A Study on Distribution Learning of Generative Adversarial Networks

Jiaping Liu

Supervised by:

Dr. Maia Fraser

uOttawa

DEPARTMENT OF MATHEMATICS AND STATISTICS

SCHOOL OF SCIENCE

UNIVERSITY OF OTTAWA

A thesis submitted in partial fulfillment of the requirements for the Master's degree in Science.

Acknowledgement

I'd like to thank the University of Ottawa for giving me the chance to explore the unknown and my supervisor Dr. Maia Fraser for helping me with academic research generously. I'd also like to thank Dr. Yongyi Mao and his students from the School of Electrical Engineering and Computer Science at the University of Ottawa and my friend Guang Li for giving me advice from another perspective, and my friend Jing Jing for providing me with computing resources.

Notation

	Notations	Descriptions
1	m (i=1,...,m)	sample size
2	p (u=1,...,p, j=1,...,p)	number of input variables
3	$z = \{z_u^{(i)}\} \in \mathbb{R}^{m \times p}$	noise samples
4	$\hat{z} = (z\ 1) \in \mathbb{R}^{m \times (p+1)}$	
5	p_Z	noise prior distribution
6	$x = \{x_j^{(i)}\} \in [0,1]^{m \times p}$	examples
7	$\hat{x} = (x\ 1) \in [0,1]^{m \times (p+1)}$	
8	p_{data}	target distribution
9	$D_x = \{D_x^{(i)}\} \in [0,1]^m$	output of the discriminator (input x)
10	$z_g = \{z_{gj}^{(i)}\} \in [0,1]^{m \times p}$	generated samples
11	$\hat{z}_g = (z_g\ 1) \in [0,1]^{m \times (p+1)}$	
12	p_G	distribution of the generated samples
13	$D_z = \{D_z^{(i)}\} \in [0,1]^m$	output of the discriminator (input z_g)
14	$W^g = \{w_{uj}^g\} \in \mathbb{R}^{p \times p}$	weights of the generator
15	$b^g = \{b_j^g\} \in \mathbb{R}^p$	bias introduced in the generator
16	$\theta^g = \bar{W}^g = ((W^g)^T\ b^g)^T \in \mathbb{R}^{(p+1) \times p}$	parameters of the generator
17	θ^{g0}	initial parameters of the generator
18	θ^d	parameters of the discriminator
19	θ^{d0}	initial parameters of the discriminator
20	$W^{d(1)}$	parameters between the input layer and the hidden layer of the discriminator with one hidden layer
21	$W^{d(2)}$	parameters between the hidden layer and the output layer of the discriminator with one hidden layer
22	(a,b)	range of θ^{g_0}
23	(c,d)	range of θ^{d_0}
24	J_d	loss function of the discriminator
25	J_g	loss function of the generator
26	$\nabla_{\theta^d} J_d$	gradient of θ^d
27	$\nabla_{\theta^g} J_g$	gradient of θ^g
28	α	learning rate of the discriminator
29	β	learning rate of the generator
30	N	iteration epochs
31	L	steps of optimizing discriminator in each epoch

In Section 4, we design three algorithms. Since parameters and hyper-parameters of the discriminator vary from algorithm to algorithm, they are defined below separately for specific algorithms.

Algorithm 1 with Maxout layer in the discriminator:

	Notations in Algorithm 1	Descriptions
1	Q (q=1,...,Q)	number of Maxout units
2	K (k=1,...,K)	number of neurons in each Maxout unit
3	$\mu = \{\mu_{jk}\} \in \{0,1\}^{p\times K}$	a binary mask used in the Maxout layer
4	$W^d = \{W^d_{jk}\} \in \mathbb{R}^{p\times K}$	weights for each Maxout unit
5	$b^d = \{b^d_k\} \in \mathbb{R}^K$	bias introduced in each Maxout unit
6	$\theta^d = (W^{d^T}\ b^d)^T \in \mathbb{R}^{(p+1)\times K}$	parameters of the discriminator
7	$v \in \mathbb{R}^Q$	a coefficient vector for combining Q Maxout units
8	$h_{z-argmax} \in \{1,...,K\}^{m\times Q}$	argmax among neurons in a Maxout unit (input z_g)

Algorithm 2 with sigmoid activation function and without hidden layers in the discriminator:

	Notations in Algorithm 2	Descriptions
1	$W^d = \{W^d_j\} \in \mathbb{R}^p$	weights of the discriminator
2	$b^d \in \mathbb{R}$	bias introduced in the discriminator
3	$\theta^d = (W^{d^T}\ b^d)^T \in \mathbb{R}^{(p+1)}$	parameters of the discriminator

Algorithm 3 with sigmoid activation function and one hidden layer in the discriminator:

	Notations in Algorithm 3	Descriptions
1	Q (q=1,...,Q)	number of neurons in the hidden layer
2	$x^1 = \{x^{1(i)}_q\} \in [0,1]^{m\times Q}$	neurons in the hidden layer (input x)
3	$\hat{x}^1 = (x^1\ 1) \in [0,1]^{m\times(Q+1)}$	
4	$z^1_g = \{z^{1(i)}_g{}_q\} \in [0,1]^{m\times Q}$	neurons in the hidden layer (input z_g)
5	$\hat{z}^1_g = (z^1_g\ 1) \in [0,1]^{m\times(Q+1)}$	
6	$W^{d(1)} = \{w^{d(1)}_{jq}\} \in \mathbb{R}^{p\times Q}$	parameters between the input layer and the hidden layer of the discriminator
7	$b^{d(1)} = \{b^{d(1)}_q\} \in \mathbb{R}^Q$	bias between the input layer and the hidden layer of the discriminator
8	$\bar{W}^{d(1)} = ((W^{d(1)})^T\ b^{d(1)})^T \in \mathbb{R}^{(p+1)\times Q}$	
9	$W^{d(2)} = \{w^{d(2)}_q\} \in \mathbb{R}^Q$	parameters between the hidden layer and the output layer of the discriminator
10	$b^{d(2)} \in \mathbb{R}$	bias between the hidden layer and the output layer
11	$\bar{W}^{d(2)} = ((W^{d(2)})^T\ b^{d(2)})^T \in \mathbb{R}^{(Q+1)}$	
12	$\theta^d = \bar{W}^{d(1)}$ and $\bar{W}^{d(2)}$	parameters of the discriminator

Abstract

This thesis is an exploration of the properties of shallow generative adversarial networks (GANs). We focus on several aspects of GANs to investigate the learnability of a class of distributions using shallow GANs and conduct experiments to explore the influence of these aspects on the performance of the GAN models. We identify and analyze several pathological phenomena in theoretical analysis and experiments, and propose potential solutions for them.

Keywords: distribution learning theory, logit-normal distribution, pathological phenomena, range of initial parameters.

Preface

Shallow perceptron networks have been extensively studied and while their deep versions still hold mysteries, the shallow ones are easy to describe and study explicitly. People know for example the decision boundary [17] of a single perceptron or shallow net of these. Generative adversarial networks (GANs) on the other hand, though they are typically built of perceptrons and have had spectacular practical success, have not had this treatment and remain somewhat mysterious. Therefore, our goal of this thesis is to explicitly study shallow GANs both theoretically and experimentally. Our original contributions include establishing the hypothesis class for the generator of a shallow GAN in a specific setting, and consequences of this for learning.

In Section 1, we give a brief introduction to generative adversarial networks, the related work on GANs and distribution learning theory. We then summarize the main contributions of the thesis.

In Section 2, we list several aspects that we will explore in the thesis and give the reasons why we choose them. We show the architecture diagrams of shallow GAN models in Section 2.1. Then we list some other aspects which are noise prior distribution and target distribution in Section 2.2, and some parameters and hyper-parameters in Section 2.3.

In Section 3, we use the generator defined in Section 2.1 with sigmoid activation function, and set both noise prior distribution and target distribution discussed in Section 2.2 to be normal distributions. In Section 3.1, we discover that the generated distribution is a logit-normal distribution. In Section 3.2, we define in Section 3.2.1 and investigate in Section 3.2.2 and 3.2.3 a distribution learning problem of the generator's task based on the results in Section 3.1.

In Section 4, we conduct experiments to explore the properties of shallow GAN models shown in Section 2.1 based on the distributions, parameters and hyper-parameters listed in Section 2.2 and 2.3. We present the architecture diagrams of shallow GANs under specific settings in Section 4.1. In Section 4.2, we choose several noise prior distributions and

target distributions, and list their combinations in a table. In Section 4.3, we introduce the visualization tools we will use to analyze the performance of the algorithms. In Section 4.4, we discuss concretely the results of the experiments. In the experiments, we identify several pathological phenomena and propose potential solutions to alleviate them. There is a remarkably wide range of different settings to consider and we structure our investigation into these in Sections 4.4.3 to 4.4.5 as follows:

(i) In Section 4.4.3, there are three parts a, b and c. In part a and part b, we discuss the performance of two activation functions in the discriminator respectively. In part c, we compare the differences between them.

(ii) In Section 4.4.4, there are also three parts a, b and c. In part a and part b, we discuss the performance when different ranges of θ^{d0} or θ^{g0} are used respectively. In part c, we summarize the pathological phenomena in part a and part b.

(iii) In Section 4.4.5, there are two parts a and b. In part a, we analyze the improvement of the algorithm with a hidden layer in the discriminator. In part b, we identify the outstanding issues unsolved by the algorithm.

In Section 5, we draw conclusions based on the discussion in Section 3 and Section 4.

In Section 6, we analyze the limitations in the theoretical investigation and experimental exploration and raise some open questions.

Contents

1 Introduction

1.1 Generative adversarial networks

Generative adversarial networks (GANs) are learning frameworks designed by Goodfellow et al in 2004 [1]. They estimate generative models via an adversarial process. In a GAN, a generative model (also called "a generator"), denoted as G, simulates the target data distribution (also called "a target distribution", or "a target") by transforming a noise prior distribution and a discriminative model (also called "a discriminator"), denoted as D, estimates the probability that a sample of the generator comes from the target distribution. Both generator and discriminator can be neural networks.

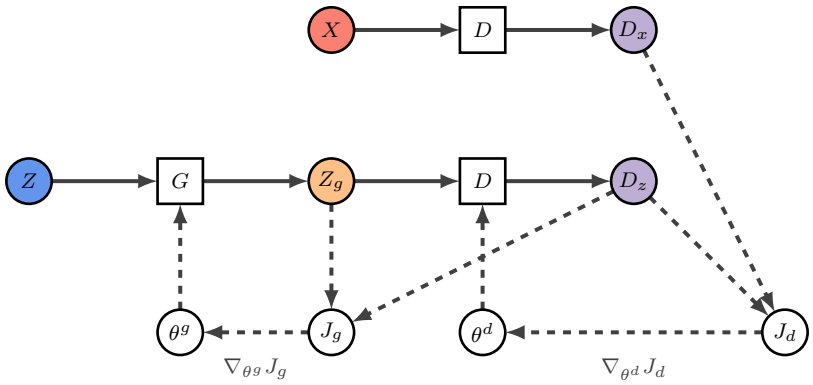
Given enough capacity and training time, a GAN is expected to recover the target distribution. The training procedure is a minimax game and its value function is

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_Z(z)}[\log(1 - D(G(z)))]. \tag{1.1}$$

During the process of training a GAN model, we alternate between several steps of optimizing D and one step of optimizing G. The discriminator maximizes the probability of making a mistake by ascending its parameters by the stochastic gradient (proposed by Lemarechal et al in 2012 [30]) of evaluation function J_d as in Equation (1.2). Then the generator minimizes the mistake by descending its parameters by the stochastic gradient of evaluation function J_g as in Equation (1.2) as below:

$$\begin{aligned} J_d &= \frac{1}{m}\sum_{i=1}^{m}\big[\log D(x^{(i)}) + \log(1 - D(G(z^{(i)})))\big], \\ J_g &= \frac{1}{m}\sum_{i=1}^{m}\log(1 - D(G(z^{(i)}))). \end{aligned} \tag{1.2}$$

The following diagram shows the process of training a GAN model:



In the diagram, z, z_g and x are noise samples, generated samples and examples (data samples from the target distribution) respectively. The generator (G in the diagram) transforms noise samples z to generated samples z_g. Then the discriminator (D in the diagram) computes D_z (respectively D_x), its estimate of the probability that the generated sample (respectively the sample from the target distribution) comes from the target distribution. Now perform back-propagation as follows (shown as dotted lines in the figure above). Compute J_d based on the probabilities D_z and D_x. Then update parameter of the discriminator θ^d by its stochastic gradient $\nabla_{\theta^d} J_d$. Input z_g and compute again the probability D_z using the discriminator with the updated parameter. Then compute J_g and its gradient $\nabla_{\theta^g} J_g$ based on generated samples z_g and the information from the discriminator. In the end, update parameter of the generator θ^g.

1.2 Related work

Generative adversarial networks are an incredible AI technology with various applications. See Langr for an overview 2019 [2]. Creswell et al in 2018 [3] and Hong et al in 2019 [4] overviewed the researches done on GANs and challenges in their theory and application. GANs can be viewed as introducing an "adversarial perturbation" to deep learning to achieve a better performance as discussed by Hazan in 2017 [5]. From a game-theoretic perspective, two neural networks of a GAN are two players in a game aiming for a joint optimization; the GAN seeks for a Nash equilibrium, as covered by Salimans et al in 2016 [6], Goodfellow et al in 2017 [7], Heusel et al in 2018 [8] and Paget in 2019 [9]. From an information-theoretic

view, Paget in 2019 [9] analyzed the theoretical behavior of GANs during optimization. From a topology perspective, inspired by optimal transport theory, Wesserstain GANs were proposed by Arjovsky et al in 2017 [12]. In 2020, in [13, 14] An et al discussed more on optimal transport by GANs.

Distribution learning of GANs was discussed by Arora et al in 2017 [16], and also Lucic et al in 2018 [18] and Khayatkhoei et al in 2019 [19] though these papers mostly contribute empirical results. More generally, the relationship between computational learning theory and neural networks was discussed by Turán in 1994 [34]. However, there remains very little distribution learning-theoretic analysis on neural networks or GANs. In the theory part of this thesis, we take a distribution learning-theoretic view to analyze the generator of specific shallow GANs.

More broadly, the performance of GANs or neural networks is known to be affected by many aspects. Salimans et al in 2016 [20] discussed the importance of weight normalization; Behnke et al in 2003 [21] stated that some activation functions introduce a non-linearity to the output which can be used for decision making; Brownlee in blog post in 2019 [22] discussed the importance of learning rates during training a neural network; Arjovsky et al in 2017 [23] discussed the unstablility and other pathological behaviors of GANs. In this thesis, we choose several aspects that may have a significant influence on GANs and experimentally explore potential pathological behaviors that may arise.

1.3 Main contributions

As discussed in the Preface, the main contribution of this thesis is to elucidate aspects of shallow GANs, inspired by the way shallow multilayer perceptrons are fully understood. Compared to these simple neural networks, however, even shallow GANs are much more complex since each GAN includes two competing neural networks – a generator and a discriminator – and is designed to both generate and judge authenticity of samples from a target distribution. Moreover, GANs require both noise samples and target examples as input. Our study therefore restricts to a certain type of noise input – normally distributed

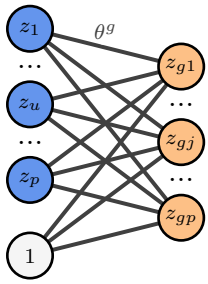
noise – and for this restricted setting achieves some first results. Our first contribution is that we consider a specific shallow GAN design and prove that for normally distributed noise this GAN will always generate samples distributed according to a logit-normal distribution. This is the first result of its kind, to our knowledge. For our second contribution, we then study the generator of this GAN from a distribution-learning point of view, i.e. as a learner of distributions, and show it is incapable of learning a specific kind of target distribution. This is the first learning-theoretic study of GAN generators to our knowledge. Finally, in experiments, we choose different options or values of the aspects to discover the properties of shallow GANs and analyze the pathological phenomena we observed.

2 Our Focus on GANs

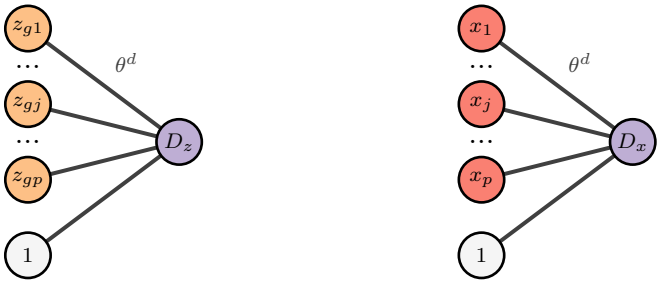
In this section, we list several aspects that we plan to study theoretically and experimentally in the thesis.

2.1 Architecture diagrams

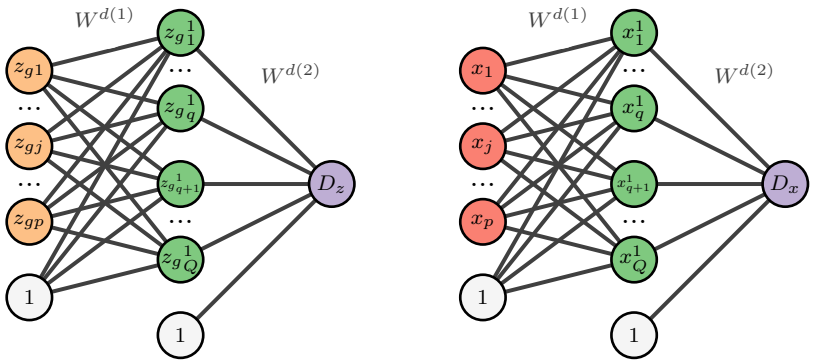
Consider there are p input variables of noise $Z = (Z_1, ..., Z_p)^T$ and example $X = (X_1, ..., X_p)^T$. Set the generator to be a one-layer shallow neural network as follows, in which θ^g are parameters of the generator. Assume there are p output variables of generated sample $z_g = (z_{g1}, ..., z_{gp})^T$.



Set the discriminator to be a one-layer perceptron without hidden layers, in which θ^d are parameters of the discriminator. Output D_z (respectively D_x) when input is z_g (respectively x).



As a richer version of the discriminator, we will also consider adding a hidden layer in the discriminator as below. Let $W^{d(1)}$ to be the parameters between the input layer and the hidden layer, and $W^{d(2)}$ to be the parameters between the hidden layer and the output layer.



2.2 Noise prior distributions and target distributions

The generator transforms noise input z sampled from the noise prior distribution p_Z to generated samples z_g distributed as a generated distribution p_G, based on the value of its parameter θ^g. As training proceeds, θ^g is adjusted gradually in order to bring the generated p_G close to our target distribution p_{data}. Given a specific p_Z and p_{data}, there might be some cases where the parameters could not be adjusted properly, i.e. for which then the generator could not bring p_Z close to p_{data}.

2.3 Parameters and hyper-parameters

There are some parameters and hyper-parameters that have a significant influence on the performance of shallow GANs. We now list some of them we plan to study experimentally in this thesis.

2.3.1 Steps of optimizing the discriminator in each epoch

If the number of steps of optimizing discriminator (L) is too large, this will optimize the discriminator to completion with a fixed generator. However, we would like to avoid this because the generator is improving during training and excessive time spent on D for non-optimal G is wasted. If L is too small, on the other hand, the discriminator may not be optimized enough, and then there will be a lack of information for the generator to improve. Therefore, we have to take an appropriate number of steps according to the situation.

2.3.2 Learning rates of the parameters

The learning rates control how quickly the models are updated. On one hand, we wish learning rates to be large enough, so that we could achieve the results in fewer epochs. However, if learning rates are too large, the models will be likely to reach suboptimal results. If learning rates are too small, we will waste much time on unnecessary training epochs. Therefore, for each model we wish to choose a proper learning rate, which leads to a global optimization of shallow GANs within an acceptable number of epochs.

2.3.3 Activation functions in the discriminator

In artificial neural networks, the activation function of a node defines the output of that node. There are different kinds of activation functions, for example linear and non-linear functions [10]. They sometimes introduce different structures into networks, for example, Maxout layer proposed by Goodfellow et al in 2013 [29] introduces sparsity. Therefore, it is valuable to explore the influence of different activation functions on the performance of shallow GANs.

2.3.4 Ranges of the initial parameters

We sample initial parameters of the discriminator (respectively generator) from a specific range (i.e. difference of the maximum and the minimum value of the parameters). Ranges of the parameters of each model then envolve during training. We investigate how the choice of the initial ranges affects this evolution and whether different ranges may result in significantly different performance of shallow GANs.

2.3.5 Hidden layers in the discriminator

A shallow discriminator may lack the ability to differentiate between real data and fake data. The performance of shallow GANs without any hidden layers in the discriminator may be significantly worse than the performance of those with hidden layers.

3 Some Theoretical Observations for Shallow GANs

In this section, we set the generator to be a one-layer shallow neural network (see the first diagram in Section 2.1) with sigmoid activation function in the output layer; we have no specific settings for the discriminator throughout the section; and we focus on the situation that the noise prior distribution p_Z is a normal distribution.

We will prove from different perspectives in Section 3.1 and Section 3.2 when example X lies within the interval $(0,1)^p$ and target distribution p_{data} is a truncated normal distribution (see [33]) truncated to $(0,1)^p$, there is a pathological phenomenon that the generated distribution p_Z fails to converge to p_{data}.

3.1 Theory related to logit-normal distribution

In this section, we prove that the generated distribution is a logit-normal distribution by probability distribution function (pdf) transformation [26] and it shapes as unimodal or bimodal depending on the value of its squared scale.

Logit-normal distributions were first defined by Aitchison et al in 1976 [37] and we define logit-normal distributions following the definition in Frederic et al in 2003 [36].

***Definition* 1.** *A random variable X is distributed as a logit-normal distribution if its logit transformation $logit(X) = log(X/(1-X))$ is normally distributed.*

Since logit and logistic transformations are inverses of each other, the above is equivalent to "a random variable $X = P(Y)$ where P is the standard logistic function is distributed as a logit-normal distribution if the random variable Y is normally distributed."

3.1.1 Transformation from normal distribution to logit-normal distribution

Assume a variable X is distributed as $p_X = N(\mu, \sigma^2)$. The sigmoid transformation of X is $X_S = Sigmoid(X)$ denoted as $S(X)$, then $X_S = S(X) = 1/(1+e^{-X})$. X_S is distributed as $p_{X_S} = LogitNormal(\mu, \sigma^2)$ denoted as $LogitN(\mu, \sigma^2)$.

Assume a set of variables $Z = (Z_1, ..., Z_p)^T$ is distributed as $p_Z = N(\mu, \Sigma)$ where $\mu = (\mu_1, ..., \mu_p)^T$ and $\Sigma = diag(\sigma_1^2, ..., \sigma_p^2)$. Then its linear combination $Y = w_1 Z_1 + ... + w_p Z_p + b$, $w_1, ..., w_p, b \in \mathbb{R}$ is distributed as $p_Y = N(\mu', \sigma'^2)$, in which $\mu' = w_1\mu_1 + ... + w_p\mu_p + b$ and $\sigma'^2 = w_1^2\sigma_1^2 + ... + w_p^2\sigma_p^2$. Its sigmoid transformation is $Z_g = S(Y) = 1/(1 + e^{-Y})$ which is distributed as $p_G = LogitN(\mu', \sigma'^2)$.

In the generator, when we use the sigmoid function as activation function, the first part of the network produces a linear combination Y which the sigmoid transformation then maps to the output $Z_g = S(Y) = \frac{1}{e^{-Y}+1}$, $Z_g \sim p_G$. Then $y = S^{-1}(z_g) = ln(\frac{z_g}{1-z_g}) = logit(z_g)$ and $|\frac{dS^{-1}(z_g)}{dz_g}| = |\frac{1}{\frac{1}{z_g}-1}\frac{1}{{z_g}^2}| = \frac{1}{z_g(1-z_g)}$, $z_g \in [0, 1]$, so $|\frac{dS^{-1}(z_g)}{dz_g}| > 1$. Then by pdf transformation, the pdf of p_G ($z_g \sim p_G$) is

$$\begin{aligned} f_G &= f_Y|\frac{dS^{-1}(z_g)}{dz_g}| \\ &= \frac{1}{\sqrt{2\pi}|w|\sigma} e^{-\frac{[S^{-1}(z_g)-(w\mu+b)]^2}{2w^2\sigma^2}} \frac{1}{z_g(1-z_g)} \\ &= \frac{1}{\sqrt{2\pi}|w|\sigma z_g(1-z_g)} e^{-\frac{[logit(z_g)-(w\mu+b)]^2}{2w^2\sigma^2}}, \end{aligned} \tag{3.1}$$

which is the pdf of the logit-normal distribution $LogitN(w\mu + b, w^2\sigma^2)$, with location $w\mu + b$ and squared scale $w^2\sigma^2$. The location and squared scale remain same as the ones of the distribution of y; however, shapes of the two distributions can be significantly different.

Logit-normal distribution shapes as unimodal or bimodal depending on its squared scale parameter. In our case, it depends on the value of $\theta^g = (w, b)^T \in R^2$ with μ and σ^2 fixed.

3.1.2 PDF transformation used in the generator in shallow GANs

***Proposition* 1.** *In a shallow GAN model, in which the generator is a one-layer perceptron with sigmoid activation function, if noise prior distribution is normal, the generated distribution will be logit-normal.*

Proof. In univariate cases, assume a variable X is distributed as $N(\mu, \sigma^2)$ over $\mathbb{R}$, then its sigmoid transformation $S(X) = 1/(1 + e^{-X})$ is distributed as $LogitN(\mu, \sigma^2)$. As we saw

above, the probability density function of $X_S = S(X)$ (see Equation (3.3)) can be derived from the pdf of X by pdf transformation:

$$f_{X_S}(x_S) = \frac{1}{\sqrt{2\pi}\sigma x_S(1-x_S)}\ exp\{-\frac{1}{2}[\frac{logit(x_S)-\mu}{\sigma}]^2\},\ x_S \in (0,1). \tag{3.2}$$

Shallow GANs however may have multivariate inputs and outputs (see the first diagram in Section 2.1), we therefore need to consider multivariate Z as well. In multivariate cases, assume $Z = (Z_1, ..., Z_p)^T$ are distributed as $N(\mu, \Sigma)$ with $\mu = (\mu_1, ..., \mu_p)^T$ and $\Sigma = diag(\sigma_1^2, , ..., \sigma_p^2)$. Let the linear combination of the variables to be $Y = w_1Z_1 + ... + w_pZ_p + b$, $w_1, ..., w_p, b \in \mathbb{R}$, which is distributed as $N(\mu', \sigma'^2)$ with $\mu' = w_1\mu_1 + ... + w_p\mu_p + b$ and $\sigma' = w_1^2\sigma_1^2 + ... + w_p^2\sigma_p^2$. Then its sigmoid transformation $Y_S = S(Y) = 1/(1+e^{-Y})$ is distributed as $LogitN(\mu', \sigma'^2)$. The pdf of Y_S is

$$\begin{aligned} f_{Y_S} &= \frac{1}{\sqrt{2\pi}\sigma' y_S(1-y_S)}\ exp\{-\frac{1}{2}[\frac{log(\frac{y_S}{1-y_S})-\mu'}{\sigma'}]^2\},\ y_S \in (0,1). \\ &= \frac{1}{\sqrt{2\pi(w_1^2\sigma_1^2 + ... + w_p^2\sigma_p^2)}\ y_S(1-y_S)}\ exp\{-\frac{[log(\frac{y_S}{1-y_S}) - (w_1\mu_1 + ... + w_p\mu_p + b)]^2}{2(w_1^2\sigma_1^2 + ... + w_p^2\sigma_p^2)}\}. \end{aligned} \tag{3.3}$$

In experiment and the assumption of this section, we assume there are three independently and identically distributed (i.i.d.) input variables $Z = (Z_1, Z_2, Z_3)^T$ in the generator distributed as $p_Z = N(\mu, \Sigma)$ with $\mu = (\mu, \mu, \mu)^T$ and $\Sigma = diag(\sigma^2, \sigma^2, \sigma^2)$. Assume there is one output variable Z_g in the generator with parameters $\theta^g = (w_1, w_2, w_3, b)^T$. The generator performs a composition of two transformations, namely a linear combination $Y = w_1Z_1 + w_2Z_2 + w_3Z_3 + b$ followed by a sigmoid transformation $Z_g = S(Y)$. Therefore, Z_g is distributed as $p_G =$

$LogitN(\mu', \sigma'^2)$ with $\mu' = (w_1 + w_2 + w_3)\mu + b$ and $\sigma'^2 = (w_1^2 + w_2^2 + w_3^2)\sigma^2$. The pdf of Z_g is

$$\begin{aligned} f_{Z_g}(z_g) &= \frac{1}{\sqrt{2\pi}\sigma' z_g(1-z_g)} \exp\Big\{ -\frac{1}{2}[\frac{logit(z_g) - \mu'}{\sigma'}]^2\Big\},\ z_g \in (0,1). \\ &= \frac{1}{\sqrt{2\pi(w_1^2 + w_2^2 + w_3^2)}\sigma z_g(1-z_g)} \exp\Big\{ -\frac{[logit(z_g) - ((w_1 + w_2 + w_3)\mu + b)]^2}{2(w_1^2 + w_2^2 + w_3^2)\sigma^2}\Big\}. \end{aligned} \tag{3.4}$$

In experiment, there are not only three inputs as above, but also three outputs of the generator. Then parameters of the generator are set to be

$$\theta^g = \begin{pmatrix} w_{11} & w_{12} & w_{13} & b_1 \\ w_{21} & w_{22} & w_{23} & b_2 \\ w_{31} & w_{32} & w_{33} & b_3 \end{pmatrix},$$

where each row corresponds to the multivariate setting just discussed above. Therefore there are three linear combinations $Y_u = \sum_{j=1}^3 w_{uj} Z_j + b_u,\ u = 1, 2, 3$ and corresponding sigmoid transformations $Z_{gu} = S(Y_u)$, u=1,2,3. They are distributed as $p_{G_u} = LogitN(\mu_u, \sigma_u^2)$ respectively with $\mu_u = \sum_{j=1}^3 w_{uj}\mu + b_u$ and $\sigma_u^2 = \sum_{j=1}^3 w_{uj}^2 \sigma^2,\ u = 1, 2, 3$.

□

With a fixed $Z \sim p_Z$, parameters of the generator θ^g are updated during the process of training and this causes the probability density function of $Z_g \sim p_G$ to change accordingly. We now look more closely at properties of the logit-normal before studying (in Section 3.2) the gradual change of p_G during learning. For the sake of convenience of visualization we set each variable of the target distribution to be i.i.d., and focus only on the marginal distribution of each generated variable and compare between each pair of distributions of variables from the example and the generated data.

3.1.3 Property of logit-normal distribution used in shallow GANs

Properties of logit-normal distributions are discussed by Aitchison et al in 1980 [24] and Frederic et al in 2008 [25].

Expectation and variance of $Z \sim LogitN(\mu', \sigma'^2)$ are determined respectively by two functions $E(Z_g) = M(\mu', \sigma'^2)$ and $V(Z_g) = V(\mu', \sigma'^2)$ (See Equation (3.6) below). Since there is no closed algebraic expression of these functions, numerical integration is used to compute their values [25].

$$\begin{aligned} E(Z_g) &= \int z_g f_{Z_g} dz_g, \\ V(Z_g) &= E(Z_g - E(Z_g))^2 = \int {z_g}^2 f_{Z_g} dz_g - E(Z_g)^2. \end{aligned} \tag{3.5}$$

Some density plots of typical logit-normal distributions are shown in Figure 1 as below. For each location scale, the density curves switch from unimodal shapes to bimodal shapes as the squared scale increases. At the same time, two modes of bimodal curves move away towards 0 and 1.

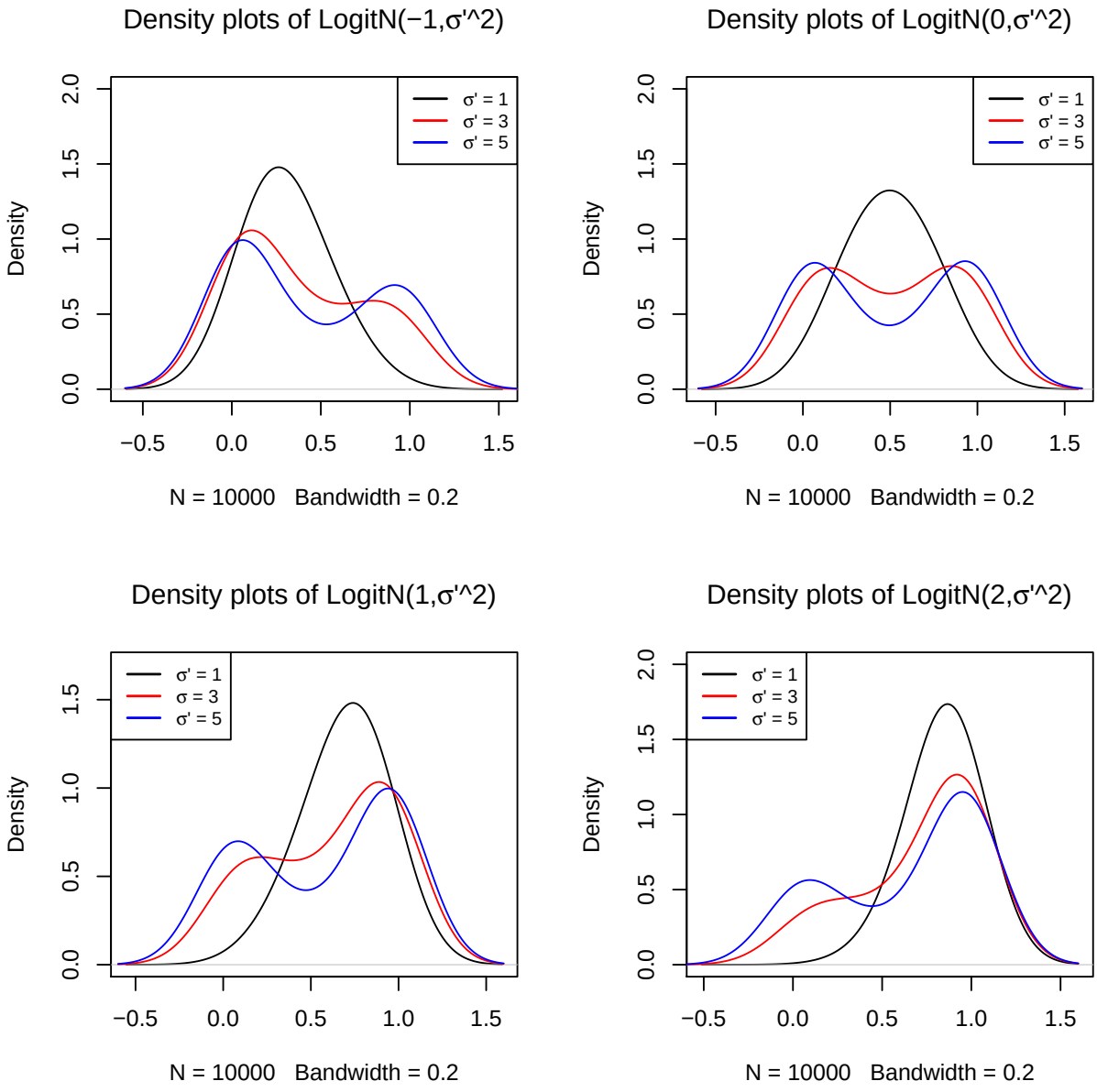


Figure 1

In the exposition by Frederic et al [25], the first and so far only direct analysis of the logit-normal family, the authors remark that while the logit-normal densities always converge to 0 as $x \to 0$ or 1, when σ' is large the densities are increasingly bimodal. This results in logit-normal tending to a delta function when $\sigma' \to 0$ and an adherent mass when $\sigma' \to +\infty$.

Following [35], the delta function $\delta_i(x)$ on $[0, 1]$ with spike at i can be loosely defined for

$i \in (0,1)$ as follows.

$$\delta_i(x) = \begin{cases} 0 & x \neq i \\ \text{“}suitably\ infinite\text{”} & x = i \end{cases}, \tag{3.6}$$

where “suitably infinite” means that for all intervals $(i-1, i+a) \subset [0,1]$, $a > 0$,

$$\text{“}\int_{i-a}^{a+i} \delta_i(x)dx = 1\text{”}. \tag{3.7}$$

In the cases that $\mu' = 0$, the spike of f_{G^*} keeps piling up to infinity and the rest goes down to 0 gradually as $\sigma' \to 0$. Based on the definition in Equation (3.7) and the property in Equation (3.8), the density f_{G^*} tends to a delta function $\delta_{0.5}(x)$, where $0 < x < 1$, as $\sigma' \to 0$.

Similarly we loosely define adherent mass function $\delta_{01}(x)$ on $[0,1]$ with mass adhering to 0 and 1 as follows. Assume $i < j$,

$$\delta_{01}(x) = \begin{cases} 0 & x \neq i, j \\ \text{“}suitably\ infinite\text{”} & x = i, j \end{cases}, \tag{3.8}$$

where $i, j \in [0,1]$ are “infinitesimally close” to 0 and 1 respectively, and “suitably infinite” means that for all intervals $(0, i+a) \subset [0,1]$ and $(j-a, 1) \subset [0,1]$, $a > 0$,

$$\text{“}\int_0^{i+a} \delta_{01}(x)dx + \int_{j-a}^1 \delta_{01}(x)dx = 1\text{”}. \tag{3.9}$$

Logit-normal gains bimodality as σ' becomes greater and two modes move away from each other approaching 0^+ and 1^- at the same time. Both spikes of the density pile up to infinity and the rest goes down to 0 as $\sigma' \to +\infty$. Based on the definition in Equation (3.9) and the property in Equation (3.10), the density f_{G^*} tends to an adherent mass function $\delta_{01}(x)$ where $0 < x < 1$ and $i \to 0^+$, $j \to 1^-$.

We now discuss in more detail this limiting behaviour as σ' tends to 0 or 1. As stated above, and from the exposition in [36] logit-normal tends to an adherent mass with its two modes approaching 0 and 1 as $\sigma' \to +\infty$, and it tends to a delta function with its single

mode at $\frac{1}{2}$ as $\sigma' \to 0$. For convenience of visualization we consider the case $\mu' = 0$ which is depicted in Figure 2 and Figure 3 below, but the limiting behaviour we describe applies to any μ' by the analysis in Frederic [25]. In both scenarios with peaks piling up, with σ' set to be different values, the density plots verify this rule (See Figure 2 and Figure 3 below) and we see moreover the behavior between the two extremes of $\sigma' \in (0, +\infty)$. When σ' is relatively small, the distribution is unimodal and more concentrated. As σ' increases, the distribution gains a bimodality gradually and its peaks move towards 0 and 1. At first the peaks are lower, but as σ' becomes even larger, two modes of the density curve continue to move away from each other while the peaks move upwards.

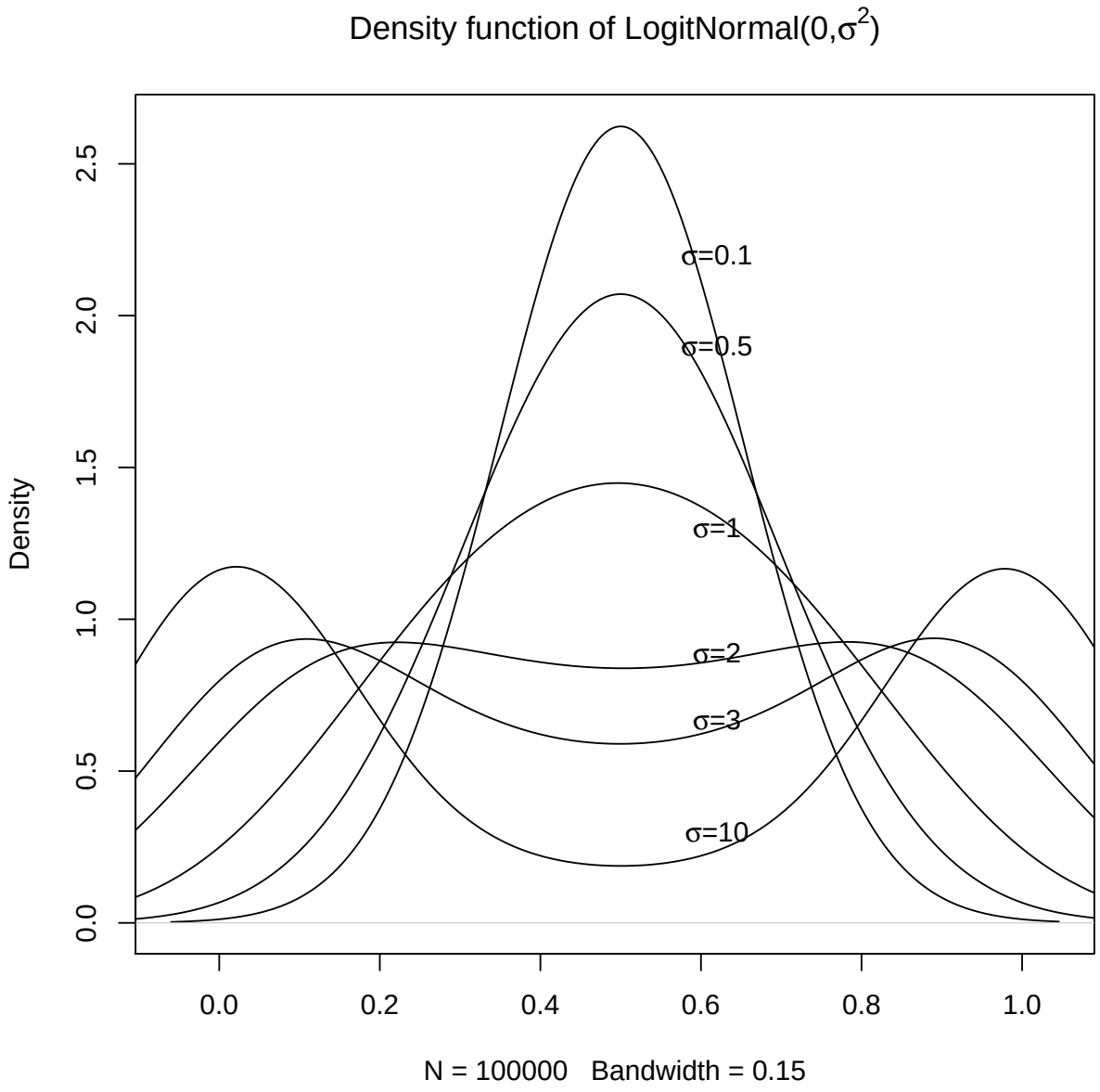

Density function of LogitNormal(0,σ²)
Density
0.0
0.5
1.0
1.5
2.0
2.5
σ=0.1
σ=0.5
σ=1
σ=2
σ=3
σ=10
0.0
0.2
0.4
0.6
0.8
1.0
N = 100000 Bandwidth = 0.15

Figure 2

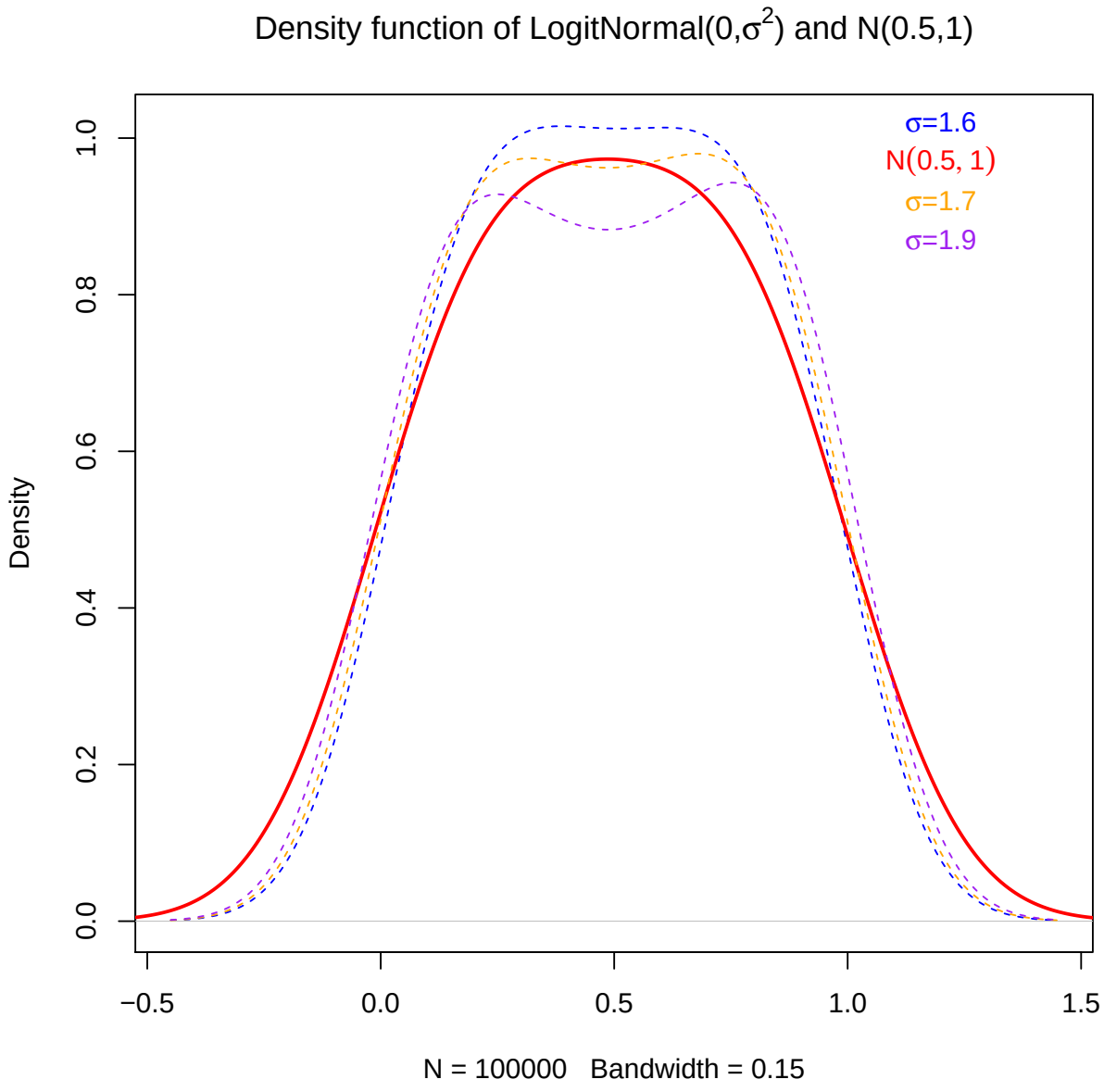


Figure 3

Assume our goal is to learn the target distribution $N(\mu_0, \Sigma_0)$, where $\mu_0 = (0.5, ..., 0.5)^p$ and $\Sigma = diag(1, ..., 1)$. To simplify the presentation, suppose furthermore the generated distribution has learned the expectation of variable X, i.e. $\mathbb{E}(Z_g) = 0.5 = \mathbb{E}(X)$. We are then in the regime of Figure 2 and Figure 3.

In the generator, each input Z_{gu} is distributed as $p_{Gu} \sim LogitN(\mu_u', {\sigma_u}'^2)$ with $\mu_u' = \sum_{j=1}^{3} w_{uj}\mu + b_u$ and ${\sigma_u}'^2 = \sum_{j=1}^{3} w_{uj}^2\sigma^2$ as shown in Section 3.1. If we still set μ_u' to be 0,

for a fixed μ and σ, the generated distribution has a larger squared scale when $\sum_{j=1}^{3} w_{uj}^2$ is larger. Since the logit-normal distribution could either be unimodal and with a small squared scale or be bimodal and with a large squared scale, if the target normal distribution has a large squared scale, intuitively it is impossible for the generated distribution to learn both its unimodality and large squared scale (see Figure 3). If the algorithm updates θ^g to reach a larger $\sum_{j=1}^{3} w_{uj}^2$, then what follows is bimodality.

Note fixing $\mu_u{}'$ as discussed in last paragraph is infeasible in experiments, since $\sum_{j=1}^{3} w_{uj}^2$ will change along with the update of θ^g. However, it is still worth analyzing in this way, because it helps us discover limitations on the ideal ability of the algorithm. In particular, we will formulate the above intuition in learning-theoretic terms in Section 3.2.

3.2 Distribution learning-theoretic analysis

Distribution learning theory was first discussed by Kearns et al in 1994 [38]. In this section, we take a distribution learning-theoretic view of the generator's task, in particular learning a class of distributions through the generator we set at the start of Section 3.

3.2.1 Definition of the distribution learning problem

Let $\mathcal{C}$ be a class of distributions over X, that is $\mathcal{C}$ is a set such that every $\Delta \in \mathcal{C}$ is a probability distribution with support in X. To analyze the distribution learnability of GANs, we focus on the generator of GANs with discriminator fixed but unknown. We set the generator to be a one-layer shallow neural network with Sigmoid activation function and would like to know if it can learn $\mathcal{C}$.

In general, in *Distribution Learning Theory (DLT)* there are two common representations used for a probability distribution Δ over X. A *generator* G_Δ takes noise samples z as input and outputs $G_\Delta(z) \in X$ distributed as the distribution Δ. This is our approach in studying the GAN *generator*: we view it as a representation of the generated distribution p_G. Alternatively, an *evaluator* P_Δ is a probability distribution function of $X \sim \Delta$. It takes any examples $x \in X$ from distribution Δ as input and outputs $P_\Delta(x)$ which denotes

the probability of x distributed as Δ. It can also take generated samples $z_g \in X$ from distribution Δ' as input and output $P_{\Delta'}(z_g)$ which denotes the probability of z_g distributed as Δ'. This will not be our focus, but it is the way in which the discriminator can be viewed as a representation of p_G.

Assume a set of examples $\mathcal{S} = \{(z_1, x_1), ..., (z_m, x_m)\}$ is given in a constant amount of time as the input, and it has access to an oracle $GEN(\Delta)$ returning a sample from the distribution Δ. The GAN training algorithm is designed to find a target function g^* that minimizes the loss function of the generator of GANs. The generator's task is shown in Equation (3.11).

$$\begin{aligned} g^* &= \arg\min_g \int L(g(z)) d\rho(z), \\ \theta^{g*} &= \arg\min_{\theta^g} \int \log\left[1 - D(G(z|\theta^g)|\theta^d)\right] d\rho(z), \end{aligned} \tag{3.10}$$

where $L(g(z)) = \log(1 - D(G(z|\theta^g)|\theta^d))$ is the loss function in which $g(z) = G(z|\theta^g)$. $G(z|\theta^g)$ denotes the function of the generator given parameters θ^g. Let $z_g = G(z|\theta^g)$, then $D(z_g|\theta^d)$ denotes the function of the discriminator given parameters θ^d. $\rho(z, x)$ is the probability distribution according to which the set of examples $\mathcal{S}$ is sampled. Therefore, the goal of DLT is to find the target function g^* given $\mathcal{S}$ sampled from ρ. In our case, the goal is to find the generator $G(z|\theta^{g*})$ with optimal parameters, denoted as G^*.

Define the distribution learnability formally as below [11]:

Definition 2. *A class of distributions $\mathcal{C}$ is called* ***learnable*** *if for every $\varepsilon > 0$ and $0 < \delta < 1$ given access to $GEN(\Delta)$ for an unknown distribution $\Delta \in \mathcal{C}$, there exists an algorithm $\mathcal{A}$, said learning algorithm of $\mathcal{C}$, that outputs a generator or an evaluator of a distribution Δ' such that*

$$Pr[d(\Delta, \Delta') \leq \varepsilon] \geq 1 - \delta,$$

where $d(\Delta, \Delta')$ is a measure of distance between distributions Δ and Δ'.

If $\Delta' \in \mathcal{C}$, then $\mathcal{A}$ is called a ***proper learning algorithm****; if $\Delta' \notin \mathcal{C}$, $\mathcal{A}$ is called an* ***improper learning algorithm****.*

If $\mathcal{A}$ is a polynomial time algorithm, then $\mathcal{C}$ is called ***efficiently learnable****. A distribution Δ is called to have a polynomial generator (respectively evaluator) if its generator (respectively evaluator) exists and can be computed in polynomial time.*

Set the class of distributions $\mathcal{C}$ to be univariate truncated normal distributions $N(\mu, \sigma^2)$ truncated to the range $(0, 1)$ and noise prior distribution p_Z to be a specific univariate normal distribution $N(\mu_0, {\sigma_0}^2)$. Based on Section 3.1, we know that the generated distribution is $p_G = LogitN(\mu', {\sigma'}^2)$ with $\mu' = (w_1^g + w_2^g + w_3^g)\mu_0 + b^g$ and $\sigma' = \sqrt{(w_1^g)^2 + (w_2^g)^2 + (w_3^g)^2}\sigma_0$ in which $\theta^g = (w_1^g, w_2^g, w_3^g, b^g)^T$ are parameters of the generator.

The set of all the distributions the generator can produce – a set sometimes referred to as the "hypothesis set" $(\mathcal{H})$ – is:

$$\mathcal{H} = \{p_G : f(z_g) = f(G(z|\theta^g)) = f(Sigmoid(\hat{z}\theta^g)) \mid \theta^g \in \mathbb{R}^{(p+1)\times p}, z \in \mathbb{R}^{m\times p}\}, \tag{3.11}$$

in which $\hat{z}$ is the noise sample z with an augmented column 1 and $f(x)$ is a probability density function or a probability mass function of a random variable X.

3.2.2 Investigation of the distribution learning problem

One standard measure of the "distance" between distributions in *DLT* is Kullback-Leibler (KL) divergence. We will use this to measure the distance between the target distribution p_{data} and the optimal generated distribution $p_{G^*} \in \mathcal{H}$, denoted as $KL(p_{data}||p_{G^*})$. The KL divergence is

$$\begin{aligned} KL(p_{data}||p_{G^*}) &= -\int_{-\infty}^{\infty} f_{data}(x) \log \frac{f_{G^*}(x)}{f_{data}(x)} dx,\ x \in (0,1), \\ f_{data}(x) &= \frac{\frac{1}{\sigma}\phi(\frac{x-\mu}{\sigma})}{\Phi(\frac{1-\mu}{\sigma}) - \Phi(\frac{-\mu}{\sigma})},\ \phi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}, \\ f_{G^*}(x) &= \frac{1}{\sqrt{2\pi}\sigma' x(1-x)}\ e^{-\frac{[logit(x)-\mu']^2}{2{\sigma'}^2}}, \end{aligned} \tag{3.12}$$

in which $\phi(x)$ is the pdf of standard normal distribution and $\Phi(x)$ is the cumulative distribution function of x. $f_{data}(x)$ is the evaluator (pdf) of the target distribution $p_{data} \in \mathcal{C}$ and $f_{G^*}(x)$

is the evaluator (pdf) of the generated distribution p_{G^*} which minimizes the loss function of the generator. Rescale pdf of normal distribution by dividing $\phi(\frac{x-\mu}{\sigma})$ by $\sigma(\Phi(\frac{1-\mu}{\sigma}) - \Phi(\frac{-\mu}{\sigma}))$, then we have the pdf of truncated normal distribution $p_{data}(x)$.

Since $f_{G^*}(x) \sim LogitN(\mu', {\sigma'}^2) \notin \mathcal{C}$, the algorithm is an improper learning algorithm. We then investigate whether the class of distributions $\mathcal{C}$ is learnable through the generator we set above. With such a goal, we will prove the following *Proposition 2.*

Proposition* 2.** *The class $\mathcal{C}$ of univariate truncated normal distributions truncated to the range $(0, 1)$ is* ***not learnable *by the algorithm in which the generator is a one-layer shallow neural network with sigmoid activation function.*

Proof. **An established property of KL divergence is that it is always non-negative.** This can be seen by recalling Equation (3.13), in which $-\log(x)$ is strictly convex and $\int f_{G^*}(x) = 1$. Then based on Jensen's inequality [28],

$$\begin{aligned} KL(p_{data}||p_{G^*}) &= -\int f_{data} \log \frac{f_{G^*}}{f_{data}} dx \\ &\geq -\log\big(\int f_{G^*} dx\big) = -\log 1 = 0. \end{aligned} \tag{3.13}$$

Another property of KL divergence is its nondegeneracy. This follows from the equality case of the Jensen's inequality (see Equation (3.14)): namely equality if and only if the log factor is zero almost everywhere, i.e. the ratio $\frac{f_{G^*}}{f_{data}}$ is 1. It implies that two density functions f_{data} and f_{G^*} coincide. However, as shown above, the algorithm is an improper algorithm, which means that the density functions of p_{data} and p_{G^*} will never coincide. Therefore, $KL(p_{data}||p_{G^*}) = 0$ does not hold and we conclude $KL(p_{data}||p_{G^*}) > 0$.

A third property of KL divergence is it is lower bounded by total variation distance. Denote the total variation distance as $\delta(p_{data}, p_{G^*})$. Based on Pinsker's inequality [32], since p_{data} and p_{G^*} are two probability distributions, the following inequality holds on a measurable space $(\mathbb{A}, \Sigma)$, where Σ is a σ-algebra on $\mathbb{A}$.

$$\delta(p_{data}, p_{G^*}) \leq \sqrt{\frac{1}{2} KL(p_{data}||p_{G^*})}. \tag{3.14}$$

The definition of total variation distance [31] is

$$\delta(p_{data}, p_{G^*}) = \sup\Big\{|p_{data}(\mathbb{X}) - p_{G^*}(\mathbb{X})|\Big|\mathbb{X} = (0,1),\ \mathbb{X} \in \Sigma \textit{ is a measurable event}\Big\}. \quad (3.15)$$

This is closely related to the L^1 norm $||f_{data} - f_{G^*}||_1$. Indeed a straightforward calculation shows:

$$\delta(p_{data}, p_{G^*}) = \frac{1}{2}||f_{data} - f_{G^*}||_1, \quad (3.16)$$

in which L^1 norm is

$$\begin{aligned} ||f_{data} - f_{G^*}||_1 &= \int_0^1 |f_{data}(x) - f_{G^*}(x)|dx \\ &= \int_0^1 \Big|\frac{\frac{1}{\sigma}\phi(\frac{x-\mu}{\sigma})}{\Phi(\frac{1-\mu}{\sigma}) - \Phi(\frac{-\mu}{\sigma})} - \frac{1}{\sqrt{2\pi}\sigma' x(1-x)}\ e^{-\frac{[logit(x)-\mu']^2}{2\sigma'^2}}\Big|dx, \end{aligned} \quad (3.17)$$

$\phi(x)$ is the probability density function of standard normal distribution ($\phi(x) = \frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}}$) and $\Phi(x)$ is the cumulative distribution function of $\phi(x)$. Therefore,

$$\begin{aligned} ||f_{data} - f_{G^*}||_1 &\le \sqrt{2KL(p_{data}||p_{G^*})} \\ KL(p_{data}||p_{G^*}) &\ge \frac{{||f_{data} - f_{G^*}||_1}^2}{2}. \end{aligned} \quad (3.18)$$

We now prove there is a uniform bound in μ', σ' for KL divergence that is strictly positive.

First, since $p_{data} \neq p_{G^*}$, the L^1 norm at any specific μ' and σ' is greater than 0.

Second, assume we follow the setting in Section 3.1 that $\mu' = 0$. In this case, as discussed in Section 3.1.3, logit-normal tends to an adherent mass as $\sigma' \to +\infty$ and a delta function as $\sigma' \to 0$, which implies $f_{G^*} \to 0$ almost everywhere except the spikes when $\sigma' \to 0$ *or* ∞.

L^1 norm grows as $\mu' = 0$ and $\sigma' \to 0$.

$$\int_{i-a}^{a+i} |f_{data}(x) - f_{G^*}(x)|dx \to \int_{i-a}^{a+i} |f_{data}(x) - \delta_{0.5}(x)|dx = \int_{i-a}^{a+i} \delta_{0.5}(x)dx = 1 \quad (3.19)$$

at the spike $x = i$ and

$$\begin{aligned}
&\int_0^{i-a} |f_{data}(x) - f_{G^*}(x)|dx + \int_{a+i}^1 |f_{data}(x) - f_{G^*}(x)|dx \\
\to &\int_0^{i-a} |f_{data}(x) - \delta_{0.5}(x)|dx + \int_{a+i}^1 |f_{data}(x) - \delta_{0.5}(x)|dx \\
= &\int_0^{i-a} f_{data}(x) - 0 \; dx + \int_{a+i}^1 f_{data}(x) - 0 \; dx \\
= &\int_0^1 f_{data}(x)dx = 1
\end{aligned} \tag{3.20}$$

except the spike. Therefore L^1 norm approaches 2 as below,

$$\begin{aligned}
L^1 \; norm &= \int_0^1 |f_{data}(x) - f_{G^*}(x)|dx \\
&= \int_0^{i-a} |f_{data}(x) - f_{G^*}(x)|dx + \int_{i-a}^{a+i} |f_{data}(x) - f_{G^*}(x)|dx + \int_{a+i}^1 |f_{data}(x) - f_{G^*}(x)|dx \\
&\to 2.
\end{aligned} \tag{3.21}$$

L^1 norm grows as $\mu' = 0$ and $\sigma' \to +\infty$.

$$\int_{i-a}^{a+i} |f_{data}(x) - f_{G^*}(x)|dx + \int_{j-a}^{a+j} |f_{data}(x) - f_{G^*}(x)|dx \to \int_{i-a}^{a+i} \delta_{01}(x)dx + \int_{j-a}^{a+j} \delta_{01}dx = 1 \tag{3.22}$$

at the spikes $x = i, j$ and

$$\begin{aligned}
&\int_0^{i-a} |f_{data}(x) - f_{G^*}(x)|dx + \int_{a+i}^{j-a} |f_{data}(x) - f_{G^*}(x)|dx + \int_{a+j}^1 |f_{data}(x) - f_{G^*}(x)|dx \\
\to &\int_0^{i-a} f_{data}(x)dx + \int_{a+i}^{j-a} f_{data}(x)dx + \int_{a+j}^1 f_{data}(x)dx = \int_0^1 f_{data}(x)dx = 1
\end{aligned} \tag{3.23}$$

except the spikes. Therefore L^1 norm approaches 2 as below,

$$\begin{aligned} L^1\ norm &= \int_0^1 |f_{data}(x) - f_{G^*}(x)|dx \\ &= \int_0^{i-a} |f_{data}(x) - f_{G^*}(x)|dx + \int_{i-a}^{a+i} |f_{data}(x) - f_{G^*}(x)|dx + \int_{a+i}^{j-a} |f_{data}(x) - f_{G^*}(x)|dx \\ &+ \int_{j-a}^{a+j} |f_{data}(x) - f_{G^*}(x)|dx + \int_{a+j}^{1} |f_{data}(x) - f_{G^*}(x)|dx \to 2 \end{aligned} \tag{3.24}$$

as $\sigma' \to +\infty$ when $\mu' = 0$.

Moreover, since it behaves continuously over $\sigma' \in (0, +\infty)$, there must be a finite non-zero σ' that minimizes it. We define the smallest distance as B.

Third, if we move μ' away from 0, it will only increase the L^1 norm, because the generated distribution will be assymetric, while the standard normal is symmetric about $x = \frac{1}{2}$. Therefore, B is still the smallest value L^1 norm.

Therefore, the L^1 norm satisfies $\mathbf{0 < B \leq ||f_{data} - f_{G^*}||_1 < 2}$. Based on Equation (3.19), we have $\mathbf{KL(p_{data}||p_{G^*}) \geq \frac{B^2}{2} > 0}$.

In the end, we conclude that there is a uniform bound for KL divergence that is strictly positive. Therefore, we cannot for $\forall \epsilon > 0$ achieve that $KL(p_{data}||p_{G^*}) \leq \epsilon$ (at all, even for large δ).

In conclusion, the class of distributions $\mathcal{C}$ is not learnable by generator of the algorithm. □

3.2.3 More discussion

We further set the KL divergence to be arbitrarily small yet achievable, i.e. $KL(p_{data}||p_{G^*}) \leq \epsilon$ but $\epsilon \geq \frac{B^2}{2}$ (see Section 3.2.2). It holds for some interval I that $\sigma' \in I \subset (0, +\infty)$. Let the algorithm try to learn such an interval of σ' to push the generated distribution towards the target. However, it may be unachievable, if the discriminator has prematurely converged to $D_x = D_z = \frac{1}{2}$. One way such artificial convergence happens is that the parameters of

the discriminator all converge to 0. In this case, looking at formula B.23, B.40 and B.57 in Appendix B we see the gradients of θ^g will converge to 0 at the same time. In practice, θ^d may not be exactly zero, just very close to zero. By postulating how fast it goes to zero, one could look at how slow the change in θ^g will be.

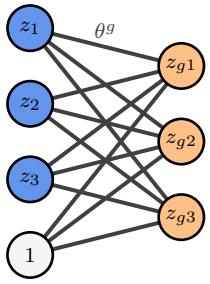
Here is a "trap" caused by the weakness of the discriminator. The discriminator forces the algorithm to stop improving too soon, so that the generated distribution is stuck at some point far away from the target. In a sense, the discriminator is convinced of the perfect job it has done; however, the reality is just the opposite. Even for partial learnability results (i.e. for some achievable ϵ) we see that a weak discriminator can make the generator's job arbitrarily slow. To conclude, the class of distributions $\mathcal{C}$ defined in Section 3.2.1 is not learnable in general by the generator of a shallow GAN, and even for achievable accuracies the efficiency of learning will be highly dependent on the strength of the discriminator.

4 Experiments

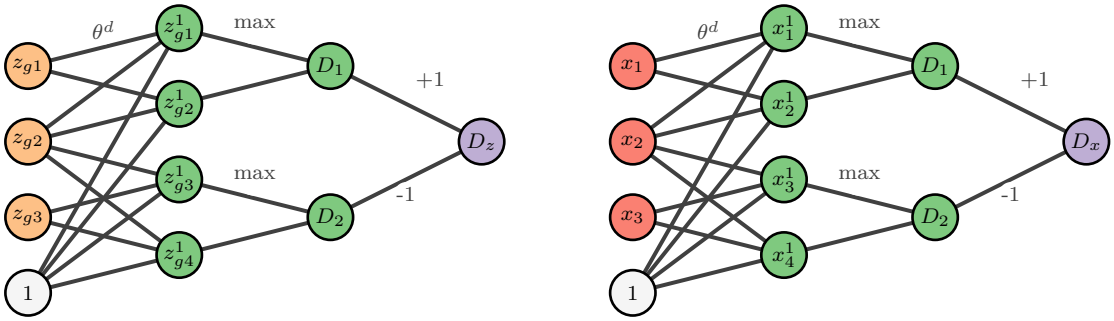
In this section, *first*, corresponding to Section 2.1 we present the architecture diagrams of three GAN models used in the experiments in Section 4.1; *second*, corresponding to Section 2.2 we set specific noise prior distributions and target distributions in Section 4.2; *third*, we introduce visualization tools in the experiments in Section 4.3, *last but not least*, we analyze in Section 4.4 the experiment results of different options or values of the parameters and hyper-parameters discussed in Section 2.4.

4.1 Architecture diagrams

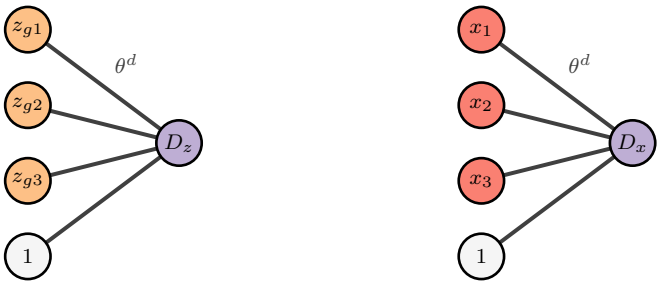
In experiments, we consider both examples and noise samples as datasets with three variables ($X = (X_1, X_2, X_3)^T$ and $Z = (Z_1, Z_2, Z_3)^T$). We set the generator to be a one-layer shallow neural network with sigmoid activation function in the output layer shown in the following diagram. This generator will be used in all of the following algorithms.



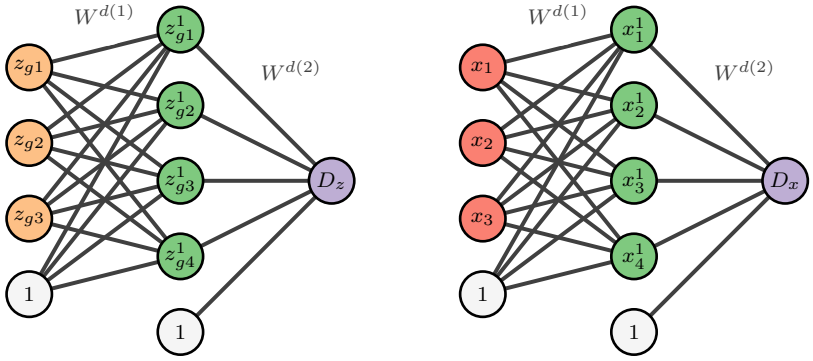
In Algorithm 1, we set the discriminator to be a one-layer shallow neural network with Maxout layer proposed by Goodfellow et al. in 2013 [29] followed by a sigmoid transformation as activation function in the output layer as shown here:



In Algorithm 2, we set the discriminator to be a one-layer perceptron with sigmoid activation function in the output layer as shown below:



In Algorithm 3, we introduce a hidden layer in the discriminator as below. Sigmoid activation function is used in both hidden layer and output layer.



4.2 Noise prior distributions and target distributions

We set noise prior distributions to be normal and sample noise Z from four specific prior distributions listed below:

	Distributions
noise prior 1 (p1)	$N(-1, 0.64)$
noise prior 2 (p2)	$N(2, 0.64)$
noise prior 3 (p3)	$N(-1, 0.09)$
noise prior 4 (p4)	$N(2, 0.09)$

Our target distributions are truncated normal and example X lies in the interval $(0, 1)$. We choose two specific distributions to be our targets listed below:

From each prior distribution, we sample a mini-batch of noise samples Z, based on which the algorithm generates samples to fit a target distribution. There are eight combinations of

	Distributions
target 1 (t1)	$N(0.5, 0.09)$ truncated to the range $(0, 1)$
target 2 (t2)	$N(0.5, 0.64)$ truncated to the range $(0, 1)$

priors and targets listed below. From prior 1 (p1) to target 1 (t1), the algorithm learns to generate samples distributed as a sharper shape and located on the right side based on noise samples with a flatter distribution and located on the left side.

	p1 → t1	p2 → t1	p3 → t1	p4 → t1	p1 → t2	p2 → t2	p3 → t2	p4 → t2
flat → sharp	✓	✓						
both sharp			✓	✓				
both flat					✓	✓		
sharp → flat							✓	✓
left → right	✓		✓		✓		✓	
right → left		✓		✓		✓		✓

4.3 Visualization tools

For observation and comparison of performance, we visualize performance of the algorithms using three groups of figures, which display density plots of data, outputs of the discriminators and ranges of the parameters. Furthermore, we formulate a unified evaluation standard to interpret the information in figures, as well as organize a clear analysis in different scenarios. Figures and descriptions are illustrated below followed by an evaluation standard.

4.3.1 Density plots

In each experiment, we will use a group of density plots to represent dynamic movements of the generated data z_g, noise samples z and examples x throughout training. Specifically, for each experiment a figure will show empirical density curves of a variable of z_g, z and x in an epoch. In each density plot, black curve, red curve and purple dashed curve represent target distribution p_{data} (density plot of examples x), noise prior distribution p_Z (density plot of noise samples z) and generated distribution p_G (density plots of generated samples z_g) respectively. In the upper right corner of a plot in an epoch, ranges of θ^d and θ^g in the

specific epoch are legended. We represent density plots in the last epoch when we discuss the results of the experiments.

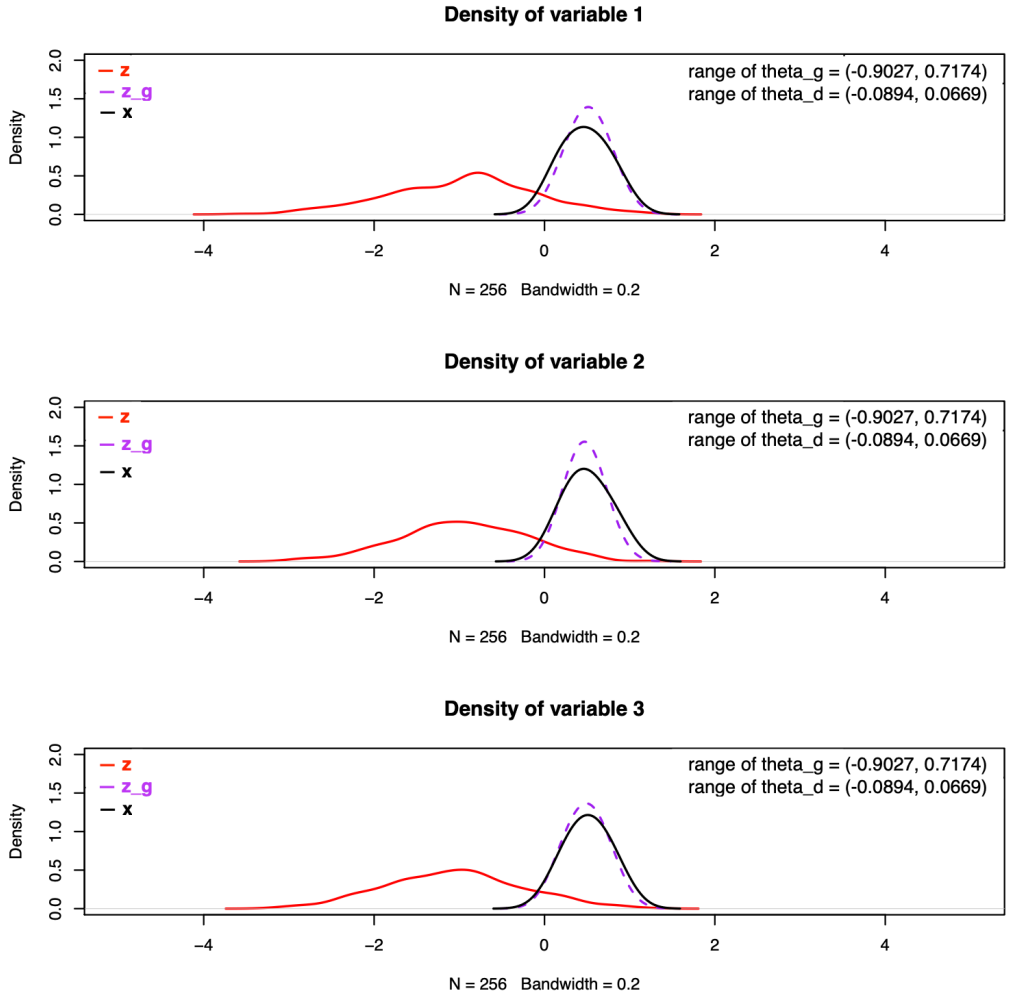


Figure 4

If the algorithm performs well, we believe that the density curves of all variables of z_g will overlap with the curves of x. Similarly, if an algorithm performs better than another, its curves of z_g will reach a closer position or have a more similar shape (e.g. unimodality) to the targets.

4.3.2 Outputs of the discriminators

Figure 5 displays outputs of a discriminator (red curve: D_x and black curve: D_z) throughout training. X-axis shows the epoch times and y-axis is labeled with values of

outputs of the discriminator from 0 to 1 and curves of D_x and D_z display fluctuations of the outputs.

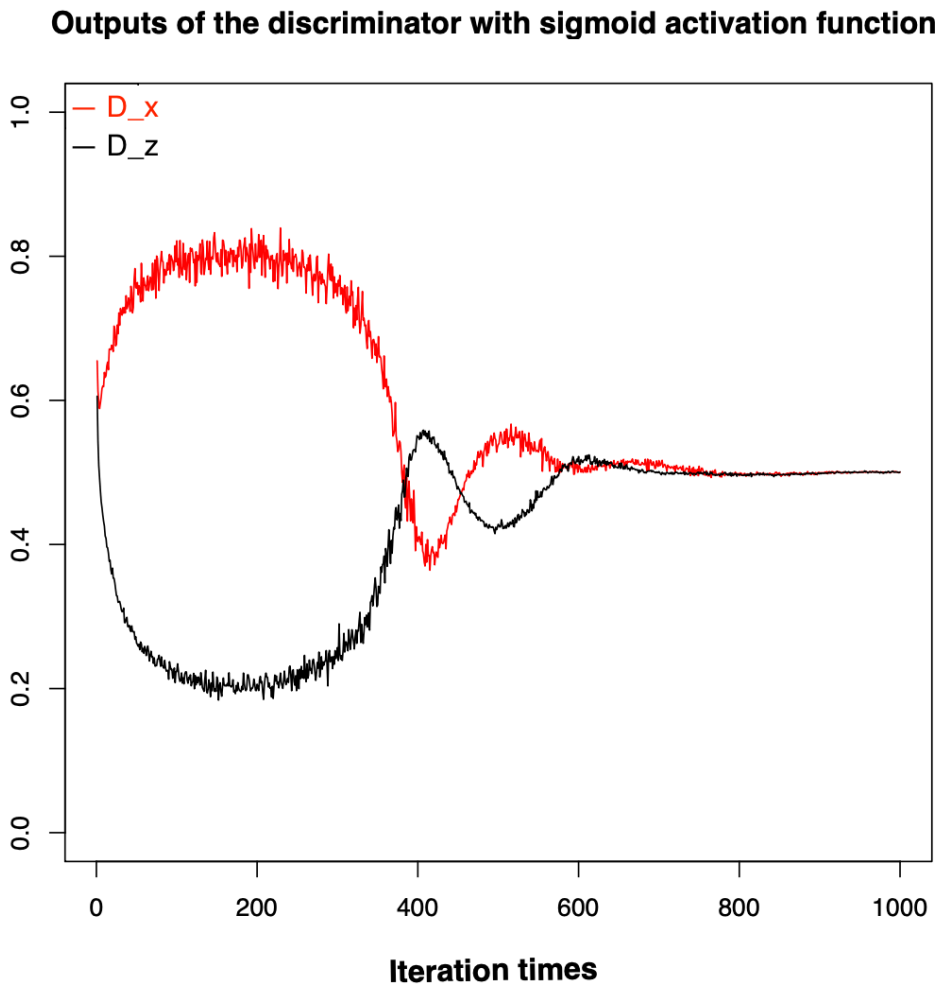


Figure 5

According to Goodfellow et al [1], D_x represents the probability that x comes from p_{data} rather than p_G. Similarly, D_z represents the probablity that z_g comes from p_{data} rather than p_G. Based on *Theorem 1* in [1], given a fixed generator, discriminator achieves its global optimal if and only if $p_G = p_{data}$, which is equivalent to $D_x = D_z = \frac{1}{2}$ based on *Proposition 1* in [1].

However, we observe that $p_G \neq p_{data}$ when $D_x = D_z = \frac{1}{2}$. Moreover, dynamic movement of the density curves of z_g throughout training is consistent with the fluctuations of D_x and D_z. Specifically, if D_x and D_z converge to one half, z_g will be stuck at some distribution

and fail to converge to target. By then, the algorithm achieves its optimum and cannot be optimized further.

Note that the algorithm must be given enough capacity and training time, so that the *Theorem 1* and *Proposition 1* in [1] hold. We know that the algorithm has been given enough training time when it achieves its best. Therefore, capacity of the discriminator might not be powerful enough, so that it cannot differentiate between real data and fake data. Additionally, if D_x and D_z fail to converge to one half in some cases, the algorithm should be given more training time to achieve its optimal performance.

4.3.3 Ranges of the parameters

There are two plots in Figure 6. The plot above displays fluctuation of the ranges of θ^g, which include minimum values (black curves), maximum values (red curves) and the differences between maximum and minimum values of θ^g (blue curves; we use ranges to represent the differences sometimes). The plot below displays fluctuation of the ranges of θ^d.

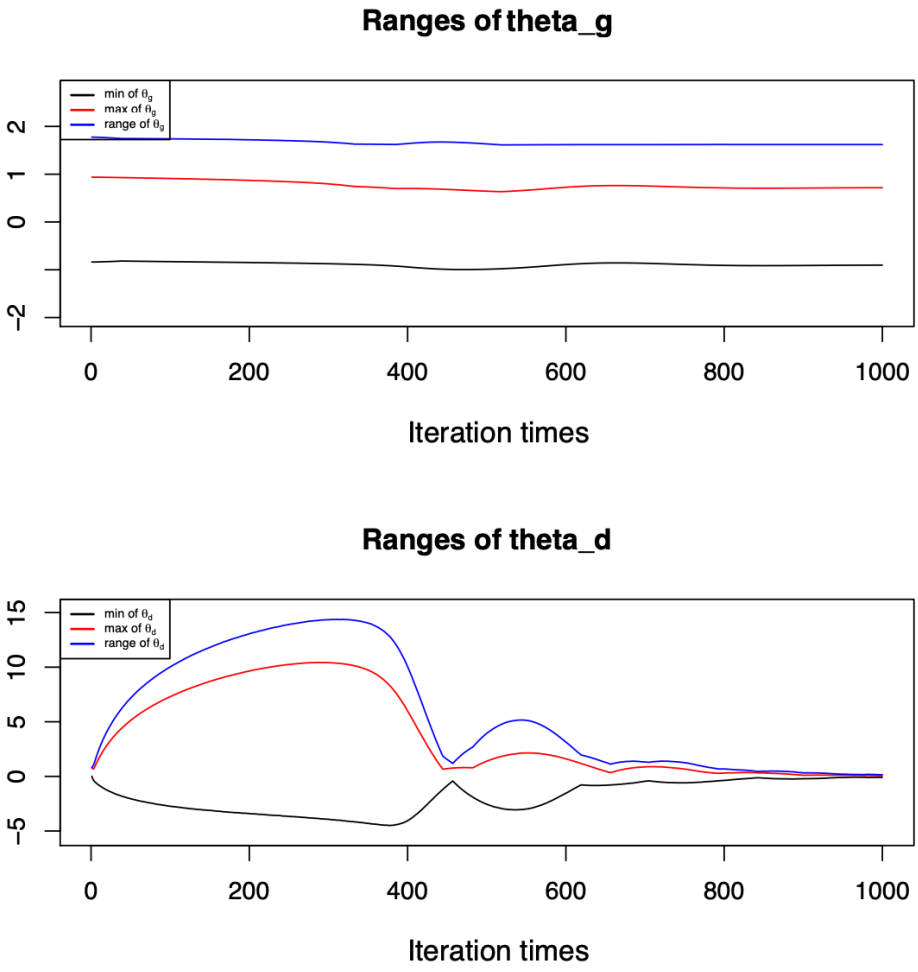


Figure 6

We explore fluctuations of the ranges of θ^d and θ^g throughout training. We alternate between L steps of updating discriminator and one step of updating generator. The updated parameters are a kind of expression of updated models. We use ranges (minimum, maximum and the difference) to illustrate the parameters of the algorithms. For the discriminator, fluctuations of the ranges of θ^d is consistent with the change of the outputs of the discriminator. To be specific, ranges of θ^d change drastically when D_x and D_z fluctuate significantly. For the generator, if the ranges of θ^g fluctuate distinctly in the epochs, density plots of z_g will also wave back and forth significantly.

4.4 Discussion of the experiments

In this section, we will discuss more on the parameters and hyper-parameters listed in Section 2.3 experimentally. We conduct experiments to explore their influence on shallow GANs.

4.4.1 Steps of optimizing the discriminator in each epoch

For the case that our target distribution is target 1 (t1), we use noise prior distribution prior 1 (p1) and set learning rates α of the discriminator and β of the generator to be fixed values. We choose three different seeds to sample different mini-batches of initial parameters (θ^{d0} and θ^{g0}), examples (x) and noise samples (z). Run each algorithm three times with different seeds. We choose different numbers of steps L to optimize the discriminator in the inner loop of training. Specific values are listed below.

Hyper-parameter	Option or Value
target distribution p_{data}	target 1 (t1)
noise prior distribution p_Z	prior 1 (p1)
size of mini-batch	$m = 256$
steps of optimizing discriminator	$L = \{1, 5, 10, 20, 50, 100, 200\}$
learning rate of θ^d	$\alpha = 0.1$
learning rate of θ^g	$\beta = 0.01$
range of initial θ^g	$[a, b] = [-1, 1]$
range of initial θ^d	$[c, d] = [-1, 1]$
seeds of mini-batch	$96, 54, 37$

We choose $L = 10$ which provides the discriminator with enough steps to update, so that the generator could receive enough information from the discriminator to optimize. It also prevents the discriminator from being optimized to completion.

4.4.2 Learning rates of the parameters

To learn the target t1, we choose prior distribution p1 to compare the effect of various learning rates on the performance of the algorithm. We set 4 options for both α and β listed below and conduct experiments to analyze all of the 16 combinations.

Hyper-parameter	Option or Value
target distribution p_{data}	target 1
noise prior distribution p_Z	prior 1
size of mini-batch	$m = 256$
steps of optimizing discriminator	$L = 10$
learning rate of θ^d	$\alpha = \{0.001, 0.01, 0.1, 0.6\}$
learning rate of θ^g	$\beta = \{0.001, 0.01, 0.1, 0.6\}$
range of initial θ^g	$[a, b] = [-1, 1]$
range of initial θ^d	$[c, d] = [-1, 1]$
seeds of mini-batch	$96, 54, 37$

Given enough training time, density plots of z_g perform similarly regardless of the values of the learning rates α and β. When α or β is 0.001, the algorithm requires too many epochs to achieve its best performance. When α is 0.1, the algorithm can reach the best performance within a proper number of epochs. Since values of θ^g are very likely to be small decimals, we choose a small β to avoid suboptimization. Therefore, we choose $\alpha = 0.1$ and $\beta = 0.01$ to use in the following experiments.

4.4.3 Activation functions in the discriminator

We present experiments to compare Algorithm 1 and 2 which use different activation functions in the discriminator. In Algorithm 1, the discriminator includes a Maxout layer with two Maxout units and two neurons in each Maxout unit. Since the loss function of GANs requires the output of the discriminator to be a vector with elements in $(0, 1)$, we further make an elementwise sigmoid transformation on the output of the Maxout layer.

In experiment, we use four noise prior distributions to learn two targets. We set some specific values of other parameters and hyper-parameters listed in the table below. Run 1000 epochs for each case and compare the performance of two algorithms within such epochs.

a. Algorithm with sigmoid activation function

The algorithm with sigmoid activation function (Algorithm 2) has the following performance. In all cases that use different noise prior distributions to learn different target distributions, the algorithm reaches or almost reaches its optimal performance within 1000 epochs. The

Hyper-parameter	Option or Value
target distribution p_{data}	both targets
noise prior distribution p_Z	all four priors
size of mini-batch	$m = 256$
steps of optimizing discriminator	$L = 10$
learning rate of θ^d	$\alpha = 0.1$
learning rate of θ^g	$\beta = 0.01$
range of initial θ^g	$[a, b] = [-1, 1]$
range of initial θ^d	$[c, d] = [-1, 1]$
seeds of mini-batch	$96, 54, 37$
epoch steps	$N = 1000$

performance is similar in all cases, and the only difference is that training time needed to reach its optimal performance varies in different cases with different seeds. We choose the case that using prior1 to learn target1 with seed1 to illustrate the performance. Density plots of z_g, outputs of the discriminator and ranges of the parameters are discussed below.

a.1 Density plots

Throughout training, density plots of the generated samples z_g move back and forth to fit the density plots of x. Most density curves of z_g stay close to the targets; however, they reach unimodal shapes whose peaks are higher than ones of the targets after 1000 epochs as shown in the figure below.

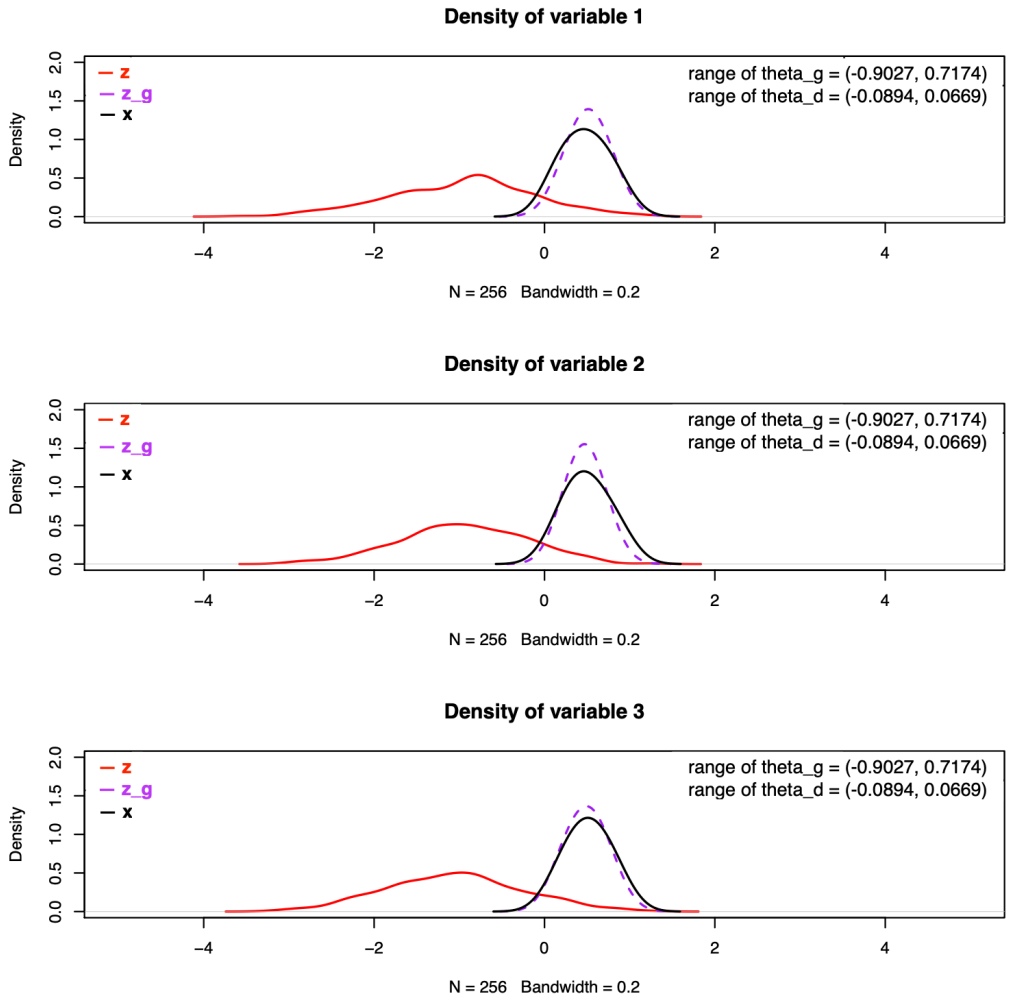


Figure 7

a.2 Discriminator

Outputs D_x and D_z converge to 0.5 in the end. Throughout training, D_x and D_z wave up and down when density curves of z_g swing back and forth significantly, which means that the fluctuations of D_z and D_x are consistent with the movement of z_g. Moreover, D_x and D_z are likely to fluctuate in opposite directions as shown in the figure below.

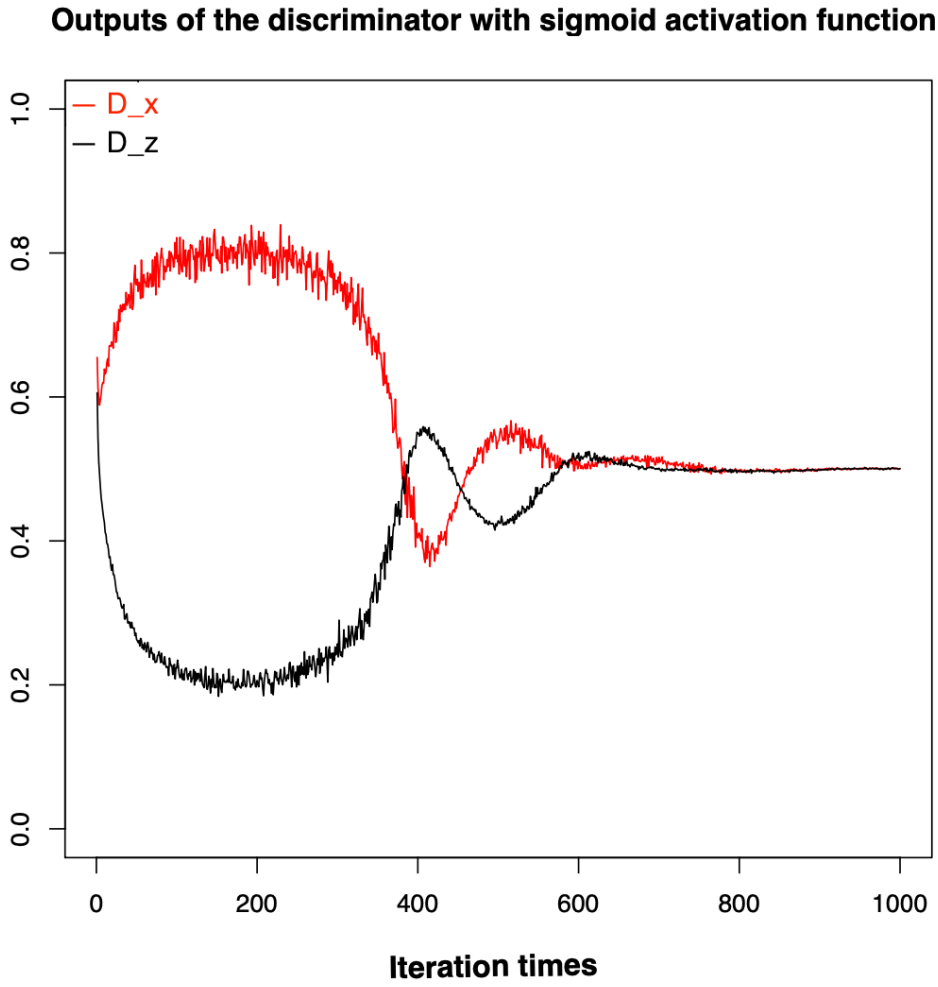


Figure 8

a.3 Ranges of the parameters

Fluctuation of the range of θ^d is consistent with the change of D_x and D_z. More specifically, when D_x or D_z becomes large, maximum value of θ^d becomes large and meanwhile minimum value becomes small, and vise versa. θ_d converges to 0 in the end. Range of θ^g fluctuates slightly and remains flat most time as shown in the figure below.

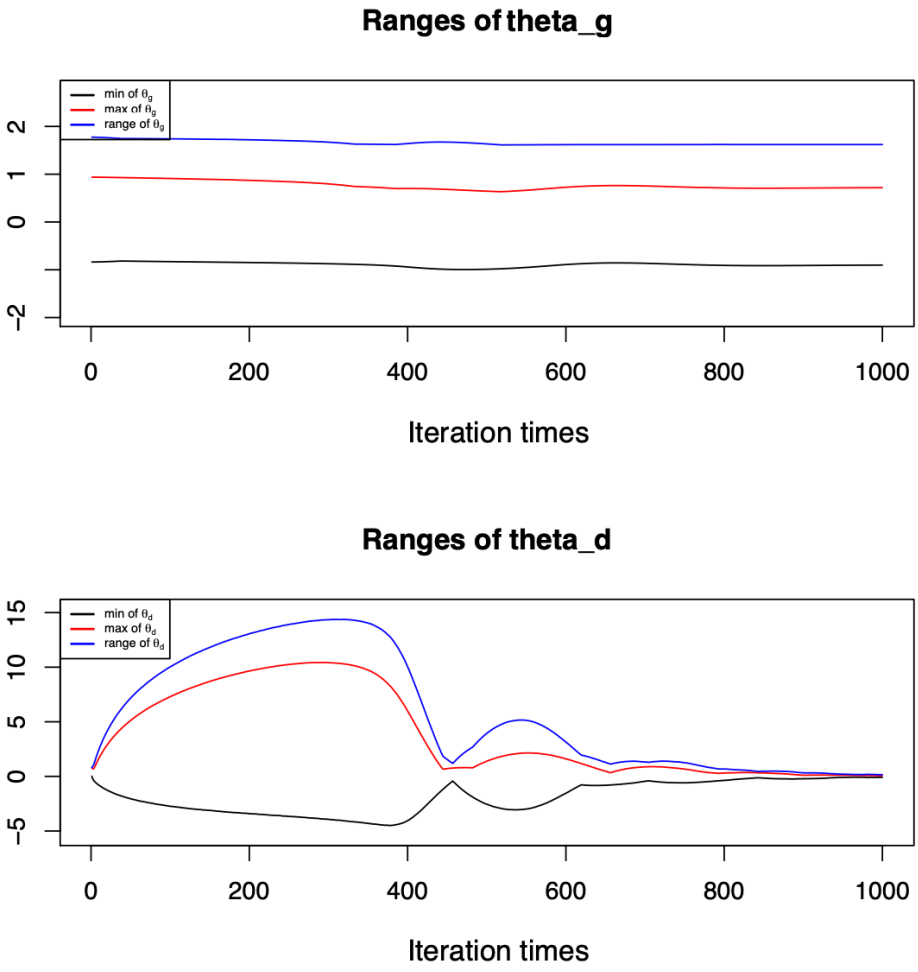


Figure 9

Based on the results above, in the discriminator, D_z and D_x converge to one half and θ^d converges to 0 in the end. In the generator, range of θ^g fluctuates slightly. Although density curves of z_g stay close to the curves of x, they merely converge to the targets. Considering both discriminator and generator, the discriminator has done its best since D_z and D_x converged to one half; however, density plots of z_g miss the targets. θ^d collapses to zero in the end, which results in the situation that the discriminator cannot be updated any more. Therefore, the discriminator might not be strong enough to distinguish fake data from real data, or to update itself throughout training. We will give a more specific explanation regarding these phenomena below.

b. Algorithm with Maxout layer

Performance of the algorithm with Maxout layer (Algorithm 1) varies from case to case. We summarize two scenarios below to illustrate its performance. We first discuss the cases that trained within 1000 epochs, and then discuss the cases given enough training time.

b.1 Within 1000 epochs

In most cases, this algorithm fails to achieve its best performance through 1000 epochs.

b.1.1 Scenario 1

Density plots of z_g stay close to the targets in the end, but have sharper unimodal shapes than targets and fail to move down. In these cases, D_z and D_x converge to one half in the end, and meanwhile ranges of θ^d become flat and do not collapse to zero. Figure 10 to Figure 12 are from the case: target1, prior1 and run1(seed1), denoted as t1p1r1. In this case, target distribution is sharper and located on the right side and noise prior distribution is flatter and located on the left side.

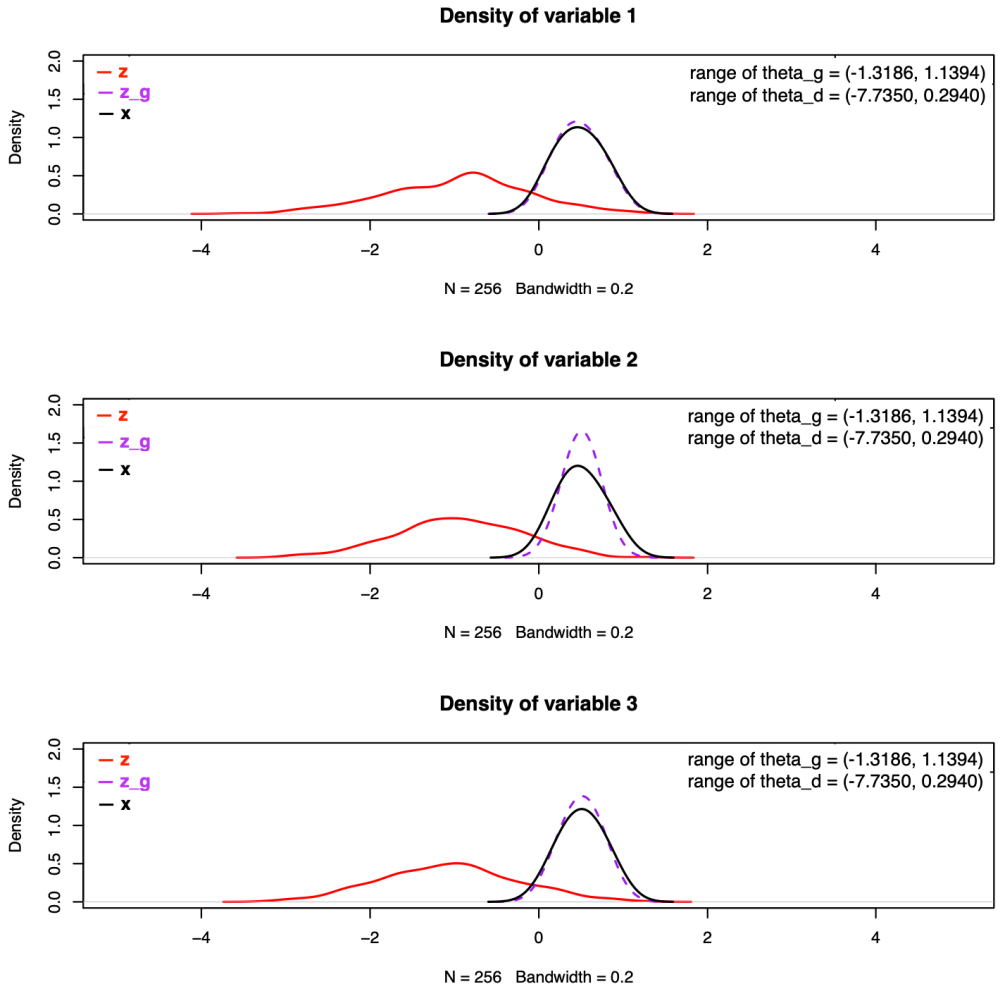

Density of variable 1
z
z_g
x
range of theta_g = (-1.3186, 1.1394)
range of theta_d = (-7.7350, 0.2940)
Density
0.0
0.5
1.0
1.5
2.0
−4
−2
0
2
4
N = 256 Bandwidth = 0.2
Density of variable 2
z
z_g
x
range of theta_g = (-1.3186, 1.1394)
range of theta_d = (-7.7350, 0.2940)
Density
0.0
0.5
1.0
1.5
2.0
−4
−2
0
2
4
N = 256 Bandwidth = 0.2
Density of variable 3
z
z_g
x
range of theta_g = (-1.3186, 1.1394)
range of theta_d = (-7.7350, 0.2940)
Density
0.0
0.5
1.0
1.5
2.0
−4
−2
0
2
4
N = 256 Bandwidth = 0.2

Figure 10

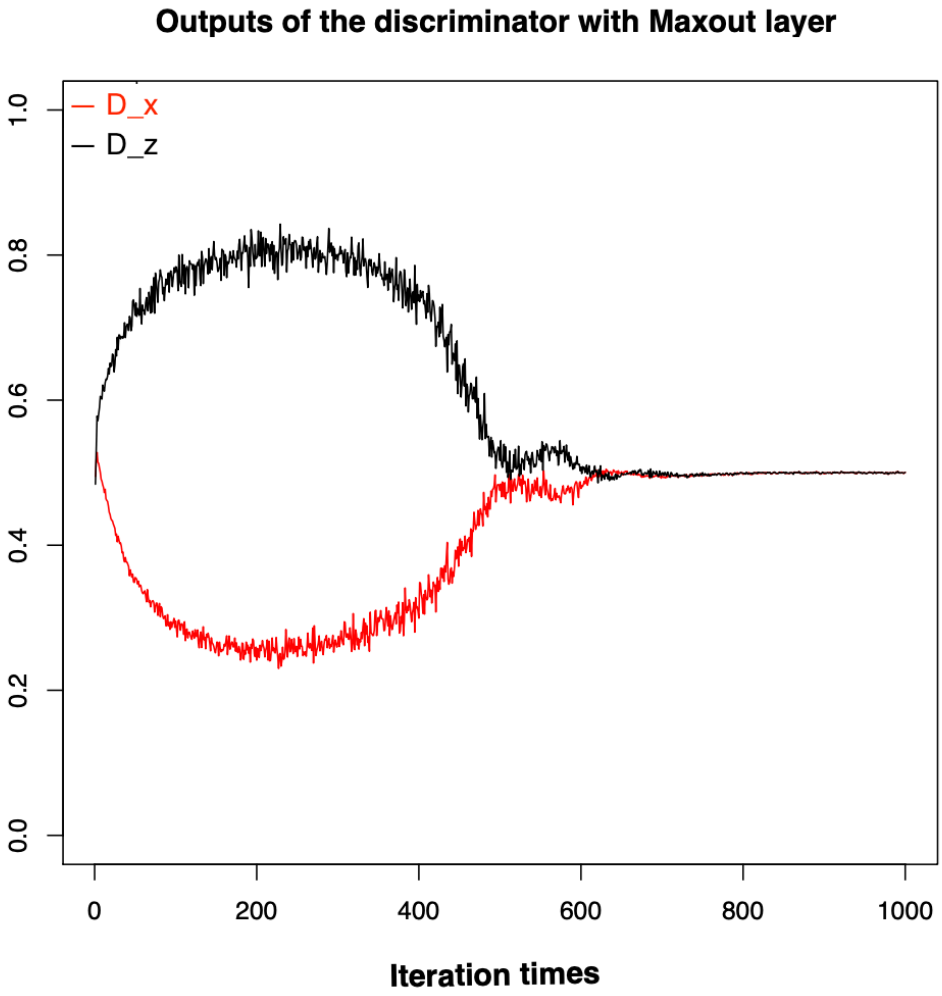

Outputs of the discriminator with Maxout layer
D_x
D_z
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000
Iteration times

Figure 11

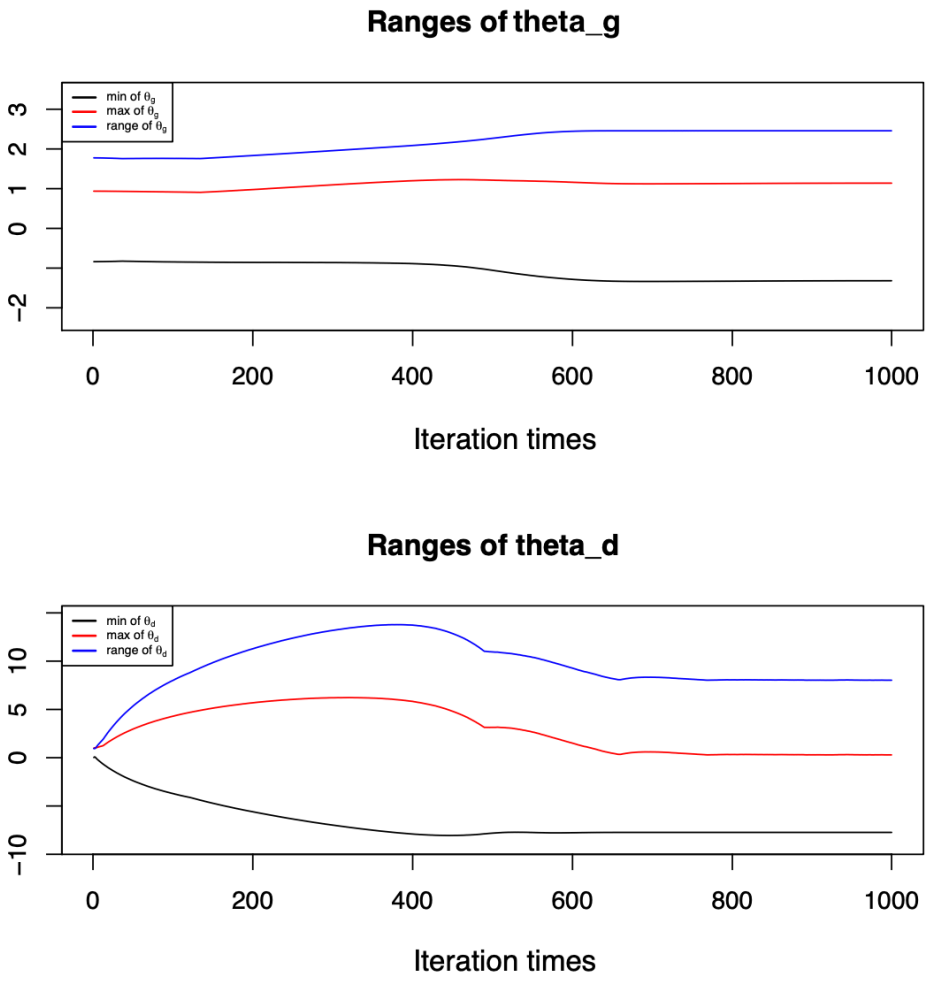


Figure 12

b.1.2 Scenario 2

Some variables of z_g are stuck at 0 or 1 and distributed sharper than targets; others behave similarly to the cases discussed in last scenario. D_z and D_x fail to converge to one half. In some cases, D_x and D_z are both less than one half and almost remain flat and parallel to each other; in other cases, D_x and D_z swing more or less than one half and cannot converge to 0. Ranges of θ^d fluctuate hugely when D_x and D_z wave significantly. Ranges of θ^d almost keep flat similar to last scenario. Figure 13 to Figure 15 are from the case: t1p4r2. In this case, target distribution and noise prior distribution are both sharp and target is located on the left side.

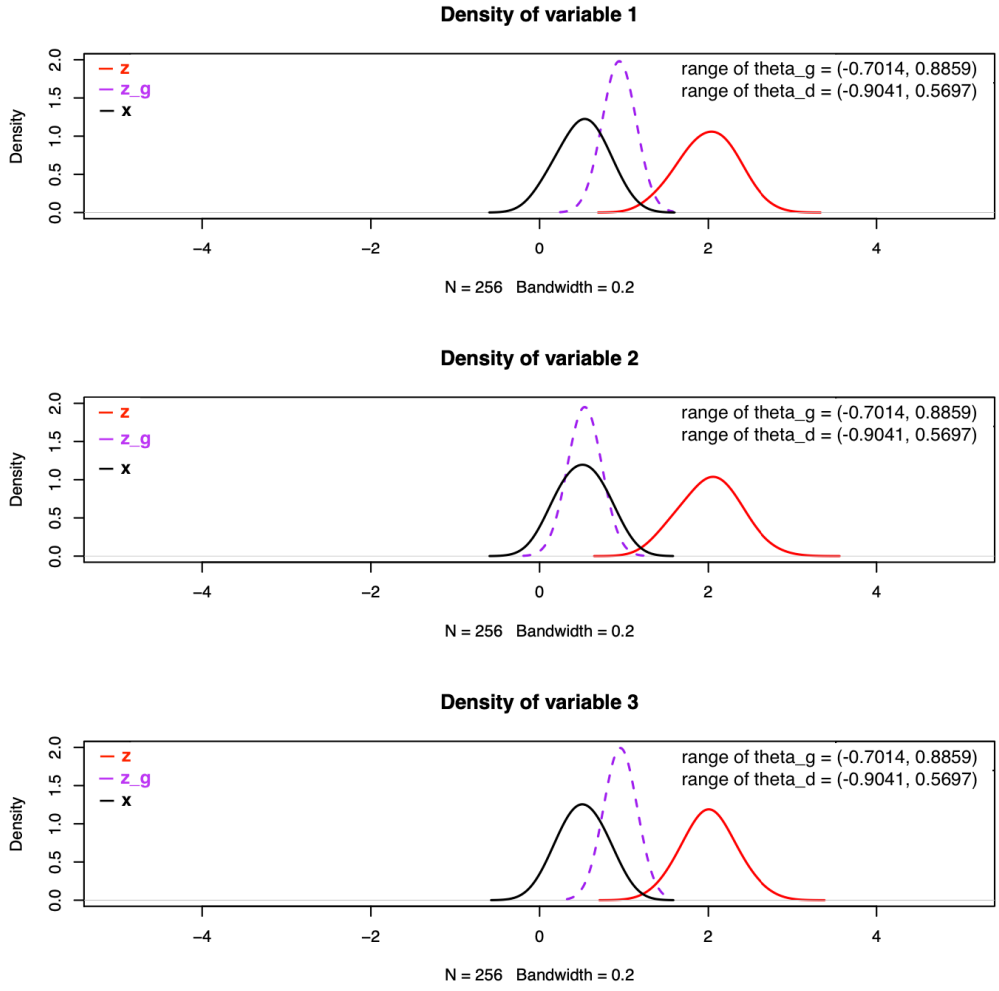

Density of variable 1
z
z_g
x
range of theta_g = (-0.7014, 0.8859)
range of theta_d = (-0.9041, 0.5697)
Density
0.0
0.5
1.0
1.5
2.0
−4
−2
0
2
4
N = 256 Bandwidth = 0.2
Density of variable 2
z
z_g
x
range of theta_g = (-0.7014, 0.8859)
range of theta_d = (-0.9041, 0.5697)
Density
0.0
0.5
1.0
1.5
2.0
−4
−2
0
2
4
N = 256 Bandwidth = 0.2
Density of variable 3
z
z_g
x
range of theta_g = (-0.7014, 0.8859)
range of theta_d = (-0.9041, 0.5697)
Density
0.0
0.5
1.0
1.5
2.0
−4
−2
0
2
4
N = 256 Bandwidth = 0.2

Figure 13

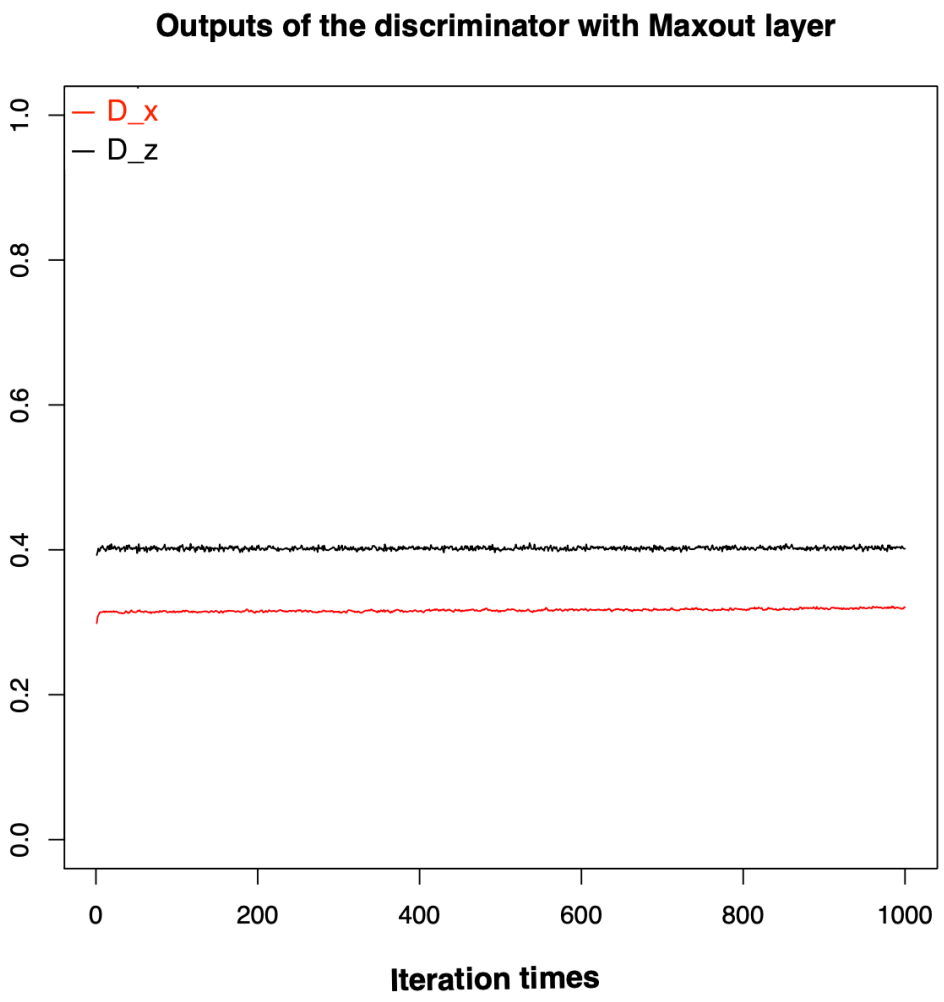

Outputs of the discriminator with Maxout layer
D_x
D_z
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000
Iteration times

Figure 14

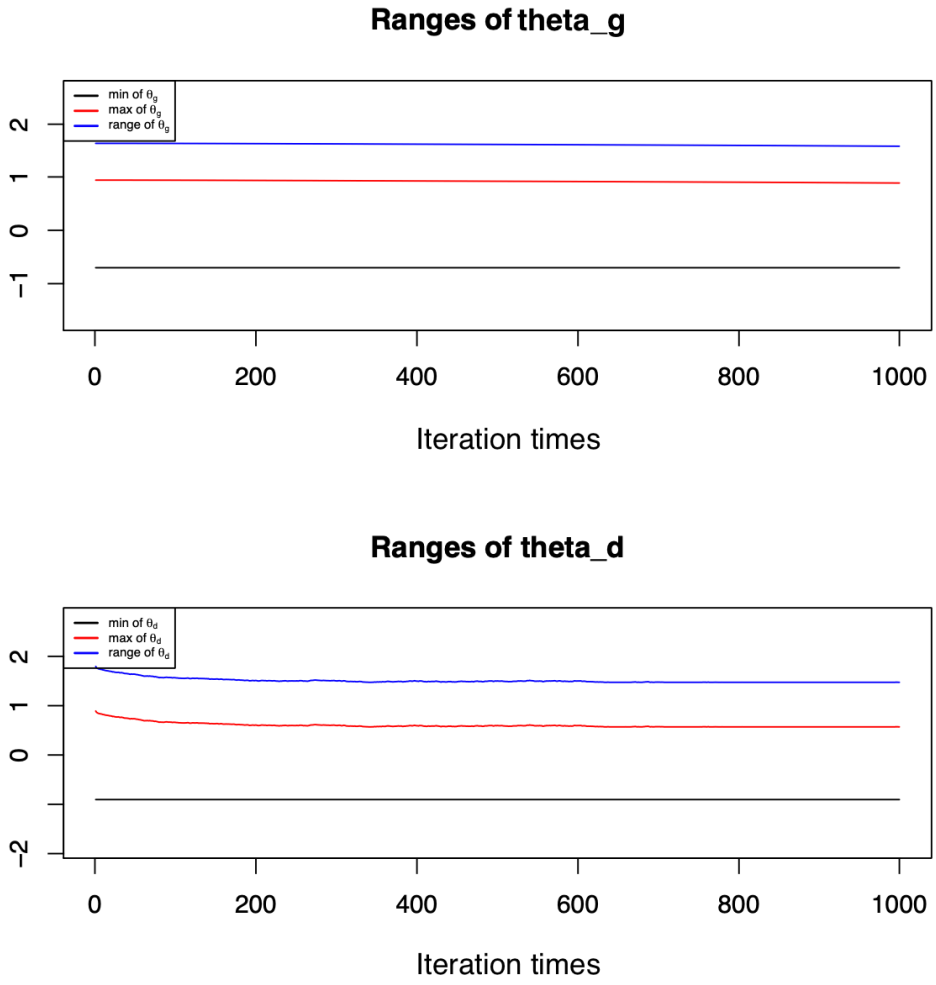


Figure 15

Figure 16 to Figure 18 are from the case: t1p4r3, which is another run of the previous case.

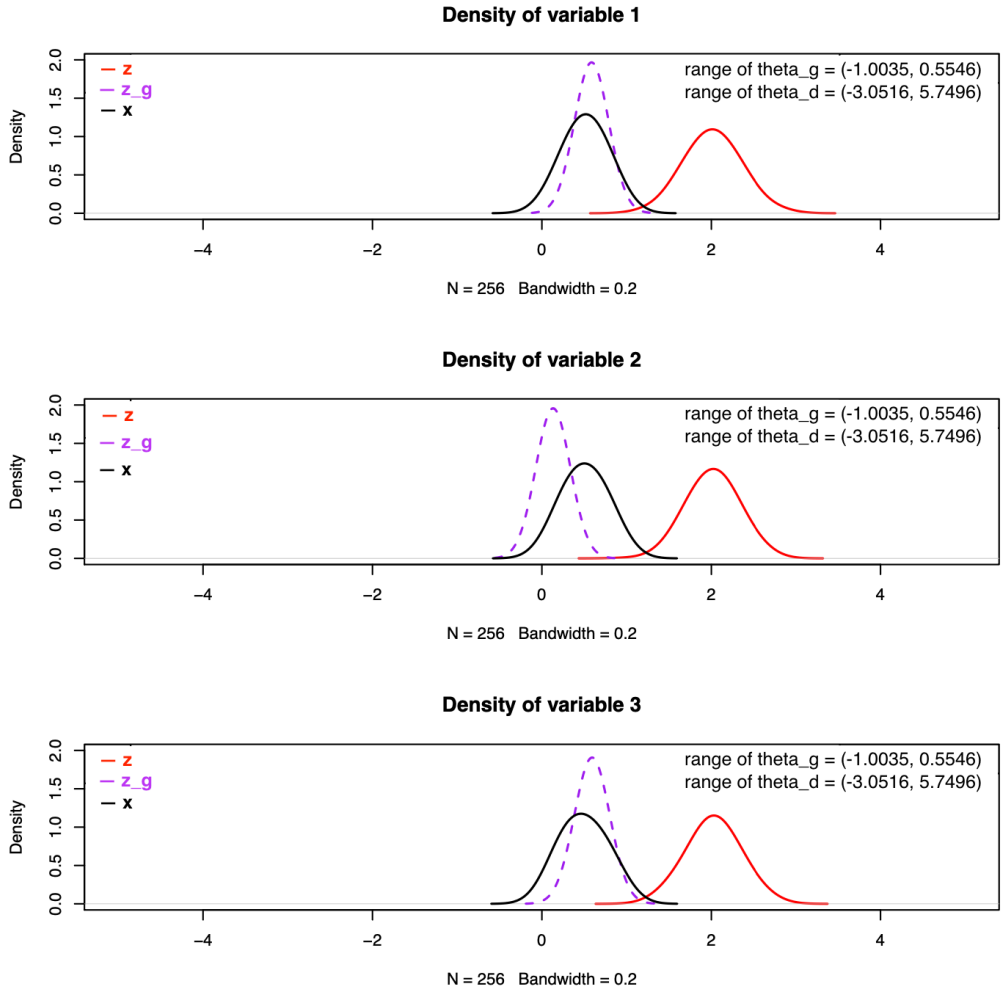

Density of variable 1
z
z_g
x
range of theta_g = (-1.0035, 0.5546)
range of theta_d = (-3.0516, 5.7496)
Density
0.0
0.5
1.0
1.5
2.0
−4
−2
0
2
4
N = 256 Bandwidth = 0.2
Density of variable 2
z
z_g
x
range of theta_g = (-1.0035, 0.5546)
range of theta_d = (-3.0516, 5.7496)
Density
0.0
0.5
1.0
1.5
2.0
−4
−2
0
2
4
N = 256 Bandwidth = 0.2
Density of variable 3
z
z_g
x
range of theta_g = (-1.0035, 0.5546)
range of theta_d = (-3.0516, 5.7496)
Density
0.0
0.5
1.0
1.5
2.0
−4
−2
0
2
4
N = 256 Bandwidth = 0.2

Figure 16

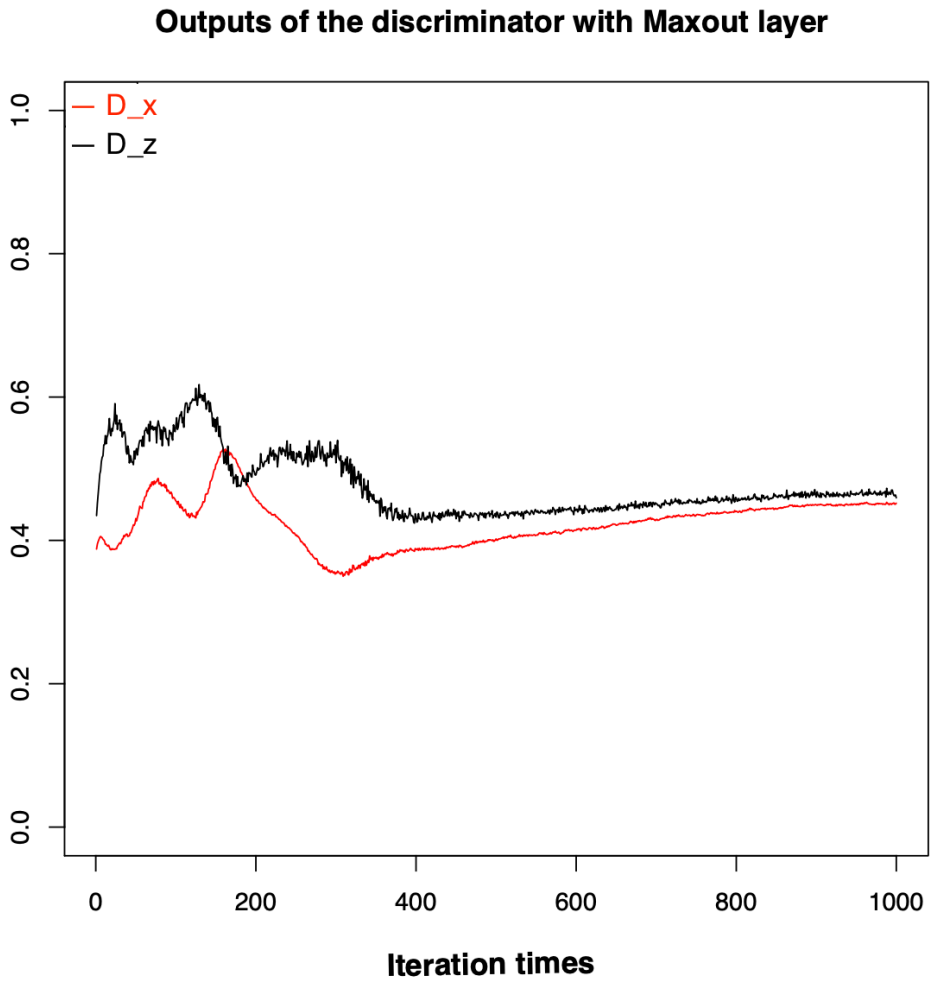

Outputs of the discriminator with Maxout layer
D_x
D_z
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000
Iteration times

Figure 17

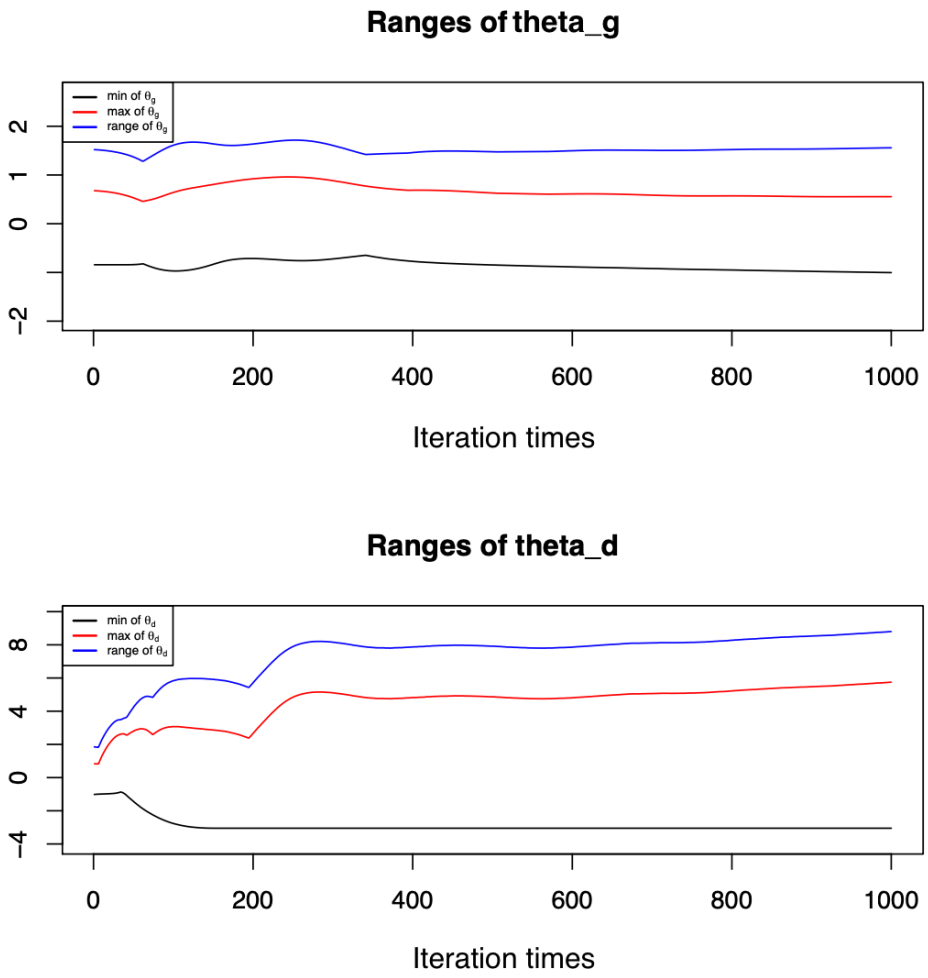


Figure 18

Ranges of θ^g remain flat and wave slightly in all scenarios.

In the first scenario, the algorithm achieves its best performance. In this case (t1p1r1), density plots of z_g and outputs of the discriminator (D_x and D_z) behave similarly to the ones in Algorithm 2. Both algorithms reach their best within around 700 epochs in the same case (t1p1r1). The only difference is parameters of the discriminator. θ^d of the Algorithm 1 does not collapse to zero throughout training.

In the second scenario, the algorithm fails to achieve its best. Since outputs of the discriminator and ranges of the parameters have various behaviors, it is challenging for us to analyze their patterns of fluctuation. Therefore, we run Algorithm 1 more epochs ($N = 5000$) for two cases visualized above to get its best performance.

b.2 Given enough training time

b.2.1 Scenario 1

In the case t1p4r2, density plots of z_g perform better than before and D_x and D_z converge to one half gradually throughout training. Ranges (maximum, minimum and their difference) of θ^d and θ^g almost remain flat. When D_x and D_z increase between 2000 and 3000 epochs, maximum and minimum values of θ^g decrease and their difference increases.

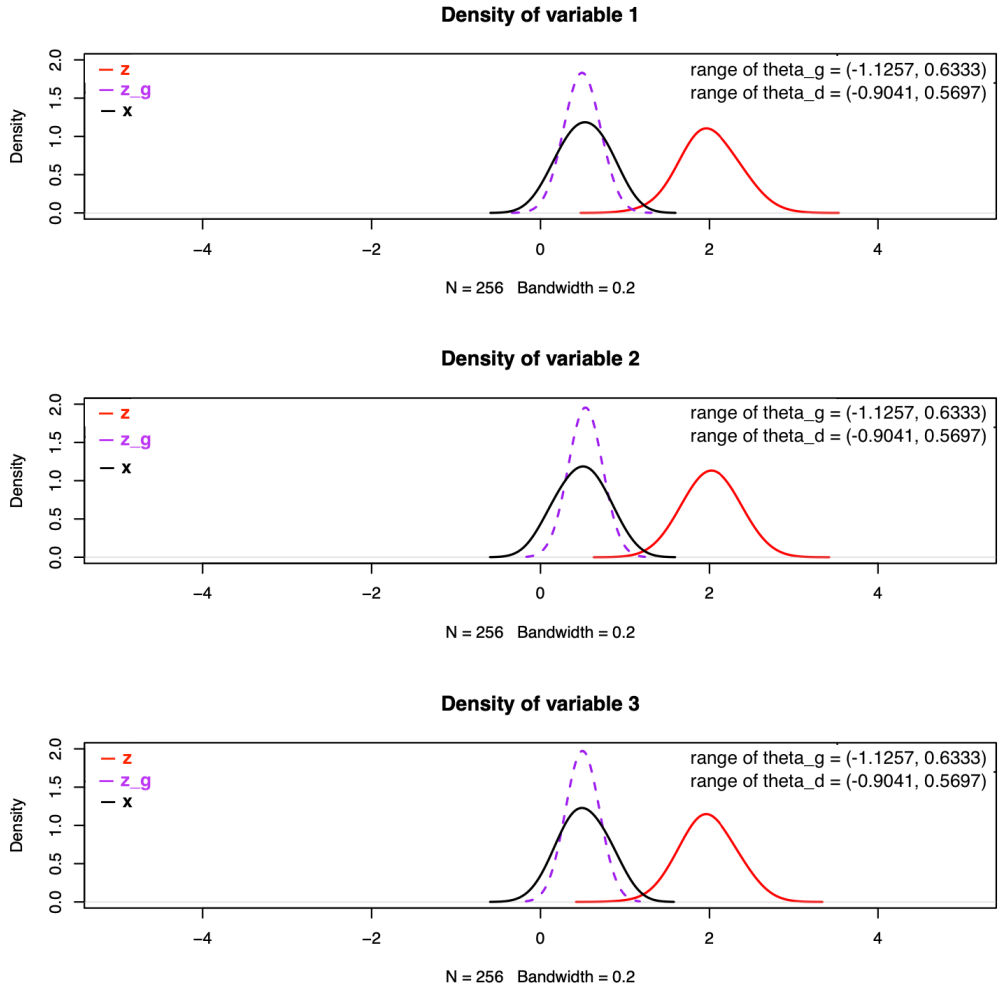


Figure 19

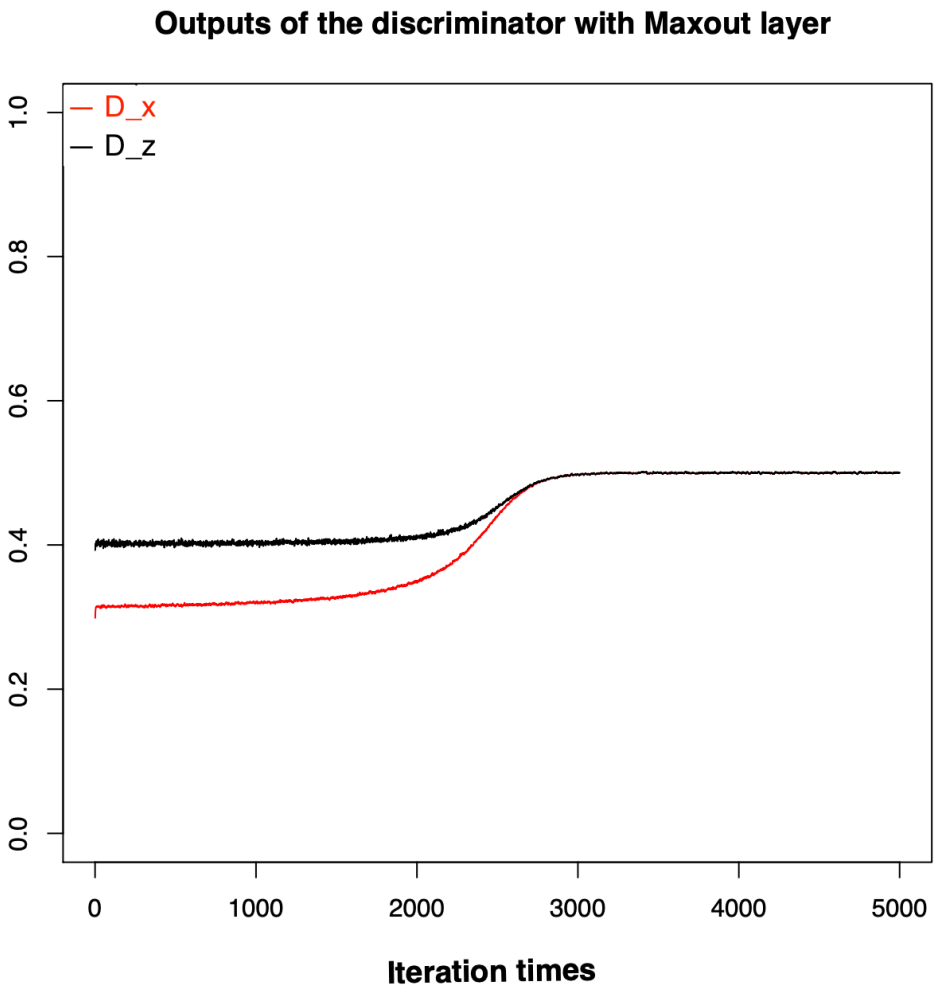

Outputs of the discriminator with Maxout layer
D_x
D_z
1.0
0.8
0.6
0.4
0.2
0.0
0
1000
2000
3000
4000
5000
Iteration times

Figure 20

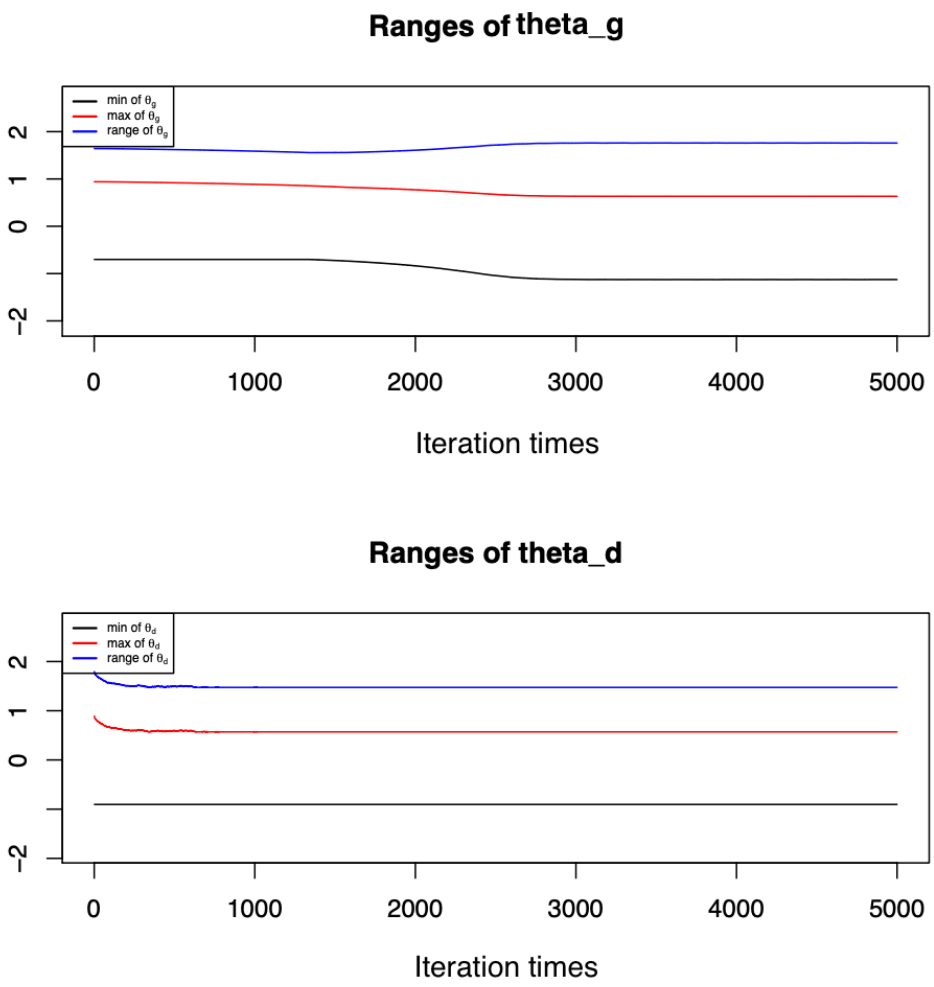


Figure 21

b.2.2 Scenario 2

In the case t1p4r3, D_x and D_z converge to one half through training; however, variable 2 of z_g is still stuck at 1 and far away from the target. Ranges of θ^d and θ^g fluctuate significantly when values of D_x and D_z change substantially between 1000 and 2000 epochs.

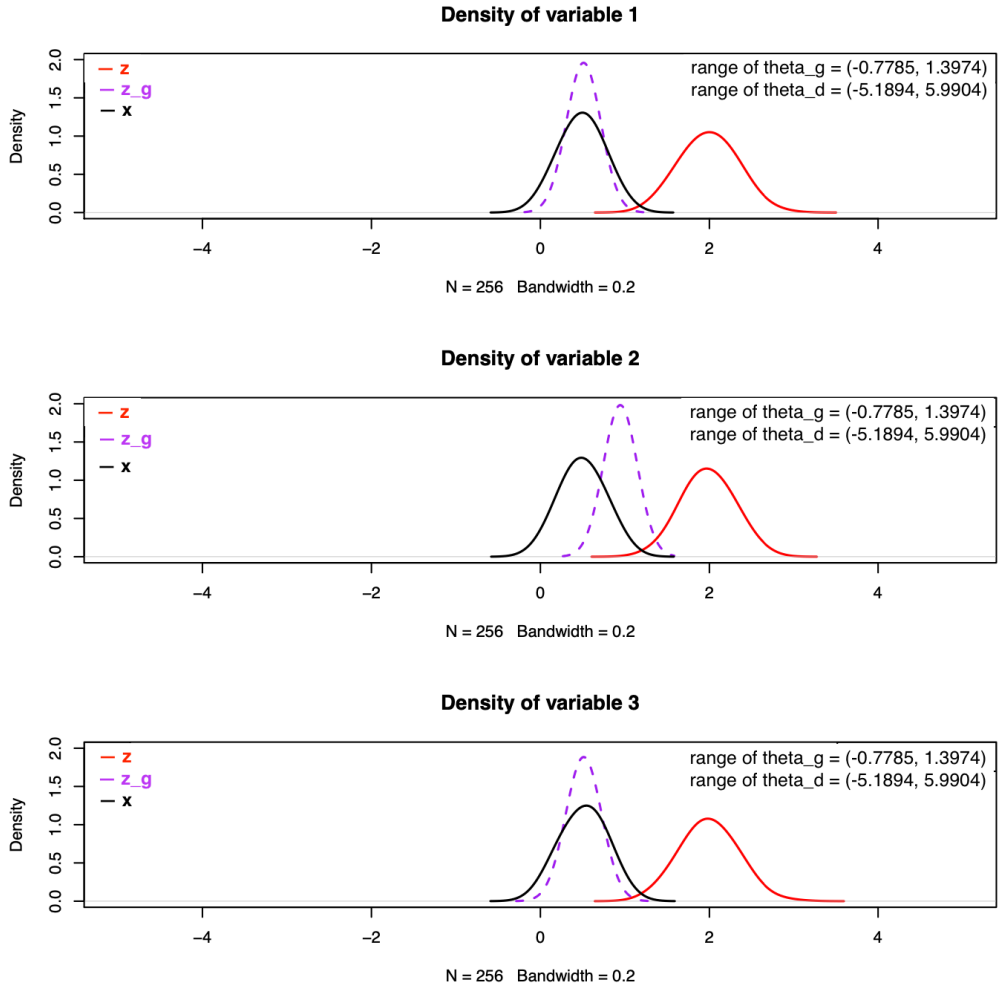

Density of variable 1
z
z_g
x
range of theta_g = (-0.7785, 1.3974)
range of theta_d = (-5.1894, 5.9904)
Density
0.0
0.5
1.0
1.5
2.0
−4
−2
0
2
4
N = 256 Bandwidth = 0.2
Density of variable 2
z
z_g
x
range of theta_g = (-0.7785, 1.3974)
range of theta_d = (-5.1894, 5.9904)
Density
0.0
0.5
1.0
1.5
2.0
−4
−2
0
2
4
N = 256 Bandwidth = 0.2
Density of variable 3
z
z_g
x
range of theta_g = (-0.7785, 1.3974)
range of theta_d = (-5.1894, 5.9904)
Density
0.0
0.5
1.0
1.5
2.0
−4
−2
0
2
4
N = 256 Bandwidth = 0.2

Figure 22

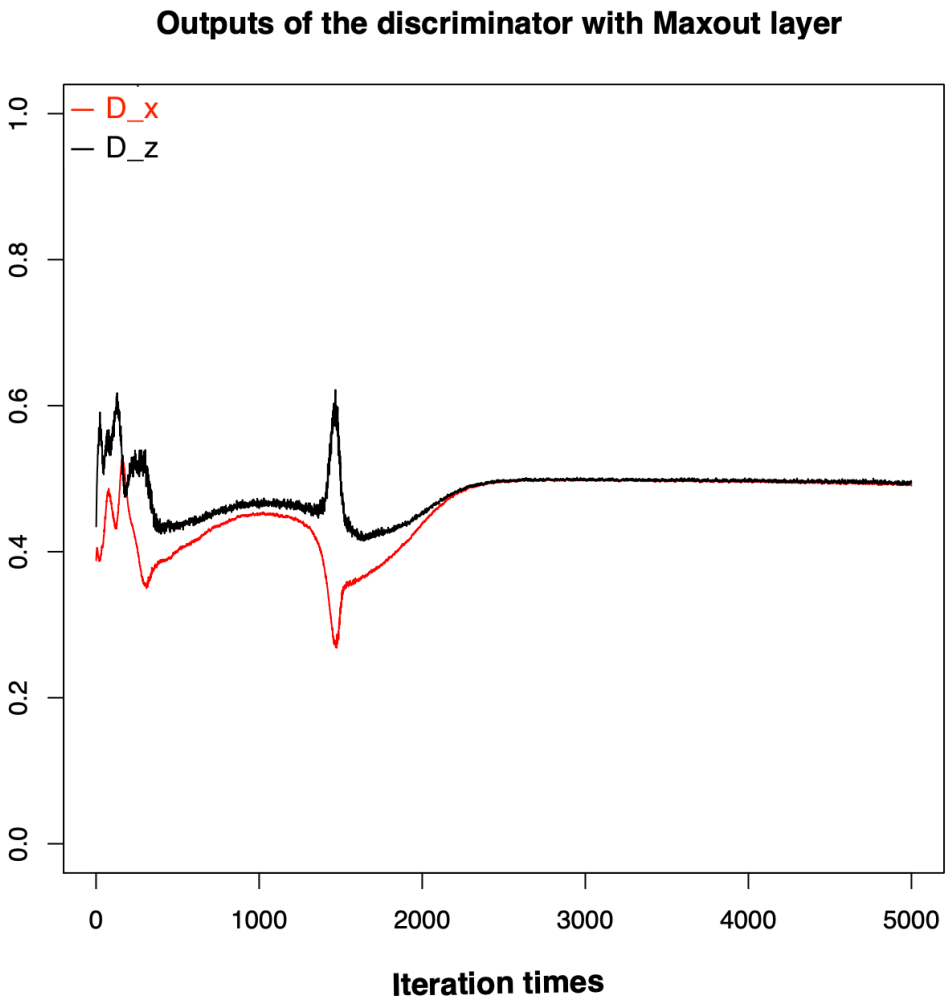

Outputs of the discriminator with Maxout layer
D_x
D_z
1.0
0.8
0.6
0.4
0.2
0.0
0
1000
2000
3000
4000
5000
Iteration times

Figure 23

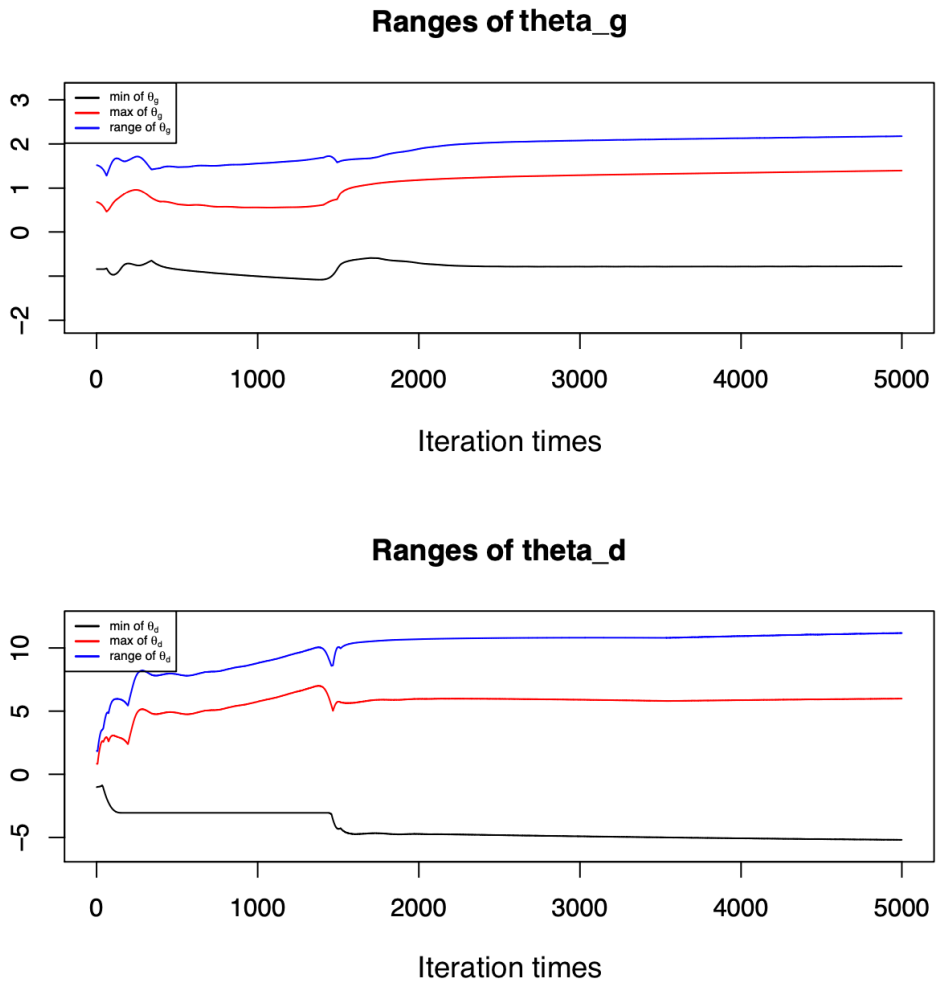


Figure 24

Both cases once again show the consistence among three groups of figures. They use same noise prior distribution to learn same target distribution, and the only difference is mini-batches are sampled from different seeds. However, one of them reflects the algorithm's ability to learn the target's location, but the other one does not. Therefore, the performance of Algorithm 1 is very unstable and highly dependes on the mini-batch of data. We will discuss this pathological performance more concretely below.

c. Comparision of two activation functions

In general, sigmoid activation function performs better than Maxout in shallow GANs when target only contains a few variables. Based on the results we observed above, there

are several pathological phenomena of the algorithms. We explore the crux that causes each phenomenon and propose potential solutions as below.

Generated samples are stuck at around zero or one and fail to move towards the targets through training. In Algorithm 1, in some cases, density plots of z_g are stuck at around zero or one (Figure 19) after D_x and D_z converged to one half (Figure 20). Since this phenomenon only happens in Algorithm 1, we conjecture that it is because of the sparsity introduced along with Maxout layer in the discriminator. To fix this problem, we could add another fully connected layer to the discriminator. Due to limited time, we did not conduct experiments for it. In the following, we give priorities to the discriminators using sigmoid activation functions, since there are more pathological phenomena we want to explore with regard to it.

The algorithms can learn the target distribution's location, but fail to learn the target's squared scale. In both algorithms, density plots of z_g distribute as unimodal shapes sharing symmetric axises with the targets (Figure 4, Figure 7, Figure 16), but are sharper than targets after D_x and D_z converged to one half (Figure 5, Figure 8, Figure 17). Therefore, we identify that the discriminators in both algorithms lack the ability to distinguish fake data from real data. Intuitively, we blame it on the structures of the discriminators which might be too simple to support a powerful discriminating capacity. To solve the problem, we consider to replace the discriminators by multi-layer perceptrons with more complex structures, for example, introducing hidden layers in the discriminators.

Parameters of the discriminator collapse to zero through training. In Algorithm 2, when outputs D_x and D_z of the discriminator converge to 0.5, θ^d always converges to 0. In the discriminator, there is a very simple structure which makes it equivalent to logistic regression. $D_x = S(w_1x_1 + w_2x_2 + w_3x_3 + b)$ and $D_z = S(w_1z_{g1} + w_2z_{g2} + w_3z_{g3} + b)$, in which D_x (respectively D_z) are outputs (dependent variables) and x (respectively z_g) are inputs (independent variables). When dependent variables converge to 0.5, the easiest solution for both regression functions is $w_1, w_2, w_3, b = 0$ since $0.5 = S(0)$. Through training, the algorithm tends to choose the easiest way to update the discriminator; however, the

discriminator loses its discriminating ability when θ^d converges to 0. This is exactly the "trap" discussed in Section 3.2.3. Against this problem, replacing the shallow discriminator by the one with a more complex structure is a potential solution. We will introduce a hidden layer in the discriminator (see Section 4.4.5) and compare it to the discriminator without hidden layers.

Ranges of θ^g keep almost flat. (See Figure 9 or Figure 15.) Ranges of θ^g only show a slight wave throughout training, which means that if we sample θ^{g0} from an interval (e.g. [-1,1]), it is highly likely that θ^g in the last epoch will still be in this interval. Since we sample the initial θ^g and θ^d from same intervals in all cases, we would like to try more intervals to sample the initial parameters. Then we could have a better understanding of the influence of ranges of the initial parameters on the fluctuations of ranges of the parameters throughout training and on the performance of the algorithm. We will discuss it more concretely in Section 4.4.4.

4.4.4 Ranges of the initial parameters

We sample the initial parameters in Algorithm 2 from different ranges and compare their influences on the performance of the algorithm. We choose three ranges which are intervals sharing a common center 0 but having different lengths listed below.

Hyper-parameter	Option or Value
target distribution p_{data}	two targets
noise prior distribution p_Z	four priors
size of mini-batch	$m = 256$
steps of optimizing discriminator	$L = 10$
learning rate of θ^d	$\alpha = 0.1$
learning rate of θ^g	$\beta = 0.01$
range of initial θ^g	$[a, b] = \{[-1, 1], [-2, 2], [-3, 3]\}$
range of initial θ^d	$[c, d] = \{[-1, 1], [-2, 2], [-3, 3]\}$
seeds	$96, 54, 37$

Given different patterns of flunctuations of the ranges of θ^d and θ^g observed above, θ^d always varies in a much wider range than θ^g. For example, in Figure 9 range of θ^g is about 2 throughout training (blue curve in the upper plot); however, range of θ^d increases up to about

15 and then decreases down to 0 (blue curve in the upper plot). Therefore, we conjecture that the range of θ^{d0} will not be a factor as important as the range of θ^{g} under this setting.

a. Range of θ^{d0}

In each case using a noise prior distribution to a target distribution, for a fixed range of θ^{g0}, different ranges of θ^{d0} do not significantly influence the results. See the figures (of the case: target1, prior1, run1, $\theta^{g0} \in [-1, 1]$, denoted as t1p2r1G1) below. In this case, target distribution is sharper and located on the left side and noise prior distribution is flatter and located on the right side. In each group of three figures below, θ^{d0} are sampled from $[-1, 1]$, $[-2, 2]$ and $[-3, 3]$ respectively.

a.1 Density plots

Density plots of each variable of z_g in three figures have similar shapes, which means that different ranges of θ^{d0} do not significantly influence the results of the generated data.

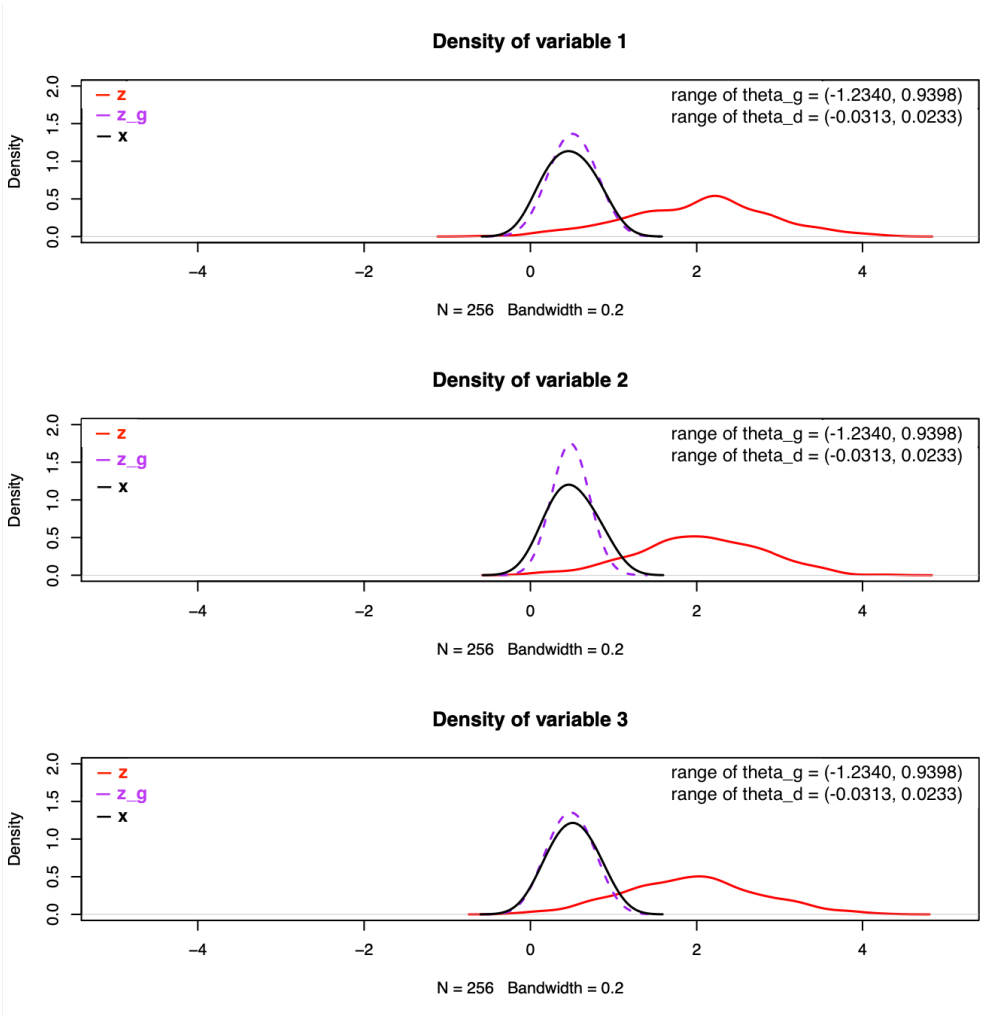

Density of variable 1
z
z_g
x
range of theta_g = (-1.2340, 0.9398)
range of theta_d = (-0.0313, 0.0233)
Density
0.0
0.5
1.0
1.5
2.0
−4
−2
0
2
4
N = 256 Bandwidth = 0.2
Density of variable 2
z
z_g
x
range of theta_g = (-1.2340, 0.9398)
range of theta_d = (-0.0313, 0.0233)
Density
0.0
0.5
1.0
1.5
2.0
−4
−2
0
2
4
N = 256 Bandwidth = 0.2
Density of variable 3
z
z_g
x
range of theta_g = (-1.2340, 0.9398)
range of theta_d = (-0.0313, 0.0233)
Density
0.0
0.5
1.0
1.5
2.0
−4
−2
0
2
4
N = 256 Bandwidth = 0.2

Figure 25

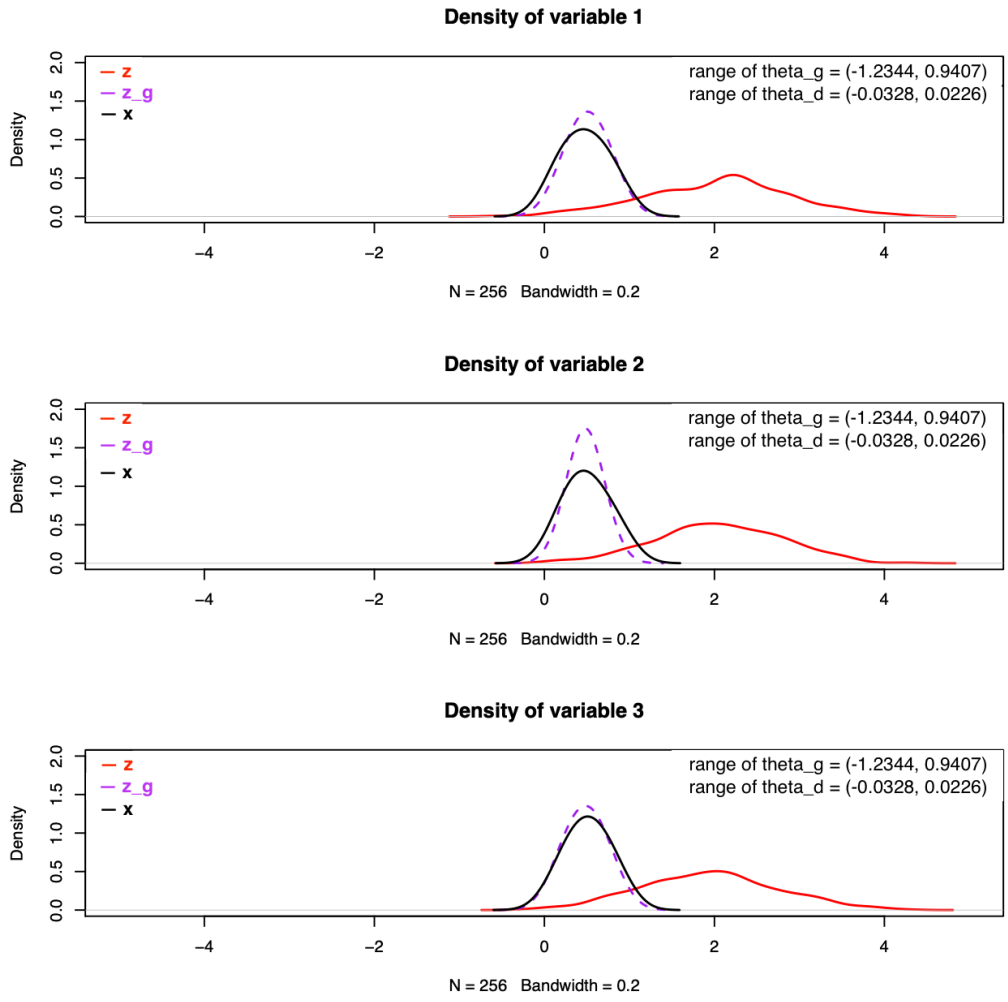

Density of variable 1
z
z_g
x
range of theta_g = (-1.2344, 0.9407)
range of theta_d = (-0.0328, 0.0226)
Density
0.0
0.5
1.0
1.5
2.0
−4
−2
0
2
4
N = 256 Bandwidth = 0.2
Density of variable 2
z
z_g
x
range of theta_g = (-1.2344, 0.9407)
range of theta_d = (-0.0328, 0.0226)
Density
0.0
0.5
1.0
1.5
2.0
−4
−2
0
2
4
N = 256 Bandwidth = 0.2
Density of variable 3
z
z_g
x
range of theta_g = (-1.2344, 0.9407)
range of theta_d = (-0.0328, 0.0226)
Density
0.0
0.5
1.0
1.5
2.0
−4
−2
0
2
4
N = 256 Bandwidth = 0.2

Figure 26

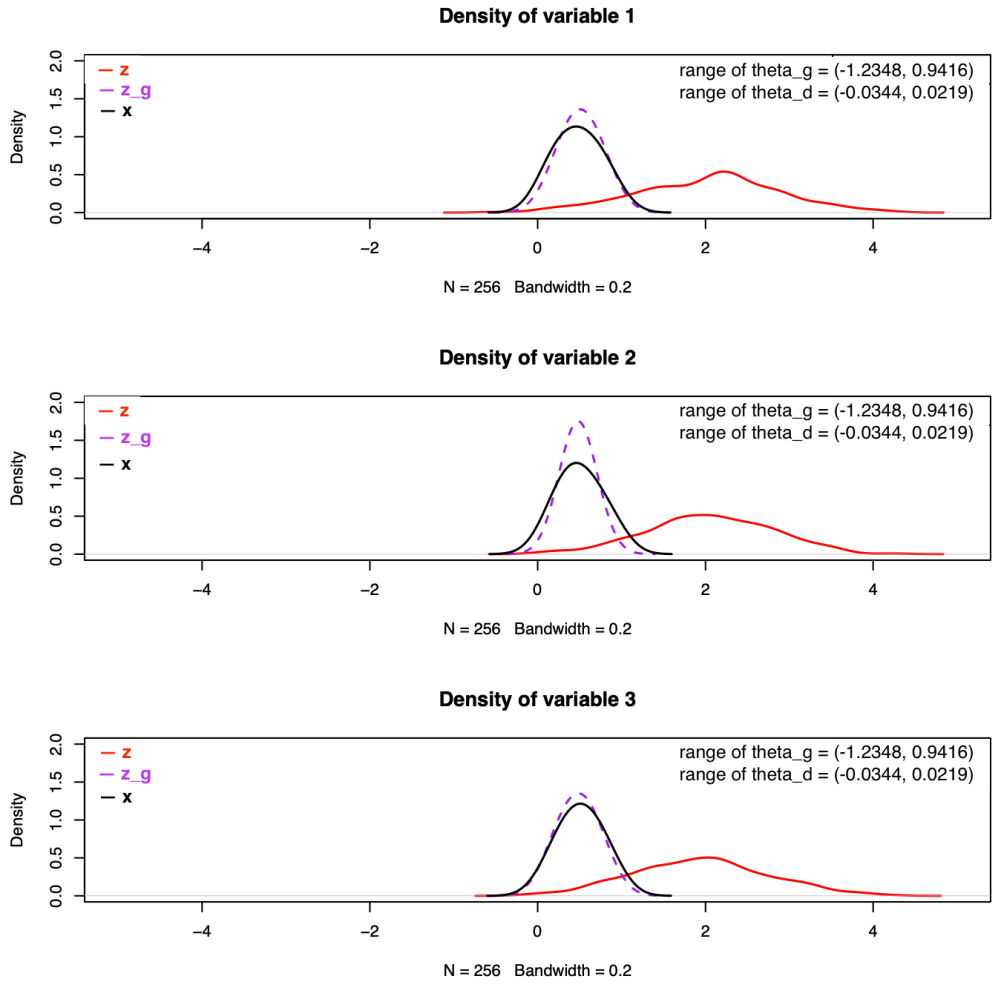


Figure 27

a.2 Discriminator

Outputs of the discriminators are highly similar. They require similar epochs (about 800 epochs) to reach their best and values of D_x and D_z throughout training are similar as well. Therefore, different ranges of θ^{d0} do not significantly influence the process of optimization of the discriminator that we investigated.

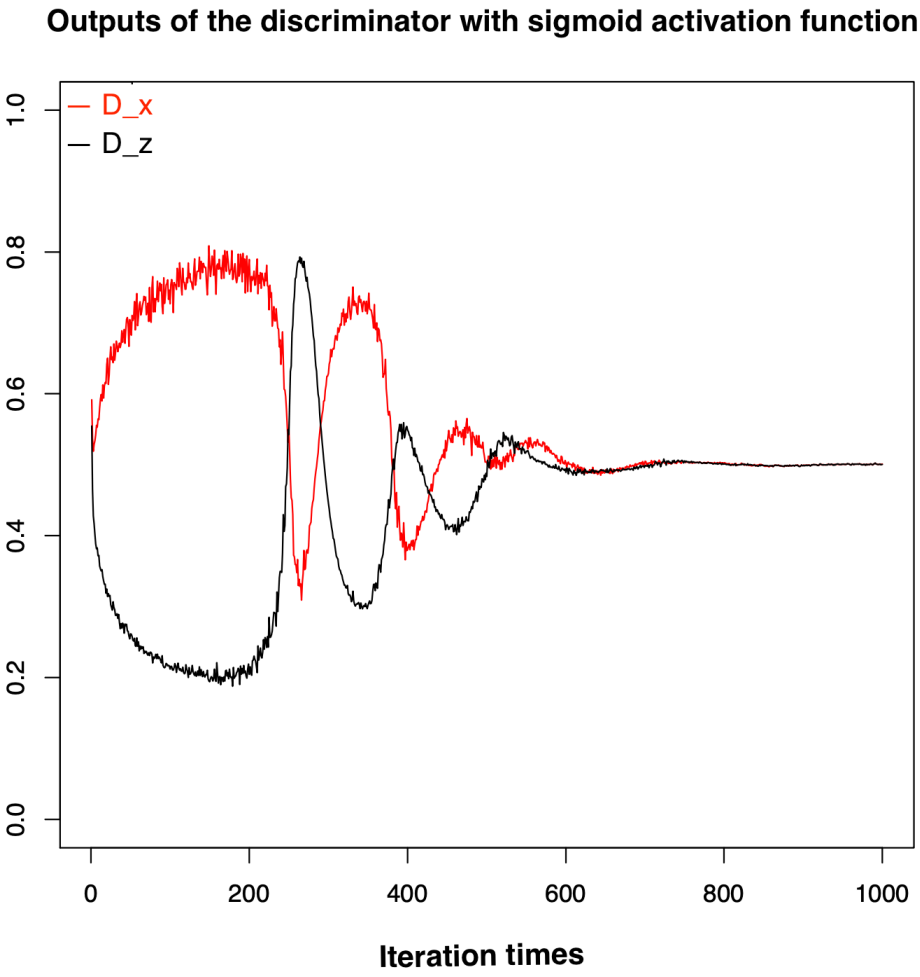

Outputs of the discriminator with sigmoid activation function
D_x
D_z
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000
Iteration times

Figure 28

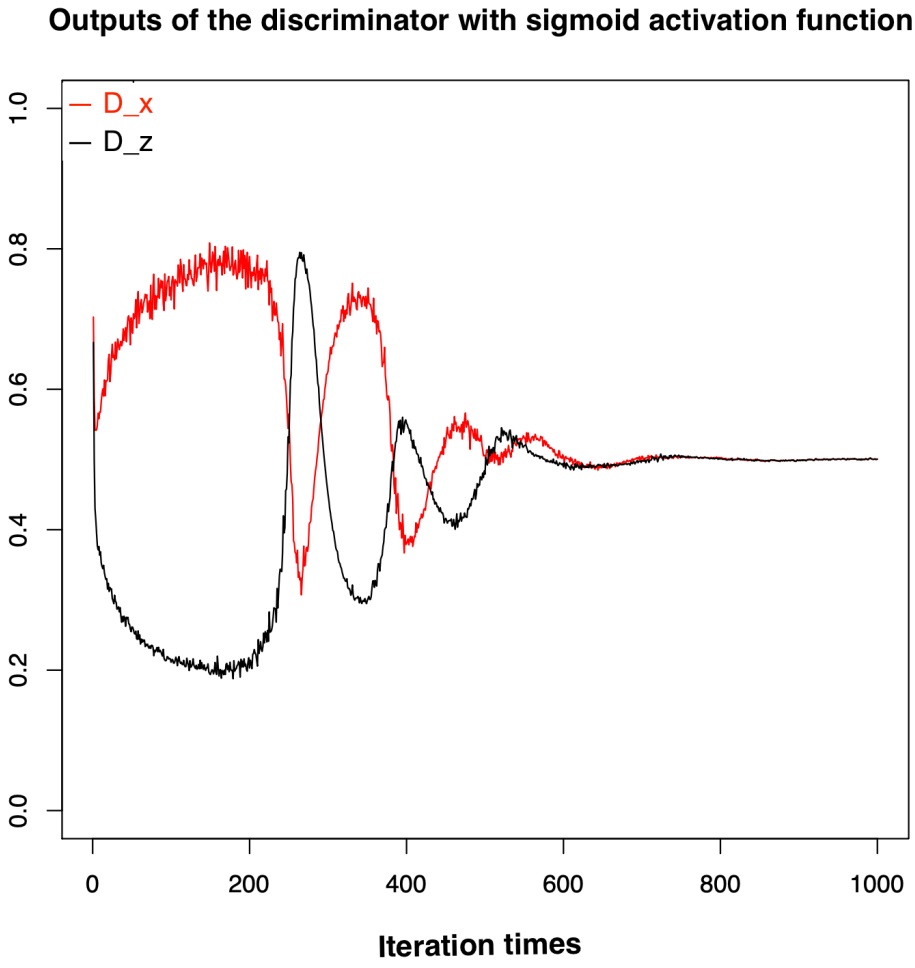

Outputs of the discriminator with sigmoid activation function
D_x
D_z
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000
Iteration times

Figure 29

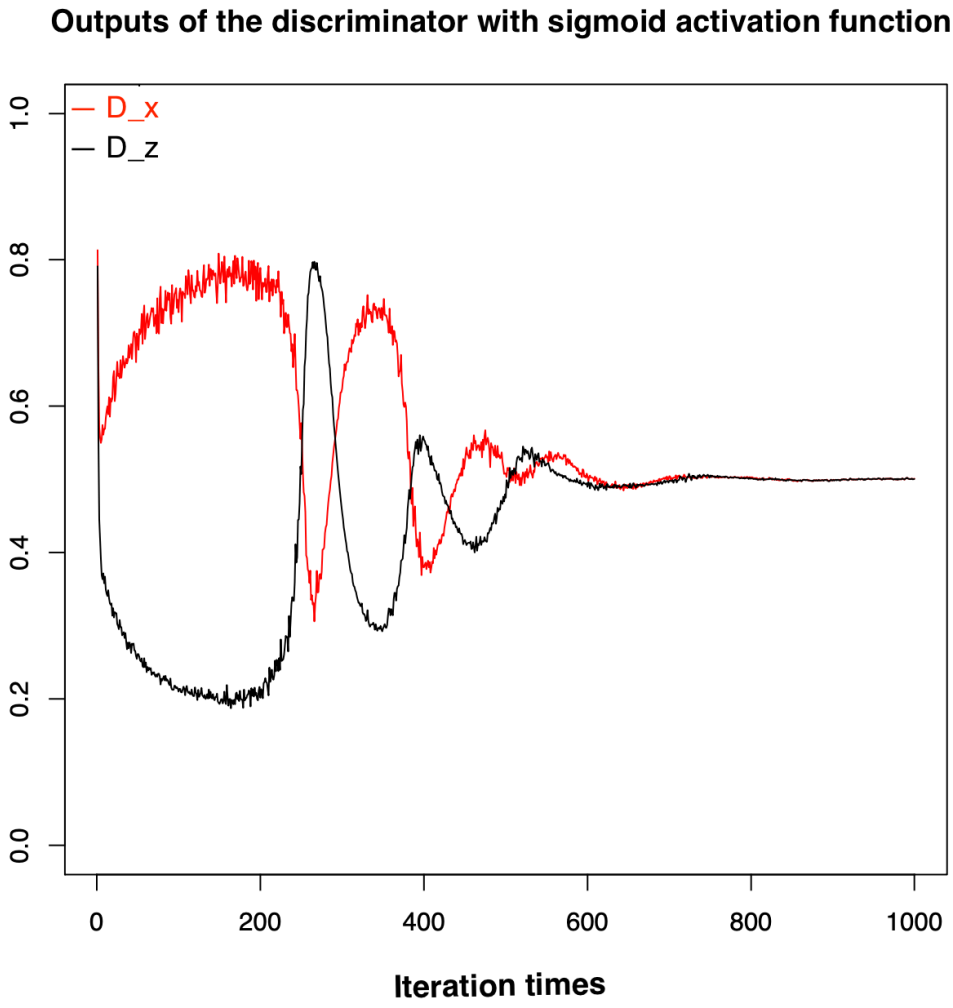


Figure 30

a.3 Ranges of the parameters

Ranges of θ^g and θ^d fluctuate similarly throughout training as well. The upper right corners of Figure 24 to Figure 26 provide ranges of the parameters in the last epoch. Ranges of three variables of θ^g in the last epoch are $(-1.2340, 0.9398)$, $(-1.2344, 0.9407)$ and $(-1.2348, 0.9416)$ respectively; ranges of three variables of θ^d in the last epoch are $(-0.0313, 0.0233)$, $(-0.0328, 0.0226)$ and $(-0.0344, 0.0219)$ respectively. There is only a slight difference among ranges of the parameters in the last epoch.

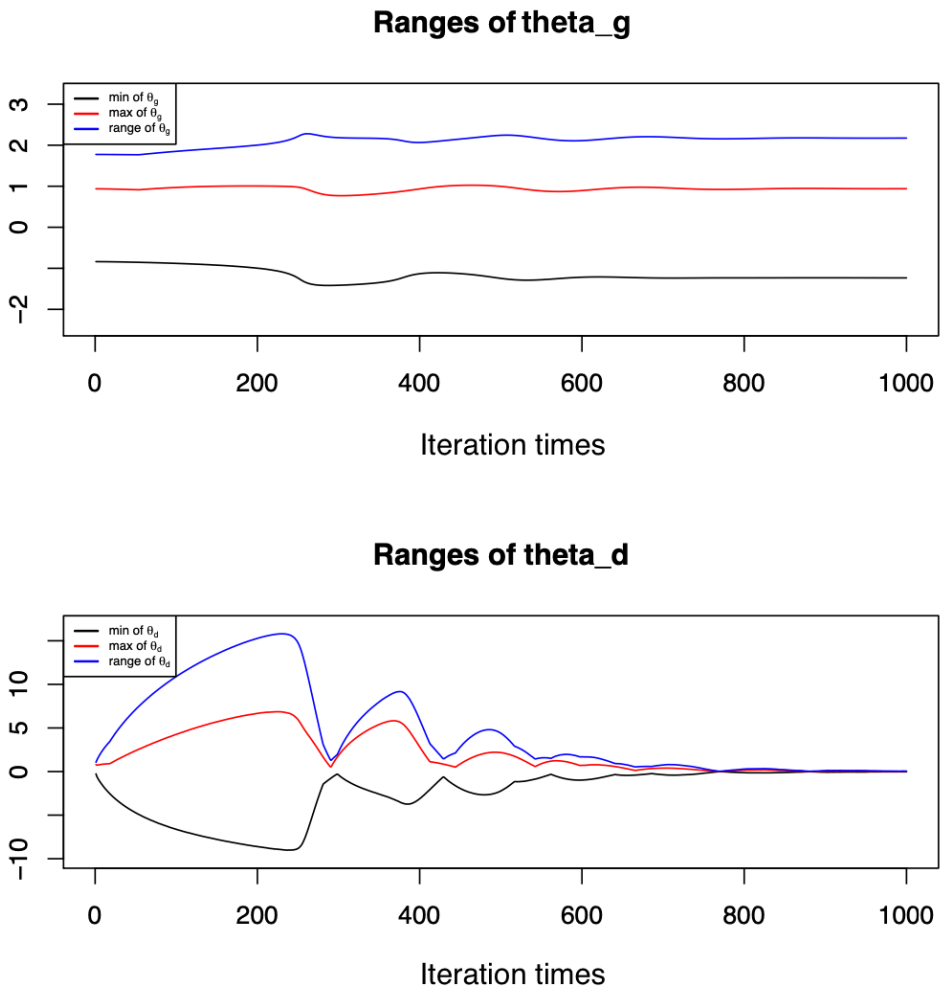


Figure 31: Ranges of the Parameters 1

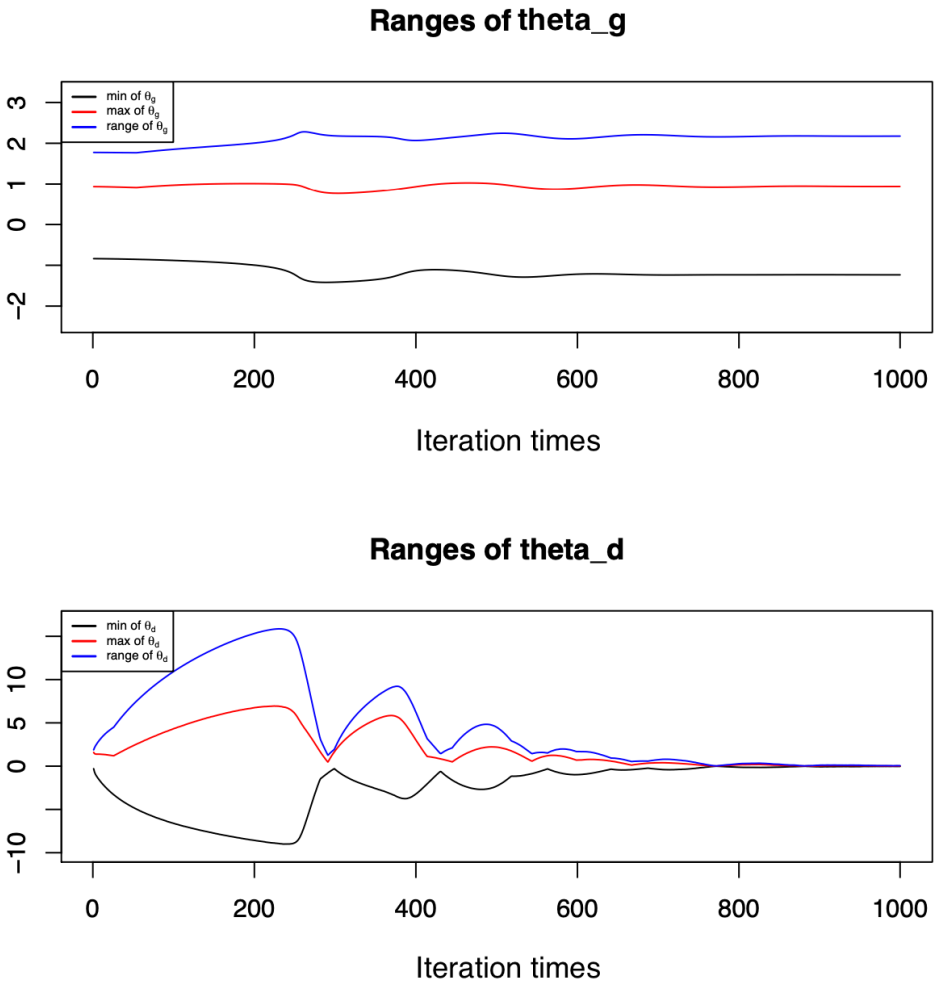


Figure 32: Ranges of the Parameters 2

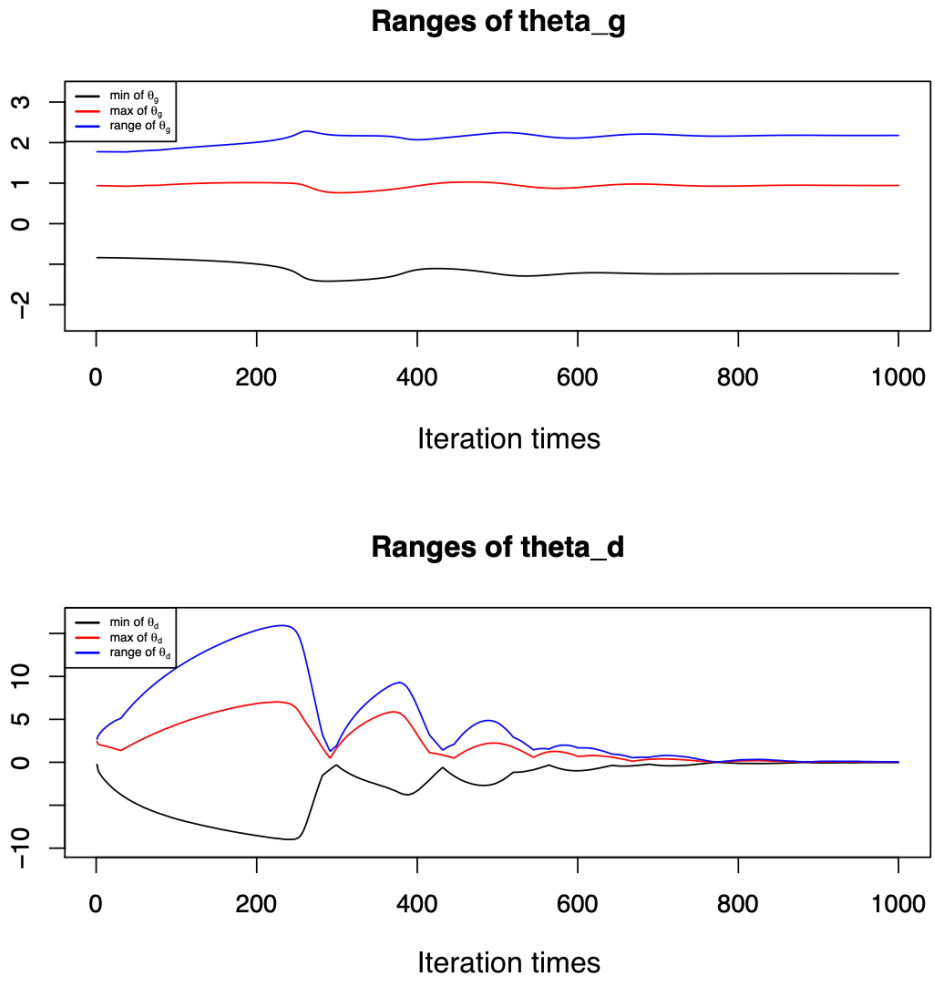


Figure 33: Ranges of the Parameters 3

Therefore, ranges of the initial θ^d do not have a significant effect on the performance of the algorithm in our cases. It partially confirms our conjecture at the beginning of Section 4.4.4.

b. Range of θ^{g0}

We will discuss two main scenarios below. In one scenario, density plots of the generated data z_g gain bimodality when wider ranges of θ^{g0} are used. In the other scenario, density plots are distributed as unimdal shapes all the time. In both scenarios, in each group of three figures below, θ^{g0} are sampled from $[-1, 1]$, $[-2, 2]$ and $[-3, 3]$ respectively.

Below list figures of the first scenario (from the case: target1, prior2, run1, $\theta^{d0} \in [-1, 1]$

denoted as t1p2r1D1). In this case, target distribution is sharper and located on the left side and noise prior distribution is flatter and located on the right side.

b.1 Scenario 1

Density plots of z_g in the last epoch change from unimodal shapes to bimodal shapes gradually when the range of θ^{g0} goes wider. Moreover, when density plots distribute as unimodal shapes, it is very likely that they are sharper than targets; however, when bimodality is gained, they distribute more flat and both modes are lower than targets.

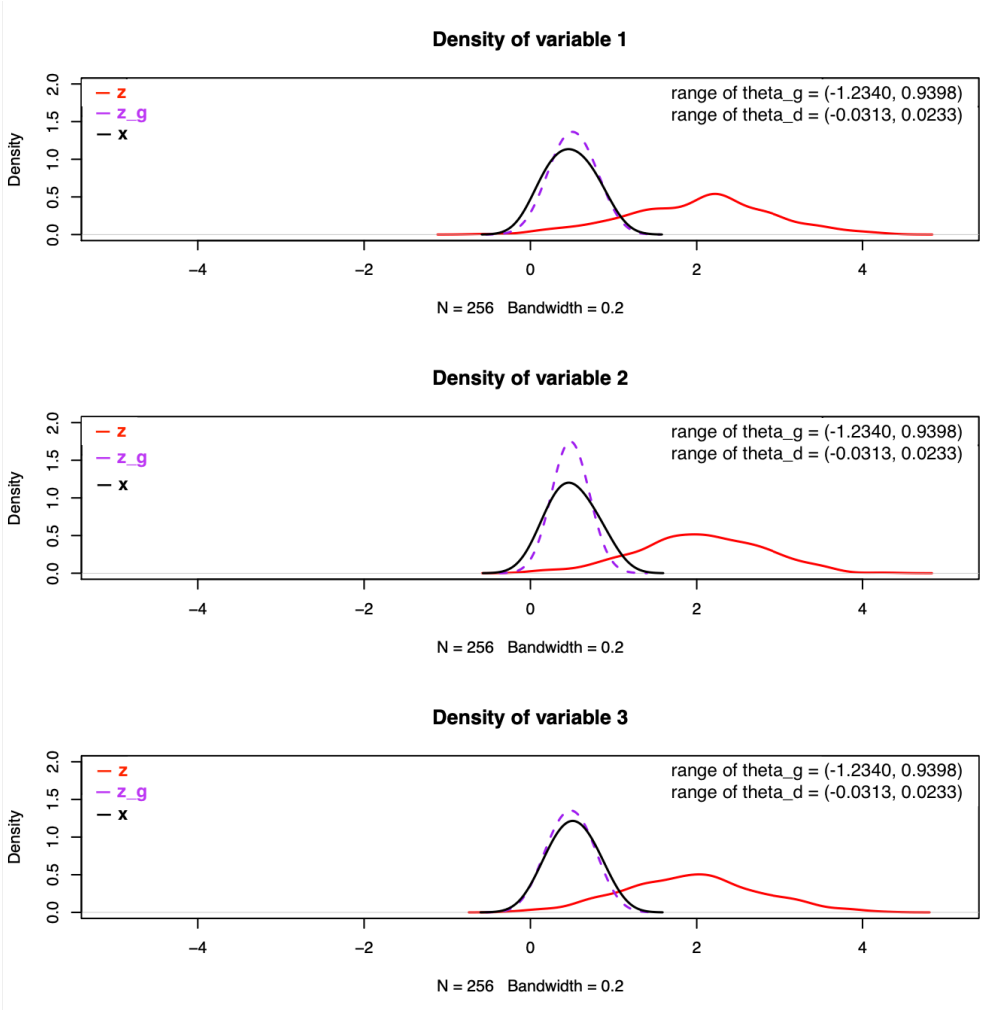


Figure 34: Density Plots 1

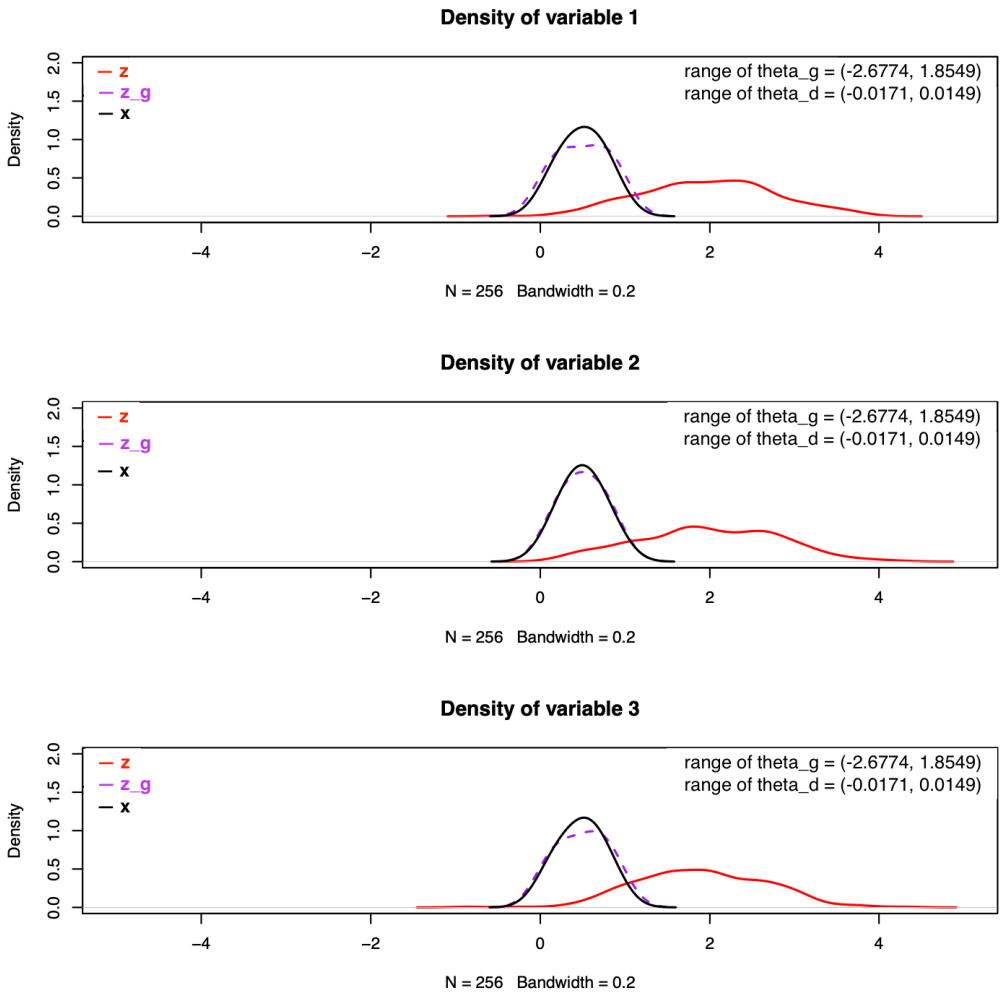


Figure 35: Density Plots 2

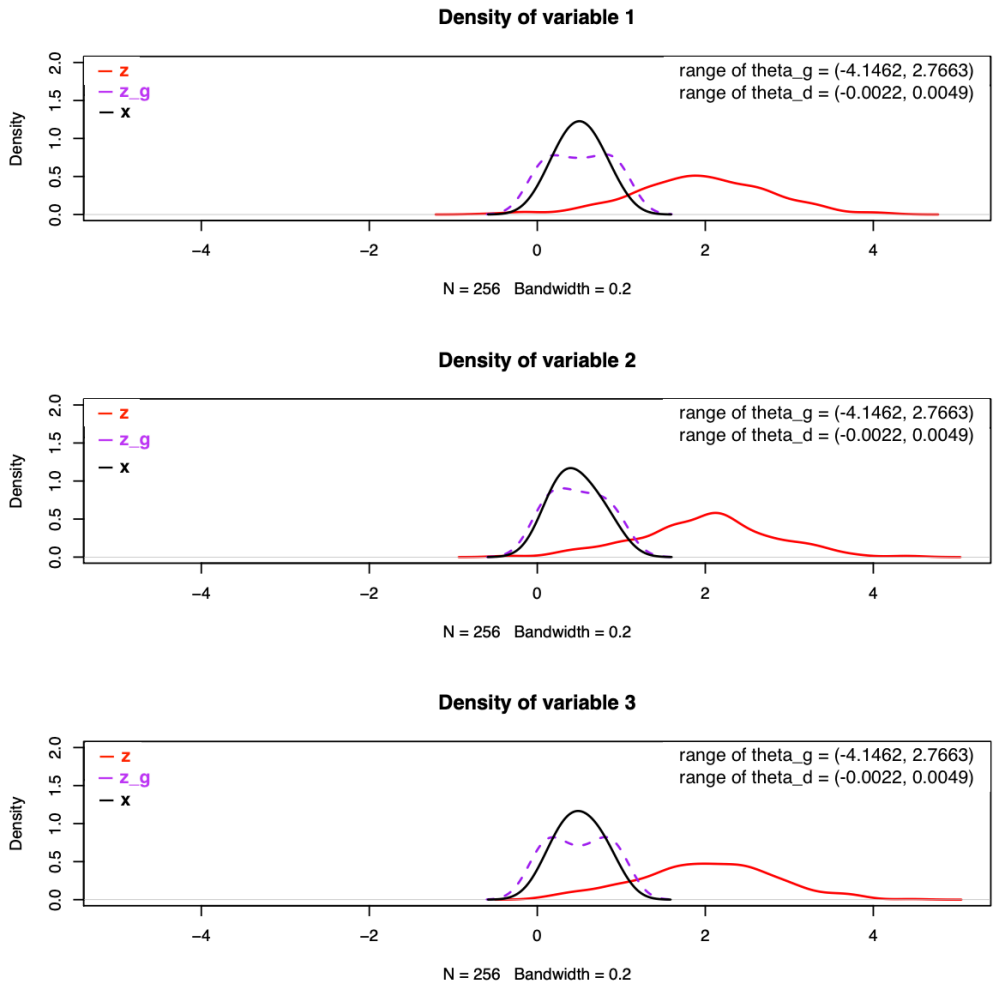


Figure 36: Density Plots 3

Apparently, the algorithm requires much more epochs to train before reaching its best (from about 800 epochs to 2500 epochs, then to 6000 epochs) when the range of θ^{g0} becomes wider. Meanwhile, D_x and D_z flucuate more extremely through training (from about $[0.2, 0.8]$ to $[0, 1]$).

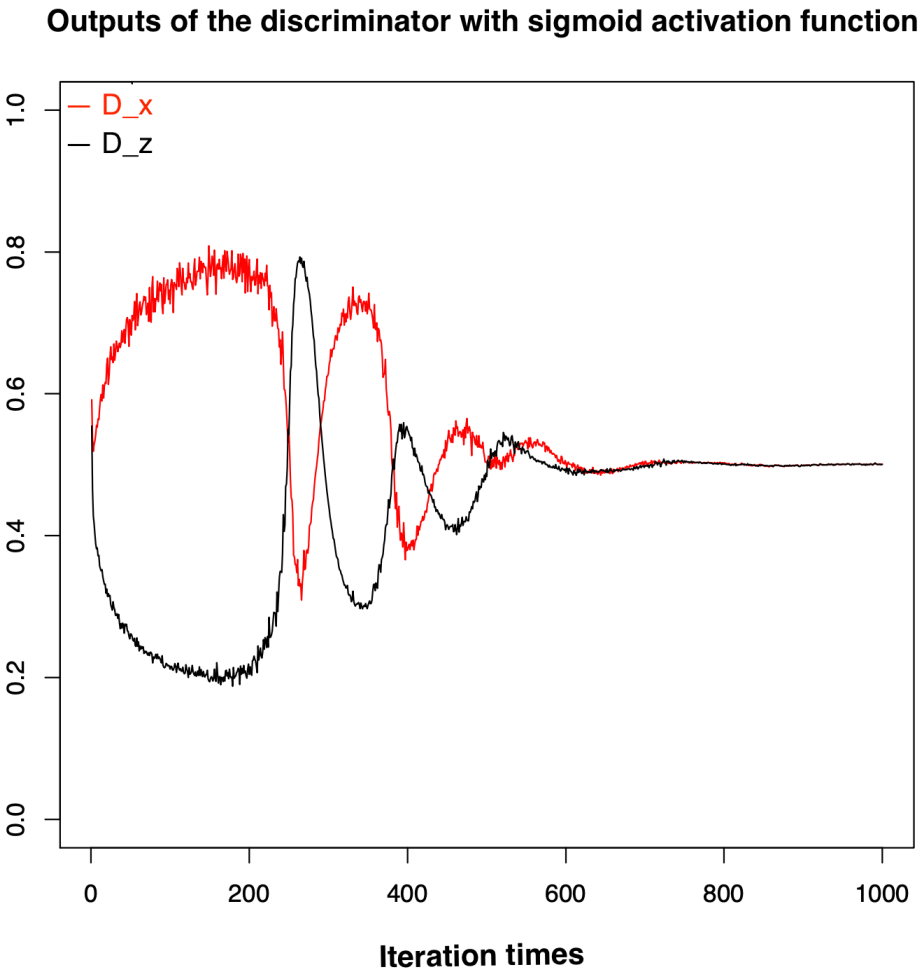


Figure 37: Outputs of the Discriminator 1

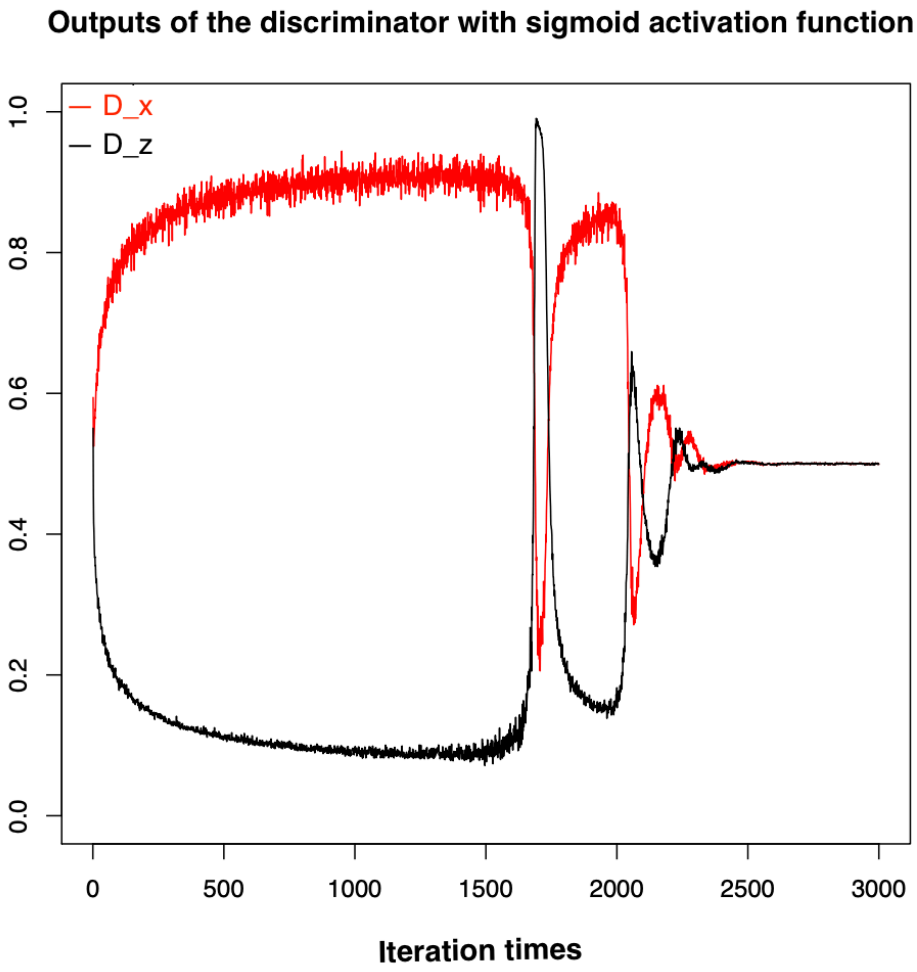


Figure 38: Outputs of the Discriminator 2

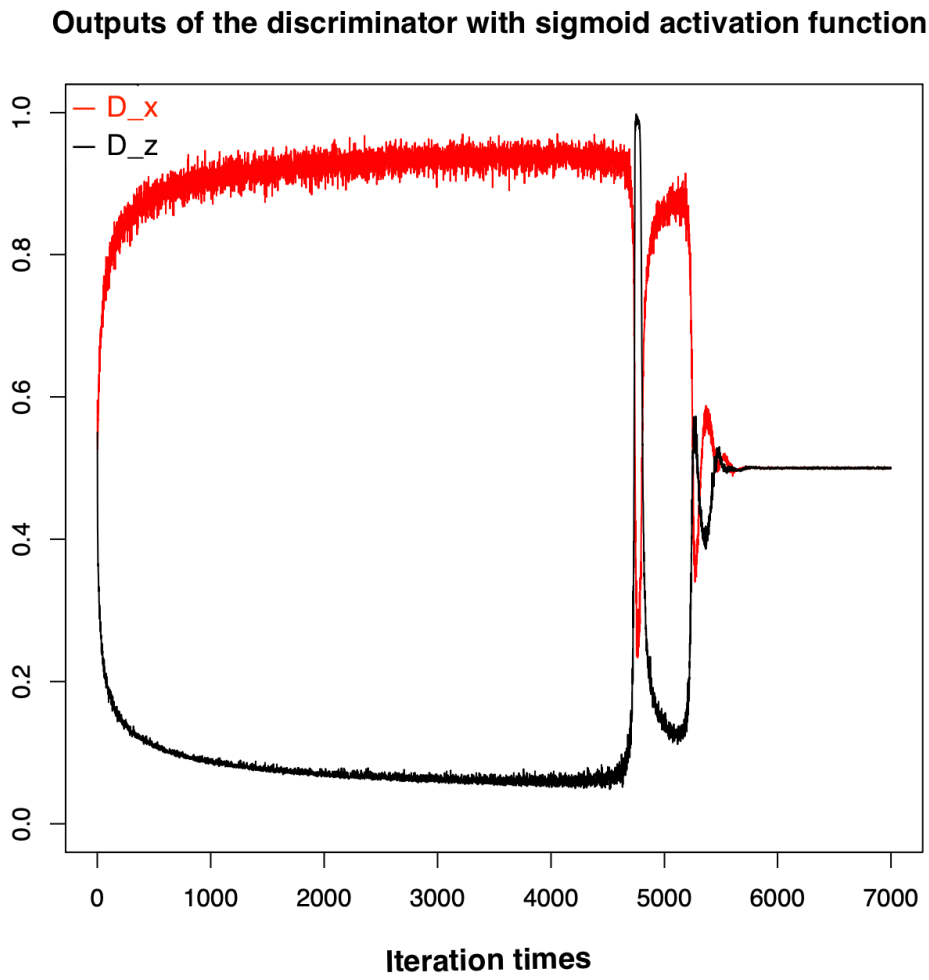


Figure 39: Outputs of the Discriminator 3

We still have θ^d converging to zero in all figures. However, the range of θ^d waves more broadly when the range of θ^{g0} becomes wider (from around $[-10, 15]$ to $[-20, 40]$, then to $[-30, 60]$). The range of θ^g becomes slightly larger (blue lines) but remains flat most time. Maximum and minimum values of θ^g (red lines and black lines) decrease slightly and remain flat most time throughout training.

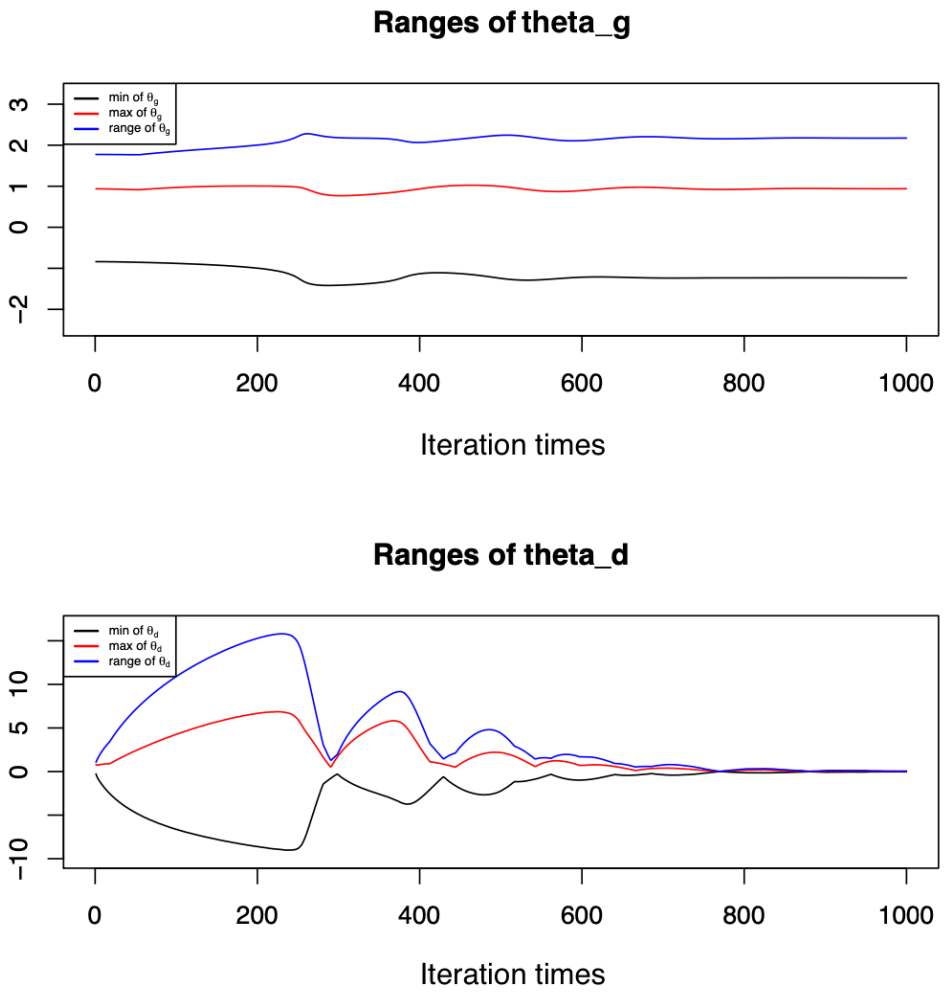


Figure 40: Ranges of the Parameters 1

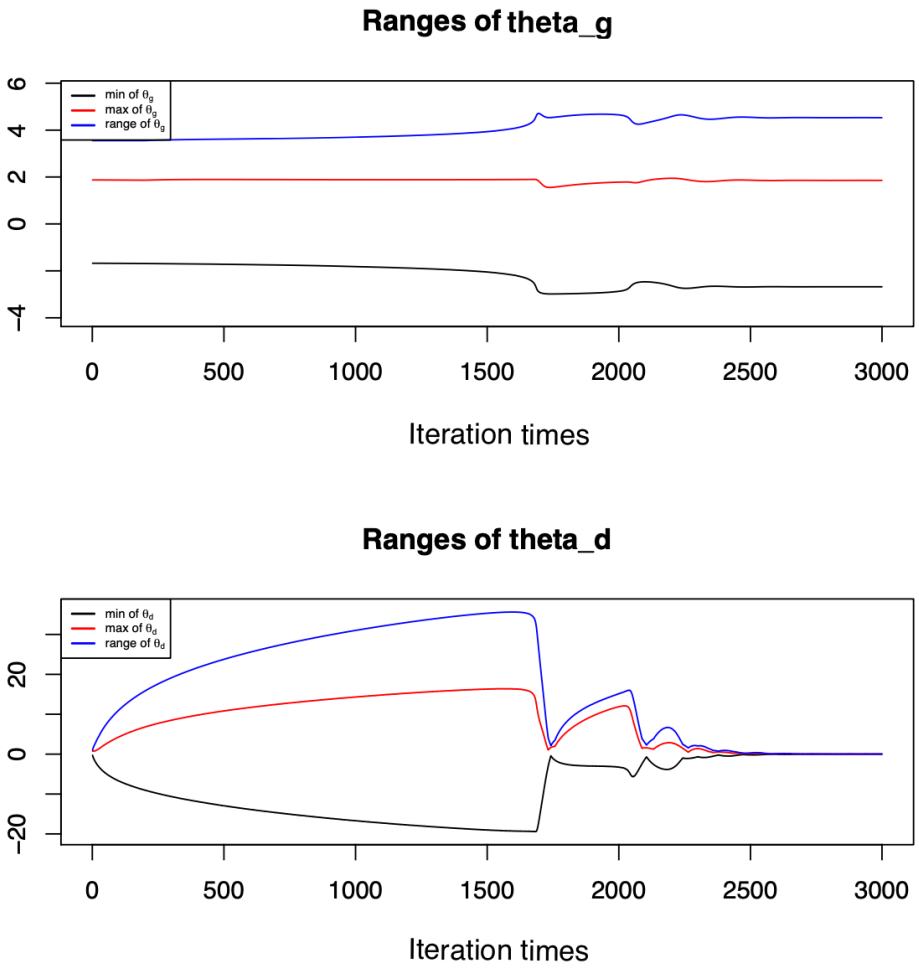


Figure 41: Ranges of the Parameters 2

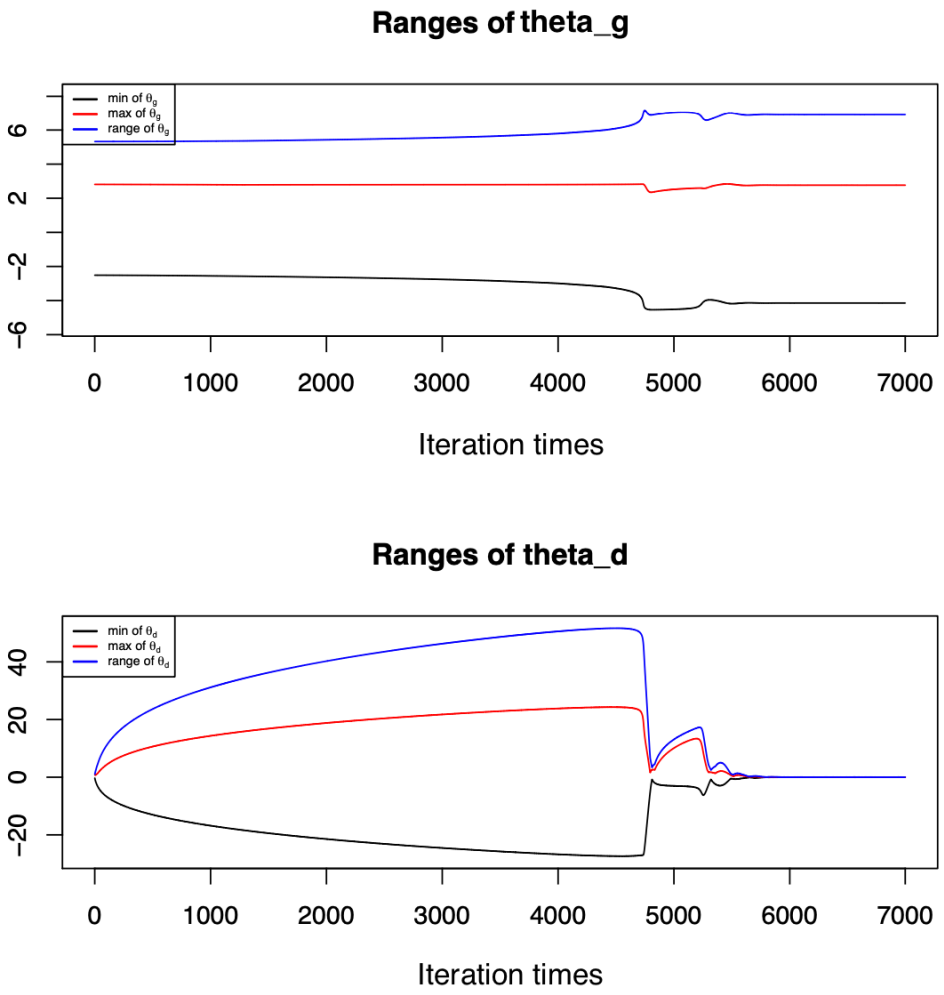


Figure 42: Ranges of the Parameters 3

b.2 Scenario 2

We will discuss the second scenario in the following (from the case: t1p4r3D1).

All density plots of z_g are unimodal and sharper than targets.

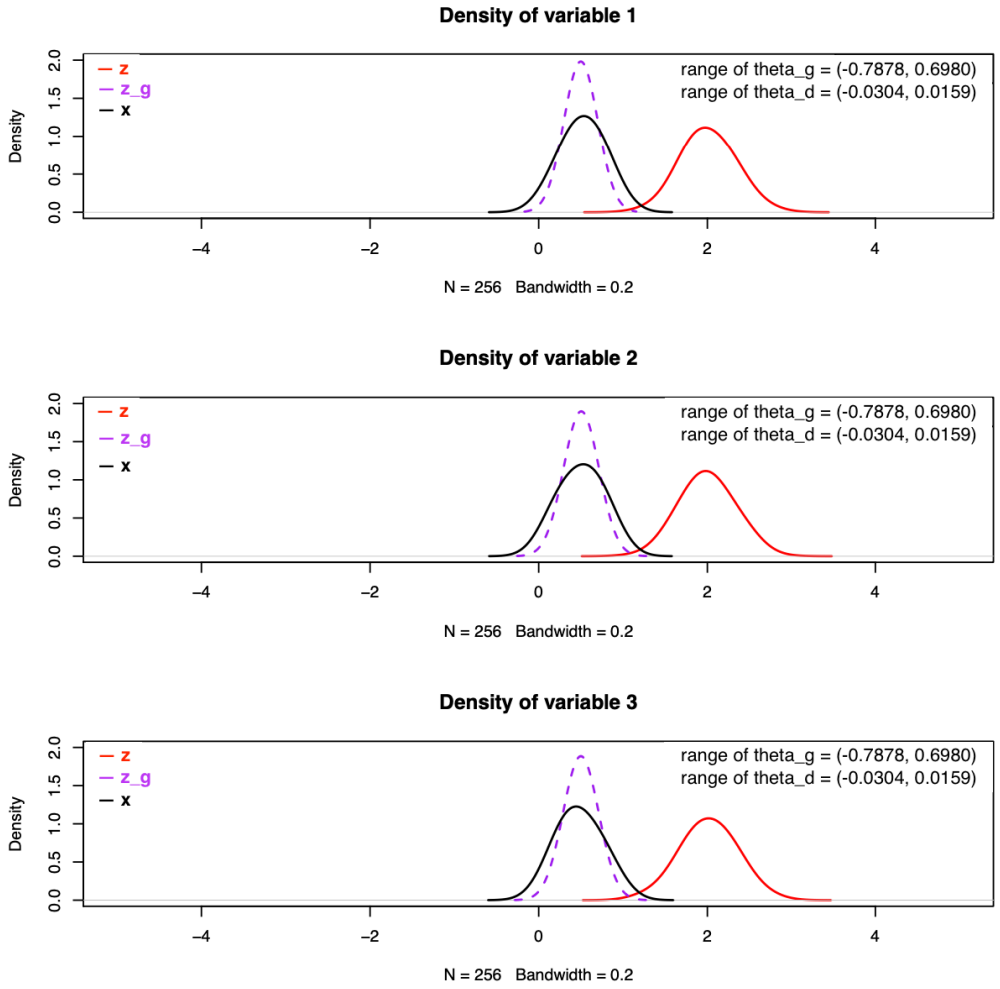


Figure 43: Density Plots 1

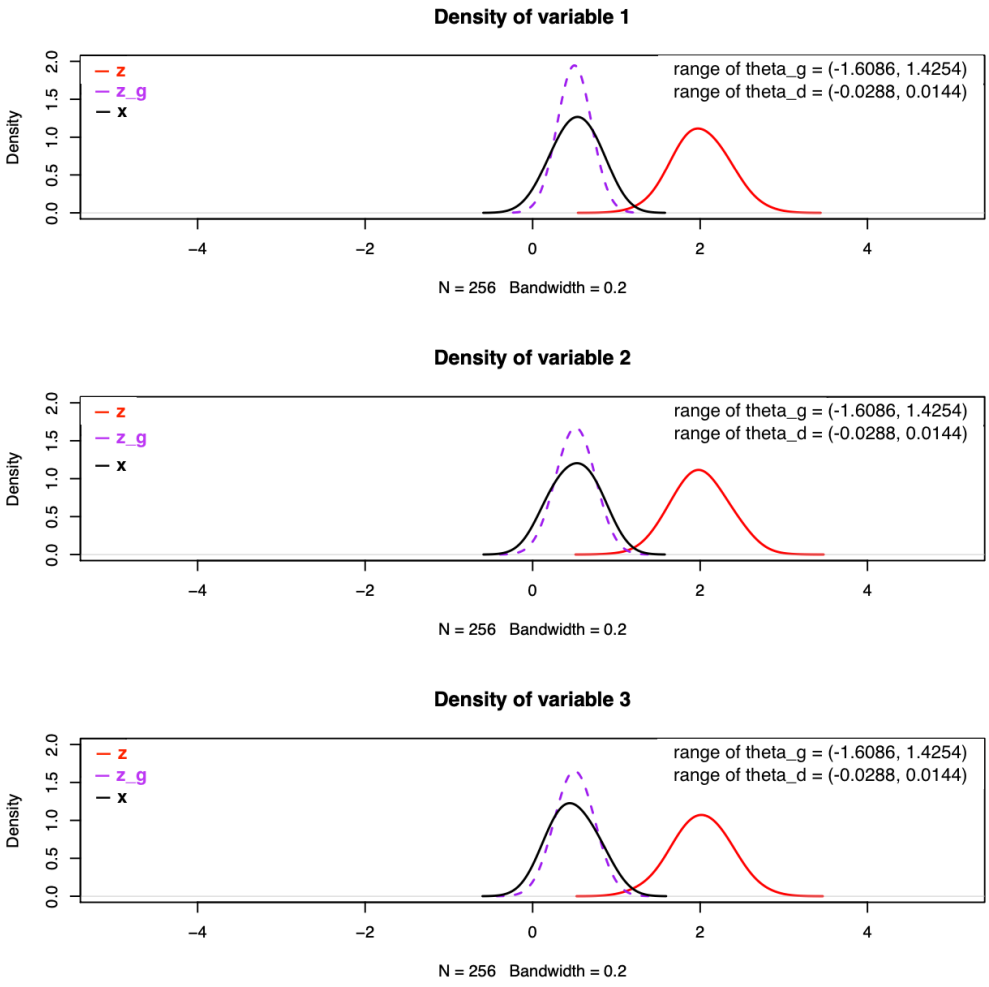


Figure 44: Density Plots 2

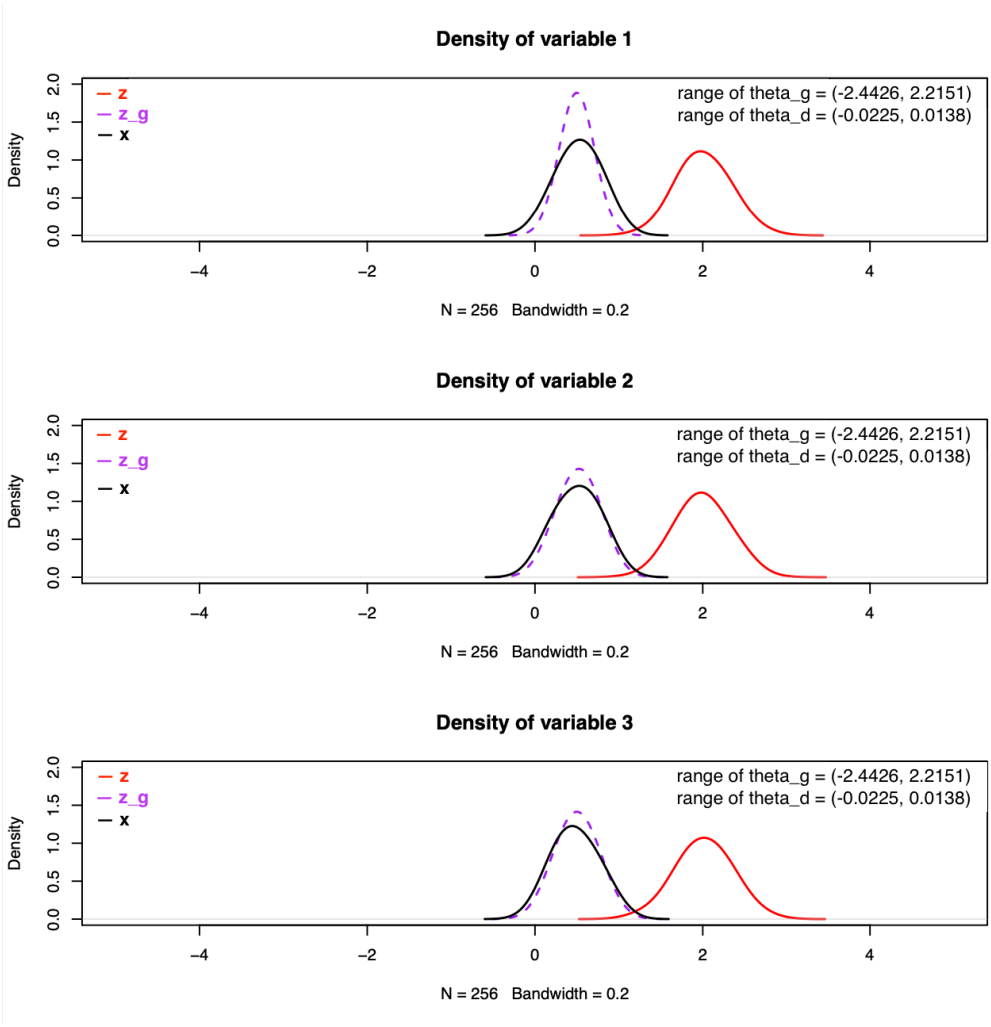


Figure 45: Density Plots 3

Performance of the discriminator's output is similar to that in the last scenario.

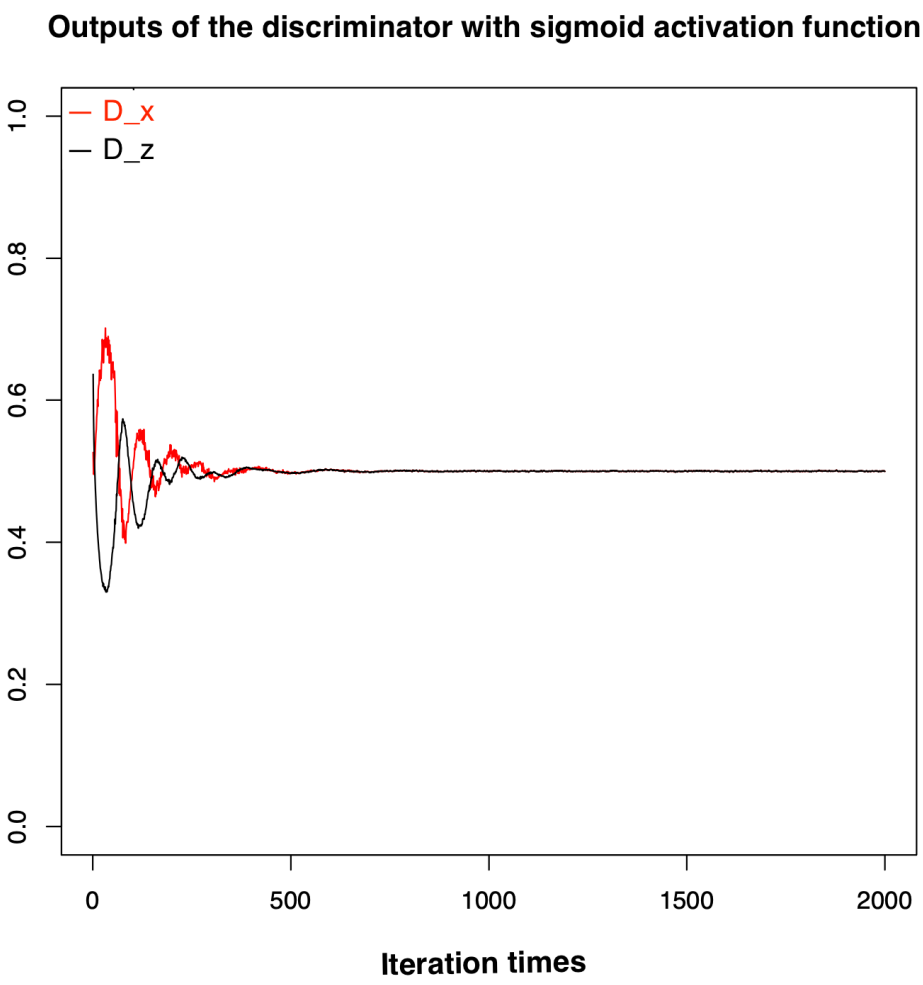


Figure 46: Outputs of the Discriminator 1

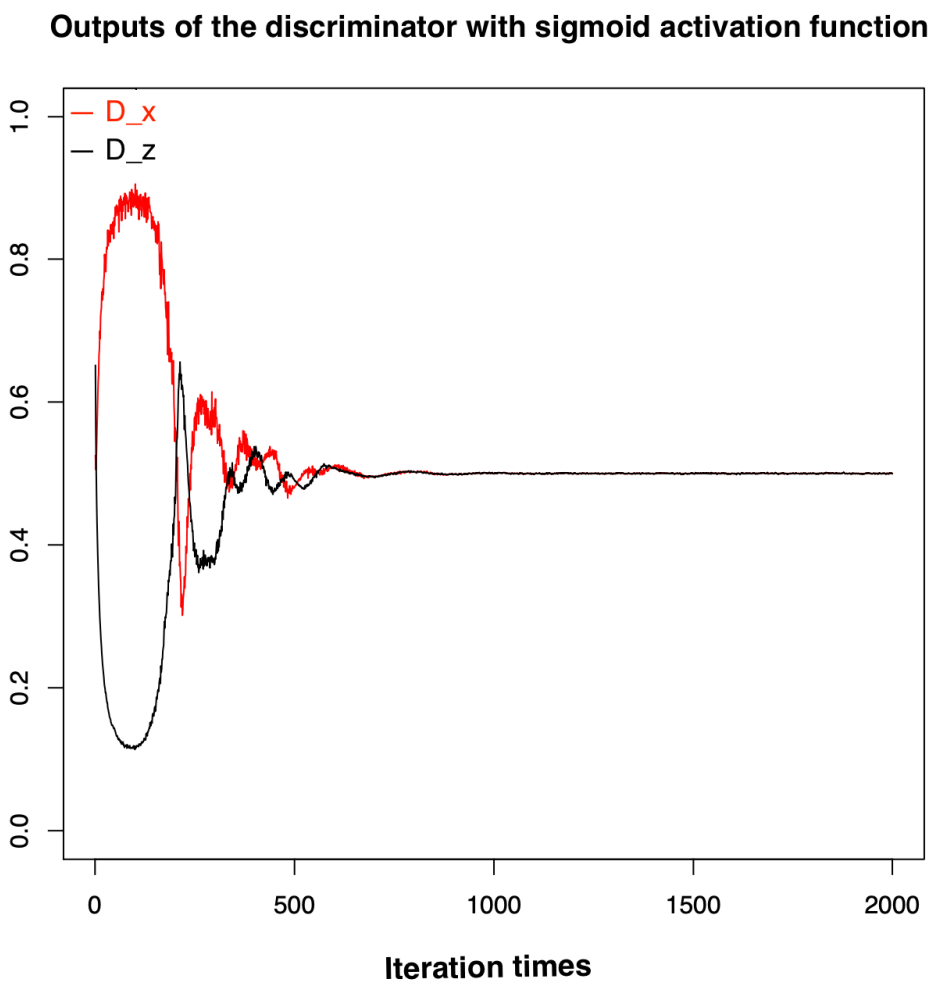


Figure 47: Outputs of the Discriminator 2

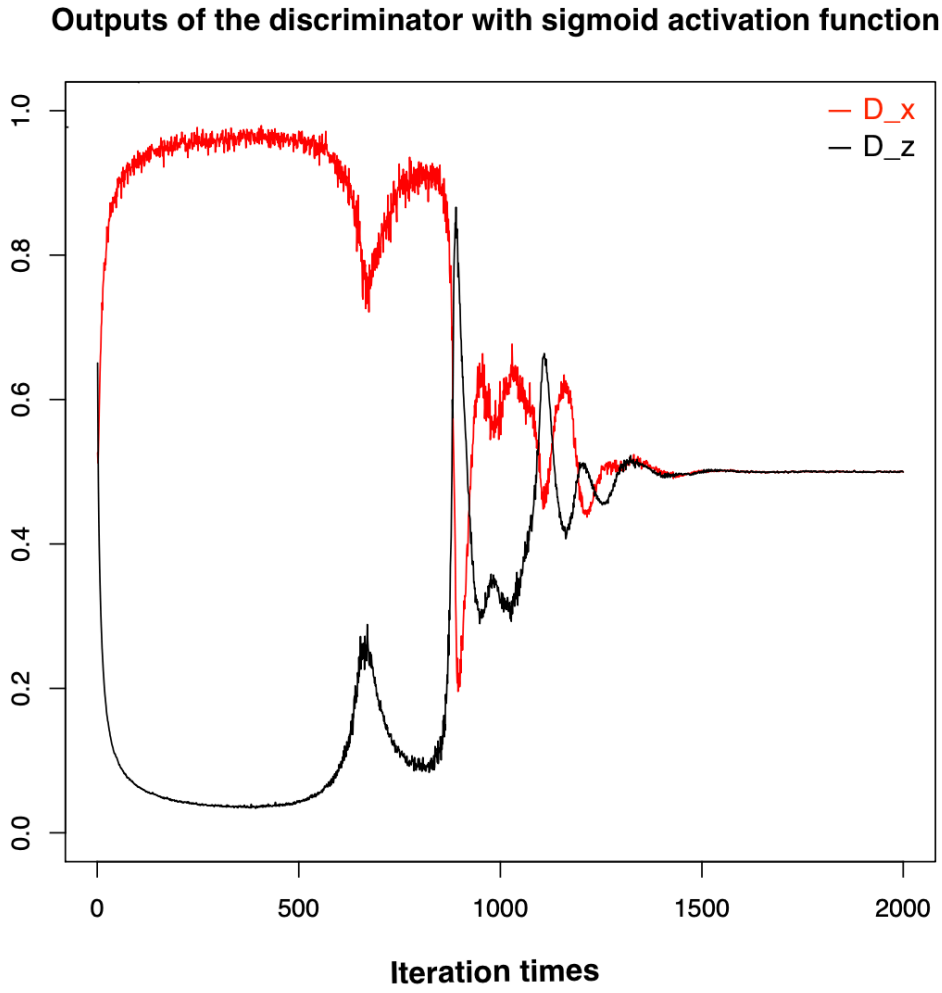


Figure 48: Outputs of the Discriminator 3

θ^d converges to zero and its range fluctuates more when the range of θ^{g0} becomes wider. Ranges (maximum, minimum and the difference) of θ^g remain flat most time during training.

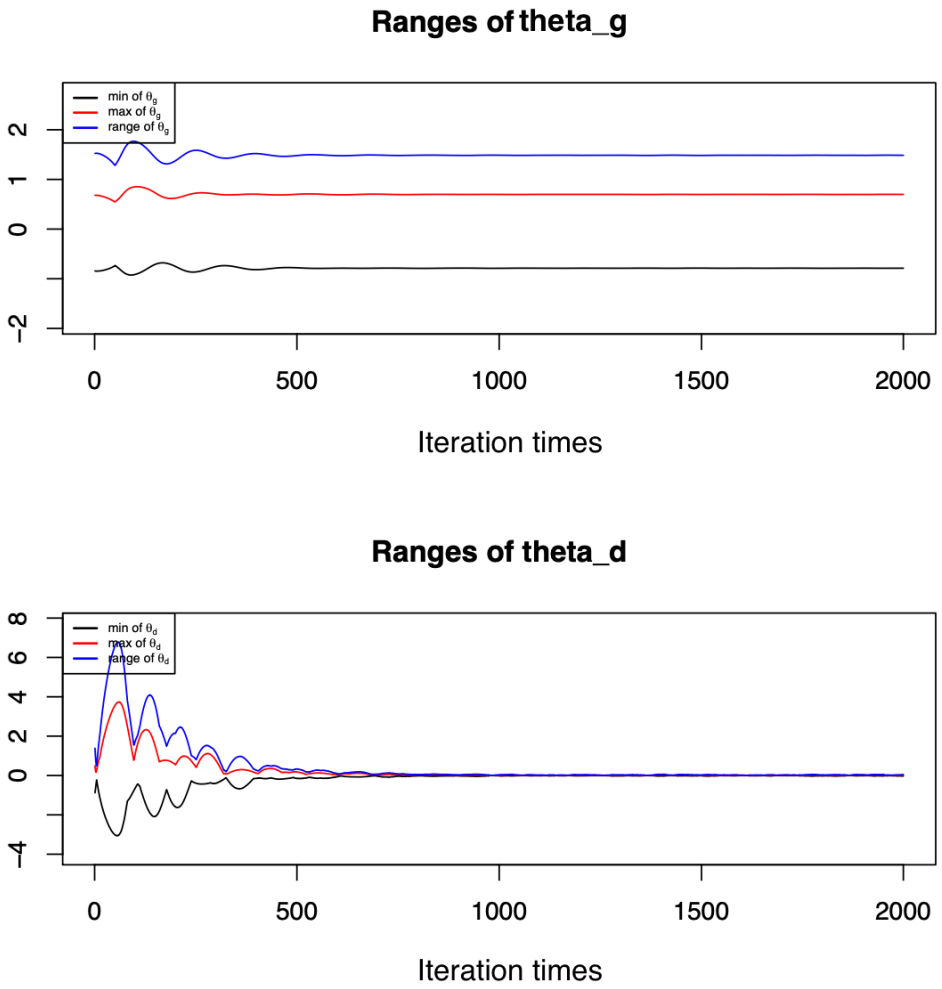


Figure 49: Ranges of the Parameters 1

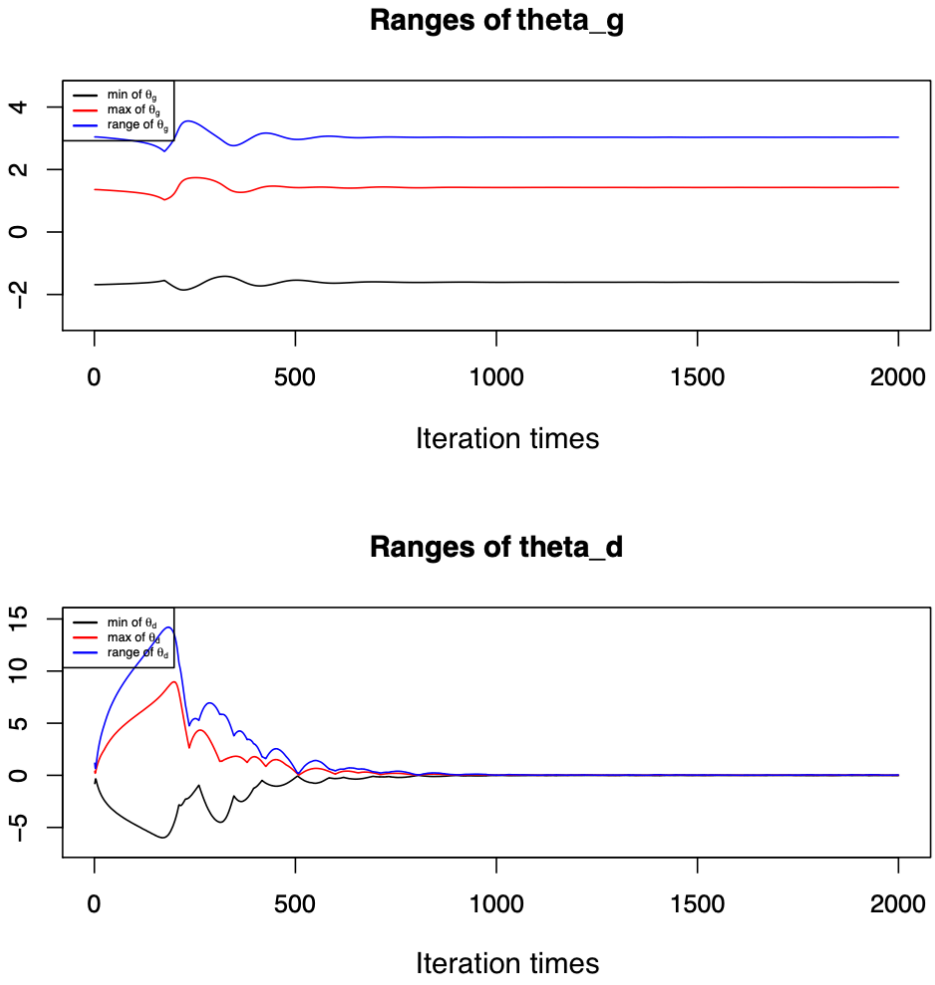


Figure 50: Ranges of the Parameters 2

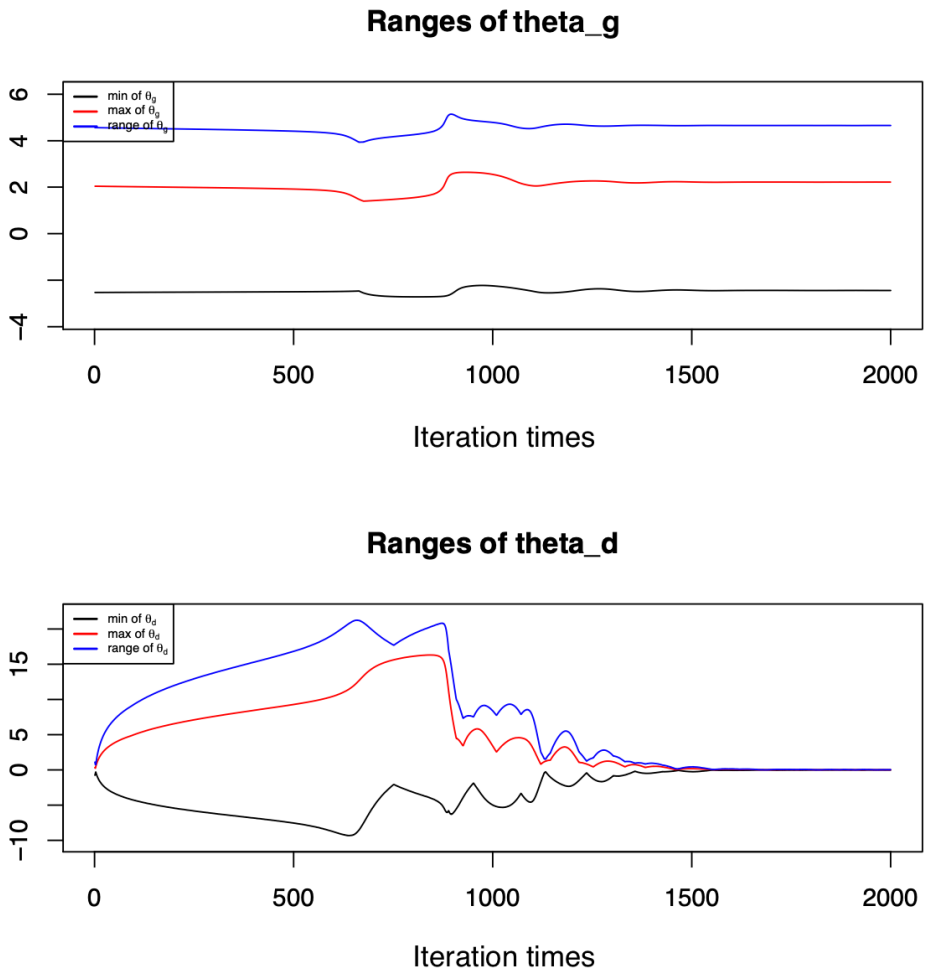


Figure 51: Ranges of the Parameters 3

c. Analysis of both ranges of θ^{d0} and θ^{g0}

In Section 4.4.4, we intend to analyze the influence of the ranges of θ^{g0} and θ^{d0} on the performance of Algorithm 2. We observe some familiar performance discussed in Section 4.4.3. In the following, we will analyze the performance exclusively appearing in Section 4.4.4.

Performance of the algorithm is not significantly influenced by the range of θ^{d0}. Similarity of the density plots of z_g reflects that different ranges of θ^{d0} do not have a significant impact on the generated distribution. Similarity of the changes of D_x and D_z shows that the discriminator is not significantly affected throughout training. Similarity of the fluctuations of the ranges of θ^d and θ^g and ranges of parameters in the last epoch

illustrates that the iterative update process of the parameters is not significantly affected. Similarity of the training epochs before reaching the best performance indicates that the convergence rates of the algorithm remain stable. Therefore, we believe that different ranges of θ^{d0} will not significantly affect the performance of Algorithm 2. Although ranges of θ^{d0} are less important in this algorithm, it requires further exploration as to whether it will have a significant impact on the performance of GANs under other settings.

Density plots of the generated data gain bimodality when the range of θ^{g0} goes wider. Ranges of θ^g remain almost flat during training. θ^g in the last epoch shares a similar range with θ^{g0}. Since ranges of θ^{g0} are centered at 0, the range of θ^g will be larger if we use a wider range of θ^{g0}. As shown in Section 3.1.2, z_g is distributed as a logit-normal distribution. Greater values of $|\theta^g|$ lead the generated distribution to be bimodal as shown in the figures above, and lower values of $|\theta^g|$ lead it to a unimodal shape. Therefore, the experiment results are consistent with previous proof in Section 3.1.3.

It takes longer time to train the algorithm before achieving its best when the range of θ^{g0} is wider. Since when the range of θ^{g0} is wider, it is likely that the range of θ^g throughout training is wider. Therefore, there are more values of θ^g that can be covered throughout training. Furthermore, if the range of θ^{g0} is wider, the absolute values of parameters of the generator will be larger. Then with fixed gradients of the parameters, it takes longer time to train if the values of parameters are larger.

Outputs and parameters of the discriminator fluctuate more exaggeratedly throughout training when the range of θ^{g0} is wider. With fixed mini-batches of noise samples, larger θ^g gives us larger values of generated samples. Then during the update of the discriminator, the gradients are larger, so that the values of θ^d change more significantly.

4.4.5 A hidden layer in the discriminator

We introduce a hidden layer in the discriminator to solve previous problems and name it Algorithm 3. Sigmoid activation function is used in both hidden layer and output layer. We use four prior distributions and choose three ranges of θ^{g0} to learn two targets. Run each

case three times with different seeds.

Hyper-parameter	Option or Value
target distribution p_{data}	two targets
noise prior distribution p_Z	four priors
size of mini-batch	$m = 256$
steps of optimizing discriminator	$L = 10$
learning rate of θ^d	$\alpha = 0.1$
learning rate of θ^g	$\beta = 0.01$
range of initial θ^g	$[a, b] = \{[-1, 1], [-2, 2], [-3, 3]\}$
range of initial θ^d	$[c, d] = [-1, 1]$
seeds	$96, 54, 37$

In general, Algorithm 3 solves some of the problems occurred when we use Algorithm 2. It improves the discriminator with a stronger discriminating capacity and a more robust parameter update process. In the following, we will discuss the improvement of Algorithm 3 and identify the unsolved issues.

a. Improvement of Algorithm 3

We compare the results of alogrithm 3 to the ones of Algorithm 2. Figure 52, Figure 54 and Figure 56 show the results of Algorithm 2 in a case (t1p1r2G3). Figure 53, Figure 55 and Figure 57 display the results of Algorithm 3 in the same case.

Density plots of the generated distribution fit the targets better especially when the range of θ^{g0} is large. In Figure 52, density curves of z_g (purple dashed lines) fail to take on the unimodality of the target curves and show bimodality instead. This problem is eliminated in this case by using Algorithm 3, since density curves of z_g are unimodal and fit the targets better (see Figure 53). Therefore, given same noise samples, examples and initial parameters, Algorithm 3 has potential to learn the targets better than Algorithm 2. Moreover, both algorithms can learn the unimodality when ranges of θ^{g0} are small, i.e. $[-1, 1]$. In Algorithm 2, density plots are more likely to distribute as bimodal shapes when θ^{g0} are sampled from a large range, i.e. $[-3, 3]$. However, Algorithm 3 can learn the unimodality even ranges of θ^{g0} are large in this case. Therefore, range of the initial parameters of the generator has less effect on the results of Algorithm 3 than on the ones of Algorithm 2.

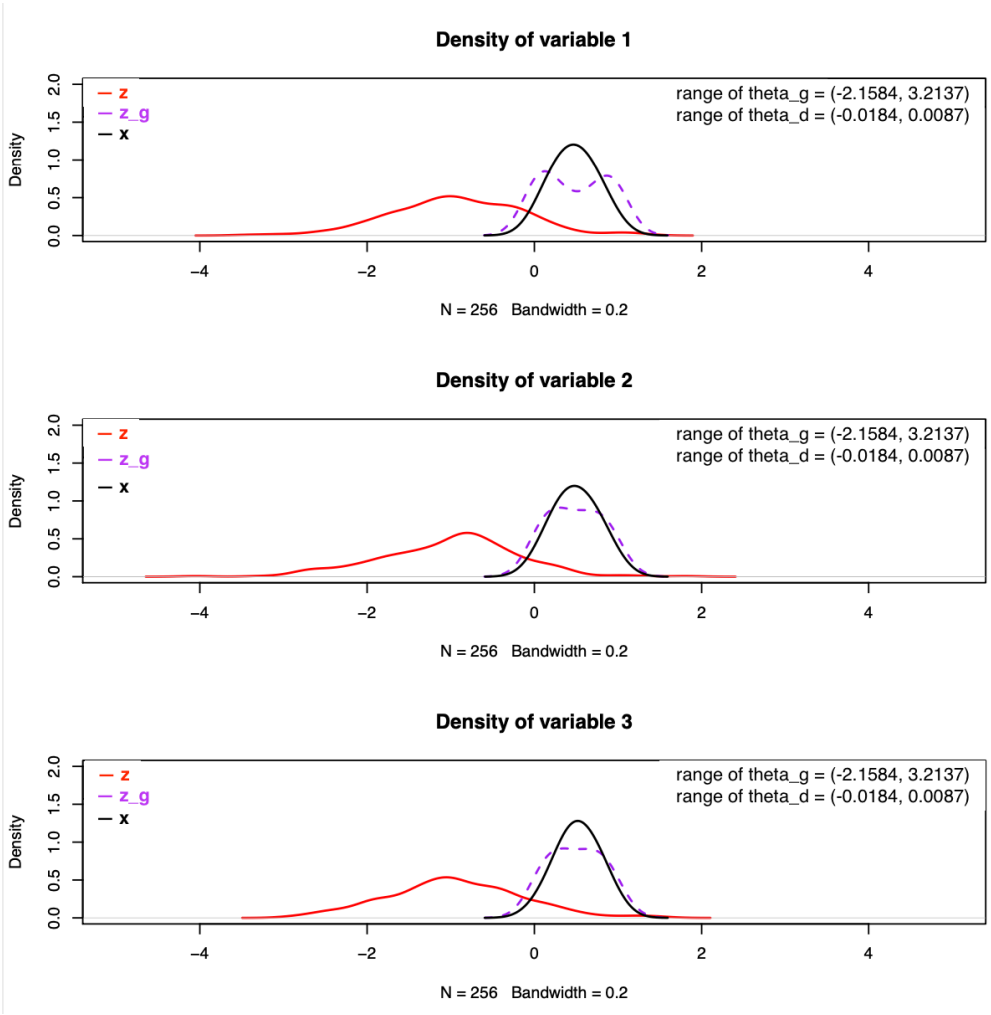


Figure 52: Density Plots 1

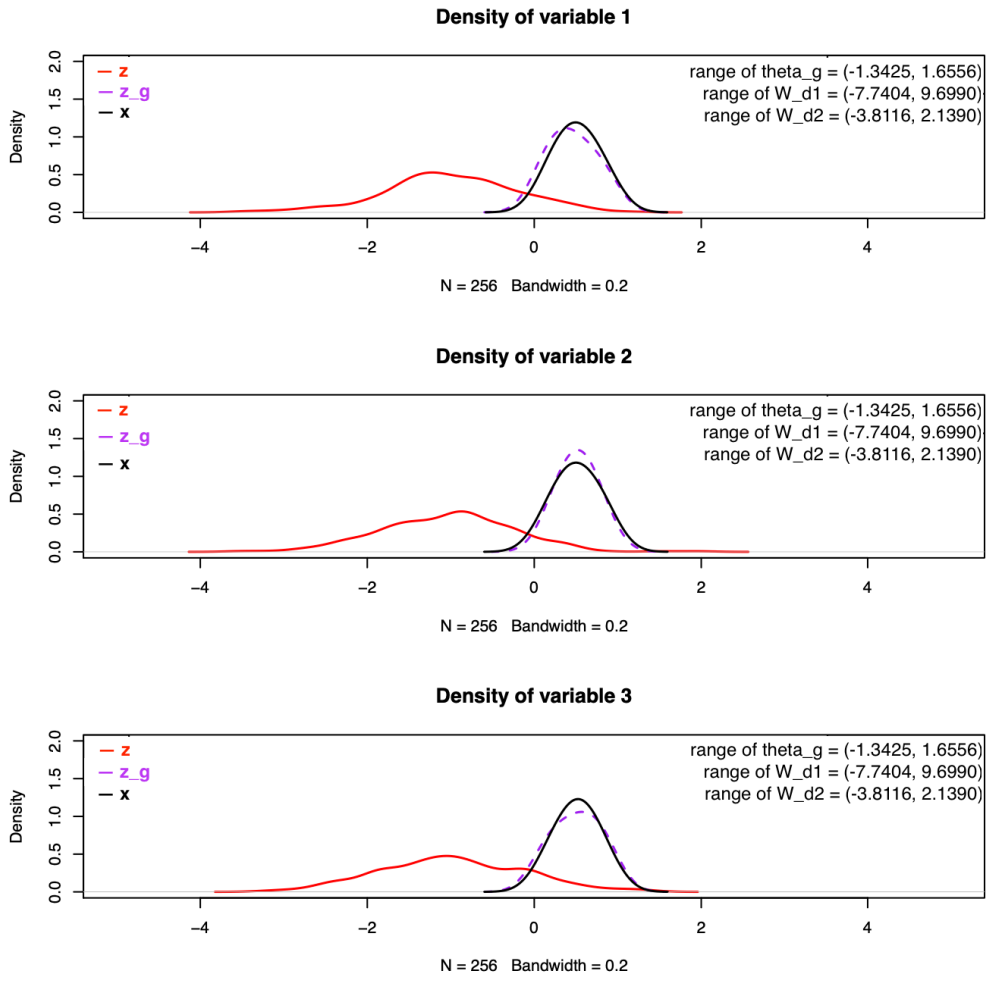


Figure 53: Density Plots 2

Discriminator gains improved discriminating capacity. θ^d are optimized more robustly and more values of θ^g can be reached throughout training. In Figure 54 and 56, D_x and D_z converge to one half and θ^d converges to 0 within 3000 epochs, which means that Algorithm 2 has been optimized within around 3000 iterations. However, in Figure 55, outputs of the discriminator do not converge to one half throughout training and D_x and D_z stay almost parallel to each other after 4000 epochs. In Figure 57, θ^d ($W^{d(1)}$ and $W^{d(2)}$) does not collapse to 0 any more, and ranges of θ^g fluctuate more widely than in Figure 56.

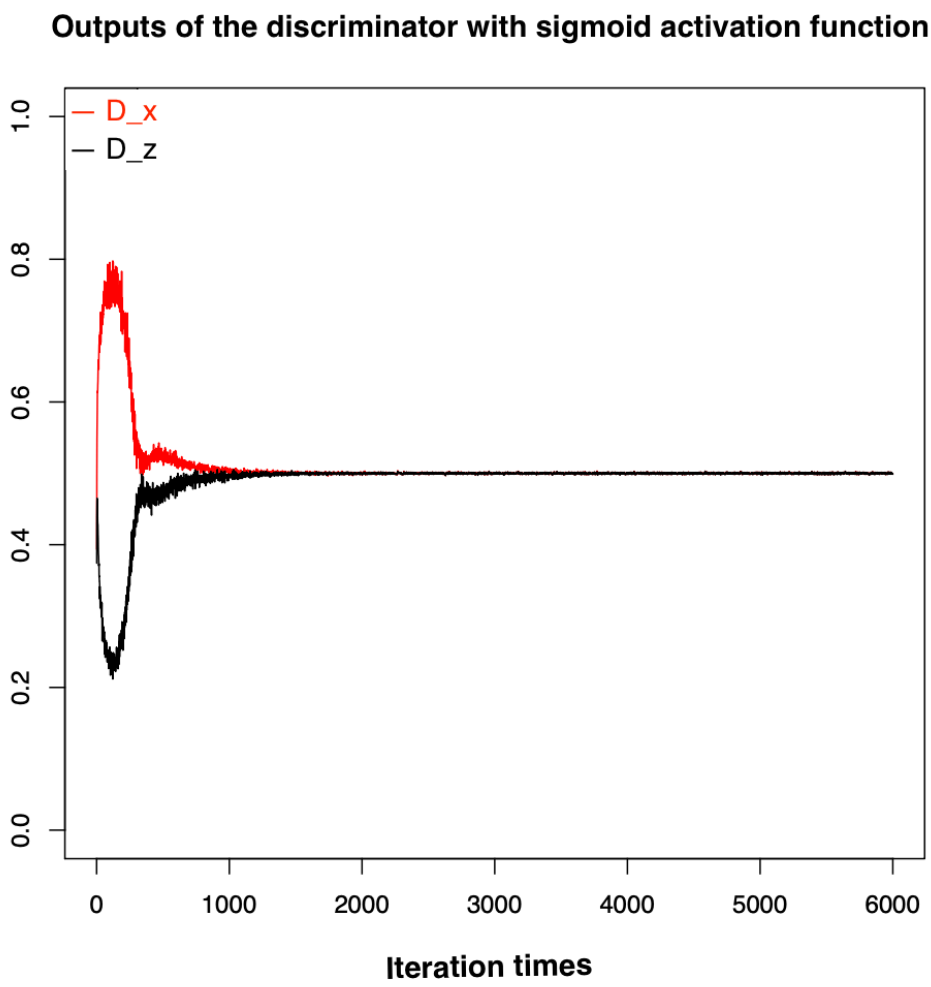


Figure 54: Outputs of the Discriminator 1

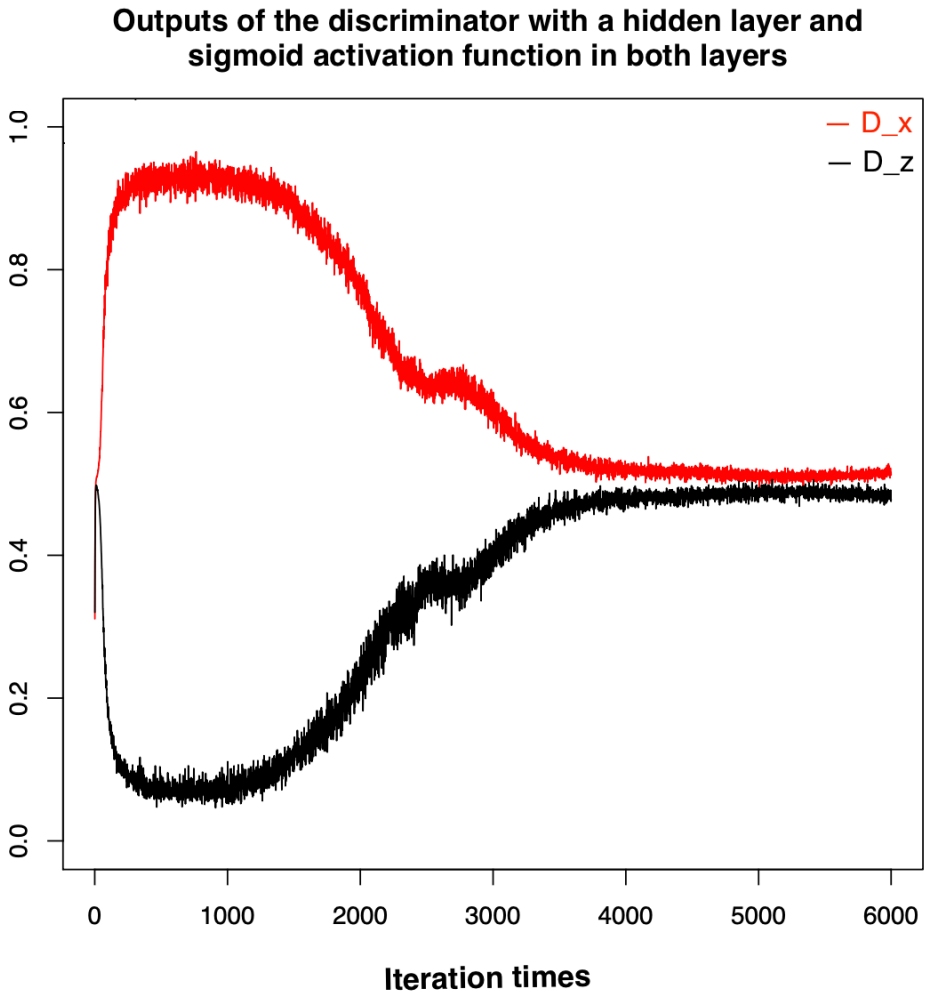


Figure 55: Outputs of the Discriminator 2

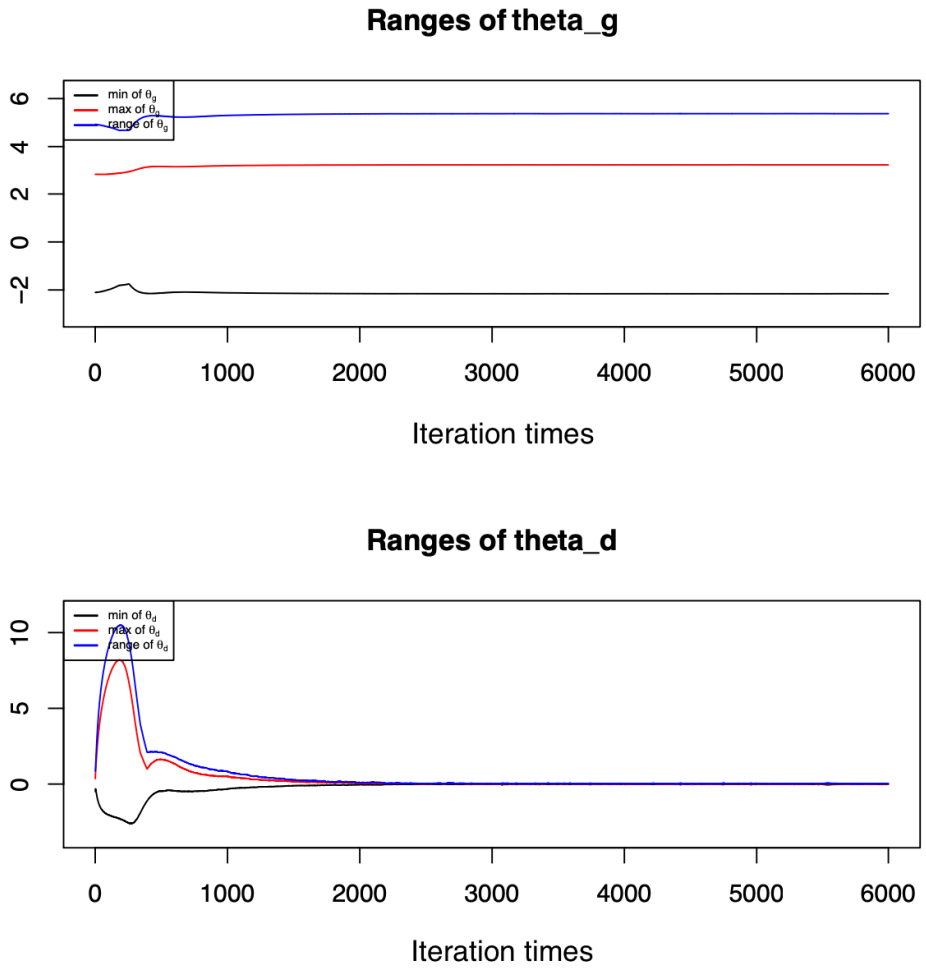


Figure 56: Ranges of the Parameters 1

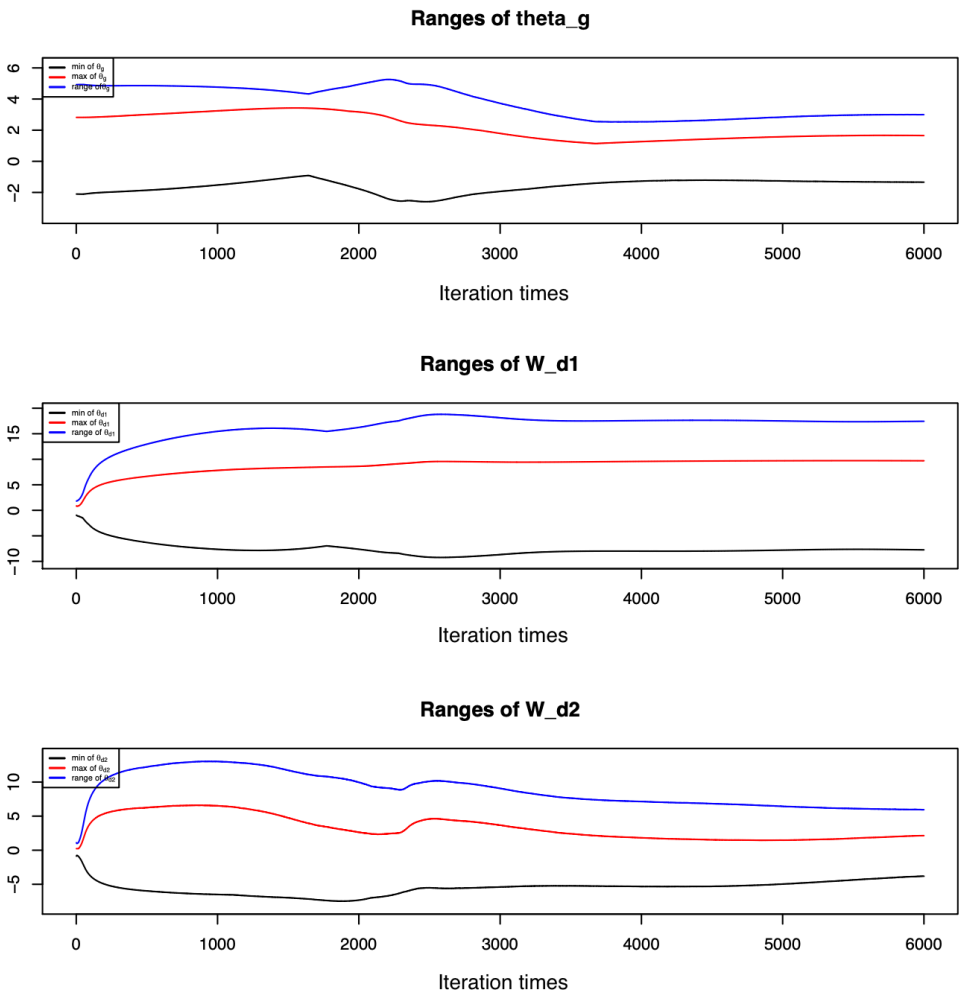


Figure 57: Ranges of the Parameters 2

b. Outstanding issues

Wider ranges of the initial parameters of the generator tend to worsen the results. Figure 58 to Figure 60 show density plots of θ^{g0} sampled from ranges $[-1, 1]$, $[-2, 2]$ and $[-3, 3]$ respectively in a case (t1p2r2). Density plots show unimodality in the first two figures; however, density curves of z_g gain bimodality in the third figure.

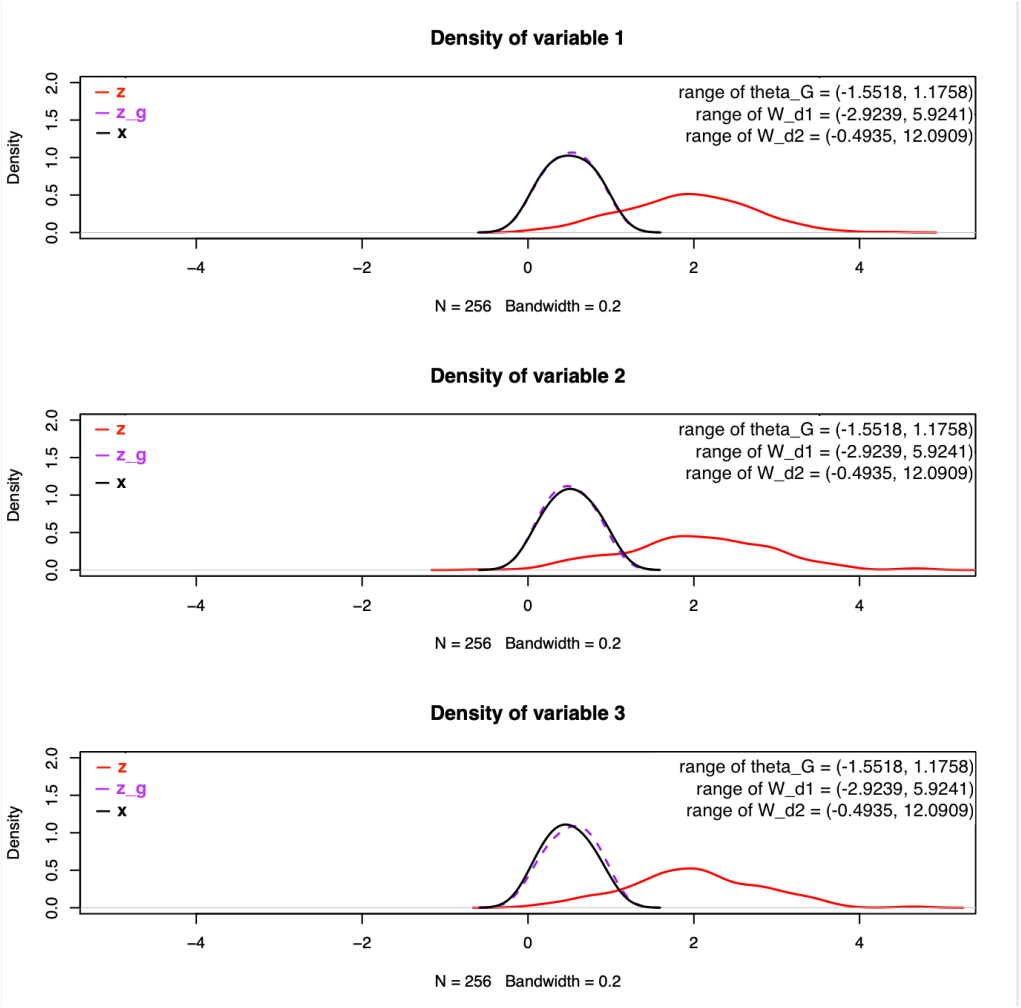


Figure 58: Density Plots 1

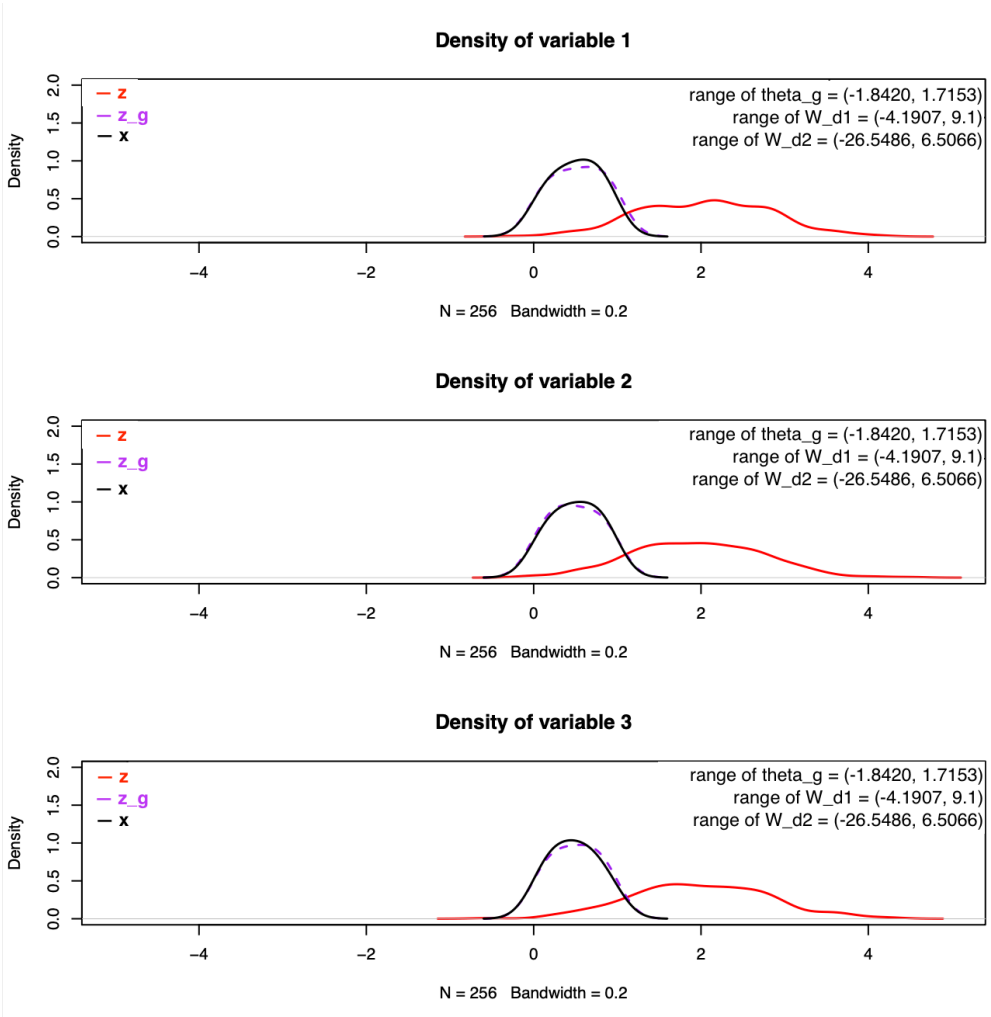


Figure 59: Density Plots 2

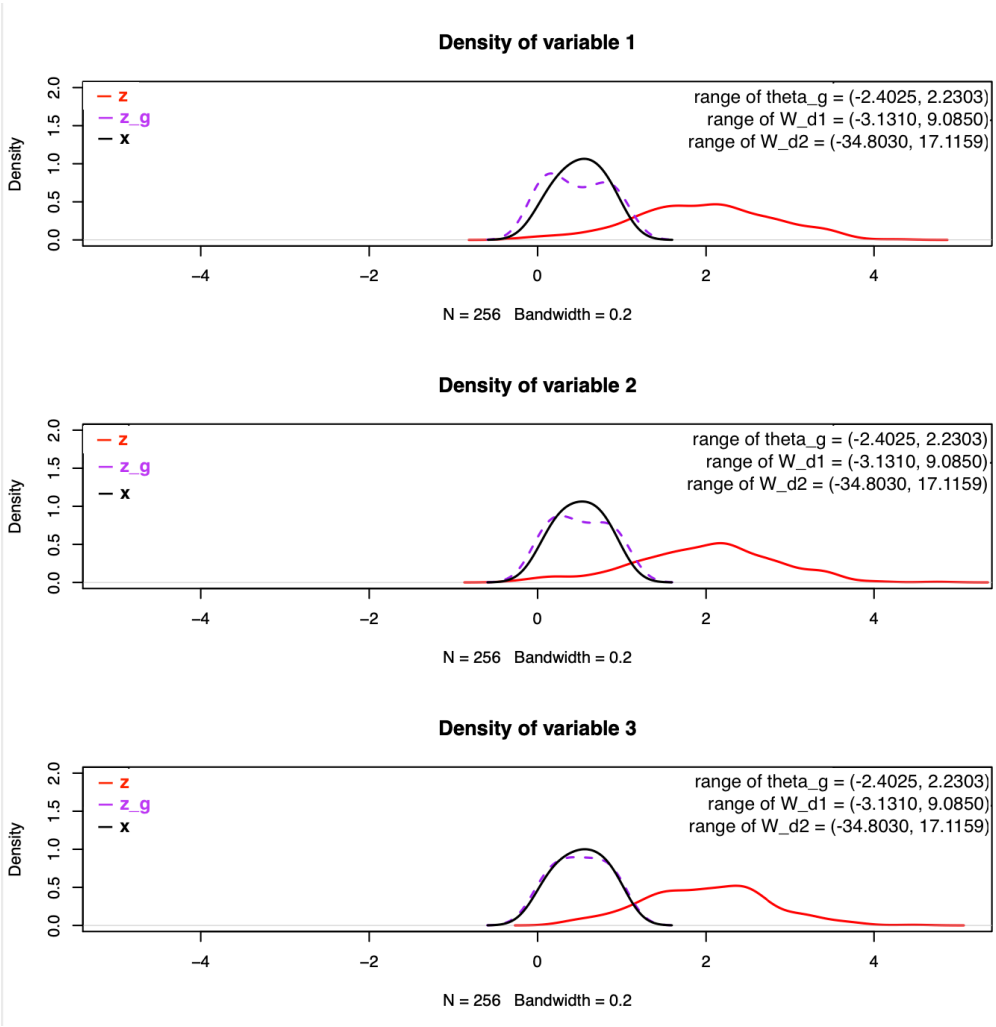


Figure 60: Density Plots 3

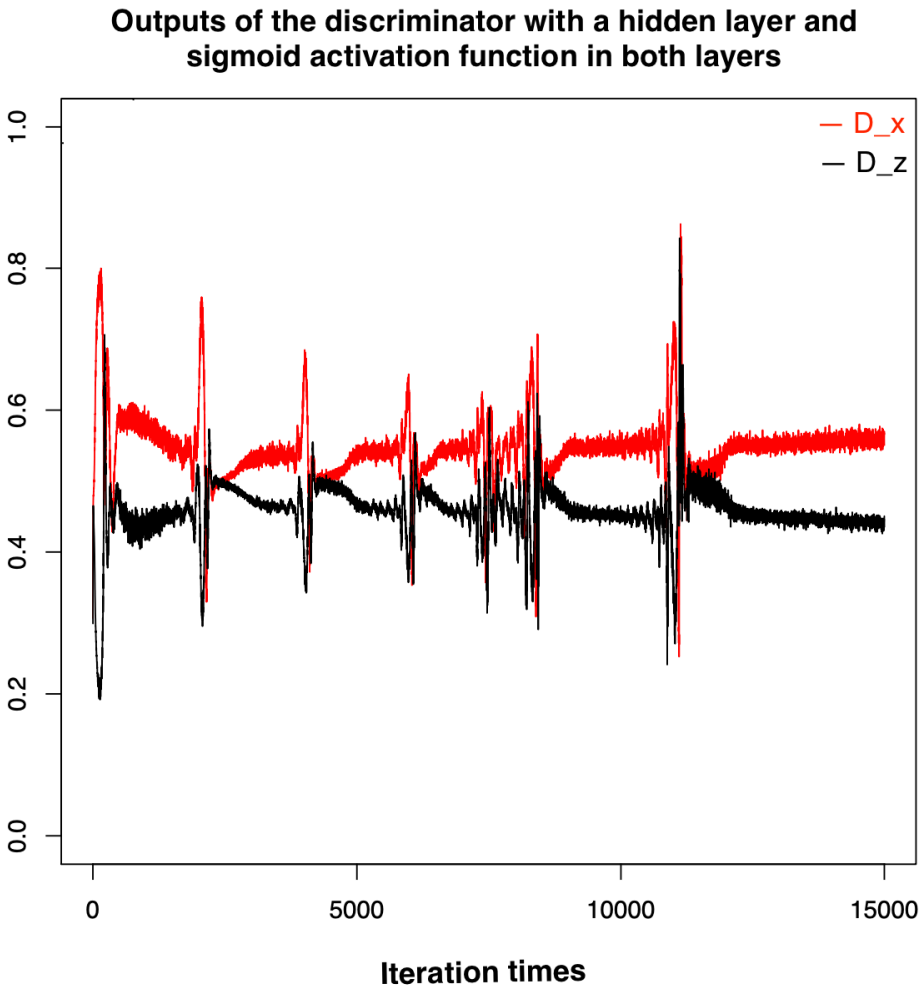


Figure 61: Outputs of the Discriminator

Discriminator shows a periodic trend and fails to converge. Figure 61 displays outputs of the discriminator corresponding to Figure 60. D_x and D_z show periodic changes and fail to converge to one half after fifteen thousand epochs. It proved once again that a wider range of the initial parameters of the generator can lead to bad performance of an algorithm. In these cases, the discriminator can differentiate between real data and fake data during the periodic update, but it cannot guide the generated distribution to move closer to the target. Moreover, when the parameters are large, the generated data is more likely to show a bimodal shape with the two peaks of the generated distribution swinging back and forth and the generated distributions failing to converge to the target.

5 Conclusions

We identify several pathological phenomena in the theoretical analysis and experiments.

When noise prior distribution is normal and generator is a one-layer perceptron using sigmoid activation function, the class of normal target distributions is not learnable through the algorithm consisting of this generator and a weak discriminator. We prove that in this case, the generated distribution is logit-normal. Then for a normal target, the generator will never generate a distribution which can converge to the target. From a distribution learning-theoretic view, we prove that the class of normal distributions is not learnable in this case. Furthermore, a weak discriminator can make the generator's work even more difficult, in the sense that even for a specific achievable generalization error ϵ, the time taken by the generator to learn this accuracy will depend directly on the weakness of the discriminator.

Sigmoid activation function performs better than Maxout in the algorithm with no hidden layers in the discriminator. In the algorithm with sigmoid activation function in the discriminator, the generated distribution can recover the target distribution's location. However, the algorithm with Maxout layer fails to learn the target's location. It may be because of the sparsity introduced to the discriminator along with the Maxout layer. Since there are only three target variables in our cases and the discriminator is a one-layer perceptron, the sparsity along with Maxout may worsen the performance by filtering out some important information.

The algorithms with sigmoid activation function and without hidden layers in the discriminator have several pathological behaviors. The probabilities D_x and D_z both converge to one half, but the generated distribution does not converge to the target. θ^d collapses to zero through training and range of θ^g keeps almost flat. The algorithm can learn the targets' locations, but it fails to learn the targets' squared scales in most cases. When the range of θ^{g0} is small, the generated distribution can learn the unimodality of the target, but has a sharper shape than the target. As the range of θ^{g0} becomes larger,

the generated distribution gains bimodality; the algorithm requires longer training time to achieve its best; outputs and parameters of the discriminator fluctuate more exaggeratedly throughout training.

The algorithm with a hidden layer in the discriminator performs better than the other two algorithms; however, there are several pathological behaviors as well. The generated distribution fits the target better using the algorithm with a hidden layer in the discriminator than using the other algorithms especially when the range of θ^{g0} is large. θ^d does not collapse to zero any more and range of θ^g does not stay almost flat any more. However, in some cases, the generated distribution still gains bimodality when the range of θ^{g0} becomes wider. The discriminator's capacity improves, but sometimes the discriminator tends to have a periodic trend and fails to converge.

6 Limitations

In this thesis, we only focus on some aspects and explore their effects on shallow GANs. This simple model has the advantage that we can obtain concrete theoretical and experimental insight. However, even here are more values or options of the aspects and more aspects that could be explored in the future.

In the learning-theoretic analysis, we only analyze the learnability of a class of distributions using a shallow GAN algorithm with a specific setting. There are more settings open to analyze, for example, the discriminator or generator with other activation functions or with more hidden layers, other noise distributions and etc.

In the experiments, there are different pathological phenomena when we use different algorithms as below:

(i) We choose several seeds to sample mini-batches of data. In some cases, the results vary significantly when different mini-batches of data are used. Due to the limitation of time, we did not sample enough many mini-batches of data in some cases.

(ii) Maxout layer in Algorithm 1 does not perform well in our cases. The performance may be better, if we use examples with more variables or introduce more hidden layers in the discriminator.

(iii) Distribution generated by Algorithm 2 gains bimodality gradually when range of θ^{g0} becomes wider. This pathological phenomenon is alleviated slightly when we add a hidden layer to the discriminator; however, it still exists. Expand the hidden layer or add another hidden layers to the discriminator may alleviate this problem further.

(iv) In some cases when p_{data} and p_G are close enough, p_{data} is unimodal but p_G can be bimodal, so p_G may never converge to p_{data}. Since the bimodality of p_G comes from the logit-normal distribution, the bimodality may be eliminated if we use a uniform prior distribution to learn a target normal distribution.

References

[1] Goodfellow, Ian J., Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative Adversarial Networks." ArXiv:1406.2661 [Cs, Stat], 2014. http://arxiv.org/abs/1406.2661.

[2] Langr, Jakub and Vladimir Bok, 2019. Gans In Action: Deep Learning With Generative Adversarial Networks. Manning Publications.

[3] Creswell, Antonia, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A. Bharath. "Generative Adversarial Networks: An Overview." IEEE Signal Processing Magazine 35, no. 1 (January 2018): 53–65. https://doi.org/10.1109/MSP.2017.2765202.

[4] Hong, Yongjun, Uiwon Hwang, Jaeyoon Yoo, and Sungroh Yoon. "How Generative Adversarial Networks and Their Variants Work: An Overview." ACM Computing Surveys 52, no. 1 (February 13, 2019): 1–43. https://doi.org/10.1145/3301282.

[5] Hazan, Tamir, George Papandreou, Daniel Tarlow. "Adversarial Perturbations of Deep Neural Networks," in Perturbations, Optimization, and Statistics , MITP, 2017, pp.311-342.

[6] Salimans, Tim, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. "Improved Techniques for Training GANs." ArXiv:1606.03498 [Cs], 2016. http://arxiv.org/abs/1606.03498.

[7] Goodfellow, Ian. "NIPS 2016 Tutorial: Generative Adversarial Networks." ArXiv:1701.00160 [Cs], 2017. http://arxiv.org/abs/1701.00160.

[8] Heusel, Martin, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium." In ArXiv:1706.08500 [Cs, Stat], 2018. http://arxiv.org/abs/1706.08500.

[9] Bryan, Paget. “An Introduction to Generative Adversarial Networks.” Université d’Ottawa / University of Ottawa, 2019. http://hdl.handle.net/10393/39603.

[10] Hinkelmann, Knut. “Neural Networks, p. 7.” University of Applied Sciences Northwestern Switzerland. http://didattica.cs.unicam.it/lib/exe/fetch.php?media=didattica:magistrale:kebi:ay_1718:ke-11_neural_networks.pdf.

[11] Wikipedia contributors, “Distribution learning theory,” Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Distribution_learning_theory&oldid=936609464 (accessed June 18, 2020).

[12] Arjovsky, Martin, Soumith Chintala, and Léon Bottou. “Wasserstein GAN.” In ArXiv:1701.07875 [Cs, Stat], 2017. http://arxiv.org/abs/1701.07875.

[13] An, Dongsheng, Yang Guo, Min Zhang, Xin Qi, Na Lei, Shing-Tung Yau, and Xianfeng Gu. “AE-OT-GAN: Training GANs from Data Specific Latent Distribution.” ArXiv:2001.03698 [Cs, Eess], 2020. http://arxiv.org/abs/2001.03698.

[14] An, Dongsheng, Yang Guo, Na Lei, Zhongxuan Luo, Shing-Tung Yau, and Xianfeng Gu. “AE-OT: A NEW GENERATIVE MODEL BASED ON EX- TENDED SEMI-DISCRETE OPTIMAL TRANSPORT.” 19, 2020. https://openreview.net/pdf?id=HkldyTNYwH.

[15] Kearns, Michael, Yishay Mansour, Dana Ron, Ronitt Rubinfeld, Robert E. Schapire, and Linda Sellie. “On the Learnability of Discrete Distributions.” In Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing - STOC ’94, 273–82. Montreal, Quebec, Canada: ACM Press, 1994. https://doi.org/10.1145/195058.195155.

[16] Arora, Sanjeev, and Yi Zhang. “Do GANs Actually Learn the Distribution? An Empirical Study.” ArXiv:1706.08224 [Cs], 2017. http://arxiv.org/abs/1706.08224.

[17] Duda, Richard O.. "Decision Boundaries" June 10, 1997. https://www.cs.princeton.edu/courses/archive/fall08/cos436/Duda/PR_simp/bndrys.htm.

[18] Lucic, Mario, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. "Are GANs Created Equal? A Large-Scale Study." In ArXiv:1711.10337 [Cs, Stat], 2018. http://arxiv.org/abs/1711.10337.

[19] Khayatkhoei, Mahyar, Ahmed Elgammal, and Maneesh Singh. "Disconnected Manifold Learning for Generative Adversarial Networks." ArXiv:1806.00880 [Cs, Stat], 2019. http://arxiv.org/abs/1806.00880.

[20] Salimans, Tim, and Diederik P. Kingma. "Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks." In ArXiv:1602.07868 [Cs], 2016. http://arxiv.org/abs/1602.07868.

[21] Behnke, Sven (2003). Hierarchical Neural Networks for Image Interpretation. Lecture Notes in Computer Science. 2766. Springer. doi:10.1007/b11963. ISBN 978-3-540-40722-5.

[22] Brownlee, Jason. "Understand the Impact of Learning Rate on Neural Network Performance." Machine Learning Mastery, January 25, 2019. https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/.

[23] Arjovsky, Martin, and Léon Bottou. "Towards Principled Methods for Training Generative Adversarial Networks." In ArXiv:1701.04862 [Cs, Stat], 2017. http://arxiv.org/abs/1701.04862.

[24] Aitchison, J, and S M Shen. "Logistic-Normal Distributions: Some Properties and Uses." Biometrika 67, no. 2 (August 1, 1980): 261–272. https://doi.org/10.1093/biomet/67.2.261.

[25] Frederic, Patrizio, and Frank Lad. "Two Moments of the Logitnormal Distribution." Communications in Statistics - Simulation and Computation 37, no. 7 (August 2008): 1263–69. https://doi.org/10.1080/03610910801983178.

[26] Casella, G. and Berger, R., 2001. Statistical Inference. 2nd ed. Belmont, CA: Brooks/COle Cengage Learning, pp.47-68.

[27] Rosasco, Lorenzo. "RegML 2016 Class 1 Statistical Learning Theory." June 27, 2016. http://lcsl.mit.edu/courses/regml/regml2016/slides/lect1.pdf.

[28] Wikipedia contributors, "Jensen's inequality," Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Jensen%27s_inequality&oldid=962626918 (accessed June 18, 2020).

[29] Goodfellow, Ian J., David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. "Maxout Networks." ArXiv:1302.4389 [Cs, Stat], 2013. http://arxiv.org/abs/1302.4389.

[30] Lemarechal, Claude. "Cauchy and the Gradient Method." Documenta Mathematica ISMP (2012) 251–254 (2012): 4.

[31] Rosenberg, David. "STAT C206A / MATH C223A : Stein's method and applications 1. Lecture 2" Aug 29, 2007. https://statweb.stanford.edu/ souravc/Lecture2.pdf.

[32] Wikipedia contributors, "Pinsker's inequality," Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Pinsker%27s_inequality&oldid=961905312 (accessed June 18, 2020).

[33] Wikipedia contributors, "Truncated normal distribution," Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Truncated_normal _distribution&oldid=958683561 (accessed June 18, 2020).

[34] Turán G. (1994) Computational Learning Theory and Neural Networks: A Survey of Selected Topics. In: Roychowdhury V., Siu KY., Orlitsky A. (eds) Theoretical Advances in Neural Computation and Learning. Springer, Boston, MA.

[35] Choksi, Rustum, 2020. Partial Differential Equations: A First Course. Volume I^*. pp.200. Department of Mathematics and Statistics, McGill University.

[36] Frederic, Patrizio, and Frank Lad. "A Technical Note on the Logitnormal distribution." University of Canterbury Mathematics and Statistics Research Report (2003).

[37] AITCHISON, J., and C. B. BEGG, Statistical diagnosis when basic cases are not classified with certainty, Biometrika, Volume 63, Issue 1, 1976, Pages 1–12, https://doi.org/10.1093/biomet/63.1.1.

[38] Kearns, Michael, Yishay Mansour, Dana Ron, Ronitt Rubinfeld, Robert E. Schapire, and Linda Sellie. "On the learnability of discrete distributions." In Proceedings of the twenty-sixth annual ACM symposium on Theory of computing, pp. 273-282. 1994.

A Functions and Pseudo-codes

All functions are highlighted in color red.

A.1 Algorithm 1

Functions:

- Sigmoid(z): Compute Sigmoid transformation of z. Input: random noise z; output: $z = 1/(1 + e^{-z})$.
- Generator(θ^g,z): Compute output of generator calling Sigmoid function. Input: noise samples z and parameters in generator θ^g; output: generated samples z_g.
- Mask(x,μ): Compute new datasets after using a binary mask. Input: examples x or generated samples z_g and a binary mask μ; output: examples $\tilde{x}$ and generated samples $\tilde{z}_g$.
- Discriminator-MaxSig(x,θ^d,μ,v): Compute output of discriminator calling Mask and Sigmoid function. Input: generated samples z_g or examples x, parameters in discriminator θ^d, a binary mask μ and a coefficient vector v for combining Maxout units ($e.g. v = (1, -1)$); output: D_z(input z_g) or D_x(input x), new datasets $\tilde{x}$ or $\tilde{z}_g$ and the argmax among nodes in a Maxout unit h_{argmax}.
- Gradient-D-MaxSig(x,z_g,θ^d,μ,v): Compute gradient of θ^d calling Discriminator-MaxSig function. Input: examples x, generated samples z_g and parameters in discriminator θ^d, a binary mask μ and a coefficient vector v for combining Maxout units; output: $\nabla_{\theta^d} J_d$: gradient of θ^d.
- Gradient-G-MaxSig(z,θ^g,θ^d,D_z,$h_{z-argmax}$,μ): Compute gradient of θ^g calling Generator function. Input: noise samples z, parameters in generator θ^g, a binary mask μ, parameters in discriminator θ^d and other information from discriminator D_z and $h_{z-argmax}$ when z_g is the input of discriminator; output: $\nabla_{\theta^g} J_g$: gradient of θ^g.

- oGAN-MaxSig(m,p,p_Z,p_{data},...,a,b,c,d,μ,α,β,N,L,v,$seeds$,Q,K): whole algorithm including θ^d and θ^g update and visualization calling all the above functions.

Pseudo-codes:

Algorithm 1 GAN with Maxout layer (function oGAN-MaxSig)

Set a seed.
Set initial parameters θ^{d0} and θ^{g0} by sampling from continuous uniform distribution.
Create numerical vectors D_x and D_z with length N to save the results of discriminator in N iterations.
Create numerical matrix $norm(\theta^g)$ and $norm(\theta^d)$ to save norms in each iteration.
for number of training iterations N **do**
  for L steps **do**
    Sample mini-batch of m noise samples $\{z^{(1)}, ..., z^{(m)}\}$ from noise prior $p_Z(z)$.
    Generator: Generate samples using Generator given $\{z^{(1)}, ..., z^{(m)}\}$ sampled above.
    Sample mini-batch of m examples $\{x^{(1)}, ..., x^{(m)}\}$ from target distribution $p_{data}(x)$.
    Update the discriminator by ascending its stochastic gradient:
    Gradient-D-Sigmoid: $\nabla_{\theta^d} J_d = \nabla_{\theta^d} \frac{1}{m} \sum_{i=1}^{m} [logD(x^{(i)}|\theta^d) + log(1 - D(z_g^{(i)}|\theta^d))]$
    $\theta^{d_{new}} = \theta^d + \alpha \nabla_{\theta^d} J_d$
  end for
  Sample mini-batch of m noise samples $\{z^{(1)}, ..., z^{(m)}\}$ from noise prior $p_Z(z)$.
  Generator: Compute generated samples in generator given new noise samples.
  Discriminator-MaxSig: Compute information from discriminator (D_z and $h_{z-argmax}$) given new generated samples and updated θ^d
  Update the generator by descending its stochastic gradient:
  Gradient-G-Sigmoid: $\nabla_{\theta^g} J_g = \nabla_{\theta^g} \frac{1}{m} log[1 - D(G(z^{(i)}|\theta^g)|\theta^{d_{new}})]$
  $\theta^{g_{new}} = \theta^g - \beta \nabla_{\theta^g} J_g$
  Compute and save norms of θ^g and θ^d.
  Visualize empirical density plots of examples x, noise samples z and generated samples z_g.

22: Compute mean of outputs of discriminator D_z.

23: Discriminator-MaxSig: Compute outputs of discriminator D_x given updated θ^d and new noise samples z for visualization and compute its mean value.

24: **end for**

25: Visualize mean values of D_x and D_z.

26: Visualize norms of θ^g and θ^d.

27: Save latest θ^g.

A.2 Algorithm 2

Functions:

- Sigmoid(z): Compute Sigmoid transformation of z. Input: noise samples z; output: $z = 1/(1 + e^{-z})$.
- Generator(θ^g,z): Compute output of generator calling Sigmoid function. Input: noise samples z and parameters in generator θ^g; output: generated samples z_g.
- Discriminator-Sigmoid(x,θ^d): Compute output of discriminator calling Sigmoid function. Input: generated samples z_g or examples x, and parameters in discriminator θ^d; output: D_z (input z_g) or D_x (input x).
- Gradient-D-Sigmoid(x,z_g,θ^d): Compute gradient of θ^d calling Discriminator-Sigmoid function. Input: examples x, generated samples z_g and parameters in discriminator θ^d; output: $\nabla_{\theta^d} J_d$: gradient of θ^d.
- Gradient-G-Sigmoid(z,θ^g,θ^d): Compute gradient of θ^g calling Generator and Sigmoid function. (Sigmoid function and θ^d provide the information D_z from discriminator when input is z_g.) Input: noise samples z, parameters in generator θ^g and parameters in discriminator θ^d; output: $\nabla_{\theta^g} J_g$: gradient of θ^g.
- oGAN-Sigmoid(m,p,p_Z,p_{data},...,a,b,c,d,α,β,N,L,$seeds$): whole algorithm including θ^d and θ^g update and visualization calling all the above functions.

Pseudo-codes:

Algorithm 2 GAN with Sigmoid Activation Function (function oGAN-Sigmoid)

Set a seed.

Set initial parameters θ^{d0} and θ^{g0} by sampling from continuous uniform distribution.

Create numerical vectors D_x and D_z with length N to save the results of discriminator in N iterations.

Create numerical matrix $norm(\theta^g)$ and $norm(\theta^d)$ to save norms in each iteration.

for number of training iterations N **do**

  for L steps **do**

    Sample mini-batch of m noise samples $\{z^{(1)}, ..., z^{(m)}\}$ from noise prior $p_Z(z)$.

    Generator: Generate samples using Generator given $\{z^{(1)}, ..., z^{(m)}\}$ sampled above.

    Sample mini-batch of m examples $\{x^{(1)}, ..., x^{(m)}\}$ from target distribution $p_{data}(x)$.

    Update the discriminator by ascending its stochastic gradient:

    Gradient-D-Sigmoid: $\nabla_{\theta^d} J_d = \nabla_{\theta^d} \frac{1}{m} \sum_{i=1}^{m} [logD(x^{(i)}|\theta^d) + log(1 - D(z_g^{(i)}|\theta^d))]$

    $\theta^{d_{new}} = \theta^d + \alpha \nabla_{\theta^d} J_d$

  end for

  Sample mini-batch of m noise samples $\{z^{(1)}, ..., z^{(m)}\}$ from noise prior $p_Z(z)$.

  Update the generator by descending its stochastic gradient:

  Gradient-G-Sigmoid: $\nabla_{\theta^g} J_g = \nabla_{\theta^g} \frac{1}{m} log[1 - D(G(z^{(i)}|\theta^g)|\theta^{d_{new}})]$

  $\theta^{g_{new}} = \theta^g - \beta \nabla_{\theta^g} J_g$

  Compute and save norms of θ^g and θ^d.

  Visualize empirical density plots of examples x, noise samples z and generated samples z_g.

  Compute outputs of discriminator D_x and D_z given updated θ^d and new samples z for visualization.

end for

Visualize outputs of discriminator D_x and D_z.

Visualize norms of θ^g and θ^d.

24: Save latest θ^g.

A.3 Algorithm 3

Functions:

- Sigmoid(z): Compute Sigmoid transformation of z. Input: noise samples z; output: $z = 1/(1 + e^{-z})$.
- Generator(θ^g,z): Compute output of generator calling Sigmoid function. Input: noise samples z and parameters in generator θ^g; output: generated samples z_g.
- Discriminator-Sigmoid-h(x,$\bar{W}^{(1)}$,$\bar{W}^{(2)}$): Compute values in hidden layer and output layer of discriminator calling Sigmoid function. Input: generated samples z_g or examples x, and parameters in discriminator θ^d ($\bar{W}^{(1)}$ and $\bar{W}^{(2)}$); output: D_z and z_g^1(input z_g), or D_x and x1(input x).
- Gradient-D-Sigmoid-h(x,z_g, $\bar{W}^{(1)}$, $\bar{W}^{(2)}$): Compute gradient of θ^d calling Discriminator-Sigmoid-h function. Input: examples x, generated samples z_g and parameters in discriminator θ^d ($\bar{W}^{(1)}$ and $\bar{W}^{(2)}$); output: gradients $\nabla_{\theta^d} J_d$.
- Gradient-G-Sigmoid-h(z,θ^g,D_z,z_g^1,$\bar{W}^{(1)}$,$\bar{W}^{(2)}$): Compute gradient of θ^g calling Generator function. Input: noise samples z, parameters in generator θ^g and information from discriminator (D_z, z_g^1) and parameters in discriminator θ^d; output: $\nabla_{\theta^g} J_g$: gradient of θ^g.
- oGAN-Sigmoid-h(m,p,Q,p_Z,p_{data},...,a,b,c,d,α,β,N,L,$seeds$): whole algorithm including θ^d and θ^g update and visualization calling all the above functions.

Pseudo-codes:

Algorithm 3 GAN with a hidden layer and Sigmoid transformation function in discriminator (function oGAN-Sigmoid-h)

1: Set a seed.

2: Set initial parameters θ^{d0} and θ^{g0} by sampling from continuous uniform distribution.

Create numerical vectors D_x and D_z with length N to save the results of discriminator in N iterations.

Create numerical matrix $norm(\theta^g)$ and $norm(\theta^d)$ to save norms in each iteration.

for number of training iterations N **do**

  for L steps **do**

    Sample mini-batch of m noise samples $\{z^{(1)}, ..., z^{(m)}\}$ from noise prior $p_Z(z)$.

    Generator: Generate samples using Generator given $\{z^{(1)}, ..., z^{(m)}\}$ sampled above.

    Sample mini-batch of m examples $\{x^{(1)}, ..., x^{(m)}\}$ from target distribution $p_{data}(x)$.

    Update the discriminator by ascending its stochastic gradient:

    Gradient-D-Sigmoid-h: $\nabla_{W^{(1)}} J_d = \nabla_{W^{(1)}} \frac{1}{m} \sum_{i=1}^{m} [logD(x^{(i)}|W^{(1)}) + log(1 - D(z_g^{(i)}|W^{(1)}))]$

    Gradient-D-Sigmoid-h: $\nabla_{W^{(2)}} J_d = \nabla_{W^{(2)}} \frac{1}{m} \sum_{i=1}^{m} [logD(x^{(i)}|W^{(2)}) + log(1 - D(z_g^{(i)}|W^{(2)}))]$

    $W^{(1)_{new}} = W^{(1)} + \alpha \nabla_{W^{(1)}} J_d$

    $W^{(2)_{new}} = W^{(2)} + \alpha \nabla_{W^{(2)}} J_d$

  end for

  Sample mini-batch of m noise samples $\{z^{(1)}, ..., z^{(m)}\}$ from noise prior $p_Z(z)$.

  Get information from discriminator (D_z and z_g^1) given newly updated θ^d.

  Update the generator by descending its stochastic gradient:

  Gradient-G-Sigmoid-h: $\nabla_{\theta^g} J_g = \nabla_{\theta^g} \frac{1}{m} log[1 - D(G(z^{(i)}|\theta^g)|\theta^{d_{new}})]$

  $\theta^{g_{new}} = \theta^g - \beta \nabla_{\theta^g} J_g$

  Compute and save norms of θ^g and θ^d.

  Visualize empirical density plots of examples x, noise samples z and generated samples z_g.

  Compute outputs of discriminator D_x and D_z given updated θ^d and new samples z for visualization.

end for

Visualize outputs of discriminator D_x and D_z.

Visualize norms of θ^g and θ^d.

Save latest θ^g.

B Proof of Gradients

B.1 Algorithm 1

1. In discriminator:

Apply a binary mask to input examples x or generated samples z_g,

$$\mu = \{\mu_{q,j}\} \in \mathbb{R}^{Q\times p},\ j = 1, ..., p,\ q = 1, ..., Q$$

$$\tilde{x}_q = \{\tilde{x}^{(i)}_{j,q}\} \in \mathbb{R}^{m\times p},\ i = 1, ..., m$$

$$\tilde{x}^{(i)}_{j,q} = x^{(i)}_j \mu_{q,j} \tag{B.1}$$

$$\tilde{z}_{g\ q} = \{\tilde{z}^{(i)}_{g\ j,q}\} \in R^{m\times p} \tag{B.2}$$

$$\tilde{z}^{(i)}_{g\ j,q} = z^{(i)}_{g\ j}\mu_{j,q}. \tag{B.3}$$

In q^{th} Maxout unit, input examples x,

$$\theta^d = ((W^d)^T\ b^d)^T = (W^d_1, ..., W^d_p, b^d)^T \in \mathbb{R}^{K\times(p+1)} \tag{B.4}$$

$$\theta^d = \{\theta^d_{j,k}\},\ j = 1, ..., p+1,\ k = 1, ..., K$$

$$\hat{x_q} = (\tilde{x_q}\ 1) = (x_{q,1}, ..., x_{q,p}, 1) \in \mathbb{R}^{m\times(p+1)}$$

$$h_{qK}(x^{(i)}) = \theta^d(\hat{x_q}^{(i)})^T \in \mathbb{R}^{K\times 1} \tag{B.5}$$

$$h_q(x^{(i)}) = max\{h_{qK}(x^{(i)})\},\ q = 1, ..., Q$$

$$k_q(x^{(i)}) = argmax_k\{h_{qK}(x^{(i)})\}$$

$$h(x^{(i)}) = \{h_q(x^{(i)})\}. \tag{B.6}$$

In q^{th} Maxout unit, input examples z_g,

$$\theta^d = ((W^d)^T\ b^d)^T = (W^d_1, ..., W^d_p, b^d)^T \in \mathbb{R}^{K\times(p+1)} \tag{B.7}$$

$$\theta^d = \{\theta^d_{j,k}\},\ j = 1, ..., p+1,\ k = 1, ..., K$$

$$\hat{z_{g\ q}} = (\tilde{z_{g\ q}}\ 1) = (z_{g\ q,1}, ..., z_{g\ q,p}, 1) \in \mathbb{R}^{m\times(p+1)}$$

$$h_{qK}(z^{(i)}) = \theta^d(\hat{z_g}_q^{(i)})^T \in \mathbb{R}^{K\times 1} \tag{B.8}$$

$$h_q(z^{(i)}) = max\{h_{qK}(z^{(i)})\},\ q = 1, ..., Q$$

$$k_q(z^{(i)}) = argmax_k\{h_{qK}(z^{(i)})\}$$

$$h(z^{(i)}) = \{h_q(z^{(i)})\}. \tag{B.9}$$

In output layer,

$$v = (v_1, ..., v_Q)^T \in \mathbb{R}^{Q\times 1}$$

$$D_x^{(i)} = S(h(x^{(i)})^T v) \tag{B.10}$$

$$D_z^{(i)} = S(h(z^{(i)})^T v) \tag{B.11}$$

2. Update parameters in discriminator:

Loss function is,

$$\begin{aligned} J_d &= \frac{1}{m}\sum_{i=1}^{m}[logD(x^{(i)}|\theta^d) + log(1 - D(z_g^{(i)}|\theta^d))] \\ &= \frac{1}{m}\sum_{i=1}^{m}[logD_x^{(i)} + log(1 - D_z^{(i)})]. \end{aligned} \tag{B.12}$$

Compute gradients $\nabla_{W^d} J_d^{(i)}$.

$$\nabla_{W^d} J_d = \{\nabla_{W_j^d} J_d\}, j = 1, ..., p$$

$$\nabla_{W^d} J_d = \frac{1}{m}\sum_{i=1}^{m} \nabla_{W^d} J_d^{(i)} \tag{B.13}$$

$$W^d = \{w_{k,j}^d\},\ k = 1, ..., K,\ j = 1, ..., p$$

$$\nabla_{W^d} J_d^{(i)} = \{\nabla_{w_{k,j}^d} J_d^{(i)}\}$$

$$\nabla M = \{\nabla M_{k,j,q}^{(i)}\} \in \mathbb{R}^{K\times(p+1)\times Q\times m}$$

$$\nabla M_{,j,q}^{(i)} = 0,\ \nabla M_{k_q(x^{(i)}),j,q}^{(i)} = x_{q,j}^{(i)}$$

$$\nabla_{w_{k,j}^d} J_d^{(i)} = \frac{\partial}{\partial w_{k,j}^d}\left[logD_x^{(i)} + log(1 - D_z^{(i)})\right] \tag{B.14}$$

$$= (1 - D_x^{(i)}) \nabla M_{k,q,}^{(i)} v$$

$$\nabla_{w_{q,j}^d} J_d = \frac{1}{m} \sum_{i=1}^{m} \left[(1 - D_x^{(i)}) \nabla M_{k,q,}^{(i)} v \right]. \tag{B.15}$$

Update θ^d,

$$W^d = W^d + \alpha \nabla_{W^d} J_d. \tag{B.16}$$

3. In discriminator:

Get information from discriminator again using newly updated θ^d.

$$Input: \ \mu = \{\mu_{q,j}\} \in \mathbb{R}^{Q \times p}, \ j = 1, ..., p, \ q = 1, ..., Q$$

$$\tilde{z}_{g\ q} = \{\tilde{z}_{g\ j,q}^{(i)}\} \in R^{m \times p}$$

$$\tilde{z}_{g\ j,q}^{(i)} = z_{g\ j}^{(i)} \mu_{j,q} \tag{B.17}$$

$$Maxoutunit: \ h_{qK}^{(i)} = \theta^d (\hat{z_{g\ q}}^{(i)})^T \in \mathbb{R}^{K \times 1} \tag{B.18}$$

$$h_q^{(i)} = max\{h_{qK}^{(i)}\}, \ q = 1, ..., Q$$

$$k_q^{(i)} = argmax_k \{h_{qK}^{(i)}\}$$

$$h^{(i)} = \{h_q^{(i)}\}$$

$$Output: \ D_z^{(i)} = S((h^{(i)})^T v). \tag{B.19}$$

4. In generator:

$$\begin{aligned} Input\ layer: \ \hat{z} &= (z\ 1), \ z = (z_1, ..., z_p) \\ Output\ layer: \ z_g &= S(\hat{z}\theta^g) \\ z_{g\ j}^{(i)} &= S(\sum_{u=1}^{p+1} z_u^{(i)} w_{u,j}^g), i = 1, ...m, \ j = 1, ..., p. \end{aligned} \tag{B.20}$$

5. Update parameters in generator:

Loss function is,

$$J_g = \frac{1}{m}\sum_{i=1}^{m} log(1 - D(G(z^{(i)}|\theta^g)\theta^d)) \tag{B.21}$$
$$= \frac{1}{m}\sum_{i=1}^{m} log(1 - D(z_g^{(i)}|\theta^d))$$
$$= \frac{1}{m}\sum_{i=1}^{m} log(1 - D_z^{(i)})$$

Given information from discriminator μ, D_z, k_q and θ^d, output of generator z_g and noise samples z, compute $\nabla_{W^g} J_g$.

$$\nabla_{W^g} J_g = \frac{1}{m}\sum_{i=1}^{m} \nabla_{W^g} J_g^{(i)} \tag{B.22}$$

$$\nabla_{W^g} J_g = \{\nabla_{w_{u,j}^g} J_g\},\ u = 1, ..., p,\ j = 1, ..., p$$

$$\nabla_{W^g} J_g^{(i)} = \nabla_{w_{u,j}^g} J_g^{(i)},\ u = 1, ..., p,\ j = 1, ..., p$$

$$\nabla_{w_{u,j}^g} J_g^{(i)} = -D_z^{(i)} \sum_{q=1}^{Q} \left[\theta^d_{k_q^{(i)},j}(z^{(i)}_{g\ j}\mu_{q,j})(1 - z^{(i)}_{g\ j}\mu_{q,j}) z_u^{(i)} v_q\right] \tag{B.23}$$

$$\nabla_{w_{u,j}^g} J_g = \frac{1}{m}\sum_{i=1}^{m} \nabla_{w_{u,j}^g} J_g^{(i)}. \tag{B.24}$$

Update θ^g,

$$W^g = W^g - \beta \nabla_{W^g} J_g \tag{B.25}$$

B.2 Algorithm 2

During iterations, there are five main steps for updating parameters in discriminator and generator.

1. In discriminator:

Given x, z_g and $\bar{\theta^d}$, compute output of discriminator. Sigmoid activation function is used in discriminator (Sigmoid function: S()).

Input $x^{(i)}$, i=1,...,m,

$$Input\ layer:\ \hat{x^{(i)}} = (x^{(i)}\ 1),\ x^{(i)} = (x_1^{(i)}, ..., x_p^{(i)})$$

$$Output\ layer:\ \theta^d = ((W^d)^T\ b^d)^T = (W_1^d, ..., W_p^d, b^d)^T,\ \theta^d = \{\theta_j^d\},\ j = 1, ..., p+1 \tag{B.26}$$

$$D_x^{(i)} = S(\hat{x^{(i)}}\theta^d) = S(\sum_{j=1}^{p+1} \hat{x_j^{(i)}}\theta_j^d). \tag{B.27}$$

Input $z_g^{(i)}$, i=1,...,m,

$$Input\ layer:\ \hat{z}_g^{(i)} = (z_g^{(i)}\ 1),\ z_g^{(i)} = (z_{g1}^{(i)}, ..., z_{gp}^{(i)})$$

$$Output\ layer:\ \theta^d = ((W^d)^T\ b^d)^T = (W_1^d, ..., W_p^d, b^d)^T,\ \theta^d = \{\theta_j^d\},\ j = 1, ..., p+1 \tag{B.28}$$

$$D_z^{(i)} = S(\hat{z_g^{(i)}}\theta^d) = S(\sum_{j=1}^{p+1} \hat{z}_g^{(i)}{}_j\theta_j^d). \tag{B.29}$$

2. Update parameters in discriminator:

Loss function is,

$$\begin{aligned} J_d &= \frac{1}{m}\sum_{i=1}^{m}[logD(x^{(i)}|\theta^d) + log(1 - D(z_g^{(i)}|\theta^d))] \\ &= \frac{1}{m}\sum_{i=1}^{m}[logD_x^{(i)} + log(1 - D_z^{(i)})]. \end{aligned} \tag{B.30}$$

Compute gradients $\nabla_{W^d} J_d^{(i)}$.

$$\begin{aligned} \nabla_{W^d} J_d &= \{\nabla_{W_j^d} J_d\}, j = 1, ..., p \\ \nabla_{W^d} J_d &= \frac{1}{m}\sum_{i=1}^{m} \nabla_{W^d} J_d^{(i)} \\ W^d &= \{w_j^d\},\ j = 1, ..., p \\ \nabla_{W^d} J_d^{(i)} &= \{\nabla_{w_j^d} J_d^{(i)}\}, j = 1, ..., p \\ \nabla_{w_j^d} J_d^{(i)} &= \frac{\partial}{\partial w_j^d}\left[logD_x^{(i)} + log(1 - D_z^{(i)})\right] \\ &= \frac{D_x^{(i)}(1 - D_x^{(i)})}{D_x^{(i)}} x_j^{(i)} + \frac{-D_z^{(i)}(1 - D_z^{(i)})}{1 - D_z^{(i)}} z_g^{(i)}{}_j \end{aligned} \tag{B.31, B.32}$$

$$= (1 - D_x^{(i)})x_j^{(i)} - D_z^{(i)} z_{g\ j}^{(i)}$$

$$\nabla_{w_j^d} J_d = \frac{1}{m} \sum_{i=1}^{m} \left[(1 - D_x^{(i)})x_j^{(i)} - D_z^{(i)} z_{g\ j}^{(i)} \right]. \tag{B.33}$$

Update θ^d,

$$W^d = W^d + \alpha \nabla_{W^d} J_d. \tag{B.34}$$

3. In discriminator:

Get information from discriminator again using newly updated θ^d,

$$\begin{aligned} Input: \ & z_g^{(i)}, \ i = 1, ..., m \\ Output: \ & D_z^{(i)} = S(\sum_{j=1}^{p+1} \hat{z_{g\ j}^{(i)}} \theta_j^d). \end{aligned} \tag{B.35}$$

4. In generator:

$$\begin{aligned} Input\ layer: \ & \hat{z} = (z\ 1), \ z = (z_1, ..., z_p) \\ Output\ layer: \ & z_g = S(\hat{z}\theta^g) \\ & z_{g\ j}^{(i)} = S(\sum_{u=1}^{p+1} z_u^{(i)} w_{u,j}^g), i = 1, ...m, \ j = 1, ..., p. \end{aligned} \tag{B.36}$$

5. Update parameters in generator:

Loss function is,

$$\begin{aligned} J_g &= \frac{1}{m} \sum_{i=1}^{m} log(1 - D(G(z^{(i)}|\theta^g)\theta^d)) \\ &= \frac{1}{m} \sum_{i=1}^{m} log(1 - D(z_g^{(i)}|\theta^d)) \\ &= \frac{1}{m} \sum_{i=1}^{m} log(1 - D_z^{(i)}) \end{aligned} \tag{B.37}$$

Given information from discriminator D_z and θ^d, output of generator z_g and noise samples z, compute $\nabla_{W^g} J_g$.

$$\nabla_{W^g} J_g = \frac{1}{m}\sum_{i=1}^{m} \nabla_{W^g} J_g^{(i)} \tag{B.38}$$

$$\nabla_{W^g} J_g = \{\nabla_{w_{u,j}^g} J_g\},\ u = 1, ..., p,\ j = 1, ..., p$$

$$\nabla_{W^g} J_g^{(i)} = \nabla_{w_{u,j}^g} J_g^{(i)},\ u = 1, ..., p,\ j = 1, ..., p$$

$$\begin{aligned}\nabla_{w_{u,j}^g} J_g^{(i)} &= \frac{\partial J_g^{(i)}}{\partial w_{u,j}^g} = \frac{\partial J_g^{(i)}}{\partial z_{g\ j}^{(i)}} \frac{\partial z_{g\ j}^{(i)}}{\partial w_{u,j}^g} \\ &= \frac{-D_z^{(i)}(1 - D_z^{(i)})}{1 - D_z^{(i)}} \theta_j^d \frac{\partial z_{g\ j}^{(i)}}{\partial w_{u,j}^g} \\ &= \frac{-D_z^{(i)}(1 - D_z^{(i)})}{1 - D_z^{(i)}} \theta_j^d z_{g\ j}^{(i)} (1 - z_{g\ j}^{(i)}) z_j^{(i)}\end{aligned} \tag{B.39}$$

$$\nabla_{w_{u,j}^g} J_g = \frac{1}{m}\sum_{i=1}^{m} \Big[\frac{-D_z^{(i)}(1 - D_z^{(i)})}{1 - D_z^{(i)}} \theta_j^d z_{g\ j}^{(i)} (1 - z_{g\ j}^{(i)}) z_j^{(i)}\Big] \tag{B.40}$$

Update θ^g,

$$W^g = W^g - \beta \nabla_{W^g} J_g \tag{B.41}$$

B.3 Algorithm 3

1. In discriminator:

Given x, z_g and θ^d ($\bar{W}^{d(1)}$ and $\bar{W}^{d(2)}$), compute output of discriminator. **Sigmoid activation function** is used in both hidden layer and output layer. (Sigmoid function: S())

Input $x^{(i)}$, i=1,...,m,

$$\begin{aligned} Input\ layer:\ & x^{(i)} = (x_1^{(i)}, ..., x_p^{(i)}),\ \hat{x}^{(i)} = (x^{(i)}\ 1) \\ Hidden\ layer:\ & x_q^{1(i)} = S(\hat{x}^{(i)} \bar{w}_{,q}^{d(1)}) \\ & = S(\sum_{j=1}^{p+1} \hat{x}_j^{(i)} \bar{w}_{j,q}^{d(1)}), q = 1, ..., Q \\ & x^{1(i)} = (x_1^{1(i)}, ..., x_Q^{1(i)}) \end{aligned} \tag{B.42}$$

$$\hat{x}^{1(i)} = (x^{1(i)}\ 1)$$
$$Output\ layer:\ D_x^{(i)} = S(\sum_{q=1}^{Q+1} \hat{x}_q^{1(i)} \bar{w}_q^{d(2)}). \tag{B.43}$$

Input $z_g^{(i)}$, i=1,...,m,

$$Input\ layer:\ z_g^{(i)} = (z_{g1}^{(i)}, ..., z_{gp}^{(i)}),\ \hat{z}_g^{(i)} = (z_g^{(i)}\ 1)$$
$$\begin{aligned} Hidden\ layer:\ z_{g\ q}^{1(i)} &= S(\hat{z}_g^{(i)} \bar{w}_{,q}^{d(1)}) \\ &= S(\sum_{j=1}^{p+1} \hat{z}_{g\ j}^{(i)} \bar{w}_{j,q}^{d(1)}), q = 1, ..., Q \end{aligned} \tag{B.44}$$
$$z_g^{1(i)} = (z_{g\ 1}^{1(i)}, ..., z_{g\ Q}^{1(i)})$$
$$\hat{z}_g^{1(i)} = (z_g^{1(i)}\ 1)$$
$$Output\ layer:\ D_z^{(i)} = S(\sum_{q=1}^{Q+1} \hat{z}_{g\ q}^{1(i)} \bar{w}_q^{d(2)}). \tag{B.45}$$

2. Update parameters in discriminator:

Loss function is,

$$\begin{aligned} J_d &= \frac{1}{m} \sum_{i=1}^{m} [logD(x^{(i)}|\theta^d) + log(1 - D(z_g^{(i)}|\theta^d))] \\ &= \frac{1}{m} \sum_{i=1}^{m} [logD_x^{(i)} + log(1 - D_z^{(i)})] \end{aligned} \tag{B.46}$$

Compute gradients $\nabla_{\theta^d} J_d^{(i)}$ ($\nabla_{W^{d(2)}} J_d$ and $\nabla_{W^{d(1)}} J_d$).

From output layer to hidden layer, compute $\nabla_{W^{d(2)}} J_d$,

$$\begin{aligned} \nabla_{W^{d(2)}} J_d &= \{\nabla_{w_q^{d(2)}} J_d\}, q = 1, ..., Q \\ \nabla_{W^{d(2)}} J_d^{(i)} &= \{\nabla_{w_q^{d(2)}} J_d^{(i)}\}, q = 1, ..., Q \\ \nabla_{w_q^{d(2)}} J_d^{(i)} &= \frac{\partial}{\partial w_q^{d(2)}} \left[logD_x^{(i)} + log(1 - D_z^{(i)})\right] \\ &= \frac{D_x^{(i)}(1 - D_x^{(i)})}{D_x^{(i)}} x_q^{1(i)} + \frac{-D_z^{(i)}(1 - D_z^{(i)})}{1 - D_z^{(i)}} z_{g\ q}^{1(i)} \end{aligned} \tag{B.47}$$

$$= (1 - D_x^{(i)})x_q^{1(i)} - D_z^{(i)} z_{g\ q}^{1(i)}$$

$$\nabla_{w_q^{d(2)}} J_d = \frac{1}{m} \sum_{i=1}^{m} \left[(1 - D_x^{(i)})x_q^{1(i)} - D_z^{(i)} z_{g\ q}^{1(i)} \right] \tag{B.48}$$

From hidden layer to input layer, compute $\nabla_{W^{d(1)}} J_d$,

$$\nabla_{W^{d(1)}} J_d = \{\nabla_{w_{j,q}^{d(1)}} J_d\}, j = 1, ..., p,\ q = 1, ..., Q$$

$$\nabla_{W^{d(1)}} J_d^{(i)} = \{\nabla_{w_{j,q}^{d(1)}} J_d^{(i)}\}, j = 1, ..., p,\ q = 1, ..., Q$$

$$\begin{aligned}
\nabla_{w_{j,q}^{d(1)}} J_d^{(i)} &= \frac{\partial J_d^{(i)}}{\partial w_{j,q}^{d(1)}} = \frac{\partial J_d^{(i)}}{\partial x_q^{1(i)}} \frac{\partial x_q^{1(i)}}{\partial w_{j,q}^{d(1)}} + \frac{\partial J_d^{(i)}}{\partial z_{g\ q}^{1(i)}} \frac{\partial z_{g\ q}^{1(i)}}{\partial w_{j,q}^{d(1)}} \\
&= \frac{D_x^{(i)}(1 - D_x^{(i)}) w_q^{d(2)}}{D_x^{(i)}} \frac{\partial x_q^{1(i)}}{\partial w_{j,q}^{d(1)}} + \frac{-D_z^{(i)}(1 - D_z^{(i)}) w_q^{d(2)}}{1 - D_z^{(i)}} \frac{\partial z_{g\ q}^{1(i)}}{\partial w_{j,q}^{d(1)}} \\
&= \frac{D_x^{(i)}(1 - D_x^{(i)}) w_q^{d(2)}}{D_x^{(i)}} x_q^{1(i)}(1 - x_q^{1(i)}) x_j^{(i)} + \frac{-D_z^{(i)}(1 - D_z^{(i)}) w_q^{d(2)}}{1 - D_z^{(i)}} z_{g\ q}^{1(i)}(1 - z_{g\ q}^{1(i)}) z_{g\ j}^{(i)} \\
&= (1 - D_x^{(i)}) w_q^{d(2)} x_q^{1(i)}(1 - x_q^{1(i)}) x_j^{(i)} - D_z^{(i)} w_q^{d(2)} z_{g\ q}^{1(i)}(1 - z_{g\ q}^{1(i)}) z_{g\ j}^{(i)}
\end{aligned} \tag{B.49}$$

$$\nabla_{w_{j,q}^{d(1)}} J_d = \frac{1}{m} \sum_{i=1}^{m} \left[(1 - D_x^{(i)}) w_q^{d(2)} x_q^{1(i)}(1 - x_q^{1(i)}) x_j^{(i)} - D_z^{(i)} w_q^{d(2)} z_{g\ q}^{1(i)}(1 - z_{g\ q}^{1(i)}) z_{g\ j}^{(i)} \right] \tag{B.50}$$

Update θ^d,

$$W^{d(1)} = W^{d(1)} + \alpha \nabla_{W^{d(1)}} J_d \tag{B.51}$$

$$W^{d(2)} = W^{d(2)} + \alpha \nabla_{W^{d(2)}} J_d \tag{B.52}$$

3. In discriminator:

Get information from discriminator again using newly updated θ^d,

$$Input: \ z_g^{(i)},\ i = 1, ..., m$$

$$Hidden: \ z_{g\ q}^{1(i)} = S(\sum_{j=1}^{p+1} \hat{z}_{g\ j}^{(i)} \bar{w}_{j,q}^{d(1)}), q = 1, ..., Q \tag{B.53}$$

$$Output: \ D_z^{(i)} = S(\sum_{q=1}^{Q+1} \hat{z}_{g\ q}^{1(i)} \bar{w}_q^{d(2)}). \tag{B.54}$$

4. In generator:

$$\begin{aligned} &Input\ layer:\ z^{(i)} = (z_1^{(i)}, ..., z_p^{(i)}),\ \hat{z}^{(i)} = (z^{(i)}\ 1) \\ &Output\ layer:\ z_g = S(\hat{z}\bar{W}^g) \\ &z_{g\ j}^{(i)} = S(\sum_{u=1}^{p+1} z_u^{(i)} w_{u,j}^g), j = 1, ..., p \end{aligned} \tag{B.55}$$

5. Update parameters in generator:

Loss function is,

$$\begin{aligned} J_g &= \frac{1}{m}\sum_{i=1}^{m} log(1 - D(G(z^{(i)}|\theta^g)\theta^d)) \\ &= \frac{1}{m}\sum_{i=1}^{m} log(1 - D(z_g^{(i)}|\theta^d)) \\ &= \frac{1}{m}\sum_{i=1}^{m} log(1 - D_z^{(i)}) \end{aligned} \tag{B.56}$$

Given information from discriminator D_z, z_g^1, $W^{d(2)}$ and $W^{d(1)}$, output of generator z_g and noise samples z, compute $\nabla_{W^g} J_g$.

$$\begin{aligned} \nabla_{W^g} J_g &= \nabla_{w_{u,j}^g} J_g,\ u = 1, ..., p,\ j = 1, ..., p \\ \nabla_{W^g} J_g^{(i)} &= \nabla_{w_{u,j}^g} J_g^{(i)},\ u = 1, ..., p,\ j = 1, ..., p \\ \nabla_{w_{u,j}^g} J_g^{(i)} &= \frac{\partial J_g^{(i)}}{\partial w_{u,j}^g} = \sum_{q=1}^{Q} \Big[\frac{\partial J_g^{(i)}}{\partial z_{g\ q}^{1(i)}} \frac{\partial z_{g\ q}^{1(i)}}{\partial z_{g\ j}^{(i)}}\Big] \frac{\partial z_{g\ j}^{(i)}}{\partial w_{u,j}^g} \\ &= \sum_{q=1}^{Q} [\frac{-D_z^{(i)}(1 - D_z^{(i)})}{1 - D_z^{(i)}} w_q^{d(2)} z_{g\ q}^{1(i)} (1 - z_{g\ q}^{1(i)}) w_{j,q}^{d(1)}] \frac{\partial z_{g\ j}^{(i)}}{\partial w_{u,j}^g} \\ &= \sum_{q=1}^{Q} [\frac{-D_z^{(i)}(1 - D_z^{(i)})}{1 - D_z^{(i)}} w_q^{d(2)} z_{g\ q}^{1(i)} (1 - z_{g\ q}^{1(i)}) w_{j,q}^{d(1)}] z_{g\ j}^{(i)} (1 - z_{g\ j}^{(i)}) z_u^{(i)} \\ &= -D_z^{(i)} \sum_{q=1}^{Q} [z_{g\ q}^{1(i)} (1 - z_{g\ q}^{1(i)}) w_q^{d(2)} w_{j,q}^{d(1)}] z_{g\ j}^{(i)} (1 - z_{g\ j}^{(i)}) z_u^{(i)} \end{aligned} \tag{B.57}$$

Update θ^g,

$$W^g = W^g - \beta \nabla_{W^g} J_g \tag{B.58}$$