GENERATIVE ADVERSARIAL NETWORKS IN FINANCIAL MARKETS

José Vinícius de Miranda Cardoso, Yixin Men, Shunkang Zhang

CSIC5011 – Topological and Geometric Data Reduction and Visualization

THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Introduction

- Generative adversarial network (GAN) is a class of machine learning framework developed by Goodfellow *et al.* 2014.
- GANs are used to **recover the statistical distribution of the input data** such that, when trained, a GAN is able to generate fake data that is indistinguishable from real data.
- The general architecture of a GAN is depicted in the Fig. 1 below:

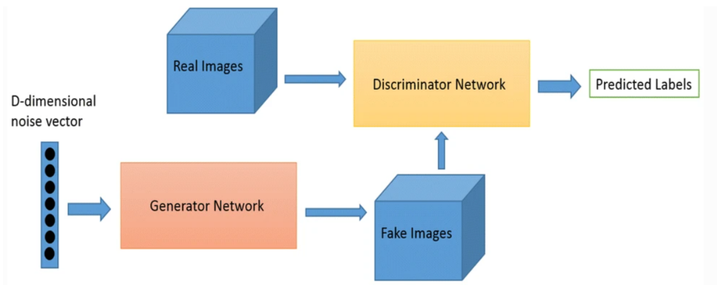

Fig. 1: The general architecture of GAN for Image data.

Model and Optimization Problem

- ***The Generator*** (G) is a network that is used to generate data using random noise prior distribution p_g. The generated samples using noise are recorded as $G(\boldsymbol{z})$. The generator objective is to increase the error rate of the discriminator, *i.e.*, fool the discriminator by producing samples that the discriminator thinks are part of the input data.
- ***The Discriminator*** (D) evaluates whether or not samples produced by the generator are representative of the input data distribution. Its output is a single number representing the probability of a sample belonging to the true data distribution.

A GAN can be ultimately trained by solving the following two-player minimax game:

$$\min_G \max_D \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]. \tag{1}$$

- Ideally, the distribution of the generator will converge to the true distribution of the data, i.e., $p_{\text{data}}(\boldsymbol{x}) = p_g(\boldsymbol{x})$ and the discriminator will output the result of a coin toss, *i.e.*, $D(\boldsymbol{x}) = \frac{1}{2}\ \forall\ \boldsymbol{x}$.

Analysis in Function Space

Given $\mu \in \mathcal{P}(\mathbb{R}^d)$ with the density q, we use the f-divergence to measure the discrepancy between μ and ν which is defined as

$$\mathbb{D}_f(q\|p) = \int p(\mathbf{x}) f\left(\frac{q(\mathbf{x})}{p(\mathbf{x})}\right) d\mathbf{x}, \tag{2}$$

where $f : \mathbb{R}^+ \to \mathbb{R}$ is a convex and continuous function satisfying $f(1) = 0$. We also require $f(\cdot)$ is twice-differentiable. Let $\mathcal{F}[q]$ denote the energy functional $\mathbb{D}_f(\cdot\|p) : \mathcal{P}(\mathbb{R}^d) \to \mathbb{R}^+ \cup \{0\}$ for simplicity. We consider a curve $\mu_t : \mathbb{R}^+ \to \mathcal{P}(\mathbb{R}^d)$ and μ_t admits the density q_t. Let $\mathbf{v}_t = -\nabla\left(\frac{\delta \mathcal{F}}{\delta q_t}(q_t)\right) : \mathbb{R}^+ \to (\mathbb{R}^d \to \mathbb{R}^d)$ be the velocity vector field with $r_t(\mathbf{x}) = \frac{q_t(\mathbf{x})}{p(\mathbf{x})}$.

Learning Algorithm

- The implementation of the training process for a basic GAN can be summarized in Algorithm 1.

Algorithm 1: Minibatch stochastic gradient descent training of generative adversarial nets.

for *number of training iterations* **do**

▷ Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.

▷ Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.

▷ Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right].$$

▷ Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.

▷ Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end

Experimental Results

Synthetic Data: we first perform an experiment with synthetic data for testing purposes. We generate synthetic data from a Gaussian distribution as $\boldsymbol{x} = [\boldsymbol{x}_1, \boldsymbol{x}_2], \boldsymbol{x}_1 \sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{I}_{p\times p}\right), \boldsymbol{x}_2 \sim \mathcal{N}\left(5 \cdot \mathbf{1}, 2\boldsymbol{I}_{p\times p}\right)$, where $p = 50$, and we observe 40000 samples from $\boldsymbol{x}_1$ and 10000 samples from $\boldsymbol{x}_2$. In this example we consider that $\boldsymbol{x}_1$ represents the random vector from the true data distribution and $\boldsymbol{x}_2$ represents outlier samples. We then apply Algorithm 1 to learn the distribution of the true data.

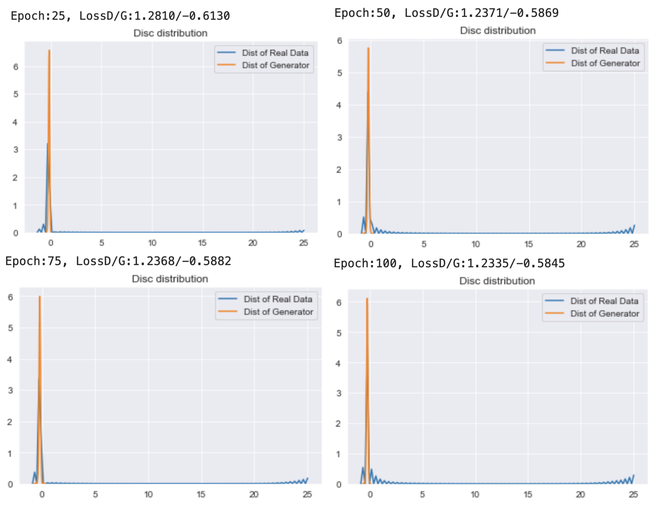

Fig. 2: Convergence of Generator Distribution

In Fig.2, we can observe the convergence of the Generator distribution over the number of epochs, i.e., complete passes over the training dataset. As expected, as the number of epochs increase, the distribution of the generator fits better the distribution of the true data.

Experimental Results

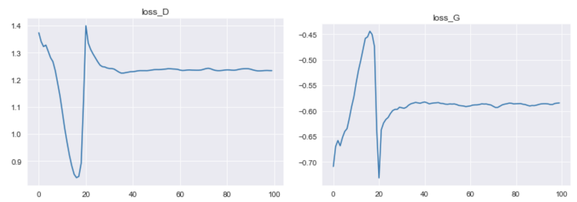

Fig. 3: Loss functions for discriminator (left) and generator (right).

In Fig.3, we can observe the loss functions of the generator and discriminator. Again, as expected, as the number of epochs increase, they convergence to a stationary point of the minimax game.

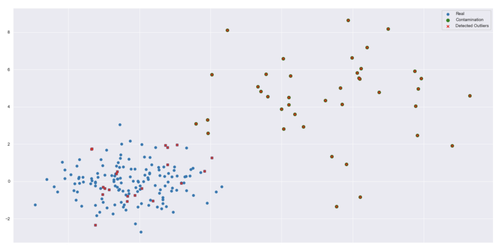

Fig. 4: Loss functions for discriminator (left) and generator (right).

In Fig.4 we observe that the discriminator is in fact able to distinguish which samples come from the true data distribution and which ones come from the outlier distribution, making only a few mistakes.
Now we shift our attention to stock market data, where we collect data from a set of stocks from 2015 to 2020. The log-prices of the stocks selected are shown in Fig.5. We use GANs to learn the empirical data distribution of this set of stocks, as depicted in Fig.6.

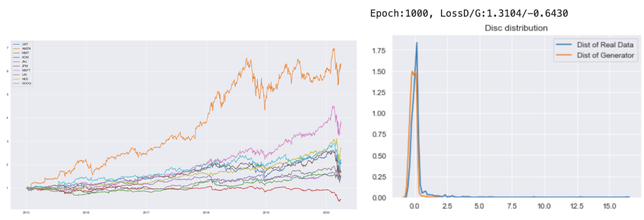

Fig. 5: S&P500 stock log-prices

As a conclusion, GANs are extensively being applied to generate synthetic financial data and that has been an increasingly important research topic. We encourage interested readers to check out references [2] and [3].

References

1 Goodfellow, Ian; *et al.*. Generative Adversarial Networks. Proceedings of the International Conference on Neural Information Processing Systems (NIPS 2014). pp. 2672–2680.

2 Marti, Gautier; CorrGAN: Sampling Realistic Financial Correlation Matrices Using Generative Adversarial Networks, arXiv:1910.09504, 2019.

3 Wiese, Magnus; *et al.*. Quant GANs: Deep Generation of Financial Time Series, arXiv:1907.06673, 2019.