UNIVERSITY OF OXFORD

Mathematical Institute

Two applications of Deep Learning in Finance

2. Learning to simulate

Rama Cont

Oxford Mathematics

Based on

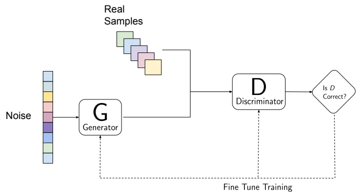
Cont, Rama, Cucuringu, Mihai, Xu, Renyuan
and Zhang, Chao (2022)
Tail-GAN: Learning to Simulate Tail Risk Scenarios.
http://dx.doi.org/10.2139/ssrn.3812973

Oxford
Mathematics

Scenario simulation

- Key ingredient of quantitative risk management.
- The estimation of loss distributions for dynamic portfolios requires the simulation of scenarios representing realistic joint dynamics of their components, with particular importance devoted to the simulation of *tail risk* scenarios.
- Commonly used parametric models have been successful in applications involving a small number of assets, but are not easy to scale to large or heterogeneous portfolios involving multiple asset classes.
- In some cases, such as intraday/ HF scenario simulation, good parametric models are simply not available.
- Problem: **sampling from a high dimensional unknown distribution in the space of scenarios.**

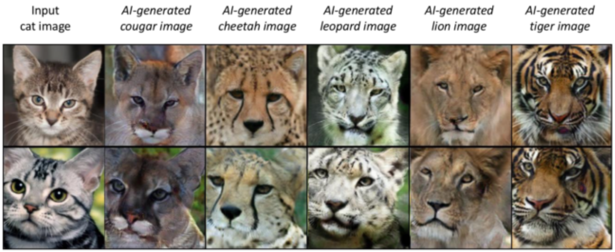
Generative Adversarial Networks (GAN)



- Generative models have been increasingly deployed for generating 'synthetic data' from high-dimensional probability distributions.
- Idea: train a pair of neural networks in 'adversarial mode' : one learns to generate samples from a dataset, while the other learns to 'discriminate' whether a given sample is drawn from the target distribution.
- Training algorithm attempts to solve a maxmin problem: given data with distribution $\mathbb{P}_r$ and an input noise distribution $\mathbb{P}_Z$

$$\min_G \max_D L(D,G) = \mathbb{E}_{X\sim\mathbb{P}_r}[\log(D(X)] + \mathbb{E}_{Z\sim\mathbb{P}_Z}[\log(1-D(G(Z)))]$$

Generative adversarial networks (GANs)


Input cat image
AI-generated cougar image
AI-generated cheetah image
AI-generated leopard image
AI-generated lion image
AI-generated tiger image

- Impressive applications in image (Goodfellow et al 2014, Radford et al 2015), audio (Oord et al 2016, Donahue et al 2018) and text generation (Fedus et al 2018, Zhang et al 2017).
- End result is a 'black box' simulator which generates samples (text, images, video, audio).
- Famous example: DeepFakes.

Generative models for scenario simulation

- GANs have been used for building 'market simulators' in finance (Wiese et al 2020, Koshiyama et al 2020, Buehler et al 2020, ...) by training a GAN on single or multi-asset return time series.
- Training criterion is cross-entropy or some probability metric (Wasserstein).
- Final result is a 'black box' simulator which generates market scenarios.

Reference	Model category	Architecture	Loss Function	Data
Quant-GAN [?]	GAN	TCN	KL divergence	Returns
[?]	VAE	NN	l_2 norm	Signatures of paths
Time-GAN [?]	GAN + Supervised Learning	RNN	Supervised, Unsupervised, Reconstruction loss	Time Series
[?, ?]	CGAN	NN	KL divergence	Returns
SigCWGAN [?]	CGAN	Residual NN	l_2 norm	Signatures
[?]	CGAN	LSTM	KL divergence	Limit order books
AIQN [?]	Supervised Learning	PixelCNN	Quantile divergence	Images
ExGAN [?]	CGAN	CNN	KL divergence + Supervised loss	Rainfall images

Generative models for scenario simulation

- GANs have been used for building 'market simulators' in finance (Wiese et al 2020, Koshiyama et al 2020, Buehler et al 2020, ...) by training a GAN on single or multi-asset return time series.
- Training criterion is cross-entropy or some probability metric (Wasserstein).
- Final result is a 'black box' simulator which generates market scenarios.
- **Model Validation: how do we know if generated scenarios are realistic/ adequate? Adequate for what?**
- Unlike images or text, user perception cannot be used for validation.
- Risk management applications involve the calculation of loss distributions, especially tail risk, for trading strategies, so the answer lies in the loss distributions arising from the generated scenarios!

TailGAN: a data-driven approach to scenario simulation tailored to risk management applications

We propose a data-driven approach for the simulation of realistic multi-asset scenarios with a focus on the accurate estimation of tail risk for a given class of static and dynamic portfolios selected by the user.

1. Target: design a data driven scenario generator which correctly preserves tail risk features of a set of benchmark portfolios/ dynamic trading strategies.
2. Idea: exploit the **joint elicitability property** of Value-at-Risk (VaR) and Expected Shortfall (ES) to design a GAN learning algorithm which learns to simulate scenarios which yield consistent estimators for VaR and and Expected Shortfall for these portfolios.
3. We demonstrate the accuracy and scalability of our method via extensive simulations using synthetic and market data.
4. Our results show that, in contrast to other data-driven scenario generators, TailGAN scenarios correctly capture the tail risk for both static and dynamic portfolios.

Outline

1 Introduction

2 Score functions for Tail Risk Measures

3 TAIL-GAN: Scenario Generation for Tail Risk Estimation

4 Model validation

5 Multiasset intraday scenario simulation

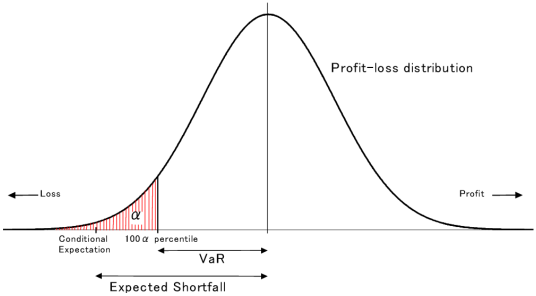
6 Conclusion

Tail risk measures

Value-at-Risk (VaR) and Expected Shortfall (ES)

Given a one-dimensional distribution μ with a finite mean and a confidence level $0 < \alpha < 1$, the α-VaR and α-ES are defined as

$$\begin{aligned} \mathrm{VaR}_\alpha(\mu) &:= \inf\left\{x \in \mathbb{R} : \mu(X \le x) \ge \alpha\right\} \\ \mathrm{ES}_\alpha(\mu) &:= \frac{1}{\alpha}\int_0^\alpha \mathrm{VaR}_\beta(\mu)\mathrm{d}\beta \end{aligned}$$



Elicitability

Definition (Gneiting)

A functional T on probability distributions is elicitable if there is a score function $S(x, y)$ such that $T(\mu) = \arg\min_x \int S(x, y)\mu(dy)$ for some distribution μ.
A score function S is called strictly consistent for T if $T(\mu) = \arg\min_x \int S(x, y)\mu(dy)$ is the unique minimizer.

Examples

$T(\mu)$	$S(x, y)$
mean	$(x-y)^2$
median	$\|x-y\|$
α-quantile	$(1_{\{y\leq x\}} - \alpha)(x-y)$

Elicitability of VaR and Expected Shortfall

Fissler et al (2015) show that $(\mathrm{VaR}_\alpha(\mu), \mathrm{ES}_\alpha(\mu))$ is jointly elicitable,:

Theorem (Fissler et al 2016, Theorem 5.2,)

If H_2 is strictly convex and $R(v,e) := \frac{1}{\alpha} v H_2'(e) + H_1(v)$ is strictly increasing in v, the score function

$$\begin{aligned} S_\alpha(v,e,x) &= (1_{\{x\le v\}} - \alpha)(H_1(v) - H_1(x)) \\ &+ \frac{1}{\alpha} H_2'(e) 1_{\{x \le v\}}(v-x) + H_2'(e)(e-v) - H_2(e). \end{aligned} \quad (1)$$

is strictly consistent for $(\mathrm{VaR}_\alpha(\mu), \mathrm{ES}_\alpha(\mu))$:

$$(VaR_\alpha(\mu), ES_\alpha(\mu)) = \arg\min_{(v,e)\in\mathbb{R}^2} \mathbb{E}_\mu[S_\alpha(v,e,X)]. \quad (2)$$

Acerbi & Szekelyi propose the choice: $H_1(v) = -\frac{W_\alpha}{2} v^2$, $H_2(e) = \frac{\alpha}{2} e^2$.
where $\mathrm{ES}_\alpha(\mu) \ge W_\alpha \mathrm{VaR}_\alpha(\mu)$

Score functions for VaR and Expected Shortfall

These resutls can be used to design an objective function whose minimization leads to **learning the tail risk measures of the data**: if we use as objective function in our learning algorithm

$$\frac{1}{N}\sum_{i=1}^{N} S_\alpha(v, e, x_i) \tag{3}$$

where x_i are samples drawn from a loss distribution μ then for a large enough sample minimizing this objective function over (v, e) leads to

$$(v^*, e^*) \simeq (\mathrm{VaR}_\alpha(\mu), \mathrm{ES}_\alpha(\mu))$$

Outline

1. Introduction
2. Score functions for Tail Risk Measures
3. TAIL-GAN: Scenario Generation for Tail Risk Estimation
4. Model validation
5. Multiasset intraday scenario simulation
6. Conclusion

TAIL-GAN: Scenario Generation for Tail Risk Estimation

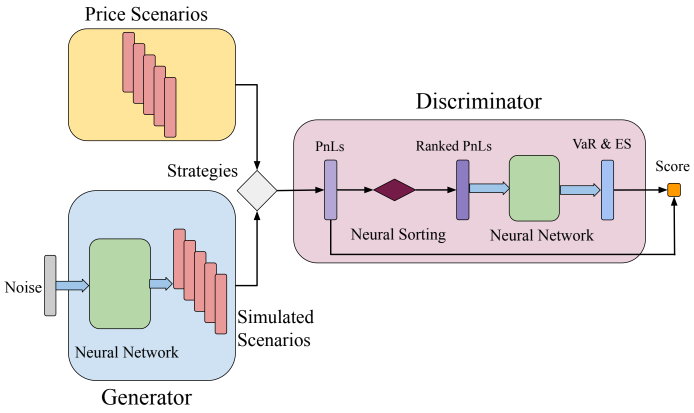


Figure 1: Architecture of TAIL-GAN. The (blue) thick arrows represent calculations with learnable parameters, and the (black) thin arrows represent calculations with fixed parameters.

Evaluation of the Generator

- The user specifies a set of **benchmark trading strategies** which may be static or dynamic. Dynamic strategies may include mean-reversion strategies, momentum strategies, single or multiasset statistical arbitrage strategies etc. .
- Data is split into sequence (trading periods) of duration T (risk horizon)
- The associated profit (PnL) of the strategy is calculated at the end of each trading period
- $\mathbf{\Pi} := (\Pi^1, \cdots, \Pi^K) : \mathbb{R}^{M \times T} \to \mathbb{R}^K$, where Π^k is k-th strategy with the price scenarios $\boldsymbol{p}$ as input and the final PnL as output.

Some notations

- Consider M assets and their price dynamics over a time horizon T
- The user specifies K **benchmark trading strategies** which may be static or dynamic
- Π^k represents the k-th strategy, with the price scenarios $\mathbf{p} \in \mathbb{R}^{M \times T}$ as the input and the final PnL as the output
- $\Pi^k \# \mathbb{P}_r$ denotes the distribution of $\Pi^k(\boldsymbol{p})$, where $\boldsymbol{p} \sim \mathbb{P}_r$

Objective of the Generator

The goal of the generator is to best match the tail risk across the set of K benchmark strategies:
Recall $\Pi^k \# \mathbb{P}_r$ is the distribution of $\Pi^k(\boldsymbol{p})$ and

$$\Big(\mathrm{VaR}_\alpha(\Pi^k \# \mathbb{P}_r), \mathrm{ES}_\alpha(\Pi^k \# \mathbb{P}_r)\Big) = \arg \min_{(v,e)\in\mathbb{R}^2} \mathbb{E}_{\boldsymbol{p}\sim\mathbb{P}_r}[S_\alpha(v, e, \Pi^k(\boldsymbol{p}))]. \quad (4)$$

$\mathbb{E}_{\boldsymbol{p}\sim\mathbb{P}_r}\Big[S_\alpha\Big(\Big(\mathrm{VaR}_\alpha(\Pi^k \# \mathbb{P}_G), \mathrm{ES}_\alpha(\Pi^k \# \mathbb{P}_G)\Big),\ \Pi^k(\boldsymbol{p})\Big)\Big]$ can evaluate the distribution of the simulated price scenario from G via strategy k, where $\Pi^k \# \mathbb{P}_G$ is the distribution of $\Pi^k(G(\boldsymbol{z}))$. The smaller the score, the better the performance of the generator.

$$\min_G \sum_{k=1}^K \mathbb{E}_{\boldsymbol{p}\sim\mathbb{P}_r}\Big[S_\alpha\Big(\Big(\mathrm{VaR}_\alpha(\Pi^k \# \mathbb{P}_G), \mathrm{ES}_\alpha(\Pi^k \# \mathbb{P}_G)\Big),\ \Pi^k(\boldsymbol{p})\Big)\Big]. \quad (5)$$

Theoretical version of the discriminator

- Input: strategy PnL distribution
- Output: two values $\Rightarrow$ aims to provide the correct $(\mathrm{VaR}_\alpha, \mathrm{ES}_\alpha)$ values

Mathematically,

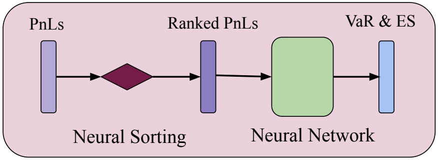
$$D^* \in \arg\min_D \frac{1}{K} \sum_{k=1}^{K} \mathbb{E}_{\boldsymbol{p} \sim \mathbb{P}_r} \left[S_\alpha \left(\overbrace{D(\underbrace{\Pi^k \# \mathbb{P}_r}_{\text{strategy PnL distribution}})}^{\text{VaR and ES prediction from D}} ; \boldsymbol{p} \right) \right],$$

Practical version of the discriminator

- Input: strategy PnL samples ($\approx$ empirical distribution)
- Output: two values $\Rightarrow$ aims to provide the correct $(\mathrm{VaR}_\alpha, \mathrm{ES}_\alpha)$ values

$$\overline{D}^* \in \arg\min_{\overline{D}} \frac{1}{K} \sum_{k=1}^{K} \mathbb{E}_{\boldsymbol{p}\sim\mathbb{P}_r^{\mathrm{emp}}} \left[S_\alpha \left(\overbrace{\overline{D}(\underbrace{\Pi^k(\boldsymbol{p}_j), j \in [n]}_{\text{strategy PnL samples}})}^{\text{VaR and ES prediction from D}} ; \boldsymbol{p} \right) \right],$$

Discriminator of Tail-GAN



Permutation invariance

α-VaR, α-ES of a distribution are **permutation invariant** to the ordering of the samples.
Denote $\boldsymbol{x}^k = (x_1^k, x_2^k, \cdots, x_n^k)^\top$ as a real-valued vector of length n, representing the PnL samples of strategy k. Let $B(\boldsymbol{x}^k) = [B_{i,j}] \in \mathbb{R}^{n\times n}$, where $B_{i,j} = |x_i^k - x_j^k|$. We then define the following permutation matrix $\Gamma(\boldsymbol{x}^k)$ [?, ?]

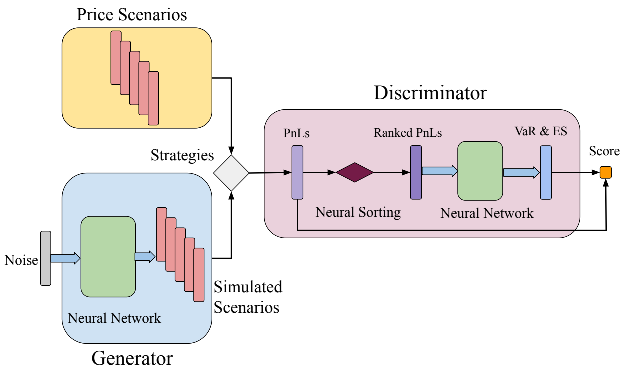
$$\Gamma_{i,j}(\boldsymbol{x}^k) = \begin{cases} 1, & \text{if } j = \arg\max((n+1-2i) - B(\boldsymbol{x}^k)\mathbf{1}), \\ 0, & \text{otherwise}, \end{cases} \tag{6}$$

where $\mathbf{1}$ is the all-ones vector. Then, $\Gamma(\boldsymbol{x}^k)\boldsymbol{x}^k$ provides a ranked vector of $\boldsymbol{x}^k$ ([?, Lemma 1] and [?, Corollary 3]).

Differentiable Neural Sorting

[?] proposes to replace the ARG-MAX operator with SOFT-MAX $\frac{e^{z_i}}{\sum_{j=1}^n e^{z_j}}$, in order to obtain a continuous relaxation $\widehat{\Gamma}^\tau$.

Summary of the architecture



- Our discriminator is **explainable**, which is different from most of other GAN frameworks

From bi-level optimization to max-min game

Our goal is to find a generator G^* and a discriminator D^* via the following bi-level optimization problem

$$G^* \in \arg\min_G \frac{1}{K} \sum_{k=1}^{K} \mathbb{E}_{\boldsymbol{p}\sim\mathbb{P}_r} \left[S_\alpha\Big(D^*\big(\Pi^k \# \mathbb{P}_G\big),\ \Pi^k(\boldsymbol{p}) \Big) \right],$$

with $\mathbb{P}_G := G\#\mathbb{P}_z$, where

$$D^* \in \arg\min_D \frac{1}{K} \sum_{k=1}^{K} \mathbb{E}_{\boldsymbol{p}\sim\mathbb{P}_r} \left[S_\alpha\Big(D\big(\Pi^k \# \mathbb{P}_r\big),\ \Pi^k(\boldsymbol{p}) \Big) \right].$$

$\hookrightarrow$ constrained optimization problem is usually difficult to solve in practice!

Mini-max game in practice

Motivated by the formulation in (1), we consider the following **sample-based min-max game** between the generator network and the discriminator network:

$$\max_{\overline{D}} \min_{\overline{G}} \frac{1}{KN} \sum_{k=1}^{K} \sum_{j=1}^{N} \Big[S_\alpha\Big(\overline{D}\Big(\Pi^k(\mathbf{q}_i); i \in [n]\Big); \boldsymbol{p}_j\Big) - \lambda S_\alpha\Big(\overline{D}\Big(\Pi^k(\mathbf{p}_i); i \in [n]\Big); \boldsymbol{p}_j\Big)\Big],$$

where $\mathbf{p}_i \sim \mathbb{P}_r$, $\mathbf{q}_i \sim \mathbb{P}_{\overline{G}}$, and $\lambda > 0$.

From bi-level optimization to max-min game

Theorem (Cont, Cucuringu, Xu and Zhang (2022))

Under mild conditions, the bi-level optimization problem is equivalent to the following max-min game:

$$\max_D \min_G \frac{1}{K} \sum_{k=1}^{K} \Bigg[\mathbb{E}_{\boldsymbol{p}\sim\mathbb{P}_r} \Big[S_\alpha\Big(D\big(\Pi^k \# \mathbb{P}_G\big),\ \Pi^k(\boldsymbol{p}) \Big) \Big] - \lambda\, \mathbb{E}_{\boldsymbol{p}\sim\mathbb{P}_r} \Big[S_\alpha\Big(D\big(\Pi^k \# \mathbb{P}_r\big),\ \Pi^k(\boldsymbol{p}) \Big) \Big] \Bigg]$$

with any Lagrange multiplier $\lambda > 0$.

- The proof relies on the elicitability property of $(\mathrm{VaR}_\alpha, \mathrm{ES}_\alpha)$ and the consistecy property of S_α.

Objective function

Recall the goal of the generator is to solve the following minimization problem

$$\min_G \sum_{k=1}^{K} \mathbb{E}_{\boldsymbol{p}\sim\mathbb{P}_r}\left[S_\alpha\left(\left(\mathrm{VaR}_\alpha(\Pi^k \# \mathbb{P}_G), \mathrm{ES}_\alpha(\Pi^k \# \mathbb{P}_G)\right),\ \Pi^k(\boldsymbol{p})\right)\right].$$

The min-max game between the generator and the discriminator can be defined as:

$$\max_D \min_G \frac{1}{K} \sum_{k=1}^{K} \quad \Big\{\mathbb{E}_{\boldsymbol{p}\sim\mathbb{P}_r}\left[S_\alpha\left(D(\Pi^k \# \mathbb{P}_G^{(n)}),\ \Pi^k(\boldsymbol{p})\right)\right] - \mathbb{E}_{\boldsymbol{p}\sim\mathbb{P}_r}\left[S_\alpha\left(D(\Pi^k \# \mathbb{P}_r^{(n)}),\ \Pi^k(\boldsymbol{p})\right)\right]\Big\}, \qquad (7)$$

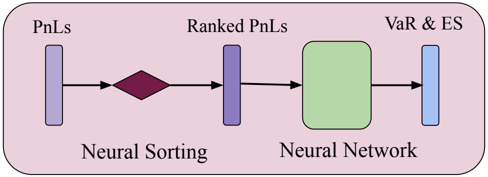
where $\Pi^k \# \mathbb{P}_G^{(n)}$ $((\Pi^k \# \mathbb{P}_r^{(n)}))$ is the empirical distribution of $\Pi^k \# \mathbb{P}_G$ $((\Pi^k \# \mathbb{P}_r))$, respectively.

TAIL-GAN: Scenario Generation for Tail Risk Estimation

$$\max_{D\in\mathcal{D}} \min_{G\in\mathcal{G}} \frac{1}{K}\sum_{k=1}^{K}\Big[s_\alpha\Big(D\big(\Pi^k(\mathbf{q}_i); i\in[n]\big); \Pi^k, \mathbb{P}_{\text{data}}\Big) - \lambda\, s_\alpha\Big(D\big(\Pi^k(\mathbf{p}_i); i\in[n]\big); \Pi^k, \mathbb{P}_{\text{data}}\Big)\Big],$$

- $\mathbf{p}_i \sim \mathbb{P}_{\text{data}}$ and $\mathbf{q}_i \sim \mathbb{P}_G$
- Discriminator $D\,(\cdot)$:
  - Input: n PnL samples
  - Output: two values $\Rightarrow$ aims to provide the correct $(\mathrm{VaR}_\alpha, \mathrm{ES}_\alpha)$ values of the Π^k PnL

Discriminator of Tail-GAN



TAIL-GAN: Scenario Generation for Tail Risk Estimation

$$\max_{D\in\mathcal{D}}\min_{G\in\mathcal{G}} \frac{1}{K}\sum_{k=1}^{K}\Big[s_\alpha\Big(D\big(\Pi^k(\mathbf{q}_i); i\in[n]\big); \Pi^k, \mathbb{P}_{\text{data}}\Big) \\ -\lambda\, s_\alpha\Big(D\big(\Pi^k(\mathbf{p}_i); i\in[n]\big); \Pi^k, \mathbb{P}_{\text{data}}\Big)\Big],$$

can be viewed as a Lagurangian relaxation of the following two-step (or constrained) optimization problem with parameter λ (hard to solve in practice)

$$G^* \in \min_{G\in\mathcal{G}} \frac{1}{K}\sum_{k=1}^{K} s_\alpha\Big(D^*\big(\Pi^k(\mathbf{q}_i); i\in[n]\big);\ \Pi^k, \mathbb{P}_{\text{data}}\Big),$$

where

$$D^* \in \arg\min_{D\in\mathcal{D}} \frac{1}{K}\sum_{k=1}^{K} s_\alpha\Big(D\big(\Pi^k(\mathbf{p}_i); i\in[n]\big);\ \Pi^k, \mathbb{P}_{\text{data}}\Big).$$

Outline

1 Introduction

2 Score functions for Tail Risk Measures

3 TAIL-GAN: Scenario Generation for Tail Risk Estimation

4 Model validation

5 Multiasset intraday scenario simulation

6 Conclusion

Setting

We test the performance of TAIL-GAN on a synthetic data set, for which we can validate the performance by comparing to the true input price scenario distribution.

Data. We simulate five financial instruments with a given correlation structure, various temporal patterns and different tail behaviors in return distributions. The models considered here include (1) Gaussian distribution, (2) AR(1) with autocorrelation $\phi_1 > 0$, (3) AR(1) with autocorrelation $\phi_2 < 0$, (4) GARCH(1, 1) with $t(\nu_1)$ noise and (5) GARCH(1, 1) with $t(\nu_2)$ noise.

Tail-GAN architectures.

- GAN trained with static single-asset portfolios (TAIL-GAN-SSP)
- GAN trained with static multi-asset portfolios (TAIL-GAN-SMP)
- GAN trained with both multi-asset portfolios and dynamic strategies (TAIL-GAN-DS)

Performance Measure

Relative Error.

$$\mathrm{RE}(n) = \frac{1}{2K}\sum_{k=1}^{K}\left(\frac{\left|\mathrm{VaR}_\alpha\left(\Pi^k \# \mathbb{P}_G^{(n)}\right) - \mathrm{VaR}_\alpha\left(\Pi^k \# \mathbb{P}_r\right)\right|}{\left|\mathrm{VaR}_\alpha\left(\Pi^k \# \mathbb{P}_r\right)\right|} + \frac{\left|\mathrm{ES}_\alpha\left(\Pi^k \# \mathbb{P}_G^{(n)}\right) - \mathrm{ES}_\alpha\left(\Pi^k \# \mathbb{P}_r\right)\right|}{\left|\mathrm{ES}_\alpha\left(\Pi^k \# \mathbb{P}_r\right)\right|}\right). \tag{8}$$

Benchmark: Sampling Error.

$$\mathrm{SE}(n) = \frac{1}{2K}\sum_{k=1}^{K}\left(\frac{\left|\mathrm{VaR}_\alpha\left(\Pi^k \# \mathbb{P}_r^{(n)}\right) - \mathrm{VaR}_\alpha\left(\Pi^k \# \mathbb{P}_r\right)\right|}{\left|\mathrm{VaR}_\alpha\left(\Pi^k \# \mathbb{P}_r\right)\right|} + \frac{\left|\mathrm{ES}_\alpha\left(\Pi^k \# \mathbb{P}_r^{(n)}\right) - \mathrm{ES}_\alpha\left(\Pi^k \# \mathbb{P}_r\right)\right|}{\left|\mathrm{ES}_\alpha\left(\Pi^k \# \mathbb{P}_r\right)\right|}\right). \tag{9}$$

Out-of-sample Performance

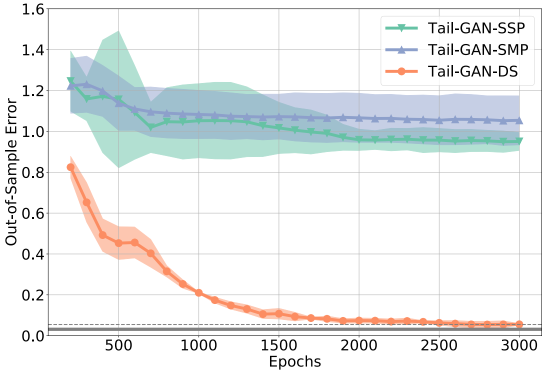
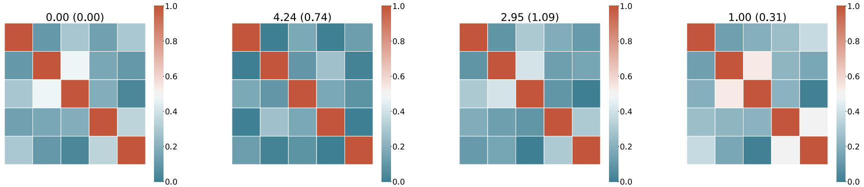


Figure 4: Out-of-sample performance $\mathrm{RE}(1000)$ comparing our simulators, for both static portfolios and dynamic strategies. Grey horizontal line: average simulation error $\mathrm{SE}(1000)$. Dotted line: average simulation error plus one standard deviation. Each experiment is repeated 5 times with different random seeds. The performance is visualized with mean (solid lines) and standard deviation (shaded areas).

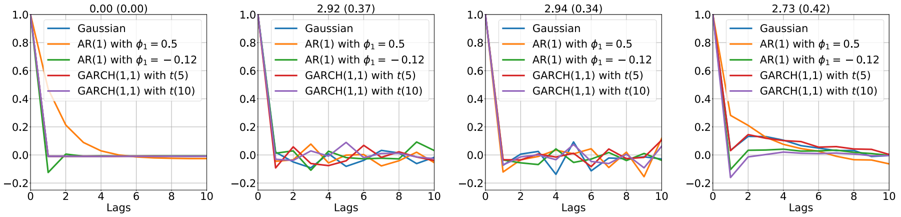
Tail-GANcan learn cross-asset correlations



(a) Input scenario. (b) Tail-GAN-SSP. (c) Tail-GAN-SMP. (d) Tail-GAN-DS.

Figure 5: Correlations of the price increments from different trained GAN models: (1) Tail-GAN-SSP, (2) Tail-GAN-SMP, and (3) Tail-GAN-DS. (Numbers on the top: mean and standard deviation (in parentheses) of the sum of the absolute element-wise difference between the correlation matrices, computed with 10,000 synthetic samples and 10,000 generated samples.)

TAIL-GANcan learn temporal autocorrelations



(a) Input scenario. (b) TAIL-GAN-SSP. (c) TAIL-GAN-SMP. (d) TAIL-GAN-DS.

Figure 6: Auto-correlations of the price increments from different trained GAN models: (1) TAIL-GAN-SSP, (2) TAIL-GAN-SMP, and (3) TAIL-GAN-DS. (Numbers on the top: mean and standard deviation (in parentheses) of the sum of the absolute difference between the auto-correlation coefficients computed with 10,000 synthetic samples and 10,000 generated samples.)

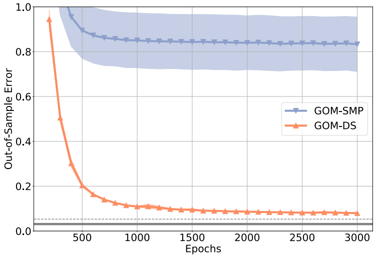
Comparison with Supervised Learning

Recall the objective of generator

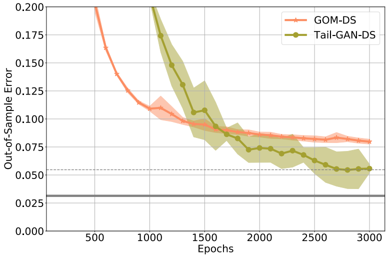
$$\min_G \sum_{k=1}^{K} \mathbb{E}_{\boldsymbol{p}\sim\mathbb{P}_r}\left[S_\alpha\left(\left(\mathrm{VaR}_\alpha(\Pi^k \# \mathbb{P}_G), \mathrm{ES}_\alpha(\Pi^k \# \mathbb{P}_G)\right),\ \Pi^k(\boldsymbol{p})\right)\right].$$

We train a supervised learning model denoted as Generator-Only Model (GOM) to solve the above problem.
Specifically, we follow the procedure of differentiable neural sorting and use $\left(x^k_{(\lfloor \alpha n \rfloor)}, \frac{1}{\lfloor \alpha n \rfloor}\sum_{i=1}^{\lfloor \alpha n \rfloor} x^k_{(i)}\right)$ to estimate the α-VaR and α-ES values, where $x^k_{(n)} \geq \cdots \geq x^k_{(2)} \geq x^k_{(1)}$ are the sorted PnLs of $\boldsymbol{x}^k = (x^k_1, \ldots, x^k_n)$.

Comparison with Supervised Learning: Accuracy



(a) Out-of-sample performance $\mathrm{RE}(1000)$ for GOM-SMP and GOM-DS.



(b) Out-of-sample performance $\mathrm{RE}(1000)$ for GOM-DS and TAIL-GAN-DS on synthetic data.

Figure 7: Performance of GOMs and comparison to TAIL-GAN-DS. (Grey horizontal line: average simulation error $\mathrm{SE}(1000)$. Dotted line: average simulation error plus one standard deviation. Each experiment is repeated 5 times with different random seeds. The performance is visualized with mean (solid lines) and standard deviation (shaded areas)).

Comparison with Supervised Learning: Generalization

Definition

A generated distribution $\mathbb{P}_G$ is said to generalize, under the divergence $d(\cdot,\cdot)$ with generalization error ϵ, if the following holds with high probability [?]

$$\left| d\left(\mathbb{P}_r^{(N)}, \mathbb{P}_G^{(N)}\right) - d\left(\mathbb{P}_r, \mathbb{P}_G\right) \right| \leq \epsilon, \tag{10}$$

where $\mathbb{P}_r^{(N)}$ is the empirical distribution of $\mathbb{P}_r$ with N samples, i.e., the distribution of the training data, and $\mathbb{P}_G^{(N)}$ is the empirical distribution of $\mathbb{P}_G$ with N samples drawn after the generator G is trained.

Comparison with Supervised Learning: Generalization

Local "quantile divergence"

Inspired by [?], we propose the following local-"divergence", focusing on the tail distribution of the strategy PnLs

$$d_q(\mathbb{P}_r, \mathbb{P}_G) := \frac{1}{K} \sum_{k=1}^{K} \int_0^{\alpha} \left[\int_{F^{-1}_{\Pi^k \# \mathbb{P}_r}(\tau)}^{F^{-1}_{\Pi^k \# \mathbb{P}_G}(\tau)} \left(F_{\Pi^k \# \mathbb{P}_r}(x) - \tau \right) \mathrm{d}x \right] \mathrm{d}\tau.$$

"Score divergence"

We can also use score functions to construct a "divergence" as follows

$$d_s\left(\mathbb{P}_r, \mathbb{P}_G\right) := \frac{1}{K} \sum_{k=1}^{K} \mathbb{E}_{\boldsymbol{p} \sim \mathbb{P}_r} \Big[S_\alpha\big(\mathrm{VaR}_\alpha\left(\Pi^k \# \mathbb{P}_G\right), \mathrm{ES}_\alpha\left(\Pi^k \# \mathbb{P}_G\right), \Pi^k(\boldsymbol{p})\big) - S_\alpha\big(\mathrm{VaR}_\alpha\left(\Pi^k \# \mathbb{P}_r\right), \mathrm{ES}_\alpha\left(\Pi^k \# \mathbb{P}_r\right), \Pi^k(\boldsymbol{p})\big) \Big]$$

Comparison with Supervised Learning: Generalization

Error metric	TAIL-GAN	Supervised learning
d_q	0.2139 (0.1785)	0.5810 (0.4199)
d_s	0.0169 (0.0139)	0.0317 (0.0262)

Table 1: Mean (in percentage) and standard deviation (in parentheses) of generalization errors under both divergences. Results are averaged over 10 testing data (synthetic data sets).

Scalability

To test the scalability of our model, we simulate 20 financial instruments with a given correlation structure, various temporal patterns and different tail behaviors in return distributions. More specifically, the models include 5 Gaussian distributions with different variances, 5 AR(1) with different autocorrelation coefficients ϕ, 5 GARCH(1, 1) with light-tailed noise and 5 GARCH(1, 1) with heavy-tailed noise. Other settings are the same as before.

Eigenportfolios

Portfolios constructed from the principal eigenvectors of the equity returns' correlation matrix are called eigenportfolios [**?**].
In particular, we extract the principal components of the empirical correlation matrix $\hat{\boldsymbol{\rho}}$ of price increments via eigendecomposition and rank the eigenvalues in decreasing order:

$$\hat{\boldsymbol{\rho}} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^{-1}, \tag{11}$$

where $\mathbf{Q}$ is the orthogonal matrix whose i-th column is the eigenvector $\boldsymbol{q}_i \in \mathbb{R}^M$ of $\hat{\boldsymbol{\rho}}$, and $\boldsymbol{\Lambda}$ is the diagonal matrix whose diagonal elements are the corresponding eigenvalues and $\boldsymbol{\Lambda}_{1,1} \geq \boldsymbol{\Lambda}_{2,2} \geq \cdots \geq \boldsymbol{\Lambda}_{M,M} \geq 0$.

Eigenportfolios

Denote $\boldsymbol{h} = diag(\sigma_1, \dots, \sigma_M)$ where σ_i is the empirical standard deviation of asset i. For i-th eigenvector $\boldsymbol{q}_i$, we consider its corresponding eigenportfolio,

$$\Pi^i(\boldsymbol{p}) = \frac{\left(\boldsymbol{h}^{-1}\boldsymbol{q}_i\right)^T \boldsymbol{p}}{|\boldsymbol{h}^{-1}\boldsymbol{q}_i|}, \tag{12}$$

where $|\boldsymbol{h}^{-1}\boldsymbol{q}_i|$ is used to normalize the portfolio weights so that the absolute weights sum to unity.

It is worth noting that eigenportfolios are uncorrelated since the eigenvectors of the correlation matrix are mutually orthogonal. The principal eigenportfolios are able to explain the most variation in the cross-section of returns.

Scalability

Due to the appealing features of eigenportfolios, two Tail-GAN architectures are considered:

1. GAN trained with 50 multi-asset random portfolios and dynamic strategies (Tail-GAN-DS(Rand)),
2. GAN trained with 20 multi-asset eigenportfolios and dynamic strategies (Tail-GAN-DS(Eig)).

The weights of static portfolios in Tail-GAN-DS(Rand) are *ramdomly* generated with the condition that the absolute sum of weights equals to 1.

The out-of-sample test consists of 50 multi-asset static portfolios, 20 mean-reversion and 20 trend-following strategies, and thus $K = 90$.

Scalability

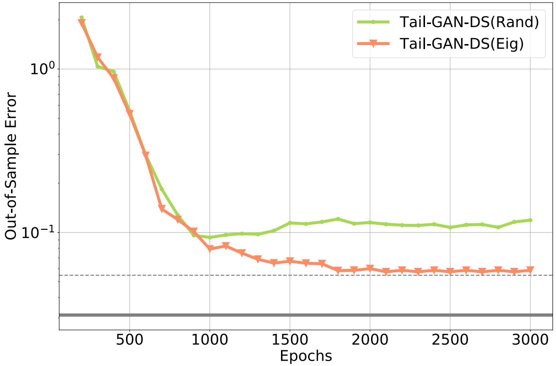


Figure 8: Out-of-sample performance. Grey horizontal line: average simulation error. Dotted line: average simulation error plus one standard deviation.

Outline

1 Introduction

2 Score functions for Tail Risk Measures

3 TAIL-GAN: Scenario Generation for Tail Risk Estimation

4 Model validation

5 Multiasset intraday scenario simulation

6 Conclusion

Application: Multiasset intraday scenario simulation

We use the Nasdaq ITCH data from LOBSTER during the intraday time interval 10:00AM-3:30PM, for the period 2011-03-23 to 2019-12-31. The reason for excluding the first and last 30 minutes of the trading day stems from the increased volatility and volume inherent in the market following and open session and preceding the closing session. The TAIL-GAN simulator is trained on the following five stocks: **AAPL, AMZN, GOOG, JPM, and QQQ**. The mid-prices (average of the best bid and the best ask) of these assets are sampled at a $\Delta = 9$-second frequency with $T = 100$ for each price series representing a financial scenario during a 15-minute interval.

Out-of-sample Performance

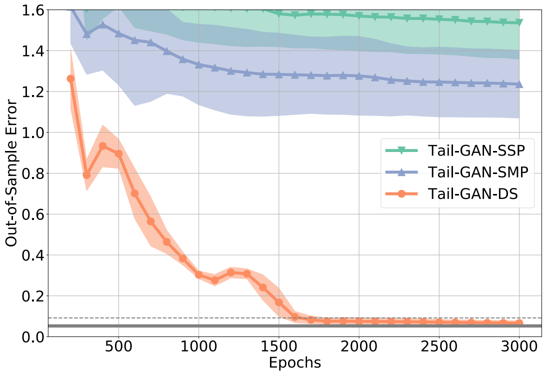
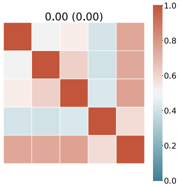
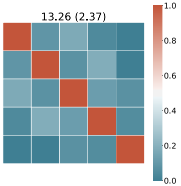
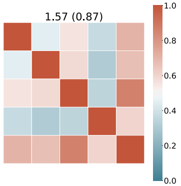


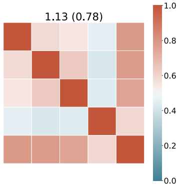
Figure 9: Out-of-sample performance $\mathrm{RE}(1000)$ of our three simulators. Grey horizontal line: average simulation error $\mathrm{SE}(1000)$. Dotted line: average simulation error plus one standard deviation. Each experiment is repeated 5 times with different random seeds. The performance is visualized with mean (solid lines) and standard deviation (shaded areas).

Correlation





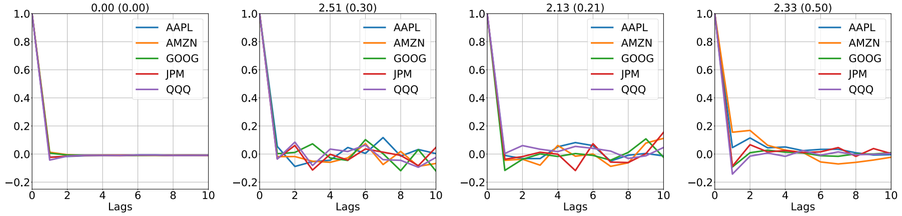




(a) Input scenario. (b) TAIL-GAN-SSP. (c) TAIL-GAN-SMP. (d) TAIL-GAN-DS.

Figure 10: Cross-asset correlations of the price increments in the market data (a) and from different trained GAN models TAIL-GAN-SSP (b), TAIL-GAN-SMP (c), and TAIL-GAN-DS (d). Numbers on the top: mean and standard deviation (in parentheses) of the sum of the absolute difference between the correlation coefficients computed with 10,000 real samples and 10,000 generated samples.

Autocorrelation



(a) Input scenario. (b) TAIL-GAN-SSP. (c) TAIL-GAN-SMP. (d) TAIL-GAN-DS.

Figure 11: Auto-correlations of the price increments from different trained GAN models: (1) TAIL-GAN-SSP, (2) TAIL-GAN-SMP, and (3) TAIL-GAN-DS. Numbers on the top: mean and standard deviation (in parentheses) of the sum of the absolute element-wise difference between auto-correlation coefficients computed with 10,000 real samples and 10,000 generated samples.

Outline

1 Introduction

2 Score functions for Tail Risk Measures

3 TAIL-GAN: Scenario Generation for Tail Risk Estimation

4 Model validation

5 Multiasset intraday scenario simulation

6 Conclusion

Conclusion

TAIL-GAN: a data driven framework for simulating multivariate financial time series data. geared towards risk measurement.

1. TAIL-GAN is capable of learning the tail properties of both static and dynamic strategies.
2. TAIL-GAN can capture the correlation structure across different assets, as well as the dynamic information along the path trajectory.
3. TAIL-GAN can be **customized** with different strategies selected by the user.
4. TAIL-GAN outperforms **supervised learning methods** in the terms of both estimation accuracy and generalization ability.
5. TAIL-GAN is **scalable** to large and heterogeneous portfolios involving multiple assets. Using eigenportfolios can improve the simulation accuracy.