Supervised GAN to approximate SDE solutions

Mohamed Raed Blel*,[1]

[1]Laboratoire de Probabilités Statistiques et Modélisation, LPSM

April 6, 2023

Contents

Introduction

Generative methods based on neural networks are gaining widespread use in machine learning in general and in quantitative finance in particular. The supervised GAN (Generative Adversarial Network) implemented here is based on [2]. This GAN aims to reproduce strongly the trajectories given from an exact simulation or discretized scheme. The method has been used on the CIR model and we have got better results than the ones obtained in the studied reference, since:

- the generation in their paper fails for small time steps, while our generators work well for small and large time steps;
- we are able to parameterize the generator with the parameters of the CIR process.

1 A brief description of the Method

Let be the probability space $(\Omega, \mathcal{F}, P)$ and let $\{W_t\}_{t\geq 0}$ be a standard Brownian motion with respect to its natural filtration $\mathcal{F}_t = \sigma(\{W_s : s \leq t\}), t \geq 0$. Let be $\mathcal{P}$ a parameter space and

*Corresponding author: www.linkedin.com/in/mohamed-raed-blel-link

let us introduce the following stochastic differential equation that depends on a parameter $\alpha \in \mathcal{P}$:

$$dS_t^\alpha = b_\alpha(S_t^\alpha, t)dt + \sigma_\alpha(S_t^\alpha, t)dW_t, \quad \forall t \in [0, T],$$

where $\{S_t\}_{t\geq 0}$ is a continuous-time random process on $\mathbb{R}$ adapted to the filtration $\mathcal{F}_t$ and where $b_\alpha(S_t^\alpha, t)$ and $\sigma_\alpha(S_t^\alpha, t)$ are $\mathcal{F}_t$-measurable random process on $\mathbb{R}$ for each $\alpha \in \mathcal{P}$. Then the Ito process is written as,

$$S_{t+\Delta t}^\alpha = S_t^\alpha + \int_t^{t+\Delta t} b_\alpha(S_s^\alpha, s)ds + \int_t^{t+\Delta t} \sigma_\alpha(S_s^\alpha, s)dW_s, \quad \forall t \geq 0 \text{ P-a.s}$$

The idea is to approximate $S_{t+\Delta t}^\alpha$ by a neural network using supervised GAN to which we give the same inputs as the reference method, i.e. we want to obtain $\hat{S}_{t+\Delta t}^\alpha = G(\hat{S}_t^\alpha, \Delta W_t)$ which approximates $S_{t+\Delta t}^\alpha$. The supervised GAN architecture is resumed in Figure 1, the GAN is composed by two neural networks, first the Generator G_θ is parameterized by $\theta \in \mathbb{R}^p$, for $p \in \mathbb{N}$, that aims to learn to approximate the trajectory, and a discriminator D_β parameterized by $\beta \in \mathbb{R}^q$, for $q \in \mathbb{N}$, that aims to discriminate the value of output of the generator compared to real value.

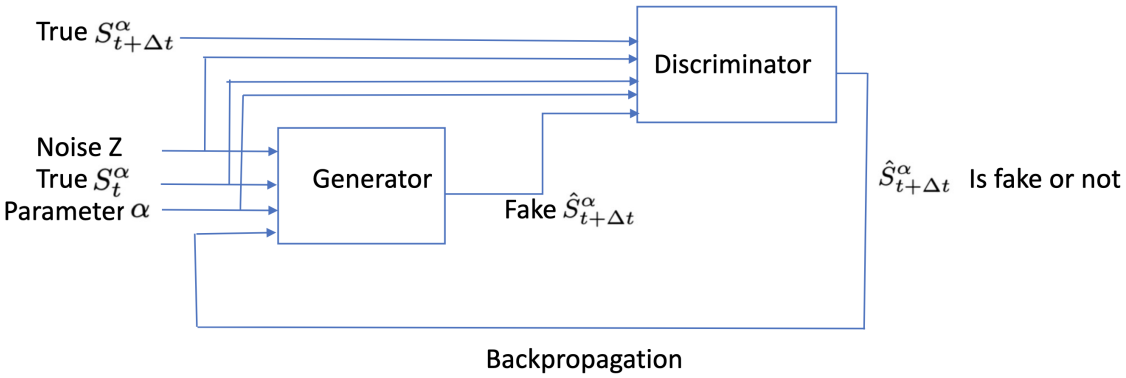

Figure 1: Supervised GAN

The idea of this architecture is given when we use a loss function such as the Binary cross entropy

$$V(G_\theta, D_\beta) = \mathbb{E}_{X\sim P^*}[\log(D_\beta(X))] + \mathbb{E}_{G_\theta(Y)\sim P_\theta}[\log(1 - D_\beta \circ G_\theta(Y))] \tag{1}$$

and hence we aim to minimize, on θ, the worst discrimination on β, which gives rise to the following inf sup game :

$$\inf_\theta \sup_\beta V(G_\theta, D_\beta) \tag{2}$$

We can also use another loss function, the Wasserstein-1 distance defined by (using the Kantorovich duality):

$$W(P_\theta, P^*) = \sup_{||f||_L \leq 1} \mathbb{E}_{X\sim P^*}[(f(X))] - \mathbb{E}_{G_\theta(Y)\sim P_\theta}[f \circ G_\theta(Y)] \tag{3}$$

that also gives rise to an inf sup formulation:

$$\inf_\theta W(P_\theta, P^*) \tag{4}$$

where the discriminator will play the role to approximate the Wasserstein distance.

Note that as we are learning by conditioning on $(S_t^\alpha, \Delta W_t)$ it means that we are learning the conditional law of $\hat{S}^\alpha_{t+\Delta t}|_{S_t^\alpha}$ and the loss function remain unnchanged:

$$V(G_\theta, D_\beta) = \mathbb{E}_{X|_y \sim P^*}[\log(D_\beta(X|_y))] + \mathbb{E}_{G_\theta(Y)|_y \sim P_\theta}[\log(1 - D_\beta \circ (G_\theta(Y)|_y)] \tag{5}$$

Section 3 in [2] explains how if we supervise our GAN with the Brownian increment we learn not only the conditional distribution but also the map that for each Brownian increment gives the value $\hat{S}^\alpha_{t+\Delta t}$ knowing the value $\hat{S}^\alpha_t$, and hence the strong approximation, while the unsupervised GAN gives a weak approximation,meaning that it only approximates the conditional distribution.

2 Application to the CIR model

All numerical tests in the following are done on the CIR model defined by the following SDE:

$$dS_t^\alpha = \kappa(\bar{S} - S_t^\alpha)dt + \sigma\sqrt{S_t^\alpha}dW_t, \quad \forall t \in [0, T], \tag{6}$$

where,

- $\bar{S}$, the mean reversion,
- κ, speed of return to the mean,
- σ the volatility,

where the parameter α is chosen to be the couple $\alpha = (\kappa, \bar{S})$ belonging to $\mathcal{P} = ([0.1, 1] \times [0.1, 1])$.

2.1 Exact simulation as a reference method

We train our generator on a trajectories simulated using the exact simulation described in [1], that we resume here:
The distribution of S_t^α given S_u^α for $u \leq t$ is a noncentral chi-squared distribution up to a scale factor, and is given by,

$$S_t^\alpha \sim \frac{\sigma^2(1 - e^{-k(t-u)})}{4k}\chi_d^2\left(4k\frac{e^{-k(t-u)}}{\sigma^2(1 - e^{-k(t-u)})}S_u^\alpha\right), \quad u \leq t \tag{7}$$

Where $\chi_d^2(\lambda)$ is the noncentral chi-squared random variable with d the degrees of freedom such that $d = \frac{4\bar{S}k}{\sigma^2}$ and noncentrality parameter $\lambda = 4k\frac{e^{-k(t-u)}}{\sigma^2(1-e^{-k(t-u)})}S_u^\alpha$.
Hence, knowing S_u^α we can sample exactly S_t^α from the noncentral chi-squared distribution. For $d > 1$ it is proven in [1] that to generate a noncentral chi-squared random variable with degrees of freedom d and noncentrality parameter λ we can generate a chi-squared random variable with degrees of freedom $d-1$ and an independent standard normal random variable Z and set,

$$\chi_d^2(\lambda) = (Z + \sqrt{\lambda})^2 + \chi_{d-1}^2. \tag{8}$$

We use this formulation in our code to simulate our real trajectories which means that we are under the weak Feller condition.
For the case $d > 0$ another expression is given and a project is under work to replace the exact simulation with a GAN.

3 Data pre-processing

Neural Networks are sensitive to standardised data that improves the convergence in the training phase i.e with zero moment and unit variance. As we don't know the expectance and the variance of our process we cannot simply standardise the dataset and invert it in the inference phase. The CIR model presents the mean reversion term $\bar{S}$ assumed to be known that we can use it to shift and scale the distribution by:

$$R^{\alpha}_{t+\Delta t}|_{S^{\alpha}_t} = \frac{S^{\alpha}_{t+\Delta t} - \bar{S}^{\alpha}}{\bar{S}^{\alpha}}, \tag{9}$$

that is approximated using the supervised GAN. In the inference phase we normally just invert (9) to get $(R^{\alpha}_{t+\Delta t}|_{S^{\alpha}_t} + 1)\bar{S}^{\alpha}$, since our process is in $\mathbb{R}^+$ we take the absolute value, $|(R^{\alpha}_{t+\Delta t}|_{S^{\alpha}_t} + 1)\bar{S}^{\alpha}|$. Note that if we change the model, we need just to adapt the code with an appropriate standardization.

3.1 Dataset

In the figure (2) is resumed the dataset architecture, we run M trajectories on N time steps between $[0, T]$ using the exact simulation [1] for many parameters $\alpha \in \mathcal{P}$, hence each line corresponds to the current value of the process, the Brownian increment and the value of the parameter α to get the $R^{\alpha}_{t+\Delta t}|_{S^{\alpha}_t}$ value.
Let us mention that this architecture is scalable, meaning that we can add as many inputs as we need, as the time step or other parameters of the considered model.

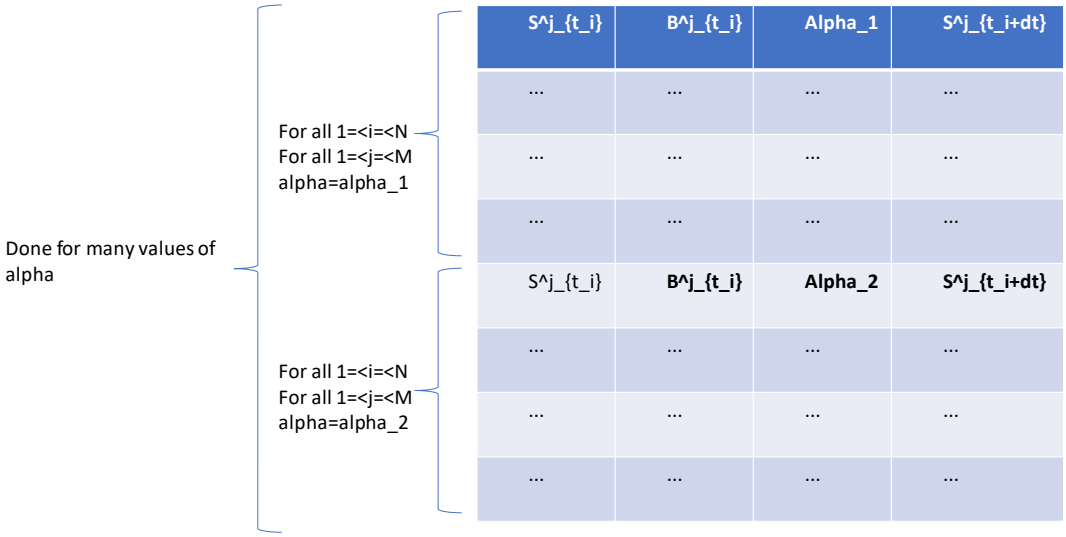

Figure 2: Training data architecture

4 Neural Network architecture

We use two neural networks one for the generator and one for the discriminator, both are similar except that for the discriminator we add at the end the sigmoid function to force

the output to be in $[0, 1]$. Hence our neural network is composed by 4 hidden layers with 200 neurones per layer. After each layer except the output layer, the LeakyRelu function $(f(x) = max(x, 0) + a \cdot min(x, 0))$ is used as an activation function, with a parameter slope $(a = 0.1)$. We don't apply an activation function after the output layer for the generator.
We use the Adam optimizer for both NNs, with a parameter learning rate $l_r = 10^{-4}$ and with $\beta_1 = 0.5$ and $\beta_2 = 0.99$ the factors for gradients and second moments of gradients.

5 Numerical results

Training phase In the training phase we run our trajectories on $[0, T] = [0, 10]$ with time step Δt equal $\Delta t = 10^{-1}$, the number of trajectories used is equal to $M = 5000$ and the parameter space $\mathcal{P} = ([0.1, 1] \times [0.1, 1])$ is discretized with a 10×10 grid from which we obtain 100 samples.
We set a number of batche size equal to 1000 and a number of epochs equal to 50. Inference phase The inference phase is done on the same parameters as in the training phase but with new samples.
In figure (3) we present different empirical distribution obtained using the GAN (in black) and we compare them to the empirical distribution obtained through the exact simulation (in orange), this is done for different parameters of $\alpha = \{(0, 4; 0, 6), (0, 2; 0, 6), (0, 8; 0, 8)\}$, we can observe that we reproduce efficiently the reference distribution for any $\alpha \in \mathcal{P}$, and we can also observe that we reproduce well the heavy tail.

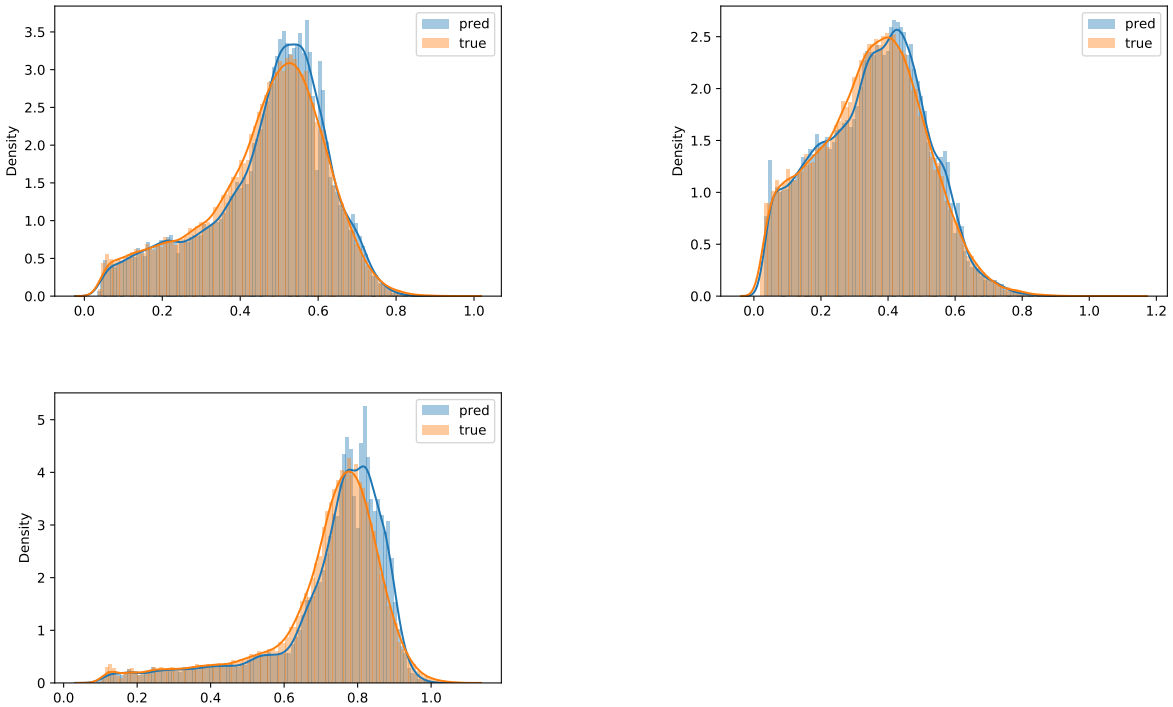

Figure 3: Emperical histogram of both predicted and exact simulations, from the left to the right with different $(\kappa; \bar{S})$ equal to $(0, 4; 0, 6), (0, 2; 0, 6), (0, 8; 0, 8)$ with time step $\Delta t = 10^{-1}$ equivalent to one month.

In figure (4) we present different trajectories obtained using the GAN (in black) an we compare them to the trajectories obtained with the exact simulation (in red), this is done

for different parameters of $\alpha = \{(0,4;0,6),(0,2;0,6),(0,8;0,8)\}$, we can observe that we reproduce efficiently the reference trajectory for any $\alpha \in \mathcal{P}$ and this is done for a time step $\Delta t = 10^{-1}$ equivalent to one month.

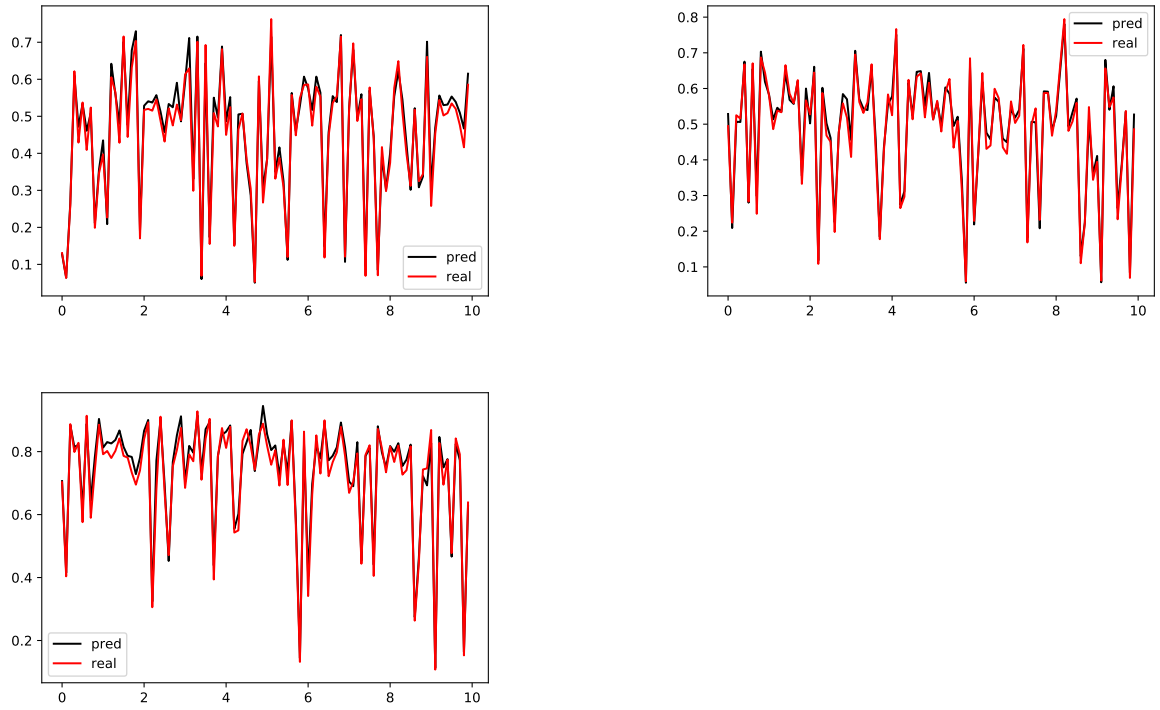

Figure 4: Trajectories, from the left to the right with different $(\kappa;\bar{S})$ equal to $(0,4;0,6),(0,2;0,6),(0,8;0,8)$ with time step $\Delta t = 10^{-1}$ equivalent to one month.

References

[1] Mark Broadie and O. Kaya. *Exact Simulation of Stochastic Volatility and Other Affine Jump Diffusion Processes*. arXiv, 2006.

[2] Jorino van Rhijn, Cornelis W. Oosterlee, Lech A. Grzelak, and Shuaiqiang Liu. Monte Carlo Simulation of SDEs using GANs. *Japan Journal of Industrial and Applied Mathematics*, pages 1–32, 2022.