Name of the Thesis

Name of the Author

Name of your College

University of Oxford

A thesis submitted in partial fulfillment of the MSc in

Mathematical Finance

May 20, 2022

Abstract

abstractasdfasdfasdfasdfasdfasdfasdfsdfasdf

Contents

1 Introduction

Motivation

Consider a typical workflow for a end-user in a trading desk: a price is needed for a particular product, a pricing function is invoked, which takes in a collection of (calibrated) input parameters, which in turn invokes the underlying numerical method, and finally an output is returned.

Arguably, one key business objective for derivatives quants at investment banks and market makers is to streamline and improve this process: for any given product and *market generator model*, and their parameters, produce a pricing function that outputs its *prices*, its *sensitivities*, and the *hedging strategy* as accurately as possible, and as quickly as possible.

To accomplish this, derivatives quants have broadly three numerical methods: analytic and semi-analytic form solutions (e.g. Black Scholes, Characteristic Transform), Monte Carlo Methods, and PDE / finite-difference methods. Each of these methods has their own drawbacks and limitations, and with the exception of few true closed-form expressions, are themselves approximation methods.

Approximation error to the 'true' model price can be reduced to some extent with greater computational effort. The computational effort may grow very large when the underlying dimensionality of the problem increases, for example with more complex payoffs and volatility models. When the *risk* sensitivities (which may also represent the hedging strategy) also need to be computed, the computational effort may also grow large, given the convergence rates in price and sensitivities may not necessarily be the same.

Towards some practical applications, it may be acceptable to trade off some accuracy in exchange for speed, for example in the context for electronic market-making or risk / VaR calculations.

Neural-networks may be one method to accomplish this. Neural networks, in this setup, may be viewed as as a model agnostic (in the sense of volatility or market generator models) and method agnostic (MC, PDE) extension to existing numeric pricing methods. Neural Networks present several desirable properties: they are universal approximation, and non-parametric data-driven relationships, sensitivities can be obtained quickly with automatic differentiation, and a fast inference time can be obtained, amortising a computationally expensive training period.

A requisite for any pricing function is adherence to no-arbitrage, and similarly, smoothness of the pricing function and sensitivities in all inputs. Naturally, if *static* arbitrage opportunities exists, this presents the possibility of a loss to the trading desk. In terms of smoothness, an exotics or OTC desk may have to price derivatives beyond standard strikes and maturities, and a non-smooth pricing function is likely to admit static arbitrage. A non-smooth sensitivities functional may also present PNL impact from hedging. A potential issue is that a standard neural network may not necessarily adhere to no-arbitrage.

The aim of this dissertation is to investigate how to construct a production-level neural network for derivatives pricing and risk that can be:

- How to construct and train a neural network efficiently, in terms of both approximation error and minimal no-arbitrage violations?
- How do neural network approximations perform under various market models (Black-Scholes, Heston, SABR, Rough Vol)?
- How well does it perform in high dimensionality?
- How well does it perform under different payoffs (Calls and Digitals, Asians), and exercise policy (Bermudan, European)

Article Structure

In section one, we review the relevant pricing problem formulations. In section two, we review neural networks and the relevant literature. In section three, we focus on numerical settings, examining how to construct and efficiently train neural networks that are able to approximate European and Bermudan payoffs.

2 Preliminaries

In this section, we review the formulation of derivatives pricing and risk calculations.

Problem Setup

Consider the problem of computing derivative prices and sensitivities with respect to some input parameters $\mathbf{X}$.

Consider a filtered probability space $(\Omega, \mathcal{F}, \mathcal{F}_t, \mathbb{P})$. Suppose there is a $\mathcal{F}_t$-adapted d-dimensional stochastic process $\mathbf{S}_t \in \mathbb{R}^d, t \in [0, T]$, representing the value of the underlying assets or cash flows through time whose evolution is determined by a fixed vector of parameters $\mathbf{X} \in \mathbb{R}^f$. Thus in effect, we consider $\mathbf{S}_t(\mathbf{X})$.

Suppose that there exists some equivalent local martingale measure $\mathbb{Q}$

In the most general case, consider a payoff determined by a continuous discounted payoff and a terminal discounted payoff:

$$f(\tau, \mathbf{S_t}, \mathbf{X}) = \mathbb{E}^{\mathbb{Q}} \left[\int_t^T h_1(u, S_u) dt + h_2(S_T) | \mathbf{S}_t, \mathbf{X} \right] \tag{1}$$

Suppose f is sufficiently smooth, in particular C^1 with respect to τ and C^2 with respect to each $\mathbf{S}_t$. Then by Feynman-Kac, the corresponding (parametric) PDE is given by:

$$f_\tau = \mathcal{L} f, \quad \forall \tau \in [0, T], \mathbf{X} \in \mathcal{X}, s \in \mathcal{S}$$

subject to some initial (in terms of time-to-maturity) condition corresponding to the European Payoff, and $\mathcal{L}$ is the generator of $\mathbf{S}$

Payoffs may also be path dependent, and depend on the entire history of $\mathbf{S_t}$:

$$\mathbb{E}^{\mathbb{Q}} \left[\int_t^T h_1(u, (S_u)_t^u) dt + h_2((S_u)_0^T) | X \right] \tag{2}$$

In a 'real-life' scenario, consider

$$\mathbb{E}^{\mathbb{Q}}[\sum_{i=1}^{n} h_i(t_i, \mathbf{S}_i) | \mathbf{S}_{t_0}, \mathbf{X}]$$

More generally, we also consider the underlying parameters $\mathbf{X}$ which determine $\mathbf{S}_t$

Suppose $\mathbb{E}[g(X_n)^2] < \infty$ the payoff is L^2 integrable.

and that $h(\mathbf{X}_t)$ is a $\mathcal{F}_T$-measurable random variable representing the (discounted) total sum of cash flows, as some function of the time t state $\mathbf{X}$, under some risk-neutral measure $\mathbb{Q}$.

Stochastic Control approach

In general, both the pricing function f and $\frac{\partial f}{\partial \theta}$ in (4), cannot be obtained in closed-form with the exception of several cases. Thus in practice we have:

$$g(T-t,\mathbf{X}) \approx f(T-t,\mathbf{X}), \quad f(T-t,\mathbf{X}) = g(\mathbf{X}) + \epsilon$$

Where ϵ denotes the error, which may itself depend on θ and the choice of numerical method. We look for a approximating function g such that ϵ is small over a relevant range of parameters $\mathbf{X}$.

In addition to the value f, the sensitivities with respect to the parameters $\mathbf{X}$ are also of interest. Thus we also require

$$\frac{\partial g}{\partial \mathbf{X}} \approx \frac{\partial f}{\partial \mathbf{X}}, \ldots, \frac{\partial^d g}{\partial \mathbf{X}}$$

For example, if $\theta = S_0$, then $\frac{\partial f}{\partial S_0}$ gives the sensitivity to the underlying, and the hedging strategy.

American

$$\sup_{\tau} \mathbb{E}[\int_t^{\tau} B_{t,T} h_1(X_t) dt + h_2(X_\tau)|X] \tag{3}$$

$$f(T-t,\mathbf{X}_t) = \mathbb{E}^{\mathbb{Q}}[h(\mathbf{X}_t)|\mathbf{X}] \tag{4}$$

Then f is the conditional expectation and value function for a derivative with latest cash flow at time T. Under some risk-neutral measure $\mathbb{Q}$, or equivalently the assumptions of no-arbitrage and market completeness, then f is the pricing function with respect to the parameters θ.

Longstaff-Schwartz

[15] introduced the Longstaff-Schwartz method, or Least Squares Monte Carlo for valuing American Options. In the one step case, consider

$$f(X_t(\theta)) = \mathbb{E}^{\mathbb{Q}}[h(X_T(\theta))|X_t(\theta)] \tag{5}$$

Where X is some appropriately chosen (Markov) state process such that $f(X_t(\theta))$. Then let g denote an approximation for the value function. In [15], they consider basis functions such as Chebyshev and Legendre polynomials.

$$g(X_t) = \sum_{b=1}^{B} \phi_b(X)$$

In the multi-period case , we consider for $t_0 < t_i < \ldots t_n = T$:

$$f(X_{t_i}) = \max\{\mathbb{E}^{\mathbb{Q}}[f(X_{t_{i+1}})\}|X_{t_i}], h(X_{t_i})\}, \quad f(X_T) = h(X_T)$$

Where h is the exercise value for the pay

[15] argues that as the number of basis functions B to infinity, g approximates the true value function, and as the number of timesteps N goes to infinity. Further

to this, in the Monte Carlo setting, we also require the number of samples M to go to infinity.

$$f(S_t) = \mathbb{E}[(S_T - K)^+ | S_t] \qquad \text{(Black-Scholes)}$$

$$\mathbb{E}[(f(S_T) - \mathbb{E}[g(S_t)|S_t])^2] = 0$$

Then by the tower property of expectation

$$\mathbb{E}[(g(S_T) - \mathbb{E}[g(S_t)|S_t])^2] = 0$$

Adjoint Automatic Differentiation

The adjoint automatic differentiation method was first brought to [**giles2004**]

In the most general case, consider the pricing function in the form of 4. Given regularity constraints

$$\frac{\partial f}{\partial \theta} = \frac{\partial}{\partial \theta}\mathbb{E}^{\mathbb{Q}}[]$$

The sensitivities of f with respect to θ, can be computed through the application of the chain rule backwards:

$$\frac{\partial S}{\partial \theta} \cdot \frac{\partial f}{\partial \theta} = \frac{\partial f}{\partial \theta}$$

where the partials are evaluated in reverse order. Programming language wise, it can be implemented via computional graphs and implementing *adjoint* operators.

No-Arbitrage Constraints

We consider the conditions for our approximating function g to be free of static arbitrage in the case of European calls. Suppose that g is continuously and and once-differentiable with respect to $\tau = T - t$ and twice differentiable with respect to K or $x = S/K > 0, y = \log(S/K)$ omitting other arguments. We note that: $\frac{\partial X}{\partial K} = \frac{S}{-K^2} = \frac{-X}{K}, \frac{\partial y}{\partial K} = \frac{-1}{K}$. Then:

$$\begin{aligned}
\frac{\partial g}{\partial x}\frac{\partial x}{\partial K} &= \frac{-x}{K}\frac{\partial g}{\partial X} \\
\frac{\partial g}{\partial y}\frac{\partial y}{\partial K} &= -\frac{1}{K}\frac{\partial g}{\partial y} \\
\frac{\partial}{\partial K}(-\frac{x}{K}\frac{\partial g}{\partial x}) &= \frac{1}{K^2}(x\frac{\partial g}{\partial X} + x^2\frac{\partial^2 g}{\partial X^2}) \\
\frac{\partial}{\partial K}(-\frac{1}{K}\frac{\partial g}{\partial y}) &= \frac{1}{K^2}(\frac{\partial g}{\partial y} + \frac{\partial^2 g}{\partial y^2})
\end{aligned}$$

Then in terms of K, x, y we have::

Static Arbitrage: [9] [5]

$\frac{\partial g}{\partial \tau} \geq 0$	carry, time-value
$\frac{\partial g}{\partial K} \leq 0, \frac{\partial g}{\partial x} \geq 0$	monotonically decreasing in strike, increasing in moneyness/underlying
$\frac{\partial^2 g}{\partial K^2} \geq 0, \frac{\partial^2 g}{\partial x} \geq 0, \frac{\partial^2 g}{y} \geq 0$	convexity in strike
$g(T-t, K) \geq (S_t - K)^+ \geq 0$	intrinsic value
$\lim_{K\to\infty} g(K, T-t) = 0$	1

We also consider no static arbitrage for European digitals. Since the payoff of a European digital $\mathbb{Q}[]$

$\frac{\partial g}{\partial \tau} \geq 0$	carry, time-value
$\frac{\partial g}{\partial K} \leq 0, \frac{\partial g}{\partial x} \geq 0$	monotonically decreasing in strike, increasing in moneyness/underlying
$\frac{\partial^2 g}{\partial K^2} \geq 0, \frac{\partial^2 g}{\partial x} \geq 0, \frac{\partial^2 g}{y} \geq 0$	convexity in strike
$g(T-t, K) \geq (S_t - K)^+ \geq 0$	intrinsic value
$\lim_{K\to\infty} g(K, T-t) = 0$	1

$$\int_{\mathbb{R}} (y-K)^+ p(y, T; s, t) dy = \int_K^\infty (y-K) p(y, T; s, t) dy$$

Breeden-Litzenberg formula. We now have an additional approach: solve for the risk-neutral transition density $p(y, T; s, t)$ and obtain the price of any european payoff via numerical integration.

No dynamic arbitrage indicates the lack of a replication strategy with zero initial wealth that leads to positive wealth $\mathbb{P}$-almost surely. For Dynamic Arbitrage, a sufficient condition may be that the function g satisfies the relevant pricing PDE.

$$\mathcal{L}g = g_\tau, \quad \text{for all } \tau, K$$

If $\mathcal{L}g \neq 0$, the dynamic arbitrage strategy is given by longing the payoff if $\mathcal{L}g$ and shorting the replicating portfolio, and $\mathcal{L}g < 0$, and shorting the payoff if $\mathcal{L}g < 0$ and longing the replicating strategy.

Example: Black-Scholes

In the most simple case, we could consider a European payoff on some underlying

$$f(T-t, S_t) = \mathbb{E}^{\mathbb{Q}}[h(S_t \cdot \frac{S_T}{S_t})|S_t]$$

In this setup, consider the case of a call option:

$$\mathbb{E}[e^{-r(T-t)}(S_T - K)^+|S_t] = \mathbb{E}[(S_t \exp(-\frac{\sigma^2(T-t)}{2} + \sigma(W_T - W_t)) - K)^+]$$

If we use Monte Carlo pricing

3 Neural Networks

Neural Network Definition

We aim to train a neural network to approximate a pricing function, over as wide a range as possible under a given market generator model, that has a minimal approximation error and minimal static and dynamic arbitrage.

A comprehensive outline of neural networks is better outlined in [11], and a reference on practical implementations using `Tensorflow/Keras` can be found in [7]. An overview of neural networks and their applications towards finance can be found in [8].

Consider some input $\mathbf{X} \in \mathbb{R}^d$. Then a n-layer standard feed-forward neural network with n layers can be characterised by :

$$\mathbf{Z}_1 = g_1(\mathbf{X}\mathbf{W}_1 + \mathbf{b}_1\mathbf{1}^\top) \tag{6}$$

$$\mathbf{Z}_2 = g_2(\mathbf{Z}_1\mathbf{W}_2 + \mathbf{b}_2\mathbf{1}^\top) \tag{7}$$

$$f(\mathbf{X};\theta) = g_{n+1}(\mathbf{Z}_n\mathbf{W}_n + \mathbf{b}_n\mathbf{1}^\top) \qquad \text{(feed-forward neural network)}$$

The first point to note, is that a neural network can be regarded as a composition of neural networks. Another point to note, is that a neural network can be regarded as a non-linear transformation into a latent representation, or set of basis functions, of the inputs.

Neural Network Properties

Neural Networks in their capacity as function approximators can be characterised as

$$\arg\min_{\theta\in\Theta} L(f(\mathbf{X}), g)$$

the minimizer of some loss function L, over a space of feasible parameters Θ.

$$\arg\min_{\theta\in\Theta} \mathbb{E}[\|\mathbf{y} - f(\mathbf{X};)\|^2]$$

Universal approximation theorem of Hornik [Theorem 2]Hornik1991 , Cybenko:

Where , $\mathbf{W}_i \in \mathbb{R}^{H_{i-1},H_i}, i = 1, \dots n-1, W_n \in \mathbb{R}$ denotes the .

Finally $g_i, i = 1, \dots, n$, is the activation function for the i-th hidden layer. The activation function is applied element-wise to the inputs $g(\mathbf{Z})_{ij} = g(z_{ij})$, and introduces non-linearities. Typically, the final layer is the identity (or linear) activation $g(z_{ij}) = z_{ij} \in \mathbb{R}^n$, such that the final value may attain any real value, although a different

Thus in effect, the neural network is , learns a set of basis functions $\mathbf{Z}_n$.

ReLU : $g(z_{ij}) = (z_{ij})^+$, Elu $g(z_{ij}) = \alpha(e^{z_{ij}} - 1)$, Softplus: $\log(e^x + 1)$, Swish, Sigmoid $\frac{1}{1+e^{-z_{ij}}}$

Another particular architecture of interest is *residual* networks, introduced in:

We consider several hidden layer:

Gated unit: A gated . In the context, this can help learn feature interactions, and also facilitate feature *selection*.

$$\mathbf{Z}_1 = g_1(\mathbf{X}\mathbf{W}_1 + \mathbf{b}\mathbf{1}^\top) \tag{8}$$
$$\mathbf{Z}_2 = g_2(\mathbf{X}\mathbf{W}_2 + \mathbf{b}\mathbf{2}^\top) \tag{9}$$
$$\mathbf{Z}_3 = \mathbf{Z}_1 \otimes \mathbf{Z}_2 \quad \text{(gated block)}$$

Residual block: Residual blocks allow for the *flow* of information from earlier layers to later layers

$$\mathbf{Z}_1 = g_1(\mathbf{X}\mathbf{W}_1 + \mathbf{b}_1\mathbf{1}^\top) \tag{10}$$
$$\mathbf{Z}_2 = g_2(\mathbf{Z}_1\mathbf{W}_2 + \mathbf{b_2}\mathbf{1}^\top) \tag{11}$$
$$\mathbf{Z}_3 = g_3(\begin{pmatrix}\mathbf{Z}_1 & \mathbf{Z}_2\end{pmatrix}\mathbf{W}_3 + \mathbf{b}_3\mathbf{1}^\top) \quad \text{(gated block)}$$

Recurrent neural networks: Recurrent neural networks may be useful in a non-markovian setting, for example, in pricing path dependent payoffs. Furthermore, they may be useful even for European payoffs, when additional market frictions such as transaction costs or market impact are included, as in [4].

$$\mathbf{Z}_1 = g_1(\mathbf{X}\mathbf{W}_1 + \mathbf{b}\mathbf{1}^\top) \tag{12}$$
$$\mathbf{Z}_2 = g_2(\mathbf{Z}_1\mathbf{W}_2 + \mathbf{b}\mathbf{1}^\top) \tag{13}$$
$$\mathbf{y} = f(\mathbf{X};\theta) = g_n(\mathbf{Z}_n\mathbf{W}_n + \mathbf{b}\mathbf{1}^\top) \quad \text{(gated block)}$$

Neural Network Construction

We characterise handcrafted neural networks as described in [17]: in short, we can embed prior knowledge into the construction of the neural network, in the choice of an appropriate loss function or architecture.

Loss Functions for Derivatives Pricing and Risks

The proposed method from [13] is to consider a joint loss function, an approach they describe as *differential machine learning*. Generate state payoff pairs: $X_i, g(X_i)$

$$L_{price}(g(X_i), f(X_i)) + \lambda L_{greeks}\left(\frac{\partial g(X_i)}{\partial X}, \frac{\partial g(X_i)}{\partial X}\right), \lambda \geq 0 \tag{14}$$

Again, $\frac{\partial f(X_i)}{\partial X}$ can be obtained at little extra cost given AAD. [13] argues that the λ, but in practice this could be determined by trial-and-error and prior knowledge.

A further extension could be to consider the, which is a *Neural PDE* approach

$$L_{price}(g(X_i), f(X_i)) + \lambda_1 L_{greeks}\left(\frac{\partial g(X_i)}{\partial X}, \frac{\partial g(X_i)}{\partial X}\right) + \lambda_2 L_{PDE}(\mathcal{L}g) \tag{15}$$

Where $\mathcal{L}$ denotes a PDE operator and $\lambda_1, \lambda_2 \geq 0$ are weights for each loss function.

$$\mathcal{L}g = \frac{\partial g}{\partial t} + rs\frac{\partial g}{\partial s} + \frac{\sigma^2 s^2}{2}\frac{\partial^2 g}{\partial s^2} - rg = 0$$

[19] argues that the inclusion of a PDE loss term leads to self-consistency, given that if the neural network approximation satisfies the pricing PDE (in their application, the Dupire Local Volatility PDE), there is no dynamic arbitrage.

An alternative is a *soft penalty* approach:

- Calls: $\frac{\partial C}{\partial K} \leq 0, \frac{\partial^2 C}{\partial K^2} \geq 0, C \geq (S-K)^+$

One such example could be a penalty $((S-K)^+ - g(S))^+$

Thus let $L_1, \ldots L_P$ denote

Architecture

Smoothness. Firstly, to obtain smooth (or at least, continuous) approximations for the d-th order partial derivatives, we require $g(\cdot)$ to be C^d continuous d-time differentiable with respect to its inputs. Given the loss functions in 14, 15, may contain even higher-order partial derivative terms, we may require even further smoothness constraints.

In order for the neural network output g to be C^d, one method could be to constrain all intermediate activation functions g_i to be sufficiently smooth, such that g as a composition of smoothness will be a smooth function. [6] uses this approach.

Hard constraints [6] *softplus* activation for the strike and sigmoid activation for the time-to-maturity, with non-negative weight constraints. Thus this guarantees that the neural network is non-negative, monotonic in strike and time, and convex in strike.

Model risk-neutral implied distribution: Let $g(T, y/K; t, X)$ be a estimate for the conditional density at time T. then

$$f(T-t, X) \approx DF_{t,T}\int_K^\infty (\frac{y}{K} - 1)^+ g(T, y; t, X)dy$$

We can consider some

$$\sum f(x_i)\Delta_i$$

Model Residuals: [1] highlights that neural networks are generally able to interpolate within the domain of training, but unable to extrapolate. This is also have particular relevance, given models have asymptotics, and payoff asymptotics correspond to no-arbitrage bounds.

The proposed solution of [1] uses a control-variate-like two-step procedure: first, they fit a *control variate function* , in their case cubic spline, with the appropriate

asymptotics, then they fit a neural network with vanishing asymptotics to the residual of the original function and the control variate.

Their proposed architecture consists of the following gaussian, or radial basis activation

$$g_i(\mathbf{x}) = \exp(-\frac{1}{2}\|\mathbf{x}\mathbf{W}_i + \mathbf{b}_i\|^2) \tag{16}$$

Such that if any $x_i \to \pm\infty$ we have $g_i(\mathbf{x}) \to 0$

Model Hedging Strategy

The Deep Hedging approach of [4] considers approximating the replicating strategy. Thus in this case, the neural network represents the delta.

$$y - \sum_{i=1}^{N} \Delta_i(X_{t_i})(S_{i+1} - S_i)$$

Learn the hedging strategy with a neural network, and price can be obtained via superhedging. However

A potential drawback of this approach is that a single neural network is tuned to

Training Neural Networks

In general, given some loss function $L(g(\mathbf{X}), f(\mathbf{X}))$, and sample data: $\mathbf{X}_j, j = 1, \ldots, N$

$$\arg\min_{\theta} L(f, g)$$

$$\theta_{t+1} \leftarrow \theta_t - \eta_t \nabla_\theta L, \quad \eta > 0 \tag{17}$$

This loss function is generally non-convex with respect to the parameters θ; as such there are no guarantees of convergence to a global minima except in.

In practice the gradients with respect to the neural network parameters $\nabla_\theta L$ also need to be estimated. Mini-batch stochastic gradient descent

$$\sum \theta_{t+1} \leftarrow \theta_t - \eta_t \nabla_\theta L, \quad \eta > 0 \tag{18}$$

Related Work

In the derivatives domain, various approaches utilising neural networks have been presented: A comprehensive review of methods can be further found in [17]:

- Supervised: Direct approximation of model prices, with automatic differentiation for sensitivities, as in [14] ,
- Calibration: learn mappings for implied volatilities/prices to parameters, or vice-versa [12]

- Semi-supervised / Reinforcement-learning; Approximation for a hedging or replicating strategy; In effect this can be seen as pricing via replication or stochastic control [4]
- *Neural PDEs*: Use of neural networks to solve (high-dimensional) PDEs []
- *Neural SDEs*: paramterise a SDE as a nueral network, as in [10] and Generative Adversarial Networks

Interpreting Neural Networks

We only use neural networks as an approximating function as an overlay on top of some market generator model and numerical method, which may alleviate some of the *black-box* issues. However, the neural network may still produce unexpected outputs. We can evaluate and interpret neural networks to some extent. In the simple case, we can use graphical methods, or evaluate behaviour around boundary conditions. In addition, we can leverage machine learning interpretability methods [16] [3].

- Dependency plots against one or two dependent variables, boundary conditions.
- First order partial derivatives, Second order (Hessian), higher order derivatives.
- Output of penultimate layer (basis functions or latent representation)
- Machine-learning interpretaabilty methods: Shapley Values, LIME

Dataset

The neural network approximation approach is more closely aligns with a Monte Carlo method. Thus there is error from both the neural network approximation g and the simulation of state-payoff pairs X_j, y_j

For example, consider the case of very deep in-the-money or out-of-the money strikes for a European all. A proposed solution of is to sample more from these regions, or similarly to set a greater weight w_j in a weighted loss function

$$\sum_{j=1}^{N} w_j L(f(X_j), g(X_j))$$

In the Longstaff-Schwartz case, paths for which $f(X_j)$ have weight zero.
Preprocessing, remove arbitrage out of the money paths.
Swaption

$$\sum_{i=1}^{n} L(T_{i-1}, T_i) P(T, T_i) - s \sum_{j=1}^{n} \cdot P(t, T_j) = 0$$

Data Generation Method:

Train, Val: Hypercube / Grid sampling vs Random sampling, Random sampling within boundary

Quasi Monte Carlo, Sobol Sequences, Control Variates

Test: Another random sample within boundary, random sample outside of boundary

4 Numerical Experiments

- Bachelier Model (Arithmetic Brownian Motion) Basket Options
- Black-Scholes (Geometric Brownian Motion), time-scaled vol + delta
- Local Vol
- Fixed Income Examples: Vasciek, Cheyette, CIR, Hull White, LIBOR market model / BGM, HJM, Swaps, Caplets, Floors, Swaptions
- Path-Dependent: Asian, Barriers,
- Optionality: Americans, Bermudans, Callables

Metrics:

- Time Complexity (Training, Evaluation of Prices + Derivatives)
- Dataset (Size (Timesteps, Number of MC paths), method)
- Architecture: ResNet, Standard
- Objective: Standard, Differential, PDE, Deep Hedging, Residual / Control-Variate, Neural SDE
- Model Complexity (No. of parameters, (width, depth)
- No. of no-arbitrage violations, asymptotic behaviour, extrapolation behaviour
- Errors (L1, L2 , Linf) in Price, First Order, Second Order sensitivities
- Comparison vs MC, PDE, Closed form, other methods
- Pathwise Hedging Error ($V_T - \sum_{i=1}^{N} \Delta_{t_i} \cdot (S_{t_i+1} - S_{t_i}) - P_0$) (Loss, CVaR / Quantiles)

An issue is that the NN may be able to , e.g. when the market regime shifts. Thus the NN must likely be trained on a very large grid of parameters.

The end-user invokes some the NN-version of the pricing function in lieu of the computationally expensive pricing function.

Alternatives:

- Signatures [18]

- Chebyschev Methods, Tensor Methods [2] https://github.com/piterbarg/altnnpub/blob/main/gss_e

$$dX(t,\theta) = b(t; X_{t,\theta}; \theta)dt + \sigma(t; X_{t,\theta}; \theta)dW_t$$

A generative adversarial network approach to calibration of local stochastic volatility models

Deep neural networks, generic universal interpolation, and controlled odes

$$S_t = S_t[rdt + \sigma(t, S_t)dW_t]$$

[10] suggests initialising multiple random seeds, to obtain a confidence interval on prices

$$P(\theta^-), P(\theta^+)$$

Workflow

- Construct Dataset
- Define Model
- Train Model
- Model Diagnosis
- Use Model for Inference

Arithmetic Brownian Motion / Bachelier Model

We consider the first example from [13], a standard European Call on a basket option where the underlyings have dynamics

$$\begin{aligned} \mathbf{S}_t &= \mathbf{S}_0 + \mathbf{L}\mathbf{W}_t, & \mathbf{S}_t \in \mathbb{R}^d, \mathbf{W}_t \in \mathbb{R}^f \text{ for all } t \in [0,T], \text{ and } \mathbf{L} \in \mathbb{R}^{d\times f} \\ \mathbf{X}_t &= \mathbf{w}^\top \mathbf{S}_t, & \mathbf{w} \in \mathbb{R}^d, \mathbf{X}_t \sim N(\mathbf{w}^\top \mathbf{S}_0, \mathbf{w}^\top \mathbf{L}\mathbf{L}^\top \mathbf{w}) \end{aligned}$$

Where $\mathbf{w}_i = \frac{U_i}{\sum_{i=1}^n U_i}$ is a random weight such that the weights of the basket sum to one, $\mathbf{L}\mathbf{L}^\top$ is the covariance matrix, $\mathbf{S}_0$ denotes the initial values of the underlying assets and

Geometric Brownian Motion / Black-Scholes

We consider the second example from [13], 1D Black-Scholes for a single time-to-maturity $T - t$ with respect to different levels of the underlying S_t.

$$\begin{aligned} \mathbf{S}_T &= \mathbf{S}_t \exp((r - \frac{\sigma^2}{2})(T-t) + \sigma(W_T - W_t)) \\ h(S_T) &= \left(\frac{S_T}{K} - 1\right)^+ \end{aligned}$$

The analytic solution is given by the Black-Scholes formula:

$$d_{\pm} = \frac{\log(S_t e^{r(T-t)}/K)}{\sigma\sqrt{T-t}} \pm \frac{(\sigma\sqrt{T-t})^2}{2} g(e^{r(T-t)}S_t/K, \sigma\sqrt{T-t}) \quad = \frac{e^{r(T-t)}S_t}{K}\Phi(d_+) - \Phi(d_-)$$

parameter	**range**
$\log(S_t/K)$	[-3, 3]
$\sigma\sqrt{T-t}$	[0, 3]

We parameterise in terms of moneyness $\frac{S_T}{K}$ and exploit the homogeneity of Black-Scholes. $\frac{\partial f}{\partial x}\frac{\partial x}{\partial s} =$

The lack. Thus we consider a parameterisation with respect to moneyness and time-scaled implied volatility

$$f(S_t/K, \sigma\sqrt{T-t})$$

We further extend this to

- Plot of Price, Delta, Vega, Gammaa vs Strike, Time
- Plot of call surface
- Plot of errors

Stochastic Volatility

We consider the SABR of Hagan

$$dS_t = S_t^{\beta}\sigma_t dW_{1,t} \tag{19}$$
$$d\sigma_t = \alpha\sigma_t dW_{2,t}, dW_{2,t} = \alpha\sigma_t[\rho dW_{1,t} + \sqrt{1-\rho^2}dW_{1,t}^{\top}] \tag{20}$$

for the Heston

$$dS_t = rdt + S_t V_t dW_{1,t} \tag{21}$$
$$dV_t = \kappa(\theta - V_t)dt + \xi\sqrt{V_t}dW_{2,t}, dW_{2,t} \tag{22}$$

American Option

Asian Option

Consider an discrete-time arithmetic average Asian option:

$$(\frac{1}{N}\sum_{i=1}^{n} S_i - K)^+$$

For the continuous time analogue, let $Y_t = \int_0^t S_t du, dY_t = S_t dt$

$$f(t,s,y) = \mathbb{E}[e^{-r(T-t}h(S_T, Y_T)|S_t = s, Y_t = y]$$

Then the corresponding PDE is given by:

$$f_t + rsf_s + sf_y + \frac{\sigma^2}{2}f_{ss} - rf = 0, f(T,s,y) = \frac{1}{T}(y-K)^+$$

The replicating strategy is f_s
Models risk: How does the NN perform on unseen parameter ranges?,

5 Appendix

Activation	$f(x)$	$f'(x)$	$f''(x)$
ReLU	$\max\{x, 0\}$	$1_{x>0}$	0
ELU	$\alpha(e^x - 1)1_{x<0} + x1_{x>0}$	$1_{x>0} + \alpha e^x 1_{x<0}$	$\alpha e^x 1_{x<0}$
Sigmoid	$\frac{1}{1+e^{-x}}$	$\frac{e^{-x}}{(1+e^{-x})^2}$	$\frac{-e^{-x}(1+e^{-x})}{(1+e^{-x})^4}$
SoftPlus	$\log(1 + e^x)$	$\frac{1}{1+e^{-x}}$	$\frac{e^{-x}}{(1+e^{-x})^2}$
Swish	$\frac{x}{1+e^{-x}}$	$\frac{(x-1)e^{-x}}{(1+e^{-x})^2}$	
GeLU	$x\Phi(x)$	$x\phi(x) + \Phi(x)$	$\phi(x) + x\phi''(x)$
tanh	$\frac{e^x - e^{-x}}{e^x + e^{-x}}$		
RBF	$e^{-\frac{x^2}{2}}$	$-xe^{-\frac{x^2}{2}}$	$(x^2 - 1)e^{-\frac{x^2}{2}}$

References

[1] Alexandre Antonov, Michael Konikov, and Vladimir Piterbarg. "Neural networks with asymptotics control". In: *Available at SSRN 3544698* (2020).

[2] Alexandre Antonov and Vladimir Piterbarg. "Alternatives to Deep Neural Networks for Function Approximations in Finance". In: *Available at SSRN 3958331* (2021).

[3] Damiano Brigo et al. "Interpretability in deep learning for finance: a case study for the Heston model". In: *Available at SSRN 3829947* (2021).

[4] Hans Buehler et al. "Deep hedging". In: *Quantitative Finance* 19.8 (2019), pp. 1271–1291.

[5] Peter Carr and Dilip B Madan. "A note on sufficient conditions for no arbitrage". In: *Finance Research Letters* 2.3 (2005), pp. 125–130.

[6] Marc Chataigner, Stéphane Crépey, and Matthew Dixon. "Deep local volatility". In: *Risks* 8.3 (2020), p. 82.

[7] Francois Chollet. *Deep learning with Python*. Simon and Schuster, 2021.

[8] Matthew F Dixon, Igor Halperin, and Paul Bilokon. *Machine learning in Finance*. Vol. 1170. Springer, 2020.

[9] Hans Föllmer and Alexander Schied. "Stochastic finance". In: *Stochastic Finance*. de Gruyter, 2016.

[10] Patryk Gierjatowicz et al. "Robust pricing and hedging via neural SDEs". In: *Available at SSRN 3646241* (2020).

[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[12] Blanka Horvath, Aitor Muguruza, and Mehdi Tomas. "Deep learning volatility". In: *Available at SSRN 3322085* (2019).

[13] Brian Norsk Huge and Antoine Savine. "Differential machine learning". In: *Available at SSRN 3591734* (2020).

[14] James M Hutchinson, Andrew W Lo, and Tomaso Poggio. "A nonparametric approach to pricing and hedging derivative securities via learning networks". In: *The journal of Finance* 49.3 (1994), pp. 851–889.

[15] Francis A Longstaff and Eduardo S Schwartz. "Valuing American options by simulation: a simple least-squares approach". In: *The review of financial studies* 14.1 (2001), pp. 113–147.

[16] Christoph Molnar. *Interpretable machine learning*. Lulu. com, 2020.

[17] Johannes Ruf and Weiguan Wang. "Neural networks for option pricing and hedging: a literature review". In: *arXiv preprint arXiv:1911.05620* (2019).

[18] Valentin Tissot-Daguette. *Projection of Functionals and Fast Pricing of Exotic Options*. 2021. DOI: 10.48550/ARXIV.2111.03713. URL: https://arxiv.org/abs/2111.03713.

[19] Zhe Wang, Nicolas Privault, and Claude Guet. “Deep self-consistent learning of local volatility”. In: *Available at SSRN 3989177* (2021).