Working paper

Portfolio drawdown optimization with generative machine learning: opportunities and pitfalls

Emiel Lemahieu[*1*2], Kris Boudt[*1]

Abstract – In light of the increased popularity and advantages of risk-based portfolio optimization in general, the minimum drawdown or peak-to-valley loss portfolio in particular offers an (1) asymmetrical and (2) autocorrelated definition of portfolio risk that results in improved long-run risk-adjusted returns. As a path-based optimizer, it imposes an objective directly on the features of the portfolio path, rather than on some parametric representation of the input paths (i.e. a variance-covariance matrix). This poses the modeling challenge of generating a host of paths that closely resemble the path features we are optimizing for. The historical time-integrated mean of these path features generally does not equate their *ex ante* expectation, such that we propose to approximate this forward-looking expectation using an ensemble mean simulated with a generative machine learning (ML) model that learns to dynamically replicate the relevant path features in synthetic samples. We introduce a kernel trick based on the universality of a non-parametric embedding of paths, called the *signature transform*, to linearly approximate expected drawdowns. After emphasizing the relevance of risk-based optimization in general, we delve into the properties of the min drawdown portfolio in particular. We perform a long-run backtest on a multi-asset class universe of equities, fixed income, commodity and real estate indices, and find superior risk-adjusted returns. We compare naive historical simulation with ML-generated paths and find evidence that an ensemble of noisy paths can bring us closer to the expected drawdown than a historical mean. Well aware that ML comes at the price of increased model complexity, our conclusions highlight the main opportunities and pitfalls of using data-driven methods for these applications.

Keywords : Portfolio Construction, Drawdown, Machine Learning, Signatures, Variational Autoencoder

1 Introduction

We have seen a proliferation of applications of risk-based optimization, or portfolio construction with a risk objective only, in recent years. The reason is twofold. Firstly, expected returns required in mean-risk optimization (such as mean-variance) are extremely hard to estimate and often introduce more noise than signal to the model (DeMiguel, Garlappi, and Uppal 2009). Secondly, increasingly popular min vol indices and related exchange-traded funds (ETFs) have delivered on- to above-par returns compared to their reference benchmarks over the longer-run[1], while displaying lower volatility, hence offering better risk-adjusted returns. Even for the ones with on-par performance, the reduction of risk is highly useful in new fintech applications such as robo-advisors, where retail investors are notoriously sensitive to market fluctuations in terms of depositing, withdrawing and even closing the account. To a certain extent this is also true for traditional funds, but with less sticky capital the need for a 'smoother ride' investment becomes more prevalent. For reasons elaborated below, drawdown optimization is particularly adept to achieve a smooth ride, but it is still relatively unexplored for these kind of applications.

This paper is structured as follows. In Section 2, we introduce the minimum drawdown portfolio and delve deeper into the properties that make it well-suited for these applications. Section 3 discusses the impact of the sampler method chosen for the paths on which the objective is imposed. Section 4 elaborates the generative machine learning architecture, a variational autoencoder (VAE). In Section 5, we introduce a new machine learning sampling method that linearly approximates *expected* drawdowns. This allows for taking an ensemble mean of noisy samples, instead of the historical time-integrated

[*1]Ghent University, Sint-Pietersplein 6, 9000 Gent, Belgium
[*2]Corresponding author at emiel.lemahieu@ugent.be

[1]e.g. D ' Auria and McDermott 2017

mean. Next in Section 6, we compare the long-run risk-adjusted returns of the set of risk-based methods on a multi-asset class universe of US Bonds, US Equity, NAREITs real estate and GSCI commodities. Section 7 discusses the advantages and possible drawbacks of using machine learning for these problem settings. Section 8 concludes.

2 Drawdown optimization

For an N-dimensional universe of instruments, let us denote by $\mathbf{w}$ the vector of weights $w_i, i \in \{1, ..., N\}$. Further, $\mathbf{\Sigma}$ is the sample variance-covariance matrix of their historical return timeseries $\mathbf{X} : [0, T] \to \mathbb{R}^N$, where $x_{i,t} = s_{i,t}/s_{i,t-1} - 1$, and $\mathbf{S} : [0, T] \to \mathbb{R}^N$ is the T by N matrix of historical spot prices or index levels. For convenience, let us also write down $\sigma^{\mathbf{2}} = diag(\mathbf{\Sigma})$, where σ_i corresponds to individual asset i's volatility. We introduce the minimum drawdown portfolio in terms of these notations below[2].

2.1 Minimum drawdown portfolio

The minimum drawdown portfolio is the solution to the following linear optimization problem:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \mathbb{E}(\xi(\mathbf{w})) \\ \text{s.t.} \quad & \xi_{\mathbf{t}} = \mathbf{m_t} - \mathbf{w}\mathbf{S}_t \\ & \mathbf{m}_t \geq \mathbf{m}_{t-1} \\ & \mathbf{m}_t \geq \mathbf{w}\mathbf{S}_t \\ & \mathbf{w}\mathbf{I}^N = 1 \\ & \mathbf{w} > 0 \end{aligned} \tag{1}$$

where we minimize the expected drawdown ξ as a function of portfolio weights $\mathbf{w}$. The drawdown ξ is a non-linear function of the portfolio path $\mathbf{P_t} = \mathbf{w}\mathbf{S}_t$, $\xi_{\mathbf{t}} = \max(\max_{t_i<t}(\mathbf{P_{t_i}}) - \mathbf{P_t}, 0)$, but can hence be written as a linear problem by instrument variable $\mathbf{m_t}$ which denotes the monotonic growth of the portfolio value $\mathbf{m}_t \geq \mathbf{w}\mathbf{S}_t$. Chekhlov, Uryasev, and Zabarankin 2005 show that the minimum drawdown measure satisfies the properties of a deviation measure[3] and generalizes them to a dynamic conditional. The focal element of (1) for this paper is the expectation $\mathbb{E}(\xi)$ which can be taken over time $[0, T]$ as a (ergodic) time-integral, or integrated over a (non-ergodic) noisy ensemble of paths (see Section 3).

[2]The benchmark portfolios - minimum volatility, inverse volatility, equal weighting - are in terms of the above notation introduced in Appendix 1.

[3]More specifically, (1) non-negativity, (2) insensitivity to a constant shift, (3) positive homogeneity and (4) convexity.

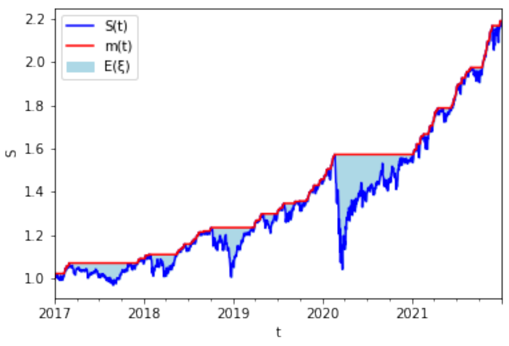

Fig. 1: Example of ξ, m_t and S_t for the US Equity index (S&P500)

2.2 On the performance difference between drawdown- and volatility-based methods

The main drivers of the theoretical performance difference between the drawdown-based method and volatility-based methods (cf. Appendix 1) are (1) *asymmetry* (positive surprises are not penalized to the same extent as negative ones), and (2) *autocorrelation* (consecutive negative surprises are more penalized than lowly autocorrelated losses)[4]. The latter path dependency is both a key strength of the approach and a key modeling challenge (see Section 3).

Let us start with a simple toy example, N=2, and then refer to the real-life backtest in Section 6. The example can be seen in Fig. 2, and is summarized in Table 1. The example is constructed such that two instruments Instr 1 and Instr 2 have (1) the same volatility $\sigma_1 = \sigma_2$ over a one-year period (250 days) of 9%, (2) a correlation of $\rho = -0.5$. The minimum volatility portfolio is in this case equal to the inverse volatility and the equally weighted portfolio, more specifically[5]:

$$w_{Instr1} = \frac{\sigma_2^2 - \sigma_1\sigma_2\rho}{\sigma_1^2 + \sigma_2^2 - 2\sigma_1\sigma_2\rho} \tag{2}$$

$$w_{Instr2} = 1 - w_{Instr1} \tag{3}$$

Given that the volatilities are symmetric, it is easy to see that $w_{Instr1} = 0.5 = w_{Instr2}$ for the min volatil-

[4]These key differences are illustrated for a simple N=2 example in Appendix 4.

[5]This result can easily be obtained by taking the Lagrangian of $var(w_{Instr1}X_1 + w_{Instr2}X_2)$, where $var(X_i) = \sigma_i^2$ and solving for w_{Instr1}. For completeness, this is included in Appendix 3.

	Vol (%)	ADD (%)	w Instr1(%)	w Instr2(%)
Min DD	6.85	0.53	75.29	24.71
Min Vol	5.86	0.66	50	50
Risk Par	5.86	0.66	50	50
Instr 1	9	0.55	100	0
Instr 2	9	3.52	0	100

Table 1: Toy example min vol versus min drawdown

ity portfolio, which is here equivalent to the inverse vol and equal weighted portfolio.

When critically looking at Fig. 2 and the drawdowns in Table 1 the instruments are far from 'symmetric' to the risk-averse investor. Both display low fluctuations, but Instr 1 has a steady increase (only a 0.55% drawdown) over the year, while Instr 2 accumulates small but steady losses (3.52% drawdown). Of course, the mean-variance investor will notice that the instruments seem to have different *drifts*, but including drift into the model has the well-known implications for overfitting and stability (DeMiguel, Garlappi, and Uppal 2009). For the volatility-based portfolios both assets are equally attractive in terms of risk, and such an optimizer will have no reason to divest low volatility instruments with persistent or autocorrelated small losses. Table 1 shows the optimal weights for the minimum drawdown portfolio. As can be seen Instr 1 is more preferred, since it has smaller losses and a shorter time to recover from these small losses. Remark that it is not fully concentrated in Instr 1 but has an approx. 75/25 distribution to achieve a 0.53% drawdown that is lower than Instr 1's drawdown.

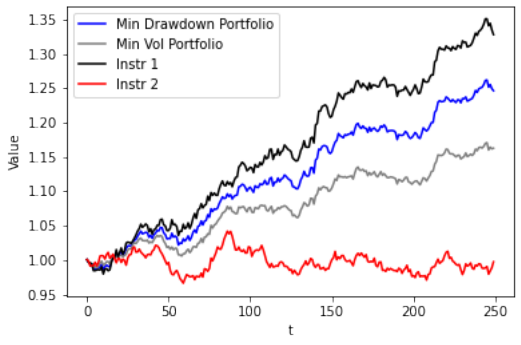

Fig. 2: Toy example (N=2) of min vol and inverse vol versus min drawdown

The analytical equivalent of Eq. (2) for the minimum drawdown portfolio requires an assumption on the path length and number of input paths as well. Even in the simplest case $N = 2$, $T = 2$ and 1 input path, the example becomes involved as Eq. (1) creates a variable m and slack variable for each time step and hence a factorial increase in complementary slack cases to be evaluated. We illustrate this in detail in Appendix 4. Although a trivial scenario compared to real-life applications with thousands of optimization variables, it already illustrates the non-linear and *asymmetric* definition of risk introduced in Eq. (1).

3 Historical scenarios versus generative ML

For the volatility-based methods, the estimation of the variance-covariance matrix $\mathbf{\Sigma}$ is a notoriously difficult exercise [6]. $\mathbf{\Sigma}$ is a parametric representation of the input paths on which we impose the objective. For the path-based method, however, we do not require an estimate of $\mathbf{\Sigma}$ from historical paths $\mathbf{S}$ but rather rely on those paths directly in (1). Taking the historical sample path, this results in one vector $\mathbf{wS_t}$, $t \in [0, T]$ and the expectation $\mathbb{E}(\xi(\mathbf{w}))$ corresponds to the historical time-integrated risk. This makes the historical sample-based path method, when compared to the sample-based volatility methods, completely *parameter-free*. As a key advantage over the other methods, one may still wonder whether the historical time-integrated average[7] is *sufficiently close*[8] to what we understand under *expected* drawdown: *ex ante* drawdown. This boils down to the implicit ergodicity assumption in sample-based methods:

$$\mathbb{E}(\xi) = \int_0^1 p(\xi) d\xi = \frac{1}{T} \int_0^T \xi(P_t) dt \tag{4}$$

where the time-integrated mean is assumed equal to the true expectation. This further would imply that for the expectation over many samples $j \in [0, N_s]$:

$$\mathbb{E}(\xi) = \frac{1}{N_s k} \int_0^{N_s} \int_0^k \xi(P_{t,j}) dt dj \tag{5}$$

the two integrals are commutative and equal to the simple $\int_0^T \xi(P_t)dt$. This is not what we generally observe in the data, as Eq. (4) assumes there is no noise in the observed $\xi(P_t)$. By contrast, the non-ergodic alternative implies that the integrals in Eq. (5) are

[6] E.g. due to the exponential scaling of the number of parameters and non-stationarity (time-varying volatility and correlation), which are tackled by shrunk covariances (Ledoit and Wolf 2004) or multi-factor models, and autoregressive or dynamic volatilities respectively

[7] For a formal definition see Peters 2019

[8] We will define this in the next section as the distance between simulated and true expected drawdown using the linear signature approximation.

non-commutative nor equal to the time-integrated average, and Eq. (5) is closer to the true $\mathbb{E}(\xi)$. Eq. (5) is an ensemble mean of many possible scenarios that are noisy by definition. Formalizing a procedure to generate these noisy samples is at the core of this paper.

In this paper we consider two alternatives, and propose two evaluation procedures. We compare historical simulation with a recent neural architecture called a variational autoencoder (VAE). These new generative models allow for increased flexibility in reproducing certain features of interest in the generated samples as compared to traditional Monte Carlo (cf. Section 5). First, we evaluate the similarity between the observed drawdown distribution $\mathbb{P}(\xi_i)$ for a range of scenarios i of both historical windows and ML generated samples with their time-shifted counterparts (future *expected* drawdown). Next, we integrate both models in our optimizer and compare the objective values in (1) with the out-of-sample drawdown.

4 Variational Autoencoders

This section discusses in more detail the variational autoencoder architecture, before we tune its objective to a proper Market Generator in the next section. It converges fast[9], is more stable than competing architectures, and it allow us to interpret the (conditional) distributions after training.

Below, we discuss the mappings $f_\Theta(X)$ and $f_\Theta^{-1}(Z)$ between the data and latent distributions, the original loss function $\mathcal{L}(X, X')$ (which we will revise in the next section), the training algorithm and the hyperparameters.

General architecture

The architecture of a VAE is summarized in Figure 5. As input we have the D-dimensional *ambient* space X or the physical data domain that we can measure X. Using a flexible neural network mapping $f_\Theta : \mathbb{R}^D \to \mathbb{R}^K, K << N$, called the encoder, we compress the dimension of the data into a K-dimensional *latent* space Z, e.g. 10-dimensional. Using the reparametrization trick (Kingma, Mohamed, et al. 2014) we map Z onto a mean μ and standard deviation σ vector, i.e. onto a K-dimensional Gaussian, e.g. a 10-dimensional multi-variate normal distribution. Starting from multi-variate normal data, we can recombine μ and σ into a K-dimensional Z. The decoder neural network $f_\Theta^{-1} : \mathbb{R}^K \to \mathbb{R}^D$ maps the latent space back to the output space $\mathbb{P}_\Theta(X')$ where X' can be considered *reconstructed* samples in the training step, or genuinely new or fake samples in a generator step. The quality of the VAE clearly depends on the similarity between $\mathbb{P}(X)$ and $\mathbb{P}_\Theta(X')$.

[9]First experiments with VAE resulted in similar performance metrics with GAN, where VAE was trained c.30 seconds and GAN c.30 minutes.

Encoder - decoder networks

Let us now zoom in on $f_\Theta(X)$ and $f_\Theta^{-1}(Z)$. Each neural network consist of one layer of J mathematical units called neurons:

$$f_{\Theta j} = A(\sum_i^D \theta_{i,j} x_i) \tag{6}$$

Every neuron takes *linear* combinations θ_i of the input data point x_i and is then *activated* using a *non-linear* activation function A, such as rectified linear units (ReLU), hyperbolic tangent (tanh) or sigmoid. In this paper we use a variant of ReLU called a *leaky ReLU*:

$$LReLU(x) = 1_{x<0}\alpha x + 1_{x\geq 0} x \tag{7}$$

where α is a small constant called the slope of the ReLU. All neurons J are linearly combined into the next layer (in this case Z):

$$Z_k := \sum_j^J \theta_{j,k} f_{\Theta j} \tag{8}$$

for every k in K. The decoder map can formally be written exactly like the encoder, but in reverse order.

Loss function

The loss function of a VAE generally consists of two components, the latent loss ($\mathcal{L}_L$) and the reconstruction loss ($\mathcal{L}_R$):

$$\mathcal{L}(X, X') = \beta \mathcal{L}_L + (1-\beta)\mathcal{L}_R \tag{9}$$

The latent loss is the Kullback-Leibler discrepancy between the latent distribution under its encoded parametrization, the posterior $f_\Theta(X) = \mathbb{P}_\Theta(Z|X)$, and its theoretical distribution, e.g. multi-variate Gaussian $\mathbb{P}(Z)$. Appendix B in Kingma and Welling 2014 offers a simple expression for $\mathcal{L}_L$. The reconstruction loss is the cost of reproducing $\mathbb{P}_\Theta(X')$ after the dimension reduction step, and originally computed by the root of the mean squared error (RMSE or $L2$-loss) between X and X'.

$$\mathcal{L}(X, X') = \beta \frac{1}{2} \sum_{k}^{K} (1 + \sigma - \mu^2 - \exp(\sigma)) + (1 - \beta)\mathbb{E}(||X - X'||^2) \tag{10}$$

Training

In terms of training, the learning algorithm is quasi identical to most deep learning methods. Optimal loss values $\mathcal{L}^*$ are determined by stochastically sampling batches of data and alternating forward and backward passes through the VAE. For each batch the data is first passed through the encoder network and decoder network (*forward pass*), after which $\mathcal{L}$ is evaluated in terms of Θ. At each layer, the derivative of $\mathcal{L}$ vis-a-vis Θ can easily be evaluated. Next (*the backward pass*), we say the calculated loss *backpropagates* through the network, and Θ are adjusted in the direction of the gradient $\nabla_\Theta \mathcal{L}$ with the *learning rate* as step size. The exact optimizer algorithm we used for this is Adam (Adaptive moments estimation, Kingma and Ba 2014). Finally, we also use a concept called regularization, which penalizes neural models that become too complex or overparametrized. We used a tool called dropout, that during training randomly sets a proportion of parameters in Θ equal to zero, and leaves those connections at zero that contribute the least to the prediction.

Hyperparameters

In summary, the hyperparameters of this architecture are: (1) the number of neurons in the encoder, (2) the number of neurons in the decoder, (3) the number of latent dimensions K, (4) the learning rate l and (5) the optimizer algorithm and (6) the dropout rate, (7) the batch size N_b, (8) batch length k and (9) number of training steps N. We opted for the following set-up, which was optimized using Grid Search: 50, 50, 10, 0.001, Adam, 0.01, 50, 22, 1000.

Generation

After training, in the sampling or generation step, we start from a random K-dimensional noise $\epsilon \sim \mathbb{P}(Z)$ which is K-variate Gaussian. Now, we simply need one decode step to generate new samples of $\mathbb{P}_\Theta(X')$.

5 Machine learning on paths: a simple Market Generator

Market Generators are generative machine learning models with the specificity of modeling financial markets, such as spot asset prices $\mathbf{S}$, option premia and implied volatilities, or order streams in limit order books. The topic has seen a recent surge in interest in the quantitative finance community (Wiese et al. 2020, Koshiyama, Firoozye, and Treleaven 2021, Buehler et al. 2020) as generative machine learning architectures[10] have found their way in simulation engines for optimal hedging, optimal order execution, backtesting trading strategies, and many more.

The aim of this paper is not to delve into their inner technical details[11], but to demonstrate their flexibility on the use case explained in Section 3, namely sampling realistic and predictive scenarios for a path-based portfolio optimizer.

Machine learning on paths[12] is a new subbranch of statistical learning and stochastic analysis that aims at summarizing path-structured data (often by means of the so-called *signature* in combination with a flexible mapping such as a neural network) and using that summary for inference or prediction on the path (e.g. human activity, brain functioning, handwritten digits, etc.). For the sake of space, we distilled two essential concepts from the field for our application here: (1) the signature transform and (2) the signature approximation.

Signature transform

The signature is defined in Eq. (11) and is the sequence of all iterated integrals of a path.

Definition (Chevyrev and Kormilitzin 2016): The signature of a path $\gamma : [0, T] \to \mathbb{R}^D$ denoted $S(\gamma)_{0,T}$ is the *collection* (an infinite series) of all the iterated integrals of γ. Formally, $S(\gamma)_{0,T}$ is the sequence of real numbers

$$S(\gamma)_{0,T} = (1, S(\gamma)^1_{0,T}, S(\gamma)^2_{0,T}, ..., S(\gamma)^D_{0,T}, S(\gamma)^{1,1}_{0,T}, S(\gamma)^{1,2}_{0,T}, ...) \tag{11}$$

where the zeroth term is 1 by convention and the

[10]Most notably generative adversarial networks (GAN), variational autoencoders (VAE) and restricted Boltzmann machines (RBM).

[11]For the architecture used here, refer to Section 4. For the other architectures please refer to the references and the references therein.

[12]For an introduction please refer to Lyons 2014, Levin, Lyons, and Ni 2013.

superscript runs along the set of *multi-indices*:

$$W = \{(i_1, i_2, ..., i_k) | k \geq 1; i_1, i_2, ..., i_k \in \{1, 2, ..., D\}\} \tag{12}$$

We often consider the M-th level truncated signature, defined as the finite collection of all terms where the superscript is of max length M:

$$S_M(\gamma) = (1, S^1(\gamma), S^2(\gamma), ..., S^M(\gamma)) \tag{13}$$

where $S^k(\gamma)$ denotes all the signature terms of order k, e.g.

$$S^1(\gamma) = (S(\gamma)^1, S(\gamma)^2, ..., S(\gamma)^D) \tag{14}$$

$$S^2(\gamma) = (S(\gamma)^{1,1}, S(\gamma)^{1,2}, ..., S(\gamma)^{D,D}) \tag{15}$$

It offers a graded[13] summary of paths, describing global and increasingly local features and preserving useful geometrical[14] properties. The signature is a bijective mapping between a path and a representation in a Hilbert space, that can be interpreted as a *feature mapping* in the machine learning sense[15]. Here we will focus on one property called the *universality* of the mapping, implying what we refer to as the signature approximation:

Universality and the signature approximation

A continuous function of the unparametrized data stream can be universally approximated by linear functionals in the signature space (see Levin, Lyons, and Ni 2013 Theorem 3.1):

Consider a compact set $K \subset \Omega([0,T], \mathbb{R}^N)$ and denote by S_M the signature transform truncated at level M. Let $f : K \to \mathbb{R}$ be any continuous function, then for any ϵ there exists a *linear* functional L acting on the truncated signature of degree M such that[16]:

$$\sup_{X \in K} |f(X) - \langle L, S_M(X) \rangle| < \epsilon \tag{16}$$

One such non-linear function was elaborated in Section 2.1, i.e. the expected time-integrated drawdown of a single path $f(P) = \int_0^T (\max_{t_i < t}(P_{t_i}) - P_t)dt$ is non-linear and not differentiable due to the max operation. Expensive sampling methods such as VAEs that numerically evaluate whether generated samples are *sufficiently close* to the original require (1) inexpensive criteria to converge with the available data and (2) differentiable criteria, such as commonly chosen squared error or cross-entropy. Moreover, f is just the function that links a path to its time-integrated expected drawdown in a computationally more efficient way (cf. infra), while the error towards its *expected* drawdown is the distance $\mathbb{E}_{\mathcal{B}} |\langle \hat{L}, S_M(X_{orig}) \rangle - \langle \hat{L}, S_M(X_{generated}) \rangle|$ where we should take the mean distance (L2-norm) over the batches $\mathcal{B}$ for each training epoch (See Appendix 4).

Therefore, we propose to initially use the training sample for a linear regression (OLS) of shorter trajectories (e.g. 20 days) on their drawdowns to find an estimate $\hat{L}$. Then we train a generative model where the signatures of the generated samples are linearly combined with $\hat{L}$ into an *expected* drawdown, and then compared with true *expected* (shifted) drawdown. This custom loss metric constitutes a reconstruction loss term in terms of Appendix 3 (section on Loss Function).

As we effectively compare inner products between mapped paths in Hilbert space, we can coin this linearization trick a *drawdown kernel trick*. However, the use of this trick can be extended to focusing on other path features than expected (mean) drawdown, such as dynamic generalizations of downside risk measures (VaR, CVaR, etc.). This is one of the opportunities discussed in Section 7. The detailed algorithm is provided below:

Algorithm 1 Market generator for path feature f

```
    Input  Historical price paths $X : [0,T] \to \mathbb{R}^N$, hyperparameters
           listed above + 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
        $k$, calculate the signatures of these paths truncated
        at level M, $S_M^{\mathcal{B}}$, calculate the drawdowns f of these
        paths $f(X^{\mathcal{B}}) = \int_0^k (max_{t_i<t}(X_{t_i}^{\mathcal{B}}) - X^{\mathcal{B}})dt$ denoted
        $\hat{f}(X_b)$
 3:     $\hat{L} \leftarrow LinearRegression(\hat{f}(X^{\mathcal{B}}), S_M^{\mathcal{B}})$
 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 en- .
            coder $g_\theta$ and decoder network $g_\theta^{-1}$
 7:         Calculate expected drawdown $\mathbb{E}(\xi)$ of the output
            sample $X'$ using the differentiable signature ap-
            proximation: $\langle \hat{L}, S_M(X') \rangle$
 8:         Define the reconstruction loss term as the .
            weighted average of RMSE error and drawdown
            loss: $\mathcal{L}_{\mathcal{R}} = \mathbb{E}_{\mathcal{B}}||X - X'||^2 + \alpha\mathbb{E}_{\mathcal{B}}||\langle \hat{L}, S_M(X) \rangle -$
            $\langle \hat{L}, S_M(X') \rangle||^2$
 9:         $\mathcal{L} = \mathcal{L}_L + \mathcal{L}_R$
10:         $\theta \leftarrow \theta - l\frac{d\mathcal{L}(\theta)}{d\theta}$
11:
```

[13]For factorial decay, see Appendix 2, Eq. (26)

[14]For these properties, see Appendix 2, section on geometric properties and financial interpretation

[15]E.g. for a maximum mean discrepancy (MMD) evaluation.

[16]From S_M being a tensor algebra of X it follows from the Stone-Weierstrass theorem that the family of all linear functionals on S_M is dense in the space of continuous functions on X.

Algorithm 2 Sampling from the Market generator

```
Input Trained VAE Market Generator $g_\theta$.
Output $N_s$ Generated samples X'

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

6 Long-term multi-asset class allocation problem

We run a long-term (30 year) backtest on 4 broad asset class indices: US Equity (S&P 500 index), US Bonds (Treasury index), Real Estate (NAREIT index), and Commodities (GSCI). The backtest covers Jan 1992 until Jan 2022, spanning the Dotcom bear market, the 2007-2008 Great Financial Crisis (GFC), and the 2009-2022 bull market (including the 2020 Covid collapse and sharp recovery).

Table 6 shows the performance of the Min Vol, Inverse Vol, Equal Weighted, Historical Min Drawdown (Min DD Hist) and Ensemble Min Drawdown (Min DD Ens) strategies, in terms of average Return and Volatility (Vol), both annualized, Average Drawdown (ADD) and the Sharpe ratio.

Portfolios were rebalanced monthly, using a rolling historical window of 2 years.

	Return (%)	**Vol (%)**	**ADD (%)**	**Sharpe**
Min Vol	6.187	**4.822**	-2.126	1.283
Min DD Hist	6.967	5.905	-1.911	1.179
Min DD Ens	7.069	5.488	**-1.827**	**1.288**
Equal Weight	**7.427**	10.22	-5.806	0.726
Inv Vol	7.087	7.736	-3.793	0.916

Table 2: Backtest performance

Figure 3 shows the evolution of the hypothetical portfolio paths over time (standardized to 100USD at the 1st of Jan 1992).

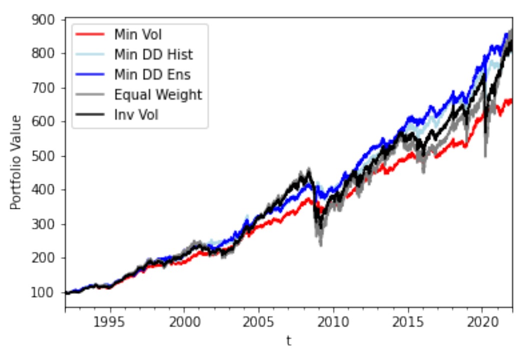

Fig. 3: Theoretical evolution of a 100USD portfolio

The difference in risk is most pronounced by looking at the evolution in drawdown (top-to-bottom loss) over the course of the 30 years. Figure 4 shows the so-called under-water curve for the Min DD Hist portfolio versus an equal weighted portfolio. During the DOTCOM crisis the -21% top-to-bottom loss of the equal weighted portfolio was limited to -8% for the Min DD Hist portfolio, for the Great Financial Crisis (GFC) and the Covid pandemic those figures were -50%/-13% and -28%/-10%.

While both strategies have an approx. 7% annual return, given that some investors are particularly prone to market fluctuations, these figures can make the difference between staying invested or leaving the market and never letting that 7% annual return materialize. Investors, and particularly retail investors, have path dependent risk preferences.

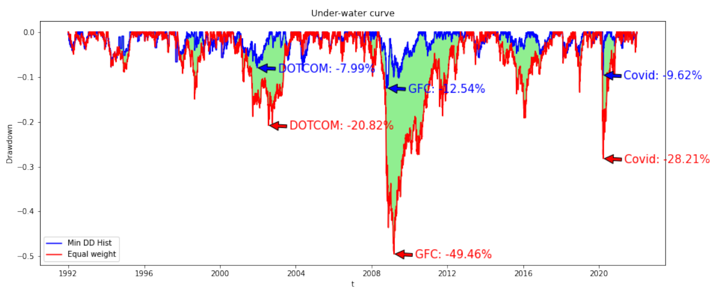

Fig. 4: Under-water curve

7 Machine learning on paths: opportunities and pitfalls

Generative machine learning opens new possibilities to look at past data and build expectations for the future. Rather than using historical data as a given to build expectations over time, it looks at chunks of that data, deliberately introduces noise to those scenarios, and then uses an ensemble (the expectation over batches) to push the generator model towards the right set of parameters that helps us distinguish between noise and the original signal in the future. Ironically, by adding noise to the data we find more structure, i.e. get closer to the true expectation, then by looking at the original samples. It has great promise, but with that comes additional caveats and risks.

Opportunities - The promises of generative ML for finance

We briefly discuss the opportunities below.

- **A truly flexible mapping**: neural networks (GAN, VAE, RBM) have the promise to approximate the underlying data-generating process (DGP), without requiring the statistician to specify it a priori. Compared to traditional Monte Carlo techniques, this has the potential to replicate richer features of the data set.

- **A truly flexible loss function**: rather than the mapping itself (the neural architecture), the objective of the generative model (also called the *loss function*, see Appendix 3) offers great promise due to its flexibility. As illustrated in this paper, the drawdown kernel trick can be used to approximate expected drawdowns and evaluate expected drawdown loss in a computationally efficient way. However, more path features could be of interest to the modeler and the *signature kernel trick* (Salvi, Cass, et al. 2021) can be leveraged for reproducing general path features (such as dynamic generalizations of other risk measures as value-at-risk (VaR) and expected shortfall (ES/CVaR) on the path space, e.g. conditional drawdown-at-risk (CDaR)).

- **Conditional sampling**: Another opportunity of not having to focus on the specification of the underlying DGP is the inclusion of exogenous variables in the simulation. Instead of a fully endogenous model of risk as proposed here, one could generate scenarios conditionally on economic conditions and evaluate the impact on the optimal portfolios. Conditional distributions have proven to better match the stylized facts[17] of financial timeseries in a handful of applications (Buehler et al. 2020, Kondratyev and Schwarz 2019).

- **Efficient computing, parallelization and quantum**: A last obvious, but important, point is the recent progress in high-performance computing and quantum computing. This made possible that expensive models that require thousands of evaluation steps to match the proposed criteria can be trained in a matter of minutes rather than days. After training, simulation is instant, and this makes real time calibration and optimization of (1) possible, whereas traditional sampling methods are notoriously slow and sample-based optimization is provably suboptimal.

[17] Cf. Cont 2001

Risks - Be careful what you wish for

- **Data issues**: It comes as no surprise that increasingly data-driven portfolio optimization methods become increasingly sensitive to data issues. Firstly, proper data cleansing for missing data and outliers is of utmost importance. The latter has been much debated, as outliers are typically the most important features of the data set. All in all, generative machine learning applications do a good job in generating timeseries with fat tails and are often closer to reality than traditional Monte Carlo that suffer from the same problems. Secondly, data availability can be a huge issue for data-driven methodologies. We worked on a small N=4 use case with relatively lots of data (30 years of daily data or 7500 data points), but 7500 samples is very limited for larger universes. The required number of samples depends on the use case and the difficulty and rate of convergence of the imposed objective. Caution and critical inspection of performance criteria are advised when training models, as compared to traditional analytical solutions a machine learning algorithm can only be trained to a local optimum.

- **Interpretability**: The generated paths should by construction be loyal to the historical sample and the focus is on simulation rather than direct prediction, so generative learning suffers less from the black box perception than traditional discriminative learning. However, paths are loyal but not identical, and the question arises on how to interpret trajectories that diverge from what happened in the past. One of the key properties of non-ergodicity is that historical samples did not reach all their possible states, such that such deviations are required to get closer to the actual expected state. At least that is the intuition, but this might still be confusing to quant modelers. Moreover, how to interpret deviations between the ML optimal portfolios from the historical sample optimal portfolios? Generally, they have the promise to be closer to the expected minimum risk but it is far from trivial to interpret individual cases. Finally, conditional sampling can help explaining how the minimal risk portfolio is constituted when key economic conditions change. This can help us using ML as an explanatory tool, rather than a prediction tool.

- **Model Complexity**: The modest increase in backtest performance induced by our Market Generator came at a substantial cost in model complexity. Not only for interpretability this can be problematic, but also for the implementation cost and model risk. Implementing this machine learning model is far from trivial and requires resources that sample estimates do not. Moreover, while it is known that sample variance-covariance matrices and other sample-based estimates misbehave (e.g. with various curses of dimensionality they get unreliable really quickly with increasing N), there is ample research as to how and when these models misbehave, so the experienced quant modeler will account for that. With novel machine learning methods, however, the main risk resides in the *unknown unknowns* when supposedly paradoxal results stem from some hidden assumption in the model, that oftentimes becomes unhidden at the most unpleasant of timings.

- **Architecture**: The choice of the architecture (GAN, VAE, RBM) can have a great impact on the performance of the model depending on the use case, both in terms of (1) statistical similarity with the original samples, as well as (2) flexibility of the model and (3) its computational performance. Generative ML offers great tools, but only in the hands of quant modelers who know how to leverage them for their use case.

- **Robustness**: While the whole point of simulation is to give more robust results in terms of Eq. (1), the generative model itself can lack robustness in terms of its hyperparameter choice (cf. Appendix 3). Standard techniques such as hyperparameter optimisation (mostly some form of Grid search with cross-validation on the data) are a bare minimum, and one should rather approach the whole modeling exercise from data cleansing, architecture choice and hyperparameter fine-tuning with due diligence.

8 Conclusions

This paper elaborated the opportunities and pitfalls of using generative machine learning for financial applications, such as long-term investing based on a sole risk objective. We started by comparing volatility- with path-based methods, and delved deeper into their performance delta. We then focused on the importance of the choice of the sampler for data on which the objective is imposed. The min risk objective should be seen as a 'deterministic' objective, where the answer is already in the data. The question is how to get most use of that data. A naive historical viewpoint is not the best way to go, because the time-integrated expectation of risk is ironically further away from the true expectation than the ensemble mean of noisy, ML generated samples. To use an analogy, to get a good perception of what a human face looks like, it is better to first add random attributes to the faces in your sample and then come to a joint consensus about what a common face looks like, rather then spending a lot of time watching 'familiar' faces and overfitting your perception on some of these characteristics. We concluded with a more bird's-eye evaluation of a data-driven approach for these kind of quantitative finance problem settings.

9 Appendix 1: Risk-based portfolio optimization

Risk-based portfolio optimization revolves around optimizing your portfolio with a risk objective solely, dropping expected returns (or assuming symmetrical returns) as they often introduce more noise than signal to the problem. This appendix introduces three benchmark methods.

9.1 Minimum volatility portfolio

The minimum volatility portfolio is the solution to the following quadratic optimization problem:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \mathbf{w\Sigma w^T} \\ \text{s.t.} \quad & \mathbf{wI}^N = 1 \\ & \mathbf{w} \geq 0 \end{aligned} \tag{17}$$

In other words, we pick the portfolio weights that minimize portfolio volatility, subject to making sure that the weights add up to one and a long-only constraint. Compared to traditional mean-variance optimization the returns are implicitly assumed to be symmetric, hence dropped from the objective.

9.2 Inverse volatility portfolio

The inverse volatility portfolio is composed such that all asset classes have weights proportional to their volatility. Without requiring optimization, we simply set the weights equal to:

$$\mathbf{w} = (1/\sigma_1, ..., 1/\sigma_N) \tag{18}$$

We can, however, formulate the inverse vol portfolio problem as optimization problem (17), where we assume both symmetric returns and correlations. Then the problem is reduced to the diagonal of $\mathbf{\Sigma}$ which is exactly (18).

9.3 Equally weighted portfolio

The equally weighted portfolio allocates equal proportions to all the assets in the investible universe:

$$\mathbf{w} = (1/N, ..., 1/N) \tag{19}$$

The '1/N'-portfolio minimizes model risk and assumes symmetrical returns, volatilities and correlations.

10 Appendix 2: Signatures

10.1 Iterated integrals

Let us recall path integrals. For a path $\gamma : [0, T] \to \mathbb{R}$ and a function $f : \mathbb{R} \to \mathbb{R}$, the path integral of γ against f is defined by

$$\int_0^T f(\gamma_t) d\gamma_t = \int_0^T f(\gamma_t) \frac{d\gamma_t}{dt} dt = \int_0^T f(\gamma_t) \dot{\gamma}_t dt \tag{20}$$

in which context f is called a *1-form* (Chen 1977). The last integral is a Riemann integral. Note that f itself is a real-valued path on $[0, T]$. This is a special case of the Riemann-Stieltjes integral of one path against another (Chevyrev and Kormilitzin 2016). In general, one can integrate any two paths on $[0, T]$, $\kappa : [0, T] \to \mathbb{R}, \gamma : [0, T] \to \mathbb{R}$, against one another:

$$\int_0^T \kappa_t d\gamma_t = \int_0^T \kappa_t \dot{\gamma}_t dt \tag{21}$$

Let us consider a particular path integral defined for any *single* index $i \in \{1, 2, ..., D\}$:

$$S(\gamma)^i_{0,T} = \int_0^T d\gamma^i = \gamma^i_T - \gamma^i_0 \tag{22}$$

which is the increment of the path along the dimension i in $\{1, 2, ..., D\}$. Now for any *pair* of indexes $i, j \in \{1, 2, ..., D\}$, let us define:

$$S(\gamma)^{i,j}_{0,T} = \int_0^T \int_0^{t_j} d\gamma^i d\gamma^j \tag{23}$$

and likewise for *triple* indices in $i, j, k \in \{1, 2, ..., D\}$:

$$S(\gamma)^{i,j,k}_{0,T} = \int_0^T \int_{t_k}^{t_j} \int_0^{t_k} d\gamma^i d\gamma^j d\gamma^k \tag{24}$$

and we can continue for the collection of indices $i_1, i_2, ..., i_k \in \{1, 2, ..., D\}$:

$$S(\gamma)^{i_1,i_2,...,i_j,...,i_k}_{0,T} = \int_0^T ... \int_{t_j}^{t_{j+1}} ... \int_{t_1}^{t_2} \int_0^{t_1} d\gamma^{i_1} d\gamma^{i_2} ... d\gamma^{i_k} \tag{25}$$

which we call the k-fold iterated integral of γ along $\{i_1, i_2, ..., i_k\}$.

10.2 Geometric and financial interpretation of a signature

As shown in (10.1), the geometric interpretation of the first order is the increment of the path along each dimension. In financial terms, this corresponds to the *drift*. It can be shown that the second order terms correspond to the *Levy area* (Chevyrev and Kormilitzin 2016), or the surface covered between the chord connecting the first and last coordinate in each dimension and the actual path, corresponding to a measure of *volatility* of the path. These two *global* features are captured by the first two orders, while more fine-grained, *local* features are captured by higher-order terms, as becomes apparent when looking at the factorial decay of S:

10.3 Factorial decay

One key property of signatures is factorial decay, which makes it a *graded summary* of paths.

As an analogue to the distributional[18] setting consider the well-known principal component analysis (PCA). In PCA we use linear combinations of the data X to decompose it into its components that maximise their variance while being mutually orthogonal (uncorrelated). It is equivalent to the eigendecomposition of the covariance matrix of X. A key

[18]Plain data generating processes of stochastic variables, in comparison to path-structured sequential random variables.

feature that we commonly see is exponential decay or rate decay, namely that the sorted absolute values of the eigenvalues of the covariance matrix of $X : \mathbb{R}^D$ decay fast enough, i.e. the j^{th} largest coefficient $|\beta|_j \leq Aj^{-a}, a \geq 1/2, \forall j$ and constants a and A do not depend on the dimension D. The latter simply implies that the first N components ($N << D$) already explain a vast part of the shared variance in the data set.

Informally, this intuition can be applied to paths using signatures as well. Lyons 2014 shows that for paths of bounded variation[19] the following similar norm can be imposed on the signature terms (with $1 \leq i_1, ..., i_n \leq D$):

$$||\int ... \int d\gamma^{i_1} d\gamma^{i_2} ... d\gamma^{i_n}|| \leq \frac{||\gamma^n||^1}{n!} \tag{26}$$

with

$$||\gamma||^1 = \sup_{t_i \subset [0,T]} \sum_i |\gamma_{t_{i+1}} - \gamma_{t_i}| \tag{27}$$

where we take the supremum over all partitions of [0,T].

This theorem proven in Lyons 2014 guarantees that higher-order terms of the signature have factorial decay, i.e. that the order of signatures imply a graded summary of the path, first describing global and increasingly more local characteristics of the path. This implies that the truncated signature for increasing orders throws away less and less information, similar to the low-rank approximation in PCA.

10.4 Signature as path moment generating function

Another key result that was recently developed by Chevyrev and Oberhauser 2018 is that the signature can be seen as the moment generating function in the path space.

In the distributional setting there are well-established metrics to compare two distributions. In machine learning, we often encounter distributional distance metrics from information theory, such as the Kullback-Leibler (KL) and Jensen-Shannon (JS) divergences between two distributions. For stochastic processes that generate vector-valued data, there are well-known statistical tests for determining whether two samples are generated by the same stochastic process, such as the sequence of (normalised) moments and the Fourier transform (complex moments).

For path-valued data, Chevyrev and Oberhauser 2018 introduce an analogue to normalised moments using the signature. They prove that for suitable normalizations λ, the sequence

$$(\mathbb{E}[\lambda(X)^m \int dX^{\otimes m}])_{m \geq 0} \tag{28}$$

determines the law of X *uniquely*[20]. They argue that the moments in the path space (or *sequential* moments) up to order m are preserved (i.e. a bijective property) for the truncated signature up to order m. Chevyrev and Kormilitzin 2016 proposes the use of this result with efficient algorithms and tools from machine learning such as MMD and kernel approximation (e.g. Salvi, Lemercier, et al. 2021) for machine learning paths. Chevyrev and Kormilitzin 2016 also argue in favor of signatures as a *provably* optimal feature map $\phi(.)$ for embedding paths generated by a stochastic process in into a linear space. The reasons are twofold: (1) *universality*, which implies that non-linear functions of the data are approximated by linear functionals in feature space (a key result used in this paper) and (2) *characteristicness*, which is exactly their merit, namely that the expected value of the feature map *characterizes the law of the random variable uniquely.*

[19] $\gamma : [0,T] \to \mathbb{R}$ is of bounded variation if all changes $\sum_i |\gamma_{t_{i+1}} - \gamma_{t_i}|$ are bounded (finite) for all partitions $0 \leq t_0 \leq t_1 \leq ... \leq T$

[20] Up to tree-like equivalance, see Chevyrev and Oberhauser 2018.

11 Appendix 3: Derivation of the minimum volatility portfolio for N = 2

This section briefly explains the derivation of formula (2). Denote again by $\mathbf{\Sigma}$ the variance-covariance matrix. When $N = 2$ the two diagonal elements σ_1^2 and σ_2^2 are the volatilities, and the two off-diagonal elements $\sigma_1\sigma_2\rho_{12}$ are the covariances, where ρ_{12} is the correlation.

The min vol portfolio can then be written as:

$$\min_{w_1,w_2} w_1^2\sigma_1^2 + w_2^2\sigma_2^2 + 2w_1w_2\sigma_1\sigma_2\rho_{12}$$
$$+\lambda_1(w_1+w_2-1)+\lambda_2(w_1+s_1^+)+\lambda_3(w_2+s_2^+) \tag{29}$$

or the variance of a two-asset portfolio written as the sum of the individual variances, covariances and Lagrange multipliers for the sum of weight and long-only constraints in (17). s_i^+ denote the slack variables for the $w_i \geq 0$ inequalities.

The first-order conditions equate the following set of derivatives to zero:

$$\frac{\delta\mathcal{L}}{\delta w_1} = 2w_1\sigma_1^2 + 2w_2\sigma_1\sigma_2\rho_{12} + \lambda_1 + \lambda_2 \tag{30}$$

$$\frac{\delta\mathcal{L}}{\delta w_2} = 2w_2\sigma_2^2 + 2w_1\sigma_1\sigma_2\rho_{12} + \lambda_1 + \lambda_3 \tag{31}$$

$$\frac{\delta\mathcal{L}}{\delta \lambda_1} = w_1 + w_2 - 1 \tag{32}$$

$$\frac{\delta\mathcal{L}}{\delta \lambda_2} = w_1 + s_1^+ \tag{33}$$

$$\frac{\delta\mathcal{L}}{\delta \lambda_3} = w_2 + s_2^+ \tag{34}$$

Assuming the case $s_1^+ > 0$, $s_2^+ > 0$, complementary slackness assures $\lambda_2 = \lambda_3 = 0$ such that:

$$\begin{aligned} w_1(2\sigma_1^2 - 2\sigma_1\sigma_2\rho_{12}) + 2\sigma_1\sigma_2\rho_{12} + \lambda_1 = 0 \\ 2\sigma_2^2 - 2w_1\sigma_2^2 + 2w_1\sigma_1\sigma_2\rho_{12} + \lambda_1 = 0 \end{aligned} \tag{35}$$

from (30) and (31) respectively and $w_1 = 1 - w_2$. Limited rearranging then yields:

$$w_1 = \frac{\sigma_2^2 - \sigma_1\sigma_2\rho_{12}}{\sigma_1^2 + \sigma_2^2 - 2\sigma_1\sigma_2\rho_{12}} \tag{36}$$

It is clear that the other cases for s_1^+ and s_2^+ are less relevant. Both equalling zero yields $w_1 = w_2 = 0$ and implies a violation of (32). One of them being zero automatically yields the other weight being 1, which is less optimal then the original case, unless in the edge case where $\rho = 1$ and the volatilities are not symmetrical, then the objective value is equal to the minimum individual volatility and optimal.

12 Appendix 4: Derivation of the minimum drawdown portfolio for N = 2

The minimum drawdown portfolio problem (Eq. (1)) for N = 2 is elaborated below. In contrast to the min vol portfolio for N = 2, the min drawdown portfolio also requires an assumption on the length of the path. We consider the simplest possible case where T = 2.

$$\min_{w_1,w_2} \tfrac{1}{2}\begin{bmatrix}\xi_1 & \xi_2\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}$$

s.t.

$$\begin{bmatrix}\xi_1\\\xi_2\end{bmatrix} = \begin{bmatrix}m_1\\m_2\end{bmatrix} - \begin{bmatrix}w_1 & w_2\end{bmatrix}\begin{bmatrix}P_{11} & P_{12}\\P_{21} & P_{22}\end{bmatrix}$$

$$\begin{bmatrix}m_1\\m_2\end{bmatrix} = \begin{bmatrix}w_1 & w_2\end{bmatrix}\begin{bmatrix}P_{11} & P_{12}\\P_{12} & P_{22}\end{bmatrix} + \begin{bmatrix}s_1^+\\s_2^+\end{bmatrix}$$

$$[m_2] = [m_1] + [s_3^+]$$

$$[1] = \begin{bmatrix}w_1 & w_2\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}$$

(37)

The Lagrangian is given by $\mathcal{L}$ below:

$$\begin{aligned} \mathcal{L}(\xi_1,\xi_2,m_1,m_2,w_1,w_2) = \frac{1}{2}(\xi_1+\xi_2) \\ +\lambda_1(m_1 - w_1P_{11} - w_2P_{21} - \xi_1) \\ +\lambda_2(m_2 - w_1P_{12} - w_2P_{22} - \xi_2) \\ +\lambda_3(-m_1 + w_1P_{11} + w_2P_{21} + s_1^+) \\ +\lambda_4(-m_2 + w_1P_{12} + w_2P_{22} + s_2^+) \\ +\lambda_5(-m_2 + m_1 + s_3^+) \\ +\lambda_6(w_1 + w_2 - 1) \\ +\lambda_7(w_1 + s_4^+) \\ +\lambda_8(w_2 + s_5^+) \end{aligned} \tag{38}$$

which already demonstrates that even in the simplest case the min drawdown problem is tedious in terms of notation. Let us therefore make another set of assumptions and simplifications: $s_4^+ > 0$ and $s_5^+ > 0$ such that $\lambda_7 = \lambda_8 = 0$, $w_1 = 1 - w_2$ (removing λ_6), $\xi_t = m_t - w_1P_{1t} - (1-w_1)P_{2t}$ (removing λ_1 and λ_2).

This gives the much simpler:

$$\begin{aligned}\mathcal{L}(m_1, m_2, w_1) = \\ +\frac{1}{2}(m_1 - w_1 P_{11} - (1-w_1)P_{21}) \\ +\frac{1}{2}(m_2 - w_1 P_{12} - (1-w_1)P_{22}) \\ +\lambda_3(-m_1 + w_1 P_{11} + (1-w_1)P_{21}) + s_1^+) \\ +\lambda_4(-m_2 + w_1 P_{12} + (1-w_1)P_{22}) + s_2^+) \\ +\lambda_5(-m_2 + m_1 + s_3^+)\end{aligned} \tag{39}$$

First-order conditions now correspond to equating the following derivatives to zero:

$$\begin{aligned}\frac{\delta\mathcal{L}}{\delta w_1} = \frac{P_{21}-P_{11}}{2} + \frac{P_{22}-P_{12}}{2} \\ +\lambda_3(P_{11}-P_{21}) + \lambda_4(P_{12}-P_{22})\end{aligned} \tag{40}$$

$$\frac{\delta\mathcal{L}}{\delta m_1} = \frac{1}{2} - \lambda_3 + \lambda_5 \tag{41}$$

$$\frac{\delta\mathcal{L}}{\delta m_2} = \frac{1}{2} - \lambda_4 - \lambda_5 \tag{42}$$

$$\frac{\delta\mathcal{L}}{\delta \lambda_3} = -m_1 + w_1 P_{11} + (1-w_1)P_{21} + s_1^+ \tag{43}$$

$$\frac{\delta\mathcal{L}}{\delta \lambda_4} = -m_2 + w_1 P_{12} + (1-w_1)P_{22} + s_2^+ \tag{44}$$

$$\frac{\delta\mathcal{L}}{\delta \lambda_5} = -m_2 + m_1 + s_3^+ \tag{45}$$

In theory, we now have to consider 8 cases for the slack variables s_i^+. However, thanks to complementary slackness we can see that if (45) is active, i.e. $s_3^+ = 0$, due to (42), (44) cannot be active. In other words, when a new running max cannot be achieved from combining the prices, (44) should not be active and the other way around. This creates a non-linear, *asymmetric* relationship between prices and weights, and can in this simple case be expressed in terms of the relative drifts.

Combining equations (43) and (44), and assuming $m_1 = w_1 P_{11} + (1-w_1)P_{21}$ or the first portfolio value is first maximum, $s_1^+ = 0$), we get:

$$w_1(P_{12}-P_{11}) + (1-w_1)(P_{22}-P_{21}) + s_2^+ - s_3^+ = 0 \tag{46}$$

$$w_1 = \frac{s_3^+ - s_2^+ - \Delta P_2}{\Delta P_1 - \Delta P_2} \tag{47}$$

The weights w_i in this simple case with only 2 time steps can be seen as proportional to their drift ΔP_i when compared with the change in running maximum. Slack variable s_3^+ can be seen as the increase in the running maximum $\Delta m = m_2 - m_1$ if $m_2 > m_1$, else 0. s_2^+ is the delta between the portfolio path at t=2, $w_1 P_{12} + (1-w_1)P_{22}$, and the previous running max if no new maximum can be achieved, else 0. Since we have separate cases for when $w_1 P_{11} + (1-w_1)P_{21} < w_1 P_{12} + (1-w_1)P_{22}$ and $w_1 P_{11} + (1-w_1)P_{21} > w_1 P_{12} + (1-w_1)P_{22}$ we achieve the non-linear and *asymmetric* relationship discussed in Section 2.

In terms of autocorrelation, it is clear that as the number of time steps T increases the number of slack variables denoting Δm increases and the host of non-linear relationships depending on the evolution of the portfolio value $w_1 P_{1t} + (1-w_1)P_{2t}, t \in [0, T]$ will explode. It becomes notationally impossible, and logically impractical, to illustrate this in a simple example. It is already clear from this example, however, that a declining or negatively evolving portfolio value over T increases the objective value, which illustrates the impact of autocorrelated losses, as opposed to volatility-based methods.

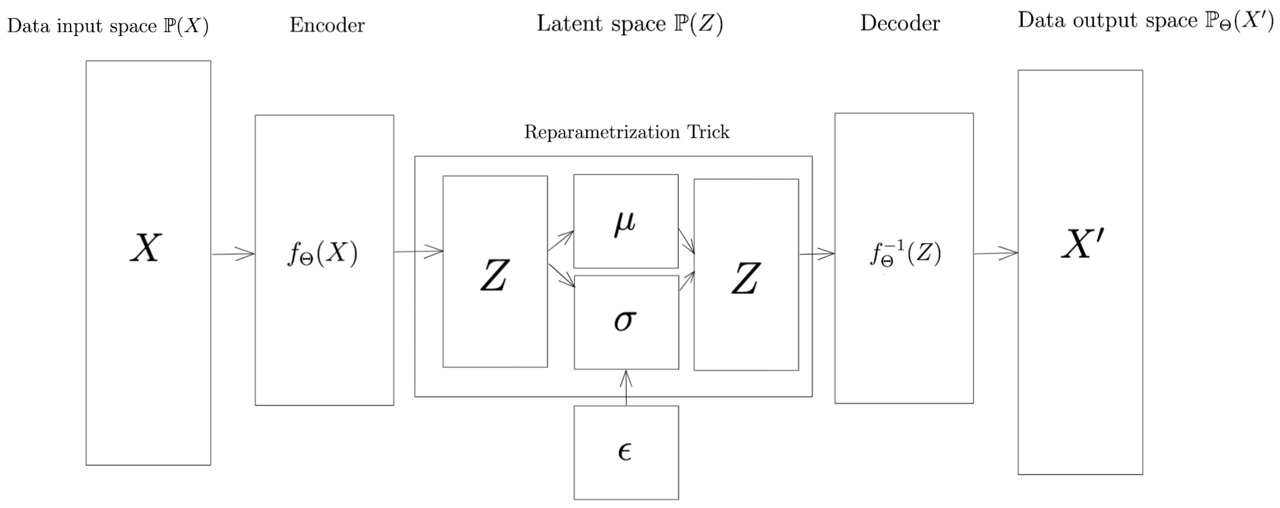

Fig. 5: Variational autoencoder architecture

References

Chen, Kuo-Tsai (1977). “Iterated path integrals”. In: *Bulletin of the American Mathematical Society* 83.5, pp. 831–879.

Cont, Rama (2001). “Empirical properties of asset returns: stylized facts and statistical issues”. In: *Quantitative finance* 1.2, p. 223.

Ledoit, Olivier and Michael Wolf (2004). “A well-conditioned estimator for large-dimensional covariance matrices”. In: *Journal of multivariate analysis* 88.2, pp. 365–411.

Chekhlov, Alexei, Stanislav Uryasev, and Michael Zabarankin (2005). “Drawdown measure in portfolio optimization”. In: *International Journal of Theoretical and Applied Finance* 8.01, pp. 13–58.

DeMiguel, Victor, Lorenzo Garlappi, and Raman Uppal (2009). “Optimal versus naive diversification: How inefficient is the 1/N portfolio strategy?” In: *The review of Financial studies* 22.5, pp. 1915–1953.

Levin, Daniel, Terry Lyons, and Hao Ni (2013). “Learning from the past, predicting the statistics for the future, learning an evolving system”. In: *arXiv preprint arXiv:1309.0260*.

Kingma, Diederik P and Jimmy Ba (2014). “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980*.

Kingma, Diederik P, Shakir Mohamed, et al. (2014). “Semi-supervised learning with deep generative models”. In: *Advances in neural information processing systems*, pp. 3581–3589.

Kingma, Diederik P and Max Welling (2014). “Stochastic gradient VB and the variational auto-encoder”. In: *Second International Conference on Learning Representations, ICLR*. Vol. 19, p. 121.

Lyons, Terry (2014). “Rough paths, signatures and the modelling of functions on streams”. In: *arXiv preprint arXiv:1405.4537*.

Chevyrev, Ilya and Andrey Kormilitzin (2016). “A primer on the signature method in machine learning”. In: *arXiv preprint arXiv:1603.03788*.

D ’ Auria, Dana M and John B McDermott (2017). “US Low and Minimum Volatility Indexes: An EmpiricalAnalysis of Factor Exposure”. In: *The Journal of Index Investing* 8.2, pp. 39–52.

Chevyrev, Ilya and Harald Oberhauser (2018). “Signature moments to characterize laws of stochastic processes”. In: *arXiv preprint arXiv:1810.10971*.

Kondratyev, Alexei and Christian Schwarz (2019). “The market generator”. In: *Available at SSRN 3384948*.

Peters, Ole (2019). “The ergodicity problem in economics”. In: *Nature Physics* 15.12, pp. 1216–1221.

Buehler, Hans et al. (2020). “A data-driven market simulator for small data environments”. In: *arXiv preprint arXiv:2006.14498*.

Wiese, Magnus et al. (2020). “Quant gans: Deep generation of financial time series”. In: *Quantitative Finance* 20.9, pp. 1419–1440.

Koshiyama, Adriano, Nick Firoozye, and Philip Treleaven (2021). “Generative adversarial networks for financial trading strategies fine-tuning and combination”. In: *Quantitative Finance* 21.5, pp. 797–813.

Salvi, Cristopher, Thomas Cass, et al. (2021). “The Signature Kernel is the solution of a Goursat PDE”. In: *SIAM Journal on Mathematics of Data Science* 3.3, pp. 873–899.

Salvi, Cristopher, Maud Lemercier, et al. (2021). “Higher Order Kernel Mean Embeddings to Capture Filtrations of Stochastic Processes”. In: *arXiv preprint arXiv:2109.03582*.