Generating Asset Returns with Quant GANs

Machine Learning in Finance Workshop 2020 @ Columbia University, New York

Magnus Wiese

August 7, 2020

TU Kaiserslautern

Disclaimer

Table of contents

Introduction

Applying RL to financial datasets

Financial data on a daily scale is scarce and non-stationary. To apply deep reinforcement learning (RL) to finance training environments, for instance through market generators, need to be created as otherwise the RL agent may try to exploit spurious characteristics in the data instead of learning the underlying properties governing the data.

Potential RL applications include, but are not limited to:

- Hedging of derivatives [2]
- Portfolio optimization [6]
- Risk management

Synthetic market generators: related work

Recently, the topic of data-driven market simulation has attracted the attention from industry and academia:

- [7, 9, 10, 12] use the classic GAN to generate macroeconomic time series, stock returns, limit-order book data and equity option markets.
- SigCWGANs [11] learn the joint-distribution by leveraging signatures and calibrate a deterministic discriminator before training.
- [3] paired variational autoencoders (VAEs) with log-signatures to generate stock returns.
- COT-GANs [14] use causal-optimal transport and Sinkhorn divergences.

Quant GANs

With Quant GANs we aim to approximate a synthetic market generator for asset returns in an unsupervised fashion by leveraging a neural network-based discriminator as in the classic GAN setup.

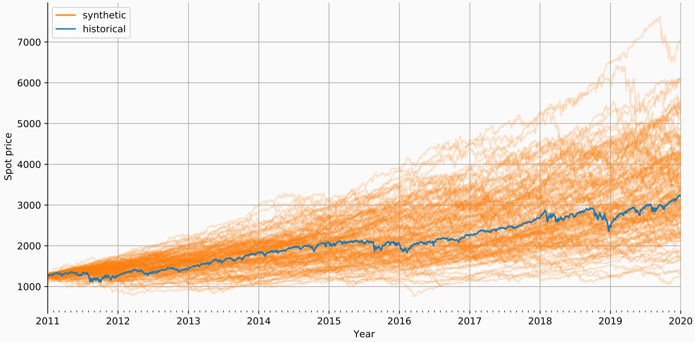

Figure 1: S&P 500 (blue) and 128 synthetic Quant GAN paths (orange).

A review of the empirical properties of asset returns

Validating the Quant GANs performance via stylized facts

Cont [5] identifies and summarises some of the most consistent empirical findings characterising asset returns which he coins as *stylized facts*.

Stylized facts, thus, are characteristic features we would like our synthetic market generator to have.

In what follows we illustrate a subset of the most prominent stylized facts by using the historical log-returns $(x_t)_{t=1}^{T}$ of the S&P 500.

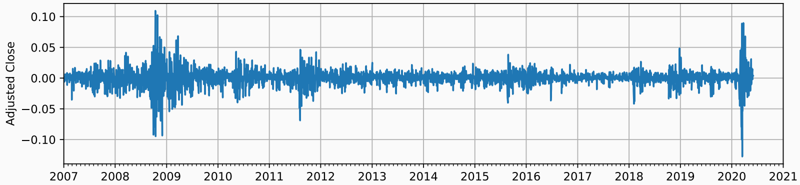

Figure 2: Adjusted close price of the S&P 500 from 3 Jan 2007 - 1 June 2020.

Stylized facts: kurtosis

Positive excess kurtosis: empirical distribution is *peaker* than the Gaussian, i.e.

$$\hat{\kappa}_h := \frac{1}{T}\sum_{t=1}^{T}\left(\frac{x_t - \bar{x}}{\hat{\sigma}(x)}\right)^4 - 3$$

where $\bar{x}$ is the average and $\hat{\sigma}(x)$ the standard deviation of $(x_t)_{t=1}^{T}$, is greater zero.

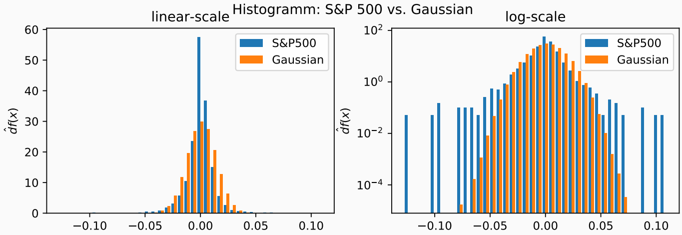

Figure 3: Unconditional distribution of the S&P 500 on the linear- (left) and log-scale. $\hat{\kappa}_h = 13.15$.

Stylized facts: fat tails

Fat-tails: tails resemble power law decay, denoting the empirical CDF by $\hat{F}_h : \mathbb{R} \to [0, 1]$ we observe

$$\ln(1 - \hat{F}_h(x)) \approx -\alpha \ln(x) \text{ for } x \gg 0.$$

and

$$\ln \hat{F}_h(x) \approx \alpha \ln(x) \text{ for } x \ll 0.$$

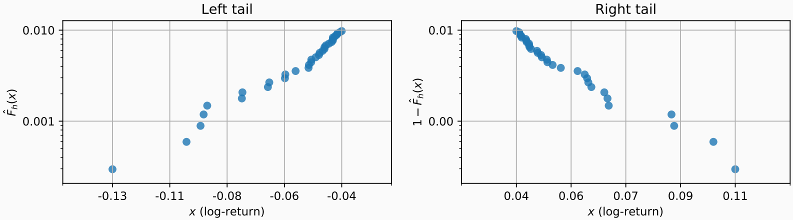

Figure 4: Left and right tail of the S&P 500. The x-axis resembles the magnitude of the log-return, and the y-axis the empirical CDF, both on the log-scale.

Stylized facts: serial autocorrelation and volatility clusters

Define the empirical auto-covariance function of $(x_t)_{t=1}^T$ as

$$\hat{\rho}_h^x(k) = \frac{1}{T-k}\sum_{t=1}^{T-k}(x_t - \bar{x})(x_{t+k} - \bar{x})$$

where $\bar{x}$ denotes the mean of $(x_t)_{t=1}^T$.[1]

No serial correlations: linear autocorrelations $\hat{\rho}_h^x(k)/\hat{\rho}_h^x(0)$, $k = 1, \dots, T$ are often insignificant.

Presence of volatility clusters: phases of low and high absolute log-returns tend to clustes, i.e. $\hat{\rho}_h^{|x|}(k)/\hat{\rho}_h^{|x|}(0)$ tends to be positive and decays *slowly*.

[1] In our notation we use h in the subscript to indicate that we are computing the estimator with respect to historical log-returns.

Stylized facts: leverage effects

Leverage effects: anti-correlation between log-returns and volatility.

$$\hat{l}_h(k) = \frac{1}{T-k} \sum_{t=1}^{T-k} \frac{(|x_t| - \overline{|x|})(x_{t+k} - \bar{x})}{\hat{\sigma}(|x|)\hat{\sigma}(x)} < 0$$

for small h.

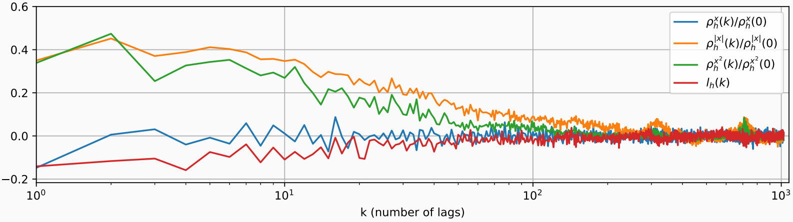

Figure 5: Autocorrelation functions of absolute, squared and serial log-returns and the leverage effect function of the S&P 500.

S&P 500: volatility clusters

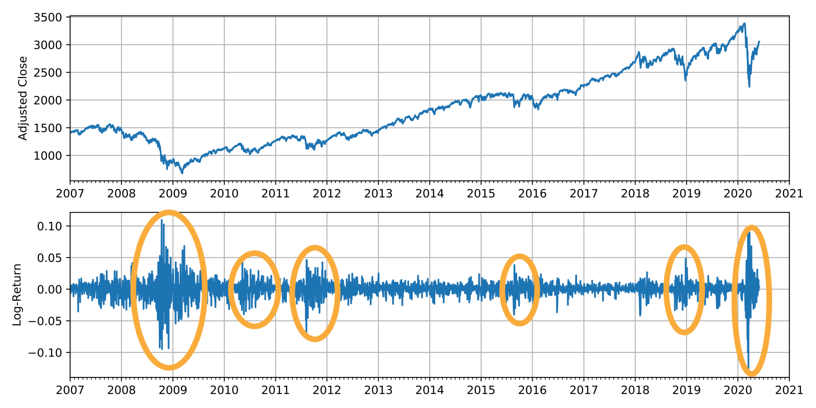

Figure 6: Spot price and log-returns of S&P 500. The orange circles highlight volatility clusters. Note that these clusters generally occur after a larger drawdown, justifying the leverage effect.

The model: Quant GANs

Problem formulation

Consider an unknown distribution ν and a sequence of (multivariate) log-returns $(x_t)_{t=1}^T$ in $\mathcal{X} \subseteq \mathbb{R}^d$ sampled from ν. Furthermore, assume that ν is supported on $\mathcal{Z} \subseteq \mathbb{R}^l$ for $d \gg l$.

In generative modelling the aim is to construct a generator function $G : \Theta \times \mathcal{Z} \to \mathcal{X}$ such that we can sample from an easy to sample distribution μ^z and $G^\theta_\# \mu^z, \theta \in \Theta$ and ν are *close* with respect to some metric. The challenge here is to find a suitable cost function such that the functions parameters $\theta \in \Theta$ are approximated and the distance between the two distributions is minimized.

Generative adversarial networks (GANs)

In GANs a neural network called the discriminator is utilized as this cost function. The discriminator $D : \mathcal{X} \to [0, 1]$ has the objective to discriminate whether a sample $x \in \mathcal{X}$ is drawn from the unknown data distribution ν or generated distribution $G^{\theta}_{\#}\mu^{z}$, whereas the generator $G : \mathcal{Z} \to \mathcal{X}$ has the objective to *fool* the discriminator.

The GAN's objective is formulated as a two-player min-max game

$$\inf_{G:\mathbb{R}^l \to \mathbb{R}^d} \sup_{D:\mathbb{R}^d \to [0,1]} \mathbb{E}_{\nu}\left[\ln D \circ X\right] + \mathbb{E}_{\mu^z}\left[\ln\left(1 - D \circ G \circ Z\right)\right]$$

Note the discriminator's objective is to minimize the binary cross-entropy.

The model: Quant GAN (1)

The Quant GAN is inspired from the recent success of generative adversarial networks (GANs) [8] in the image domain (see e.g. [1]).

To model temporal dynamics and capture longer-ranging dependencies, the functions D and G are constructed through *temporal convolutional networks* (*TCNs*). TCNs are convolutional neural networks that utilize dilated caual convolutions to capture longer-ranging dependencies.

The model: Quant GAN (2)

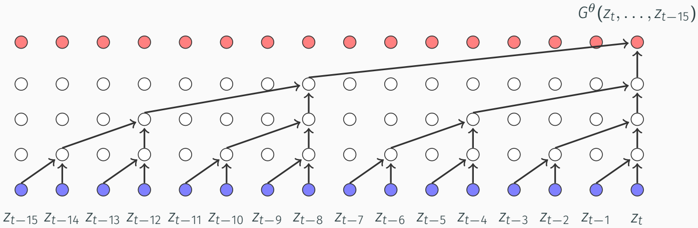

Figure 7: Depiction of a Quant GAN generator with 4 hidden layers and a dilation factor of 2. The generator's receptive field size is 16. (For details refer to [13] or to the code `github.com/magnumw/QuantGANs`.)

Model advantages and disadvantages

Advantages:

When compared to RNNs:

- No backpropagation through time required.
- Stable and parallelizable generation.
- More stable gradients due to the use of convolutions.

Disadvantages:

- Unconditional generator - no initial conditions can be inferred.
- Stochastic discriminator and no criterion when to stop training.[2]
- Not parsimonious.
- Construction of the generator causes synthetic log-returns to be independent after a finite number of lags.

[2] See for example [11] for a deterministic discriminator which is approximated before training and also [3] which pair log-signatures with VAEs.

Numerical results

Test metrics: distributional

To assess the convergence and observe if the Quant GAN learns stylized facts we compute a number of test metrics of the unconditional distribution during training.

Denote by $\hat{df}_{\cdot}, \hat{\kappa}_{\cdot}, \hat{s}_{\cdot}$ for $\cdot \in \{h, G\}$ the unconditional distribution, kurtosis and skew of historical and generated data respectively. We track the following test metrics:

Density metric:

$$|\hat{df}_h - \hat{df}_G|_1$$

Absolute difference of kurtosis:

$$|\hat{\kappa}_h - \hat{\kappa}_G|$$

Absolute difference of skew:

$$|\hat{s}_h - \hat{s}_G|$$

Test metrics: dependence

Furthermore, we compute different test metrics to assess the dependence structure of synthetic data.

Serial ACF metric:

$$\frac{1}{64}\sum_{h=1}^{64}|\hat{\rho}_h^x(h)/\hat{\rho}_h^x(0)-\hat{\rho}_G^x(h)/\hat{\rho}_G^x(0)|$$

Absolute difference of the ACF of absolute log-returns:

$$\frac{1}{64}\sum_{h=1}^{64}|\hat{\rho}_h^{|x|}(h)/\hat{\rho}_h^{|x|}(0)-\hat{\rho}_G^{|x|}(h)/\hat{\rho}_G^{|x|}(0)|$$

Absolute difference of the leverage effect function:

$$\frac{1}{16}\sum_{h=1}^{16}|\hat{l}_h(h)-\hat{l}_G(h)|$$

Training settings

Hyperparameters	
Batch size	512
Learning rates	1e-4 / 3e-4 (G / D)
Optimizer	Adam
Latent dimension	3
Lookback	127 / 127 (G / D)
Total number of parameters	85424 / 83824 (G / D)
Hardware	
GPU	RTX 2070
Required memory	2663MiB
Total training time	1:49 minute
Dataset	
Asset	S&P 500 closing price
Time horizon	3 Jan 2011 - 31 Dec 2019

Can Quant GANs learn stylized facts?

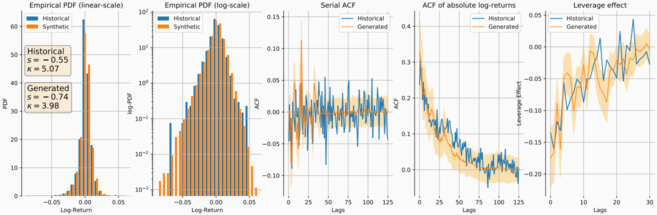

Figure 8: From left to right: comparison of the empirical PDFs on the linear- and log-scale, serial ACF, ACF of absolute log-returns and the leverage effect function. The colors blue and orange indicate the estimators computed from historical and synthetic data respectively.

Synthetic paths

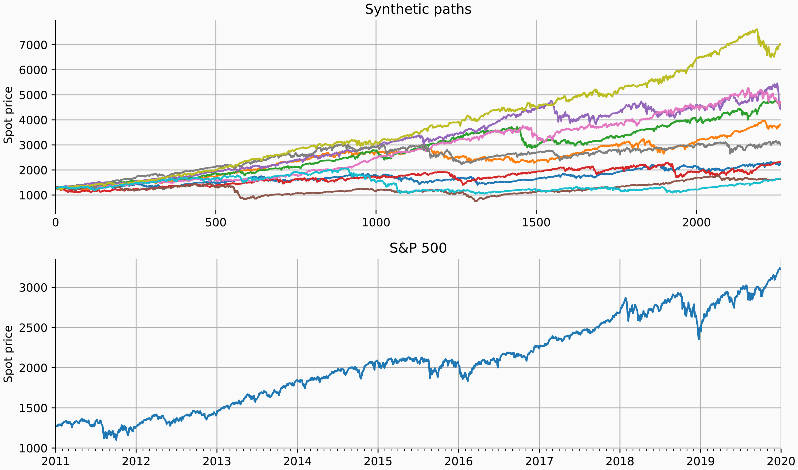

Figure 9: Ten synthetic paths (top) and the historical S&P 500 path (bottom).

How well does the model converge?

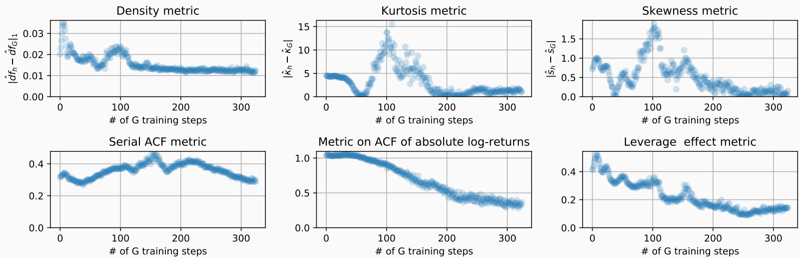

Figure 10: Development of the test metrics during training. Observe that most of the test metrics, except the serial ACF metric, follow a downward trend. This confirms that the discriminator of the Quant GAN is able to detect stylized facts.

Concluding remarks

Concluding remarks

- Numerical results demonstrate that Quant GANs can learn stylized facts in an unsupervised fashion.
- However, the unconditional structure of the model and the independence assumption (see [13, Remark 5.3]) may limit the models utility for further applications.
- Although Quant GANs could be trained with larger receptive fields in practice optimisation becomes unstable, leaving future research.
- Furthermore, more elaborate tests need to be developed to test the proximity of the generated distribution to the historical, see e.g. [3, 4].

References i

A. Brock, J. Donahue, and K. Simonyan.
Large scale gan training for high fidelity natural image synthesis.
In *International Conference on Learning Representations*, 2018.

H. Buehler, L. Gonon, J. Teichmann, and B. Wood.
Deep hedging.
Quantitative Finance, pages 1–21, 2019.

H. Bühler, B. Horvath, T. Lyons, I. P. Arribas, and B. Wood.
A data-driven market simulator for small data environments.
Technical report, arXiv.org, 2020.

References ii

I. Chevyrev and H. Oberhauser.
Signature moments to characterize laws of stochastic processes.
arXiv preprint arXiv:1810.10971, 2018.

R. Cont.
Empirical properties of asset returns: stylized facts and statistical issues.
2001.

A. Filos.
Reinforcement learning for portfolio management.
arXiv preprint arXiv:1909.09571, 2019.

R. Fu, J. Chen, S. Zeng, Y. Zhuang, and A. Sudjianto.
Time series simulation by conditional generative adversarial net.
arXiv preprint arXiv:1904.11419, 2019.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio.
Generative adversarial nets.
In *Advances in neural information processing systems*, pages 2672–2680, 2014.

A. Koshiyama, N. Firoozye, and P. Treleaven.
Generative adversarial networks for financial trading strategies fine-tuning and combination.
arXiv preprint arXiv:1901.01751, 2019.

J. Li, X. Wang, Y. Lin, A. Sinha, and M. P. Wellman.
Generating realistic stock market order streams.
2018.

H. Ni, L. Szpruch, M. Wiese, S. Liao, and B. Xiao.
Conditional sig-wasserstein GANs for time series generation, 2020.

M. Wiese, L. Bai, B. Wood, and H. Buehler.
Deep hedging: Learning to simulate equity option markets.
SSRN Electronic Journal, 2019.

M. Wiese, R. Knobloch, R. Korn, and P. Kretschmer.
Quant GANs: deep generation of financial time series.
Quantitative Finance, pages 1–22, 2020.

T. Xu, L. K. Wenliang, M. Munn, and B. Acciaio.
COT-GAN: Generating sequential data via causal optimal transport, 2020.