# Machine Learning in Finance

Josef Teichmann

Budapest October 2018

## Goal of this talk ...

- to present a standard perspective on machine learning.
- highlight on two examples (based on joint work with Bühler-Cuchiero-Gonon-Wood).
- the paradigm of reservoir computing.
- reservoir computing for scenario generation (based on joint work with Gonon-Grigoryeva-Ortega).

## Goal of this talk ...

- to present a standard perspective on machine learning.
- highlight on two examples (based on joint work with Bühler-Cuchiero-Gonon-Wood).
- the paradigm of reservoir computing.
- reservoir computing for scenario generation (based on joint work with Gonon-Grigoryeva-Ortega).

## Goal of this talk ...

- to present a standard perspective on machine learning.
- highlight on two examples (based on joint work with Bühler-Cuchiero-Gonon-Wood).
- the paradigm of reservoir computing.
- reservoir computing for scenario generation (based on joint work with Gonon-Grigoryeva-Ortega).

# Neural Networks

Neural networks are nowadays frequently used to approximate functions due to ubiquitous universal approximation properties. A neural network, as for instance graphically represented in Figure 1,

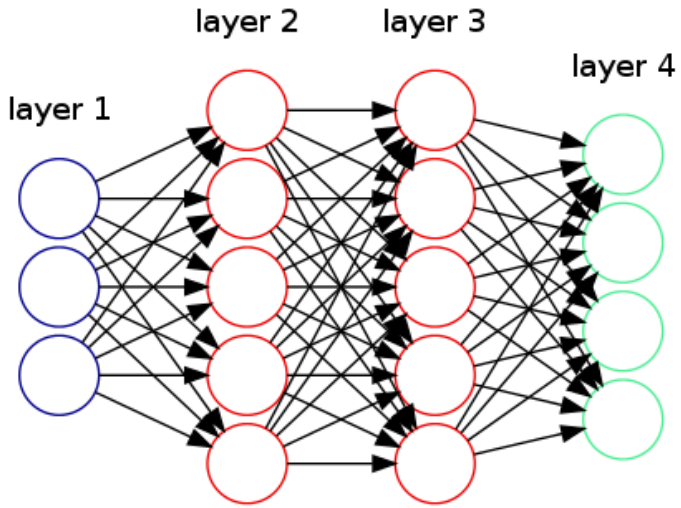



Figure: A 2 hidden layers neural network with 3 input and 4 output dimensions

just encodes a certain concatenation of affine and non-linear functions by composition in a well specified order. There are feed forward, recursive neural networks. Input as well as output dimension are fixed.

## Neural Networks and Universal Approximation

- Neural networks appeard in the 1943 seminal work by Warren McCulloch and Walter Pitts inspired by certain functionalities of the human brain aiming for artificial intelligence (AI).
- Arnold-Kolmogorov Theorem represents functions on unit cube by sums and uni-variate functions (Hilbert's thirteenth problem), i.e.

$$F(x_1, \ldots, x_d) = \sum_{i=0}^{2d} \varphi_i \Big( \sum_{j=1}^{d} \psi_{ij}(x_j) \Big)$$

- Universal Approximation Theorems (George Cybenko, Kurt Hornik, et al.) show that *one hidden layer networks* can already *uniformly approximate* any continuous function on the unit cube.

## Neural Networks and Universal Approximation

- Neural networks appeard in the 1943 seminal work by Warren McCulloch and Walter Pitts inspired by certain functionalities of the human brain aiming for artificial intelligence (AI).
- Arnold-Kolmogorov Theorem represents functions on unit cube by sums and uni-variate functions (Hilbert's thirteenth problem), i.e.

$$F(x_1, \ldots, x_d) = \sum_{i=0}^{2d} \varphi_i \Big( \sum_{j=1}^{d} \psi_{ij}(x_j) \Big)$$

- Universal Approximation Theorems (George Cybenko, Kurt Hornik, et al.) show that *one hidden layer networks* can already *uniformly approximate* any continuous function on the unit cube.

## Neural Networks and Universal Approximation

- Neural networks appeard in the 1943 seminal work by Warren McCulloch and Walter Pitts inspired by certain functionalities of the human brain aiming for artificial intelligence (AI).
- Arnold-Kolmogorov Theorem represents functions on unit cube by sums and uni-variate functions (Hilbert's thirteenth problem), i.e.

$$F(x_1, \ldots, x_d) = \sum_{i=0}^{2d} \varphi_i \Big( \sum_{j=1}^{d} \psi_{ij}(x_j) \Big)$$

- Universal Approximation Theorems (George Cybenko, Kurt Hornik, et al.) show that *one hidden layer networks* can already *uniformly approximate* any continuous function on the unit cube.

# Applications of neural networks

- *Learning* is the specification of a neural network which approximates a certain non-linear function on some input space.
- Modern learning technology performs training tasks in a highly accessible and very efficient way (Tensorflow, Theano, Torch).
- Deep neural networks (actually depth 4 is sufficient) easily approximate wavelet basis functions, whence the theory of all sorts of wavelet approximations of non-linear functions applies.
- "Unreasonable effectiveness" of learning (image and language recognition, classification tasks, etc).

# Applications of neural networks

- *Learning* is the specification of a neural network which approximates a certain non-linear function on some input space.
- Modern learning technology performs training tasks in a highly accessible and very efficient way (Tensorflow, Theano, Torch).
- Deep neural networks (actually depth 4 is sufficient) easily approximate wavelet basis functions, whence the theory of all sorts of wavelet approximations of non-linear functions applies.
- "Unreasonable effectiveness" of learning (image and language recognition, classification tasks, etc).

# Applications of neural networks

- *Learning* is the specification of a neural network which approximates a certain non-linear function on some input space.
- Modern learning technology performs training tasks in a highly accessible and very efficient way (Tensorflow, Theano, Torch).
- Deep neural networks (actually depth 4 is sufficient) easily approximate wavelet basis functions, whence the theory of all sorts of wavelet approximations of non-linear functions applies.
- "Unreasonable effectiveness" of learning (image and language recognition, classification tasks, etc).

# Applications of neural networks

- *Learning* is the specification of a neural network which approximates a certain non-linear function on some input space.
- Modern learning technology performs training tasks in a highly accessible and very efficient way (Tensorflow, Theano, Torch).
- Deep neural networks (actually depth 4 is sufficient) easily approximate wavelet basis functions, whence the theory of all sorts of wavelet approximations of non-linear functions applies.
- "Unreasonable effectiveness" of learning (image and language recognition, classification tasks, etc).

# Applications in Finance

- instead of using established numerical algorithms one *learns* solutions.
- Successfully implemented in several areas: deep hedging (Bühler-Gonon-Teichmann-Wood 2017), deep calibration (Cuchiero-Teichmann et al. 2016-17-18),
- we see solutions of problems which we have never seen before and we are at the edge of fully realistic modelling: high dimensional risk management tasks with trading constraints and transaction costs can be treated, high dimensional calibration problems can be solved in real time. Networks with *tens of thousands of parameters* are trained to perform this work.

# Applications in Finance

- instead of using established numerical algorithms one *learns* solutions.
- Successfully implemented in several areas: deep hedging (Bühler-Gonon-Teichmann-Wood 2017), deep calibration (Cuchiero-Teichmann et al. 2016-17-18),
- we see solutions of problems which we have never seen before and we are at the edge of fully realistic modelling: high dimensional risk management tasks with trading constraints and transaction costs can be treated, high dimensional calibration problems can be solved in real time. Networks with *tens of thousands of parameters* are trained to perform this work.

# Applications in Finance

- instead of using established numerical algorithms one *learns* solutions.
- Successfully implemented in several areas: deep hedging (Bühler-Gonon-Teichmann-Wood 2017), deep calibration (Cuchiero-Teichmann et al. 2016-17-18),
- we see solutions of problems which we have never seen before and we are at the edge of fully realistic modelling: high dimensional risk management tasks with trading constraints and transaction costs can be treated, high dimensional calibration problems can be solved in real time. Networks with *tens of thousands of parameters* are trained to perform this work.

# Discrete-time market model with frictions

- Trading: at time points $t_0 = 0 < t_1 < \ldots < t_n = T$.
- Prices of hedging instruments: stochastic process $(S_{t_k})_{k=0,\ldots,n}$ in $\mathbb{R}^d$.
- Work on a (finite) probability space $(\Omega, \mathcal{G}, \mathbb{P})$ with filtration $\mathbb{G} = (\mathcal{G}_{t_k})_{k=0,\ldots,n}$, for simplicity $\mathcal{G}_{t_k} = \sigma(S_{t_0}, \ldots, S_{t_k})$.
- At $t = 0$ sell a contingent claim with (random) payoff $Z$ at $T > 0$.
- Specify a (smaller) filtration $\mathbb{F} \subset \mathbb{G}$ for hedging.
- Charging price $p_0$ and hedging according to an $\mathbb{F}$-predictable strategy $\delta$, terminal profit and loss is (with $\cdot$ discrete-time stochastic integration)

$$\mathrm{PL}_T(Z, p_0, \delta) := -Z + \underbrace{p_0}_{\text{price}} + \underbrace{(\delta \cdot S)_T}_{\text{trading gains}} - \underbrace{C_T(\delta)}_{\text{cum. transaction costs}}.$$

# Setup and problem formulation in detail

$$\mathrm{PL}_T(Z, p_0, \delta) := -Z + \underbrace{p_0}_{\text{price}} + \underbrace{(\delta \cdot S)_T}_{\text{trading gains}} - \underbrace{C_T(\delta)}_{\text{cum. transaction costs}} . \qquad (1)$$

(1) in more detail:

- $(\delta \cdot S)_T = \sum_{k=1}^{n} \delta_{t_k} \cdot (S_{t_k} - S_{t_{k-1}})$.
- $C_T(\delta) = \sum_{k=0}^{n} c_k(\delta_{t_k} - \delta_{t_{k-1}}, S_{t_0}, \ldots, S_{t_k})$ with $\delta_{t_{-1}} := 0, \delta_{t_n} := 0$.
- Example: transaction costs proportional to transaction amount, i.e. $c_k(\delta_{t_k} - \delta_{t_{k-1}}, S_{t_0}, \ldots, S_{t_k}) = \sum_{i=1}^{d} \varepsilon_i |\delta^i_{t_k} - \delta^i_{t_{k-1}}| S^i_{t_k}$.
- Note: $\mathrm{PL}_T(Z, p_0, \delta) \geq 0$ represents a gain for seller.

# Setup and problem formulation in detail

$$\mathrm{PL}_T(Z, p_0, \delta) := -Z + \underbrace{p_0}_{\text{price}} + \underbrace{(\delta \cdot S)_T}_{\text{trading gains}} - \underbrace{C_T(\delta)}_{\text{cum. transaction costs}} . \qquad (1)$$

(1) in more detail:

- $(\delta \cdot S)_T = \sum_{k=1}^{n} \delta_{t_k} \cdot (S_{t_k} - S_{t_{k-1}})$.
- $C_T(\delta) = \sum_{k=0}^{n} c_k(\delta_{t_k} - \delta_{t_{k-1}}, S_{t_0}, \ldots, S_{t_k})$ with $\delta_{t_{-1}} := 0, \delta_{t_n} := 0$.
- Example: transaction costs proportional to transaction amount, i.e. $c_k(\delta_{t_k} - \delta_{t_{k-1}}, S_{t_0}, \ldots, S_{t_k}) = \sum_{i=1}^{d} \varepsilon_i |\delta^i_{t_k} - \delta^i_{t_{k-1}}| S^i_{t_k}$.
- Note: $\mathrm{PL}_T(Z, p_0, \delta) \geq 0$ represents a gain for seller.

# Indifference pricing and optimal hedging:

- Describe risk-preferences by a convex risk-measure $\rho$.
- Denote $\mathcal{H}$ the set of available hedging strategies.
- The indifference price is the (unique) solution $p(Z)$ to

$$\inf_{\delta \in \mathcal{H}} \rho\left(\mathrm{PL}_T(Z, p(Z), \delta)\right) = \inf_{\delta \in \mathcal{H}} \rho\left(\mathrm{PL}_T(0, 0, \delta)\right). \tag{2}$$

- Optimal hedging strategy is minimizer $\delta^*$ (if it exists) in left-hand-side of (2).

Numerical calculation of $p(Z)$ and $\delta^*$:

- Highly challenging by classical numerical techniques (very high-dimensional problem) in particular in the 2Filtration setting.
- $\rightarrow$ in practice more simplistic models are used (parametric, continuous-time, no transaction costs).

# Indifference pricing and optimal hedging:

- Describe risk-preferences by a convex risk-measure $\rho$.
- Denote $\mathcal{H}$ the set of available hedging strategies.
- The indifference price is the (unique) solution $p(Z)$ to

$$\inf_{\delta \in \mathcal{H}} \rho\left(\mathrm{PL}_T(Z, p(Z), \delta)\right) = \inf_{\delta \in \mathcal{H}} \rho\left(\mathrm{PL}_T(0, 0, \delta)\right). \tag{2}$$

- Optimal hedging strategy is minimizer $\delta^*$ (if it exists) in left-hand-side of (2).

Numerical calculation of $p(Z)$ and $\delta^*$:

- Highly challenging by classical numerical techniques (very high-dimensional problem) in particular in the 2Filtration setting.
- $\rightarrow$ in practice more simplistic models are used (parametric, continuous-time, no transaction costs).

- an approximate calculation is feasible thanks to modern deep learning techniques (exploited for other problems in finance e.g. in works by Cont, Sirignano, E, Han, Jentzen, Cheridito, Becker, ...).
- Straight forward Approach: consider only hedging strategies $\delta = (\delta_{t_k})_{k=1,\dots,n}$ of the form

$$\delta_{t_k} = F^{\theta_k}(S_{t_{k-1}}, \delta_{t_{k-1}}), \quad k = 1, \dots, n$$

  where $F^{\theta_k}$ is a neural network with weights parametrized by $\theta_k$.
- Key point 1: neural networks are surprisingly efficient at approximating multivariate functions (see works by Bölcskei, Grohs, Kutyniok, Petersen, Wiatowski, ...).
- Key point 2: efficient machine learning optimization algorithms (stochastic gradient-type and backpropagation) and implementations (Tensorflow, Theano, Torch, ...) are available.

# Approximate indifference price

- By cash-invariance, the indifference price $p(Z)$ is given as

$$p(Z) = \pi(-Z) - \pi(0), \text{ where}$$
$$\pi(X) := \inf_{\delta \in \mathcal{H}} \rho(X + (\delta \cdot S)_T - C_T(\delta)).$$

- For suitable parameter set $\Theta_M \subset \mathbb{R}^r$ and $\delta^\theta_{t_k} = F^{\theta_k}(S_{t_{k-1}}, \delta^\theta_{t_{k-1}})$ as above, approximate $\pi(X)$ by

$$\pi_M(X) := \inf_{\theta \in \Theta_M} \rho\Big(X + (\delta^\theta \cdot S)_T - C_T(\delta^\theta)\Big).$$

- Directly amenable to machine learning optimization algorithms (stochastic gradient-type and backpropagation) if $\rho$ is entropic risk measure ($\rightarrow$ exponential utility indifference price) or more generally an *optimized certainty equivalent*, i.e.

$$\rho(X) := \inf_{w \in \mathbb{R}} \{w + E[\ell(-X - w)]\} \tag{3}$$

for $\ell \colon \mathbb{R} \to \mathbb{R}$ continuous, non-decreasing and convex.

# Approximate indifference price

- By cash-invariance, the indifference price $p(Z)$ is given as

$$p(Z) = \pi(-Z) - \pi(0), \text{ where}$$
$$\pi(X) := \inf_{\delta \in \mathcal{H}} \rho(X + (\delta \cdot S)_T - C_T(\delta)) .$$

- For suitable parameter set $\Theta_M \subset \mathbb{R}^r$ and $\delta^\theta_{t_k} = F^{\theta_k}(S_{t_{k-1}}, \delta^\theta_{t_{k-1}})$ as above, approximate $\pi(X)$ by

$$\pi_M(X) := \inf_{\theta \in \Theta_M} \rho\Big(X + (\delta^\theta \cdot S)_T - C_T(\delta^\theta)\Big) .$$

- Directly amenable to machine learning optimization algorithms (stochastic gradient-type and backpropagation) if $\rho$ is entropic risk measure ($\rightarrow$ exponential utility indifference price) or more generally an *optimized certainty equivalent*, i.e.

$$\rho(X) := \inf_{w \in \mathbb{R}} \{w + E[\ell(-X - w)]\} \tag{3}$$

for $\ell \colon \mathbb{R} \to \mathbb{R}$ continuous, non-decreasing and convex.

# Example Study: Heston model with CVar

$$dS_t^{(1)} = \sqrt{V_t} S_t^{(1)} dB_t, \quad S_0^{(1)} = s_0$$

$$dV_t = \alpha(b - V_t)dt + \sigma\sqrt{V_t} dW_t, \quad V_0 = v_0$$

B and W are Brownian motions with $d\langle B, W\rangle = \rho dt$

$$(\alpha, b, \rho, \sigma, v_0, s_0) = (1, 0.04, -0.7, 2, 0.04, 100)$$

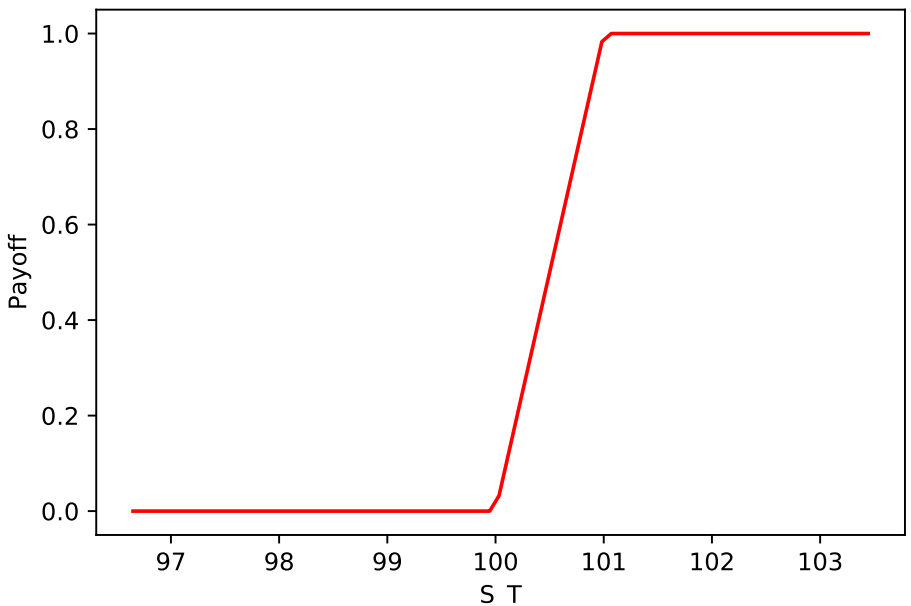
## Payoff and Hedging

- Payoff: Call spread (see next slide) with maturity $T = 30$ days.
- Hedging instruments: Trade in $S^{(1)}$ and variance swap $S^{(2)}$.
- Trading: Daily rebalancing of portfolio.
- Risk-measure: $\alpha$-CVar (expected shortfall),

$$\rho(X) := \inf_{w \in \mathbb{R}} \left\{ w + \frac{1}{1-\alpha} E[(-X - w)^+] \right\}.$$

## Call spread

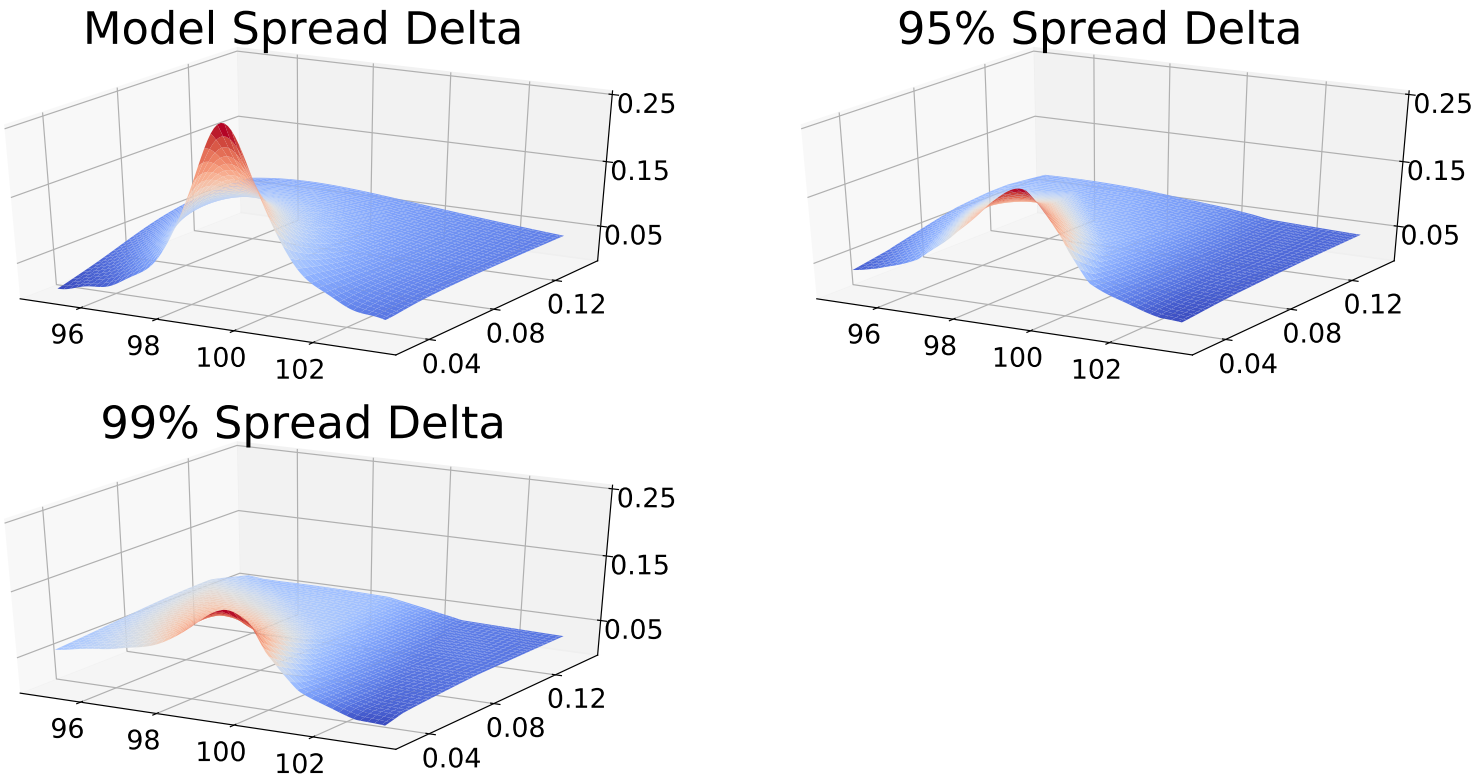
- Used by traders for (approximate) pricing / hedging of binary options.
- Payoff: $-\frac{1}{K_2-K_1}[(S_T^{(1)} - K_1)^+ - (S_T^{(1)} - K_2)^+]$ for $K_1 < K_2$.
- Here $K_1 = s_0 = 100$, $K_2 = 101$:



# Neural network approximation

- $\delta_{t_k} = F^{\theta_k}(S^{(1)}_{t_{k-1}}, S^{(2)}_{t_{k-1}}, \delta_{t_{k-1}})$ and for each $k$, $F^{\theta_k}$ is a feed-forward neural network with two hidden layers (15 nodes each) and ReLU activation function ($x \mapsto x^+$).
- Use Adam (batch size 256) for training.

# $\delta_t^{(s)}$ as a function of $(s_t, v_t)$ for $t = 15$:



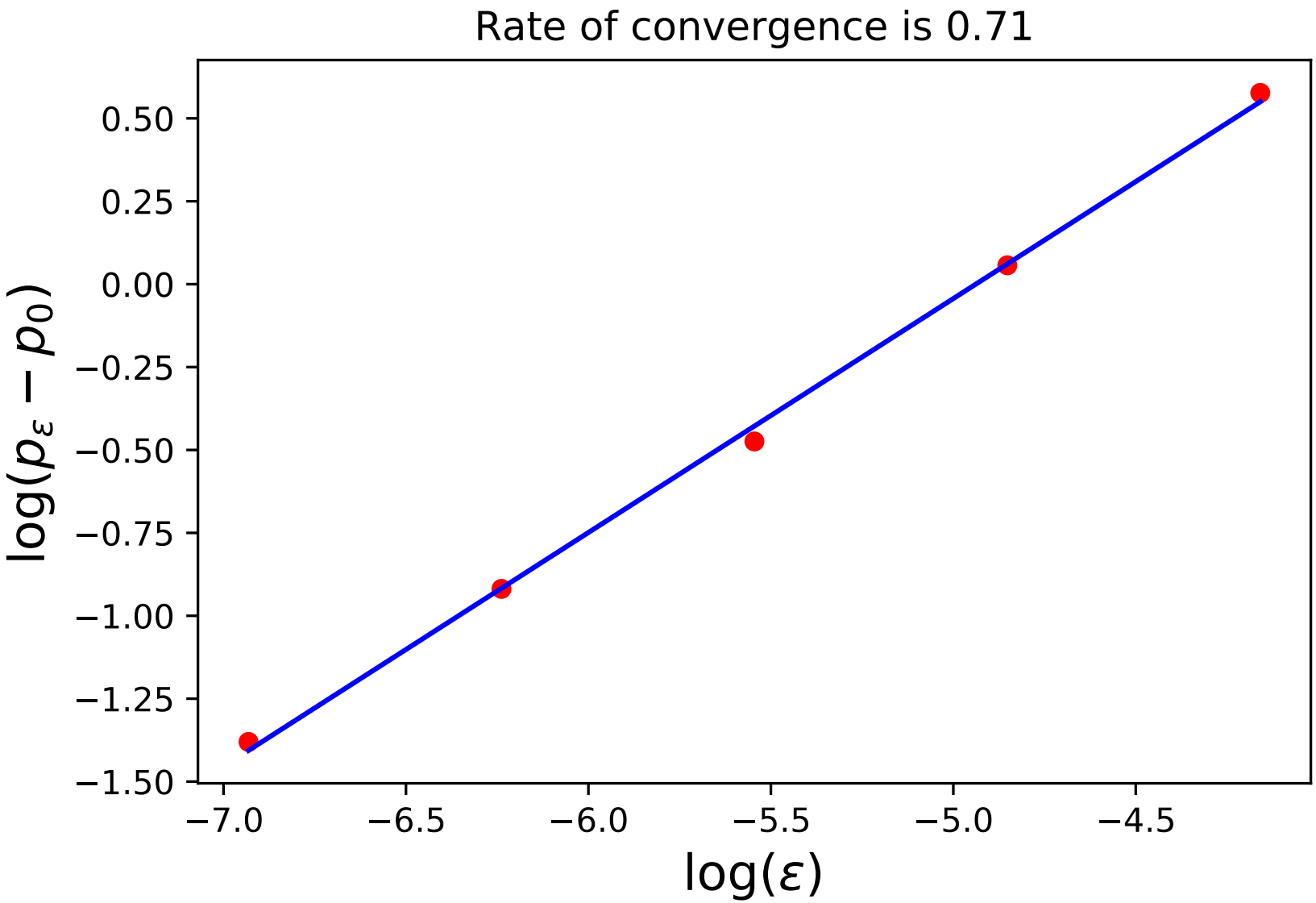

Higher risk-aversion $\leftrightarrow$ barrier shift

# Price asymptotics: proportional transaction costs

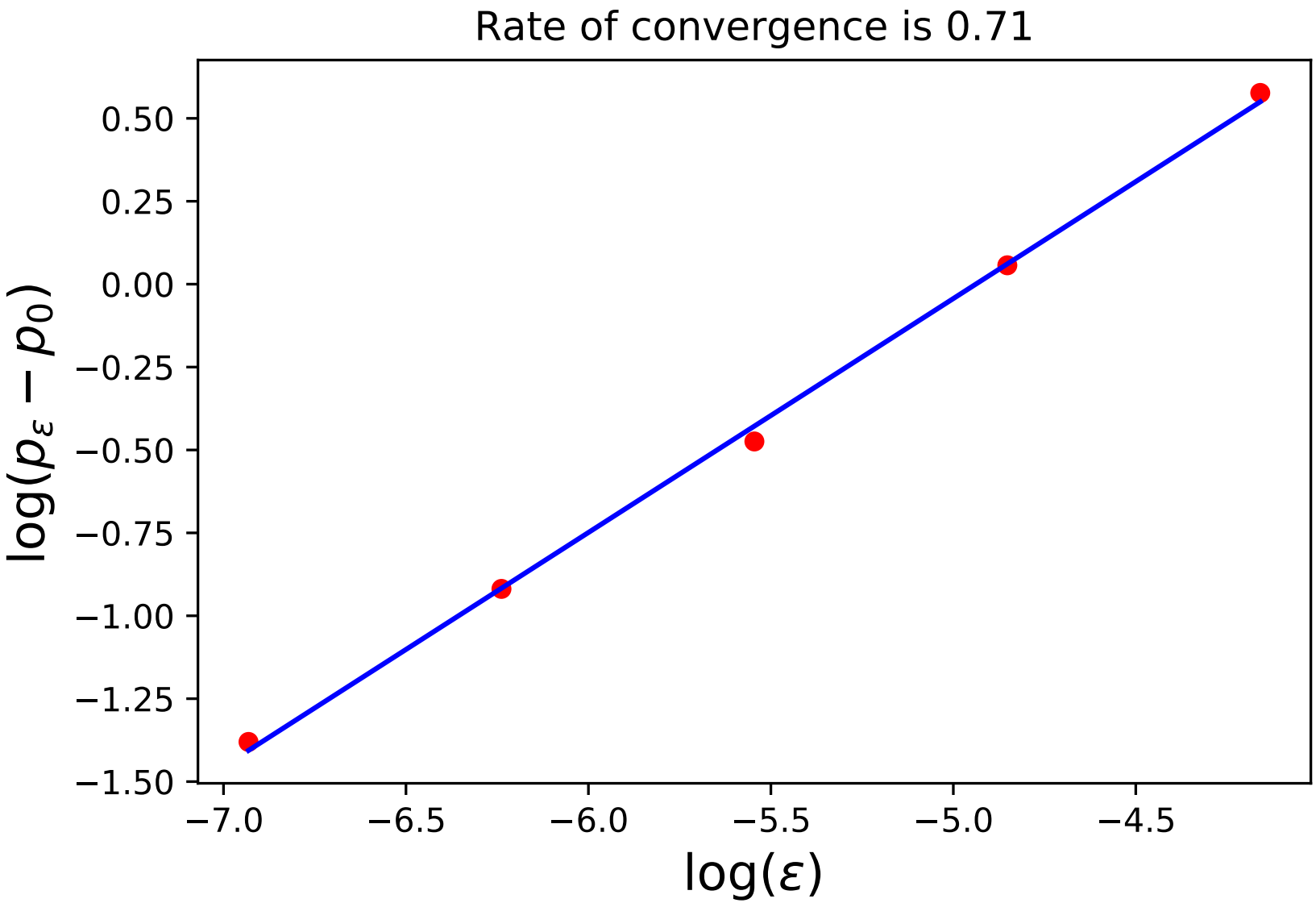
- $p_\varepsilon = p_\varepsilon(Z)$ is the exponential utility indifference price of $Z$ for proportional transaction costs of size $\varepsilon$.
- For continuous-time models with $d = 1$:

$$p_\varepsilon - p_0 = O(\varepsilon^{2/3}), \qquad \text{as } \varepsilon \downarrow 0. \tag{4}$$

- Our methodology is good enough to reproduce (4) in a Heston model with $d = 2$ hedging instruments. For this case (or any other model with $d > 1$) no results on (4) have been available previously (neither theoretical nor numerical).




Another instance of the Unreasonable effectiveness of neural networks!




Another instance of the Unreasonable effectiveness of neural networks!

# Concept

- learn the solution of the inverse problem, i.e. the map

$$\text{Data} \mapsto \text{Model parameters}$$

- with or without regularizations it works remarkably well. Examples from interest rate models (jointly with Cuchiero-Hernandez) or from stochastic volatility models (jointly with Cuchiero-Khosrawi).
- Several approaches: learn from offline calculations in a supervised way, or apply stochastic gradient flow methods directly to learn the map *inserted* in the pricing equations.

# Concept

- learn the solution of the inverse problem, i.e. the map

$$\text{Data} \mapsto \text{Model parameters}$$

- with or without regularizations it works remarkably well. Examples from interest rate models (jointly with Cuchiero-Hernandez) or from stochastic volatility models (jointly with Cuchiero-Khosrawi).
- Several approaches: learn from offline calculations in a supervised way, or apply stochastic gradient flow methods directly to learn the map *inserted* in the pricing equations.

# Concept

- learn the solution of the inverse problem, i.e. the map

$$\text{Data} \mapsto \text{Model parameters}$$

- with or without regularizations it works remarkably well. Examples from interest rate models (jointly with Cuchiero-Hernandez) or from stochastic volatility models (jointly with Cuchiero-Khosrawi).
- Several approaches: learn from offline calculations in a supervised way, or apply stochastic gradient flow methods directly to learn the map *inserted* in the pricing equations.

# Calibration of a Hull-White extended Vasicek model following Andres Hernandez

- Goal: calibrate the parameters of the Hull-White interest rate model using ATM swaptions and the current interest rate.
- The Hull-White model is given as follows

$$dr_t = (b(t) - \alpha r_t)dt + \sigma dW_t,$$

where $W$ is a Brownian motion, the mean reversion $\alpha$ and the volatility $\sigma$ are constants and $t \mapsto b(t)$ is a deterministic function one-to-one with the current yield curve $\varphi$.

- The problem is now: Find $h \in \mathcal{NN}_{M,K+\dim(\Phi),2}$ such that

$$(\alpha, \sigma) = h(\pi^{\text{mkt}}(\tau_1), \ldots, \pi^{\text{mkt}}(\tau_K), \varphi).$$

# Procedure suggested by A. Hernandez

- Get the historical price data, here a time series of swaption volatilities and the yield curves.
- Calibrate the Hull-White extended Vasicek model classically to obtain a time series of (typical) model parameters $\alpha$ and $\sigma$ (and the function $t \mapsto b(t)$).
- Pre-process the data and generate randomly new combinations of parameters $\alpha$, $\sigma$ and the yield curve (equivalently $t \mapsto b(t)$).
- Compute for these random parameters the swaption prices.
- The training data set now consists of a large set of inputs and outputs (swaption volatilities, yield curve) $\mapsto (\alpha, \sigma)$.
- Perform supervised learning, i.e. train a neural network on this training data set.
- Test the neural network on out-of-sample data.

# Data set and historical parameters

- Historical data from January 2nd, 2013 to June 1st, 2016, contains
  - ... ATM implied volatility quotes for GBP swaptions. The option maturities are 1 to 10 years, 15 years and 20 years. The swap terms from 1 to 10 years, plus 15, 20 and 25 years. Each day we have 156 data points.
  - ...yield curve data given at 44 maturities (0, 1, 2, 7, 14 days; 1 to 24 months; 3-10 years; plus 12, 15, 20, 25, 30, 40 and 50 years) bootstrapped from 6M-Libor products (FRAs and swaps).
- To obtain the historical parameters a Levenberg-Marquardt local optimizer implemented in Quant-Lib is first applied to minimize the equally-weighted average of squared yield or implied volatility differences.

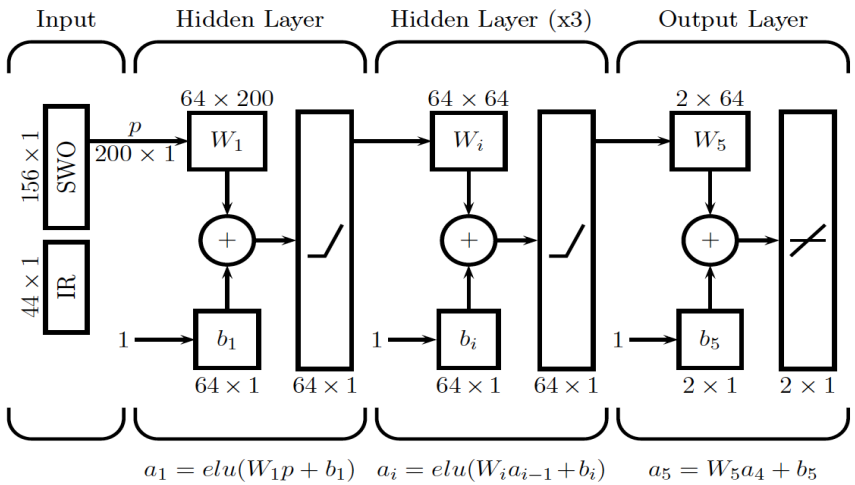
# Training data set generation

From this calibration history one generates the training data set:

- obtain errors for each calibration instrument for each day,
- take logarithms of positive parameters, and rescale parameters, yield curves, and errors to have zero mean and variance 1,
- apply a principal component analysis to the yield curve,
- generate random normally distributed vectors consistent with given covariance,
- apply inverse transformations, i.e. rescale to original mean, variance and exponentiate,
- apply random errors to results.

# Training and testing the network

- To have out of sample data only historical data up to June 2015 is used to generate the training data set.
- Goal: learn the function
  $h : \mathbb{R}^{200} \to \mathbb{R}^2$; $(\underbrace{\text{swaption volatilities}}_{\in\mathbb{R}^{156}}, \underbrace{\text{yield curve}}_{\in\mathbb{R}^{44}}) \mapsto (\alpha, \sigma)$.
- With a sample set of 150000 training data points a feed-forward neural network with 4 hidden layers all of dimension 64 is trained.

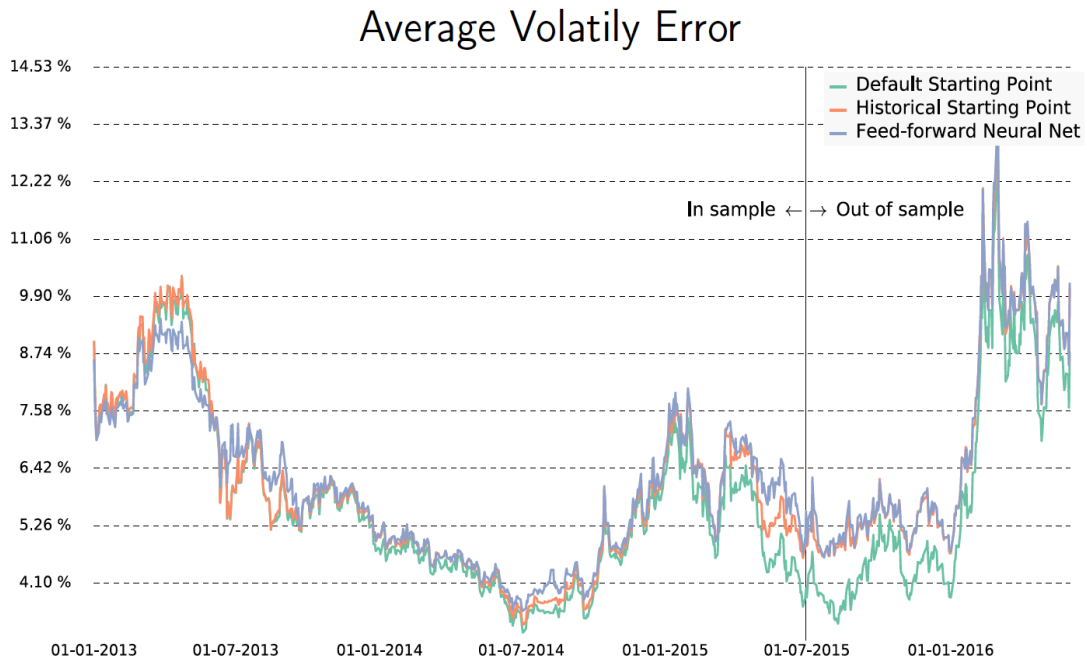



- Out of sample testing is done with the historical data itself.

# Results

The following graph illustrates the result. Average volatility error here means

$$\frac{1}{156}\sum_{j=1}^{156}|\sigma_{\text{impl}}^{\text{mkt}}(\tau_j)-\sigma_{\text{impl}}^{\text{mod}}(\tau_j)|.$$



# Drawbacks of 'classical' machine learning

- In dynamical or in very high dimensional situations the *static* theory of universal approximation appears rigid.
- Take for instance time series analysis, optimization problems, filtering problems: there inputs have changing dimension, ...
- Learning can be computationally very expensive, often shortcuts through random or generic choices are used and work surprisingly well.

# Drawbacks of 'classical' machine learning

- In dynamical or in very high dimensional situations the *static* theory of universal approximation appears rigid.
- Take for instance time series analysis, optimization problems, filtering problems: there inputs have changing dimension, ...
- Learning can be computationally very expensive, often shortcuts through random or generic choices are used and work surprisingly well.

# Drawbacks of 'classical' machine learning

- In dynamical or in very high dimensional situations the *static* theory of universal approximation appears rigid.
- Take for instance time series analysis, optimization problems, filtering problems: there inputs have changing dimension, ...
- Learning can be computationally very expensive, often shortcuts through random or generic choices are used and work surprisingly well.

# Reservoir Computing

... We aim to learn an input-output map on a high- or infinite dimensional input state space.

## Paradigm

Split the input-output map into a generic part (the *reservoir*), which is *not* trained and a readout part, which is trained.

# A motivation from stochastic analysis

Consider a stochastic differential equation or any controlled system

$$dX_t = \sum_{i=1}^{d} V_i(X_t) \circ dB_t^i, \; X_0 = x \in \mathbb{R}^N$$

for some smooth vector fields $V_i : \mathbb{R}^N \to \mathbb{R}^N$, $i = 1, \ldots, d$ and $d$ independent Brownian motions $B^i$. This describes a stochastic dynamics on $\mathbb{R}^N$.

We want to learn the map

(input noise $B$) $\mapsto$ (solution $X_T$).

Obviously a complicated, non-linear map, ...

# Iterated integrals, signatures, enhanced models

Stochastic Taylor expansion (for rough input signals rough path theory, for more complicated noises the theory of regularity structures) gives a satisfying answer to this important question.

We can split the map $(B_t)_{0\le t\le T} \mapsto X_T$ into two parts:

- A linear systems in the free algebra with $d$ generators (the product is denoted by $\otimes$).

$$dY_t = \sum_{i=1}^{d} Y_t \otimes e_i \circ dB_t^i, \; Y_0 = 1$$

  This is an infinite dimensional system (called, e.g., the signature process) whose solution is just the collection of all iterated Ito-Stratonovich integrals and which does *not* depend on the specific dynamics (the reservoir).

- A linear map $Y \mapsto WY$ which is trained (linear regression!) which is chosen to explain $X_T$ as good as possible (the readout).

# Iterated integrals, signatures, enhanced models

Stochastic Taylor expansion (for rough input signals rough path theory, for more complicated noises the theory of regularity structures) gives a satisfying answer to this important question.

We can split the map $(B_t)_{0\le t\le T} \mapsto X_T$ into two parts:

- A linear systems in the free algebra with $d$ generators (the product is denoted by $\otimes$).

$$dY_t = \sum_{i=1}^{d} Y_t \otimes e_i \circ dB_t^i, \; Y_0 = 1$$

  This is an infinite dimensional system (called, e.g., the signature process) whose solution is just the collection of all iterated Ito-Stratonovich integrals and which does *not* depend on the specific dynamics (the reservoir).

- A linear map $Y \mapsto WY$ which is trained (linear regression!) which is chosen to explain $X_T$ as good as possible (the readout).

# A random localized signature

Instead of the previously (fixed) infinite dimensional system there might be better choices to construct a reservoir: fix an activation function $\sigma$.

## A random localized signature

- there is a set of hyper-parameters $\theta \in \Theta$, and a dimension $M$.
- depending on $\theta$ choose randomly matrices $A_1, \ldots, A_d$ on $\mathbb{R}^M$ as well as shifts $\beta_1, \ldots, \beta_d$ such that maximal non-integrability holds on a starting point $x \in \mathbb{R}^M$.
- one can tune the hyper-parameters $\theta \in \Theta$ and dimension $M$ such that

$$dZ_t = \sum_{i=1}^{d} \sigma(A_i Z_t + \beta_i) \circ dB_t^i \,,\; Z_0 = z$$

locally (in time) approximates $X_t$ via a linear readout up to arbitrary precision.

# Applications of Reservoir computing

- Often reservoirs can be realized physically, whence ultrafast evaluations are possible. Only the readout map $W$ has to be trained.
- One can learn dynamic phenomena *without* knowing the specific vector fields $V_i$ just from its results, even only along *one trajectory*.
- It works unreasonably well with generalization tasks, in particular state space constraints are often unreasonably well respected.

# Applications of Reservoir computing

- Often reservoirs can be realized physically, whence ultrafast evaluations are possible. Only the readout map $W$ has to be trained.
- One can learn dynamic phenomena *without* knowing the specific vector fields $V_i$ just from its results, even only along *one trajectory*.
- It works unreasonably well with generalization tasks, in particular state space constraints are often unreasonably well respected.

# Applications of Reservoir computing

- Often reservoirs can be realized physically, whence ultrafast evaluations are possible. Only the readout map $W$ has to be trained.
- One can learn dynamic phenomena *without* knowing the specific vector fields $V_i$ just from its results, even only along *one trajectory*.
- It works unreasonably well with generalization tasks, in particular state space constraints are often unreasonably well respected.

# A view towards regularity structures

Rough paths and signatures can be seen from the point of view of regularity structures. One can formulate the problem of reservoir computing also from this perspective.

- Understand the structure of local expansions.
- generate them by dynamical systems.

# A view towards regularity structures

Rough paths and signatures can be seen from the point of view of regularity structures. One can formulate the problem of reservoir computing also from this perspective.

- Understand the structure of local expansions.
- generate them by dynamical systems.

# A view towards regularity structures

Rough paths and signatures can be seen from the point of view of regularity structures. One can formulate the problem of reservoir computing also from this perspective.

- Understand the structure of local expansions.
- generate them by dynamical systems.

# Market scenario generation

- The market provides us with one trajectory of $N$ prices $X$ of underlyings, derivatives, etc over some period of time $0 \dots T$.
- Construct a normalized noisy signal of dimension $d$, for instance take the most liquidly traded underlyings $S^1, \dots, S^K$ and estimate their instantanous covariance $\Sigma$ along the trajectory, then normalize infinitesimally by its root, i.e. define

$$B_t := \int_0^t \sqrt{\Sigma_s^{-1}} dS_s \,.$$

  By statistical tests one should check whether $B$ has the properties of a Brownian path.
- Choose a reservoir (by tuning some hyperparameters) and learn the readout map $W$ such that along $B$ the output path $X$ is optimally explained.
- Generalize into the future by prolonging $B$ as a Brownian motion.

# Market scenario generation

- The market provides us with one trajectory of $N$ prices $X$ of underlyings, derivatives, etc over some period of time $0 \dots T$.
- Construct a normalized noisy signal of dimension $d$, for instance take the most liquidly traded underlyings $S^1, \dots, S^K$ and estimate their instantanous covariance $\Sigma$ along the trajectory, then normalize infinitesimally by its root, i.e. define

$$B_t := \int_0^t \sqrt{\Sigma_s^{-1}} dS_s \,.$$

  By statistical tests one should check whether $B$ has the properties of a Brownian path.
- Choose a reservoir (by tuning some hyperparameters) and learn the readout map $W$ such that along $B$ the output path $X$ is optimally explained.
- Generalize into the future by prolonging $B$ as a Brownian motion.

# Market scenario generation

- The market provides us with one trajectory of $N$ prices $X$ of underlyings, derivatives, etc over some period of time $0 \dots T$.
- Construct a normalized noisy signal of dimension $d$, for instance take the most liquidly traded underlyings $S^1, \dots, S^K$ and estimate their instantanous covariance $\Sigma$ along the trajectory, then normalize infinitesimally by its root, i.e. define

$$B_t := \int_0^t \sqrt{\Sigma_s^{-1}} dS_s \,.$$

  By statistical tests one should check whether $B$ has the properties of a Brownian path.
- Choose a reservoir (by tuning some hyperparameters) and learn the readout map $W$ such that along $B$ the output path $X$ is optimally explained.
- Generalize into the future by prolonging $B$ as a Brownian motion.

# Market scenario generation

- The market provides us with one trajectory of $N$ prices $X$ of underlyings, derivatives, etc over some period of time $0 \dots T$.
- Construct a normalized noisy signal of dimension $d$, for instance take the most liquidly traded underlyings $S^1, \dots, S^K$ and estimate their instantanous covariance $\Sigma$ along the trajectory, then normalize infinitesimally by its root, i.e. define

$$B_t := \int_0^t \sqrt{\Sigma_s^{-1}} dS_s \,.$$

  By statistical tests one should check whether $B$ has the properties of a Brownian path.
- Choose a reservoir (by tuning some hyperparameters) and learn the readout map $W$ such that along $B$ the output path $X$ is optimally explained.
- Generalize into the future by prolonging $B$ as a Brownian motion.

# Remarks

- Notice that at no point of this procedure a parametric model (like, e.g., GARCH or Heston) was specified.
- Notice that we are actually dealing with a high dimensional situation with several constraints.
- Notice that the approach is completely data driven, however, we can always – through the choice of the reservoir – include additional features like extreme events into the picture.

# Remarks

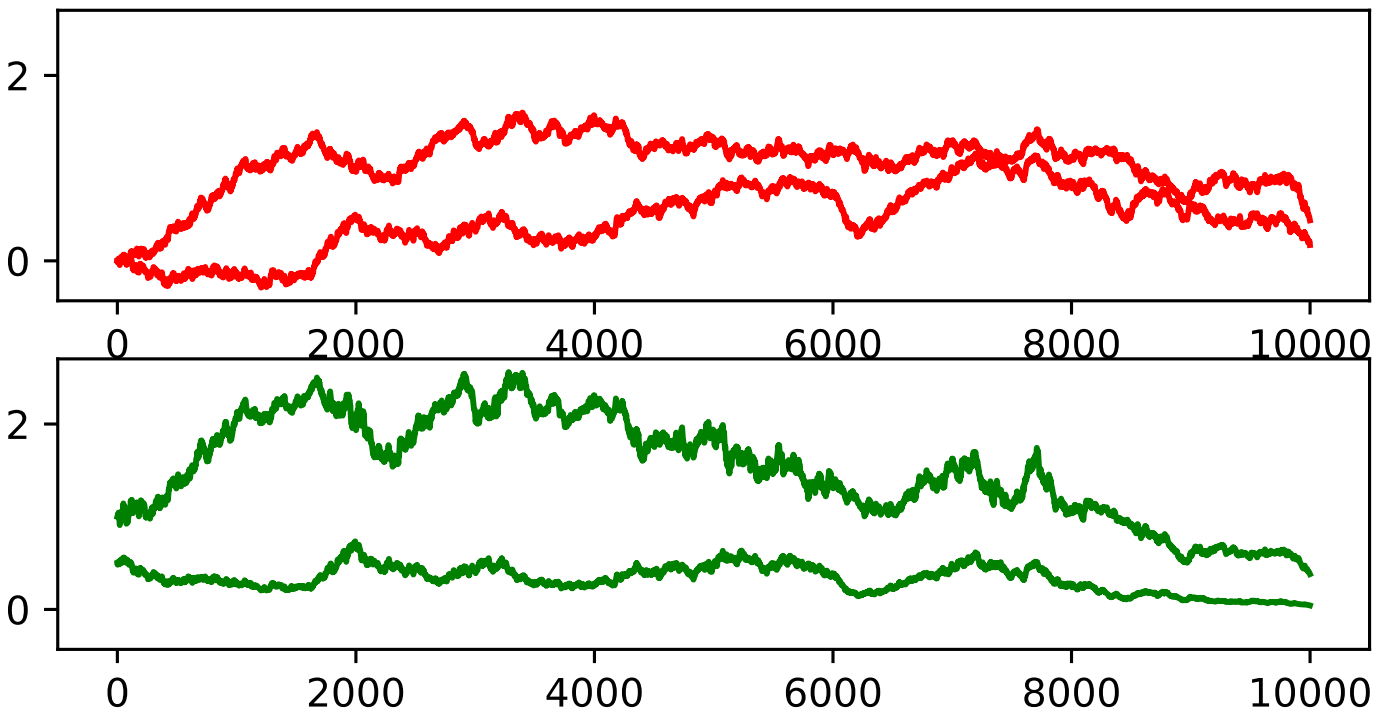
- Notice that at no point of this procedure a parametric model (like, e.g., GARCH or Heston) was specified.
- Notice that we are actually dealing with a high dimensional situation with several constraints.
- Notice that the approach is completely data driven, however, we can always – through the choice of the reservoir – include additional features like extreme events into the picture.

# Remarks

- Notice that at no point of this procedure a parametric model (like, e.g., GARCH or Heston) was specified.
- Notice that we are actually dealing with a high dimensional situation with several constraints.
- Notice that the approach is completely data driven, however, we can always – through the choice of the reservoir – include additional features like extreme events into the picture.

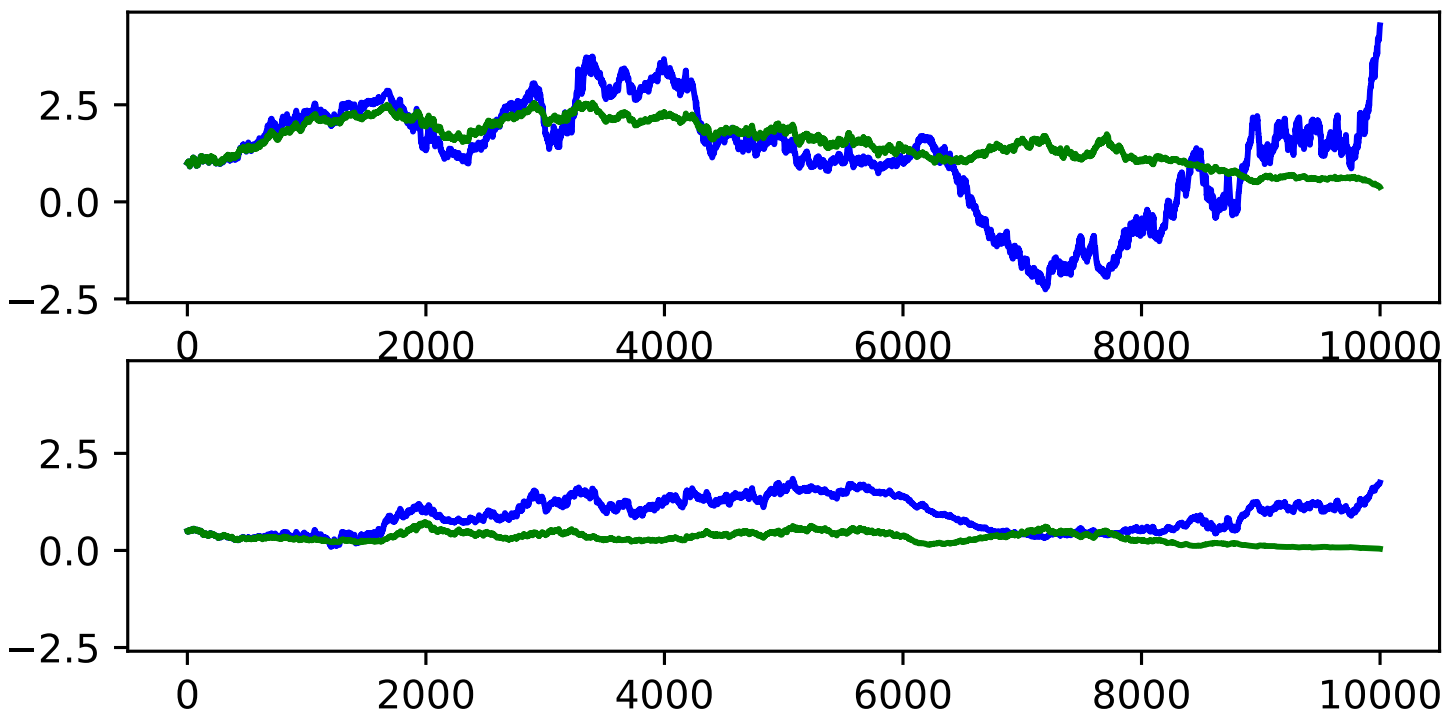
# Learning a SABR market from one trajectory

We generate for one 2d Brownian trajectory (red) a SABR market trajectory (green), i.e. a price process together with a stochastic variance process for 10000 ticks.



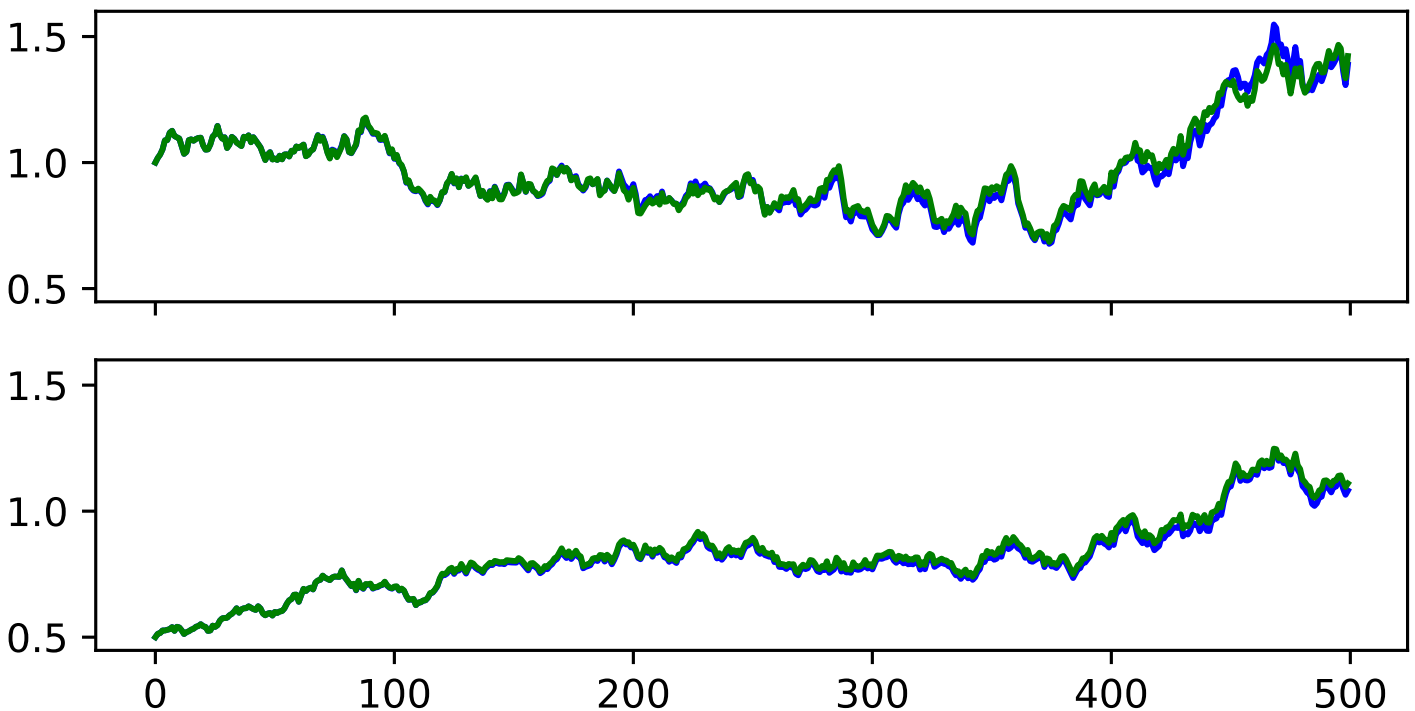
# Learning a SABR market from one trajectory

We generate a 50 dimensional reservoir trajectory along the above Brownian path with random vector fields and perform a regression for 1000 ticks. The result is shown in blue. Already the visible generalization beyong 1000 ticks is amazing.



# Generalization

We generalize for another Brownian trajectory and check whether the regression (blue) on the reservoir trajectory represents the SABR trajectory (green) accurately along 500 ticks.



# Outlook

- test market generators in different environments.
- explore different explanations for the observed phenomena.
- explore the phenomena within the theory of regularity structures.

# Outlook

- test market generators in different environments.
- explore different explanations for the observed phenomena.
- explore the phenomena within the theory of regularity structures.

# Outlook

- test market generators in different environments.
- explore different explanations for the observed phenomena.
- explore the phenomena within the theory of regularity structures.

## References


- H. Bühler, L. Gonon, J. Teichmann, and B. Wood: *Deep Hedging*, Arxiv, 2018.
- M. Hairer: *Theory of regularity structures, Inventiones mathematicae*, 2014.
- T. Lyons, *Rough paths, Signatures and the modelling of functions on streams Terry Lyons*, Arxiv, 2014.

Successful science relies on a broad education, critical thinking, curiosity and *freedom of thought and speech*.
Universities and the scientific communities all over the world stand for those qualities and should defend them.

Governments should *not* interfere in this process!

It is therefore completely unacceptable that education and research in gender studies is forbidden by governmental decrees.

Every scientist should stand up against such interventions.