Using Wassersein GAN to approximate stochastic process

August 17, 2020

Abstract

We proposed the method based on generative adversarial Networks(GANs) to learn the stochastic process such as geometric brownian motion and Ornstein-Uhlenbeck(OU) process.

1 Introduction

Gererative Adversarial Networks(GANs) have achieved great success at generating realistic and sharp looking images and semisupervised learning e.g. stabilizing sequence learning methods for speech and language, 3D modelling and designing cryptographic primitives [1] [2] [3] [4]. GANs have proven to be a promising alternative to traditional maximum likelihood approaches [5]. In this paper, we furthermore design the framework based on GANs to learn the stochastic process.

In GANs, two models plays minimax game. A generator samples synthetic data and discriminator distinguishes the data is real or synthetic. Conceivably some of the instability often observed while training GANs is that it is hard to find the Nash equilibrium in the zero-sum game. This makes it hard to experiment with new variants or use them in new domain [6] [7] [8], [9]. A standard analysis due to [10] shows that given the large number of parameters and samples, the win by generator implies the distribution of GAN sampling is close to target distribution. However, this cannot apply when target data for GANs is stochastic process. If minimizing the loss of generator is our priority, generator cannot capture the path of training process. We also conduct another experiments to let discriminator win the game. However, it occurs the same issue. When our priority is minimizing the loss of discriminator, the result is not stable, since some experiments stops to train when the loss of discriminator approaches to zero but the loss of generator still large. We conclude that when the discriminator or generator dominate the competition, it cannot learn the process well.

Recently, [11] [12] suggested that using Wassersein GAN(WGAN) in practice reduce instability. Although our experiments shows that WGAN learns the process well than GAN especially in avoiding mode collapsing, when we let either of discriminator and generator win the game, the model still instability. Therefore, the important open theoretical issue is whether an equilibrium always exist in the this game between generator and discriminator. The corresponding necessary condition is in a two-player game is an equilibrium. This insight allows us to construct the new rule for the generator and discriminator network to approximate equilibrium that is pure.

1.1 Our Contributions

In order to force the game toward the Nash equilibrium, we provide the new criteria for the training. We transform all of the required quantity into geometry. Fixed the sides to be 10^{-6} that connects with the loss of discriminator and generator. The model is trained until the loss of discriminator and generator with the sides can formed a right triangle. The stopping criteria has good property that it force the process approaches the minimum at $P_g = P_r$ Moreover, it doesn't require knowledge of the unknown $P_r(x)$ to optimize it. The method only applies on WGAN since GAN stops to converge at some iteration, so the method fail to apply on it.

It is interesting to note that not only the architecture of the model, the types of noise data also influence the quality of generated samples. Our experiments shows that the nosie data for the generator affects the simulation performance of GAN and WGAN style model. In GANs, the best simulation appear when the noise data is uniform distribution. However, in WGAN, the best simulation results appear when the noise data is brownian motion. Furthermore, noise data for the generator also affect the computational cost. When the noise data is uniform distribution, the loss converges faster than other types of distribution of noise data.

2 Model

2.1 Generative Adversarial Networks (GANs)

During the training, discriminator D and generator G play the following two-player minimax game during the training. The discriminator takes the input data from real and fake data and calculate the score respectively to discriminate whether a data sample comes from real data or generated by generator. The activation for the output layer of discriminator is sigmoid function so that the range of the value is $[0, 1]$. Therefore, the value can represent the probability of the input comes from real data. The generator takes the random noise data z as inout. It tries to output the data samples from distribution to fool the discriminator.

The optimizing approach for GAN is based on [10] which is composed with two loss function. The output is a sample from the input distribution P_g on $\mathbb{R}^d$ which has to be close to target distribution P_r. In the first inner loop of training, the goal is to let

$$P_r(x) > P_g(x) \tag{1}$$

Then, the GAN thinks the x has higher probability of coming from real data than being a generated sample. it updates the discriminator by ascending its stochastic gradient:

$$\bigtriangledown_x \frac{1}{m} \sum_{t=1}^{m} \left[\log D(x^{(t)}) + \log \left(1 - D\left(G\left(z^{(t)}\right)\right)\right)\right] \tag{2}$$

where x represents the variables for discriminator and t indicates the time for each vector and $z^{(t)}$ is noise sample.

After finishing the first inner loop of training, the next goal for the second training is

$$P_r(x) < P_g(x) \tag{3}$$

Then, the GAN thinks the x has lower probability of coming from real data than being a generated sample. It updates the generator by descending its stochastic gradient with only one step:

$$\bigtriangledown_z \frac{1}{m} \sum_{t=1}^{m} \log \left(1 - D\left(G\left(z^{(t)}\right)\right)\right) \tag{4}$$

where z represents the variables for generator. As the value of $D(G(z))$ increase, discriminator evaluate the output of G is real data.

The drawback of GAN is that the training is unstable. When some of the generated vectors could maximize the value of discriminator $D(G(z))$, generator will keep generated the vector. Then, the specific vectors gradually dominate the entire sampling. Therefore, the generated sampling occurs the mode collapse. From Figure 3, it obvious that some paths appear in the sampling repeatedly.

2.2 Wassersein GAN

The framework of Wassersein-GAN(WGAN) is provided by [11]. It is an alternative to traditional GAN training. The mode is trained to minimize the Earth-Mover(EM) distance between real distribution and model distribution. The EM distance between distribution of two point processes is:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \sup_{||f||_L \leq 1} E_{x \sim P_r}[f(x)] - \sup_{||f||_L \leq 1} E_{x \sim P_g}[f(x)] \tag{5}$$

[11] proved the theorem that if the function is locally Lipschitz continuous, then $W(\mathbb{P}_r, \mathbb{P}_g)$ is continuous everywhere and almost differentiable everywhere. If $W(\mathbb{P}_r, \mathbb{P}_g)$ is continuous and differentiable, then as the parameter $\theta_g \to \theta_r$ the distribution $P_g \to P_r$. The fact is that Wasserstein is differentiable almost everywhere, so the more we train the discriminator the more reliable gradient of the Wasserstein we could get.

To satisfy the distance equation (5), the loss equation for calculating gradient descent of discriminator is similar to (4)

$$\bigtriangledown_{\theta_d} \frac{1}{m} \left[\sum_{i=1}^{m} D(G(z)) - \sum_{i=1}^{m} D(x) \right] \tag{6}$$

However, we are not look for the $P_r = P_g$, so the output value does not have to be transformed by log. Since WGAN aim to minimize (5), so (6) is optimized the parameters to let the expected value of $D(G(z))$ and $D(x)$ equivalent. Because the distance is continuous and differentiable, it can can be trained until optimally. From Figure 5 and 7, it shows that it improves the mode collapse problem.

2.2.1 Noise data

The input for training generator is called noise data. We investigate three types of noise data: uniform distribution, normal distribution and brownian motion. Generator aim to capture the distribution of real examples to imitate the data sample from real data. If generator win the game, GAN evaluate the generated samples come real data. Therefore, in each iteration, the parameters is optimized so that the generated samples can foolish the discriminator. We found out that the initial distribution of noise data for generator influence the performance of GAN.

Given the same condition for other parameters In GAN. If the noise data is uniform distribution, its generated sampling is most similar to real data than normal distribution and brownian motion. However, given the same condition for other parameters In WGAN, when the noise data is brownian motion, the generated sampling is mot similar to real data. The detail of the experiment and results will described in Sec. 4.

2.3 The process of target data

To verify our model can simulate stochastic process, we select geometric brownian motion and Ornstein-Uhlenbeck(OU) process to be the stochastic process for our model.

2.3.1 Geometric Brownian Motion

The stochastic process X_t is said to follow GBM if it satisfying the following SDE

$$dX_t = \mu X_t dt + \sigma X_t dW_t \tag{7}$$

W is the Brownian motion which determine the process from beginning $S_{t=0}$ to $S_{t=T}$.

$$W_k = \sum_{t=1}^{k} b_t, \quad k = 1, \ldots, m \tag{8}$$

where b is the added randomness to the model. which stores a random number coming from the standard normal distribution $N(0, 1)$.

At fixed time t, geometric brownian motion $S_0 exp(\mu t + \sigma W(t))$ has a lognormal distribution with mean $\ln(S_0) + \mu t$ and standard deviation $\sigma\sqrt{t}$

$$f_t(x) = \frac{1}{\sqrt{2\pi t}\sigma x} \exp\left(-\frac{1}{2}\left[\frac{\ln(x) - \ln(S_0) - \mu t}{\sigma\sqrt{t}}\right]^2\right) \tag{9}$$

The solution of above SDE has the analytic solution

$$S_k = S_0 \prod_{i=1}^{k} e^{\left(\mu - \frac{1}{2}\sigma^2\right)t + \sigma W(t)} \tag{10}$$

2.3.2 Ornstein-Uhlenbeck process

The Ornstein-Uhlenbeck process differential equation is given by

$$dX_t = aX_t dt + \sigma dW_t \tag{11}$$

An additional drift term is sometimes added:

$$dX_t = \sigma dW_t + a(\mu - x_t)dt \tag{12}$$

where σ and a is constants and $\{W_t, t \leq 0\}$ is a standard Brownian motion.

The solution in terms of integral is

$$X_t = e^{\theta t} X_0 + \mu(1 - e^{\theta t}) + \sigma \int_0^t e^{\theta(s-t)} dW_s \tag{13}$$

3 Existence of Nash Equilibrium

The reason why GANs-style network are difficult to train is that both the generator model and the discriminator model are trained simultaneously in a zero-sum game. When we discuss the zero-sum game, we use the WGAN terminology, relating the min player P_1 as the generator and the max player P_2 as the discriminator. The game object for WGAN is

$$\min_u \max_v M(u, v) = \mathbb{E}_{x\sim\mathbb{P}_{real}}[D(x)] - \mathbb{E}_{x\sim\mathbb{P}_z}[D(G_u(z))] \tag{14}$$

where Player P_1 and P_2 aim to select a sequence of strategies u and v respectively to maximize its profit.

3.1 New loss function

A Nash equilibrium suppose to be a point such that the loss of discriminator and generator is at the minimum [8]. However, gradient descent fails to converge since as we reduce the loss of discriminator, the loss of generator increase and vice versa. The competition can be illustrated by Figure 1a and 1b. It is obvious that the convergence of generator is not stable.

Therefore, in order to force the loss of generator and discriminator achieve minimum, compared to the previous research, we added the gradient descent for

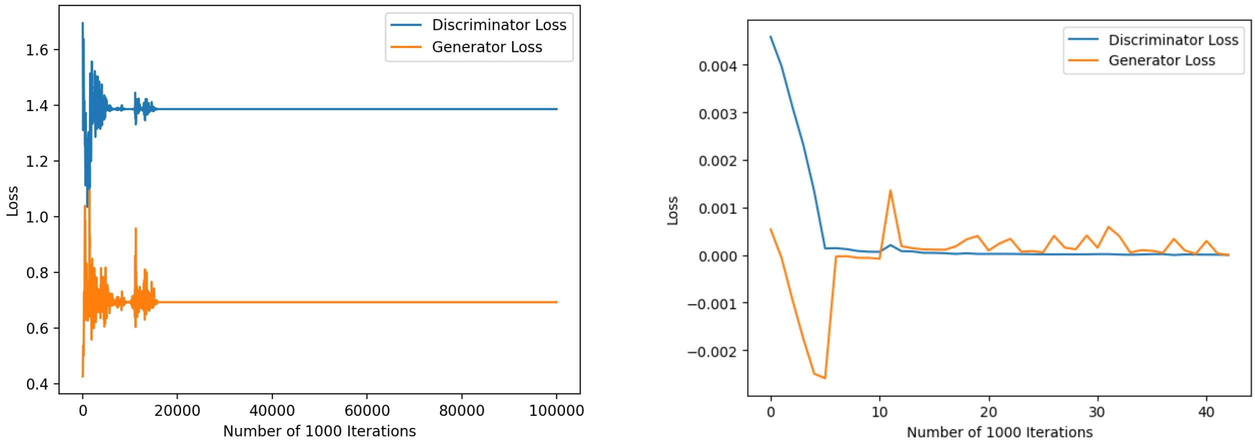
$$||D_w(x)^2 + G_\theta(z)^2|| \leq 10^{-6} \tag{15}$$

after computing the graduate descent for discriminator and generator. (15) can be explained in geometrically.

Fixed the sides to be 10^{-6} that connects with the loss of discriminator and generator. The model is trained until the loss of discriminator and generator with the sides can formed a right triangle. In case the criteria does not meet, if the number of iteration is over 40000, it allows to leave the loop if the (15) is less than 10^{-5}. Since the randomness of the intialization of parameters in model also affect the number of iterations requires to converge. The new criteria also guarantee the model can be trained until it both the loss of discriminator and generator converge. The procedure is described in Algorithm 1.

```
Algorithm 1: WGAN for stochastic process. The default values $\alpha = 0.00005, c = 0.01, n_{\text{critic}} = 5$
  Input: $x$, The geometric brownian motion paths. $z$, noise data. $D$, discriminator, $G$, generator, $\ell$, loss
    function, $m$, number of training data, $\theta_0$, initial critic parameters, $w_0$, initial generator parameters.
  Let $L$ be the sum of loss of discriminator and generator. while $L > \varepsilon$ do
    for $i = 0, \dots, N$ do
      for $j = 0, \dots, n_{critic}$ do
        $D_w \leftarrow \nabla_w \frac{1}{M}[\sum_{k=1}^{M} D_w(x_k^{(i)}) - \sum_{k=1}^{M} G(z^{(i)})]$
        $w \leftarrow +\alpha * RMSProp(w, D_w)$
        $w \leftarrow \text{clip}(w, -c, c)$
      end for
      $G_\theta \leftarrow \nabla_\theta \frac{1}{M}[\sum_{k=1}^{M} D_\theta(z_k^{(i)})]$ $\theta \leftarrow +\alpha * \text{RMSProp}(\theta, D_\theta)$
      $L \leftarrow ||D_w^2 + G_\theta^2||$
    end for
  end while
```

Many papers suggested to use minibatch sampling to train the GAN. However, In our experiment, if it trains every batch for the whole samples, the complexity rise to $O(n^3)$ and the result occurred the mode collapse. Therefore, we decide to train the entire data for one iteration.



(a) The loss of discriminator and generator for every 1000 iteration based on the new stopping rule.

(b) The loss of discriminator and generator for every 1000 iteration based on the new stopping rule.

4 Empirical results

To verify the process generated by GAN or WGAN is same stochastic process as input, the following experiments is evaluated by Finite Dimension Distribution(fdd).

4.1 Finite Dimension Distribution

If the two process have the same distribution, they have same finite-dimensional distribution. Therefore, in order to check whether the process generated by model is the same stochastic process as the training data, we compute the finite dimension distribution for two process. Finite-dimensional distribution are the joint distribution of $S_{t_1}, \dots, S_{t_n}, n \in \mathbb{N}$.

[13] and [14] provided the method to compute joint distribution (M_t, B_t). Consider the Brownian Motion(BM) $B_t = \sigma W_t + \mu t$ where W_t is the one-dimensional BM. Let $M_t = \max_{0 \le s \le t} B_s$ denote the maximum process. The joint distribution of (M_t, B_t) for Brownian motion(BM) can be found as

$$\begin{aligned} &P(M_t \ge x_t, B_t \le x_s) = \\ &\begin{cases} \exp\left(\frac{2\mu x_t}{\sigma^2}\right)\Phi\left(\frac{x_s - 2x_t - \mu t}{\sigma\sqrt{t}}\right), & \text{if } x_s \le x_t \\ \Phi\left(\frac{x_s - \mu t}{\sigma\sqrt{t}}\right) - \Phi\left(\frac{x_t - \mu t}{\sigma\sqrt{t}}\right) + \exp(\frac{2\mu x_t}{\sigma^2})\Phi\left(\frac{-x_t - \mu t}{\sigma\sqrt{t}}\right), & \text{if } x_s > x_t \end{cases} \end{aligned} \tag{16}$$

To apply (16) for computing joint distribution for geometric Brownian motion, we have to transform S_t into Brownian motion. Therefore, in geometric brownian motion, for $0 \le t_0 < t_1 \cdots < t_n$, its relative increments $\frac{S_{t_n}}{S_{t_{n-1}}}$ are independent lognormal r.v.s. For example,

$$L_1 = \frac{S_{t_1}}{S_{t_0}} = e^{S_{t_1}}, \quad L_2 = \frac{S_{t_2}}{S_{t_1}} = e^{S_{t_2} - S_{t_1}} = e^{\sigma(W_{t_2} - W_{t_1})} \tag{17}$$

For example, Given the initial value $S_0 = 10$. To estimate the probability that the value between 9 to 13 between t and s, we need to find

$$\begin{aligned} &Pr(9 \ge P_t \le 13) \\ &= Pr\left\{|N(0, t-s)|(1/\sigma)\ln\left(\frac{9}{S_0}\right) \le B(t) \le (1/\sigma)\ln\left(\frac{13}{S_0}\right)\right\} \end{aligned} \tag{18}$$

Then, applying the (16), since $\mu = 0$, we get

$$\Phi\left(\frac{13 - \mu t}{\sigma\sqrt{t}}\right) - \Phi\left(\frac{9 - \mu t}{\sigma\sqrt{s}}\right) \tag{19}$$

Then, we could obtain the probability of the value between 9 to 13 in time t to time s.

For computing the joint distribution for experimental data, the GAN and WGAN sampling results were split into 50x50 meshgrids. Then, the ratio of number of points in the grids to the number of all the points in given time interval is its density. The theoritical joint distribution of the grids in given time interval is computed by

$$\Phi\left(\frac{x_s - \mu t}{\sigma\sqrt{t}}\right) - \Phi\left(\frac{x_t - \mu t}{\sigma\sqrt{t}}\right) \tag{20}$$

4.2 The simulation of Geometric Brownian Motion by GAN

Given the real data $\{s(i) \in A\}, i = 1, \dots, 10\}$ with intial value $[-1, 1]$ for all $A \subset \mathbb{R}^{10}$. The real data is generated by following (10). We use uniform distribution, normal distribution and brownian motion to be the candidate of noise distribution. The reason we choose these three distribution is that since the geometric brownian motion is dependent process, we assume the dependent process can fool the discriminator easily. Therefore, we use uniform and normal distribution to be the independent process group and brownian motion to be the dependent group. Then, we can investigate the GAN's performance under different types of noise data.

We compared the real distribution with model distribution for each dimension after training. When the noise data is uniform distribution, the probability distribution is most close to target distribution, so we use uniform distribution to illustrate the comparison of model distribution and target distribution in each dimension. The results of output distribution are shown in Figure 2.

The results were evaluated by fdd. The joint distribution (S_{t-1}, S_t) for all the grids is calculated by (20). If the noise data for generator is uniform distribution, the average difference between theoreticaland experimental probability for all the grid is0.00099.

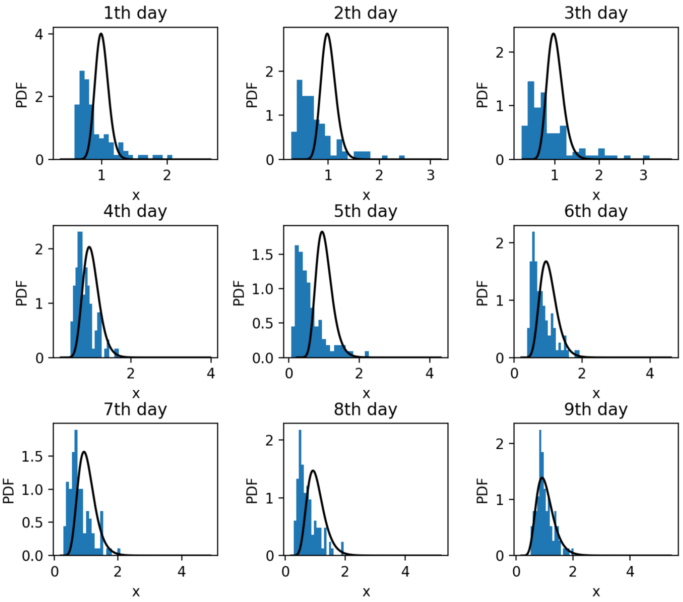


Figure 2: Probability distribution of GAN sampling and theoritical distribution in each dimension.

The path simulation for three experiment is shown in Figure 3. The model collapse exisit GAN in three experiment which means that some of paths dominate the generative sampling. The generator kept producing samples that come from that specific mode and ignoring the other classes most of time. Moreover, if the model distribution is close to the real distribution in some specific dimensions, the mode collapse have larger impact on those dimension.

4.3 The simulation of Geometric Brownian Motion by WGAN

Given the real data $\{s(i) \in A\}, i = 1, \ldots, 10\}$ with intial value $s(0) = 1$ for all $A \subset \mathbb{R}^{10}$. The real data is followed by geometric Brownian motion which is generated by following (10). The setting of noise data is same us the GAN experiment.

Compared with GAN, each dimension of probability distribution of the WGAN sampling is close to the dimension of theoretical probability. When the noise data is brownian motion, the model distribution is closet to target distribution, so we use brownian motion to illustrate its ability to learn target distribution. The probability distribution result is presented in Figure 4.

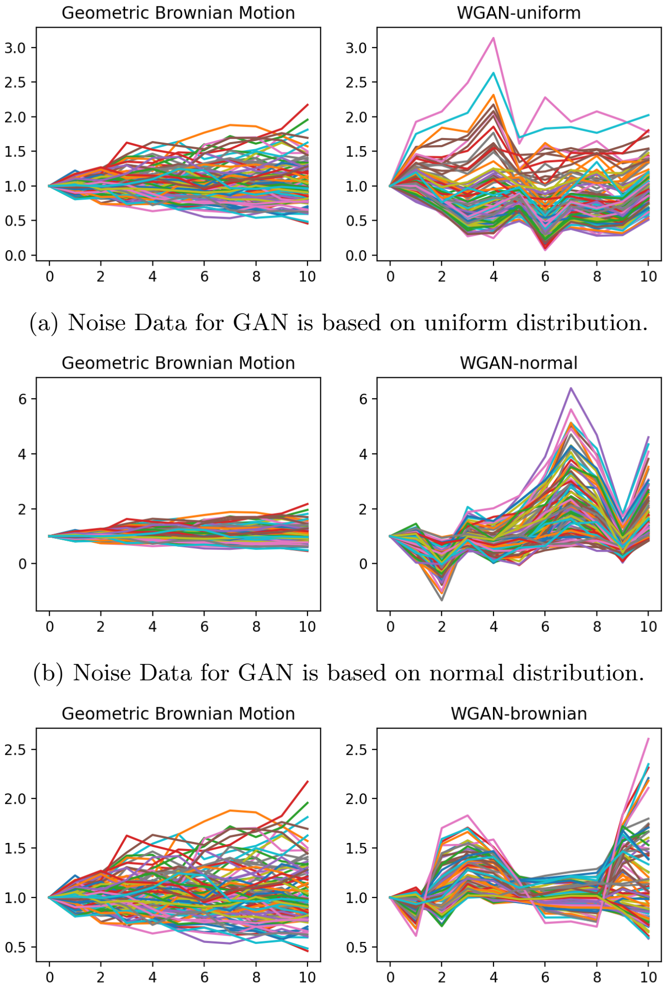
The simulation results are shown in Figure 5. The generated sampling in three experiment looks more close to the real paths than the sampling created by GAN. Although the best result appear when the noise data follow the brownian process, if the noise data is uniform distribution, it converges fastest.

Furthermore, after using small dataset to train WGAN, the WGAN can simulate the larger dataset well. For example, we used 100 geometric Brownian motion for real and noise data. Then, using the trained model to generate 1000 data points for simulating geometric Brownian motion. From Figure 6, the larger output data follows the patter of real data.

4.4 The simulation of Ornstein-Uhlenbeck process by WGAN

In this experiment, we use another stochastic process: OU process to be the target data for WGAN. The setting of noise data is same as previous experiments. The dimension in OU process is specified as time interval from $t = 0$ to $t = 2$. The simulation results of OU process are shown in Figure 7.

The results were evaluated by fdd. The joint distribution (S_{t-1}, S_t) for all the grids is calculated by (20). The average of the difference for the joint distribution between theoretical and empirical data is 0.00488.



(a) Noise Data for GAN is based on uniform distribution.

(b) Noise Data for GAN is based on normal distribution.

(c) Noise Data for GAN is based on brownian distribution.

Figure 3: Simulation results of GAN

5 Conclusion

We presented a novel approach for WGAN to learn dynamic process. The problem this paper concerned with is unsupervised learning. It does not require prior knowledge about the underlying true process. The paper investigated the difficulties to find the equilibrium in zero-sum game. Furthermore, the paper proposed the new criteria that it force WGAN to be trained until it approach ϵ-Nash Equilibrium. We also observe the noise data for generator influence the performance of WGAN. When the noise data is brownian motion, WGAN generated samples look more realistic to the target paths than the noise data based on uniform distribution and normal distribution.

For the future work, First, we will propose a new approach that guarantee both the discriminator and generator can approach the Nash equilibrium in the training phase; Second, providing the mathematical proof to analyze the Nash-Equilibrium in WGAN when the real data is stochastic process. Third, investigating the relationship between noise data and WGAN to explain why some types of noise data can improve to the performance of WGAN.

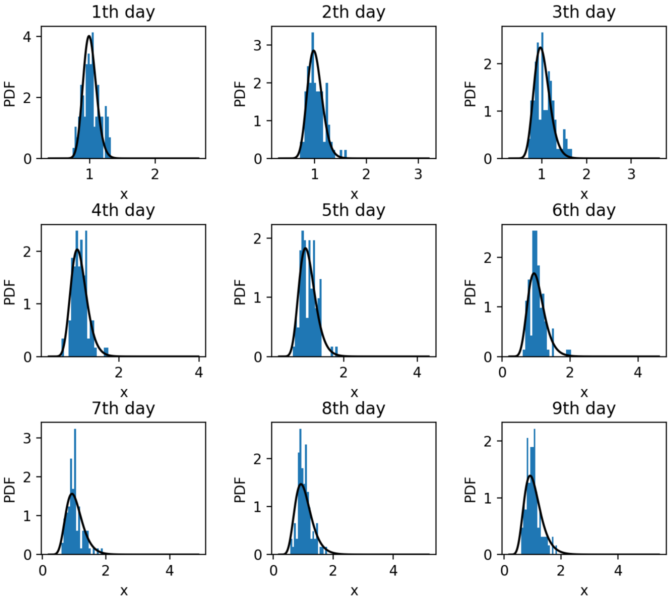
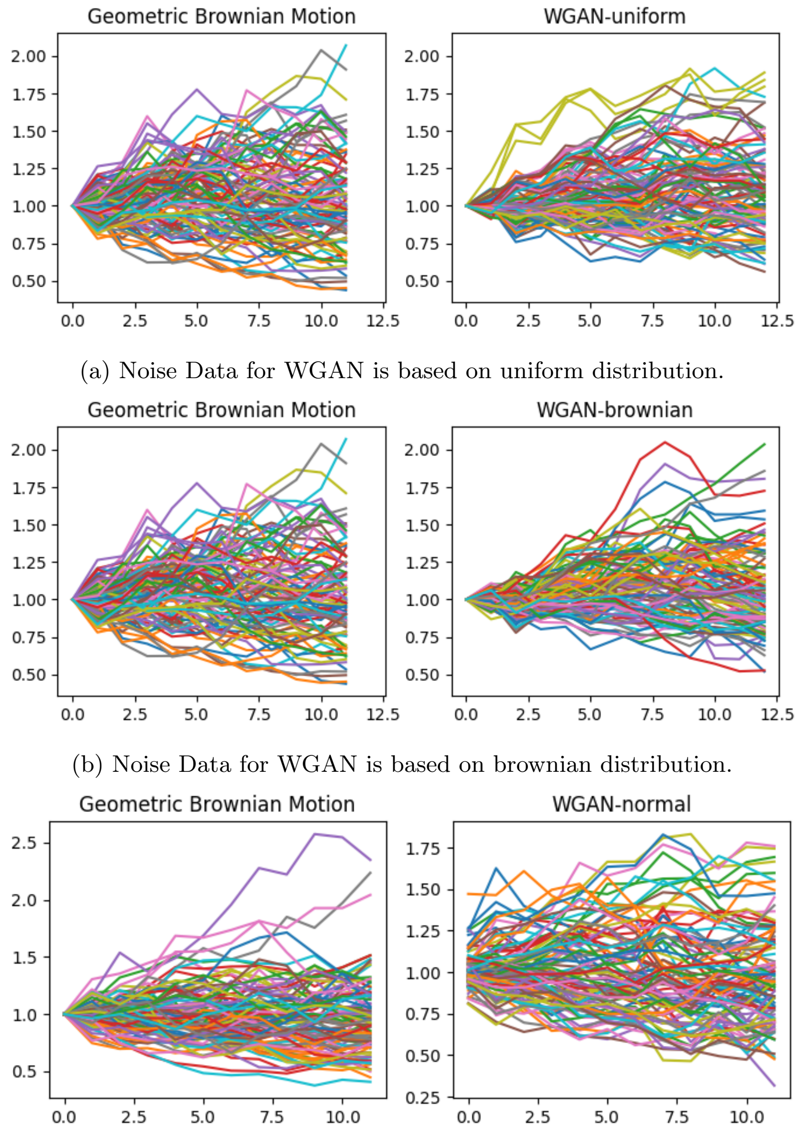


Figure 4: Probability distribution of WGAN sampling and theoritical distribution in each dimension.



(a) Noise Data for WGAN is based on uniform distribution.

(b) Noise Data for WGAN is based on brownian distribution.

(c) Noise Data for WGAN is based on normal distribution.

Figure 5: Simulating geometric brownian motion by WGAN

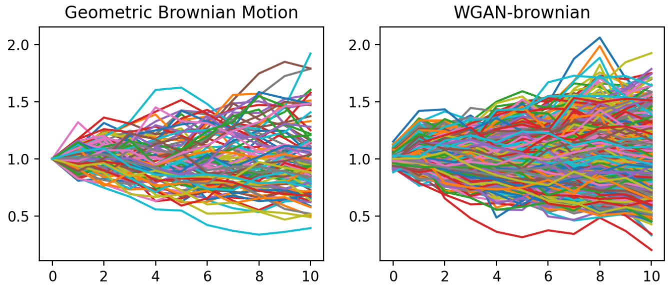
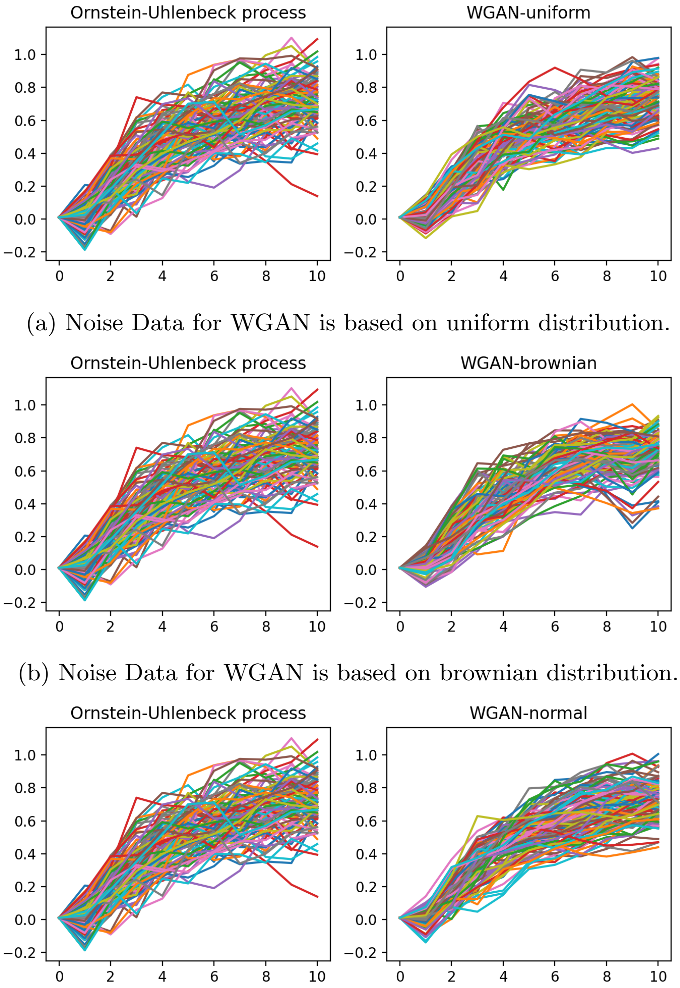


Figure 6: We using smaller data for training. Then, applying the trained WGAN model to simulate larger dataset.



(a) Noise Data for WGAN is based on uniform distribution.

(b) Noise Data for WGAN is based on brownian distribution.

(c) Noise Data for WGAN is based on normal distribution.

Figure 7: Simulating Ornestein-Uhlenbeck process by WGAN

References

[1] Martn Abadi and David G. Andersen. Learning to protect communications with adversarial neural cryptography, 2016.

[2] Muhammad Sarmad, Hyunjoo Jenny Lee, and Young Min Kim. Rl-gan-net: A reinforcement learning agent controlled gan network for real-time point cloud shape completion, 2019.

[3] Guy Tevet, Gavriel Habib, Vered Shwartz, and Jonathan Berant. Evaluating text GANs as language models. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2241–2247, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[4] William Fedus, Ian Goodfellow, and Andrew Dai. Maskgan: Better text generation via filling in the. 2018.

[5] Ferenc Huszr. How (not) to train your generative model: Scheduled sampling, likelihood, adversary?, 2015.

[6] Martin Arjovsky and Lon Bottou. Towards principled methods for training generative adversarial networks, 2017.

[7] Sanjeev Arora and Yi Zhang. Do gans actually learn the distribution? an empirical study, 2017.

[8] Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. Generalization and equilibrium in generative adversarial nets (gans), 2017.

[9] Paulina Grnarova, Kfir Y. Levy, Aurélien Lucchi, Thomas Hofmann, and Andreas Krause. An online learning approach to generative adversarial networks. *ArXiv*, abs/1706.03269, 2017.

[10] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS14, page 26722680, Cambridge, MA, USA, 2014. MIT Press.

[11] Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *ArXiv*, abs/1701.07875, 2017.

[12] Shuai Xiao, Mehrdad Farajtabar, Xiaojing Ye, Junchi Yan, Le Song, and Hongyuan Zha. Wasserstein learning of deep generative point process models, 2017.

[13] Chin-Shan Chuang. Joint distribution of brownian motion and its maximum, with a generalization to correlated bm and applications to barrier options. *Statistics & Probability Letters*, 28(1):81 – 90, 1996.

[14] Brownian motion and stochastic flow systems, j. m. harrison, wiley, new york, 1985. isbn 0-471-81939-5. no. of pages: xxi + 1. price: 36.95. *Applied Stochastic Models and Data Analysis*, 4(2):139–140, 1988.