Deep Asset Liability Management (joint works with Thomas Krabichler, Thorsten Schmidt)

Josef Teichmann

ETH Zürich

September 27, 2021

1 Introduction

2 Machine learning in Finance: Deep Hedging

3 Machine Learning in Finance: Model free Deep Hedging

Section 1

Introduction

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

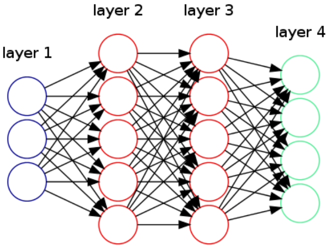


Figure: A 2 hidden layers neural network with 3 input and 4 output dimensions

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

Universal Approximation

- Neural networks appeared 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.

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

- Universal Approximation Theorems (George Cybenko, Kurt Hornik, et al.) show that *one hidden layer networks* can *approximate* any continuous function on the unit cube.
- Many generalizations available, in particular for the purposes of finance: signature transforms, UAT on path spaces, etc.

Universal Approximation

- Neural networks appeared 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.

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

- Universal Approximation Theorems (George Cybenko, Kurt Hornik, et al.) show that *one hidden layer networks* can *approximate* any continuous function on the unit cube.
- Many generalizations available, in particular for the purposes of finance: signature transforms, UAT on path spaces, etc.

Universal Approximation

- Neural networks appeared 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 *approximate* any continuous function on the unit cube.
- Many generalizations available, in particular for the purposes of finance: signature transforms, UAT on path spaces, etc.

Universal Approximation

- Neural networks appeared 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 *approximate* any continuous function on the unit cube.
- Many generalizations available, in particular for the purposes of finance: signature transforms, UAT on path spaces, etc.

Recent progress in UAT

Universal approximation theorems are still an ongoing matter of research, in particular with applications from finance in view.

Recent progress has been made on ...

- ... on path spaces (beyond local compactness)
- ... for recurrent networks.
- ... with signature features of paths.

Also with a view towards provable approximation results.

Training or Learning

- Given a generic loss function (which expresses a task often with respect to data), training is just the construction a dynamical system on the space of network parameters, which decreases loss, or improves the performance with respect to the task.
- Loss functions allow for explicit regularization, but also show implicit regularizations stemming from, e.g., random initializations.
- Spaces of network parameters are usually high dimensional.

One should see these procedures (SGD, ADAM, simulated annealing, etc) from a Bayesian perspective.

Randomness appears prominently in learning

- random initialization of network weights (choice of π_0).
- stochastic gradient descent (use noisy approximations of the loss function).
- dropouts.
- generic (random) architectures with depth often work surprisingly well.
- most radical appearance of randomness: reservoir computing – only train a small portion of trainable parameters!

Takeway message

- Training is a still enigmatic procedure where classical algorithms meet fascinating random effects on high dimensional parameter space.
- Implicit and explicit regularizations appear and shape the properties of the solution decisively.
- Transfer learning: training hidden layers and multi-variate outputs leads to passing information from one output dimension to another one (L^1 implicit regularizations appear). Energy efficiency!

Takeway message

- Training is a still enigmatic procedure where classical algorithms meet fascinating random effects on high dimensional parameter space.
- Implicit and explicit regularizations appear and shape the properties of the solution decisively.
- Transfer learning: training hidden layers and multi-variate outputs leads to passing information from one output dimension to another one (L^1 implicit regularizations appear). Energy efficiency!

Takeway message

- Training is a still enigmatic procedure where classical algorithms meet fascinating random effects on high dimensional parameter space.
- Implicit and explicit regularizations appear and shape the properties of the solution decisively.
- Transfer learning: training hidden layers and multi-variate outputs leads to passing information from one output dimension to another one (L^1 implicit regularizations appear). Energy efficiency!

Applications in Finance

- approximation of path space functionals, or more generally, predictable strategies by neural networks on relevant factors.
- Examples: deep hedging, deep portfolio optimization, deep drift estimation, signature based pricing and hedging, sig-SDEs, reservoir computing for learning dyncamics, stochastic optimization, stochastic games beyond Markovian paradigms, etc.
- some of these applications are quite successful, but still lack a full theoretical foundation why the non-convex optimization problem can be solved so efficiently or why existing approximation results are generically sufficient.

Applications in Finance

- approximation of path space functionals, or more generally, predictable strategies by neural networks on relevant factors.
- Examples: deep hedging, deep portfolio optimization, deep drift estimation, signature based pricing and hedging, sig-SDEs, reservoir computing for learning dyncamics, stochastic optimization, stochastic games beyond Markovian paradigms, etc.
- some of these applications are quite successful, but still lack a full theoretical foundation why the non-convex optimization problem can be solved so efficiently or why existing approximation results are generically sufficient.

Applications in Finance

- approximation of path space functionals, or more generally, predictable strategies by neural networks on relevant factors.
- Examples: deep hedging, deep portfolio optimization, deep drift estimation, signature based pricing and hedging, sig-SDEs, reservoir computing for learning dyncamics, stochastic optimization, stochastic games beyond Markovian paradigms, etc.
- some of these applications are quite successful, but still lack a full theoretical foundation why the non-convex optimization problem can be solved so efficiently or why existing approximation results are generically sufficient.

Section 2

Machine learning in Finance: Deep Hedging

Deep Hedging

- Risk management of future liabilities in a real world market including transaction costs, liquidity constraints, price impacts, etc.
- Classical approach: choose a model (calibrated to the market), choose preferences, simplify the model to make it analytically tractable and solve the respective optimization problems.
- Modern approach: choose a model (calibrated to the market), choose preferences, parameterize artificial trading agents and train them to handle the optimization tasks.

Deep Hedging

- Risk management of future liabilities in a real world market including transaction costs, liquidity constraints, price impacts, etc.
- Classical approach: choose a model (calibrated to the market), choose preferences, simplify the model to make it analytically tractable and solve the respective optimization problems.
- Modern approach: choose a model (calibrated to the market), choose preferences, parameterize artificial trading agents and train them to handle the optimization tasks.

Deep Hedging

- Risk management of future liabilities in a real world market including transaction costs, liquidity constraints, price impacts, etc.
- Classical approach: choose a model (calibrated to the market), choose preferences, simplify the model to make it analytically tractable and solve the respective optimization problems.
- Modern approach: choose a model (calibrated to the market), choose preferences, parameterize artificial trading agents and train them to handle the optimization tasks.

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 $(\Omega, \mathcal{F}, \mathbb{P})$ with filtration $\mathbb{F} = (\mathcal{F}_{t_k})_{k=0,\ldots,n}$, for simplicity $\mathcal{F}_{t_k} = \sigma(S_{t_0}, \ldots, S_{t_k})$.
- At $t = 0$ sell a contingent claim with (random) payoff Z at $T > 0$.
- Charging price p_0 and hedging according to an $\mathbb{F}$-predictable strategy δ, 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

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

(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:

- Following e.g. Föllmer, Klöppel, Leukert, Schweizer, Sircar, Xu, ... :
- Describe risk-preferences by a convex risk-measure ρ.
- Denote $\mathcal{H}$ the set of available hedging strategies.
- The indifference price is the (unique) solution $p(Z)$ to

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

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

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

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

Indifference pricing and optimal hedging:

- Following e.g. Föllmer, Klöppel, Leukert, Schweizer, Sircar, Xu, ... :
- Describe risk-preferences by a convex risk-measure ρ.
- 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 δ^* (if it exists) in left-hand-side of (2).

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

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

- We show: now approximate calculation is feasible thanks to modern deep learning techniques.
- 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^{θ_k} is a neural network with weights parametrized by θ_k.
- Key point 1: neural networks are surprisingly efficient at approximating multivariate functions.
- Key point 2: efficient machine learning optimization algorithms (stochastic gradient-type and backpropagation) and implementations (Tensorflow, Theano, Torch, ...) are available.

Our method is sample-based and highly flexible: the same algorithm and implementation can handle wide range of market specifications.

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 $d\langle B, W\rangle = \rho dt$

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

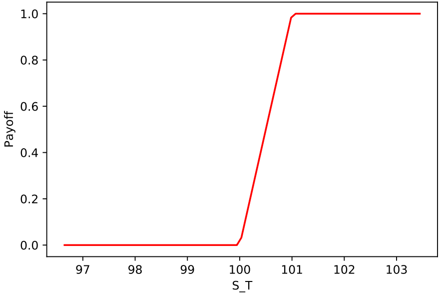
Payoff and Hedging

- 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: α-CVar (expected shortfall),

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

Call spread

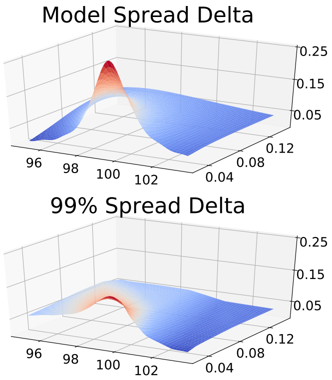
- Used by traders for (approximate) pricing / hedging of binary options.
- Payoff: $-\frac{1}{K_2-K_1}[(S_T^{(1)}-K_1)^+-(S_T^{(1)}-K_2)^+]$ for $K_1<K_2$.
- Here $K_1=s_0=100$, $K_2=101$:



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^{θ_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

Section 3

Machine Learning in Finance: Model free Deep Hedging

Estimate noting

A well known Bayesian approach to model selection has been introduced by Chris Rogers and Moritz Dümbgen.

- Choose a pool of models Θ with pricing, hedging and prediction operators mapping current states and contract specifications to the relevant quantities or operations.
- Choose a prior on Θ.
- Choose a likelihood which compares incoming data to model quantities and update the prior according to Bayes formula.
- After a burn-in phase the posterior is *not* used to select a model, but the posterior is rather applied as defining a model mixture. All sorts of operations are weighted with respect to this mixture. In this sense no model is selected.

Estimate noting

A well known Bayesian approach to model selection has been introduced by Chris Rogers and Moritz Dümbgen.

- Choose a pool of models Θ with pricing, hedging and prediction operators mapping current states and contract specifications to the relevant quantities or operations.
- Choose a prior on Θ.
- Choose a likelihood which compares incoming data to model quantities and update the prior according to Bayes formula.
- After a burn-in phase the posterior is *not* used to select a model, but the posterior is rather applied as defining a model mixture. All sorts of operations are weighted with respect to this mixture. In this sense no model is selected.

Estimate noting

A well known Bayesian approach to model selection has been introduced by Chris Rogers and Moritz Dümbgen.

- Choose a pool of models Θ with pricing, hedging and prediction operators mapping current states and contract specifications to the relevant quantities or operations.
- Choose a prior on Θ.
- Choose a likelihood which compares incoming data to model quantities and update the prior according to Bayes formula.
- After a burn-in phase the posterior is *not* used to select a model, but the posterior is rather applied as defining a model mixture. All sorts of operations are weighted with respect to this mixture. In this sense no model is selected.

Estimate noting

A well known Bayesian approach to model selection has been introduced by Chris Rogers and Moritz Dümbgen.

- Choose a pool of models Θ with pricing, hedging and prediction operators mapping current states and contract specifications to the relevant quantities or operations.
- Choose a prior on Θ.
- Choose a likelihood which compares incoming data to model quantities and update the prior according to Bayes formula.
- After a burn-in phase the posterior is *not* used to select a model, but the posterior is rather applied as defining a model mixture. All sorts of operations are weighted with respect to this mixture. In this sense no model is selected.

Model free Deep Hedging

The colorful advantage of artificial traders is their incredible robustness with respect to market frictions.

- classic Deep Hedging: scenarios are exogenously given and constitute the training data.
- idea: train the artificial trader for any model $\theta \in \Theta$ (choose Θ appropriately to allow for appropriate continuities).
- use the Dümbgen-Rogers approach to hedge via mixing the strategies for each θ.

Model free Deep Hedging

The colorful advantage of artificial traders is their incredible robustness with respect to market frictions.

- classic Deep Hedging: scenarios are exogenously given and constitute the training data.
- idea: train the artificial trader for any model $\theta \in \Theta$ (choose Θ appropriately to allow for appropriate continuities).
- use the Dümbgen-Rogers approach to hedge via mixing the strategies for each θ.

Model free Deep Hedging

The colorful advantage of artificial traders is their incredible robustness with respect to market frictions.

- classic Deep Hedging: scenarios are exogenously given and constitute the training data.
- idea: train the artificial trader for any model $\theta \in \Theta$ (choose Θ appropriately to allow for appropriate continuities).
- use the Dümbgen-Rogers approach to hedge via mixing the strategies for each θ.

Results

- A fully data driven hedging algorithm for asset liability management.
- An adaptive artifical trader who can learn new regimes online: at any point in time Θ can be extended to include further models.
- Also preferences can be θ dependent.

Results

- A fully data driven hedging algorithm for asset liability management.
- An adaptive artifical trader who can learn new regimes online: at any point in time Θ can be extended to include further models.
- Also preferences can be θ dependent.

Results

- A fully data driven hedging algorithm for asset liability management.
- An adaptive artifical trader who can learn new regimes online: at any point in time Θ can be extended to include further models.
- Also preferences can be θ dependent.

Outlook

- use different computing architectures (reservoir computing).
- use different mixing methods (the Dümbgen-Rogers one is closely related to quadratic optimization problems).

Outlook

- use different computing architectures (reservoir computing).
- use different mixing methods (the Dümbgen-Rogers one is closely related to quadratic optimization problems).

References

- H. Bühler, L. Gonon, J. Teichmann, and B. Wood:
  Deep Hedging, Arxiv, 2018.
- M. Dümgen, R. Rogers:
  Estimate nothing, Arxiv, 2014.
- T. Krabichler, J. Teichmann:
  Deep Asset Liability management, Arxiv, 2020.