Pathwise methods and Robust GANs for Pricing and Hedging

Blanka Horvath
University of Oxford, the Oxford Man Institute

International Conference on Computational Finance
—ICCF24 —
3rd April 2024, Amsterdam

Motivation: DL and Automation for Financial Applications

Example: Hedging

Motivation: DL and Automation for Financial Applications

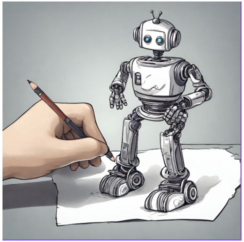
Example: Deep Hedging

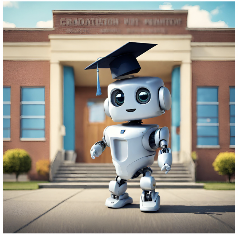
Motivation: DL and Automation for Financial Applications

Example: Deep Hedging

Deep Hedging

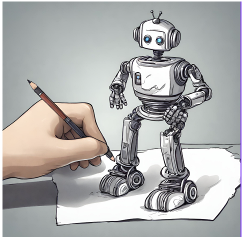
- Designing your model and features
- Learning on a “training dataset”
- Testing on an unseen dataset “out of sample”

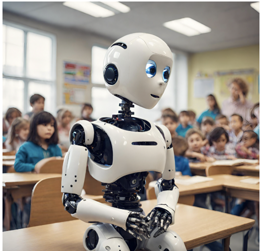


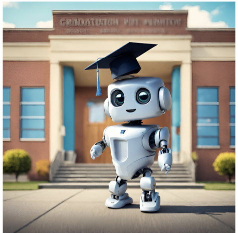


Deep Hedging

- Designing your model and features
- Learning on a “training dataset”
- Testing on an unseen dataset “out of sample”







Generating Synthetic Training Data

(ML Model design + TrainingData) $\Rightarrow$ Program (= Trained Network)
(Program + TestData) $\Rightarrow$ Output (= Strategy)

Generating Synthetic Training Data

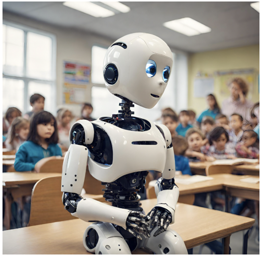
$$(\textbf{ML Model design} + \textbf{TrainingData}) \Rightarrow \textbf{Program} \ (= \textbf{Trained Network})$$
$$(\textbf{Program} + \textbf{TestData}) \Rightarrow \textbf{Output} \quad (= \textbf{Strategy})$$

Task "Market Generator":
Find (synthetic) **TrainingData** for the network, such that performance is optimized when **TestData** = **Future Market Data**.

Generating Synthetic Training Data

$$(\textbf{ML Model design} + \textbf{TrainingData}) \Rightarrow \textbf{Program} \; (= \textbf{Trained Network})$$
$$(\textbf{Program} + \textbf{TestData}) \Rightarrow \textbf{Output} \quad (= \textbf{Strategy})$$

Task "Market Generator":
Find (synthetic) **TrainingData** for the network, such that performance is optimized when **TestData** = **Future Market Data**.

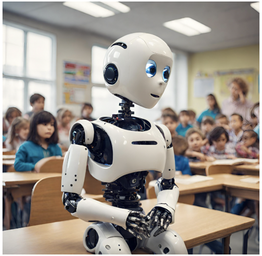


Generating Synthetic Training Data

$$(\textbf{ML Model design} + \textbf{TrainingData}) \Rightarrow \textbf{Program} \ (= \textbf{Trained Network})$$
$$(\textbf{Program} + \textbf{TestData}) \Rightarrow \textbf{Output} \ (= \textbf{Strategy})$$

Task "Market Generator":

Find (synthetic) **TrainingData** for the network, such that performance is optimized when **TestData** = **Future Market Data**.



How to train the model to prepare it for future challenges?

Generating Synthetic Training Data

$$\textbf{(ML Model design + TrainingData)} \Rightarrow \textbf{Program } \color{red}{\textbf{(= Trained Network)}}$$
$$\textbf{(Program + TestData)} \Rightarrow \textbf{Output } \color{red}{\textbf{(= Strategy)}}$$

Task "Market Generator":
Find (synthetic) **TrainingData** for the network, such that performance is optimized when **TestData** = **Future Market Data**.

Training on the Black & Scholes Model

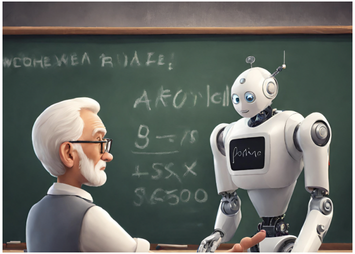
Generating Synthetic Training Data

(ML Model design + TrainingData) $\Rightarrow$ Program (= Trained Network)

(Program + TestData) $\Rightarrow$ Output (= Strategy)

Task "Market Generator":

Find (synthetic) **TrainingData** for the network, such that performance is optimized when **TestData = Future Market Data**.



Training on synthetic data from (scholarly) stochastic models:
B&S, Heston, ...and more complex models that are well-undestood

How about training on real data?

Generating Synthetic Training Data

(ML Model design + TrainingData) ⇒ Program (= Trained Network)
(Program + TestData) ⇒ Output (= Strategy)

How about training on real data?

Generating Synthetic Training Data

(ML Model design + TrainingData) $\Rightarrow$ Program (= Trained Network)

(Program + TestData) $\Rightarrow$ Output (= Strategy)

How about training on real data? **Training on daily stock market data**

(1) Too few datapoints to train ML

Challenges with real (historical) data:

(1) **Data availability:** In many real situations, there is very limited data available for training including limitations due to **data privacy**... and many more.

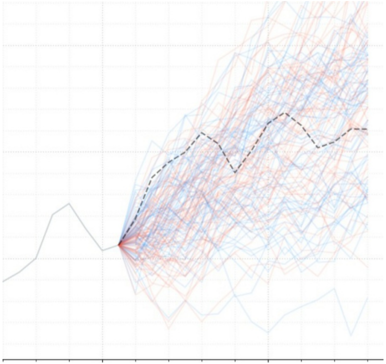
(2) **Computational limitations:** Some limitations imposed on datasets (real & synthetic) by computational- and memory considerations (examples later).

(3) **Data changes over time:** Markets are heteroskedastic and non-stationary

Challenges with real (historical) data:

Task "Market Generator": Generate (synthetic) **TrainingData** for the network, such that performance is optimized when **TestData** = Market Data.

The challenge that market generators face is to produce **genuinely new data** samples that in aggregation have the same distribution as the test data they will later be exposed to, **though we only see one realisation of the path**. (Image by Zach Issa:)



Reality only happens once.

$$(\textbf{ML Model design} + \textbf{TrainingData}) \Rightarrow \textbf{Program} \; {\color{red}(= \textbf{Trained Network})}$$
$$(\textbf{Program} + \textbf{TestData}) \Rightarrow \textbf{Output} \; {\color{red}(= \textbf{Strategy})}$$

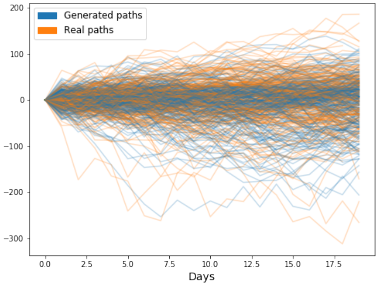
How about training on real data? **Training on daily stock market data**

(3) Raw training data often not informative enough

Generating Synthetic Data

Task "Market Generator": Generate (synthetic) **TrainingData** for the network, such that performance is optimized when **TestData** = Market Data.

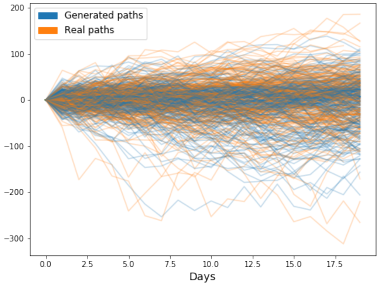
The challenge is to produce **genuinely new data** samples that in aggregation have the same distribution as the provided sample.



Market Simulation and Market Generation

(1) Classical and Neo-classical stochastic market models: $dS_t = rS_t dt + \sigma S_t dW_t \ldots$

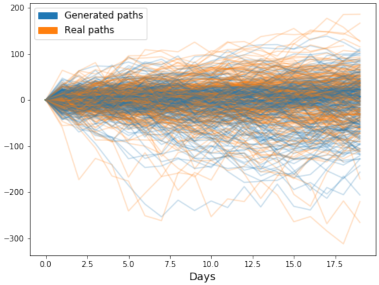
(2) DNN-based Generative Modelling for time series applications:
Data driven (no a-priori assumption on distribution of stochastic process)
$f \in \mathcal{N}_r(I, d_1, \ldots, d_{r-1}, O; \sigma_1, \ldots \sigma_r)$ "non-parametric" + very flexible

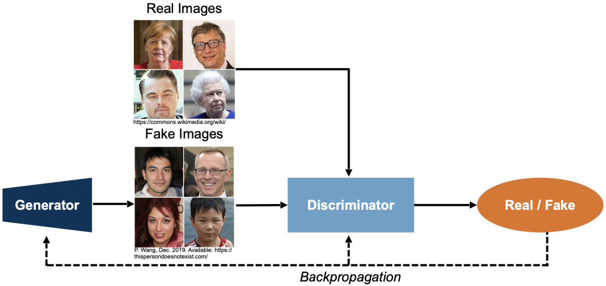


Market Simulation and Market Generation

(1) Classical and Neo-classical stochastic market models: $dS_t = rS_t dt + \sigma S_t dW_t \ldots$

(2) DNN-based Generative Modelling for time series applications:
Data driven (no a-priori assumption on distribution of stochastic process)
$f \in \mathcal{N}_r(I, d_1, \ldots, d_{r-1}, O; \sigma_1, \ldots \sigma_r)$ "non-parametric" + very flexible



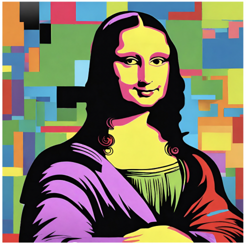


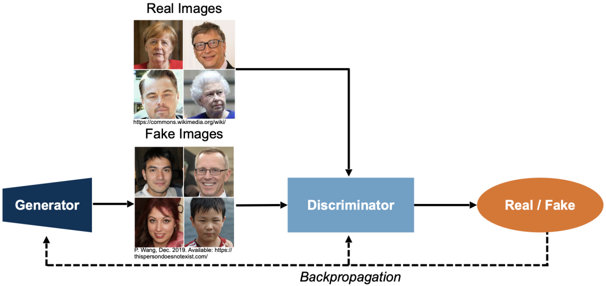
Market Simulation and Market Generation

(1) Classical and Neo-classical stochastic market models: $dS_t = rS_t dt + \sigma S_t dW_t \ldots$

(2) DNN-based Generative Modelling for time series applications:
Data driven (no a-priori assumption on distribution of stochastic process)
$f \in \mathcal{N}_r(I, d_1, \ldots, d_{r-1}, O; \sigma_1, \ldots \sigma_r)$ "non-parametric" + very flexible

Advantage: Ability to make use of data in different settings (via conditioning).

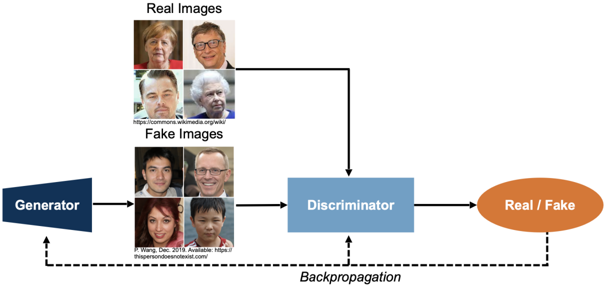




Market Simulation and Market Generation

(1) Classical and Neo-classical stochastic market models: $dS_t = rS_t dt + \sigma S_t dW_t \ldots$

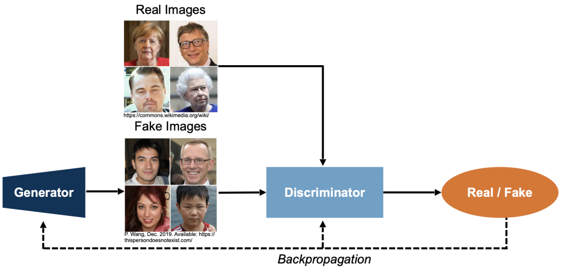
(2) DNN-based Generative Modelling for time series applications:
Data driven (no a-priori assumption on distribution of stochastic process)
$f \in \mathcal{N}_r(I, d_1, \ldots, d_{r-1}, O; \sigma_1, \ldots \sigma_r)$ "non-parametric" + very flexible



Market Simulation and Market Generation

(1) Classical and Neo-classical stochastic market models: $dS_t = rS_t dt + \sigma S_t dW_t \ldots$

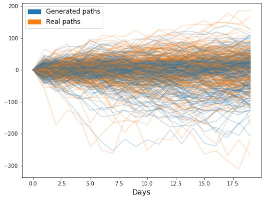
(2) DNN-based Generative Modelling for time series applications:
Data driven (no a-priori assumption on distribution of stochastic process)
$f \in \mathcal{N}_r(I, d_1, \ldots, d_{r-1}, O; \sigma_1, \ldots \sigma_r)$ "non-parametric" + very flexible

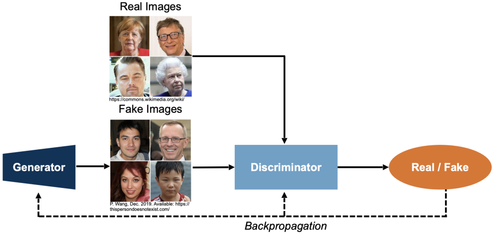


Market Simulation and Market Generation

(1) Classical and Neo-classical stochastic market models: $dS_t = rS_t dt + \sigma S_t dW_t \ldots$

(2) DNN-based Generative Modelling for time series applications:
Data driven (no a-priori assumption on distribution of stochastic process)
$f \in \mathcal{N}_r(I, d_1, \ldots, d_{r-1}, O; \sigma_1, \ldots \sigma_r)$ "non-parametric" + very flexible





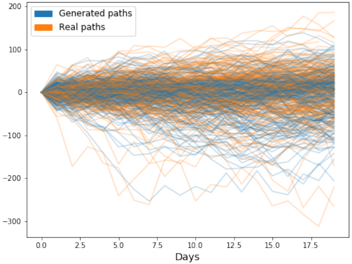
Market Simulation and Market Generation

(1) Classical and Neo-classical stochastic market models: $dS_t = rS_t dt + \sigma S_t dW_t \ldots$

(2) DNN-based Generative Modelling for time series applications:
Data driven (no a-priori assumption on distribution of stochastic process)
$f \in \mathcal{N}_r(I, d_1, \ldots, d_{r-1}, O; \sigma_1, \ldots \sigma_r)$ "non-parametric" + very flexible

Cornerstone for (2): Evaluating the "quality" of produced synthetic data samples (especially for data streams)

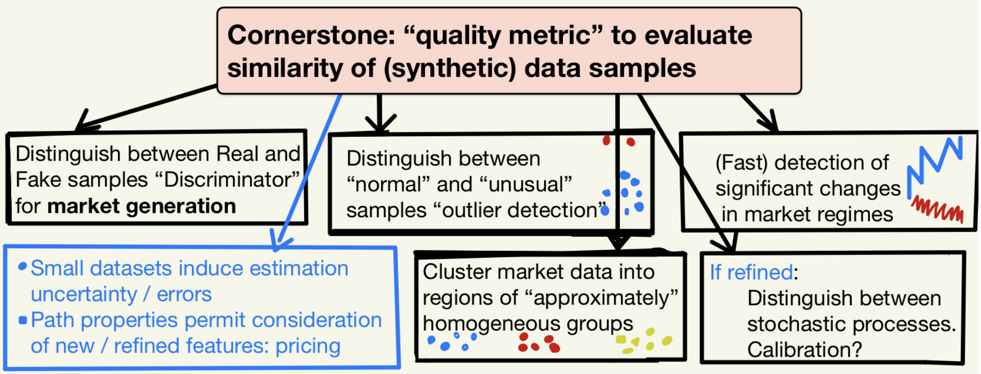
(Generator)



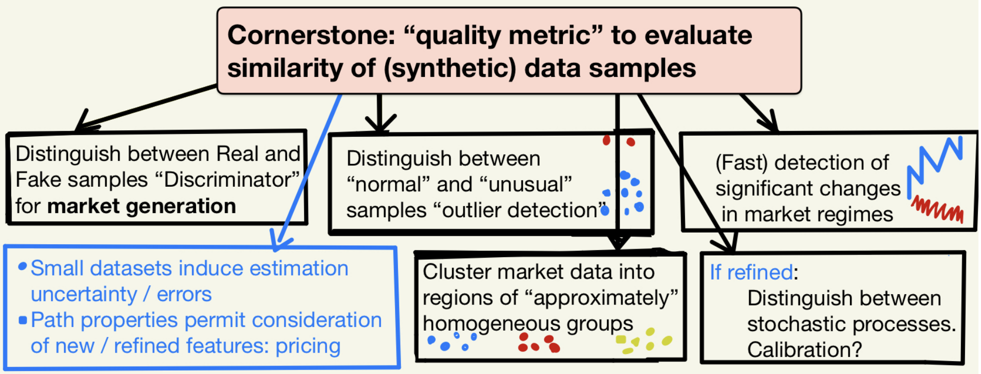
(Discriminator)

Key: Similarity metric and possible applications

Key: Similarity metric and possible applications

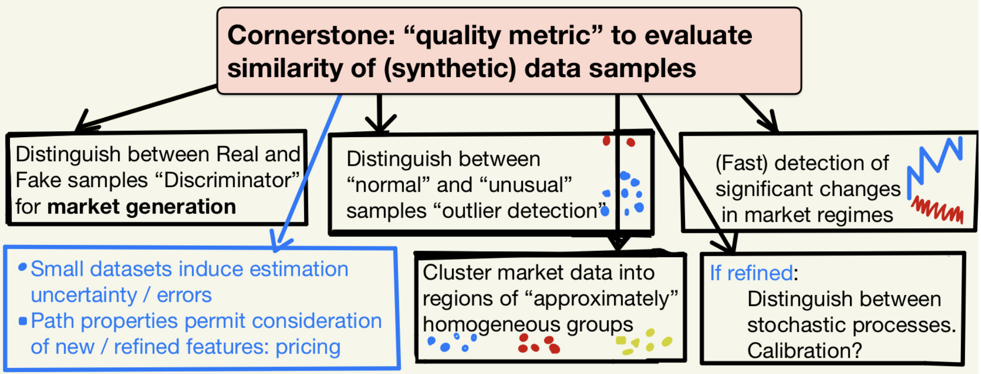


Key: Similarity metric and possible applications



(1) Truncated signature MMD [BHLPW'20] (see more details later) 1st market gen

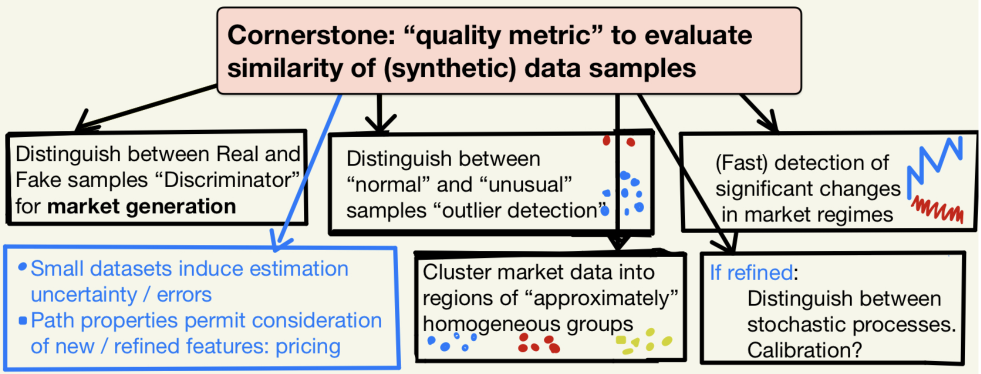
Key: Similarity metric and possible applications



(1) Truncated signature MMD [BHLPW'20] (see more details later) 1st market gen

(2),(3) Higher rank sig-kernel (with kernel trick) [HLLMS'22] comp, [HI'23] regime det

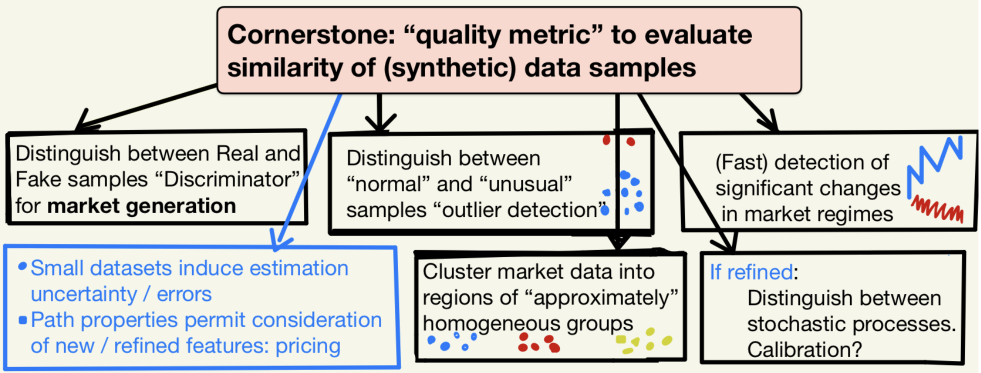
Key: Similarity metric and possible applications



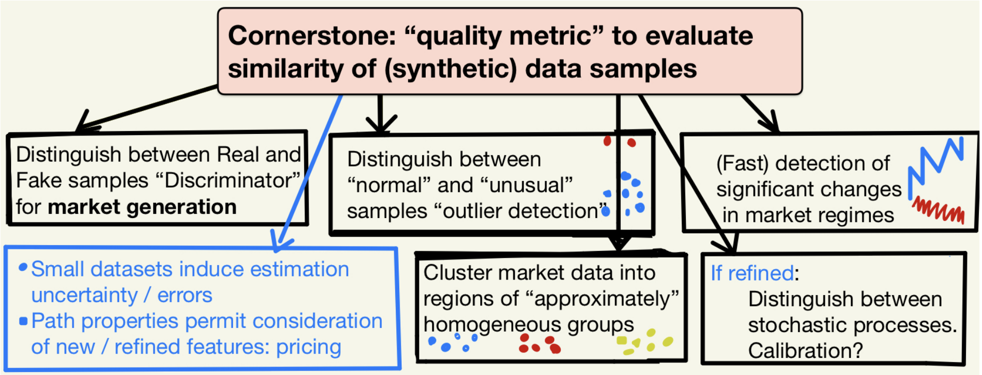
(1) Truncated signature MMD [BHLPW'20] (see more details later) 1st market gen

(2),(3) Higher rank sig-kernel (with kernel trick) [HLLMS'22] comp, [HI'23] regime det

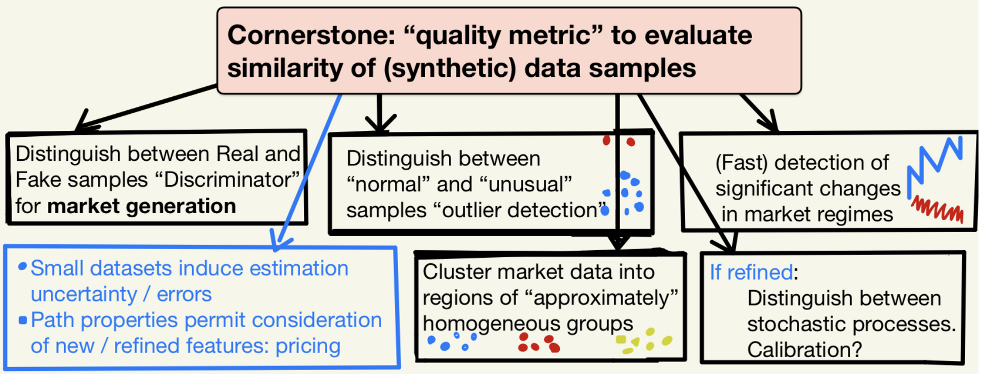
(4) **Kernel Scoring Rules [IHLS'23]** significantly improved conditional market gen



Clustering Market Regimes Using the Wasserstein Distance with Z. Issa and A. Muguruza, 2021.

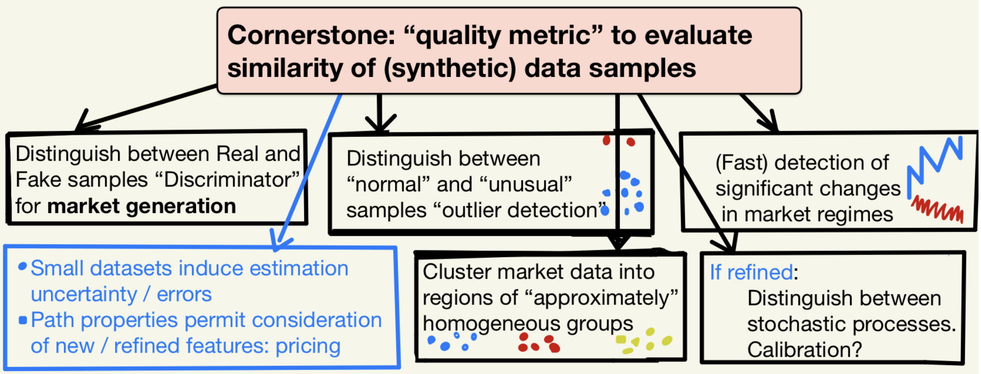


Non-parametric Online Market Regime Detection and Regime Clustering for Multidimensional and Path-Dependent Data Structures, H. Issa, 2023.



A Hybrid Quantum Wasserstein GAN with F. Fuchs, 2023

Key: Similarity metric and possible applications



(1) Truncated signature MMD [BHLPW'20] (see more details later) 1st market gen

(2),(3) Higher rank sig-kernel (with kernel trick) [HLLMS'22] comp, [HI'23] regime det

(4) **Kernel Scoring Rules [IHLS'23]** significantly improved conditional market gen

The signature (and truncated signature)

- Recall the signature $S(x)$ of a path $x \in \mathcal{X}$ is given by $S(x) = (1, S^1(x), S^2(x), \ldots,)$, where

$$S^k(x) := \int_{0<t_1<\ldots<t_k<T} dx_{t_1} \otimes dx_{t_2} \otimes \ldots \otimes dx_{t_k}, \quad k \in \mathbb{N}.$$

- $S(x)$ has values in the tensor algebra $T((\mathbb{R}^d)) = \prod_{k=0}^{\infty} (\mathbb{R}^d)^{\otimes k}$

- This feature map, with a truncation at level $N \in \mathbb{N}$ was used in several of the aforementioned works: Given $N \in \mathbb{N}$, the truncated signature of order N is

$$S(x)^{\leq N} := (1, S(x)^1, \ldots, S(x)^N).$$

The signature kernel

- The associated kernel $k_{\mathrm{sig}} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is given by

$$k_{\mathrm{sig}}(x, y) = \sum_{k \geq 0} \langle S^k(x), S^k(y) \rangle_k$$

  where $\langle \cdot, \cdot \rangle_k$ is the inner product on $(\mathbb{R}^d)^{\otimes k}$.

- A "kernel trick" for k_{sig} via solving a Goursat PDE ($\Rightarrow$ no truncation) [SCFLY'21]
- Computation of higher rank sig kernel ($\Rightarrow$ incorporation of differences in filtration and pathwise effects /non-Markovianity) [H.SLLDL'21].

(Signature) Maximum Mean Discrepancy

Maximum mean discrepancy (MMD): μ and ν Borel probability measures on $\mathcal{X}$.

$$MMD_{\mathcal{G}}(\mu, \nu) := \sup_{f \in \mathcal{G}} \left| \int_{\mathcal{X}} f(x)\mu(dx) - \int_{\mathcal{X}} f(x)\nu(dx) \right|.$$

The $MMD_{\mathcal{G}}$ is a **metric** (point separating) if $\mathcal{G}$ is rich enough (e.g. a RKHS).

(Signature) Maximum Mean Discrepancy

Maximum mean discrepancy (MMD): μ and ν Borel probability measures on $\mathcal{X}$.

$$MMD_{\mathcal{G}}(\mu, \nu) := \sup_{f \in \mathcal{G}} \left| \int_{\mathcal{X}} f(x)\mu(dx) - \int_{\mathcal{X}} f(x)\nu(dx) \right|.$$

The $MMD_{\mathcal{G}}$ is a **metric** (point separating) if $\mathcal{G}$ is rich enough (e.g. a RKHS).

If $(\mathcal{H}, k)$ is a RKHS with kernel k and $\mathcal{G} := \{f \in \mathcal{H} : ||f||_{\mathcal{H}} \leq 1\}$ then

$$MMD^2_{\mathcal{G}}(\mu, \nu) = \mathbb{E}\left[k(X, X')\right] + \mathbb{E}\left[k(Y, Y')\right] - 2\mathbb{E}\left[k(X, Y)\right]$$

for $X, X' \sim \mu$ independent integrable r.v. and $Y, Y' \sim \nu$ independent integrable r.v.

(Signature) Maximum Mean Discrepancy

Maximum mean discrepancy (MMD): μ and ν Borel probability measures on $\mathcal{X}$.

$$MMD_{\mathcal{G}}(\mu,\nu) := \sup_{f\in\mathcal{G}} \left| \int_{\mathcal{X}} f(x)\mu(dx) - \int_{\mathcal{X}} f(x)\nu(dx) \right|.$$

The $MMD_{\mathcal{G}}$ is a **metric** (point separating) if $\mathcal{G}$ is rich enough (e.g. a RKHS).

If $(\mathcal{H}, k)$ is a RKHS with kernel k and $\mathcal{G} := \{f \in \mathcal{H} : ||f||_{\mathcal{H}} \leq 1\}$ then

$$MMD^2_{\mathcal{G}}(\mu,\nu) = \mathbb{E}\left[k(X,X')\right] + \mathbb{E}\left[k(Y,Y')\right] - 2\mathbb{E}\left[k(X,Y)\right]$$

for $X, X' \sim \mu$ independent integrable r.v. and $Y, Y' \sim \nu$ independent integrable r.v.

Characteristicness of the signature kernel implies an associated signature MMD, which can be used as a metric on path space [CO '18]. We choose $k(\cdot,\cdot)$ to be the (normalised) *signature* kernel and use $MMD_{\mathcal{G}}(\mu,\nu)$ as a hypothesis test:

$$H_0 : \quad \nu = \mu$$
$$H_1 : \quad \nu \neq \mu$$

The Signature MMD as a Two-Sample Test

The aforementioned MMD has been one of the most-employed tools in this context. However, only very few results are available on understanding how the signature kernel MMD functions as a statistical tool.

For (independent) samples $X_1, \ldots, X_n \sim \mathbb{P}_X$ and $Y_1, \ldots, Y_n \sim \mathbb{P}_Y$ there is an unbiased estimator $MMD_n^2(X_1, \ldots, X_n; Y_1, \ldots, Y_n)$ and strongly consistent. It for $n \to \infty$ it converges to the (theoretical) MMD

$$MMD_n^2(X_1, \ldots, X_n; Y_1, \ldots, Y_n) \longrightarrow_{n \to \infty} MMD_{\mathcal{G}}^2(\mathbb{P}_X, \mathbb{P}_Y) \quad \text{a.s.}$$

Scoring rules [GR07]

Definition (Scoring rule)

Let $\mathcal{P}$ be a convex class of measures on a space $(\mathcal{X}, \mathcal{A})$. A scoring rule $s : \mathcal{P} \times \mathcal{X} \to [-\infty, \infty]$ is any function s.t. $s(\mathbb{P}, \cdot)$ is $\mathcal{P}$-quasi integrable for all $\mathbb{P} \in \mathcal{P}$.

Definition (Properness)

Let $\int(\mathbb{P}, \mathbb{Q}) = \mathbb{E}_{y\sim\mathbb{Q}}[s(\mathbb{P}, y)]$ denote the expected scoring rule. The scoring rule $s : \mathcal{P} \times \mathcal{X} \to [-\infty, +\infty]$ is called proper (relative to the class $\mathcal{P}$) if $\int(\mathbb{P}, \mathbb{P}) \leq \int(\mathbb{Q}, \mathbb{P})$ for all $\mathbb{P}, \mathbb{Q} \in \mathcal{P}$.
It is called strictly proper if $\mathbb{Q} = \mathbb{P}$ is the unique minimiser.

Natural way to define a divergence: $\mathcal{D}_s(\mathbb{P}||\mathbb{Q}) = \int(\mathbb{Q}, \mathbb{P}) - \int(\mathbb{P}, \mathbb{P})$ for strictly proper s.

Signature kernel scores I

- For a given kernel k on $\mathcal{X}$, the associated kernel scoring rule s_k is given by

$$s_k(\mathbb{P}, y) = \mathbb{E}_{x,x'\sim\mathbb{P}}[k(x,x')] - 2\mathbb{E}_{x\sim\mathbb{P}}[k(x,y)]$$

- Denote by $\phi_{\text{sig}} : \mathcal{P}(\mathcal{X}) \times \mathcal{X} \to \mathbb{R}$ the kernel scoring rule associated to the signature kernel k_{sig}

Proposition 1 (IH.LS'23 Proposition 3.3)

For any compact $\mathcal{K} \subset \mathcal{X}$, ϕ_{sig} is a strictly proper kernel score relative to $\mathcal{P}(\mathcal{K})$, i.e. $\mathbb{E}_{y\sim\mathbb{Q}}[\phi_{sig}(\mathbb{Q}, y)] \leq \mathbb{E}_{y\sim\mathbb{Q}}[\phi_{sig}(\mathbb{P}, y)]$ for all $\mathbb{P}, \mathbb{Q} \in \mathcal{P}(\mathcal{K})$, with equality if and only if $\mathbb{P} = \mathbb{Q}$.

Signature kernel scoring rules II

- Given samples $\{x_i\}_{i=1}^m \sim \mathbb{P}$ and $y \in \mathcal{X}$, an unbiased estimator of ϕ_{sig} is given by

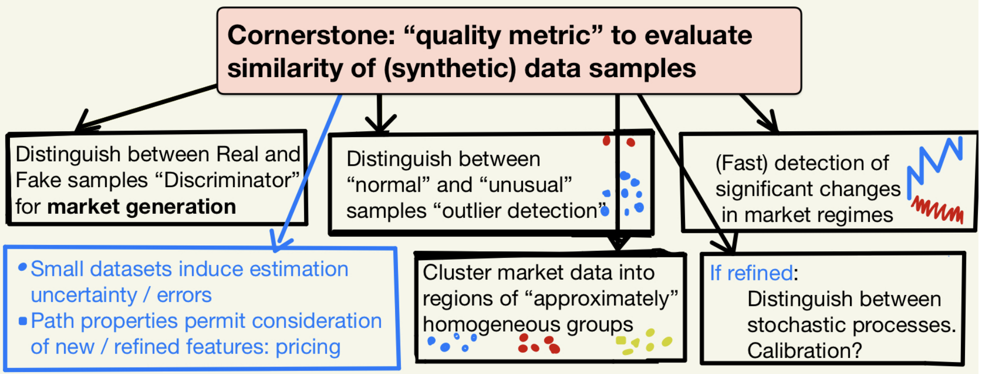
$$\hat{\phi}_{\text{sig}}(\mathbb{P}, y) = \frac{1}{m(m-1)} \sum_{i \neq j} k_{\text{sig}}(x_i, x_j) - \frac{2}{m} \sum_{i=1}^{M} k_{\text{sig}}(x_i, y)$$

- Note that

$$\mathcal{D}_{\text{sig}}(\mathbb{P}, \mathbb{Q})^2 = \phi_{\text{sig}}(\mathbb{P}, \mathbb{Q}) + \mathbb{E}_{y,y' \sim \mathbb{Q}}[k_{\text{sig}}(y, y')],$$

that is, we recover the (squared) signature maximum mean discrepancy (MMD).

Key: Similarity metric and possible applications

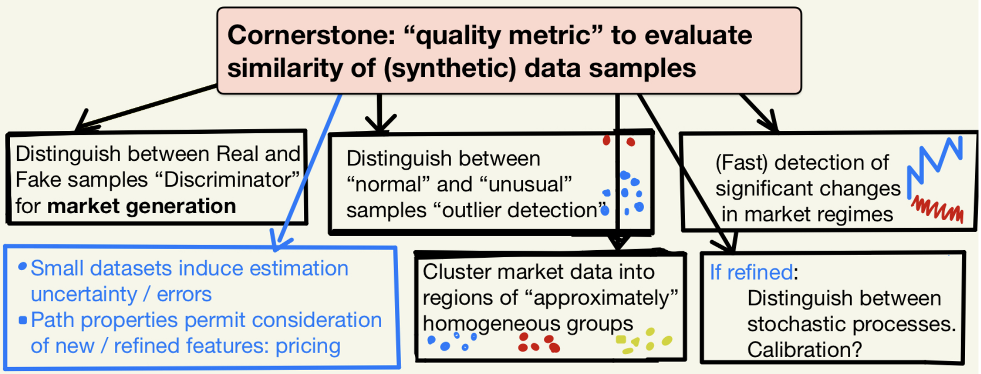


(1) Truncated signature MMD [BHLPW'20] (see more details later) 1st market gen

(2),(3) Higher rank sig-kernel (with kernel trick) [HLLMS'22] comp, [HI'23] regime det

(4) **Kernel Scoring Rules [IHLS'23]** Link to paper, Link to Code

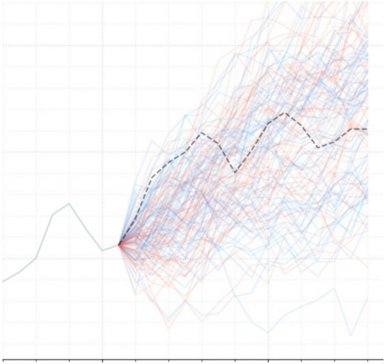
Dealing with uncertainty


Cornerstone: "quality metric" to evaluate similarity of (synthetic) data samples
Distinguish between Real and Fake samples "Discriminator" for **market generation**
Distinguish between "normal" and "unusual" samples "outlier detection"
(Fast) detection of significant changes in market regimes
• Small datasets induce estimation uncertainty / errors
• Path properties permit consideration of new / refined features: pricing
Cluster market data into regions of "approximately" homogeneous groups
If refined: Distinguish between stochastic processes. Calibration?

Challenges with real (historical) data:

Task "Market Generator": Generate (synthetic) **TrainingData** for the network, such that performance is optimized when **TestData** = Market Data.

The challenge that market generators face is to produce **genuinely new data** samples that in aggregation have the same distribution as the test data they will later be exposed to, **though we only see one realisation of the path**. (Image by Zach Issa:)



Reality only happens once.

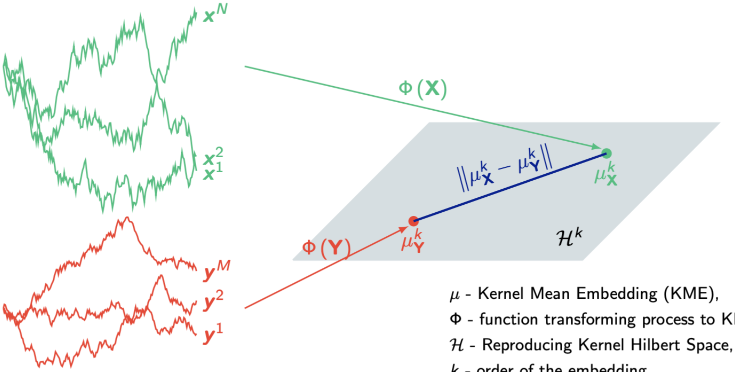
Challenges with real (historical) data:

(1) **Data availability:** In many real situations, there is very limited data available for training/estimation). $\Rightarrow$ small datasets may induce higher estimation errors.

(2) **Computational limitations:** Some limitations imposed on datasets (real & synthetic) by computational- and memory considerations.

(3) **Data changes over time:** Markets are heteroskedastic and non-stationary $\Rightarrow$ may be a possible reason for (1) limitations in (training) data availablibIty

The Signature MMD as a Two-Sample Test



μ - Kernel Mean Embedding (KME),
Φ - function transforming process to KME,
$\mathcal{H}$ - Reproducing Kernel Hilbert Space,
k - order of the embedding.

Image: Andrew Alden

The Signature MMD as a Two-Sample Test

- Each collection of paths $\{\boldsymbol{x}^k\}_{k=1}^r, \{\boldsymbol{y}^k\}_{k=1}^m$ results in a different distance value.
- Ambiguity in feature vectors.

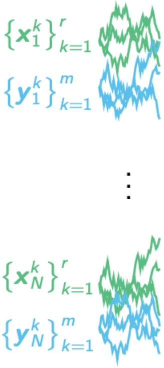


Image: Andrew Alden

The Signature MMD as a Two-Sample Test

The aforementioned MMD has been one of the most-employed tools in this context. However, only very few results are available on understanding how the signature kernel MMD functions as a statistical tool.

For (independent) samples $X_1, \ldots, X_n \sim \mathbb{P}_X$ and $Y_1, \ldots, Y_n \sim \mathbb{P}_Y$ there is an unbiased estimator $MMD_n^2(X_1, \ldots, X_n; Y_1, \ldots, Y_n)$ and strongly consistent. It for $n \to \infty$ it converges to the (theoretical) MMD

$$MMD_n^2(X_1, \ldots, X_n; Y_1, \ldots, Y_n) \longrightarrow_{n\to\infty} MMD_{\mathcal{G}}^2(\mathbb{P}_X, \mathbb{P}_Y) \quad \text{a.s.}$$

The Signature MMD Two-Sample Test (Base Case, truncated signatures)

To asses whether a generative model generate "realistic" paths, sample real paths $Y_1, \dots, Y_n$, for some $n \in \mathbb{N}$, and sample generated paths $X_1, \dots, X_n$ and apply the two-sample test in [Chevyrev and Oberhauser '18]. Signature-based MMD test statistic $T(X_1, \dots, X_n; Y_1, \dots, Y_n)$ where $k(\cdot, \cdot)$ is the *signature kernel*:

$$T(X_1, \dots, X_n; Y_1, \dots, Y_n) := \tfrac{1}{n(n-1)} \sum_{i,j;i \neq j} k(X_i, X_j) - \tfrac{2}{n^2} \sum_{i,j} k(X_i, Y_j) + \tfrac{1}{n(n-1)} \sum_{i,j;i \neq j} k(Y_i, Y_j),$$

Then, given a confidence level $\alpha \in (0, 1)$, compute $c_\alpha(n) := 4\sqrt{-n^{-1} \log \alpha}$ (**threshold**). Generated paths are **realistic with a confidence** α if $T_n^2 < c_\alpha(n)$.

The Signature MMD Two-Sample Test (Base Case, truncated signatures)

To asses whether a generative model generate "realistic" paths, sample real paths $Y_1, \ldots, Y_n$, for some $n \in \mathbb{N}$, and sample generated paths $X_1, \ldots, X_n$ and apply the two-sample test in [Chevyrev and Oberhauser '18]. Signature-based MMD test statistic $T(X_1, \ldots, X_n; Y_1, \ldots, Y_n)$ where $k(\cdot, \cdot)$ is the *signature kernel*:

$$T(X_1, \ldots, X_n; Y_1, \ldots, Y_n) := \frac{1}{n(n-1)} \sum_{i,j;i \neq j} k(X_i, X_j) - \frac{2}{n^2} \sum_{i,j} k(X_i, Y_j) + \frac{1}{n(n-1)} \sum_{i,j;i \neq j} k(Y_i, Y_j),$$

Then, given a confidence level $\alpha \in (0, 1)$, compute $c_\alpha(n) := 4\sqrt{-n^{-1} \log \alpha}$ (**threshold**). Generated paths are **realistic with a confidence** α if $T_n^2 < c_\alpha(n)$.
Note how threshold depends on the number n of samples considered.

The Signature MMD Two-Sample Test (Base Case, truncated signatures)

To asses whether a generative model generate "realistic" paths, sample real paths $Y_1, \ldots, Y_n$, for some $n \in \mathbb{N}$, and sample generated paths $X_1, \ldots, X_n$ and apply the two-sample test in [Chevyrev and Oberhauser '18]. Signature-based MMD test statistic $T(X_1, \ldots, X_n; Y_1, \ldots, Y_n)$ where $k(\cdot, \cdot)$ is the *signature kernel*:

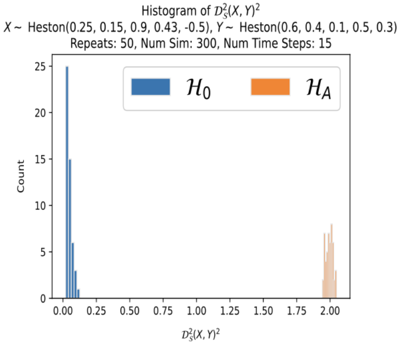
$$T(X_1, \ldots, X_n; Y_1, \ldots, Y_n) := \frac{1}{n(n-1)} \sum_{i,j; i \neq j} k(X_i, X_j) - \frac{2}{n^2} \sum_{i,j} k(X_i, Y_j) + \frac{1}{n(n-1)} \sum_{i,j; i \neq j} k(Y_i, Y_j),$$

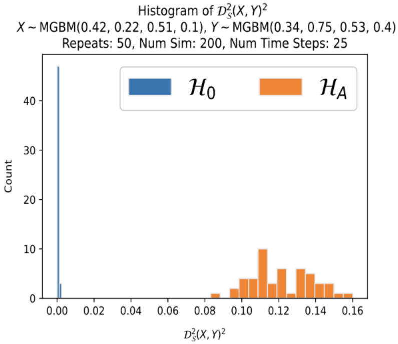
Then, given a confidence level $\alpha \in (0,1)$, compute $c_\alpha(n) := 4\sqrt{-n^{-1} \log \alpha}$ (**threshold**). Generated paths are **realistic with a confidence** α if $T_n^2 < c_\alpha(n)$.
Note how threshold depends on the number n of samples considered.

This base-case has been considered in "Signature-based validation of real-world economic scenarios" [Andres, Boumezoued, Jourdain '23] for $n \neq m$.

The Signature MMD as a Two-Sample Test

- Perform a two-sample test: $\mathcal{H}_0\colon \mathbb{P}_{X|\mathcal{F}_X} = \mathbb{P}_{Y|\mathcal{F}_Y}$ and $\mathcal{H}_A\colon \mathbb{P}_{X|\mathcal{F}_X} \neq \mathbb{P}_{Y|\mathcal{F}_Y}$.

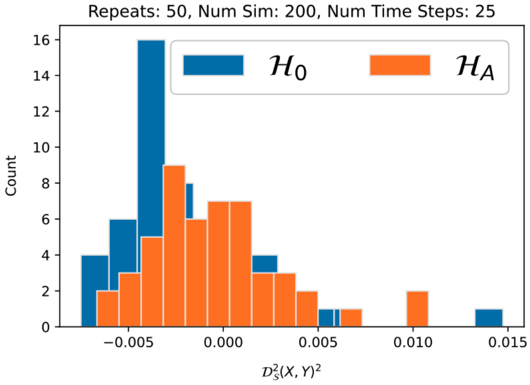




- **Empirical estimate** of 2$^{\text{nd}}$-order MMD has a long **run-time**.

The Signature MMD as a Two-Sample Test

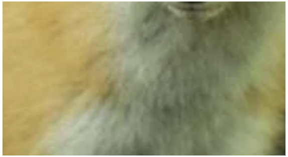
Claim: In this setting, robustification comes (in some parts) naturally through small sample sizes of data:



Path scalings and type II errors

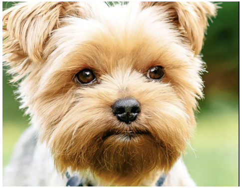
1. Path scalings can help reduce the incidence of type II error for small batch sizes.
2. Since $\varphi(x) = x^k/k!$ is increasing in x, (larger) scaling has the effect of increasing the numerical size of higher-order signature terms.
3. **Issue:** what is the (most) appropriate scaling to tell apart two different stochastic proceses?





Path scalings to deal with type II errors

1. Path scalings can help reduce the incidence of type II error for small batch sizes.
2. Since $\varphi(x) = x^k/k!$ is increasing in x, (larger) scaling has the effect of increasing the numerical size of higher-order signature terms.
3. **Issue:** what is the (most) appropriate scaling to tell apart two different stochastic proceses?



Path scalings and type II errors

1. Since $\varphi(x) = x^k/k!$ is increasing in x, (larger) scaling has the effect of increasing the numerical size of higher-order signature terms.
2. Can help reduce the incidence of type II error for small batch sizes
3. **Issue:** what is the (most) appropriate scaling to tell apart two different stochastic proceses?

Answer: May need multiple scalings for use-cases.

Path scalings and type II errors

1. Since $\varphi(x) = x^k/k!$ is increasing in x, (larger) scaling has the effect of increasing the numerical size of higher-order signature terms.
2. Can help reduce the incidence of type II error for small batch sizes
3. **Issue:** what is the (most) appropriate scaling to tell apart two different stochastic proceses?

Answer: May need multiple scalings for use-cases.
Solution: An adversarial (GAN) structure may be needed to run through the most challenging scalings to create realistic data (ongoing Zach Issa ...)

Dealing with Uncertainty: Pricing (path dependent) payoffs with the help of the MMD (link) with A. Alden, C. Ventre, G. Lee, 2022.

Dealing with Uncertainty: Pricing (path dependent) payoffs with the help of the MMD (link) with A. Alden, C. Ventre, G. Lee, 2022.

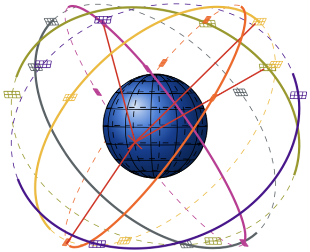
Solution:

Estimate simultaneously from multiple angles to sharpen perspective (estimate MMD to several reference points/models)

Pricing path dependent payoffs with the help of the MMD

To obtain the price of a derivative priced under the stochastic process $\mathbb{Y}$:

- Select N base processes as reference models (satellites).
- Compute the distance from $\mathbb{Y}$ to each base process (distance to satellites).
- Use these distances to price (GPS location).



Adapted from Walcott, K. (2012) *Three-dimensional graphics with PGF/TiKZ* [67] and Trzeciak, T. (2008) *Example: Stereographics and cylindrical map projections* (`https://texample.net/tikz/examples/map-projections/`) [64]

Image: Andrew Alden

Pricing path dependent payoffs with the help of the MMD

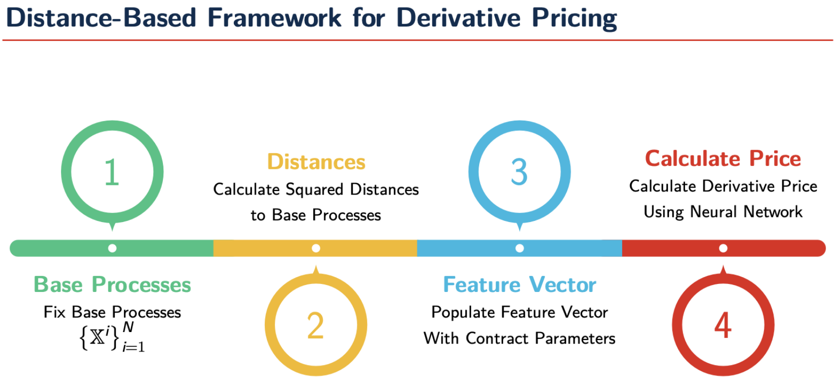


Image: Andrew Alden

Introducing robustness into Deep Hedging/ Deep Trading with the help of pathwise similarity metrics (MMD)

Introducing robustness into Deep Hedging/ Deep Trading with the help of pathwise similarity metrics (MMD)

Robust Hedging Gans (link) with Y. Limmer, 2023.
Signature Trading: A Path-Dependent Extension of the Mean-Variance Framework with Exogenous Signals with O. Futter and M. Wiese, 2023 (link).

Introducing robustness into Deep Hedging/ Deep Trading with the help of the MMD

Robust Hedging Gans, with Y. Limmer, 2023 (link).
Signature Trading: A Path-Dependent Extension of the Mean-Variance Framework with Exogenous Signals with O. Futter and M. Wiese, 2023 (link).

Robust Hedging Gans

(1) **Data availability:** In many real situations, there is very limited data available for training/estimation). $\Rightarrow$ small datasets may induce higher estimation errors.

(2) **Computational limitations:** Some limitations imposed on datasets (real & synthetic) by computational- and memory considerations (examples later).

(3) **Data changes over time:** Markets are heteroskedastic and non-stationary
$\Rightarrow$ may be a possible reason for (1) limitations in (training) data availabliblty
$\Rightarrow$ large changes in the data may require retraining / changing the network, **but...**

...small changes in data & estimation errors should not throw the application off track.

Needed: Appropriate (smooth) robustification of tasks towards small changes in input

Robust Hedging Gans

In the context hedging and trading strategies, the (pathwise) MMD can be helpful to introduce a smooth ambiguity-aversion effect into the hedging / trading objective.

Introducing robustness into DH/DT with the MMD

The original objective of a trading can be (most generally) given as

$$\max_{(\phi_t)_{t\in[0,T]}} \mathbb{E}\Big[U(V_T)\Big], \quad \text{where} \quad V_T = \sum_{m=1}^{d} \int_0^T \phi_t^m dS_t^m,$$

where V_T is the terminal profit and loss of our trading strategy.

Introducing robustness into DH/DT with the MMD

The original objective of a trading can be (most generally) given as

$$\max_{(\phi_t)_{t\in[0,T]}} \mathbb{E}\Big[U(V_T)\Big], \quad \text{where} \quad V_T = \sum_{m=1}^{d} \int_0^T \phi_t^m dS_t^m,$$

where V_T is the terminal profit and loss of our trading strategy.
Robustify: Extend the set of possible models, but restrict this set to make it feasible.
Often robustification entails allowing alternative models within a δ-ball $B_\delta(\mathbb{P})$ around $\mathbb{P}$.

$$\max_{(\phi_t)_{t\in[0,T]}} \min_{\mathbb{Q}\in B_\delta(\mathbb{P})} \mathbb{E}^{\mathbb{Q}}\Big[U(V_T)\Big],$$

Implicit: All alternative models $\mathbb{Q} \in B_\delta(\mathbb{P})$ are equally likely to materialize.

Introducing robustness into DH/DT with the MMD

The original objective of a trading can be (most generally) given as

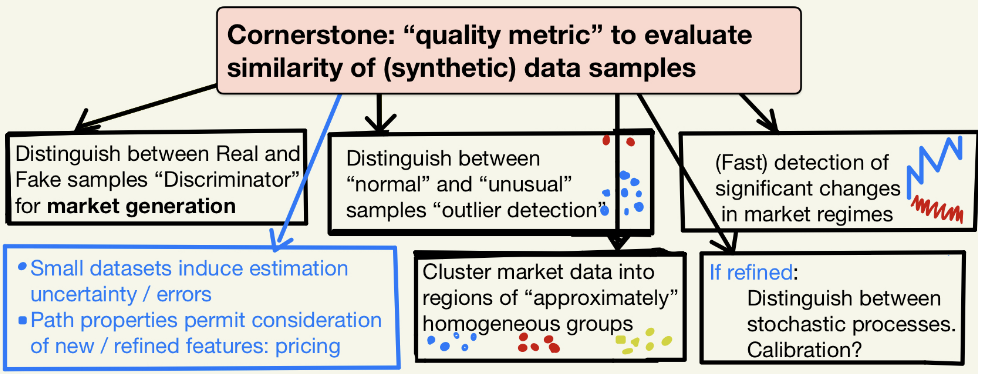
$$\max_{(\phi_t)_{t\in[0,T]}} \mathbb{E}\Big[U(V_T)\Big], \quad \text{where} \quad V_T = \sum_{m=1}^{d} \int_0^T \phi_t^m dS_t^m,$$

where V_T is the terminal profit and loss of our trading strategy.
Robustify: Extend the set of possible models, but restrict this set to make it feasible.
Our approach to robustification: Smoother version of model ambiguity via MMD

$$\max_{(\xi_t)_{t\in[0,T]}} \min_{\mathbb{Q}\in\mathcal{Q}} \mathbb{E}^{\mathbb{Q}}\Big\{U(V_T) + \tfrac{1}{\eta} d(\mathbb{P},\mathbb{Q})\Big\},$$

where $d(\mathbb{P},\mathbb{Q})$ denotes the (MMD) distance of a realised model $\mathbb{Q}\in\mathcal{Q}$ to our reference model $\mathbb{P}$ and $\frac{1}{\eta}$ is a scaling parameter representing the investor's aversion to model ambiguity. Taking $d(\cdot,\cdot)$ as the Sig-MMD allows to take a fully pathwise perspective.

Key: Similarity metric and possible applications



(1) Truncated signature MMD [BHLPW’20] (see more details later) 1st market gen

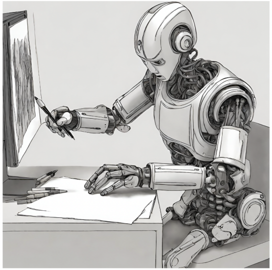
(2),(3) Higher rank sig-kernel (with kernel trick) [HLLMS’22] comp, [HI’23] regime det

(4) **Kernel Scoring Rules [IHLS’23]** significantly improved conditional market gen

Non-Adversarial Training of Neural SDEs with Signature Kernel Scores: Issa, H., Lemercier, Salvi '23

- Components:
  - A generator $G_\theta : \Theta \times \mathcal{Z} \to \mathcal{X}$
  - A discriminator $D : \mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{X}) \to \mathbb{R}$, and
  - A training procedure.

- In the case where the discriminator is parametrized, $D = D_\phi$, and training objective is adversarialized (which it is not), we recover the classic GAN:

(Generator)   (Discriminator)

Non-Adversarial Training of Neural SDEs with Signature Kernel Scores: Issa, H., Lemercier, Salvi '23

- Components:
  - A generator $G_\theta : \Theta \times \mathcal{Z} \to \mathcal{X}$
  - A discriminator $D : \mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{X}) \to \mathbb{R}$, and
  - A training procedure.
- In the case where the discriminator is parametrized, $D = D_\phi$, and training objective is adversarialized, we recover the classic GAN [GAMX+'20]
- Goal: a plug-and-play pipeline for training market gen. on path space, which is
  - mesh-free,
  - stable,
  - memory efficient,
  - easily able to be conditionalized, and
  - can handle inputs/outputs taking values in infinite-dimensional spaces

Related work: Application to MG

[BHLPW'20]: VAE-based conditional generative model using the truncated signature MMD. Required signature inversion.

[K'21]: SDE-GAN, adversarialising the training objective via a Neural CDE. Class conditioning examples.

[NSzSXW'21]: Sig-Wasserstein Gans for Time-series Generation. Training Log-RNN, AR-FNN generator against the Sig-Wasserstein distance.

- Again uses truncation of the signature
- Conditioning examples make use of on relationship between past truncated signature and future which is hard to recreate in practice

[WWPKBMB'21]: Multi-asset spot and option market simulation, 2021. Not mesh-free, and not conditional (mentioned as a future extension)

Non-adversarial training of Neural SDEs

- The training objective is

$$\min_{\theta} \mathcal{L}(\theta) \quad \text{where} \quad \mathcal{L}(\theta) = \mathbb{E}_{y \sim \mathbb{P}_{X^{\text{true}}}}[\phi_{\text{sig}}(\mathbb{P}_{X^\theta}, y)] + \lambda \|\theta\|_{L_2}$$

- The training objective is minimised when $\mathbb{P}_{X^\theta} = \mathbb{P}_{X^{\text{true}}}$ (since striclty proper sr)
- Training procedure can be summarized by

$$\textbf{Generator:} \; X^\theta \approx \text{SDESolve}(\theta),$$
$$\textbf{Discriminator:} \; \mathcal{L}(\theta) \approx \text{PDESolve}(X^\theta, X^{\text{true}}).$$

- Both procedures are able to be backpropagated through
  - **Generator:** Through the SDE solver
  - **Discriminator:** Via solving another system of adjoint PDEs [LSCBDL'21]

Experiments: rBergomi model

Goal: Train a Neural SDE to learn the rough stochastic volatility model

$$dy_t = -\frac{1}{2}V_t dt + \sqrt{V_t}dW_t \quad \text{where} \quad d\xi_t^u = \xi_t^u \eta\sqrt{2\alpha+1}(u-t)^\alpha dB_t.$$

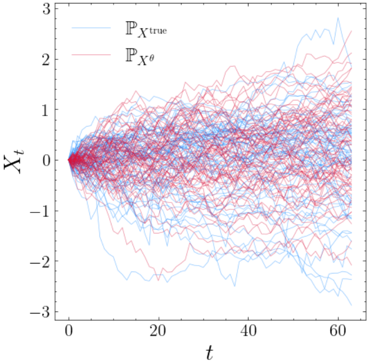
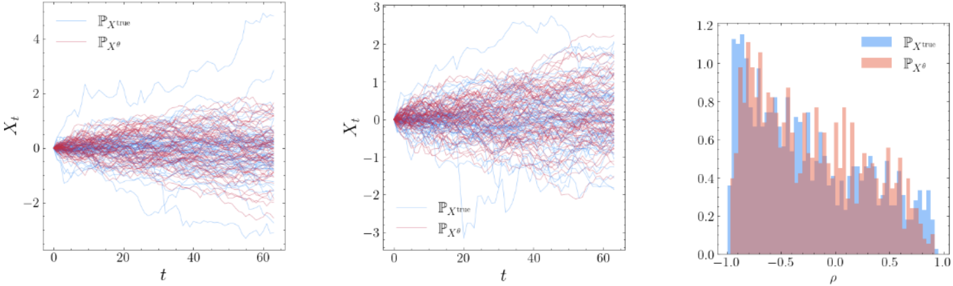


Figure 1: Neural SDE trained with ϕ_{sig} where $X^{\text{true}} \sim$ rBergomi(η, ρ, H).

Experiment: Currency pairs EUR/USD and USD/JPY



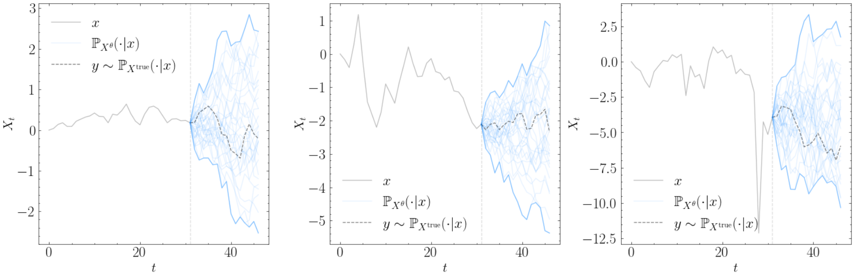
(a) EUR/USD **(b)** USD/JPY **(c)** Correlation coefficients

Figure 2: Neural SDE trained with ϕ_{sig}, EUR/USD and USD/JPY price pairs

Experiments: Conditional generation EURUSD

- Conditioning variables are time-augmented EUR/USD trajectories $\mathbb{Q} \sim x : [t_0 - dt, t_0] \to \mathbb{R}^2$
- Target variables: future trajectories $\mathbb{P}(\cdot|x) \sim X^{\text{true}} : [t_0, t_0 + dt'] \to \mathbb{R}^2$
- Encode conditioning variables via the order 5 log-signature of the input trajectories
- Train to minimise the conditional expected signature kernel score
- Many hyperparameters to consider... in general, path scaling is the most important (same is true for previous case!)

Experiments: Conditional generation EURUSD

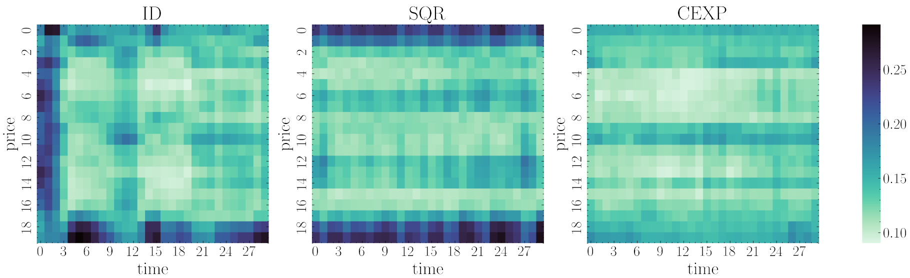


Given a conditioning path $x \sim \mathbb{Q}$, the generator provides (in blue) the conditional distribution $\mathbb{P}_{X^\theta}(\cdot|x)$. The dotted line gives the true path. $y \sim \mathbb{P}_{X^{\text{true}}}(\cdot|x)$.

Resultant path is often captured in the envelope of the associated conditional distribution

Experiment: Limit order books

Train a Neural SPDE model on NASDAQ LOB data, composing the signature kernel with three different SE-T type kernels



KS test average scores for each spatiotemporal marginal, 100 runs, NASDAQ data.

Thank you for your attention!