A Mean-Field Optimal Control Formulation of GANs

by

Yirong Bian

A senior thesis submitted in partial fulfillment

of the requirements for the degree of

Bachelor's in Honors Mathematics

New York University Shanghai

May, 2022

Professor Mathieu Laurière

Acknowledgements

I would like to express my deep and sincere gratitude to my research supervisor, Professor Mathieu Laurière for giving me the opportunity to do research and providing invaluable guidance throughout this research. He is knowledgeable, enthusiastic and patient. He encouraged me when I was overwhelmed by the content and felt frustrated. Even with a 12 hour time difference between New York and Shanghai and such difficult lockdown situation, he always provided me with immediate, constructive and detailed help. It was my honor to carry out this research under his supervision and I am grateful that he gradually introduced me to such an interesting, integrated, fast-developing field of study.

I am thankful for all my mathematics professors for preparing me with sufficient and substantial knowledge to understand and research this topic. I would also like to thank my peers Yuchen Zhu, Lexie Zhu for listening to my presentation and giving useful feedback. Discussions with Winston Liang, Travis Luo, Jimmy Zhu, Wenbo Bao were very helpful for me to sort things out and proceed with confidence.

I am extremely grateful for my parents for their love, caring, and sacrifices for educating me and preparing me for the future. I really appreciate the emotional support from my friends. Special thanks go to Tony Chen and Evelyn Sheng. This thesis would have not been possible without countless cheer ups from you.

Abstract

Although deep learning is extending the frontier of data science with rapidly emerging applications, it is often hard to explain and interpret the neural networks. Recent works have shown that with continuous time idealization, residual neural networks can be viewed as a mean field optimal control problem. This thesis extends the theoretical formulation to generative adversarial network composed by two residual neural networks and explores the game between the generator and the discriminator. Additionally, with experiments on different distributions, we visualize the dynamical process and provide insights into the effect of each layer.

Contents

List of Figures

1 Introduction

Deep learning is driving advances in the machine learning field by implementing various kinds of neural networks inspired by the architecture of human brains. Generative models are a specific branch of deep learning, which is an unsupervised model that can be applied to artistic style transfer, image synthesis, image-to-image translation, etc. [1][2][3]. For example, a trained model can generate several credible images based on a text description [4]. Despite the empirical success of the algorithm of generative adversarial networks (GANs) [5], the neural networks are often viewed as a black box. In other words, it is hard to know what each layer or each neuron does. To increase the explain-ability, Weinan E et al. [6] has connected the modeling and analysis of the residual neural networks (resNet) learning problem [7] with the mean field control theory [8] in order to provide insights into the role of hidden layers within a rigorous mathematical framework. In this thesis, we will examine GANs composed by two resNet following the same framework to theorize, visualize and the effect of each layer, and hence better understand the black box. The thesis is organized as the following. In Chapter 2, we first introduce the idea of GANs and resNet. Then we begin with the classical optimal control theory and extend it to mean field dynamical systems. Especially, we state two fundamental theorems in optimal control: Hamilton–Jacobi–Bellman equation and Pontryagin Maximum Principle. In Chapter 3, we specify a mathematical formulation of the GAN with 2 resNet problem in terms of a mean field dynamical system. In Chapter 4, we conduct experiments on the learning of simple distributions and visualize the dynamical process of both the resNet generator and the resNet discriminator

to visualize and give insights into what each layer does. Finally, in Chapter 5, we present the conclusion and further discussion of the problem.

2 Background

This chapter will include the foundation to proceed to the next chapter, the mean field optimal control formulation of resNets GAN. We will introduce several important concepts including the optimal control formulation of resNet, the algorithm of GAN and the mean field optimal control theory.

2.1 GANs

In this thesis, we focus on the theoretical formulation of GANs and this section we introduce the algorithm of GANs, first proposed by Ian J. Goodfellow [5, Theorem 1]. The idea is that given a data set (e.g. a set of photos of cats), we want to train a model such that it can generate new data (new photos of cats) that is similar to the given data set.

The GAN framework contains two neural networks, generator G and discriminator D, with parameters θ^G, θ^D respectively. We denote the true data distribution by p_x, and define a prior noise distribution to be p_z. The discriminator's goal is to discriminate whether the input comes from true data or noise data while the generator on the contrary aims to cheat the discriminator with its generated data similar to the true ones. In other words, the generator G will transform $z \sim p_z$ to $G(z, \theta^G) \sim p_g$, where p_g is the generated distribution. We expect $p_g = p_x$ by finding the global

optimality of the two-player minimax game:

$$\min_{\theta^G} \max_{\theta^D} V(\theta^D, \theta^G) = \mathbb{E}_{x\sim p_x}[\log D(x)] + \mathbb{E}_{z\sim p_z}[\log(1 - D(G(z)))].$$

In the next section, we will introduce a mean field control formulation of GAN. Before that, in the rest of this section, we recall how the optimization of a single neural network can be phrased as a mean field control problem.

2.2 From Residual Neural Networks to Optimal Control

In this section, we follow the formulation of Weinan E et al. [6], and present the optimal control formulation for residual neural networks. Note that the following formulation requires labelled data, which is slightly different from the unsupervised GAN this thesis focuses on, we will address the difference in the next chapter.

We assume the data is drawn from a joint probability distribution $(x_0, y_0) \sim \mu$, where $x_0 \in \mathbb{R}^d, y_0 \in \mathbb{R}^l$. For a N_T layered resNet, we have the feed-forward propagation difference equation:

$$x_{t+1} = x_t + f(x_t, \theta_t), \quad t = 0, 1, 2, ..., N_T - 1$$

where $\theta_t \in \Theta \subset \mathbb{R}^m$ is the control from the control set Θ and $f : \mathbb{R}^d \times \mathbb{R}^m \mapsto \mathbb{R}^d$ is the controlled feed-forward rule. To study it in the context of optimal control theory, we adopt a continuous-time idealization, which changes the above difference equation to a differential equation:

$$\dot{x}_t = f(x_t, \theta_t), \quad t \in [0, T]$$

Then, x_T is the output of this algorithm and with the terminal loss function $\Phi : \mathbb{R}^d \times \mathbb{R}^l \mapsto \mathbb{R}$, the running cost function (regularizer) $L : \mathbb{R}^d \times \Theta \mapsto \mathbb{R}$. By training the parameters (controls)

$\boldsymbol{\theta} = \{\theta_t\}_{t=0}^{t=T}$, we want to minimize the following loss :

$$J(\boldsymbol{\theta}) = \Phi(x_T, y_0) + \int_0^T L(x_t, \theta_t)\, dt,\ \ \boldsymbol{\theta} \in L^\infty([0,T], \Theta)$$

Let us remind that (x_0, y_0) comes from a probability distribution, which adds a mean field layer to the original optimal control problem. Hence, the deep learning problem to minimize the population risk can be structured as to minimize the expectation loss:

$$J(\boldsymbol{\theta}) = \mathbb{E}_{(x_0,y_0)\sim\mu}\left[\Phi(x_T, y_0) + \int_0^T L(x_t, \theta_t)\, dt\right],\ \ \boldsymbol{\theta} \in L^\infty([0,T], \Theta) \tag{2.1}$$

Note that the mean field emphasizes that the control itself is deterministic and only depends on the distribution instead of some individual data from the distribution. Also, θ_t is an open-loop control which means that it only depends on t instead of a feedback control that depends on both t and x_t. In other words, the parameters in each layer are fixed numbers.

2.3 Optimal Control Theory

In order to solve for the above optimal control problem and hence, develop a valid model, we need theorems that construct and characterize the control $\boldsymbol{\Theta}$. In this section, we will first introduce the Pontryagin Maximum Principle (PMP) and Hamilton–Jacobi–Bellman (HJB) equation in the classical optimal control theory [8]. Then, we will state their mean field form in the above resNet case.

2.3.1 Classical Optimal Control Theory

In this subsection, we study the most generic case in the optimal control theory. In specific, it is a fixed time, free end point problem:

Problem 1. *Given $A \subset \mathbb{R}^m$ and $f : \mathbb{R}^n \times A \mapsto \mathbb{R}^n, x^0 \in \mathbb{R}^n$. We denote the set of admissible controls by*

$$\mathcal{A} = \{\alpha(\cdot) : [0, \infty) \mapsto A \mid \alpha(\cdot) \text{ is measurable}\}.$$

Then, given $\alpha(\cdot)$ we can solve for the evolution of x controlled by:

$$\begin{cases} \dot{x}(t) = f(x(t), \alpha(t)) \ (t \geq 0) \\ x(t) = x^0 \end{cases}$$

We also have the payoff functional

$$P[\alpha(\cdot)] = \int_0^T L(x(t), \alpha(t))dt + \Phi(x(T)),$$

where $T > 0$ is the terminal time, $L : \mathbb{R}^n \times A \mapsto \mathbb{R}$ is the given running payoff, $\Phi : \mathbb{R}^n \mapsto \mathbb{R}$ is the given terminal payoff. The problem is to find a control $\alpha^*(\cdot)$, such that

$$P[\alpha^*(\cdot)] = \min_{\alpha(\cdot) \in \mathcal{A}} P[\alpha(\cdot)]$$

Definition 2.1. The control theory Hamiltonian is the function

$$H(x, p, a) := f(x, a) \cdot p + L(x, a) \quad x, p \in \mathbb{R}^n, a \in A$$

Theorem 2.2 (Pontryagin Maximum Principle)**.** *Assume $\alpha^*(\cdot)$ is optimal for **Problem 1**, and $x^*(\cdot)$*

is the corresponding trajectory. Then there exists a function $p^*(\cdot) : [0, T] \mapsto \mathbb{R}^n$ *such that*

$$\begin{aligned}
\dot{x}^*(t) &= \nabla_p H(x^*(t), p^*(t), \alpha^*(t)) = f(x^*(t), \alpha^*(t)), \\
\dot{p}^*(t) &= -\nabla_x H(x^*(t), p^*(t), \alpha^*(t)), \\
x^*(0) &= x^0 \\
p^*(T) &= \nabla\Phi(x^*(T)), \\
H(x^*(t), p^*(t), \alpha^*(t)) &= \min_{a \in A} H(x^*(t), p^*(t), a), \quad \forall 0 \le t \le T
\end{aligned}$$

In addition, the mapping

$$t \mapsto H(x^*(t), p^*(t), \alpha^*(t)) \text{ is constant}$$

The general form of the theorem is proved in the book by Fleming and Rishel [9]. Note that the maximum principle provides a rule to design the optimal control backwards by solving the costate $p^*(t)$ at the same time. However, the computation depends on a certain trajectory of x and hence it is a local solution. In other words, when we start with a different initial value x^0, we will have to do the computation all over again to find the optimal control specific to this initial value. It motivates us to study the value function to solve the problem globally. Although the value function is often not explicit, the following HJB equation gives a good characterization of it.

Definition 2.3. For $x \in \mathbb{R}^n, 0 \le t \le T$, define the value function $v(x, t)$ to be the smallest payoff possible if we start at $x \in \mathbb{R}^n$, at time t. In other words,

$$v(x, t) := \inf_{\alpha(\cdot) \in \mathcal{A}} P[\alpha(\cdot)]$$

Theorem 2.4 (Hamiltonian-Jacobi-Bellman Equation)**.** *Assume that the value function* v *is a* C^1

function of the variables (x, t)*. Then* v *solves the nonlinear partial differential equation,*

$$v_t(x,t) + \min_{a \in A} \{f(x,a) \cdot \nabla_x v(x,t) + L(x,a)\} = 0 \quad (x \in \mathbb{R}^n, 0 \le t < T)$$

with the terminal condition

$$v(x,T) = \Phi(x)$$

2.3.2 Mean Field Optimal Control Theory

In this subsection, we introduce a stochastic feature to the above model and hence derive the PMP and HJB in the problem of training the resNet with the equation (2.1). Namely, we will find the optimal control not for a single input value but for a value comes from a certain distribution. Paper by Weinan E et al. [6] has a complete proof on the following two theorems.

Theorem 2.5 (Mean Field PMP)**.** *Assume the function* f *is bounded;* f, L *are continuous in* θ*; and* f, L, Φ *are continuously differentiable with respect to* x*. Assume the distribution* μ *has bounded support in* $\mathbb{R}^d \times \mathbb{R}^l$*. If* $\boldsymbol{\theta}^* \in L^\infty([0,T], \Theta)$ *is a solution of* ***Problem 1****, then there exists absolutely continuous stochastic processes* x^*, p^* *such that*

$$\begin{aligned}
&\dot{x}_t^* = f\left(x_t^*, \theta_t^*\right), && x_t^* = x_0 \\
&\dot{p}_t^* = -\nabla_x H\left(x_t^*, p_t^*, \theta_t^*\right), && p_T^* = \nabla_x \Phi\left(x_T^*, y_0\right) \\
&\mathbb{E}_\mu H\left(x_t^*, p_t^*, \theta_t^*\right) = \min_{\theta \in \Theta} \mathbb{E}_\mu H\left(x_t^*, p_t^*, \theta\right), && \text{a.e. } t \in [0,T]
\end{aligned}$$

where the Hamiltonian function $H : \mathbb{R}^d \times \mathbb{R}^d \times \Theta \to \mathbb{R}$ *is given by*

$$H(x,p,\theta) = f(x,\theta) \cdot p + L(x,\theta)$$

Definition 2.6.

$$v^*(t,\mu) = \inf_{\boldsymbol{\theta} \in L^\infty([0,T],\Theta)} J(t,\mu,\boldsymbol{\theta})$$

Definition 2.7. Let $\bar{f}, \bar{L}, \bar{\Phi}$ be functions that extend to the $\mathbb{R}^{d+l}$ space defined by:

$$\begin{aligned} \bar{f}(w,\theta) &= (f(x,\theta), y) \\ \bar{L}(w,\theta) &= L(x,\theta) \qquad \text{for } w = (x,y) \in \mathbb{R}^{d+l} \\ \bar{\Phi}(w) &= \Phi(x,y) \end{aligned}$$

Theorem 2.8 (Mean Field HJB)**.** *Let v be the solution to*

$$\begin{cases} \frac{\partial v(t,\mu)}{\partial t} + \inf_{\theta_t \in \Theta} \int_{\mathbb{R}^{d+l}} (\partial_\mu v(t,\mu)(w) \cdot \bar{f}(w,\theta_t) + \bar{L}(w,\theta_t)) \mu(dw) = 0 \quad \text{on } [0,T) \times \mathcal{P}_2\left(\mathbb{R}^{d+l}\right) \\ v(T,\mu) = \int_{\mathbb{R}^{d+l}} \bar{\Phi}(w) \mu(dw) = \mathbb{E}_{(x,y)\sim\mu}[\Phi(x,y)] \end{cases}$$

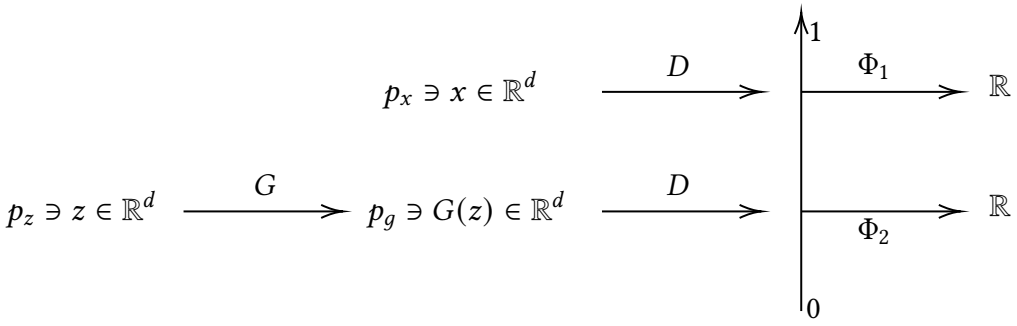
and there exists $\theta^* : (t,\mu) \mapsto \theta$, attaining the infimum, then $v(t,\mu) = v^*(t,\mu)$, and θ^* is the optimal feedback control.

Note that the above two theorems is analogous to the ones in the optimal control theory but the only change is that we evaluate the cost function in the sense of expectation over a distribution and try to minimize the expected cost.

3 | Mathematical Formulation of the Problem

In this chapter, we will present the optimal control formulation of the GAN deep learning algorithm composed by two residual neural networks (x). There is a resNet generator G which takes input from a fake distribution p_z and transforms it to another probability distribution p_g. The resNet discriminator D aims to discriminate whether the data comes from the true distribution p_x or the generated distribution. Given enough capacity and training time, we expect the generated distribution p_g to approximate the true data distribution p_x. Following the formulation of the Background 2, we idealize the feed-forward dynamics to be a differential equation instead of the difference equation and study it in continuous time. The following graph shows a general idea of the process of points in the algorithm.

Figure 3.1: Process of Points in the Algorithm



Definition 3.1. We stack the true data sample x and the fake one z as a single input X drawn from the joint distribution μ.

$$X_0 := (x_0, z_0)^T \sim \mu = (p_x, p_z)^T, \quad X_t := (x_t, z_t)$$

where $x, z \in \mathbb{R}^d, X_t \in \mathbb{R}^{2d}$, p_x is the distribution of true data and p_z is the distribution of fake input.

Definition 3.2. Define the loss function as

$$\Phi(X) := \Phi(x, z) = \Phi_1(h(x)) + \Phi_2(h(z))$$

where $h : \mathbb{R}^d \mapsto (0, 1)$ is a preset reduction function that reduces the dimension of data on $\mathbb{R}^d$ (e.g. a sigmoid function)

$\Phi_1 : (0, 1) \mapsto \mathbb{R}$ is an increasing function and in the typical setting $\Phi_1(x) = \log(x)$

$\Phi_2 : (0, 1) \mapsto \mathbb{R}$ is an decreasing function and in the typical setting $\Phi_2(h(x)) = \log(1 - x)$

Assume g, f to be the feed-forward rule for the generator of layer T_1 and the discriminator of layer T_2 respectively. Then

$$\dot{z}_t = g(z_t, \theta_t^G) \quad 0 < t < T_1, \quad g : \mathbb{R}^d \times \Theta^G \mapsto \mathbb{R}^d$$
$$\dot{x}_t = f(x_t, \theta_t^G) \quad T_1 \leq t \leq T_2, \quad f : \mathbb{R}^d \times \Theta^D \mapsto \mathbb{R}^d$$

We concatenate the parameters in each resNet $\theta_t^G \in \Theta^G, \theta_t^D \in \Theta^D$ to define

$$\theta_t := (\theta_t^G, \theta_t^D)^T \in \Theta.$$

Then we can write the dynamics of the system as:

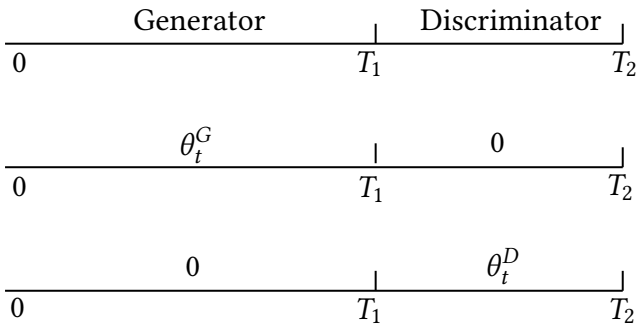
$$\begin{cases} \dot{X}_t = F_t(X_t, \theta_t) \\ X(0) = X_0 = (x_0, z_0)^T \end{cases}$$

$$\text{where } F_t(X_t, \theta_t) = \begin{cases} \left(0, g(z_t, \theta_t^G)\right)^T & 0 \le t < T_1 \\ \left(f(x_t, \theta_t^D), f(z_t, \theta_t^D)\right)^T & T_1 \le t \le T_2 \end{cases}$$

Problem. *Given the terminal loss function Φ, the running cost / regularization function $L : \mathbb{R}^{2d} \times \Theta \mapsto \mathbb{R}$ and the distribution μ, if we write the terminal time as $T_1 + T_2 = T$ we want to find the optimal for the following minimax problem:*

$$J(t, \mu, \theta) = \min_{\theta \in L^\infty([0,T_1)\times\Theta)} \max_{\theta \in L^\infty([T_1,T_2]\times\Theta)} \mathbb{E}_{X\sim\mu}\left[\Phi(X_T) + \int_t^T L(X_s, \theta_s) ds\right]$$

Figure 3.2: Formulation



The Figure 3.2 shows clearly how θ_t is used to compute the outputs. The idea is to keep the true data constant when processed through the generator and fix the change to the fake data when processed through the discriminator. Note that there are two ways of formulating this: one way is to directly set the evolution of the upper part in $0 \le t < T_1$ to be 0 (as specified in the

above differential equations) but the lack of uniformity may cause difficulties in further solving the problem; The other way requires an assumption on the generator feed-forward function. If there exists a constant $c(= 0$ in the below graph) such that $g(z_t, c) = 0,\ \forall z_t \in \mathbb{R}^d$. Then instead of programming the dynamics, we can restrict the control space to enforce the feed-forward rule being 0 on $[0, T_1)$ for true data points.

4 Experiments and Visualization

In this chapter, we will implement the GAN with two resNets algorithm on known true data distributions and try to visualize the evolution of data points and interpret the algorithm.

Let us first define the resNet that will be used in the following examples.

Algorithm 1 Forward Propagation Generator with N_g as the number of layers

Input: $z \in \mathbb{R}^d$ from prior input noise distribution
Output: $G(z) \in \mathbb{R}^d$ as generated point

$dt \leftarrow 0.1$
$X \leftarrow z$
for $i \leftarrow 1$ to N_g **do**
    $X \leftarrow X + dt * Dense_{\tanh}(X; \theta_i^G)$
end for
return X

$Dense_{\tanh} : \mathbb{R}^d \times \Theta \mapsto \mathbb{R}^d$ is a simple neural layer, with tanh as the activation function and θ_i^G as parameters.

Algorithm 2 Forward Propagation Discriminator with N_d as the number of layers

Input: $x \in \mathbb{R}^d$

Output: $D(x) \in [0, 1]$

$dt \leftarrow 0.1$

$X \leftarrow x$

for $i \leftarrow 1$ to N_d **do**

$X \leftarrow X + dt * Dense_{\tanh}(X; \theta_i^D)$

end for

$X \leftarrow Dense_{\sigma}(X)$

return X

$Dense_{\tanh} : \mathbb{R}^d \times \Theta \mapsto \mathbb{R}^d$ is a simple neural layer, with tanh as the activation function and θ_i^G as parameters. $Dense_{\sigma} : \mathbb{R}^d \mapsto [0, 1]$, defined as $Dense_{\sigma}(X) = \sigma(w^T X + b)$, is a fixed simple neural layer with the sigmoid activation function with parameters w, b.

Algorithm 3 Minibatch stochastic gradient descent training of generative adversarial nets.

Denote $\theta_g = (\theta_1^G, ..., \theta_{N_g}^G)$; $\theta_d = (\theta_1^D, ..., \theta_{N_d}^D)$

Parameters: Learning rate of generator and discriminator: α_g, α_d

for number of iteration steps **do**

- Sample minibatch of m noise samples $\{z_1, ..., z_m\}$ from prior input noise distribution p_z
- Sample minibatch of m true examples $\{x_1, ..., x_m\}$ from true data distribution p_x

Update the discriminator by ascending its stochastic gradient:

$$\theta_g \leftarrow \theta_g + \alpha_g \nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} [\log D(x_i) + \log(1 - D(G(z_i)))]$$

Update the generator by descending its stochastic gradient:

$$\theta_d \leftarrow \theta_d - \alpha_d \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [\log(1 - D(G(z_i)))]$$

end for

4.1 Uniform Distribution Experiment

We begin with learning the true distribution of a uniform $[0, 1] \times [0, 0.5]$ distribution and we fix the prior input noise distribution as a uniform $[0, 1] \times [0.5, 1]$ distribution, shown as the following figure.

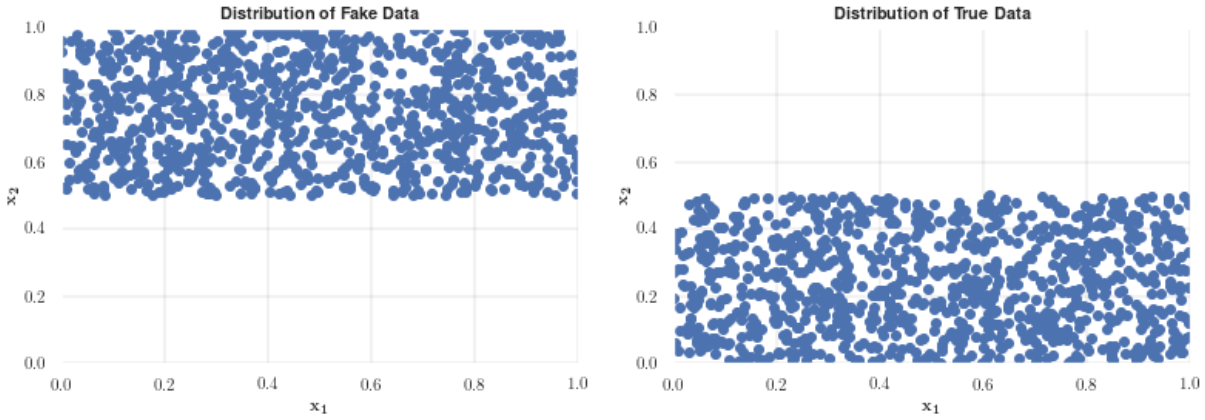


Figure 4.1: Distribution of true and fake data

4.1.1 Fix Generator; Train the Discriminator

We first fix the generator and train the discriminator and after the training, we will try visualizing the process by feeding the discriminator points from both true and the fake distribution. The generator we choose is simply the identity one, meaning $G(z) = z$. The discriminator is a resNet according to Algorithm 2 with $N_d = 20$ layers. We set the learning rates at $\alpha_g = 0, \alpha_d = 0.001$. We use a mini-batch stochastic gradient descent with batch size $m = 100$ and 5000 iterations. Figure 4.2 shows the loss for both the generator and discriminator.

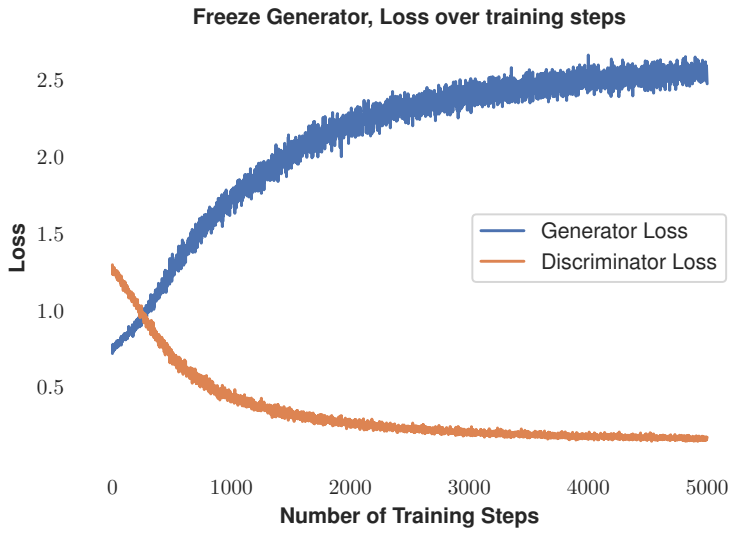


Figure 4.2: Loss over iteration

We observe that the discriminator loss decreases as generator loss increases, which is consistent to the fact that they are in a minimax game. The improvement of discriminator is at the cost of the higher generator loss. Although it is a simple training experiment, we can visualize the discriminator of how it discriminates data from true distribution and fake distribution through forward propagation, in other words, the evolution of points. Figure 4.3 shows the evolution of true/fake data points in the process of the discriminator. The same plot also shows whether the capacity of the discriminator resNet is enough. Note that in the forward propagation algorithm, the maximum change in each coordinate of input x is ± 0.1 because the layer is activated by $\tanh(x) \in (-1, 1)$. Therefore in the two dimensional case with a 20-layer resNet, the maximum distance x can move is $2\sqrt{2}$. We will see if the points can be moved in the correct regions within such distance.

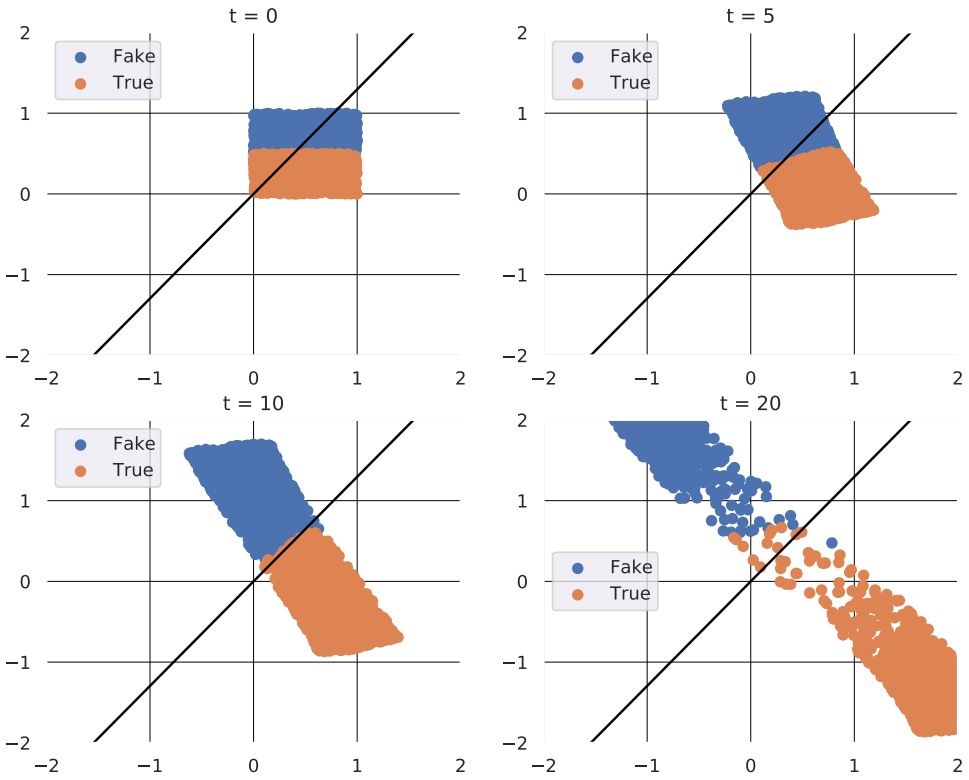


Figure 4.3: Evolution of Points through Discriminator

The black line is defined as $w^T x + b = 0$ according to the Algorithm 2, which is the decision boundary fixed by the last layer of the discriminator. We can see that as the process goes on, points are moved further away from the line accordingly. It shows how the control "drags" the points to their corresponding regions in order to minimize the terminal cost. Since moving distance of the points to the corresponding region is much less than $20\sqrt{2}$, we can determine that the discriminator resNet has enough capacity in this case.

4.1.2 Fix Discriminator; Train Generator

We now fix the discriminator with decision boundary $x_2 = 0.5$, meaning that points with $x_2 < 0.5$ will be classified as points from true distribution. We build the generator according to Algorithm 1 with $N_g = 20$. We will use a learning rate $\alpha_g = 0.001, \alpha_d = 0$ and a mini-batch stochastic

gradient descent with batch size $m = 100$ and 5000 iterations. Figure 4.4, 4.5 shows the loss over the iteration steps and evolution of points under the control of generator.

Figure 4.4: Loss over iteration

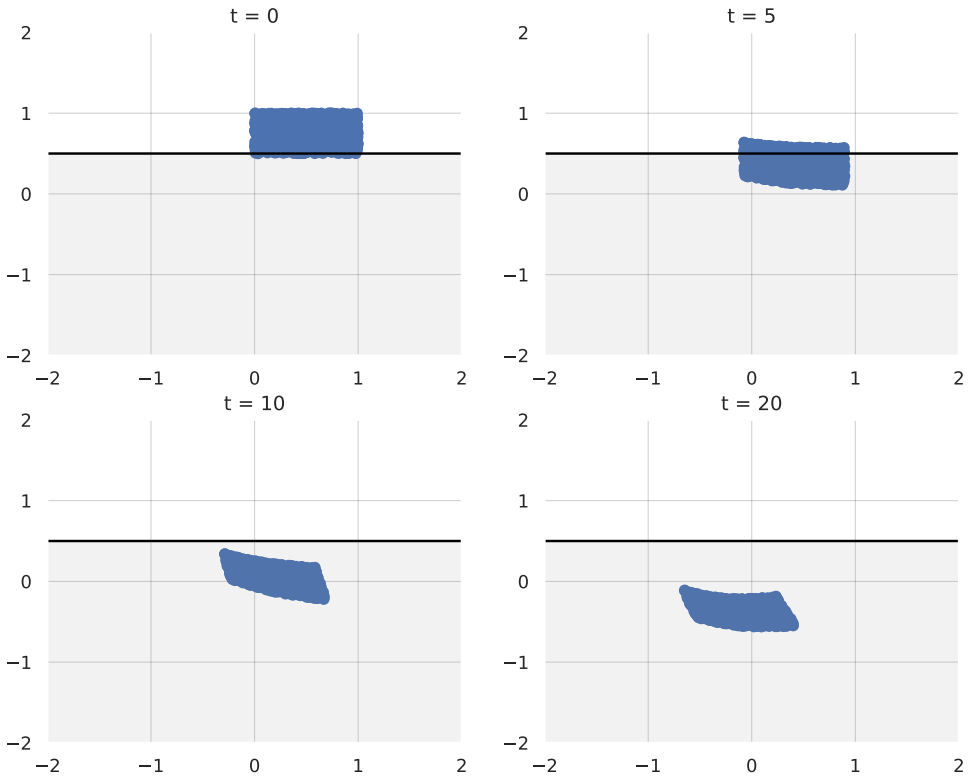


Figure 4.5: Evolution of Points through Generator

Similarly, we can observe that the generator loss reduces at the cost of discriminator loss, illustrating the minimax game structure between the two. Moreover, in the evolution figure, the grey-shaded area is the target region and we can see the generator drags the points deeper and deeper into the target region to minimize the generator loss.

4.1.3 GAN: Train the Generator and Discriminator at the same time

Now, we have all the trivial cases done. Let's train the generator and discriminator together at the same time according to Algorithm 3. After training, we expect to have a good generator which can transform the noise inputs from the upper half (i.e. $[0, 1] \times [0.5, 1]$) to points from the true distribution, the lower half (i.e. $[0, 1] \times [0, 0.5]$). Moreover, in order to have a good generator, we must have a good discriminator beforehand to learn the rules of the true distribution and hence,

"guide" the generator. In the experiment, I find the generator learns quicker than the discriminator with the learning rate pair $(\alpha_g, \alpha_d) = (0.001, 0.001)$, meaning the minimax game is dominated by the generator. However, as the generator becomes too good before discriminator learns the features of true distribution, the algorithm requires more iterations to reach an equilibrium. In the following plots, I use $(\alpha_g, \alpha_d) = (0.0001, 0.001)$. Additionally, comparing to trivial cases in the previous subsections, the real game dynamic between the generator and the discriminator complicates the training process and hence I use 10000 iterations. I will keep $N_g = N_d = 20$ the same as above because it has been validated that they have enough capacity.

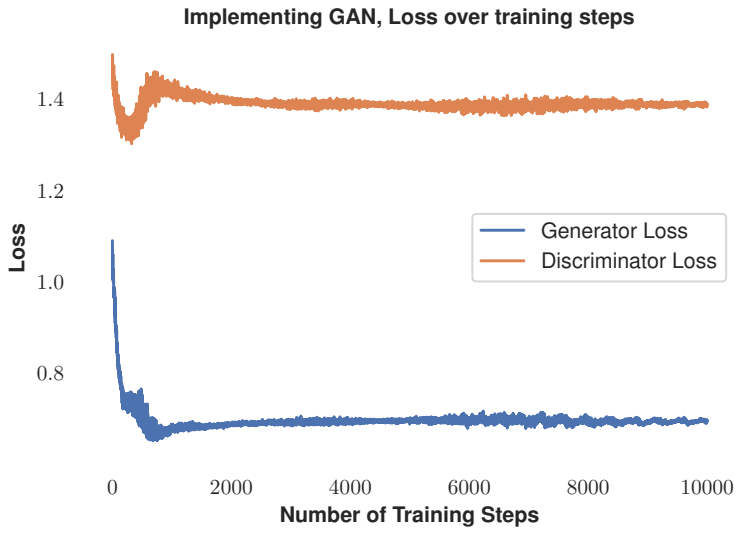


Figure 4.6: Loss over iteration

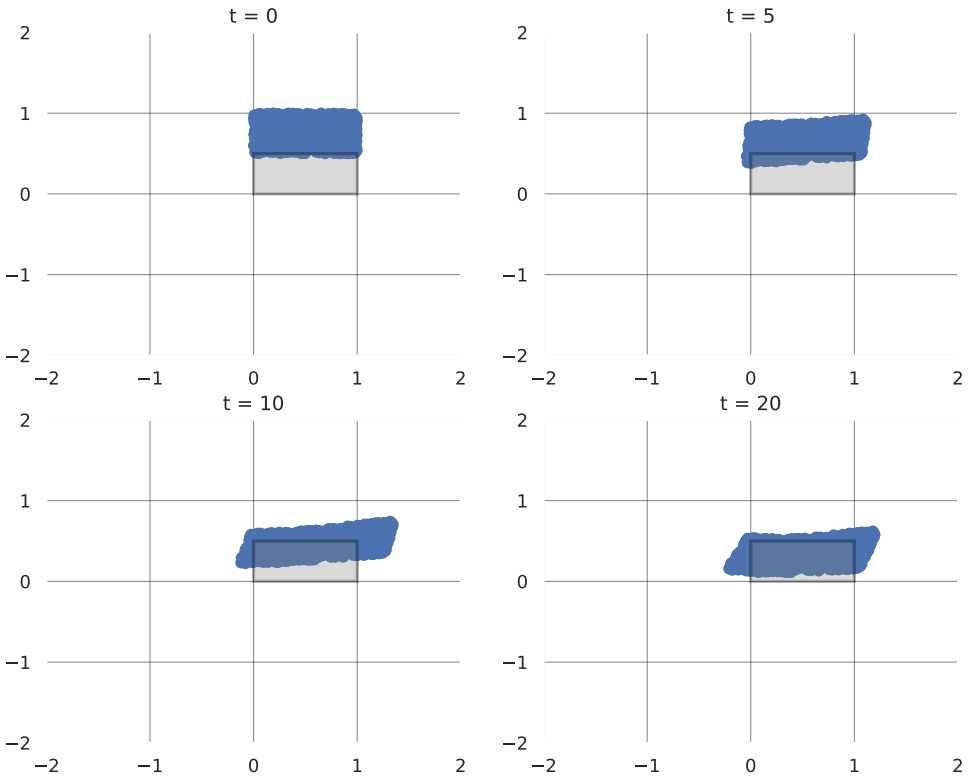


Figure 4.7: Evolution of Points through Generator

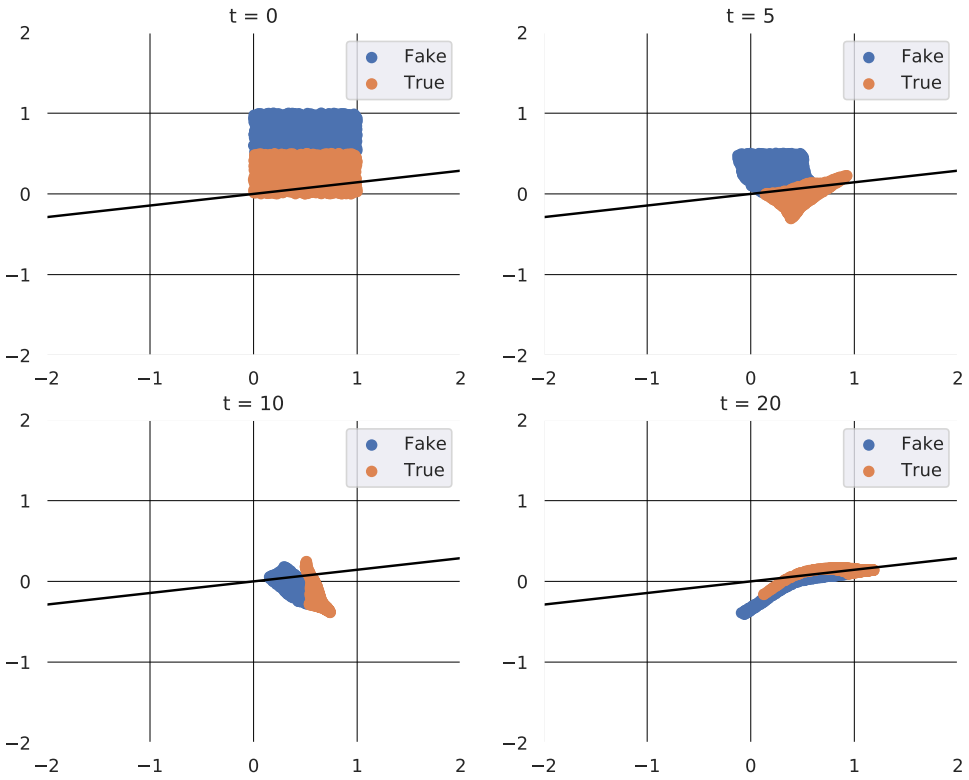


Figure 4.8: Evolution of Points through Discriminator

In Figure 4.7, the shaded area is the target region. We can see that a trained generator will move the points from the fake distribution in a way to imitate the true distribution in the end. However, we can see that some points are still out of the true distribution box. It means the generator is still not perfect after 10000 iterations and the two players have not reached a global optimality. In Figure 4.8, we observe that when $t = 20$, data points from both the true distribution and fake distribution clustered along the decision boundary. It illustrates that the generator is good enough to cheat the discriminator so that it is not able to decide whether the point is from the true distribution and fake distribution. Overall, we can conclude that given the discriminator, the generator is clever enough to cheat it while the discriminator has to learn more about the true distribution to guide the generator to a better result. In Figure 4.6, because of the minimax game structure between the generator and the discriminator, it is difficult to visualize the improvement of the algorithm with only the plot of the loss over iterations. It is likely that the stable loss in

later training steps means the algorithm is trapped in a local optimality while it can also mean both the generator and the discriminator are learning. We will provide more details in the next example.

4.2 Normal Distribution Experiment

In this section, we will conduct the same experiment on a different distribution: normal distribution. Let the true distribution be $(x_1, x_2)^T \sim \mathcal{N}\left((0,0)^T, \Sigma\right)$ and the fake distribution be $(z_1, z_2)^T \sim \mathcal{N}\left((1,1)^T, \Sigma\right)$, where $\Sigma = \begin{bmatrix} 0.05 & 0 \\ 0 & 0.05 \end{bmatrix}$. The discriminator is more difficult to train compared to the previous experiment because the data points from fake distribution and true distribution overlap. It means that even with a primitive generator (e.g. the identity), given a data point, the discriminator can not easily determine which distribution it comes from.

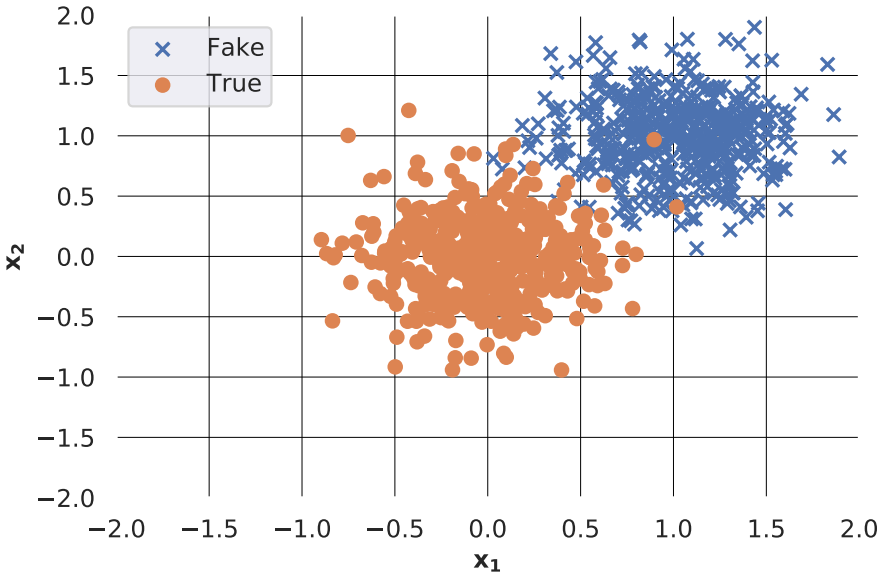


Figure 4.9: Gaussian Distribution

Figure 4.9 shows samples from the true and fake distribution. In this experiment, we will omit the preliminary parts similar to section 4.1.1, 4.1.2 and we directly train according to the 3

algorithm.

As mentioned in the previous section, it is difficult to directly visualize the success of the algorithm by looking at the loss over iterations. Therefore, we will need a new measure of success. Since the true distribution is known, we will define the generated likelihood from a fixed noise input $z_1, z_2, ..., z_n$ as:

$$\mathcal{L}(\theta^G) = \prod_{i=1}^{n} f\left(G_{\theta^G}(z_i)\right), \ f \text{ is the probability density function of true distribution}$$

In this case,

$$f(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^2 \cdot \det(\Sigma)}} e^{-\frac{1}{2}x^T\Sigma^{-1}x}, \quad \Sigma \text{ is defined above}$$

We can further simplify in our case that to maximize the likelihood is equivalent to minimize the following

$$l(\theta^G) = \sum_{i=1}^{n} G(z_i)^T\Sigma^{-1}G(z_i) \propto \sum_{i=1}^{n} ||G(z_i)||^2,$$

We will call this number the negative log-generated likelihood and we will use this to measure the success of the algorithm. Furthermore, to simplify, we will fix a group of test prior input noise points at the beginning and compute the number based on the points generated by those points. Ideally, we will see this number decrease which indicates that the probability that such group of generated points are sampled from the true distribution increases, meaning the generator is improving. Due to the computational cost, we will take $n = 10$ samples from noise input and compute the log-generated likelihood. We apply the same logic to deduce the capacity of the generator: We find that in this case the distance points from fake distribution have to travel is $(1, 1) \rightarrow (0, 0) = \sqrt{2}$ and we a 20-layered resNet has the maximum capacity to move points for a distance of $2\sqrt{2}$. Hence the capacity of $N_g = 20$ is reasonable. Similar to the case in uniform distribution, when training the GAN, we implement the learning rate $\alpha_g = 0.0001, \alpha_d = 0.001$ to avoid the dominance of the generator. We will set $N_d = 20$ and run the iterative algorithm 3

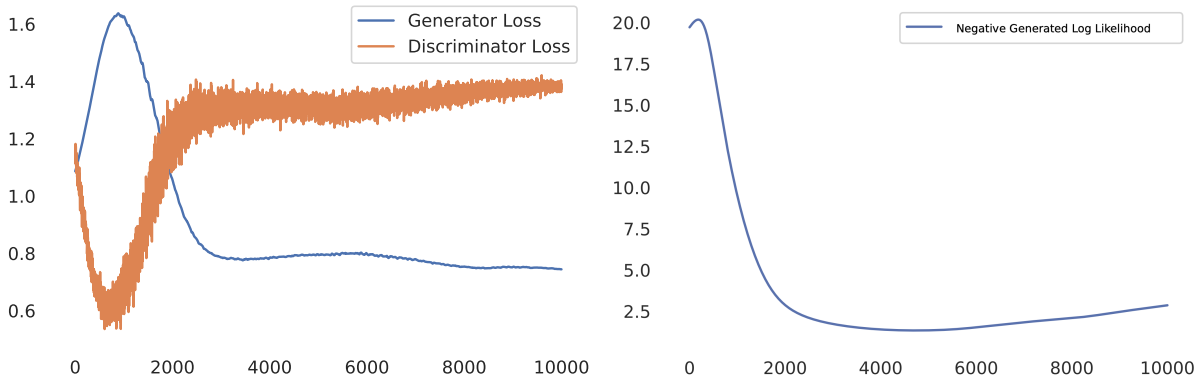


Figure 4.10: Loss, negative log-generated likelihood over iteration

for 10000 times. We can see from Figure 4.10 that during the first 2000 iterations, the generator loss and discriminator loss changes significantly and accordingly, the log generated likelihood decreases. It means that our algorithm is working. Moreover, we observe that the discriminator loss decreases at the beginning at the cost of the generator loss. We prefer this structure instead of the other way around because without an accurate discriminator, the generator will definitely fail in imitating the true distribution. Also, it validates that our learning rate pair is wisely chosen. After about 2000 iterations, we find the losses are relatively stable and the log generated likelihood remains almost the same. It suggests that after 2000 training steps, the algorithm ceases to learn and is close to a local optimality. Figure 4.11 shows the result of generated distribution. We can see that the generated distribution and true distribution are very similar. Figure 4.12 shows how points from two distributions evolves under the control of the discriminator. It is worth noting that in this case, the discriminator outputs all the points centered along the decision boundary, indicating that the generator is well trained. Figure 4.13 shows how points from noise prior distribution evolve under the control of the discriminator.

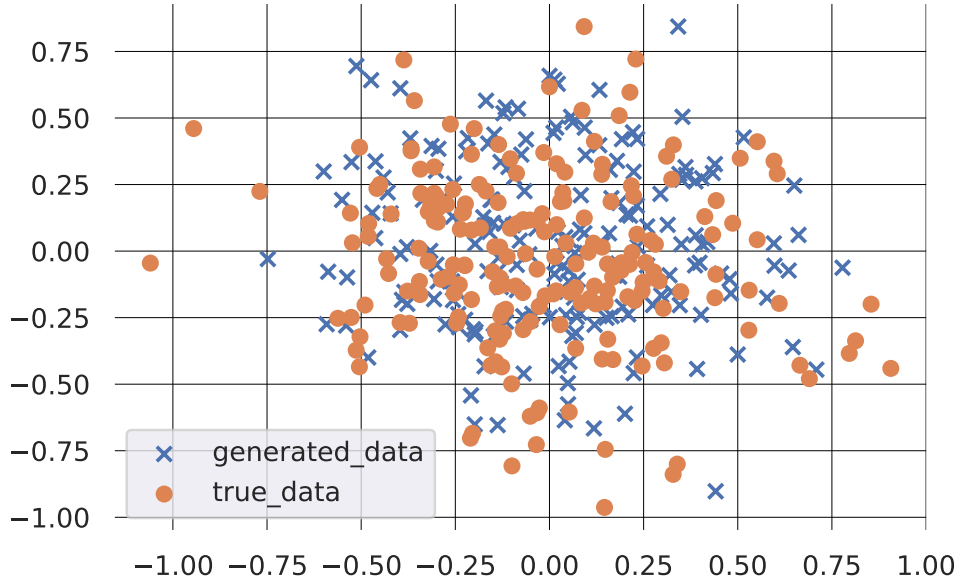


Figure 4.11: Generated Data vs. True Data

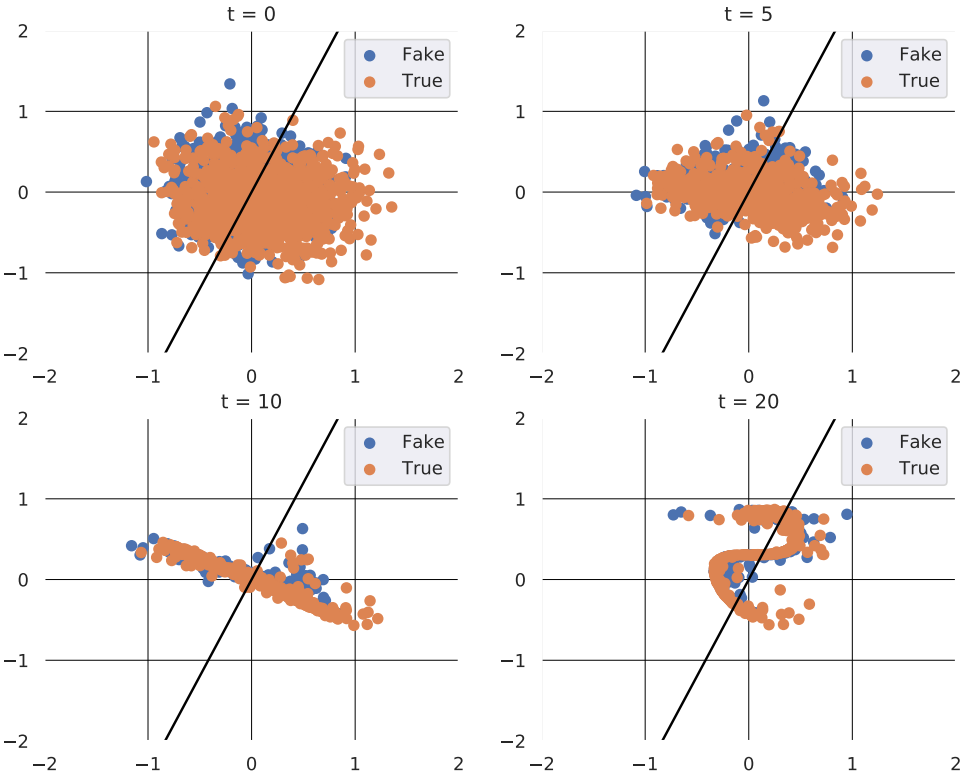


Figure 4.12: Evolution of Points through Discriminator

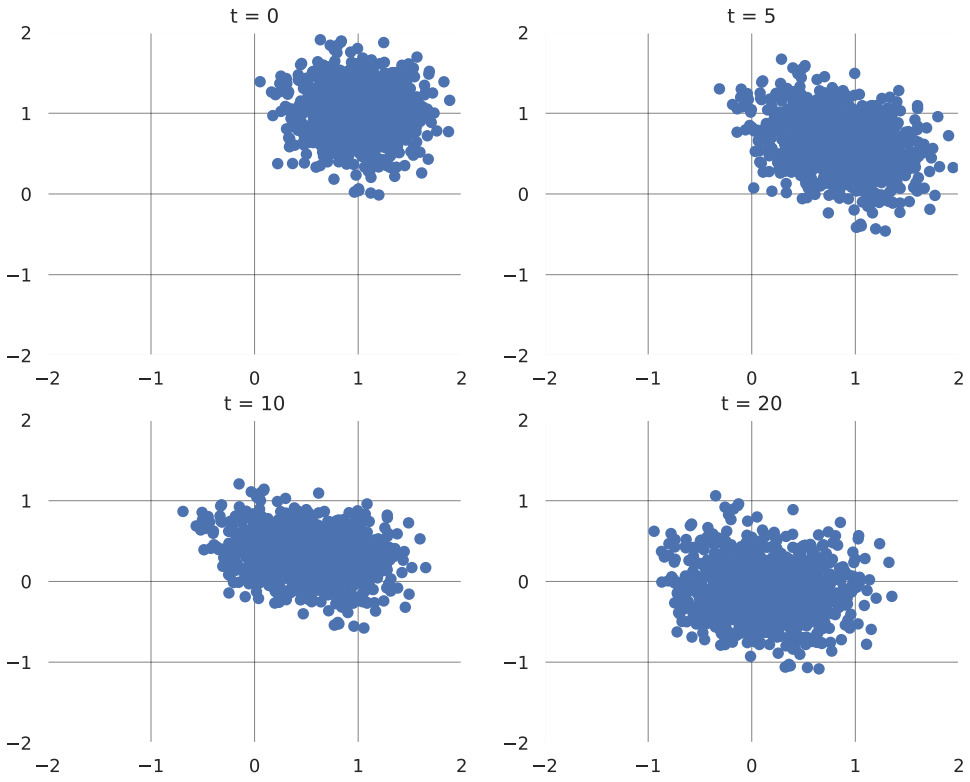


Figure 4.13: Evolution of Points through Generator

5 Conclusion and Discussions

5.1 Conclusion

In this paper, we introduce a mathematical formulation of the GAN algorithm in terms of the mean field optimal control theory. Moreover, experiments on imitation of normal distribution and uniform distribution are done to provide empirical understanding on the training strategy. Lastly, the evolution of points under the control of both the discriminator and the generator is visualized to understand the resNet from the perspective of dynamical systems.

5.2 Discussion

As the GAN algorithm is theoretically formulated in the paper in terms of mean field optimal control, we can further apply important results from the optimal control theory like HJB equation and Pontrygin maximum principle to further analyze the problem theoretically. Moreover, as the minimax game structure is embedded in the GAN algorithm, it may be possible to find a relationship between optimal control and the Nash Equilibrium. In the experiments, the results on simple distributions are encouraging and we only experiment on the same family of distribution for both fake and true data distribution. It will be interesting to visualize how generator controls a uniform distribution to imitate a normal distribution. More experiments and visualizations on more complicated distributions may be insightful in the problem.

Bibliography

[1] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 53–65, 2018.

[2] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," in *International conference on machine learning*, pp. 1060–1069, PMLR, 2016.

[3] C. Li and M. Wand, "Precomputed real-time texture synthesis with markovian generative adversarial networks," in *European conference on computer vision*, pp. 702–716, Springer, 2016.

[4] S. E. Reed, Z. Akata, S. Mohan, S. Tenka, B. Schiele, and H. Lee, "Learning what and where to draw," *Advances in neural information processing systems*, vol. 29, 2016.

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[6] E. Weinan, J. Han, and Q. Li, "A mean-field optimal control formulation of deep learning," *arXiv preprint arXiv:1807.01083*, 2018.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[8] A. Bensoussan, J. Frehse, P. Yam, *et al.*, *Mean field games and mean field type control theory*, vol. 101. Springer, 2013.

[9] W. H. Fleming and R. W. Rishel, *Deterministic and stochastic optimal control*, vol. 1. Springer Science & Business Media, 2012.