Generating drawdown-realistic markets using path signatures

Emiel Lemahieu[1][2][3], Kris Boudt[1], Maarten Wyns[2]

Ghent University, InvestSuite

Abstract

A drawdown is a price fall relative to the historical maximum. Simulation of drawdown-realistic markets is a difficult problem as this path dependent measure depends on the drift, volatility and autocorrelation of the underlying process. But it is an important task, for instance in pricing drawdown insurance options or portfolio drawdown optimization. Handcrafting a parametric process that fully encapsulates drawdown parameters in its dynamics has not been done in literature. Mostly researchers have referred to standard processes such as Brownian motion, where the drawdown distribution can be derived from Lévy's theorem. We advocate an essentially non-parametric approach from machine learning, combining a variational autoencoder generative model with a path signature-based drawdown reconstruction loss function. Machine learning requires a system of differentiable equations that can provide numerical simulations by iteration or learning. Drawdown as an evaluation metric is highly non-trivial as its complexity depends on path length, and as it is discontinous in the continously differentiability sense which impedes evaluating the impact of a change in a parameterised path on its drawdown. By expressing drawdown as essentially a non-linear dynamic system, we propose to resolve this smoothing by approximating drawdown using a path signature's universality property. We appreciate that this gives a mathematically non-trivial, non-commutative exponential alternative for smoothed, differentiable expressions of drawdown that gives one leeway to simulate drawdown-realistic markets by including a drawdown evaluation metric in the learning objective.

Keywords: Path functions, Signatures, Drawdowns
MSC: 91B28, 91B84

1. Introduction

1.1. Problem setting

Market generators - or generative machine learning models with the specificity of modeling financial markets like stock prices, order books, volatility surfaces, etc. - have gained popularity in recent years (Wiese et al. [1], Koshiyama et al. [2], Cont et al. [3], Bergeron et al. [4]). Generative machine learning in general has gained widespread attention with the successes of deep fake generative adversarial networks (GAN) in images and large-scale language models like ChatGPT

[1]Ghent University, Sint-Pietersplein 6, 9000 Gent, Belgium
[2]InvestSuite, `www.investsuite.com`
[3]Corresponding author at emiel.lemahieu@ugent.be

in text. However, the technology is never more important than the problem it claims to solve better than existing solutions. With market generators, arguably the main advantage over existing simpler solutions is being able to simulate realistic financial time series without having to prespecify the dynamics, i.e. handcraft a data generating process (DGP) a priori. With realistic we think about distributional divergences (difference in moments or max mean discrepancy (MDD), entropy or Kullback-Leibler divergence, Wasserstein distance,. . .) between the real and generated return distributions, and distances between their autocorrelation functions and tail indices. Some striking results in this regard have been found in Wiese et al. [1], Cont et al. [3], Buehler et al. [5] and Vuletić et al. [6].

One of the problems we claim it can help solve is closely linked to this non-parametric nature. In a context where the drawdown of the simulated trajectories matter, e.g. when pricing drawdown options for max loss insurance (Carr et al. [7],Atteson and Carr [8]) or controlling for drawdown in a portfolio optimization context (Chekhlov et al. [9], [10]), the link between the underlying data generating process and the drawdown is far from obvious. The drawdown is the difference at any point in time between a price and its historical maximum up until that point. For a price path (in levels) $S : [0, T] \rightarrow \mathbb{R}$ define drawdown function Ξ as:

$$\xi_t = \Xi(S)_t = \max(\max_{k<t}(S_k) - S_t, 0) \tag{1}$$

It is clear that the drawdown is a non-trivial function of the underlying DGP and captures at least three dimensions. Firstly, the drift of S or essentially the deterministic tendency of achieving a new max. Secondly, the dispersion in S or its volatility, albeit in a negative deviation sense, which is in essence the probability of a loss vis-a-vis this monotonic drift. Thirdly, and often overlooked, the autocorrelation of losses or the probability that the nonzero drawdown will persist, i.e. the duration of the drawdown. The link between a DGP that captures these three components and its drawdown distribution is generally unknown and depends on the process. In the drawdown literature, one often assumes (geometric) Brownian motion and uses Lévy's theorem (Lévy [11]) to find the distribution of drawdown:

$$(M_t - B_t, M_t) \approx (B_t, L(B_t)) \tag{2}$$

where B_t is a Brownian motion at t, M_t is its running maximum and $L(B)_t$ is the so-called local time of B. This yields closed-form expressions of the distribution $\mathcal{P}(\xi)$ when S is a Brownian motion (e.g. see Douady et al. [12] and more concisely Rej et al. [13]). However, martingale processes like Brownian motion do not exhibit return autocorrelation and one expects a sequence of consecutive losses to result in larger drawdown scenarios, such that martingales can merely be optimistic lower bounds on worst case drawdown scenarios (also stressed in Rej et al. [13]).

One way to cope with this issue is to simply not embed these drawdown parameters in the DGP or derive its distribution from said DGP assumptions, but to *rely on a fundamentally non-parametric simulation approach*. We show that one can rather rely on a market generator to abstract both the DGP and the link between DGP and drawdown distribution, and learn to reproduce this distribution in a Monte Carlo. This Monte Carlo could be used to devise strategies that control drawdown (such as drawdown optimization) or as a simulation engine for pricing drawdown insurance in a fully non-parametric way.

The main issue with the drawdown function, however, is that at first glance it looks unsuited for learning:

- **Differentiability**: as we will discuss below, Ξ is non-differentiable w.r.t. a parametrized path S_θ. At first glance it looks impossible to change the DGP params θ, e.g. stretch, squeeze, tilt a generated path towards a level of drawdown by changing its parametrized moments, i.e. by including 'feedback' on $\frac{\delta(\Xi(S_\theta))}{\delta(\theta)}$.

- **Complexity**: evaluating the maximum of a vector of n numbers takes n operations or $O(n)$. Naively evaluating the running maximum takes $n(n+1)/2$ or $O(n^2)$ operations. If one would use a smoothed approximation of the maximum (like smooth normalized exponentials [14]) to resolve non-differentiability, this would be computationally prohibitive (especially for long n paths). Moreover, accuracy of such naive exponential smoothing would rely on the scale of S and the variation around the local maxima.

1.2. Contribution

The main contribution of this paper is that these issues can be jointly overcome by approximating the drawdown of a path by using linear regression on its signature terms.

The signature of a path is a mathematically principled way of representing a path. It is a sequence of ordered coefficients of the path called the iterated integrals that fully describe the path. In our context it serves as a feature map, which is ordered in the sense that it provides a global and increasingly local or more granular description of factorial decaying variation, of the path. It is fully describing it in the sense that it provides a bijective mapping between path and signature (up to tree-like equivalences). Next to this uniqueness property, we rely on the universality of the feature map in the next sections to introduce the approximation. In plain terms, universality means that linear functions on the feature map basis approximate non-linear functions on the original features basis. Think about Taylor's approximation theorem that states that the power series of a variable can approximate a point in a polynomial by essentially using derivatives in ordered (factorial decaying) linear coefficients. Another example is neural networks, that by concatenating affine linear weighting functions and non-linear activation functions learn a basis to approximate complex functions by linear function in the final layer. The link with these universal approximation methods, Stone-Weierstrass theorem and signatures will be briefly touched upon in the next sections as well.

1.3. Related work

The potential benefits of generative machine learning for modeling financial time series have been early recognized by Kondratyev and Schwarz [15] and Henry-Labordere [16], and Wiese et al. [1], who first applied restricted Boltzmann machines (Smolensky [17]) and GANs (Goodfellow et al. [18]) respectively to financial sequences data. Since those papers, a host of use cases Cuchiero et al. [19] , Ni et al. [20], Li et al. [21] , Storchan et al. [22] , Benedetti [23] , Xu et al. [24], Pardo and López [25], Buehler et al. [26] , Ni et al. [27], Pfenninger et al. [28], Rosolia and Osterrieder [29] , Koshiyama et al. [2] , van Rhijn et al. [30], Marti et al. [31], Coyle et al. [32], Wiese et al. [33] , Lezmi et al. [34] , Wang [35], Buehler et al. [5] , Cont et al. [3], Fung [36] , Frandsen [37], Bergeron et al. [38], Ning et al. [39] and Vuletić et al. [6] have been proposed that include time series forecasting, trading strategy backtesting, hands-off volatility surface modeling, option pricing and *deep hedging* and more.

Closest related to this work is the paper of Buehler et al. [5] who also use the combination of variational autoencoder and signatures, but with the general aim of reproducing stylized facts of financial time series. However, they learn signatures in the input space and then output signature values, which implies that deploying their model requires scalable inversion of signatures back to paths, which is far from trivial. This paper's model does not output signatures but only uses them in the loss evaluation step. Moreover, much work on market generators has revolved around adjusting the loss function such that desired features of the timeseries are reproduced. This is very close to our proposed approach. Cont et al. [3] evaluates the tail risk (value-at-risk and expected shortfall) of the simulations to evaluate their adequacy. Recently, Vuletić et al. [6] included P&L similarity and Sharpe ratios in the loss function to increase the financial realism of the generated scenarios. This paper builds on top of that. Drawdown is a similar metric that in most financial contexts would be a useful feature to reproduce, because it captures both P&L and autocorrelation, but in fact is in some financial contexts the most important metric, such as portfolio drawdown control, optimization, or drawdown insurance. Think about asset managers who want to backtest their strategies which get closed down if their strategy breaches a certain threshold. For these applications drawdown is not a *nice-to-have* but a necessary feature to reproduce. However, as a path functional, rather than a function on a P&L distribution like a tail loss or Sharpe ratio, drawdown is much more difficult to evaluate. For that we introduce the approximation trick.

1.4. Organization of results

The remainder of the paper is structured as follows. The next section 2 describes drawdown as a non-linear dynamic system driven by the underlying price path, and introduces the signature approximation. We run numerical experiments on Brownian and fractional Brownian simulated data, and empirical real-world data.

In section 3 we discuss the implications of this approximation for a market generator architecture. We generate drawdown-realistic markets by including the path signature-based drawdown approximation in the drawdown reconstruction cost objective term. We find that the drawdown distribution is only adequately learned with the additional loss term and we get realistic drawdown scenarios. Finally, the appendix gives a concise introductory overview of the basic building blocks from rough path theory that were used throughout this paper.

Future work could use this architecture for pricing drawdown insurance options through a non-parametric Monte Carlo or controlling portfolio drawdown in a fully non-parametric way, as per learning on data rather than calibrating a known DGP on data.

2. Drawdown as a non-linear dynamic system and the signature approximation

2.1. Drawdown as a non-linear system

Assume the price path $S : [0,T] \to \mathbb{R}$ is a piecewise linear continuous path, such as daily stock prices or NAV values. Firstly, we need to remark the following: while from (1) we know the max operator makes Ξ non-differentiable (also see Eq. 4 below), the variation in Ξ is bounded by the variation in the paths. When taking two bounded[4]- e.g. finite-variance - paths, the distance between their maximum values is bounded by a norm on the distance between their path values:

$$|\max_{0<i<T}(S^1_i) - \max_{0<j<T}(S^2_j)| \leq \max_{i,j\in[0,T]} |S^1_i - S^2_j| = ||S^1 - S^2||_\infty \tag{3}$$

Or in words, the distance between two maxima is capped by the maximum distance between any two values on path S^1 and S^2, which can be written as a norm. More specifically, the maximum is Lipschitz-K continous, with distance inf-norm and K=1. Similar arguments can be made for

[4]For the definition of boundedness in the *p-variation* sense see [40].

ξ, i.e. apply max twice and a linear combination of two piecewise linear paths is Lipschitz continuous. Now the question is what this more general continuity, a more general smoothness assumption, would imply for *local approximation*. For this, we have to think of the differentials of ξ and S,i.e. treat drawdown as a dynamic system, which is unnatural as non-differentiability implies $\frac{d\xi_t}{dS_t}$ is *not* a continuous function, as one would expect for Taylor series-like expansions. Consider the following dynamic system (also see [12]):

$$d\xi_t = f(\xi_t, dS_t) = \begin{cases} -dS_t, \xi_t > 0 \\ \max(0, -dS_t), \xi_t = 0 \end{cases} \tag{4}$$

Depending on the current level of drawdown the effect of a price change is either linear (-1) or none (0). This is exactly what creates the non-differentiability. One realizes that because of the conditionality, direct integration of the right-hand-side, as a path integral, does not make sense. Nor does a smooth approximation like a Taylor series make sense as the derivatives do not exist. However, the differentials $|d\xi_t| \leq |dS_t|$ are clearly bounded. This makes local approximation still feasible: it does make sense to do an interpolation between the -1 and 0 effects, and because of this boundedness one would not be arbitrarily off. For instance, one could assume linear dynamics but with a stochastic component derived from the average time in drawdown, i.e. the probability of a -1 effect. But the solution of this SDE would still be based on the full path, i.e. be path dependent. This is the second motivation for rough path theory, that there is essentially a bounded function linking ξ and S described by intervals, or non-commutative sequences of increments of S, rather than increments (also see Friz and Hairer [41], and e.g. the discussion on the Ito-Lyons map and *previsible integrands* in the introduction).

Rough path theory is concerned with local approximation of (smooth) functions on (rough) paths by means of the path signature. A path signature is the infinite series of iterated integrals that can be constructed on the path. The definition of controlled differential equations, smooth functions on rough paths and signatures is included in appendix (section 4), but for brevity removed here.

2.2. Signature approximation of drawdown

One of the foremost insights from rough path theory is that the signature linearizes the relationship between ξ and $\Phi(S)$, also known as the universality of the signature. In addition, contrary to traditional Taylor expansion where we take the derivatives of ξ as coefficients of the power series of S, we are not necessarily interested in the explicit values of f (or $f^{\otimes n}$) for any value of ξ_t. Also see the appendix for a brief recap on the link between path integrals and Taylor expansion. We are just interested in:

$$\hat{\xi}(N)_t = \xi_0 + \sum_{n=1}^{N} L_n \underbrace{\int \dots \int_{u_1 < \dots < u_n, u_1, \dots, u_n \in [0,t]} dS_{u_1} \otimes \dots \otimes dS_{u_n}}_{\Phi^n(S)} \tag{5}$$

where L_n is a vector of linear coefficients linking the drawdown at t with the signature terms of order n of the path S. In other words, considering (4) we do not know whether the impact of an indivual increment is -1 or 0, we consider the nested impact of intervals on the drawdown. This works as a strictly non-commutative version of the exponential function Taylor expansion would suggest (see section 4 for details). The ordered iterated integrals represent the drift, Levy area, and higher order moments of the path distribution (see Chevyrev and Oberhauser [42]). What rough path theory basically tells us is that drawdown, when bounded, can be written as a linear function of these moments. Moreover, leveraging factorial decay of the approximation error for Lip functions, for $N \to \infty$ (i.e. the full signature) we get an arbitrarily close approximation of ξ, where the rate decay depends on the roughness of the underlying process (Levin et al. [43], Boedihardjo et al. [44]).

A shortened interpretation of L is due to the Stone-Weierstrass theorem (De Branges [45]). Stone-Weierstrass universal approximation theorem is a crucial theorem in proving the universal approximating capabilities of neural networks (for instance see Cotter [46] and Hornik [47]). Neural networks concatenate linear weighing functions and non-linear sigmoids or ReLU activation functions, that serve as a learnable[5] basis to linearly approximate (the final fully-connected layer) the relationship between any output Y and input X, e.g. where f(X) = Y is some complex polynomial. The same intuition can be applied to signatures. As Eq. (.10) from appendix section 4 nicely shows, signatures are a natural basis to express a path function Y_t in. This motivated Levin, Lyons and Ni [43] (also see [48] for a more recent statement of the theorem) to introduce the signature approximation theorem, which states than any continuous[6] h on the path space of X can be linearly approximated on the signature space of $\Phi(X)$:

$$|h(X) - \langle L, \Phi(X) \rangle| \leq \epsilon \tag{6}$$

The proof is extremely concise, and that is why we summarize it here. Since $\Phi(X)$ is a tensor algebra of X, the family of all linear combinations of $\Phi(X)$ is an algebra[7]. Constant functions are preserved by the constant zeroth term of $\Phi(X)$. The algebra separates the points as comes naturally from [50] Corollary 2.16. Stone-Weierstrass then yields that the linear family on $\Phi(X)$ is dense in the space of continuous functions on X.

Now we propose to replace Eq. (6) by:

$$|\Xi(S) - \langle L, \Phi(S) \rangle| \leq \mu \tag{7}$$

for our use case. In theory, this approximation can be done with arbitrary precision μ, but from the above we know that for truncated signatures at level M, there will be an error κ that decays factorially in M.

$$\Xi(S) = \langle L, \Phi^M(S) \rangle + \kappa \tag{8}$$

Note that we could also do an equivalent truncation of the number of linear coefficients $len(L)$ rather than the signature order. However, from Eq. (5) we know that it is more natural to choose a set of linear coefficients that corresponds to a number of signature terms following the choice of M.

Remaining is the choice of L. As the relationship is essentially linear, Eq. (8) can be estimated using simple linear regression (OLS) and higher order polynomials are not required. However, since in practice the sample is limited and the number of signature terms scales exponentially with M and T, one has to be mindful of the overfitting problem, i.e. $len(\Phi^M) > T$. Therefore, we study the impact of regularized linear regression of $\Xi(X)$ on $\Phi^M(X)$ with a penalty for the number

[5]I.e. as a function of the network weights θ.

[6]Sufficiently smooth, i.e. Lipschitz function.

[7]This can be seen from the shuffle product property, as explained in [49] and the primer on emiellemahieu.github.io.

(absolute shrinkage or selection) and size (proportional shrinkage) of estimated coefficients (i.e. the elastic net (ElNet) regression). Hence, we conclude by specifying:

$$\hat{L} = \min_L(||\Xi(S) - \langle L, \Phi^M(S)\rangle||_2 + \lambda_1||L||_1 + \lambda_2||L||_2) \tag{9}$$

$$\hat{\Xi}(S) = \langle \hat{L}, \Phi^M(S)\rangle \tag{10}$$

where $\lambda_1 = \lambda_2 = 0$ corresponds to OLS, $\lambda_1 = 0$ corresponds to Ridge and $\lambda_2 = 0$ to LASSO regression [51]. In applications below, lambdas were set using 10-fold cross-validation.

2.3. Numerical experiments

2.3.1. Bottom-up simulations

We expect to find uniform decay of drawdown approximation error κ, where the decay constant is a function of the roughness of the underlying process (Boedihardjo et al. [44], [50]). Since the roughness of empirical data is unknown and has to be estimated, it is useful to test the approximation in a experimental set-up. We first test the approximation (9) on simulated Brownian motion (BM) and fractional Brownian motion (fBM) paths. Consider first the simplest case of homoskedastic BM ($dS_t = \mu dt + \sigma d\epsilon$). In this simple case, the price path S can thus be seen as the cumulative sum process of a random uncorrelated Gaussian, scaled with σ and added a deterministic drift μ. We consider piecewise linear paths of length T=20 days with values $\mu = 1\%/252$ and $\sigma = 40\%/252$. The rate at which the drawdown approximation error decays is a function of the smoothness of the process (Boedihardjo et al. [44]). fBM implements BM where the uncorrelated Gaussian increments are replaced by fractional Gaussian increments that have a long-memory structure. The martingale property that the autocovariance between Gaussian increments has expectation zero, is replaced by a generalized autocovariance function for two increments at t and s (i.e. lag $t - s$):

$$E[B_H(t)B_H(s)] = \frac{1}{2}[|t|^{2H} + |s|^{2H} - |t-s|^{2H}] \tag{11}$$

where H is the so-called Hurst exponent (H). Note that a $H = 0.5$ corresponds to Brownian increments, while $H > 0.5$ yields smooth, persistent, positively autocorrelated paths and $H < 0.5$ yields rough, antipersistent negatively autocorrelated paths. Intuition also tells us that the smaller H the more granular details the path has and the worse the approximation can become.

There are hence three 'dimensions' to this simulation study. We want to evaluate the error κ as a function of (1) roughness H, (2) signature approximation order M and (3) simulation size N. Figure 1 shows a heatmap that captures these three dimensions which go down in order of N, then H, then M. We thus expect the approximation to be better as we progress to the bottom of the map. At first it might look like an atypical picture but it collects the most important findings in one image:

- As we go down the approximation does become better, both in $RMSE$ and R^2 terms.
- However, the discrepancy between in-sample and out-of-sample accuracy looks high(er) for the small sample, high M approximations, which is indicative of overfitting. We will need to delve deeper into this relationship.
- Globally, the first bullet looks true, which means the approximation becomes better with sample size N.

- Locally, the red lines separate the different N so in between them only H and M vary. It is clear that both R^2 and $RMSE$ improve much with increasing H.
- Even more fine-grained, the grey lines separate the H, so in between only M varies. It is obvious for LinReg and larger samples that higher M implies higher accuracy.
- For small sample sizes the in- and out-of-sample performances differ a lot for LinReg, which indicates overfitting while this is not the case for large N or Elnet regression.
- Although the OOS variance is more consistent, the overall performance of biased regularized regression is much worse.
- Within the grey lines Elnet regression seems to have little difference, higher signature orders seem to contribute little or none with regularization.
- Finally, in- and out-of-sample consistency is found for larger ($N > 1000$) simulations.

Figures 2, 3 and 4 and 5 take a slice of these heatmaps and investigates individual dimensions. Given our first comments, we are initially interested in the stability of linear regression and evaluate the in- and out-of-sample discrepancy as a function of sample size. Figure 2 shows the difference between in- and out-of-sample accuracy as a function of sample size $log(N)$. To be more precise, we averaged the difference for all simulation sizes over either the values of M to get a curve per value of H (blue line) or the opposite (red line). The conclusions are twofold: (1) for a sufficiently large sample ($log(N) > 3$) the overfitting problem disappears, which argues in favour of the proposition that the relationship is not spurious, and (2) the range in initial IS-OOS accuracy divergence with roughness H or M as additional factor looks the same, i.e. the overfitting can not be brought back to confounder H or M. Next we look in Figure 3 the train and test R^2 as a function of the signature approximation order M. Now we take the average performance over N and H and find uniform increase of performance, except for a kink around $M = 5$. When deconfounding for sample size N, which you can find on the right hand side of Figure 3, we find that only small samples do not show uniform increase, which suggest that signature order $M > 5$ will overfit for $N < 500$. This also makes sense given previous comments. When looking at the $RMSE$ in Figure 4, we have exactly the same findings. Train $RMSE$ shows nice uniform decay, except for $M > 5$ where it starts to overfit. When deconfounding for N in the same figure on the right hand side, we see this problem only occurs for small sample sizes ($N < 500$). Finally, we check the third dimension of the simulation, the relationship between the accuracy of the approximation and the roughness H of the assumed process. We find approximate uniform increase of R^2 and uniform decay in $RMSE$.

In summary, the simulation study confirms with our initial expectation but raises some warnings as well:

- For large sample, say $N > 1000$, one generally does not get into any issues and finds uniform decay in κ with both H and M, both test and training, i.e. without overfitting.
- Linear regression is then the best unbiased estimator and the advised way to go.
- For small samples, there are no guarantees that in-sample fit is indicative for out-of-sample fit, especially for large M, because that implies more regression terms and therefore more room for overfitting. For small samples, say $N < 500$, we expect the biased regularized ElNet to give more robust results, and in general signatures of relative low order $M < 5$ are advised as not to use too many terms in the regression.

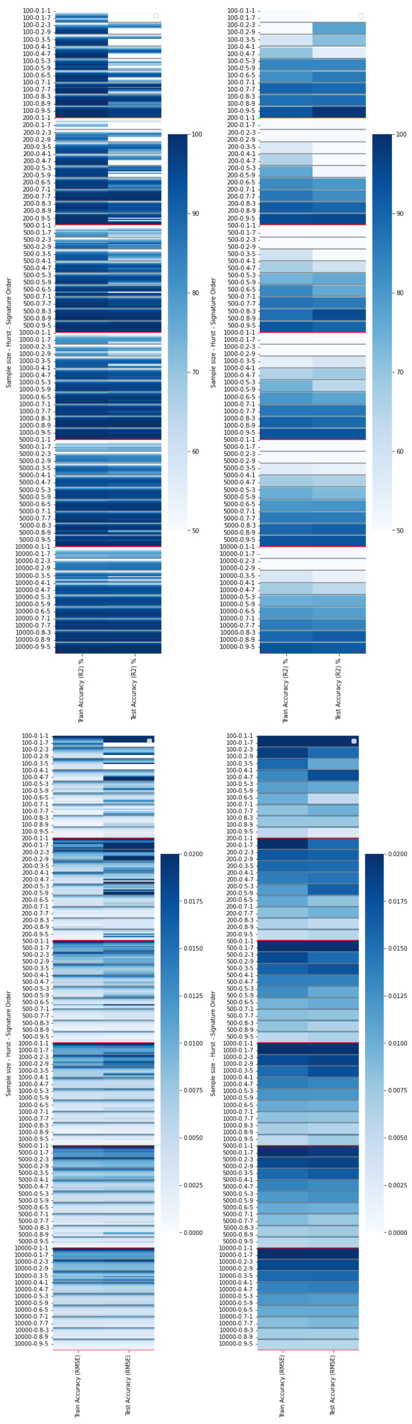


Figure 1: R2 (top) and RMSE (bottom) overview heatmaps for LinReg (left) and Elnet (right)

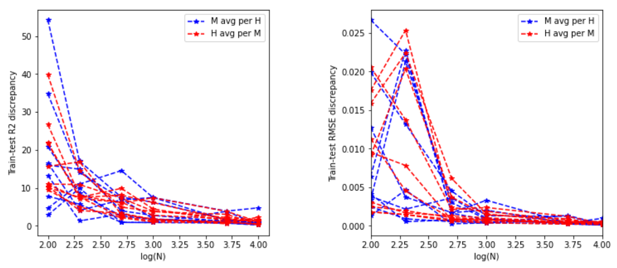


Figure 2: In- and out-of-sample accuracy difference as a function of sample size $log(N)$ (Red = averaging out over H to get a line per value of M, and blue = opposite).

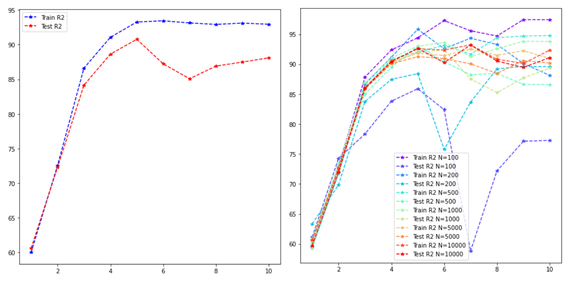


Figure 3: Train and test R^2 (y-axis) as a function of signature approximation order M. Average over N and H is shown on the left. An average per N is shown on the right.

- Finally, and importantly, (1) the accuracy that is good enough for the application at hand of course depends on the application, (2) we will deal with data sets that are considered large for these standards.

2.3.2. *Real-world portfolios*

Consider U=4 investible instruments: Equity (S&P500), fixed income (US Treasuries), commodities (GSCI index) and real estate (FTSE NAREIT index). We collect price data (close prices) between Jan 2000 and May 2022, which gives us T=5840 daily observations. Clearly, these 4 different asset classes have different return, volatility and drawdown characteristics. This can clearly be seen from Figure 6.

As an investor, we are interested in the drawdown of our portfolio $\mathbf{w}$, $w_i, i \in 1, ..., U$, which allocates a weight to each investible asset. In this experiment, we attach random weights to each asset, which gives rise to sample paths like shown below in Figure 7, where we pick a $\tau < T$ such that we have $T - \tau$ overlapping sample paths, or $N_p(T - \tau)$ portfolio paths. Here, $\tau = 20$, or approximately monthly sample paths.

The drawdown distribution of these paths is shown in Figure 8. This is the probability measure of crucial interest in our next application, or in the above-mentioned real-world financial problems. The focal question is whether we can adapt input sample paths, or specifically their parameterized version, such that this distribution converges towards the portfolio drawdown distribution that we expect. We also compare the distribution with the distribution that is implied

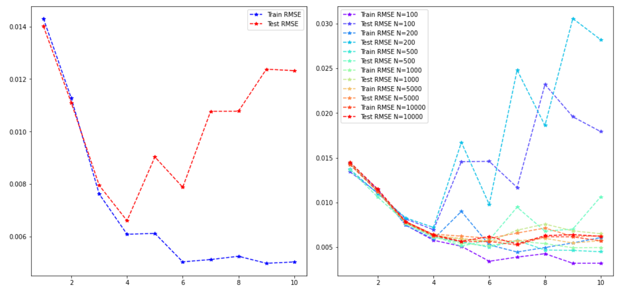


Figure 4: Train and test $RMSE$ (y-axis) as a function of signature approximation order M. Average over N and H is shown on the left. An average per N is shown on the right.

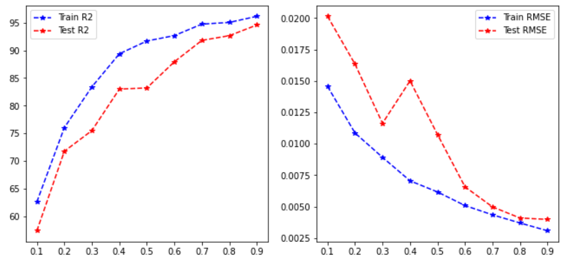


Figure 5: Train and test R^2 (left) and $RMSE$ (right) as a function of the roughness H

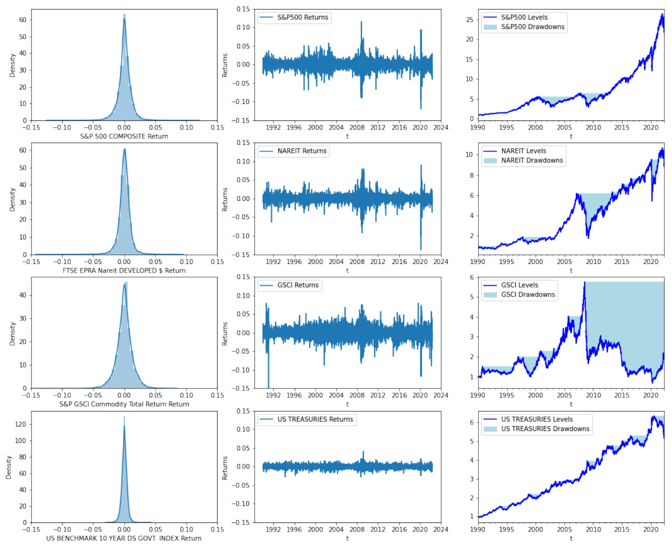


Figure 6: Data overview

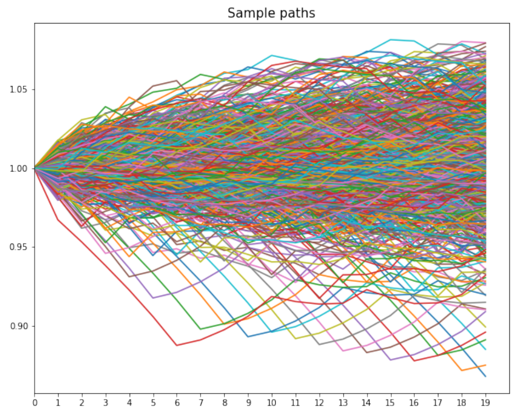


Figure 7: Sample paths

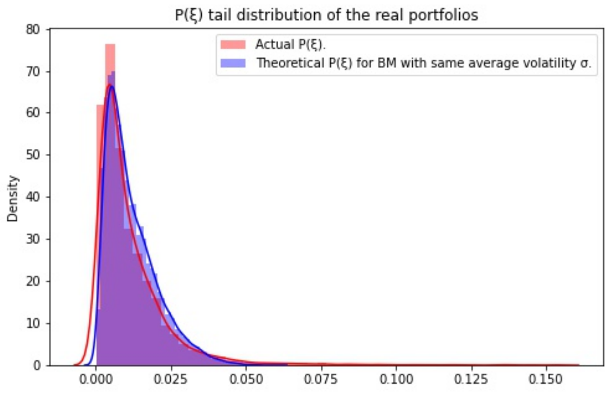


Figure 8: Drawdown distribution

by simulated Brownian motion with the same average volatility σ parameter as the portfolios we have constructed. Note that one could also use the closed forms from Douady et al. [12], Rej et al. [13] and alike here. Importantly, we clearly see that the blue density is merely an optimistic lower bound on the true distribution of portfolio drawdown, and misses out on the tails. This can also be seen in Figure 9. The link between generalized stochastic processes and this distribution is far from trivial, as one can not rely on Lévy's theorem anymore. In the next section, our methodology thus advocates a fundamentally non-parametric approach to simulate paths that closely resemble the correct drawdown distribution.

Concerning the approximation on real-world data our conclusions are analogous to the simulation experiments. The accuracies are shown in Tables 1 and 2. Given the large N Train en Test R^2 and $RMSE$ are consistent, answering our biggest concern. From $M > 5$ we get accurate approximations of ξ both in R^2 and $RMSE$ terms, after which it further improves as expected at a much slower rate. We also included the computation time for calculating one signature of the order. Clearly for large samples the computational bottleneck, i.e. the hardware, will influence the exact choice of M. As expected, Elnet performs significantly worse again. From $M > 4$ the regressions are not improved, a bit better than what we observed in the simulations, but still bad. Train and Test accuracy is as consistent as LinReg, so one would be inclined to stick to ordinary least squares. It is worthwile to note that, ceteris paribus, assuming N and M are large, the maximum $80\%R^2$ and minimum 6.5 bps $RMSE$ error corresponds to a roughness of our portfolios of about $H = 0.5$. Actually, slightly less from R^2 point of view and slightly more from $RMSE$ perspective. However, this roughly confirms our expectation that real-life portfolios are approximately Brownian, or approximately efficient on this timescale, although with much fatter tails. The approximation seems to grasp both, as e.g. Figures .14, .15 and .16 in Appendix 2 (section 4) suggest. It has an overall good fit without missing out on the higher drawdown scenarios.

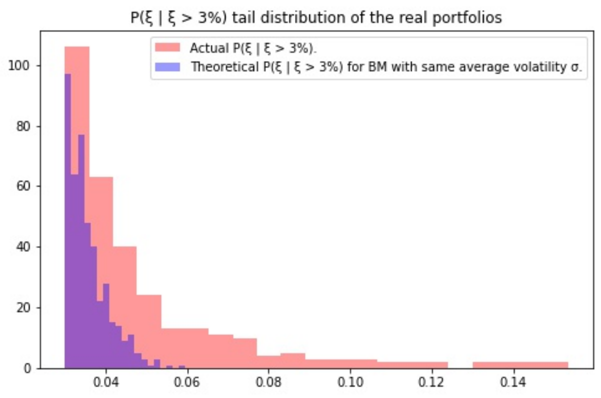


Figure 9: Drawdown distribution

Sig Order	Train R2	Time	Len	Test R2	Train RMSE	Test RMSE
1	0.0100	0.00211	2	0.0110	0.0142	0.0141
2	0.1973	0.00460	6	0.2036	0.0128	0.0126
3	0.4833	0.00268	14	0.4819	0.0102	0.0102
4	0.6670	0.00283	30	0.6728	0.0082	0.0081
5	0.6991	0.00208	62	0.7065	0.0078	0.0077
6	0.7335	0.00209	126	0.7408	0.0074	0.0072
7	0.7488	0.00271	254	0.7559	0.0071	0.0070
8	0.7677	0.00201	510	0.7752	0.0069	0.0067
9	0.7843	0.00284	1022	0.7899	0.0066	0.0065
10	0.8021	0.00301	2046	0.8070	0.0063	0.0062

Table 1: Linear Regression Fit

Sig Order	Train R2	Time	Len	Test R2	Train RMSE	Test RMSE
1	0.0008	0.0021	2	-0.0020	0.0140	0.0136
2	0.0616	0.0022	6	0.0600	0.0136	0.0132
3	0.4367	0.0016	14	0.4459	0.0105	0.0101
4	0.6348	0.0015	30	0.6449	0.0085	0.0081
5	0.6611	0.0015	62	0.6725	0.0081	0.0078
6	0.6612	0.0015	126	0.6732	0.0081	0.0077
7	0.6594	0.0024	254	0.6719	0.0082	0.0078
8	0.6580	0.0018	510	0.6707	0.0082	0.0078
9	0.6568	0.0026	1022	0.6697	0.0082	0.0078
10	0.6559	0.0021	2046	0.6688	0.0082	0.0078

Table 2: Elastic Net CV(10) Fit

3. The drawdown market generator

3.1. The problem statement revisited

The advantage of our problem setting is that it speaks to the imagination. Imagine a path with a certain level of drawdown. If we would invite you to gradually transform the path - i.e. tilt, stretch, squeeze, rescale it - such that it better reflects a level of drawdown we are trying to calibrate to, how would you do this?

Crafting a DGP and obtaining a certain level of drift, volatility or higher order moments is relatively easy, as those are typically the equations that constitute the DGP. The link with calibrating to realistic value-at-risk (VaR) or expected shortfall (ES) levels is less straightforward but can be done by appreciating the direct link between quantiles and moments, i.e. the higher order cumulants in a Cornish-Fisher-like approach.

With drawdown, this relationship is even more convoluted. One would expect drawdown to be a function of the moments of the stochastic process described by the DGP (e.g. Douady et al. [12] and Rej et al. [13] in the Markovian case) and the autocorrelation function in the non-Markovian case. In fact, one would expect drawdown to be a function of the moments of the path, rather than the stochastic variable. Indeed, one of the main ideas of this paper is to treat S as a single mathematical object as described by $\Phi(S)$, rather than the dynamics of S_t evolving over time. Increments are like pictures of the price path, while signatures are the movie that describes scenes rather than frames. A less imaginative interpretation is defining signatures as exactly the moment generating function (see again Chevyrev and Oberhauser [42]) of the path. The contribution of this paper in the previous sections basically boiled down to saying that drawdown is a function of these path moments rather than the usual moments. And it is a very simple function, just a linear combination.

Now the next insight, which we get completely for free from this linear combination, is that this approximation is a smooth function of the path. The signature is essentially just a non-parametric sum of the path's values. It is a complex sum because of its iterative nature, but it is just a sum. (Weighted) sums are differentiable. That is why Kidger and Lyons [52] focus on this differentiability property for efficient CPU and GPU implementations in their *signatory*[8] package, which we use in our Python code as well. Now drawdowns are (approximately) just weighted versions of these moments, so what we propose in our drawdown market generator in the next section is to simulate paths, evaluate their path moments, and not just take the divergence between their moments like a MMD (e.g. Buehler et al. [5]) but to weigh their moments according to pre-trained linear combinations that effectively measure the drawdown divergence between the input and the output samples. This gives an imaginative answer to the start of this subsection, and a technical answer to the two issues outlined in the introduction 1.

3.2. The algorithm

The very core of our algorithm is a variational autoencoder (VAE), which is a general generative machine learning architecture that links an encoder and decoder network to generate new samples from a noisy, in this case Gaussian, latent space. The whole idea of a Monte Carlo is to transform noise into samples that are indistinguishable from the real samples by scaling them, adding drift, and so and so forth. In other words, the neural network that constitutes the decoder is our non-parametric DGP. It does contain the parameters of the neural network, of course, but

[8]`https://pypi.org/project/signatory/`

it is non-parametric in the sense that we do not have to specify the dynamics in a handcrafted formula before we can do Monte Carlo. We rather rely on the universal approximation theorems behind (even shallow) neural networks (Hornik [47] and Cotter [46]) to approximate a realistic DGP by simply iterating data through the network and updating the parameters θ with feedback on the drawdown distribution of the batch, assuring that the approximated DGP converges in train and validation loss to the empirical DGP.

We will not discuss the VAE architecture in depth here, but refer the interested reader to Kingma and Welling [53] for details on the encoder and decoder networks g, backpropagation, the latent Kullback-Leibler $\mathcal{L}_L$ loss and the standard $L2$ reconstruction loss $\mathcal{L}_R$. Although we should stress that the main reason for picking a rather standard VAE (over restricted Boltzmann machines, generative adversarial networks, generative moment matching networks or normalizing flow-based VAEs) is their simplicity, speed, flexibility, scalability and stability during training. Boltzmann machines are very efficient to train, but the energy-based loss function and their binary values makes them very inflexible for adjusting loss functions. Through the discriminator mechanism GANs are most flexible and a very popular choice in related literature, but notoriously expensive to train in terms of required data and speed, and associated instabilities such as *mode collapse* and *vanishing gradients* leading to subpar results.

The proposed algorithm[9] is provided in Algorithm 1 and 2. In short, we propose to include the divergence between the observed drawdown distribution of a batch and the synthetic drawdown distribution in the reconstruction loss function. The market generator can loosely be interpreted as a moment matching network, focusing on the moments of drawdown rather than returns, where the distance between them over a batch is like the distance between real and fitted distributions in Figure 8, which was our initial concern. Regarding moments it is also correct to say that the model does not look at the usual moments of the path's stochastic process but at the path moments, and weighs them according to their importance for capturing the path's drawdown.

Algorithm 1 VAE market generator given important path feature Ξ

```
    Input Historical price paths $S : [0,T] \to \mathbb{R}$, hyperparameters, signature truncation level M and feature weight $\alpha$.
    Output Trained VAE Market Generator $g_\theta$

 1: procedure TRAIN
 2:     Divide historical sample into batches $\mathcal{B}$ of length $\tau$, calculate the signatures of these paths truncated at level M, $\Phi_M^{\mathcal{B}}$,
        calculate the drawdowns $\Xi$ of these paths $\Xi(S^{\mathcal{B}})_\tau = \int_0^\tau (max_{t_i<t}(S^{\mathcal{B}}_{t_i}) - S^{\mathcal{B}})dt$ denoted $\hat{\Xi}(S_b)$
 3:     $\hat{L} \leftarrow LinearRegression(\hat{\Xi}(S^{\mathcal{B}}), \Phi_M^{\mathcal{B}})$
 4:     Initialize the parameters $\theta$ of the VAE.
 5:     for $i : \{1, ..., N\}$ do:
 6:         Sample a batch $\mathcal{B}$ and pass it through the encoder $g_\theta$ and decoder network $g_\theta^{-1}$ .
 7:         Calculate drawdown $\Xi(S')$ of the output sample $S'$ using the differentiable signature approximation: $\langle \hat{L}, \Phi_M(S') \rangle$
 8:         Define the reconstruction loss term as the weighted average of RMSE error and drawdown loss: $\mathcal{L}_{\mathcal{R}} = \mathbb{E}_{\mathcal{B}}||S - .$
            $S'||^2 + \alpha \mathbb{E}_{\mathcal{B}}||\langle \hat{L}, \Phi_M(S) \rangle - \langle \hat{L}, \Phi_M(S') \rangle||^2$
 9:         $\mathcal{L} = \mathcal{L}_L + \mathcal{L}_R$
10:         $\theta \leftarrow \theta - l\frac{d\mathcal{L}(\theta)}{d\theta}$
11:
```

[9]The implementation can be found on `https://github.com/emiellemahieu` for reproduction purposes, where one can also find the code of the simulation studies in 2.3 for full reproducibility. Also see `https://emiellemahieu.github.io` for updates.

Algorithm 2 Sampling from the market generator

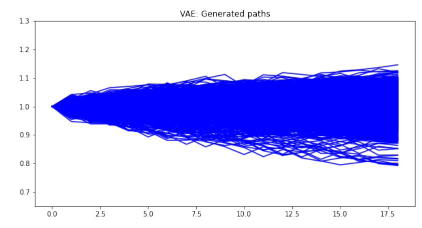
```
    Input Trained VAE Market Generator $g_\theta$.
    Output $N_s$ generated samples S'

1: procedure GENERATE
2:     for $j : \{1, ..., N_s\}$ do
3:         Sample a random Gaussian variable Z
4:         $S' \leftarrow g_\theta^{-1}(Z)$
5:
```

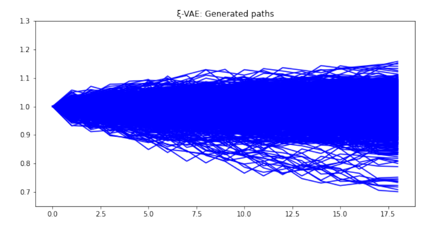
3.3. *Results and discussion*

The main findings are summarized in Figures 10, 11, 12 and 13. Figure 10 shows the generated paths of the standard VAE model versus the drawdown market generator ξ-VAE. It is clear that the original VAE does not do a bad job at generating realistic scenarios, but it is kind of reminiscent of Figure 8 in the sense that the paths are non-Brownian, or returns non-normal, but still too centered around the Brownian-like distribution (the densely colored areas).

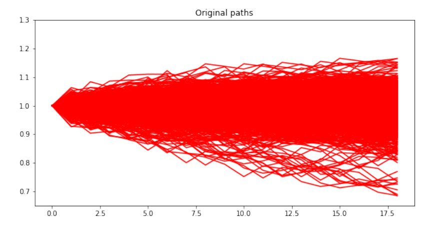
This is also clear from Figure 12. To summarize our architecture rather simplisticly, the standard VAE focuses on reproducing the top distribution and tries to do a good average job over a batch ($L2$ loss) such that it misses out on the tails. The ξ-VAE fits both the top and the bottom distribution such that it includes more tail scenarios. It is again obvious from an autocorrelation point of view that a standard VAE thus shows a lack of extreme adverse scenarios (remember this important point in the introduction 1). This also becomes apparent in Figure 11, which plots the fake and real drawdowns as a scatter. The standard VAE does a good job in capturing most part of the moderate drawdowns, but is scattered towards the higher drawdowns and produces none in the tail, while this is resolved with the ξ-VAE. A closely related graph is Figure 13, which plots the same scatter but using ordered observations (i.e. quantiles) and a first bisector line which would fit the data if the synthetic drawdown distribution has exactly the same quantiles as the original distribution. Again we find that the VAE misses out on capturing tail drawdown scenarios, while ξ-VAE does a much better job. Importantly, this could be done by just memorizing the training samples and reproducing them exactly. However, the whole point of the *bottleneck* architecture of an autoencoder is to summarize the training data in fewer parameters (i.e. the non-parametric DGP) as a dimension reduction, rather than trivially mirroring the training data. That is why one sees different simulated paths than identical ones to the input samples, and why by using the trained decoder as a non-parametric DGP one can create an *infinite* amount of genuinely new data with the same distribution as the training data. Moreover, both train and validation losses flat out at convergence (see appendix 4 with details on the training convergence) while purely mirroring data would imply training losses would be minimal at the cost of high validation losses.



(a) Generated VAE

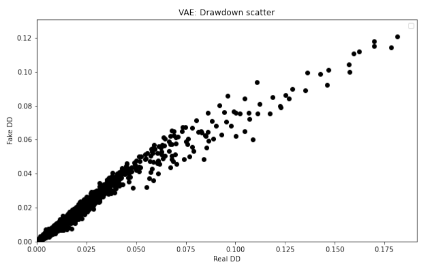


(b) Generated ξ-VAE

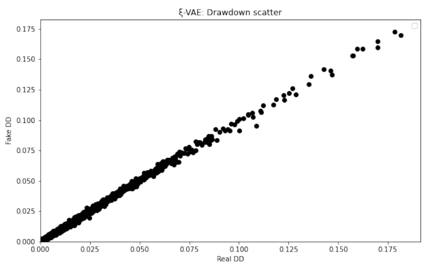


(c) Original

Figure 10: Generated paths vs original paths

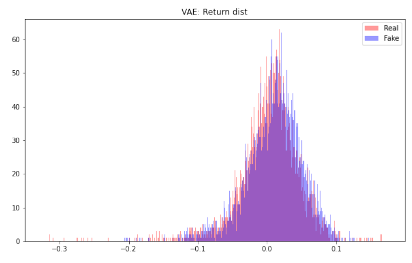


(a) VAE

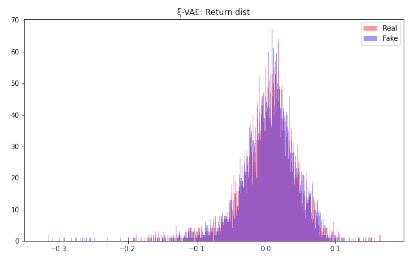


(b) ξ-VAE

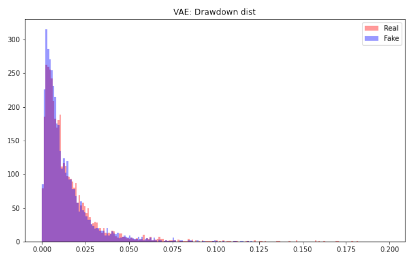
Figure 11: Synthetic versus real drawdowns scatter



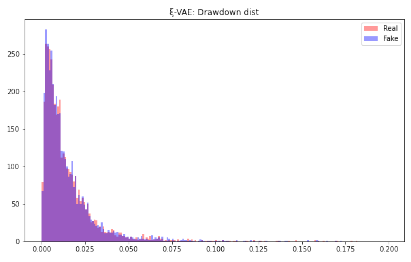
(a) Return distribution VAE



(b) Return distribution ξ-VAE

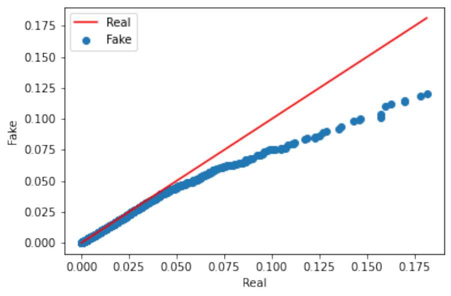


(c) Drawdown distribution VAE

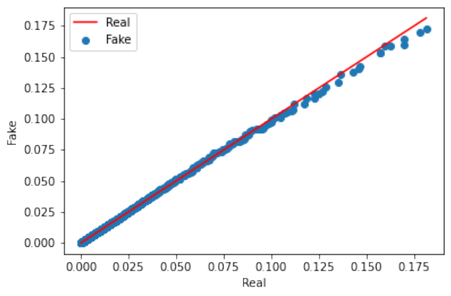


(d) Drawdown distribution ξ-VAE

Figure 12: Generated versus real drawdowns and returns distribution



(a) VAE fake versus real ξ quantiles



(b) ξ-VAE fake versus real ξ quantiles

Figure 13: Drawdown QQ plots

4. Conclusion

Learning functions on paths is a highly non-trivial task. Fortunately, rough path theory offers tools to approximate functions on paths that are sufficiently smooth by considering a controlled differential system and iterating the effect of intervals of the (rough) path on the (smooth) outcome function. One key path dependent risk functional in finance is a portfolio value path's drawdown. This paper takes the perspective of portfolio drawdown as a non-linear dynamic system (a controlled differential equation $d\xi$), rather than a perspective on the solution, the exact expression of Ξ, directly. It relied on some important insights from the theory of rough paths and path signatures, summarized in the first appendix. Further, it pinpointed that by taking this perspective, rather than continuous differentiability, a more general smoothness condition of bounded differentials is sufficient to interpolate between two types of effects of a path on its resulting drawdown and thus locally approximating this drawdown as a function of a path, without having to evaluate Ξ. The paper built on this very idea to propose a linear signature-based drawdown approximator, which furthermore yields a differentiable approximation of portfolio drawdown. On simulated Brownian, fractional Brownian and on real-world data, regression results exhibit a good fit for ordinary least squares linear regression, and do not favor regularized versions (LASSO, ElNet) when one has a reasonable sample size. Finally, a proposed application of the approximation is a so-called *market generator* model that evaluates the synthetic time series samples in terms of their drawdown. We argue that by doing so one gets more realistic scenarios than the standard VAE model and one is able to reproduce the drawdown distribution without trivially memorizing it.

Future work will focus on extending this application and further applying it to portfolio drawdown optimization, where one can e.g. test a data hungry drawdown control strategy over a host of synthetic scenarios rather than the single historical one. One of the use cases that has not been mentioned yet is non-parametric Monte Carlo as a mathematically principled data augmentation technique. Denoising autoencoders have served as a tool to denoise historical data by, ironically, adding noise to the data and let the model learn the difference. A regression, classification or optimization model can be trained as an *ensemble* of the noisy predictions, such that by construction on average one is close to the true (denoised) regression prediction, classification or optimal decision. For financial time series a similar motivation could be made. Defining an optimal portfolio as an ensemble average (e.g. bootstrapped) of multiple historical scenarios (implicitly) assumes noise is cancelled out, or robustness is achieved, by taking an additional mean. With the dimension reduction tool proposed in this paper and the added *noise-by-construction* one makes this idea explicit, rather than removing *noise-by-assumption*. This non-parametric Monte Carlo idea could hence robustify his or her methodology further as a mathematically principled data augmentation technique. Moreover, the fundamentally non-parametric nature of our Monte Carlo engine opens possibilities to full non-parametric pricing of max loss or drawdown insurance options.

Appendix: Controlled differential equations and path signatures

Controlled differential equations

We are generally interested in a CDE of the form:

$$dY_t = g(Y_t)dX_t \tag{.1}$$

where X is a path on $[0, T] \to \mathbb{R}$, called the driving signal of the dynamic system. g is a $\mathbb{R} \to \mathbb{R}$ mapping called the physics that models the effect of dX_t on the response dY_t. A controlled differential equation (CDE) distinguishes itself from a ordinary differential equation in the sense that the system is controlled or driven by a path (dX) rather than time (dt) or a random variable (stochastic SDEs, $d\epsilon$).

Signatures

A series of coefficients of the path that naturally arrives from this type of equation is the series of iterated integrals of the path, or the path signature Φ. The signature of a path $X : [0, T] \to \mathbb{R}$ can be defined as the sequence of ordered coefficients:

$$\Phi(X) = (1, \Phi_1, ..., \Phi_n, ...) \tag{.2}$$

where for every integer n (order of the signature):

$$\Phi_n(X) = \int\limits_{u_1<...<u_n, u_1,...,u_n \in [0,T]} ... \int dX_{u_1} \otimes ... \otimes dX_{u_n} \tag{.3}$$

where we define the n-fold iterated integral as all the integrals over the n ordered intervals u_i in [0,T]. The signature is the infinite collection for $n \to \infty$, although typically lower level M truncations are used.

$$\Phi^M(X) = (1, \Phi_1, ..., \Phi_M) \tag{.4}$$

Picard Iterations

The idea behind a Picard iteration is to define for:

$$dY_t = g(Y_t)dX_t \tag{.5}$$

a sequence of mapping functions $Y(n) : [0, T] \to \mathbb{R}$ recursively such that for every $t \in [0, T]$:

$$Y(0)_t \equiv y_0 \tag{.6}$$

$$Y(1)_t = y_0 + \int_0^t g(y_0)dX_s \tag{.7}$$

$$Y(n+1)_t = y_0 + \int_0^t g(Y(n)_s)dX_s \tag{.8}$$

Now by simple recursion one finds that (for a linear g):

$$Y(n)_t = y_0 + \sum_k^n g^{\otimes k}(y_0) \int\limits_{u_1<...<u_n, u_1,...,u_n \in [0,T]} ... \int dX_{u_1} \otimes ... \otimes dX_{u_n} \tag{.9}$$

Such that a solution for Y_t would be:

$$Y_t = y_0 + \sum_k^{\infty} g^{\otimes k}(y_0) \int\limits_{u_1<...<u_k, u_1,...,u_k \in [0,T]} \dots \int dX_{u_1} \otimes ... \otimes dX_{u_k} \tag{.10}$$

This result shows how the signature, as an iterative representation of a path over ordered intervals, naturally arises from solving CDEs using Picard iterations, and how it is a natural generalization of Taylor series on the path space when the physics is linear.

$$g^{\circ 1} = g \tag{.11}$$

$$g^{\circ n+1} = D(g^{\circ n})g \tag{.12}$$

then it is natural to define the N-step Taylor expansion for Y_t by $\hat{Y}(N)_t$ as:

$$\hat{Y}(N)_t = y_0 + \sum_{n=1}^{N} g^{\circ n}(y_0) \int\limits_{u_1<...<u_n, u_1,...,u_n \in [0,T]} \dots \int dX_{u_1} \otimes ... \otimes dX_{u_n} \tag{.13}$$

Clearly, $\hat{Y}(N)_t$ is linear in the truncated signature of X of order N[10].

Example. The simplest example of:

$$dY_t = g(Y_t)dX_t \tag{.14}$$

Is a linear physics for a linear path X:

$$dY_t = Y_t dX_t \tag{.15}$$

where:

$$g = g^{\circ} \tag{.16}$$

$$g^{\circ n+1} = D(g^{\circ n})g \tag{.17}$$

$$X_t = X_0 + \frac{X_T - X_0}{T} t \tag{.18}$$

and assuming:

$$y_0 = 1 \tag{.19}$$

$$X_0 = 0 \tag{.20}$$

Indeed, this yields the exponential function $Y_t = \exp(X_t)$. For non-linear driving signals (where the order of the events matter), one generally gets a non-commutative version of the exponential function in Eq. (.10)! For linear time, the order of events does not matter and we generally get the increment of the path raised to the level of the iterated integral, divided by the level factorial (i.e. the area of an n-dimensional simplex).

[10]Moreover, the error bounds of $\hat{Y}(N)_t$ to approximate Y_t yield a factorial decay in terms of N, i.e. $|Y_t - \hat{Y}(N)_t| \leq C\frac{|X|^{N+1}_{1,[0,t]}}{N!}$. This result can be extended to p-geometric rough paths where g is a $Lip(K)$ where $K > p - 1$ [44].

This can be seen from:

$$Y_t = y_0 + \sum_{n=1}^{N} Y^{\circ n} \underset{u_1<...<u_n, u_1,...,u_n \in [0,T]}{\int ... \int} dX_{u_1} \otimes ... \otimes dX_{u_n} \tag{.21}$$

now it is easy to see that:

$$\begin{aligned}
\Phi^n &= \underset{u_1<...<u_n, u_1,...,u_n \in [0,t]}{\int ... \int} dX_{u_1} \otimes ... \otimes dX_{u_n} \\
&= \underset{0<u_1<...<u_n<t}{\int ... \int} d(\frac{X_t}{t}u_1)...d(\frac{X_t}{t}u_n) \\
&= \prod_{j=1}^{n}(\frac{X_t}{t}) \underset{0<u_1<...<u_n<t}{\int ... \int} du_1...du_n \\
&= \frac{1}{t^N}\prod_{j=1}^{n}(X_t)\frac{t^N}{n!} = \frac{(X_t)^n}{n!}
\end{aligned} \tag{.22}$$

such that:

$$Y_t = y_0 + \sum_{n=1}^{N} y_0 \frac{1}{n!}(X_t)^n = 1 + (X_t) + \frac{(X_t)^2}{2!} + \frac{(X_t)^3}{3!} + ... \tag{.23}$$

Which is the classical Taylor expansion for the exponential function, i.e. linear physics integrated over a linear path (time). Now the signature approximation is the generalization of this idea to the path space (i.e. Y is a function on a path), where the path X_t need not be a linear map of time, and the physics g need not be linear (e.g. drawdown Eq. (4)).

Appendix 2: Regression fits details

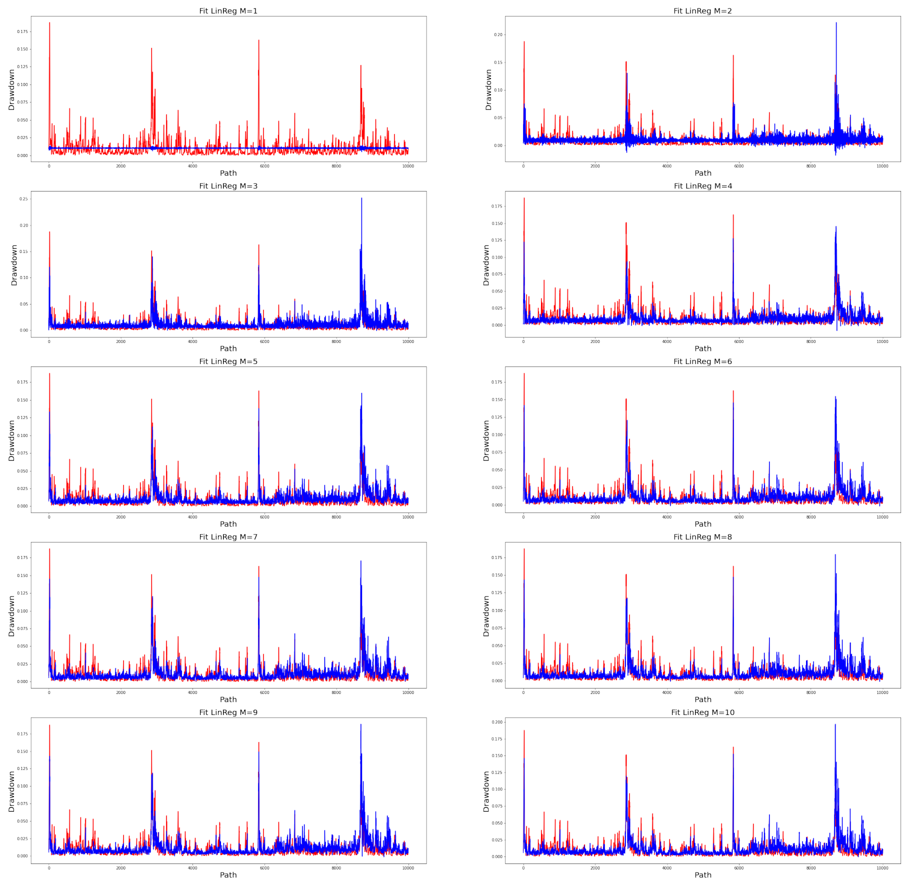


Figure .14: Linear regression fit

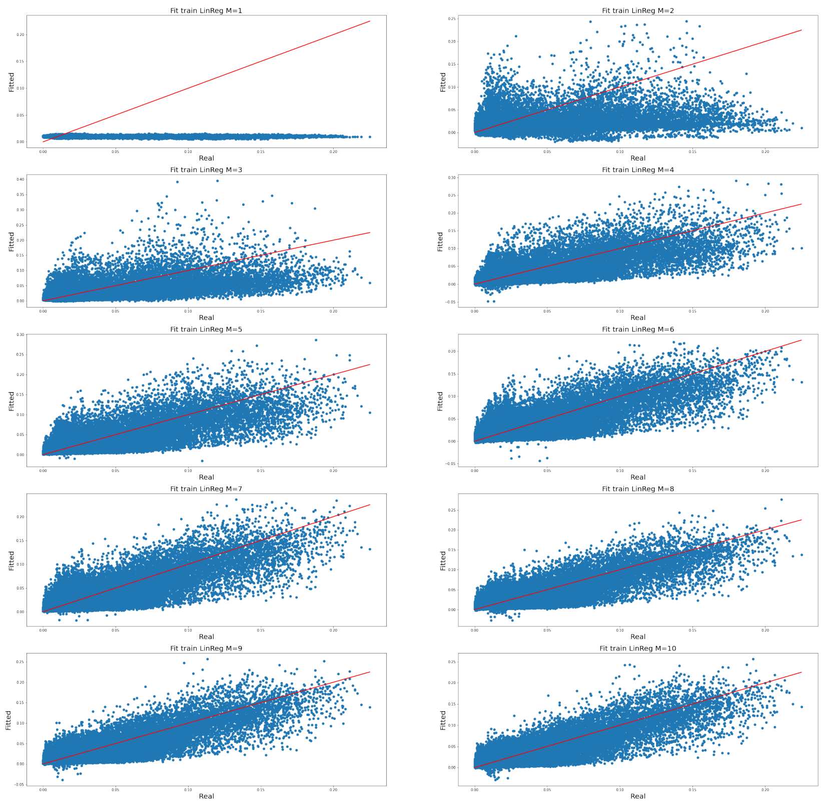


Figure .15: Linear regression train fit scatter

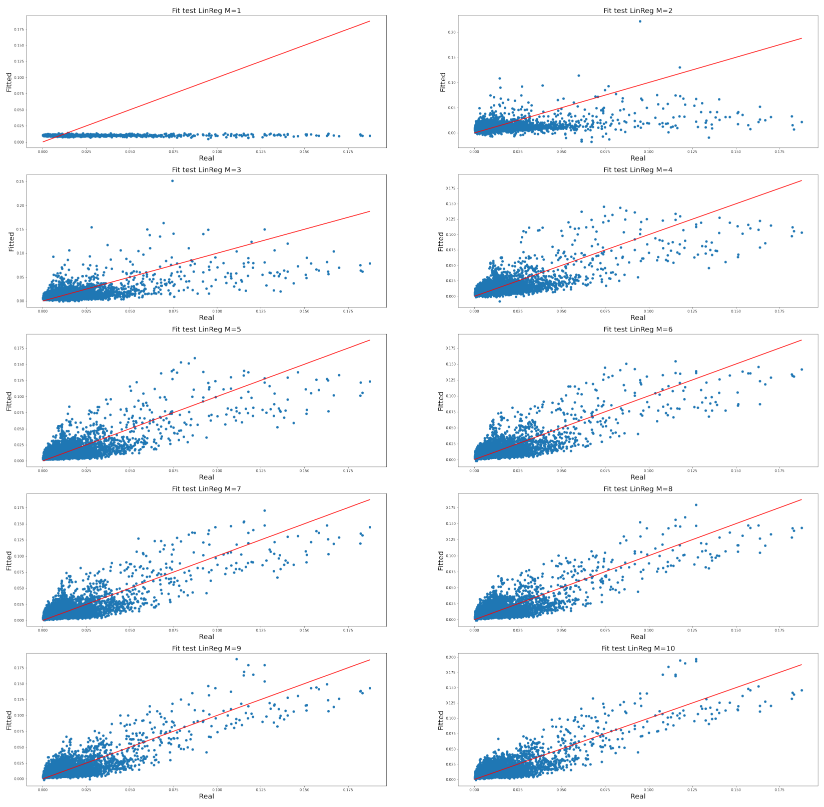


Figure .16: Linear regression test fit scatter

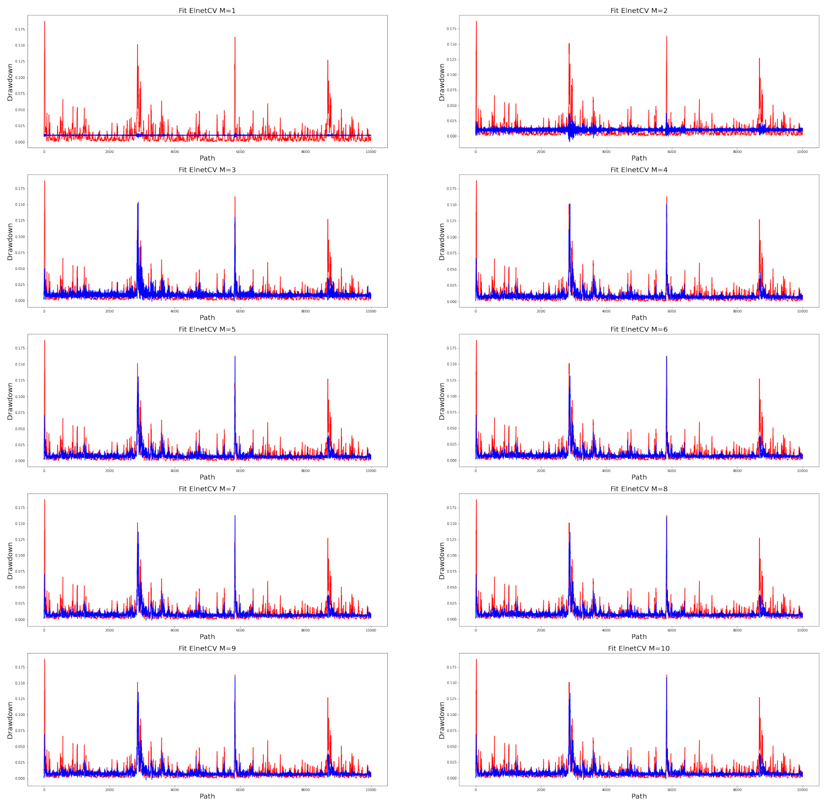


Figure .17: ElNet CV(10) fit

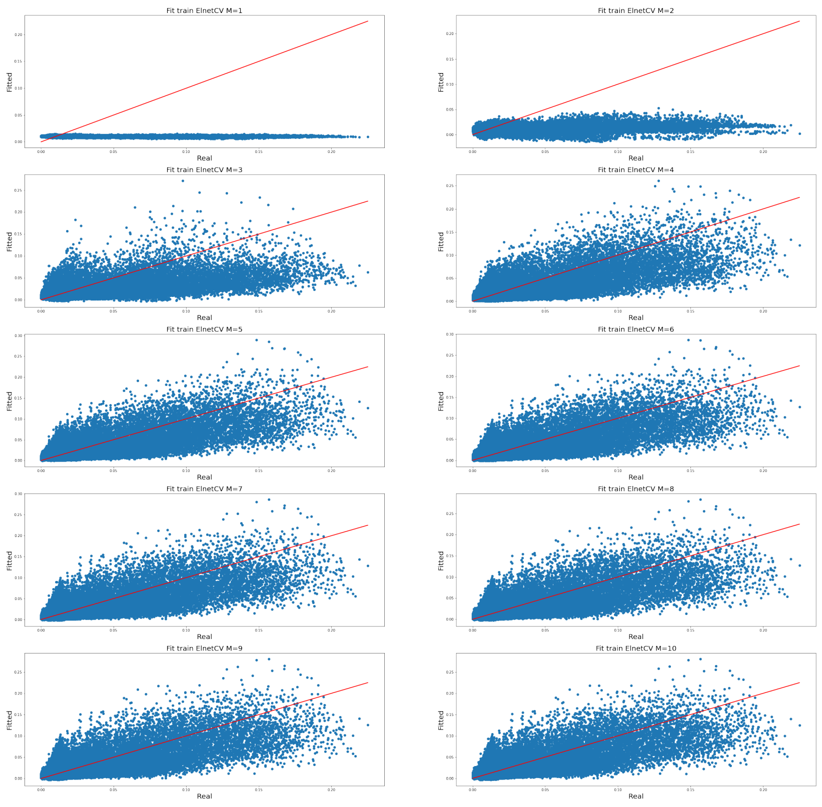


Figure .18: ElNet CV(10) train fit scatter

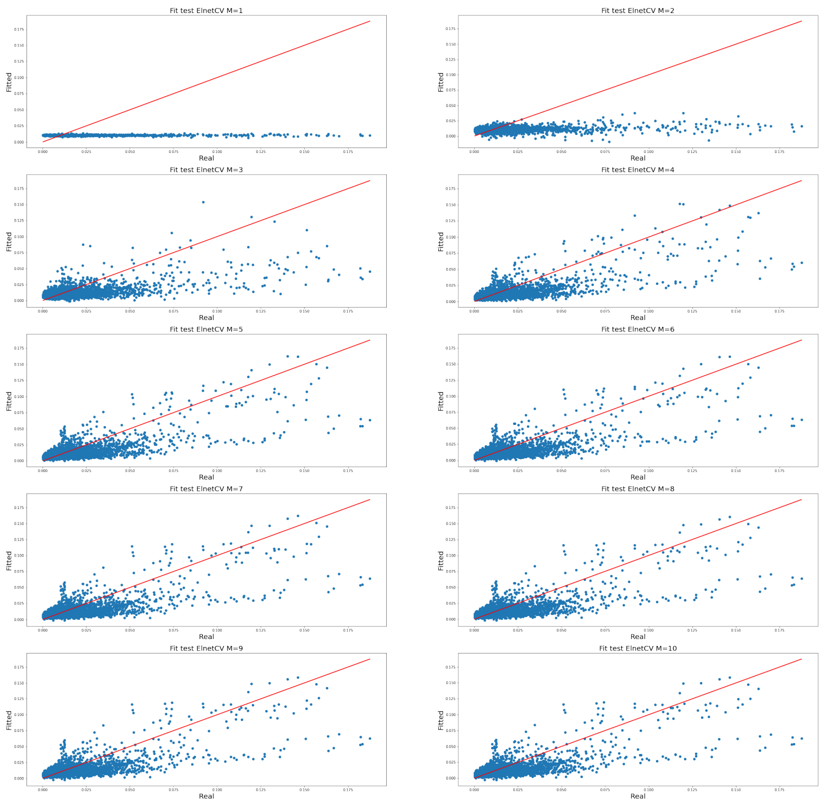


Figure .19: ElNet CV(10) test fit scatter

Appendix 3: drawdown market generator details

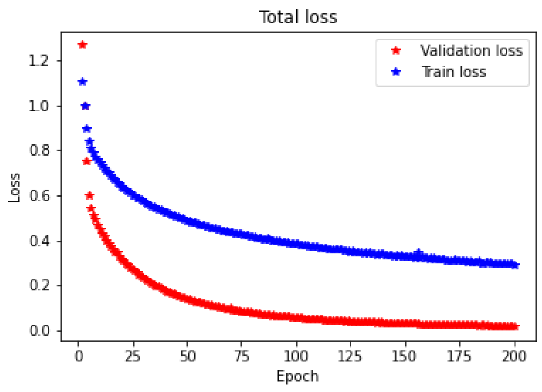


Figure .20: Drawdown market generator: total loss convergence (rescaled)

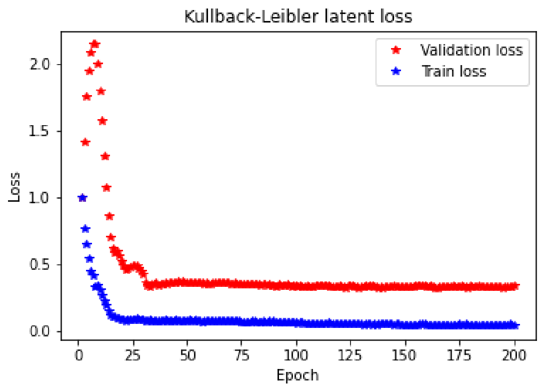


Figure .21: Drawdown market generator: latent (KL) loss convergence (rescaled)

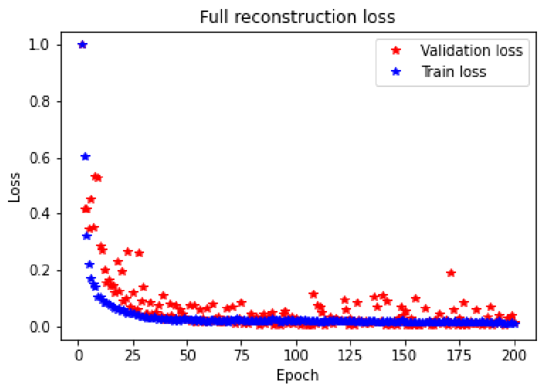


Figure .22: Drawdown market generator: total reconstruction loss convergence (rescaled)

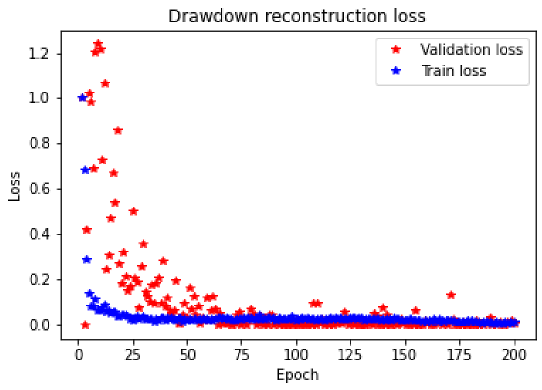


Figure .23: Drawdown market generator: drawdown reconstruction loss convergence (rescaled)

References

[1] M. Wiese, R. Knobloch, R. Korn, P. Kretschmer, Quant gans: Deep generation of financial time series, Quantitative Finance 20 (2020) 1419–1440.

[2] A. Koshiyama, N. Firoozye, P. Treleaven, Generative adversarial networks for financial trading strategies fine-tuning and combination, Quantitative Finance 21 (2021) 797–813.

[3] R. Cont, M. Cucuringu, R. Xu, C. Zhang, Tail-gan: Nonparametric scenario generation for tail risk estimation, arXiv preprint arXiv:2203.01664 (2022).

[4] M. Bergeron, N. Fung, J. Hull, Z. Poulos, A. Veneris, Variational autoencoders: A hands-off approach to volatility, The Journal of Financial Data Science 4 (2022) 125–138.

[5] H. Buehler, B. Horvath, T. Lyons, I. P. Arribas, B. Wood, A data-driven market simulator for small data environments, arXiv preprint arXiv:2006.14498 (2020).

[6] M. Vuletić, F. Prenzel, M. Cucuringu, Fin-gan: Forecasting and classifying financial time series via generative adversarial networks (2023).

[7] P. Carr, H. Zhang, O. Hadjiliadis, Maximum drawdown insurance, International Journal of Theoretical and Applied Finance 14 (2011) 1195–1230.

[8] K. Atteson, P. Carr, Carr memorial: Maximum drawdown derivatives to a hitting time, The Journal of Derivatives 30 (2022) 16–31.

[9] A. Chekhlov, S. Uryasev, M. Zabarankin, Portfolio optimization with drawdown constraints, in: Supply chain and finance, World Scientific, 2004, pp. 209–228.

[10] A. Chekhlov, S. Uryasev, M. Zabarankin, Drawdown measure in portfolio optimization, International Journal of Theoretical and Applied Finance 8 (2005) 13–58.

[11] P. Lévy, Sur certains processus stochastiques homogènes, Compositio mathematica 7 (1940) 283–339.

[12] R. Douady, A. N. Shiryaev, M. Yor, On probability characteristics of" downfalls" in a standard brownian motion, Theory of Probability & Its Applications 44 (2000) 29–38.

[13] A. Rej, P. Seager, J.-P. Bouchaud, You are in a drawdown. when should you start worrying?, Wilmott 2018 (2018) 56–59.

[14] C. M. Bishop, N. M. Nasrabadi, Pattern recognition and machine learning, volume 4, Springer, 2006.

[15] A. Kondratyev, C. Schwarz, The market generator, Available at SSRN 3384948 (2019).

[16] P. Henry-Labordere, Generative models for financial data, Available at SSRN 3408007 (2019).

[17] P. Smolensky, Information processing in dynamical systems: Foundations of harmony theory, Technical Report, Colorado Univ at Boulder Dept of Computer Science, 1986.

[18] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial networks, Communications of the ACM 63 (2020) 139–144.

[19] C. Cuchiero, W. Khosrawi, J. Teichmann, A generative adversarial network approach to calibration of local stochastic volatility models, Risks 8 (2020) 101.

[20] H. Ni, L. Szpruch, M. Wiese, S. Liao, B. Xiao, Conditional sig-wasserstein gans for time series generation, arXiv preprint arXiv:2006.05421 (2020).

[21] J. Li, X. Wang, Y. Lin, A. Sinha, M. Wellman, Generating realistic stock market order streams, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pp. 727–734.

[22] V. Storchan, S. Vyetrenko, T. Balch, Mas-gan: Adversarial calibration of multi-agent market simulators. (2020).

[23] G. Benedetti, Yield curve quantization and simulation with neural networks, Available at SSRN 3577555 (2020).

[24] T. Xu, L. K. Wenliang, M. Munn, B. Acciaio, Cot-gan: Generating sequential data via causal optimal transport, arXiv preprint arXiv:2006.08571 (2020).

[25] F. D. M. Pardo, R. C. López, Mitigating overfitting on financial datasets with generative adversarial networks, The Journal of Financial Data Science 2 (2020) 76–85.

[26] H. Buehler, P. Murray, M. S. Pakkanen, B. Wood, Deep hedging: Learning to remove the drift under trading frictions with minimal equivalent near-martingale measures, arXiv preprint arXiv:2111.07844 (2021).

[27] H. Ni, L. Szpruch, M. Sabate-Vidales, B. Xiao, M. Wiese, S. Liao, Sig-wasserstein gans for time series generation, arXiv preprint arXiv:2111.01207 (2021).

[28] M. Pfenninger, S. Rikli, D. N. Bigler, Wasserstein gan: Deep generation applied on financial time series, Available at SSRN 3877960 (2021).

[29] A. Rosolia, J. Osterrieder, Analyzing deep generated financial time series for various asset classes, Available at SSRN 3898792 (2021).

[30] J. van Rhijn, C. W. Oosterlee, L. A. Grzelak, S. Liu, Monte carlo simulation of sdes using gans, arXiv preprint arXiv:2104.01437 (2021).

[31] G. Marti, V. Goubet, F. Nielsen, ccorrgan: Conditional correlation gan for learning empirical conditional distributions in the elliptope, in: International Conference on Geometric Science of Information, Springer, pp. 613–620.

[32] B. Coyle, M. Henderson, J. C. J. Le, N. Kumar, M. Paini, E. Kashefi, Quantum versus classical generative modelling in finance, Quantum Science and Technology 6 (2021) 024013.

[33] M. Wiese, B. Wood, A. Pachoud, R. Korn, H. Buehler, P. Murray, L. Bai, Multi-asset spot and option market simulation, arXiv preprint arXiv:2112.06823 (2021).

[34] E. Lezmi, J. Roche, T. Roncalli, J. Xu, Improving the robustness of trading strategy backtesting with boltzmann machines and generative adversarial networks, Available at SSRN 3645473 (2020).

[35] R. Wang, Discriminating modelling approaches for point in time economic scenario generation, arXiv preprint arXiv:2108.08818 (2021).

[36] N. J. C.-k. Fung, Variational Autoencoders for Volatility Surfaces, Ph.D. thesis, 2021.

[37] M. G. Frandsen, Greeks need not apply (2021).

[38] M. Bergeron, N. Fung, Z. Poulos, J. C. Hull, A. Veneris, Variational autoencoders: A hands-off approach to volatility, Available at SSRN 3827447 (2021).

[39] B. Ning, S. Jaimungal, X. Zhang, M. Bergeron, Arbitrage-free implied volatility surface generation with variational autoencoders, arXiv preprint arXiv:2108.04941 (2021).

[40] T. Lyons, Rough paths, signatures and the modelling of functions on streams, arXiv preprint arXiv:1405.4537 (2014).

[41] P. K. Friz, M. Hairer, A course on rough paths, Springer, 2020.

[42] I. Chevyrev, H. Oberhauser, Signature moments to characterize laws of stochastic processes, arXiv preprint arXiv:1810.10971 (2018).

[43] D. Levin, T. Lyons, H. Ni, Learning from the past, predicting the statistics for the future, learning an evolving system, arXiv preprint arXiv:1309.0260 (2013).

[44] H. Boedihardjo, T. Lyons, D. Yang, Uniform factorial decay estimates for controlled differential equations, Electronic Communications in Probability 20 (2015) 1–11.

[45] L. De Branges, The stone-weierstrass theorem, Proceedings of the American Mathematical Society 10 (1959) 822–824.

[46] N. E. Cotter, The stone-weierstrass theorem and its application to neural networks, IEEE transactions on neural networks 1 (1990) 290–295.

[47] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural networks 4 (1991) 251–257.

[48] T. Lyons, A. D. McLeod, Signature methods in machine learning, arXiv preprint arXiv:2206.14674 (2022).

[49] I. Chevyrev, A. Kormilitzin, A primer on the signature method in machine learning, arXiv preprint arXiv:1603.03788 (2016).

[50] T. J. Lyons, M. Caruana, T. Lévy, Differential equations driven by rough paths, Springer, 2007.

[51] T. Hastie, R. Tibshirani, J. H. Friedman, J. H. Friedman, The elements of statistical learning: data mining, inference, and prediction, volume 2, Springer, 2009.

[52] P. Kidger, T. Lyons, Signatory: differentiable computations of the signature and logsignature transforms, on both cpu and gpu, arXiv preprint arXiv:2001.00706 (2020).

[53] D. P. Kingma, M. Welling, Stochastic gradient vb and the variational auto-encoder, in: Second International Conference on Learning Representations, ICLR, volume 19, p. 121.