Generative Modeling Project

Yassine BEN JEMAA, Irshath NAGOURI

December 22, 2023

1 Introduction

The goal of this project is to implement a generative model as part of a financial study. The X dataset is composed of n1 = 720 samples made up of 4 features. This is not a Time Series, as the time intervals are not fixed, but must be considered as independent and identically distributed according to the distribution μ. The aim of this project will be to explain the theoretical part of our generative model in order to define potential optimizations given the data we generate. The metrics used are the Anderson-Darling distance and the absolute Kendall Error.

2 Dataset

Before implementing our generative model, let's analyze the observed data. The dataset consists of four features, all centered around zero and with no negative values. This distribution suggests that our data may follow a half-cut Gaussian or similar distribution, which is a common assumption in many generative models. However, we also observed the presence of extreme values, which are significantly different from the majority of the data.

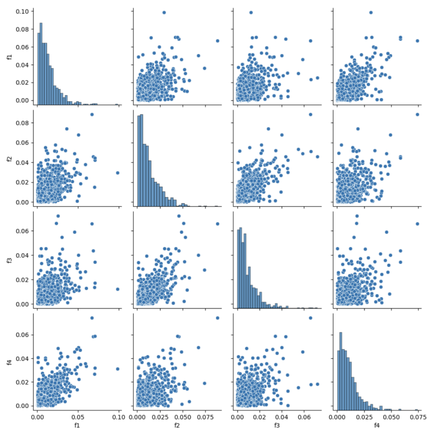

Figure 1: Correlation of the features

Note that the f2 and f3 components are the most correlated. To take this into account, a regularization term can be added. However, as the dimension is already low, this will not be done.

In term pre-processing, it will be incoherent to normalize our data because the goal of the project is to sample extreme event and normalizing them will give us negative (so truly positive) values.

3 VAE

VAE's are generative model that are primarly used for high-dimensional space space which is not the case here. However, VAE's shows interesting properties such as weak assumptions and fast training. Recall that the goal of VAE is to maximize the likelihood of the Z-induced law in the decompression phase:

$$P(X) = \int (P(X|z;\theta)P(z)dz) \tag{1}$$

The choice of this output distribution is often Gaussian, i.e., $P(X|z;\theta) = \mathcal{N}(X|f(z;\theta), \sigma^2 I)$ where z is the latent variables, the mean is parameterized by a vector θ and σ a **hyper-parameter** that'll be tuned in order to improve our VAE. In fact, $f(z;\theta)$ will be a network using its first few layers to map the normally distributed z's to the latent values and use later layers to map those latent values to a fully-rendered sample.

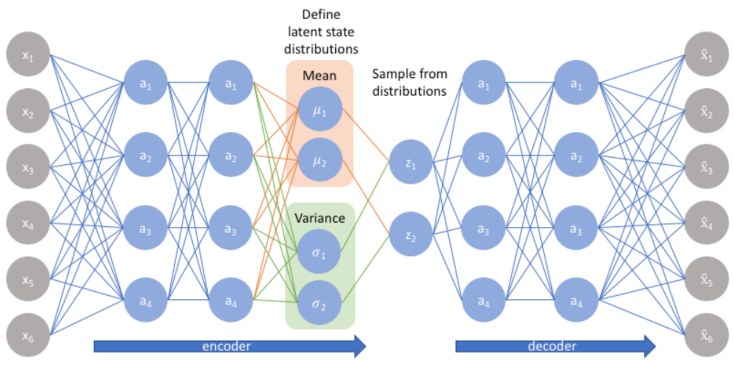

Figure 2: VAE Architecture

Besides, the *parametrization trick* need to be applied in order to back-propagate the error through a layer that samples z from $Q(z|X)$, which is a non-continuous operation and has no gradient. To do so, the sampling is moved to an input layer, meaning that $z = \mu(X) + \Sigma^{1/2}(X) * \varepsilon$

3.1 Classic Approach

Optimizing their performance involves fine-tuning various hyperparameters that significantly impact the model's overall effectiveness. One crucial hyperparameter is the architecture, i.e the number of layers in the encoder and decoder networks. Experimenting with the depth and width of these networks can influence the VAE's capacity to capture complex latent structures within the data. Additionally, hyperparameters such as the latent space dimension and the variance parameter σ, play important roles in determining the convergence speed and the expressiveness of the latent representation. The number of training epochs and batch size and the learning rate are fundamental temporal considerations, influencing the duration and granularity of the learning process.

Hyperparameters
Number of layers in encoder and decoder
Learning rate
Variance parameter (σ) in the latent space
Number of training epochs
Number of batches
Activation Layers (*Leaky Relu* and *Elu*)
Latent Space dimension (2 or 3)

Table 1: Hyperparameters for VAE Optimization

Let's focus more on σ. Among all these hyper-parameters, σ controls the amount of noise introduced during the sampling process in the latent space. In a mathematical viewpoints, it comes in play in the Reconstruction Loss and don't intervene in the KL distance:

$$logP(X|z) = C - \frac{1}{2\sigma^2}\|X - f(z)\|^2 \tag{2}$$

If σ is increased, the model will generate more diverse outputs, as more noise is introduced during sampling. However, this might also lead to more unrealistic outputs, as the model is allowed to explore more distant points in the latent space.

If sigma is small, the model is more confident about the reconstruction and the penalty for not matching the original data exactly is larger. However, using a small sigma can lead to overfitting.

3.2 First Result

By playing with all those parameters, we obtain the following results:

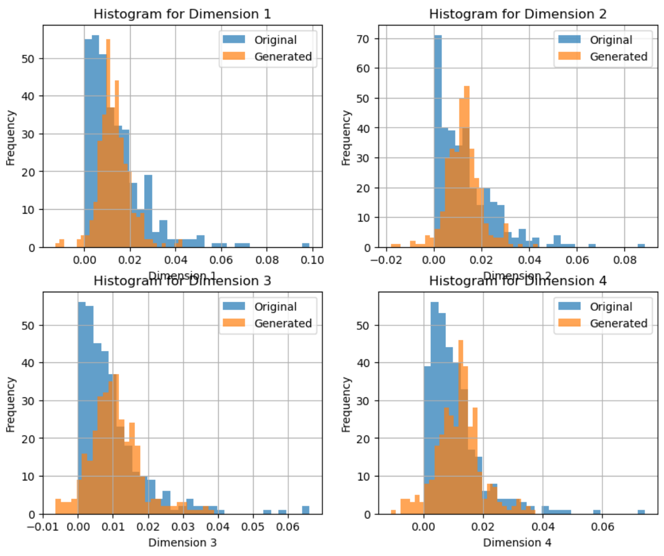

Figure 3: $\sigma = 1$

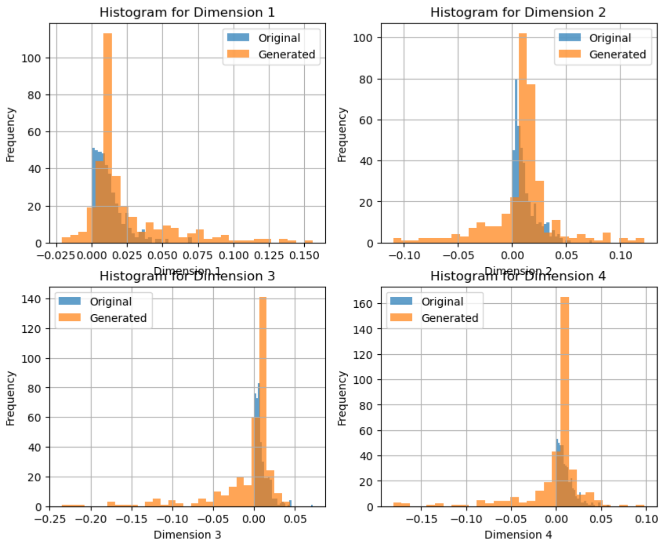

Figure 4: $\sigma = 0.01$

To avoid overfitting, $sigma = 1$ in all the process.

Let's see the results of some metrics. The first metric used is the Anderson-Darling distance. The Anderson-Darling distance is a measure of the difference between the empirical cumulative distribution function of the sample data and the cumulative distribution function of the theoretical distribution.

Another metric is the Absolute Kendall Error which measures the ordinal association between two quantities. The Absolute Kendall Error is a measure between 0 and 1.

Table 2: Table of metrics by the size of the architecture

Metric	Size		
	Enc = Dec	Enc < Dec	Enc > Dec
Anderson Darling Error	2.13	2.41	1.07
Absolute Kendall Error	0.25	0.20	0.23

Using an asymmetrical architecture with a larger encoder produces more convincing results. Indeed, small values are given in the encoder so it needs to be larger in order to capture more information. Moreover, as small values is generated, *leaky-relu* and *elu* activation function are favorable. Also, there is an inverse relationship between Anderson-Darling and Kendall Error which is normal because one compute the marginal error and the other the dependence.

To **summarize**, a classic approach allow to have satisfactory result for the Marginals metrics but the dependence metric struggles to improve. Let's try another implementation.

4 Other Approaches

4.1 WAE

Instead of using the classic VAE loss, we propose using Wasserstein distance as a measure of distributional similarity.

The Wasserstein distance W(u,v), being a symmetric metric, is defined as an optimal transport from one distribution u to the other v:

$$W_p(u,v) = \left(\inf_{\gamma \in \tau(u,v)} \int d(x,y)\, p\, d\gamma(x,y) \right)^{\frac{1}{p}} \tag{3}$$

the Wasserstein distance would not explode with extreme value such as 0 or become meaningless when comparing two distributions without any overlap.

Wasserstein distance tends to encourage the model to cover more modes of the data distribution, potentially addressing issues related to mode dropping (reducing variety in the samples), where certain modes are not effectively captured by the VAE. However, its principal disadvantages is its computationally more demanding than the traditional KL divergence. Also, understanding the implications of Wasserstein distance on the latent space and the generated samples might require additional analysis.

The Wasserstein distance is given by the following formula:

$$(W_2(Q(z|x) \| P(z)) = \|\mu - 0\|_2^2 + \mathrm{Tr}\left(\left(\sigma + I - 2(I^{1/2} \sigma I^{1/2})^{1/2} \right) \right) \tag{4}$$

where μ and σ are the parameters learned in the encoder.

The new elbow method would be:

$$ELBO_\lambda = \int Q(z|x) \log(P(x|z))\, dz - \lambda W_2(Q(z|x) \| P(z)) \tag{5}$$

where λ is a new hyper-parameter to fine-tuned that that controls the weight of the Wasserstein distance term.

Results: It appears that the KL divergence was more adapted to our problem as we obtain this results:

$$Anderson\ Darling\ Distance = 2.3\ and\ Absolute\ Kendall\ Error = 0.29 \tag{6}$$

Besides, the models depends a lot of the hyperparameter λ. A potential improvment would be that instead of fine-tuning the λ parameter, it could have dynamically been adjusting. A time series model such as *ARIMA* generates a predicted T_{t+1}, which is used to decide whether λ_{t+1}) should increase or decrease. ([1])

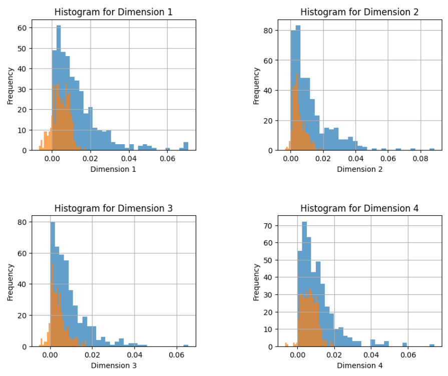

Figure 5: WAE results with $lambda = 0.01$

4.2 Resampling the prior for VAE

According to [2], using a simple prior such as the gaussian distribution $\pi(z) = \mathcal{N}(z; 0, I)$ may lead to underfitting. Which is why they proposed a new method called Learned Accept/Reject Sampling (**Lars**). The goal is to approximate a more complex distribution $q(z)$ which we can not evaluate from but sample from it. We approximate $q(z)$ with the following density :

$$p(z) = \frac{\alpha_\lambda(z)\pi(z)}{Z} || Z = \int \alpha_\lambda(z)\pi(z)dz \tag{7}$$

Where $\alpha_\lambda(z)$ is a neural network that will be trained during backpropagation representing the probability of accepting a sample from $\pi(z) = \dot{S}$o we have :

$$p(z) < \frac{\pi(z)}{Z} \tag{8}$$

We can thus apply the rejection sampling algorithm to sample from $p(z)$ since we know how to sample from $\pi(z)$.

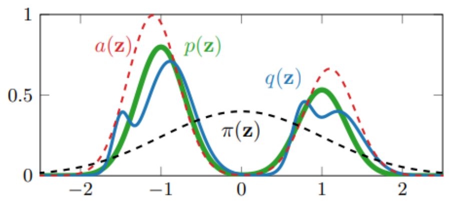

Figure 6: Illustration of the Lars method

Here, for a given point x_r in the dataset, we will sample $z_r \sim q(z|x_r)$. So the $q(z)$ we previously discussed about will be the conditional law of the decoder. We then modify the classical VAE by simulating Z and adding additional steps in the network as described below in Algorithm 1 and 2.

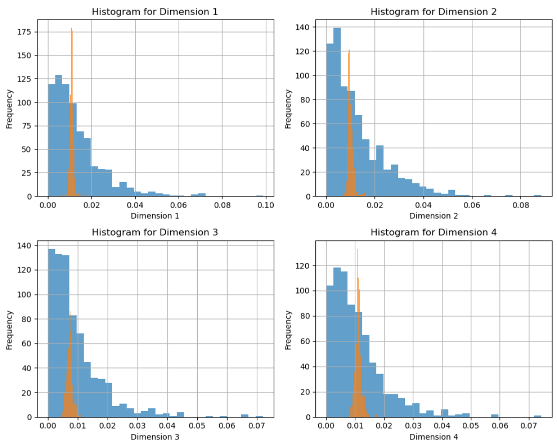

Figure 7: Results from using lars

When we utilize $p(z)$ as an approximation, it essentially empowers us to generate new latent data by drawing from a more intricate distribution. This approach aims to produce data that exhibits characteristics resembling those from the desired distribution. We believe we can obtain better results if we redesign the neural networks modeling the acceptance probability. We also limited the latent dimension to two since rejection sampling struggles in high dimension. We implemented this model ourselves, but with limited computational resources, we have limited performance. This algorithm involved Monte Carlo methods that could not be used to their full potential as simulating a huge amount of numbers in our personal computers is difficult.

```
Algorithm 1: Estimation of log $Z$ (Eq. (A.30)) in the objective during training
  input  : Minibatch of samples $\{\mathbf{z}_r\}_r^R$ from $\{q(\mathbf{z}|\mathbf{x}_r)\}_r^R$; $S$ samples $\{\mathbf{z}_s\}_s^S$ from $\pi$; previous moving average $\langle Z \rangle_i$
  output : Updated moving average $\langle Z \rangle_{i+1}$; log $Z$ estimate for current minibatch $\{\mathbf{x}_r\}_r^R$
1 $Z_S \leftarrow \frac{1}{S}\sum_s^S a(\mathbf{z}_s)$
2 for $r \leftarrow 1$ to $R$ do
3     $Z_{r,\mathrm{curr}} \leftarrow \frac{1}{S+1}\left[SZ_S + \texttt{stop\_grad}\left(\frac{\pi(\mathbf{z}_r)}{q(\mathbf{z}_r)}\right) a(\mathbf{z}_r)\right]$
4     $Z_{r,\mathrm{smooth}} \leftarrow (1-\epsilon)\langle Z \rangle_i + \epsilon Z_{r,\mathrm{curr}}$
5     $Z_r \leftarrow Z_{r,\mathrm{curr}} + \texttt{stop\_grad}(Z_{r,\mathrm{smooth}} - Z_{r,\mathrm{curr}})$
6 end
7 $\langle Z \rangle_{i+1} \leftarrow \frac{1}{R}\sum_r^R \texttt{stop\_grad}(Z_{r,\mathrm{smooth}})$
8 $\log Z \leftarrow \frac{1}{R}\sum_r^R \log Z_r$
```

Algorithm 2: Pseudo code for training of a VAE with resampled prior (for untruncated resampling)

```
input  : dataset of images $\mathcal{D}_{\text{train}} = \{\mathbf{x}_n\}_n^N$
output : trained model parameters $\varphi, \theta$ (encoder/decoder) and $\lambda, \log Z$ (resampled prior)
initialize parameters of encoder $\varphi$ and decoder $\theta$
initialize parameters of the resampled prior $\lambda$ and the moving average $\langle Z \rangle$
for $it \leftarrow 1$ to $N_{it}$ do
    sample a minibatch of data $\{\mathbf{x}_r\}_r^R$
    sample latents $\{\mathbf{z}_r\}_r^R$ from the encoder distribution $\{q_\varphi(\mathbf{z}|\mathbf{x}_r)\}_r^R$ (using reparameterization)
    evaluate the reconstruction term: recon $\leftarrow \frac{1}{R}\sum_r^R \log p_\theta(\mathbf{x}|\mathbf{z}_r)$
    update moving average $\langle Z \rangle$ and compute $\log Z$ using Algorithm 1
    evaluate the KL term: kl $\leftarrow \frac{1}{R}\sum_r^R [\log q_\varphi(\mathbf{z}_r|\mathbf{x}_r) - \log \pi(\mathbf{z}_r) - \log a_\lambda(\mathbf{z}_r)] + \log Z$
    evaluate the objective: recon $-$ kl
    backpropagate gradients into all parameters $\varphi, \theta$ and $\lambda$
end
compute final normalizer for evaluation: $Z \leftarrow \frac{1}{S_{\text{eval}}}\sum_s^{S_{\text{eval}}} a(\mathbf{z}_s)$ with $\mathbf{z}_s \sim \pi(\mathbf{z})$
```

5 Conclusion

In summary, VAEs captures quite well complex data distributions. On the other hand, Wasserstein Autoencoders (WAEs) is less stable. The inclusion of Layer-wise Adaptive Rate Scaling (LARS) for dynamic learning rate adjustments enhances the optimization process for both VAEs and WAEs, promoting faster convergence and improved generalization but its computation limits its use. GAN and WGAN could be implemented to see if they can provide more interesting results.

References

[1] Zichuan, Chen. *Towards efficient Variatonal Auto-Encoder using Wasserstein. 2021*

[2] Matthias Bauer, Andriy Mnih *Resampled Priors for Variational Autoencoders 2019*

[3] Doersch, Carl. *Tutorial on Variational Autoencoders 2021*