set the weight for each class propositional to the number of data points of each class. Since we have access to the true label of test data, we record the classification accuracy on the test dataset. It shows that our method achieves two-class error as low as 5%.
As most patients did not pass away, the dataset is also notoriously known as imbalanced for the two classes. Therefore, to measure the performance of our method in this sense, we also report the F1 score. The F1 score is defined as

$$\text{F1 Score} = 2 \times \frac{\text{precision * recall}}{\text{precision + recall}}. \tag{9}$$

Hereby, the precision is defined as the number of true positives over the number of true positives plus the number of false positives, and the recall is defined as the number of true positives over the number of true positives plus the number of false negatives. It is well-known that the F1 score is able to handle class imbalance to some extent. Our model achieves F1 score as high as 0.86, thus can handle the imbalanced data automatically.

Table 4: Prediction performance after data imputation.

Name	F1 Score	Classification Accuracy
Fixed-value-filling	0.47	89%
Nuclear Norm Minimization	0.48	90%
Classic GAN	0.85	$94 \pm 0.46\%$
Stackelberg GAN (Ours)	**0.86**	$\mathbf{95 \pm 0.27\%}$

4.4 Analysis and Discussion

We compare the performance of Stackelberg GAN with several baseline approaches — fixed-value-filling, nuclear norm minimization, and classic GAN. For the fixed-value-filling method, it fills in a fixed value, e.g., the mean or median, for all missing positions. The method has been widely applied in all aspects of fields with missing data as it is easy to implement and enjoys not-so-bad performance. In particular, we fill in the value of 100 for all the missing positions in the data matrix, as many available values in the data matrix are of magnitude 100. It shows that the method achieves classification accuracy 89%. However, the F1 score is as low as 0.47. The nuclear norm minimization achieves certain improvement over the fixed-value-filling method. However, since nuclear norm minimization only explores the linear structure in the data, the improvement is limited. The method of GAN achieves great improvement. By increasing the number of generators, our approach achieves the highest performance on both metrics.

5 Conclusions

We study the problem of imputing medical missing data by Stackelberg GAN. To solve the real problems, we apply the proposed Stackelberg GAN based imputation method to the MIMIC-III dataset. The percentage of

missing records in this dataset is as high as 74%, which causes potential difficulties of mining and analysis of this dataset. To resolve the issue, we apply our proposed model to impute the dataset. Our algorithm obtains outstanding performance on this dataset, achieving 0.86 F1 score and 95% prediction accuracy after we pre-process/complete the data by our algorithm. Our result reveals the possibility of imputing missing data in the real problems, which might be of independent interest more broadly.

Acknowledgements. We thank Petuum Inc. for providing us GPU resource and MIMIC-III dataset, and thank Hongbao Zhang and Pengtao Xie for offering the code in their paper for our comparison.

References

[ACW18] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

[BGO10] S van Buuren and Karin Groothuis-Oudshoorn. MICE: Multivariate imputation by chained equations in R. *Journal of Statistical Software*, pages 1–68, 2010.

[CR09] E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009.

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[GW17] Lovedeep Gondara and Ke Wang. Multiple imputation using deep denoising autoencoders. *arXiv preprint arXiv:1705.02737*, 2017.

[JPS+16] Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, 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*, 3:160035, 2016.

[KW13] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[MHT10] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *Journal of machine learning research*, 11:2287–2322, 2010.

[NCT16] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-GAN: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems*, pages 271–279, 2016.

[RHW85] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[SB11] Daniel J Stekhoven and Peter Bühlmann. Missforest—non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118, 2011.

[SKC18] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. *arXiv preprint arXiv:1805.06605*, 2018.

[SL10] Ruslan Salakhutdinov and Hugo Larochelle. Efficient learning of deep boltzmann machines. In *International Conference on Artificial Intelligence and Statistics*, pages 693–700, 2010.

[TLZJ17] Luan Tran, Xiaoming Liu, Jiayu Zhou, and Rong Jin. Missing modalities imputation via cascaded residual autoencoder. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1405–1414, 2017.

[XLZ+18] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. *arXiv preprint arXiv:1801.02610*, 2018.

[YJvdS18] Jinsung Yoon, James Jordon, and Mihaela van der Schaar. GAIN: Missing data imputation using generative adversarial nets. *arXiv preprint arXiv:1806.02920*, 2018.

[ZXX18] Hongbao Zhang, Pengtao Xie, and Eric Xing. Missing value imputation based on deep generative models. *arXiv preprint arXiv:1808.01684*, 2018.