Machine Learning in mathematical Finance

Josef Teichmann

ETH Zürich

December 15, 2017

Mathematical Challenges in mathematical Finance

- High dimensional stochastic control problems often of a non-standard type (hedging in markets with transaction costs or liquidity constraints).
- High-dimensional inverse problems, where models (PDEs, stochastic processes) have to be selected to explain a given set of market prices optimally.
- High-dimensional prediction tasks (long term investments, portfolio selection).
- High-dimensional feature selection tasks (limit order books).

Mathematical Challenges in mathematical Finance

- High dimensional stochastic control problems often of a non-standard type (hedging in markets with transaction costs or liquidity constraints).
- High-dimensional inverse problems, where models (PDEs, stochastic processes) have to be selected to explain a given set of market prices optimally.
- High-dimensional prediction tasks (long term investments, portfolio selection).
- High-dimensional feature selection tasks (limit order books).

Mathematical Challenges in mathematical Finance

- High dimensional stochastic control problems often of a non-standard type (hedging in markets with transaction costs or liquidity constraints).
- High-dimensional inverse problems, where models (PDEs, stochastic processes) have to be selected to explain a given set of market prices optimally.
- High-dimensional prediction tasks (long term investments, portfolio selection).
- High-dimensional feature selection tasks (limit order books).

Mathematical Challenges in mathematical Finance

- High dimensional stochastic control problems often of a non-standard type (hedging in markets with transaction costs or liquidity constraints).
- High-dimensional inverse problems, where models (PDEs, stochastic processes) have to be selected to explain a given set of market prices optimally.
- High-dimensional prediction tasks (long term investments, portfolio selection).
- High-dimensional feature selection tasks (limit order books).

Neural Networks

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

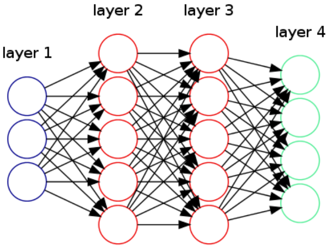


Figure: A 2 hidden layers neural network with 3 input and 4 output dimensions

just encodes a certain concatenation of affine and non-linear functions by composition in a well specified order.

Universal Approximation

- Neural networks appear in the 1943 seminal work by Warren McCulloch and Walter Pitts inspired by certain functionalities of the human brain aiming for articial 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 *approximate* any continuous function on the unit cube.
- Connections between *deep* neural networks and sparse representations in certain wavelet basis (Helmut Bölcskei, Philipp Grohs et al.) explaining their incredible representation power.

Universal Approximation

- Neural networks appear in the 1943 seminal work by Warren McCulloch and Walter Pitts inspired by certain functionalities of the human brain aiming for articial 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 *approximate* any continuous function on the unit cube.
- Connections between *deep* neural networks and sparse representations in certain wavelet basis (Helmut Bölcskei, Philipp Grohs et al.) explaining their incredible representation power.

Universal Approximation

- Neural networks appear in the 1943 seminal work by Warren McCulloch and Walter Pitts inspired by certain functionalities of the human brain aiming for articial 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 *approximate* any continuous function on the unit cube.
- Connections between *deep* neural networks and sparse representations in certain wavelet basis (Helmut Bölcskei, Philipp Grohs et al.) explaining their incredible representation power.

Universal Approximation

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

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

- Universal Approximation Theorems (George Cybenko, Kurt Hornik, et al.) show that *one hidden layer networks* can *approximate* any continuous function on the unit cube.
- Connections between *deep* neural networks and sparse representations in certain wavelet basis (Helmut Bölcskei, Philipp Grohs et al.) explaining their incredible representation power.

An example: reservoir computing paradigm

- in many situations input is a time series object of varying length.
- a part of the neural network, which represents the input-output map, is chosen as a *generic dynamical system* (often with physical realization and, of course, with relationship to the input-output map). The goal of this choice is to transform the input into relevant information pieces.
- only the last layer is trained, i.e. a linear regression on the generic network's output is performed.
- this reminds of stochastic differential equations which can be written – in a quite regular way – as linear maps on the input signal's signature, i.e. the collection of all iterated integrals (universal limit theorem of rough path theory).

An example: reservoir computing paradigm

- in many situations input is a time series object of varying length.
- a part of the neural network, which represents the input-output map, is chosen as a *generic dynamical system* (often with physical realization and, of course, with relationship to the input-output map). The goal of this choice is to transform the input into relevant information pieces.
- only the last layer is trained, i.e. a linear regression on the generic network's output is performed.
- this reminds of stochastic differential equations which can be written – in a quite regular way – as linear maps on the input signal's signature, i.e. the collection of all iterated integrals (universal limit theorem of rough path theory).

An example: reservoir computing paradigm

- in many situations input is a time series object of varying length.
- a part of the neural network, which represents the input-output map, is chosen as a *generic dynamical system* (often with physical realization and, of course, with relationship to the input-output map). The goal of this choice is to transform the input into relevant information pieces.
- only the last layer is trained, i.e. a linear regression on the generic network's output is performed.
- this reminds of stochastic differential equations which can be written – in a quite regular way – as linear maps on the input signal's signature, i.e. the collection of all iterated integrals (universal limit theorem of rough path theory).

An example: reservoir computing paradigm

- in many situations input is a time series object of varying length.
- a part of the neural network, which represents the input-output map, is chosen as a *generic dynamical system* (often with physical realization and, of course, with relationship to the input-output map). The goal of this choice is to transform the input into relevant information pieces.
- only the last layer is trained, i.e. a linear regression on the generic network's output is performed.
- this reminds of stochastic differential equations which can be written – in a quite regular way – as linear maps on the input signal's signature, i.e. the collection of all iterated integrals (universal limit theorem of rough path theory).

Deep Networks in Finance

Recent ideas to use machine learning in Finance

- Deep pricing: use neural networks to constitute efficient regression bases in, e.g., the Longstaff Schwartz algorithm for pricing call-able products like American options.
- Deep hedging: use neural networks to approximate hedging strategies in, e.g., hedging problems in the presence of market frictions (joint work with Hans Bühler, Lukas Gonon, and Ben Wood).
- Deep filtering: use neural networks on top of well selected dynamical systems to approximate laws of signals conditional on "noisy" observation.
- Deep calibration: use machine learning to approximate the solution of inverse problems (model selection) in Finance (joint work with Christa Cuchiero).

Deep Networks in Finance

Recent ideas to use machine learning in Finance

- Deep pricing: use neural networks to constitute efficient regression bases in, e.g., the Longstaff Schwartz algorithm for pricing call-able products like American options.
- Deep hedging: use neural networks to approximate hedging strategies in, e.g., hedging problems in the presence of market frictions (joint work with Hans Bühler, Lukas Gonon, and Ben Wood).
- Deep filtering: use neural networks on top of well selected dynamical systems to approximate laws of signals conditional on "noisy" observation.
- Deep calibration: use machine learning to approximate the solution of inverse problems (model selection) in Finance (joint work with Christa Cuchiero).

Deep Networks in Finance

Recent ideas to use machine learning in Finance

- Deep pricing: use neural networks to constitute efficient regression bases in, e.g., the Longstaff Schwartz algorithm for pricing call-able products like American options.
- Deep hedging: use neural networks to approximate hedging strategies in, e.g., hedging problems in the presence of market frictions (joint work with Hans Bühler, Lukas Gonon, and Ben Wood).
- Deep filtering: use neural networks on top of well selected dynamical systems to approximate laws of signals conditional on "noisy" observation.
- Deep calibration: use machine learning to approximate the solution of inverse problems (model selection) in Finance (joint work with Christa Cuchiero).

Deep Networks in Finance

Recent ideas to use machine learning in Finance

- Deep pricing: use neural networks to constitute efficient regression bases in, e.g., the Longstaff Schwartz algorithm for pricing call-able products like American options.
- Deep hedging: use neural networks to approximate hedging strategies in, e.g., hedging problems in the presence of market frictions (joint work with Hans Bühler, Lukas Gonon, and Ben Wood).
- Deep filtering: use neural networks on top of well selected dynamical systems to approximate laws of signals conditional on "noisy" observation.
- Deep calibration: use machine learning to approximate the solution of inverse problems (model selection) in Finance (joint work with Christa Cuchiero).

Calibration by machine learning

- Terry Lyons (Oberwolfach 2017) on problems of calibrating rough volatility models: “Why don’t you learn it?”
- If calibration is technologically a bottleneck why not using machine learning for it to easen time constraints.

Calibration by machine learning

- Terry Lyons (Oberwolfach 2017) on problems of calibrating rough volatility models: "Why don't you learn it?"
- If calibration is technologically a bottleneck why not using machine learning for it to easen time constraints.

Calibration by Machine learning following Andres Hernandez

We shall provide a brief overview of a procedure introduced by Andres Hernandez (2016) as seen from the point of view of Team 3's team challenge project 2017 at UCT:

Algorithm suggested by A. Hernandez

- Getting the historical price data.
- Calibrating the model, *a single factor Hull-White extended Vasiček model* to obtain a time series of (typical) model parameters, here the yield curve, the rate of mean reversion α, and the short rate's volatility σ.
- Pre-process data and generate new combinations of parameters.
- With a new large training data set of (prices,parameters) a neural network is trained.
- The neural network is tested on out-of-sample data.

The data set

- The collected historical data are ATM volatility quotes for GBP from January 2nd, 2013 to June 1st, 2016. 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.
- The yield curve is given 44 points, i.e. it is discretely sampled on 0, 1, 2, 7, 14 days; 1 to 24 months; 3-10 years; plus 12, 15, 20, 25, 30, 40 and 50 years. Interpolation is done by Cubic splines if necessary.

Classical calibration a la QL

Historical parameters

- a Levenberg-Marquardt local optimizer is first applied to minimize the equally-weighted average of squared yield or IV differences.
- calibration is done twice, with different starting points:
  - at first, $\alpha = 0.1$ and $\sigma = 0.01$ are the default choice
  - second the calibrated parameters from the previous day (using the default starting point) are used for the second stage of classical calibration.

Calibration results along time series

The *re-calibration problem* gets visible ... and it is indeed a feasible procedure.

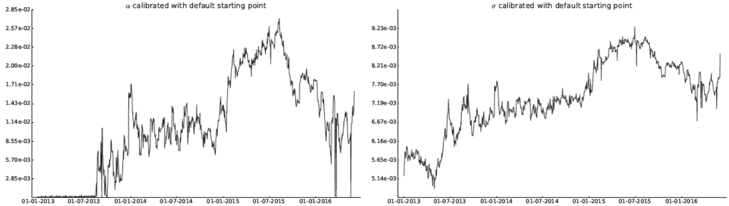


Figure: Calibration using default starting point

How do neural networks enter calibration?

Universal approximation of calibration functionals

- Neural networks are often used to approximate functions due to the universal approximation property.
- We approximate the calibration functional (yields,prices) $\mapsto$ (parameters) which maps (yields, prices) to optimal model parameters by a neural network.

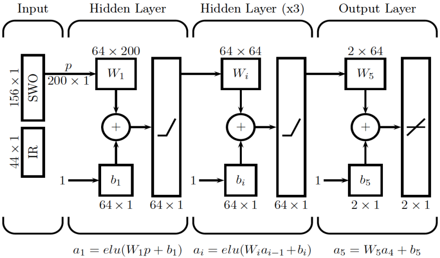
Neural Networks : Training Set Generation

With the calibration history A. Hernandez proceeds by generating the training set

- obtain errors for each calibration instrument for each day,
- take logarithms of of positive parameters, and rescale parameters, yield curves, and errors to have zero mean and variance 1,
- apply a principal component analysis and an appropriate amount of the first modes,
- 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.

Neural Networks: Training the network

- With a sample set of 150 thousand training data points, A. Hernandez suggests to train a feed-forward neural network.
  - The architecture is chosen feed-forward with 4 hidden layers, each layer with 64 neurons using an ELU (Exponential Linear Unit)



Neural Networks: testing the trained network

- two neural networks were trained using a sample set produced where the covariance matrix was estimated based on 40% of historical data.
- the second sample set used 73% of historical data.
- for training, the sample set was split into 80% training set and 20% cross-validation.
- the testing was done with the historical data itself (i.e. a backtesting procedure was used to check the accuracy of the data).

Results of A. Hernandez

The following graphs illustrate the results. Average volatility error here just means

$$\frac{\sum_{n=1}^{156} \left| impvol^{mkt} - impvol^{model} \right|}{156} \quad (1)$$

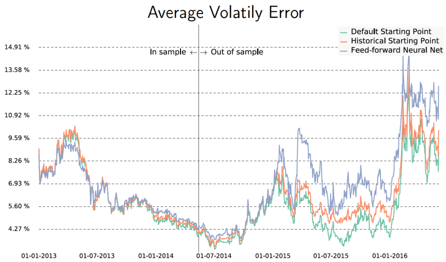


Figure: Correlation up to June 2014

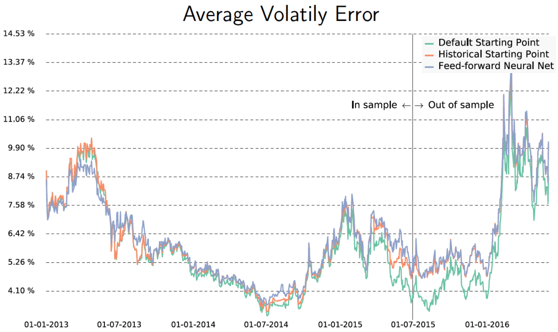


Figure: Correlation up to June 2015

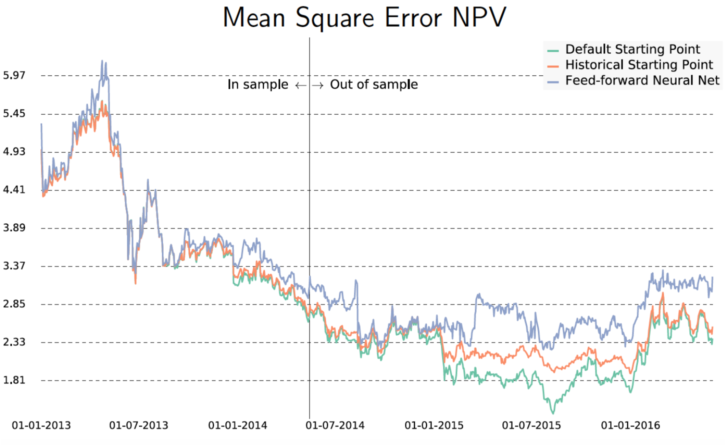


Figure: Correlation up to June 2014

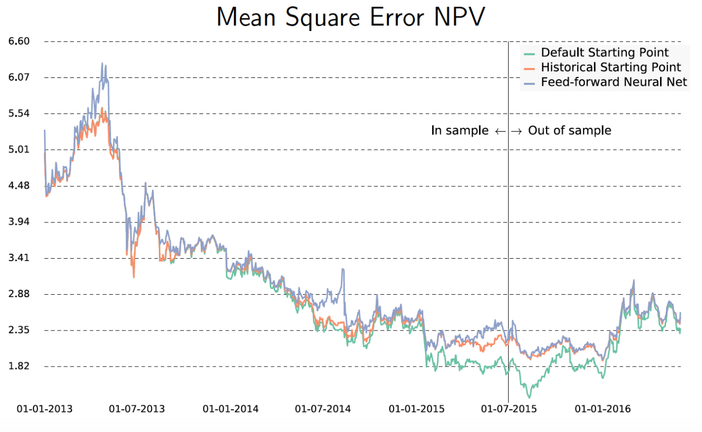


Figure: Correlation up to June 2015

Towards a Bayesian model

Consider the Hull-White extended Vasiček models (on a space $(\Omega, \mathcal{F}, (\mathcal{G}_t)_{t\geq 0}, \mathbb{P})$):

$$dr_t^{(1)} = (\beta_1(t) - \alpha_1 r_t^{(1)})\, dt + \sigma_1\, dW_t\,,$$

$$dr_t^{(2)} = (\beta_2(t) - \alpha_2 r_t^{(2)})\, dt + \sigma_2\, dW_t\,.$$

We assume that r is is a mixture of these two models with constant probability $\pi \in [0, 1]$, i.e.

$$\mathbb{P}(r_t \leq x) = \pi \mathbb{P}\left(r_t^{(1)} \leq x\right) + (1-\pi)\mathbb{P}\left(r_t^{(2)} \leq x\right).$$

Of course the observation filtration generated by daily ATM swaption prices and a daily yield curve is smaller than the filtration $\mathbb{G}$, hence the theory of the first part applies.

Bayesian model: setup

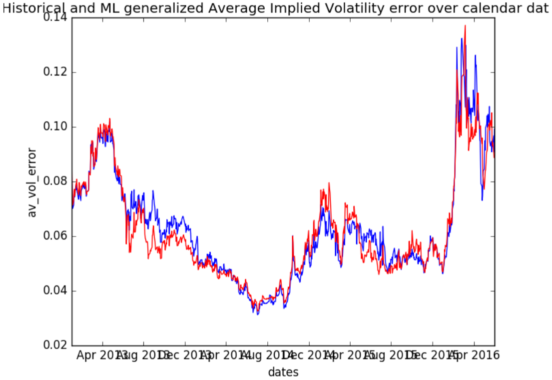
We still have the same set-up (in terms of data):

- $N = 156 + 44 = 200$ input prices (swaptions + yield curve)
- $n = 44 + 4 + 1 = 49$ parameters to estimate. These are $\alpha_1, \alpha_2, \sigma_1, \sigma_2, \pi$ and $\text{yield}_1(t)$ (or, equivalently, $\text{yield}_2(t)$) at 44 maturities (notice that given $\alpha_1, \alpha_2, \sigma_1, \sigma_2, \pi$ there is a one-to-one map between yields and βs).
- Hence, the calibration function is now

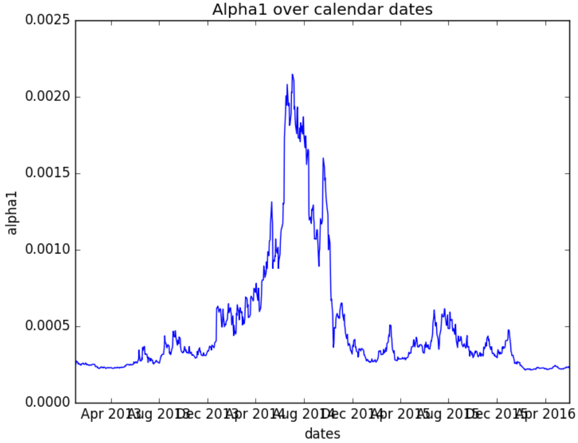
$$\Theta : \mathbb{R}^{200} \longrightarrow \mathbb{R}^{49}, \qquad \begin{pmatrix} \text{SWO1} \\ \text{SWO2} \\ \cdots \\ \text{yield}(0) \\ \text{yield}(1) \\ \cdots \end{pmatrix} \mapsto \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \sigma_1 \\ \sigma_2 \\ \pi \\ \text{yield}_1(0) \\ \text{yield}_1(1) \\ \cdots \end{pmatrix}$$

Bayesian model: training

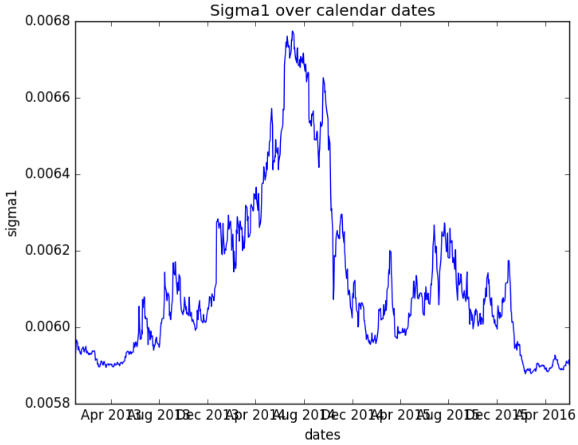
We generated a new training set and trained, tested another neural network with a similar architecture: the quality of the new calibration is the same as the QuantLib calibration and better than previous ML results, in particular out of sample.



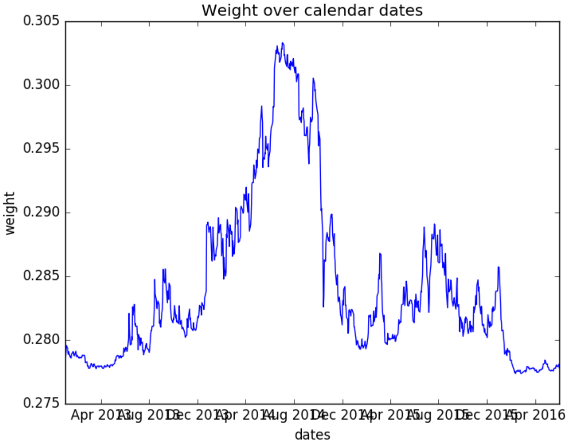
Mixture Model: α_1



Mixture Model: σ_1



Mixture Model: π



- it works to train networks the information of calibration functionals: usually calibration functionals are of a hash function type, i.e. it is easy to calculate prices from given parameters, but it is difficult to re-construct parameters from given prices. Still it is easy to generate training data.
- the "unreasonable effectiveness" is visible by absence of the 'curse of dimension'.
- it will be interesting to train *universal calibrators* of realistic models by offline algorithms which allow to circumvent high-dimensional delicate calibration procedures.

- it works to train networks the information of calibration functionals: usually calibration functionals are of a hash function type, i.e. it is easy to calculate prices from given parameters, but it is difficult to re-construct parameters from given prices. Still it is easy to generate training data.
- the “unreasonable effectiveness” is visible by absence of the 'curse of dimension'.
- it will be interesting to train *universal calibrators* of realistic models by offline algorithms which allow to circumvent high-dimensional delicate calibration procedures.

- it works to train networks the information of calibration functionals: usually calibration functionals are of a hash function type, i.e. it is easy to calculate prices from given parameters, but it is difficult to re-construct parameters from given prices. Still it is easy to generate training data.
- the “unreasonable effectiveness” is visible by absence of the ’curse of dimension’.
- it will be interesting to train *universal calibrators* of realistic models by offline algorithms which allow to circumvent high-dimensional delicate calibration procedures.

Frame of ideas

- Many problems in Finance are of filtering nature, i.e. calculating conditional laws of a true signal X_{t+h}, at some point in time $t+h$, given some noisy observation $(Y_s)_{0 \le s \le t}$.
- Such problems often depend in a complicated, non-robust way on the trajectory of Y, i.e. no Lipschitz dependence on Y: regularizations are suggested by, e.g., the theory of regularity structures, and its predecessor, rough path theory. By lifting input trajectories Y to more complicated objects (later called *models*) one can increase robustness to a satisfactory level.
- The idea is to write an abstract theory of expansions as developed by Martin Hairer in a series of papers, understand it as an "expressive" dynamical system and learn the output layer (which is of high regularity).

Frame of ideas

- Many problems in Finance are of filtering nature, i.e. calculating conditional laws of a true signal X_{t+h}, at some point in time $t+h$, given some noisy observation $(Y_s)_{0 \leq s \leq t}$.
- Such problems often depend in a complicated, non-robust way on the trajectory of Y, i.e. no Lipschitz dependence on Y: regularizations are suggested by, e.g., the theory of regularity structures, and its predecessor, rough path theory. By lifting input trajectories Y to more complicated objects (later called *models*) one can increase robustness to a satisfactory level.
- The idea is to write an abstract theory of expansions as developed by Martin Hairer in a series of papers, understand it as an "expressive" dynamical system and learn the output layer (which is of high regularity).

Frame of ideas

- Many problems in Finance are of filtering nature, i.e. calculating conditional laws of a true signal X_{t+h}, at some point in time $t+h$, given some noisy observation $(Y_s)_{0\leq s\leq t}$.
- Such problems often depend in a complicated, non-robust way on the trajectory of Y, i.e. no Lipschitz dependence on Y: regularizations are suggested by, e.g., the theory of regularity structures, and its predecessor, rough path theory. By lifting input trajectories Y to more complicated objects (later called *models*) one can increase robustness to a satisfactory level.
- The idea is to write an abstract theory of expansions as developed by Martin Hairer in a series of papers, understand it as an “expressive” dynamical system and learn the output layer (which is of high regularity).

- Many solutions of problems in stochastics can be translated to solving fixed point equation on modelled distributions.
- By applying the reconstruction operator the modeled distribution is translated to a real world object, which then depends – by inspecting precisely its continuities – in an at least Lipschitz way on the underlying model, i.e. stochastic inputs.
- The theory of regularity structures tells precisely how 'models' have to be specified such that stochastic inputs actually constitute models: this yields a theory of input specifications.
- Supervised learning: by creating training data (in appropriate input format!) one can learn the input-output map.
- Applications: solutions of stochastic differential equations (Friz, Lyons, Victoir, etc), solutions of correlated filtering problems (Crisan, Friz, etc), solutions of sub-critical stochastic partial differential equations (Hairer, Gubinelli, etc).

- Many solutions of problems in stochastics can be translated to solving fixed point equation on modelled distributions.
- By applying the reconstruction operator the modeled distribution is translated to a real world object, which then depends – by inspecting precisely its continuities – in an at least Lipschitz way on the underlying model, i.e. stochastic inputs.
- The theory of regularity structures tells precisely how 'models' have to be specified such that stochastic inputs actually constitute models: this yields a theory of input specifications.
- Supervised learning: by creating training data (in appropriate input format!) one can learn the input-output map.
- Applications: solutions of stochastic differential equations (Friz, Lyons, Victoir, etc), solutions of correlated filtering problems (Crisan, Friz, etc), solutions of sub-critical stochastic partial differential equations (Hairer, Gubinelli, etc).

- Many solutions of problems in stochastics can be translated to solving fixed point equation on modelled distributions.
- By applying the reconstruction operator the modeled distribution is translated to a real world object, which then depends – by inspecting precisely its continuities – in an at least Lipschitz way on the underlying model, i.e. stochastic inputs.
- The theory of regularity structures tells precisely how 'models' have to be specified such that stochastic inputs actually constitute models: this yields a theory of input specifications.
- Supervised learning: by creating training data (in appropriate input format!) one can learn the input-output map.
- Applications: solutions of stochastic differential equations (Friz, Lyons, Victoir, etc), solutions of correlated filtering problems (Crisan, Friz, etc), solutions of sub-critical stochastic partial differential equations (Hairer, Gubinelli, etc).

- Many solutions of problems in stochastics can be translated to solving fixed point equation on modelled distributions.
- By applying the reconstruction operator the modeled distribution is translated to a real world object, which then depends – by inspecting precisely its continuities – in an at least Lipschitz way on the underlying model, i.e. stochastic inputs.
- The theory of regularity structures tells precisely how 'models' have to be specified such that stochastic inputs actually constitute models: this yields a theory of input specifications.
- Supervised learning: by creating training data (in appropriate input format!) one can learn the input-output map.
- Applications: solutions of stochastic differential equations (Friz, Lyons, Victoir, etc), solutions of correlated filtering problems (Crisan, Friz, etc), solutions of sub-critical stochastic partial differential equations (Hairer, Gubinelli, etc).

- Many solutions of problems in stochastics can be translated to solving fixed point equation on modelled distributions.
- By applying the reconstruction operator the modeled distribution is translated to a real world object, which then depends – by inspecting precisely its continuities – in an at least Lipschitz way on the underlying model, i.e. stochastic inputs.
- The theory of regularity structures tells precisely how 'models' have to be specified such that stochastic inputs actually constitute models: this yields a theory of input specifications.
- Supervised learning: by creating training data (in appropriate input format!) one can learn the input-output map.
- Applications: solutions of stochastic differential equations (Friz, Lyons, Victoir, etc), solutions of correlated filtering problems (Crisan, Friz, etc), solutions of sub-critical stochastic partial differential equations (Hairer, Gubinelli, etc).

Prediction Tasks

- consider certain noisy observations of a true signal and model them by a corresponding regularity structure (this might be necessary in since there is no reason why non-linear functions of noisy objects should be well defined).
- construct solutions of the optimal filter by solving a fixed point equation on modelled distributions.
- reconstruct the real world filter by the reconstruction operator, which yields – under appropriate regularity conditions – a non-linear, Lipschitz map from the space of observations (the 'models') to the optimal filter.
- Learn this map on regularized noises.

Deep Hedging

- given a generic market situation: scenarios generated by one or many different models fitting aspects of the market environment.
- given transaction costs, liquidity constraints, bid ask spreads. etc.
- given a derivative and a risk objective.
- approximate hedging strategies by deep neural networks of all appropriate factors, which creates a dense subset of admissible strategies,
- minimize the given risk objective over all possible deep hedges.

Deep Hedging

- given a generic market situation: scenarios generated by one or many different models fitting aspects of the market environment.
- given transaction costs, liquidity constraints, bid ask spreads. etc.
- given a derivative and a risk objective.
- approximate hedging strategies by deep neural networks of all appropriate factors, which creates a dense subset of admissible strategies,
- minimize the given risk objective over all possible deep hedges.

Deep Hedging

- given a generic market situation: scenarios generated by one or many different models fitting aspects of the market environment.
- given transaction costs, liquidity constraints, bid ask spreads. etc.
- given a derivative and a risk objective.
- approximate hedging strategies by deep neural networks of all appropriate factors, which creates a dense subset of admissible strategies,
- minimize the given risk objective over all possible deep hedges.

Deep Hedging

- given a generic market situation: scenarios generated by one or many different models fitting aspects of the market environment.
- given transaction costs, liquidity constraints, bid ask spreads. etc.
- given a derivative and a risk objective.
- approximate hedging strategies by deep neural networks of all appropriate factors, which creates a dense subset of admissible strategies,
- minimize the given risk objective over all possible deep hedges.

Deep Hedging

- given a generic market situation: scenarios generated by one or many different models fitting aspects of the market environment.
- given transaction costs, liquidity constraints, bid ask spreads. etc.
- given a derivative and a risk objective.
- approximate hedging strategies by deep neural networks of all appropriate factors, which creates a dense subset of admissible strategies,
- minimize the given risk objective over all possible deep hedges.

Advantages

- particular models play a minor role, very data driven.
- tractability, which is a delicate problem, for high dimensional non-linear PIDEs, does not play a role for setting up the problem: even very high dimensional reinforcement learning problems can be solved in a satisfying way.
- market frictions can be easily included.
- idea: set up a describable set of hedging strategies which allow to ϵ approximate the optimal solution.

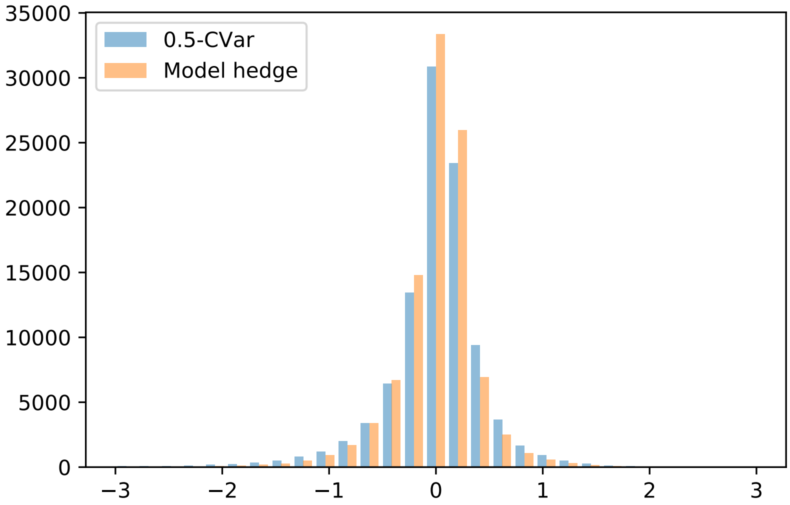


Figure: Comparison of model hedge and deep hedge associated to 50%-expected shortfall criterion.

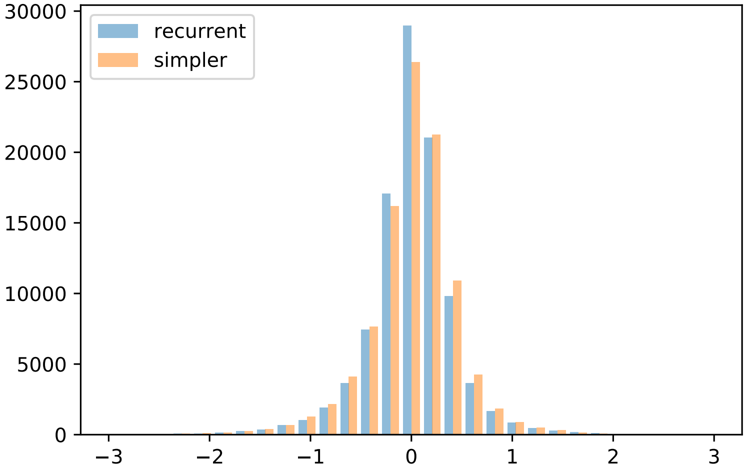
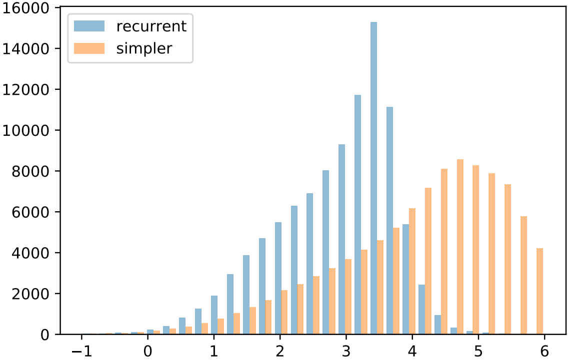


Figure: Comparison of recurrent and simpler network structure (no transaction costs).



	Mean Loss	Price	Realized CVar
recurrent	0.0018	5.5137	-0.0022
simpler	0.0022	6.7446	-0.0

Figure: Network architecture matters: Comparison of recurrent and simpler network structure (with transaction costs and 99%-CVar criterion).

- Machine Learning for calibration works, even where classical calibration would have difficulties. Recent project (jointly with C. Cuchiero, A. Hernandez, and W. Khosrawi-Sardroudi): machine learn calibration of local stochastic volatility models which are widely used but where classical calibration is very delicate.
- Why does it work so well: explain the "unreasonable effectiveness": sparse wavelet representation seem to provide a key.
- How to choose input variables? Universal approximation depends a lot on this: rough paths or regularity structures as a solution concept.
- Reservoir computing: use real-world dynamical systems (i.e. build from financial markets) to provide prototype input-output maps: on top of those "generic" maps specific tasks can solved by regression.

- Machine Learning for calibration works, even where classical calibration would have difficulties. Recent project (jointly with C. Cuchiero, A. Hernandez, and W. Khosrawi-Sardroudi): machine learn calibration of local stochastic volatility models which are widely used but where classical calibration is very delicate.
- Why does it work so well: explain the "unreasonable effectiveness": sparse wavelet representation seem to provide a key.
- How to choose input variables? Universal approximation depends a lot on this: rough paths or regularity structures as a solution concept.
- Reservoir computing: use real-world dynamical systems (i.e. build from financial markets) to provide prototype input-output maps: on top of those "generic" maps specific tasks can solved by regression.

- Machine Learning for calibration works, even where classical calibration would have difficulties. Recent project (jointly with C. Cuchiero, A. Hernandez, and W. Khosrawi-Sardroudi): machine learn calibration of local stochastic volatility models which are widely used but where classical calibration is very delicate.
- Why does it work so well: explain the "unreasonable effectiveness": sparse wavelet representation seem to provide a key.
- How to choose input variables? Universal approximation depends a lot on this: rough paths or regularity structures as a solution concept.
- Reservoir computing: use real-world dynamical systems (i.e. build from financial markets) to provide prototype input-output maps: on top of those "generic" maps specific tasks can solved by regression.

- Machine Learning for calibration works, even where classical calibration would have difficulties. Recent project (jointly with C. Cuchiero, A. Hernandez, and W. Khosrawi-Sardroudi): machine learn calibration of local stochastic volatility models which are widely used but where classical calibration is very delicate.
- Why does it work so well: explain the "unreasonable effectiveness": sparse wavelet representation seem to provide a key.
- How to choose input variables? Universal approximation depends a lot on this: rough paths or regularity structures as a solution concept.
- Reservoir computing: use real-world dynamical systems (i.e. build from financial markets) to provide prototype input-output maps: on top of those "generic" maps specific tasks can solved by regression.

References

- C. Cuchiero, I. Klein, and J. Teichmann:
  A fundamental theorem of asset pricing for continuous time large financial markets in a two filtration setting, Arxiv, 2017.
- A. Hernandez:
  Model Calibration with Neural Networks, SSRN, 2016.
- C. Cuchiero, A. Marr, M. Mavuso, N. Mitoulis, A. Singh, and J. Teichmann:
  Calibration with neural networks, report on the FMTC 2017, working paper, 2017.
- M. Hairer:
  Theory of regularity structures, Inventiones mathematicae, 2014.