Learning Extremal Dependence with Normalising Flows

Daniela Castro-Camilo and Chenglei Hu

GAME 2026

Motivation: the what, they why, and the challenge

What are we trying to do?

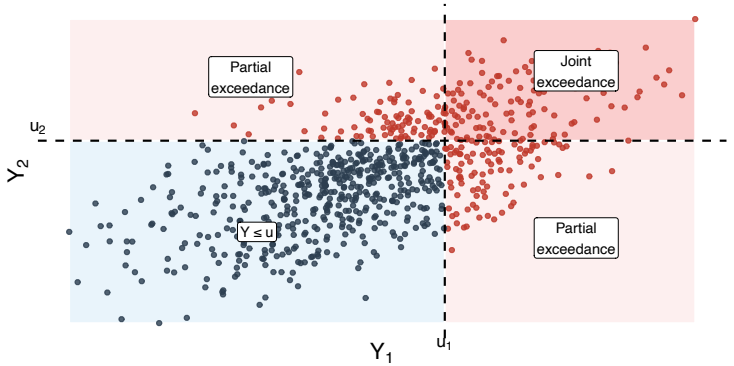
We observe a multivariate process $\boldsymbol{Y} = (Y_1, \ldots, Y_d)$, and want to model the distribution of observations for which at least one component is extreme:

$$\boldsymbol{Y} \not\leq \boldsymbol{u}$$

What are we trying to do?

We observe a multivariate process $\boldsymbol{Y} = (Y_1, \ldots, Y_d)$, and want to model the distribution of observations for which at least one component is extreme:

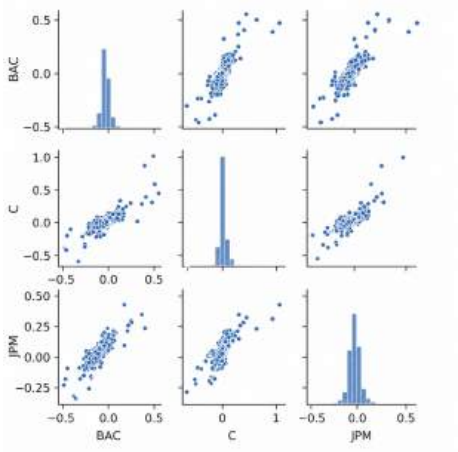
$$\boldsymbol{Y} \not\leq \boldsymbol{u}$$



What are we trying to do?

Neg log-returns of 3 US banks

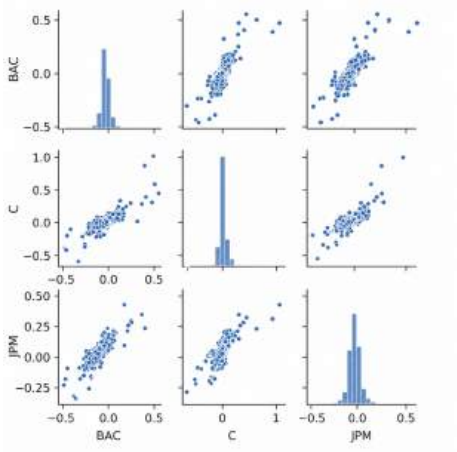
To measure how much a stock loses in value, in log scale



What are we trying to do?

Neg log-returns of 3 US banks

To measure how much a stock loses in value, in log scale



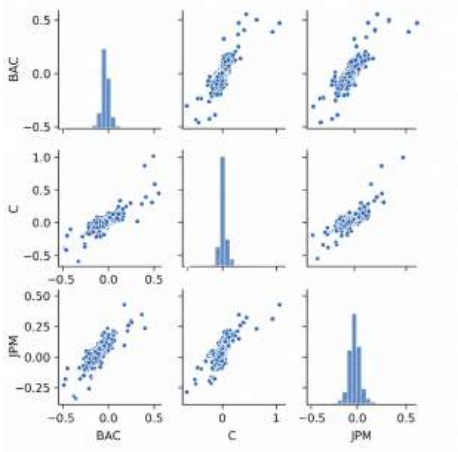
Practical questions when studying $\boldsymbol{Y} \not\leq \boldsymbol{u}$ include:

- What is the prob that several variables are extreme together?
- What is the prob that only a subset is extreme?
- How does one component behave when another is under stress?

What are we trying to do?

Neg log-returns of 3 US banks

To measure how much a stock loses in value, in log scale



Practical questions when studying $\boldsymbol{Y} \not\leq \boldsymbol{u}$ include:

- What is the prob that several variables are extreme together?
- What is the prob that only a subset is extreme?
- How does one component behave when another is under stress?

The challenge

Extremes are rare, dependence is high-dimensional, and extrapolation might require structure.

What does the theory tell us?

Under max-domain of attraction conditions[1], the multivariate generalised Pareto distribution (mGPD) arises as the limiting law of $\boldsymbol{Y}$ conditional on at least one component being extreme: $\boldsymbol{Y} - \boldsymbol{u} \mid \boldsymbol{Y} \not\leq \boldsymbol{u}$.

[1] after suitable rescaling, the max of a large sample converges to an extreme-value distribution

Under max-domain of attraction conditions[1], the multivariate generalised Pareto distribution (mGPD) arises as the limiting law of $\boldsymbol{Y}$ conditional on at least one component being extreme: $\boldsymbol{Y} - \boldsymbol{u} \mid \boldsymbol{Y} \not\leq \boldsymbol{u}$.

For exceedances $\boldsymbol{X} = \boldsymbol{Y} - \boldsymbol{u} \mid \boldsymbol{Y} \not\leq \boldsymbol{u}$, the mGPD is determined by:

$$\underbrace{(\sigma_1, \ldots, \sigma_d)}_{\text{marginal scales}}, \quad \underbrace{(\gamma_1, \ldots, \gamma_d)}_{\text{marginal tails}}, \quad \underbrace{\ell(\cdot)}_{\text{extremal dependence}}$$

[1] after suitable rescaling, the max of a large sample converges to an extreme-value distribution

Under max-domain of attraction conditions[1], the multivariate generalised Pareto distribution (mGPD) arises as the limiting law of $\boldsymbol{Y}$ conditional on at least one component being extreme: $\boldsymbol{Y} - \boldsymbol{u} \mid \boldsymbol{Y} \not\leq \boldsymbol{u}$.

For exceedances $\boldsymbol{X} = \boldsymbol{Y} - \boldsymbol{u} \mid \boldsymbol{Y} \not\leq \boldsymbol{u}$, the mGPD is determined by:

$$\underbrace{(\sigma_1, \ldots, \sigma_d)}_{\text{marginal scales}}, \quad \underbrace{(\gamma_1, \ldots, \gamma_d)}_{\text{marginal tails}}, \quad \underbrace{\ell(\cdot)}_{\text{extremal dependence}}$$

Main difficulty

The mGPD is characterised by properties, and therefore doesn't have a unique representation. Only a few tractable parametric models are available in practice, mostly due to the work of A. Kiriliouk, H. Rootzen, J. Segers and J. Wadsworth (2018a, 2018b, 2019).

[1] after suitable rescaling, the max of a large sample converges to an extreme-value distribution

A stochastic representation of the mGPD

- A standard mGPD vector $\boldsymbol{Z}$ (i.e., fully characterised by the dependence structure) can be constructed using the so-called **T-representation:**

$$\boldsymbol{Z} = E + \boldsymbol{T} - \max(\boldsymbol{T}), \quad \text{where} \quad E \sim \text{Exp}(1),$$

and $\boldsymbol{T} \perp\!\!\!\perp E$ is a random vector obeying some mild conditions , called **the generator of** Z, controlling extremal dependence.

A stochastic representation of the mGPD

- A standard mGPD vector $\boldsymbol{Z}$ (i.e., fully characterised by the dependence structure) can be constructed using the so-called **T-representation:**

$$\boldsymbol{Z} = E + \boldsymbol{T} - \max(\boldsymbol{T}), \quad \text{where} \quad E \sim \text{Exp}(1),$$

and $\boldsymbol{T} \perp\!\!\!\perp E$ is a random vector obeying some mild conditions , called **the generator of** Z, controlling extremal dependence. The density h of $\boldsymbol{Z}$ is given by

$$h(\boldsymbol{z}) = \frac{\mathbf{1}\{\max(\boldsymbol{z} > 0)\}}{\exp\{\max(\boldsymbol{z})\}} \int_{-\infty}^{\infty} f_{\boldsymbol{T}}(\boldsymbol{z}+s)\, ds.$$

← The important thing here is that we have a way to express h as a function of $f_{\boldsymbol{T}}$

A stochastic representation of the mGPD

- A standard mGPD vector $\boldsymbol{Z}$ (i.e., fully characterised by the dependence structure) can be constructed using the so-called **T-representation:**

$$\boldsymbol{Z} = E + \boldsymbol{T} - \max(\boldsymbol{T}), \quad \text{where} \quad E \sim \text{Exp}(1),$$

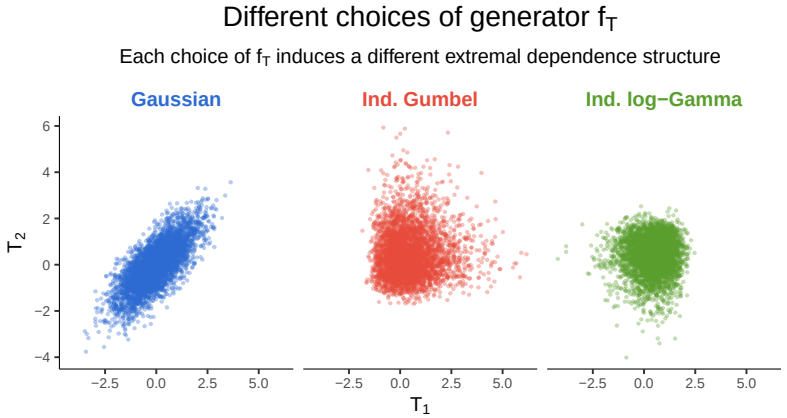
and $\boldsymbol{T} \perp\!\!\!\perp E$ is a random vector obeying some mild conditions , called **the generator of** Z, controlling extremal dependence. The density h of $\boldsymbol{Z}$ is given by

$$h(\boldsymbol{z}) = \frac{\mathbf{1}\{\max(\boldsymbol{z} > 0)\}}{\exp\{\max(\boldsymbol{z})\}} \int_{-\infty}^{\infty} f_{\boldsymbol{T}}(\boldsymbol{z}+s)\, ds.$$

← The important thing here is that we have a way to express h as a function of $f_{\boldsymbol{T}}$

- There are alternative reps, which can be transformed into the T-rep by adding some constraints.
- The generator $\boldsymbol{T}$ can be almost any distribution on $\mathbb{R}^d$, provided its exponential moments are finite. Different choices of $f_{\boldsymbol{T}}$ induce different extremal dependence structures, which is why modelling $f_{\boldsymbol{T}}$ is challenging.

f_T is the hard part



The modelling bottleneck

classical mGPD models require us to choose f_T from a small catalogue of parametric families. GPDFlow replaces this choice by a flexible learned generator.

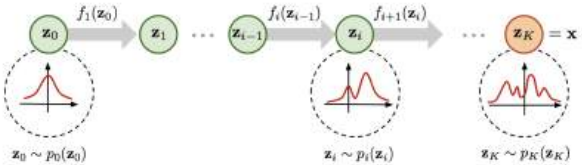
Going with the flow

Not this one though!

this one either...

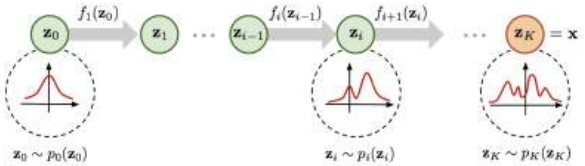
Introducing: Flows

A normalising flow transforms a simple base distribution into a complex target distribution:

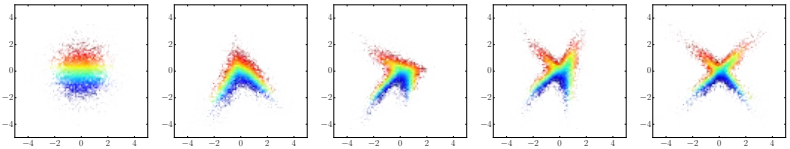


Introducing: Flows

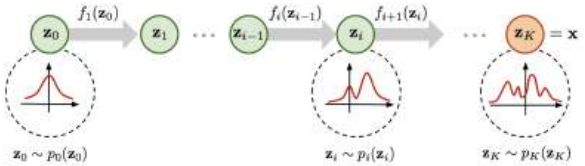
A normalising flow transforms a simple base distribution into a complex target distribution:



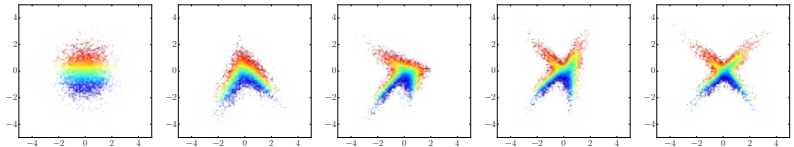
Two-dimensional example (Papamakarios et al., 2021):



A normalising flow transforms a simple base distribution into a complex target distribution:



Two-dimensional example (Papamakarios et al., 2021):



Open problem: how to choose the sequence of transformations

Our choice: Real NVP

Real NVP = Real-valued Non-Volume Preserving

An invertible transformation that can stretch and compress space while retaining a tractable Jacobian

Real NVP = Real-valued Non-Volume Preserving

An invertible transformation that can stretch and compress space while retaining a tractable Jacobian

Not to be confused with

Normalising flows: Real NVP in a nutshell

Real NVP constructs a complicated transformation by stacking many simple invertible transformations.

$$U \sim \mathcal{N}(0, I) \qquad \Longrightarrow \qquad T = g(U)$$

Normalising flows: Real NVP in a nutshell

Real NVP constructs a complicated transformation by stacking many simple invertible transformations.

$$U \sim \mathcal{N}(0, I) \qquad \Longrightarrow \qquad T = g(U)$$

Example: One Real NVP coupling layer

Normalising flows: Real NVP in a nutshell

Real NVP constructs a complicated transformation by stacking many simple invertible transformations.

$$U \sim \mathcal{N}(0, I) \qquad \Longrightarrow \qquad T = g(U)$$

Example: One Real NVP coupling layer

- Split the vector into two parts: $U = (U_A, U_B)$., leave one part unchanged and transform the other:

$$T_A = U_A, \quad T_B = U_B \odot \exp\{s(U_A)\} + t(U_A), \qquad s(\cdot) \text{ and } t(\cdot) \text{ are NNs.}$$

"Coupling layer" $\rightarrow$ One part of the vector (U_A) controls the transformation of the other (U_B), creating dependence

Normalising flows: Real NVP in a nutshell

Real NVP constructs a complicated transformation by stacking many simple invertible transformations.

$$U \sim \mathcal{N}(0, I) \qquad \Longrightarrow \qquad T = g(U)$$

Example: One Real NVP coupling layer

- Split the vector into two parts: $U = (U_A, U_B)$., leave one part unchanged and transform the other:

$$T_A = U_A, \quad T_B = U_B \odot \exp\{s(U_A)\} + t(U_A), \qquad s(\cdot) \text{ and } t(\cdot) \text{ are NNs.}$$

"Coupling layer" $\rightarrow$ One part of the vector (U_A) controls the transformation of the other (U_B), creating dependence

- Inverse U_B is immediate (no numerical optimisation needed!)

Normalising flows: Real NVP in a nutshell

Real NVP constructs a complicated transformation by stacking many simple invertible transformations.

$$U \sim \mathcal{N}(0, I) \qquad \Longrightarrow \qquad T = g(U)$$

Example: One Real NVP coupling layer

- Split the vector into two parts: $U = (U_A, U_B)$., leave one part unchanged and transform the other:

$$T_A = U_A, \quad T_B = U_B \odot \exp\{s(U_A)\} + t(U_A), \qquad s(\cdot) \text{ and } t(\cdot) \text{ are NNs.}$$

"Coupling layer" $\rightarrow$ One part of the vector (U_A) controls the transformation of the other (U_B), creating dependence

- Inverse U_B is immediate (no numerical optimisation needed!)
- Jacobian is easy (triangular):

$$\log|\det J| = \sum_j s_j(U_A)$$

$\rightarrow$ The determinant depends on the values of s_j, but not on how s_j is constructed.

Normalising flows: Real NVP in a nutshell

Real NVP constructs a complicated transformation by stacking many simple invertible transformations.

$$U \sim \mathcal{N}(0, I) \qquad \Longrightarrow \qquad T = g(U)$$

Example: One Real NVP coupling layer

- Split the vector into two parts: $U = (U_A, U_B)$., leave one part unchanged and transform the other:

$$T_A = U_A, \quad T_B = U_B \odot \exp\{s(U_A)\} + t(U_A), \qquad s(\cdot) \text{ and } t(\cdot) \text{ are NNs.}$$

"Coupling layer" $\rightarrow$ One part of the vector (U_A) controls the transformation of the other (U_B), creating dependence

- Inverse U_B is immediate (no numerical optimisation needed!)
- Jacobian is easy (triangular):

$$\log|\det J| = \sum_j s_j(U_A)$$

$\rightarrow$ The determinant depends on the values of s_j, but not on how s_j is constructed.

- We can consider many, many layers and stack them such that

$$U = Z_0 \rightarrow Z_1 \rightarrow Z_2 \rightarrow \cdots \rightarrow T$$

- Inference with normalising flows can be based on standard MLE because **we can get an expression for the likelihood!**

- Let $\boldsymbol{\theta}$ denote the flow parameters (e.g., NN weights). The density of the transformed variable is:

$$f_{\boldsymbol{Y}}(\boldsymbol{y} \mid \boldsymbol{\theta}) = f_{\boldsymbol{U}}(\boldsymbol{u}) \cdot |\det J_g(\boldsymbol{u} \mid \boldsymbol{\theta})|^{-1}, \quad \boldsymbol{u} = g^{-1}(\boldsymbol{y} \mid \boldsymbol{\theta})$$

Why this matters:

- We obtain the exact likelihood, not just an approximation.
- Enables principled training via maximum likelihood.

GPDFlow

- Recall the **T-representation** and associated density:

$$\boldsymbol{Z} = E + \boldsymbol{T} - \max(\boldsymbol{T}), \quad \text{where } \boldsymbol{T} \text{ is the generator and } E \sim \mathrm{Exp}(1)$$

$$h(\boldsymbol{z}) = \frac{\mathbf{1}\{\max(\boldsymbol{z} > 0)\}}{\exp\{\max(\boldsymbol{z})\}} \int_{-\infty}^{\infty} f_{\boldsymbol{T}}(\boldsymbol{z} + s)\mathrm{d}s.$$

- Recall the **T-representation** and associated density:

$$\boldsymbol{Z} = E + \boldsymbol{T} - \max(\boldsymbol{T}), \quad \text{where } \boldsymbol{T} \text{ is the generator and } E \sim \text{Exp}(1)$$

$$h(\boldsymbol{z}) = \frac{\mathbf{1}\{\max(\boldsymbol{z} > 0)\}}{\exp\{\max(\boldsymbol{z})\}} \int_{-\infty}^{\infty} f_{\boldsymbol{T}}(\boldsymbol{z} + s)\mathrm{d}s.$$

- **GPDFlow idea:** Represent $\boldsymbol{T}$ using a real NVP. In this case, the unstandardised density becomes

$$f(\boldsymbol{x}; \boldsymbol{\sigma}, \boldsymbol{\gamma}, \boldsymbol{\theta}) = \frac{\mathbb{1}\{\max(\boldsymbol{z}) > 0\}}{\exp\{\max(\boldsymbol{z})\}} \int_{-\infty}^{\infty} f_U\left(g^{-1}(\boldsymbol{z} + s; \boldsymbol{\theta})\right) \left|\det J_g(\boldsymbol{z} + s; \boldsymbol{\theta})\right|^{-1} ds \times \prod_{j=1}^{d} \frac{1}{\sigma_j + \gamma_j x_j}$$

where $\boldsymbol{z} = g_{\text{std}}(\boldsymbol{x})$ is the standardised version.

So GPDFlow is just a d-dimensional distribution with the above density.

$$f(\mathbf{x};\boldsymbol{\sigma},\boldsymbol{\gamma},\boldsymbol{\theta}) = \frac{\mathbb{1}\{\max(\boldsymbol{z})>0\}}{\exp\{\max(\boldsymbol{z})\}} \int_{-\infty}^{\infty} f_U\left(g^{-1}(\boldsymbol{z}+s;\boldsymbol{\theta})\right) \left|\det J_g(\boldsymbol{z}+s;\boldsymbol{\theta})\right|^{-1} ds \times \prod_{j=1}^{d} \frac{1}{\sigma_j+\gamma_j x_j}$$

Inference

Likelihood easily derived from the expression above!

Important:

- Although $\boldsymbol{z}$ is d-dimensional, the integral is only over a scalar variable s.
- We update $(\boldsymbol{\theta},\boldsymbol{\sigma},\boldsymbol{\gamma})$ by maximising the full likelihood $\ell(\boldsymbol{\sigma},\boldsymbol{\gamma},\boldsymbol{\theta})$.

$$f(\mathbf{x};\boldsymbol{\sigma},\boldsymbol{\gamma},\boldsymbol{\theta}) = \frac{\mathbb{1}\{\max(\boldsymbol{z})>0\}}{\exp\{\max(\boldsymbol{z})\}} \int_{-\infty}^{\infty} f_U\left(g^{-1}(\boldsymbol{z}+s;\boldsymbol{\theta})\right) \left|\det J_g(\boldsymbol{z}+s;\boldsymbol{\theta})\right|^{-1} ds \times \prod_{j=1}^{d} \frac{1}{\sigma_j+\gamma_j x_j}$$

Inference

Likelihood easily derived from the expression above!

Important:

- Although $\boldsymbol{z}$ is d-dimensional, the integral is only over a scalar variable s.
- We update $(\boldsymbol{\theta},\boldsymbol{\sigma},\boldsymbol{\gamma})$ by maximising the full likelihood $\ell(\boldsymbol{\sigma},\boldsymbol{\gamma},\boldsymbol{\theta})$.

Two perspectives on GPDFlow

1. **Statistical:** GPDFlow is an mGPD with a highly flexible dependence structure.
2. **Machine Learning:** Samples from f are generated by a composition of transformations.

GPDFlow: Short note on simulation study

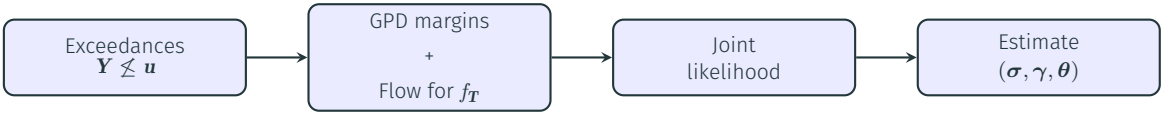
- We assess whether GPDFlow can learn extremal dependence and compare against parametric mGPD models (three different generators $\boldsymbol{T}$) under correct model specification and misspecification.

- We evaluated them in terms of how well they can recover extremal dependence and estimate tail dependence coefficients.

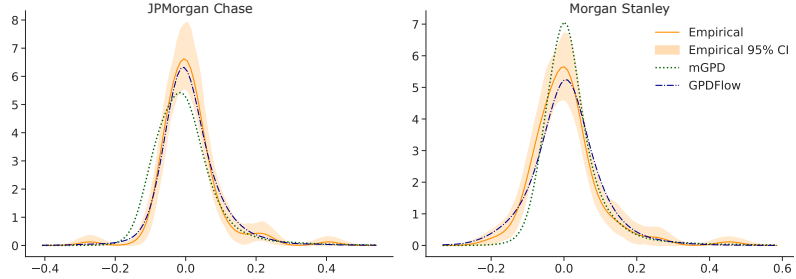
- Results are comparable under correct specification, GPDFlow outperforms under misspecification.

Application on Systemic financial risk: Estimated densities

- We analyse the 5-day negative log returns of 5 major US banks (2005-2025)
- GPDFlow steps:



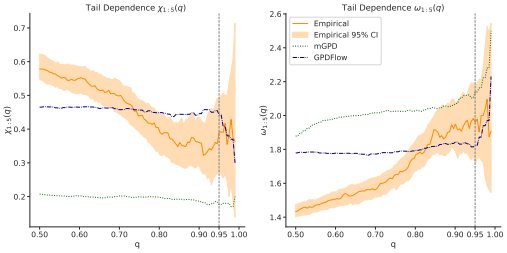
- We compare GPDFlow with the best parametric mGPD (selected via AIC)



Application on Systemic financial risk: Extremal dependence

- $\chi_{1:5}(q) \in [0, 1]$: If one component is extreme, how likely are all the others to be extreme as well?
- $\omega_{1:5}(q) = \frac{\Pr(\text{at least one variable is extreme})}{\Pr(\text{one variable is extreme})} \in (1, d)$: measures of the spread of extreme events across variables

Measure	Strong extremal dependence *exceedances tend to occur together*	Weak extremal dependence *exceedances tend to occur separately*
$\chi_{1:d}$	1	0
$\omega_{1:d}$	1	d

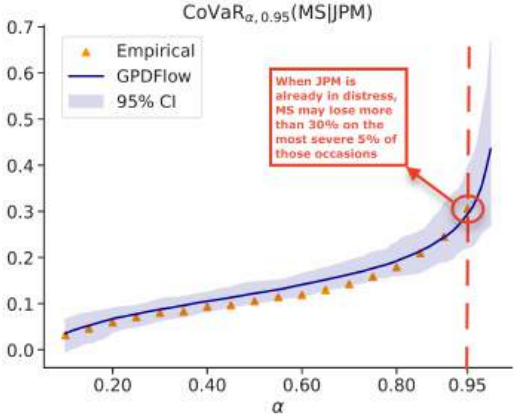


Application on Systemic financial risk: CoVaR estimation

- **Risk measure**: Conditional Value-at-Risk (CoVaR)

$$\text{CoVaR}_{\alpha,\beta}(Y \mid X) = \text{VaR}_{\alpha}\left(Y \mid X \geq \text{VaR}_{\beta}(X)\right)$$

$\Rightarrow$ Measures how bad things can get for one institution when another institution is already in trouble.



Final thoughts

- No sensitivity on the choice of flow

Final thoughts

· No sensitivity on the choice of flow

· The real NVP uses a Gaussian base, but this does not imply asymptotic independence of the resulting GPDFlow model necessarily, since the flow is non-linear

→ Joint tail behaviour of T is induced by the learned scaling factors $\exp\{s(U_A)\}$

Final thoughts

· No sensitivity on the choice of flow

· The real NVP uses a Gaussian base, but this does not imply asymptotic independence of the resulting GPDFlow model necessarily, since the flow is non-linear

→ Joint tail behaviour of T is induced by the learned scaling factors $\exp\{s(U_A)\}$

· Can GPDFlow genuinely span both asymptotically dependent and asymptotically independent regimes?

→ We derive χ for a fitted GPDFlow model, but we do not characterise which asymptotic dependence classes are attainable under Real NVP generators.

Final thoughts

· No sensitivity on the choice of flow

· The real NVP uses a Gaussian base, but this does not imply asymptotic independence of the resulting GPDFlow model necessarily, since the flow is non-linear

→ Joint tail behaviour of T is induced by the learned scaling factors $\exp\{s(U_A)\}$

· Can GPDFlow genuinely span both asymptotically dependent and asymptotically independent regimes?

→ We derive χ for a fitted GPDFlow model, but we do not characterise which asymptotic dependence classes are attainable under Real NVP generators.

· So far, not entirely clear if real NVP can learn the correct tail geometry from finite data, but for GPDFlow, the situation is somewhat better:

- we fit only exceedances (not the entire dist);
- EVT supplies the marginal tail behaviour;
- the flow only learns the dependence generator.

Final thoughts

Question	Answer
Can Real NVP represent heavy tails?	✓
Can Real NVP represent asymptotic dependence?	Probably
Can Real NVP learn the correct tail geometry from finite data?	?
Are there theoretical guarantees?	Few
Is this an active research area?	✓

Key references

- Kiriliouk, A., Rootzén, H., Segers, J., & Wadsworth, J. L. (2019). Peaks over thresholds modeling with multivariate generalized Pareto distributions. *Technometrics*, 61(1), 123-135.
- Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., & Lakshminarayanan, B. (2021). Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1-64.
- Rootzén, H., Segers, J., & L. Wadsworth, J. (2018a). Multivariate peaks over thresholds models. *Extremes*, 21(1), 115-145.
- Rootzén, H., Segers, J., & Wadsworth, J. L. (2018b). Multivariate generalized Pareto distributions: Parametrizations, representations, and properties. *JMVA*, 165, 117-131.

Scan here for full manuscript
(currently being updated)