Alma Mater Studiorum · Università di Bologna

Scuola di Scienze

Corso di Laurea Magistrale in Matematica

CALIBRATION OF LOCAL-STOCHASTIC VOLATILITY MODELS WITH NEURAL NETWORKS

Relatore:
Chiar.mo Prof.
ANDREA PASCUCCI

Presentata da:
TIZIANO TODESCHI

Sessione Unica
Anno Accademico 2019/2020

Introduzione

Durante gli ultimi vent'anni vari modelli sono stati proposti per migliorare il classico paradigma di Black-Scholes per la valutazione di contratti derivati su azioni. In particolare il modello originale assumeva che la volatilità del sottostante fosse una costante σ. Al contrario, empiricamente si può osservare come la volatilità implicita σ_I, cioè quel valore che inserito nella formula di Black-Scholes permette di replicare il prezzo di mercato, non sia affatto costante, ma dipenda altresì dal prezzo di esercizio K e dalla scadenza T del contratto. Si osserva dunque sul mercato una superficie di volatilità implicita $\sigma_I(K, T)$.

Tra le varie classi di modelli proposti, due filoni di ricerca, in particolare, sono stati ampiamente sviluppati ed utilizzati: i modelli a volatilità locale [12][11] e i modelli a volatilità stocastica [27][25][41] nei quali l'ipotesi originale di BlackScholes di un coefficiente di volatilità costante viene effettivamente rilassata. I modelli a volatilià locale considerano la volatilità come una funzione deterministica del titolo sottostante e del tempo mentre i modelli a volatilità stocastica considerano la volatilità stessa come un processo stocastico. Il primo tipo di modelli permette una buona calibrazione rispetto ai prezzi quotati sul mercato delle opzioni europee. Al contrario il secondo tipo di modelli riesce a riprodurre una dinamica più realistica della volatilità implicita.

Recentemente un nuovo modello è stato proposto, generalizzando i due precedenti: il cosiddetto modello "Local-Stochastic Volatility" [38]. In questo caso la volatilità è data dal prodotto tra una componente deterministica ed una componente stocastica. In questo modo, utilizzando una volatilità ibrida locale-stocastica, è possibile sfruttare i vantaggi di entrambi i modelli base, i quali possono effettivamente essere interpretati come casi particolari di questo nuovo modello generalizzato.

In questa tesi ci focalizziamo sulla calibrazione di un modello a volatilità locale-stocastica (LSV). La calibrazione rappresenta la scelta di *un modello* tra un *insieme di modelli*, in base ai dati storici e correnti del mercato. I modelli LSV hanno attirato molta attenzione grazie alla caratteristica di poter realizzare una calibrazione potenzialmente perfetta allo smile di volatilità del mercato. Gli studi di maggior successo si riferiscono all'utilizzo di metodi Monte Carlo [19][20][9], metodi basati sulle equazioni Fokker-Planck non lineari [38][42] e tecniche di risoluzione di problemi inversi. [40].

Finora la scelta del modello non era stata influenzata solo dalla capacità di replicare le caratteristiche del mercato osservate empiricamente, ma anche dalla trattabilità dal punto di vista computazionale del processo di calibrazione. In questo contesto sta avvenendo un grande cambiamento poiché le tecnologie di apprendimento automatico, o *machine learning*, offrono nuove prospettive sulle prestazioni computazionali per la calibrazione del modello.

Possiamo distinguere tre tipi di approcci che utilizzano tecniche di machine learning per la calibrazione ai dati del mercato. Un primo approccio deriva dal fatto che, avendo risolto il problema inverso molte volte, si può "imparare" direttamente da questo processo la mappa di calibrazione dai dati del mercato ai parametri del modello [24]. Come secondo approccio, è possibile ricavare la funzione che mappa i parametri del modello nei prezzi generati dal modello e dopodichè invertire questa funzione con tecniche di machine learning [31]. Infine, la calibrazione può essere interpretata come la ricerca di un modello che genera i prezzi del mercato in cui possono essere utilizzate tecnologie inerenti le reti generative avversarie, per la prima volta introdotte da Goodfellow nel 2014 [17]. Questo significa sostanzialmente parametrizzare l'insieme dei possibili modelli in modo da rendere possibile l'utilizzo di tecniche di machine learning e l'interpretazione del problema inverso come un addestramento di una rete generativa, la cui qualità verrà valutata da una rete "avversaria". Proponiamo quindi un algoritmo, basandoci sui lavori di C. Cuchiero, W. Khosrawi e J. Teichmann [10] e di S. Ben Hamida e R. Cont [3], che si ispira a questa metodologia e usa come modelli generativi delle cosiddete equazioni differenziali stocastiche neurali (neural SDE), il che significa sostanzialmente che il termine di drift e di volatilità di un processo di Ito governato da una SDE vengono parametrizzate attraverso reti neurali.

Questa tesi è articolata secondo la seguente struttura.

Chapter 1: iniziamo esaminando i principali e più diffusi modelli di option pricing: Black-Scholes, volatilità locale e volatilità stocastica. Analizziamo caratteristiche, pregi e difetti di questi modelli per poi arrivare a presentare la teoria generale sui modelli a volatilità locale-stocastica;

Chapter 2: introduciamo le basi e i principali risultati della teoria del deep learning, definiamo la struttura delle reti neurali artificiali e presentiamo il metodo di backpropagation insieme ai principali algoritmi di ottimizzazione per l'addestramento delle reti neurali;

Chapter 3: descriviamo il celebre metodo Monte Carlo, approfondendone l'utilizzo nell'ambito dell'option pricing, includendo le principali tecniche di riduzione della varianza. Presentiamo anche due possibili strategie di hedging che saranno cruciali nell'algoritmo di calibrazione che proponiamo;

Chapter 4: definiamo la parametrizzazione tramite rete neurale del nostro modello LSV ed evidenziamo il funzionale di calibrazione da minimizzare. Descriviamo l'algoritmo di calibrazione con diverse possibilità di ottimizzazione, quindi mostriamo lo pseudo-codice dell'algoritmo per chiarire l'implementazione del metodo.

Nell'appendice sono presentati alcuni importanti risultati riguardanti i processi stocastici e le equazioni differenziali stocastiche.

Introduction

During the last twenty years several models have been proposed to improve the classic Black-Scholes framework for equity derivatives pricing. In particular, the original model assumed that the volatility of the underlying was a costant σ. On the contrary, it can be empirically observed as the implied volatility σ_I, that is the value that inserted in the Black-Scholes formula allows to replicate the market price, is not at all constant, but also depends on the strike price K and on the expiry T of the contract. A surface of implicit volatility $\sigma_I(K,T)$ is therefore observed on the market.

Among the various classes of models proposed, two main strands of research, in particular, have been widely developed and used: local volatility [12][11] and Stochastic volatility [27][25][41]. Both these approaches relaxed the Black-Scholes hypothesis of a constant volatility. In fact, local volatility models assume volatility to be a deterministic function of the underlying asset and time, whereas Stochastic volatility models consider volatility as a random process itself. While the former models are able to be well calibrated to traded vanilla options, the latter can reproduce a more realistic dynamics of implied volatility.

Recently a new model, generalization of the two previous ones, has been proposed: the "Local-Stochastic Volatility Model" [38]. This model considers volatility as the product between a deterministic and a stochastic term. In this way, using an hybrid local-stochastic volatility, it is possible to take the advantages of both the two basic models, which, in fact, can be considered as special cases of this generalized model.

In this thesis we focus on the calibration of a LSV model. Calibration is the choice of *one* model from a *pool of models*, given current market and historical data. LSV models have attracted, due to their appealing feature of a potentially perfect smile calibration and their econometric properties, a lot of attention from the calibration and implementation point of view. The most successful approaches involve Monte Carlo methods [19][20][9], PDE methods based on nonlinear Fokker-Planck equations [38][42] and inverse problem techniques [40].

So far the model choice was not only driven by the capacity of capturing empirically observed market features well, but also by the computational tractability of the calibration process. This is now undergoing a big change since machine learning technologies offer new perspectives on model calibration.

We can distinguish three kinds of machine learning-inspired approaches for calibration to current market prices. First, having solved the inverse problem already several times, one can learn from this experience (i.e., training data) the calibration map from market data to model parameters directly [24]. Second, one can learn the map from model parameters to model prices and then invert this map possibly with machine learning technology [31]. Third, the calibration problem is considered to be the search for a model which generates given market prices and where additionally technology from generative adversarial networks, first introduced by Goodfellow in 2014 [17], can be used. This means parameterizing the model pool in a way which is accessible for machine learning techniques and interpreting the inverse problem as a training task of a generative network, whose quality is assessed by an adversary. We propose a calibration algorithm, based on works of C. Cuchiero, W. Khosrawi and J. Teichmann [10] and of S. Ben Hamida and R. Cont [3], that pursue this approach and use as generative models so-called neural stochastic differential equations (SDE), which just means to parameterize the drift and volatility of an Ito-SDE by neural networks.

This thesis is articulated according to the following structure.

Chapter 1: we start by reviewing the fundamental option pricing models, Black-Scholes, local volatility and Stochastic volatility models. We analyze the pros and cons of these models and then come to present the general theory about local Stochastic volatility models;

Chapter 2: we introduce foundations and main results of deep learning theory, define the structure of artificial neural networks and present the backpropagation method with the main optimization algorithms for training networks;

Chapter 3: we describe the famous Monte Carlo method, deepening its use in the context of option pricing, including the main variance reduction techniques. We also present possible hedging strategies that will be crucial in the calibration algorithm we propose.

Chapter 4: we specify the parametrization by neural network of our LSV model and then point out the calibration functional to minimize. We describe the calibration algorithm with different optimization possibilities, and then show the pseudo-code algorithm with technical details.

In the appendix there are presented some interesting theoretical results concerning stochastic process and stochastic differential equations which we mention in the thesis.

Contents

Chapter 1

Option pricing models

One of the central problems in modern mathematical finance is derivative pricing. A derivative is a financial contract which value depends on an underlying asset which can be an equity stock, an interest rate or any different financial asset.
An option is the simplest example of a derivative instrument. An option is a contract that gives the right (but not the obligation) to its holder to buy or sell some amount of the underlying asset at a future date, for a prespecified price. Therefore in an option contract we need to specify:

- an underlying asset;
- an exercise price K, the so-called strike price;
- a date T, the so-called maturity.

A *Call* option gives the right to buy, whilst a *Put* option gives the right to sell. An option is called *European* if the right to buy or sell can be exercised only at maturity, and it is called *American* if it can be exercised at any time before maturity. European Put and Call depends only on the value of the underlying at maturity T and are the simplest examples of options, called *Plain Vanilla options*. Other typologies of options are the so called *Exotic options*, which value depends on the trend of the underlying towards the maturity.

Given a call option with strike price K and value of the underlying asset S_t, we say that the option is currently:

- In-the-money (ITM) when $S_t > K$;
- At-the-money (ATM) when $S_t \simeq K$;
- Out-the-money (OTM) when $S_t < K$.

In the first case the option is worth exercising and it is expensive, while in the third case the option is worthless and it is cheap. Of course if we are dealing with a put option the terminology is reversed.

1.1 Black-Scholes model

The well known Black-Scholes model was first introduced in 1973 by Fischer Black, Myron Scholes [5] and Robert Merton [33]. Nowadays it represents an universal accepted framework for derivative pricing in the financial industry.

1.1.1 Hypothesis and results

The original Black-Scholes model assumes the existence of a risk free asset B_t and of an underlying asset S_t, following respectively a deterministic and a geometric Brownian motion dynamics:

$$dB_t = rB_t dt, \tag{1.1}$$

$$dS_t = \mu S_t dt + \sigma S_t dW_t, \tag{1.2}$$

where the deterministic constant μ, σ and r represent respectively the local mean rate of return of the asset, the volatility of the asset and the short rate interest. W_t is a standard Wiener process [A.3].
Let's consider a *simple contingent claim* of the form

$$\chi = \phi\left(S_T\right), \tag{1.3}$$

namely a derivative paying at maturity an amount χ depending only on the value S_T of the underlying itself at maturity. The function ϕ is the so called pay-off at maturity of the derivative contract. Let's further assume that this contingent claim can be traded on a liquid market and that its price $\pi(t) = \pi(t;\phi)$ has the form

$$\pi(t) = F\left(S_t, t\right), \tag{1.4}$$

for some smooth function F. This means that the price of the derivative at subscription time t depends only on the time itself and on the value of the underlying asset S_t at time t.

Theorem 1.1.1 (Black-Scholes equation).
Assuming that the market is specified by (1.1) *and* (1.2)*, we want to price a contingent claim of the form* (1.3)*. Then the only pricing function of the form* (1.4) *which is consistent with the absence of arbitrage is when F is the solution of the following boundary value problem in the domain $[0,T] \times \mathbb{R}^+$.*

$$\frac{\partial F}{\partial t}(s,t) + rs\frac{\partial F}{\partial s}(s,t) + \frac{1}{2}s^2\sigma^2\frac{\partial^2 F}{\partial s^2}(s,t) - rF(s,t) = 0, \tag{1.5}$$

$$F(s,T) = \phi(s). \tag{1.6}$$

This equation is precisely of the form which can be solved using the Feynman-Kac stochastic representation formula [A.3.5], that establishes a link between parabolic partial differential equations (PDEs) and stochastic processes.
The solution is given by

$$F(s,t) = e^{-r(T-t)}\mathbb{E}_{s,t}\left[\phi\left(X_T\right)\right], \tag{1.7}$$

where the process X_u is defined by the dynamics:

$$dX_u = rX_u du + \sigma X_u dW_u. \tag{1.8}$$

The process X_t above has precisely the same form of the price process S_T. The only, but important, change is that whereas S_t has the local rate of return μ, the X_t-process has the short rate of interest r as its local rate of return. This is the so called change of martingale measure which implies the pricing valuation in a risk-neutral world. It is now possible to state the following central result for derivative pricing, well explained by Bjork in [4].

Theorem 1.1.2 (Risk Neutral Valuation).
The arbitrage free price of the claim $\phi(S_T)$ is given by $\pi(t;\phi) = F(t, S_t)$, where F is given by the formula

$$F(s,t) = e^{-r(T-t)}\mathbb{E}^Q_{s,t}\left[\phi\left(S_I\right)\right], \tag{1.9}$$

where the Q-dynamics of S are

$$dS_t = rS_t dt + \sigma S_t dW_t. \tag{1.10}$$

Let's now consider the problem of pricing an european Call option.
Given the strike price K and the maturity T, the payoff function is given by

$$\phi\left(S_T\right) = \max\left(S_T - K, 0\right).$$

After some calculations it is possible to get the following famous result, which is known as Black-Scholes formula for European options.

Proposition 1.1.3 (Black-Scholes formula).
The price of a European call option with strike price K and time of maturity T is given by the formula $\pi(t) = F(S_t, t)$ where

$$F(s,t) = s\mathcal{N}\left[d_1(s,t)\right] - e^{-r(T-t)}K\mathcal{N}\left[d_2(s,t)\right]. \tag{1.11}$$

Here $\mathcal{N}$ denotes the cumulative density function for the normal standard distribution and

$$\begin{aligned} d_1(s,t) &= \frac{1}{\sigma\sqrt{T-t}}\left[\ln\left(\frac{s}{K}\right) + \left(r + \frac{1}{2}\sigma^2\right)(T-t)\right], \\ d_2(s,t) &= d_1(s,t) - \sigma\sqrt{T-t}. \end{aligned}$$

In what follows we will indicate the Black-Scholes formula (1.11) for an European Call option with the notation $C_{BS}(S, t, K, T, r, \sigma)$.

1.1.2 Volatility and model limits

Since it allows a closed form formula for several kind of derivatives, the Black-Scholes is a very appealing framework. However, the original model is not consistent with market prices. In particular, it is unable to correctly reproduce all the vanilla option prices mainly because contracts with different strikes and maturities exhibit different volatilities.
In fact, given all the model parameters and the observed price of an European option it is possible to invert the Black-Scholes formula finding the so-called **implied-volatility** σ_I.

Definition 1.1 (Implied volatility)**.** Let $C^{mkt}(K,T)$ the market price of a european Call option $C^{mkt}(K,T)$ with strike K and maturity T.
The implied volatility $\sigma^{imp}(K,T)$ is the value that satisfy the equation

$$C^{mkt}(S,t,K,T,r) = C^{BS}(S,t,K,T,r,\sigma^{imp}(K,T)). \tag{1.12}$$

For European options under the Black-Scholes model, calculation of the implied volatility seems to be a straightforward exercise since a closed-form presentation exists for the price. However, this closed-form doesn't allow an analytical computation of the implied volatility and it's necessary to solve the nonlinear equation (1.12). However it can be solved easily numerically.
Let $f(\sigma) = C^{BS}(S,t,K,T,r,\sigma)$ and $\nu = C^{mkt}(S,t,K,T,r)$, then the equation becomes

$$f(\sigma) = \nu. \tag{1.13}$$

Since f is monotone increasing and differentiable, the equation has a unique solution and we can apply each variant of the Newton method, which Quarteroni, Sacco and Valeri review in [36].
In particular the classical Newton-method can be used.
Given an initial guess for σ_0, $\forall k > 0$ until $k > k_{max}$:

$$\sigma_{k+1} = \sigma_k - \frac{f(\sigma_k) - \nu}{f'(\sigma_k)}. \tag{1.14}$$

The final value of $\sigma_{k_{max}}$ is a good approximation of the implied volatility $\sigma_I(K,T)$.

The original Black-Scholes model assumes that the volatility is a constant σ across strikes and maturity dates. However it is empirically evident how σ_I depends on the value of the strike and the time to maturity of the option, namely $\sigma_I = \sigma_I(K;T)$. These two effects are, respectively, known as volatility smile or volatility strike structure and volatility term structure of option prices. At any fixed maturity, implied volatility changes with the strike price. In particular almost always in-the-money call options exhibit higher implied volatilities than out-the-money option, while the minimum of implied volatility is usually in the at-the-money region. That's way we talk about the "volatility smile" since the strike structure of implied volatility is usually concave resembling precisely a smile. Concerning the term structure of implied volatility, for any fixed strike, it varies with the maturity. Often options with longer maturity have higher implied volatilities.

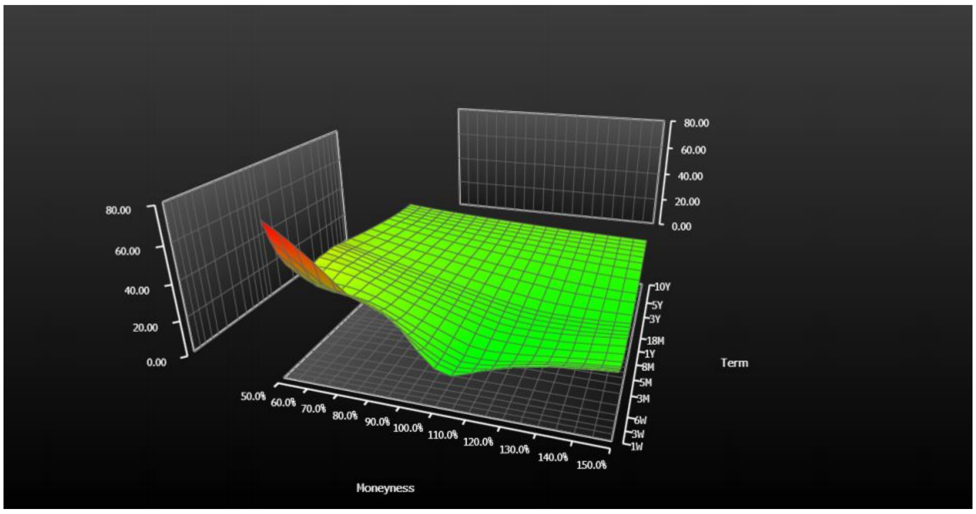

Figure 1.1: Implied volatility surface of the SPX500 index at 1 st August 2012

In order to take into account the empirical evidence of a non constant volatility several models have been proposed during the last twenty years, developing and generalizing the Black-Scholes framework.

- **Local volatility Models (LVM)**
  These models assume that the diffusion coefficient of the underlying asset is no longer a constant value but instead a deterministic function of time and of the underlying asset itself:

$$dS_t = rS_t dt + \sigma_{LV}\left(S_t, t\right) S_t dW_t \tag{1.15}$$

- **Stochastic volatility Models (SVM)**
  In this class of models the volatility itself is considered to be a stochastic process with its own dynamics. Thus, this is a two-factor model, driven by two correlated Wiener processes W_t and Z_t.

$$dS_t = rS_t dt + b\left(V_t\right) S_t dW_t, \tag{1.16}$$

$$dV_t = a\left(V_t, t\right) dt + c\left(V_t, t\right) dZ_t, \tag{1.17}$$

$$dW_t dZ_t = \rho dt. \tag{1.18}$$

In the last ten years they have been widely studied in academic literature as well as used at the equity trading desks of investment banks. We now discuss advantages and disadvantages of them and then introduce the generalization given by the local-Stochastic volatility.

1.2 Local volatility

In 1994 Dupire [12] and Derman and Kani [11] introduced a new model generalizing the Black-Scholes'one. They consider a non constant deterministic volatility $\sigma_{LV}(S,t)$, called local volatility surface, and they assume the following stochastic differential equation for the evolution of the underlying asset.

Definition 1.2 (Local volatility Model - LVM).

$$dS_t = rS_t dt + \sigma_{LV}\left(S_t, t\right) S_t dW_t \tag{1.19}$$

The corresponding pricing equation is a straightforward generalization of the Black-Scholes equation. Thus the price of an European Call option can be computed simply solving the problem below.

Proposition 1.2.1 (Generalized Black-Scholes equation).
Under a local volatility model the price of an European Call option is given by the following generalized Black-Scholes equation:

$$\begin{cases} \dfrac{\partial C}{\partial t} + \dfrac{1}{2}\sigma_{LV}^2(S,t)S^2\dfrac{\partial^2 C}{\partial^2 S} + rS\dfrac{\partial C}{\partial S} - rC = 0 & \text{on } Q = [0,T) \times (0,\infty) \\ C(0,t) = 0 & \forall t \in (0,T) \\ \lim\limits_{S\to\infty} C(S,t) - S + Ke^{-r(T-t)} = 0 & \forall t \in (0,T) \\ C(T,S) = (S-K)^+ & \forall S \in (0,\infty) \end{cases}$$

This model seems to be a simple and straightforward generalization of the original Black-Scholes framework since we are simply considering a non constant, deterministic, diffusion coefficient. However it is not straightforward as well to understand how to extract the surface σ_{LV} from the market.

1.2.1 The Dupire equation

The Black-Scholes backward parabolic equation in the variables (S,t) is the Feynman-Kac representation of the discounted expected value of the final option value. It is possible to find the same option price solving a dual problem, namely a forward parabolic equation in the variables (K,T) known as dual Black-Scholes equation or **Dupire's equation**

Proposition 1.2.2 (Dupire's equation).
The value of a call option as a function of the strike price K and the time to maturity T given the present value of the stock S is given by the following

forward parabolic equation known as Dupire's equation.

$$\begin{cases} \dfrac{\partial C}{\partial T} - \dfrac{1}{2}\sigma_{LV}^2(K,T)K^2\dfrac{\partial^2 C}{\partial^2 K} + rK\dfrac{\partial C}{\partial K} = 0 \quad \text{on } Q = [0,\infty) \times (0,\infty) \\ C(0,t) = 0 \quad \forall t \in (0,T) \\ \lim_{S\to\infty} C(S,t) - S + Ke^{-r(T-t)} = 0 \quad \forall t \in (0,T) \\ C(T,S) = (S-K)^+ \quad \forall S \in (0,\infty) \end{cases}$$

Thanks to the Dupire equation we have accomplished a double result. In fact, on the one hand we have now a very useful, dual equation for derivative pricing in the two variable (K; T).
On the other hand, we have now a formula to evaluate the local volatility $\sigma_{LV}(s,t)$ from option prices, known as **Dupire formula**:

$$\sigma_{LV}^2(K,T) = 2\frac{C_T + rKC_K}{K^2 C_{KK}} \tag{1.20}$$

Assuming a continuum of option prices quoted on the market for every strikes K and time to maturity T thanks to the above formula it is possible to easily evaluate the local volatility surface. Moreover this formula ensures existence and uniqueness of a local volatility surface which reproduces exactly the market prices. Unfortunately it is not possible to observe on the market a continuum of plain vanilla prices. In fact only some options with certain strikes and maturities are actually traded. Therefore it is not possible to use directly (1.20) to evaluate σ_{LV} for every K and T. In particular it is needed to interpolate and extrapolate the Call prices from the market and then to numerically approximate the derivatives. This procedure is rather sensitive to numerical instabilities and errors. Particularly critical is the second derivative $\frac{\partial^2 C}{\partial K}$ at the denominator which stands alone by itself. This derivative can be very small for options deeply in-the-money or out-the-money and then very sensitive to numerical errors. Furthermore this value is multiplied by K^2 resulting in big errors, sometimes even producing negative values and then resulting in negative variance. Because of these drawbacks the Dupire formula, practically speaking, is not very useful.

1.2.2 Properties and CEV model

Local volatility models are the most popular ones with endogenous volatility, due to some main features. First of all, since there is only one source of randomness, all these models are complete. This means that is possible to determine the unique neutral risk price of an option and a hedging strategy can always be found, at least theoretically. Actually the dependence of σ on S_t does not seem to be easily justified from an intuitive point of view. Nevertheless, another important advantage of these models is their high flexibility that make them able to give the theoretical price of an option in accordance (at least approximately) with the implied volatility surface of the market.
We now introduce the most popular example of LV model.

Example 1.1 (CEV Model)**.** The constant elasticity of variance (CEV) model is a particular parametric local volatility model that was introduced in 1975 by Cox [7] . The risk-neutral dynamics are assumed to follow

$$dS_t = rS_t dt + \sigma(t) S_t^{\beta} dW_t, \tag{1.21}$$

where as usual r is the risk-free rate , $\beta \in]0,1[$ and $\sigma(t)$ is a deterministic function of time. Note that the CEV model generalizes BS which is obtained when we set $\beta = 1$ and $\sigma(t) \equiv \sigma$.
The popularity of the CEV model is due to its tractability.
By writing (1.21) as

$$\frac{dS_t}{S_t} = r dt + \sigma(t) S_t^{\beta-1} dW_t, \tag{1.22}$$

we see that there is a negative relationship between price level and instantaneous volatility when $\beta < 1$. The CEV model is therefore able to capture some of the skew that is observed empirically in practice. It is also worth noting that when $\beta < \frac{1}{2}$, there is a strictly positive probability that the CEV process will hit zero.
An important result for the CEV model is given by the following theorem.

Theorem 1.2.3.
The implied volatility generated by the CEV model (1.21) *with* $\beta \in]0,1[$*, is*

approximated by the following formula

$$\sigma_{\text{CEV}}(S_t, T, K) = \frac{\sqrt{V_{t,T}}}{F_t^{1-\beta}} \left(1 + \frac{(1-\beta)(2+\beta)}{6}\left(\frac{F_t - K}{F_t}\right)^2 + \frac{(1-\beta)^2(T-t)V_{t,T}}{24F_t^{2(1-\beta)}}\right) \tag{1.23}$$

where

$$V_{t,T} = \frac{1}{T-t}\int_t^T e^{2r(T-\tau)(1-\beta)}\sigma^2(\tau)d\tau$$

and

$$F_t = \frac{e^{r(T-s)}S_t + K}{2}.$$

As we previously highlighted, the LV models are consistent with the market, since they are able to reproduce the observed market prices. Unfortunately these models have a wrong implied volatility smile dynamics. Rebonato outlines in [37] that the implied volatility smile generated by the LV models tends to become almost flat whereas in the reality the smile persist over time.

1.3 Stochastic volatility

A simple observation of equity markets would make natural to model the volatility itself as a stochastic process. This is precisely the main feature of a stochastic volatility model (SVM). While the standard Black-Scholes model assumes a constant volatility term σ, a SVM considers volatility as a function $b(\cdot)$ of a stochastic process V_t. A first stochastic volatility approach in option pricing was presented in 1991 by E. M. Stein and J. C. Stein [41], but we can trace the root of all modern stochastic volatility models to Heston's 1993 paper [25], which offered a new, closed-form, approach for pricing bond options and foreign-exchange options under stochastic volatility dynamic.

Generally speaking a stochastic volatility model assumes the following dynamics.

Definition 1.3 (Stochastic volatility model).

$$\begin{aligned} dS_t &= \mu S_t dt + b\left(V_t\right) S_t dW_t, \\ dV_t &= a\left(V_t, t\right) dt + c\left(V_t, t\right) dZ_t, \\ dW_t dZ_t &= \rho dt. \end{aligned} \tag{1.24}$$

As usual S_t denotes the underlying asset, t the time, μ the (deterministic) instantaneous drift and W_t,Z_t are two Wiener processes with correlation ρ. Gatheral show in [14], using non-arbitrage arguments, that the price of an European call option under a SVM satisfies the following equation:

$$\begin{aligned} \frac{\partial C}{\partial t} + \frac{1}{2}s^2 b^2(v)\frac{\partial^2 C}{\partial s^2} + \frac{1}{2}c^2(v,t)\frac{\partial^2 C}{\partial v^2} + \rho b(v)c(v,t)s\frac{\partial^2 C}{\partial v \partial s} \\ + rs\frac{\partial C}{\partial s} + (a(v,t) - \lambda c(v,t))\frac{\partial C}{\partial v} - rC = 0. \end{aligned} \tag{1.25}$$

In this equation two new parameters have been introduced, namely r, the usual risk free interest rate, and λ the so called *market price of volatility risk*. While the use of r instead of μ has been already explained previously, describing the Black-Scholes model, some words are needed about λ. The standard BS model assumes only one source of randomness W_t related to one traded asset S_t. In this way it is possible to hedge the risk generated by W_t through S_t. Hence the model is said to be complete, see [4]. On the contrary a SV model assumes two sources of randomness W_t , Z_t and only one traded asset S_t depending on both these sources. In this case we cannot hedge the risk and the model is said to be incomplete. The concept of completeness of the model is strictly related to the existence of an equivalent martingale measure [A.7] and to the Girsanov theorem [A.1.1], that shows it is possible to substitute "arbitrarily" the drift of an Ito process [A.2] by modifying appropriately the considered probability measure and Brownian motion, while keeping unchanged the diffusion coefficient. In fact the second fundamental theorem of option pricing [A.2.2] states that if the model is complete then it exists only one equivalent measure and the price of every derivative is uniquely determined. On the other hand if the model used is incomplete there exist several different martingale measures and then the same derivative has several possible prices depending on λ. Once the value of λ is chosen it is possible to define a risk neutral drift $\tilde{a} = a - \lambda c$ for the process V_t. In this way it is possible to redefine the dynamics for the SV model in the risk-neutral world as follows:

$$\begin{aligned} dS_t &= rS_t dt + b\left(V_t\right) S_t dW_t, \\ dV_t &= \tilde{a}\left(V_t, t\right) dt + c\left(V_t, t\right) dZ_t, \\ dW_t dZ_t &= \rho dt. \end{aligned} \tag{1.26}$$

1.3.1 Heston model

The Heston model was introduced in 1993 by Steven L. Heston [25] and nowadays it is probably the most popular stochastic volatility model. The underlying asset S_t follows the usual log-normal dynamics while the square of the volatility, the variance V_t is a CIR process, first proposed in 1985 by J. C. Cox, J. E. Ingersoll and S. A. Ross [8]:

$$\begin{aligned} dS_t &= rS_t dt + S_t\sqrt{V_t} dW_t \\ dV_t &= \kappa\left(\theta - V_t\right) dt + \eta\sqrt{V_t} dZ_t \\ dW_t dZ_t &= \rho dt \end{aligned} \tag{1.27}$$

The Heston model is characterized by five constant parameters, namely κ , θ , η , ρ and the initial value of the variance V_0. The parameter θ can be thought as the long term variance, κ as the rate of mean reversion and η as the volatility of volatility. As usual ρ represents the instantaneous correlation between the Brownian motions W_t and Z_t. Since we cannot directly observe V_0 as we do for S_0 we need to calibrate also the initial condition of the variance. Thus we consider V_0 as the fifth parameter. In order to use the model we need to calibrate from the market all these five parameters:κ , θ , η , ρ are strictly positive while $\rho \in (-1, 1)$, being a correlation.

The Heston model has several properties which makes it very suitable for equity option pricing. Stochastic variance is mean-reverting, continuous and positive. The model allows a good fit of market implied volatilities and a realistic smile dynamics. However the reason that makes this model so popular and used is probably the fact that it has a semi-closed form solution for plain vanilla options. This enables a fast and computational efficient valuation of European options which becomes critical when calibrating the model to known option prices.

Let's consider the Heston pricing equation:

$$\begin{aligned}\frac{\partial C}{\partial t}+\frac{1}{2}s^2v\frac{\partial^2 C}{\partial s^2}+\frac{1}{2}\eta^2v\frac{\partial^2 C}{\partial v^2}+\rho\eta sv\frac{\partial^2 C}{\partial v\partial s}\\+rs\frac{\partial C}{\partial s}+\kappa[\theta-v]\frac{\partial C}{\partial v}-rC=0,\end{aligned}\tag{1.28}$$

with the proper initial and boundary conditions.

In its original work, Heston looked for a solution similar to the Black-Scholes'one, namely:

$$C\left(S_t,V_t,t,T\right)=S_tP_1-Ke^{-r(T-t)}P_2.\tag{1.29}$$

He managed to show that this is indeed a solution of the equation defined as follows:

$$\begin{aligned}P_j\left(S_t,V_t,t,T\right)&=\frac{1}{2}+\frac{1}{\pi}\int_0^\infty \mathrm{Re}\left(\frac{e^{-i\omega\ln(K)}}{i\omega}f_j\left(S_t,V_t,t,T,\omega\right)\right)d\omega\\ f_j\left(S_t,V_t,t,T,\omega\right)&=e^{C(T-t,\omega)+D(T-t,\omega)V_t+i\omega\ln(S_t)}\\ C(\tau,\omega)&=i\omega r+\frac{\kappa\theta}{\eta^2}\left[\left(b_j-\rho\eta\omega i+d\right)\tau-2\ln\left(\frac{1-ge^{dr}}{1-g}\right)\right]\\ D(\tau,\omega)&=\frac{b_j-\rho\eta\omega i+d}{\eta^2}\left(\frac{1-e^{dr}}{1-ge^{dr}}\right)\\ g&=\frac{b_j-\rho\eta\omega i+d}{b_j-\rho\eta\omega i-d}\\ d&=\sqrt{\left(\rho\eta\omega i-b_j\right)^2-\eta^2\left(2u_j\omega i-\omega^2\right)}\end{aligned}$$

for $j=1,2$, where:

$$u_1=\frac{1}{2},\quad u_2=-\frac{1}{2},\quad b_1=\kappa-\rho\eta,\quad b_2=\kappa.$$

Despite this formula looks quite demanding, it is actually rather explicit, easy and fast to evaluate. The only part that requires some computational effort is the evaluation of the integral along a not bounded interval. However such integration can be performed using standard numerical methods.

1.3.2 SABR model

Another popular model, also used in the modelling of fixed income markets, is the so called SABR (Stochastic Alpha Beta Rho) model proposed and analyzed in 2002 by Hagan, Kumar, Lesniewski and Woodward [21]. The SABR model is the natural extension of the classical CEV model to stochastic volatility: the risk-neutral dynamics of the forward price $F_t = e^{r(T-t)}S_t$ is given by

$$\begin{aligned} dF_t &= V_t F_t^{\beta} dW_t \\ dV_t &= \nu V_t dZ_t \end{aligned} \tag{1.30}$$

where (W, Z) is a Brownian motion with constant correlation ρ. Note that the SABR model generalizes CEV with costant $\sigma(t) \equiv \sigma$, which is obtained when we set $\nu \equiv 0$.
A similar result to which we presented for the CEV model is given by the following approximating formula for the implied volatility in the SABR model:

$$\begin{aligned} \sigma(K, T, F_0, r) = & \frac{V_0}{(F_0K)^{\frac{1-\beta}{2}} \left(1 + \frac{(1-\beta)^2}{24} \log^2\left(\frac{F_0}{K}\right) + \frac{(1-\beta)^4}{1920} \log^4\left(\frac{F_0}{K}\right)\right)} \frac{z}{x(z)} \cdot \\ & \cdot \left(1 + \left(\frac{(1-\beta)^2 V_0^2}{24\,(F_0K)^{1-\beta}} + \frac{\rho\beta\nu V_0}{4\,(F_0K)^{(1-\beta)/2}} + \frac{(2-3\rho^2)\,\nu^2}{24}\right) T\right) \end{aligned}$$

where

$$z = \frac{\nu}{V_0} (F_0K)^{(1-\beta)/2} \log \frac{F_0}{K}$$

and

$$x(z) = \log \frac{\sqrt{1 - 2\varrho z + z^2} + z - \varrho}{1 - \varrho}$$

1.4 LSV Models

Summing up, local volatility models [11] [12] are consistent with the market and can fit almost perfectly the volatility surface but they have a wrong implied volatility smile dynamics as the smile generated tends to become almost flat whereas in the reality the smile persist over time.

Instead, stochastic volatility models [25] [41] account for long term smiles and skew but they cannot give rise to realistic short-term implied volatility patterns.
The two class of models presented seem to have specular proprieties and for this reasons an interesting solution can be given by the mix between them, namely a generalized **Local-Stochastic Volatility Model** which combines the realistic smile dynamics of the SVM with the good fit of market price of the LVM. A general LSV model is given by:

$$\begin{aligned} dS_t &= \mu_1 \left(S_t, t\right) dt + L\left(S_t, t\right) \sigma_1 \left(S_t, V_t, t\right) dW_t \\ dV_t &= \mu_2 \left(V_t, t\right) dt + \sigma_2 \left(V_t, t\right) dZ_t \\ dW_t dZ_t &= \rho dt \end{aligned} \tag{1.31}$$

where the coefficient $\sigma_1 \left(S_t, V_t, t\right)$ incorporate both local and stochastic volatility. The diffusion coefficient of S_t is controlled by a function $L(S_t; t)$, called **leverage function**, determined on market information and that has the role to weight local and stochastic volatilities.

The academic research about this new kind of model is rather recent. The first contribution was given by Jex, Henderson and Wang in 1999 [28] who first suggested a local-stochastic volatility dynamics and proposed a two-dimensional trinomial tree for the calibration. Developing the idea of mixing the three standard models (LVM, SVM and JDM) Lipton suggested in 2002 a universal volatility model [30] which actually contains as a particular case the LSVM. Some years later other theoretical contributions were given by Alexander and Nogueira [2] and moreover by Ren, Madan and Qian in 2007 [38] who suggested a procedure to calibrate a LSVM. Their work has been further developed in two different strands of research. The first is based on the work of Labordère in 2009 [23] and Guyon and Labordère in 2012 [19], who exploited the so-called Markovian projections method. The other strand of research has been developed by Abergel and Tachet in 2010 [1] and by Engelmann, Koster and Oeltz in 2012 [13].

Example 1.2 (Dupire-Heston model)**.** The Dupire-Heston model is a generalization of stochastic volatility Heston model in which it also adds the leverage function. The spot price dynamics S_t and stochastic variance V_t under risk neutral measure in the Dupire-Heston model is:

$$\begin{aligned} dS_t &= rS_t dt + L\left(S_t, t\right)\sqrt{V_t} S_t dW_t, \\ dV_t &= k\left(\theta - V_t\right)dt + \eta\sqrt{V_t} dZ_t, \\ dW_t dZ_t &= \rho dt \end{aligned} \tag{1.32}$$

Example 1.3 (LSV SABR model)**.** Similarly, the extension to Stochastic-local volatility of SABR model is given by the following forward price and volatility dynamics:

$$\begin{aligned} dF_t &= V_t L\left(F_t, t\right) F_t^{\beta} dW_t \\ dV_t &= \nu V_t dZ_t \\ dW_t dZ_t &= \rho dt \end{aligned} \tag{1.33}$$

The construction of a model which is able to fit the vanilla prices and the observed volatility smile and, at the same time, that it is able to show a realistic dynamics is of primary importance for the pricing of path-dependent exotic options.
We want to focus on calibration of LSV models, which is still an intricate task, both from a theoretical as well as practical point of view. For the rest of this thesis, we will assume a zero risk interest rate $r = 0$ and will consider only the one-dimensional case, but the setup easily translate to more general case with a straight forward extension.

1.4.1 Calibration of LSV models

LS and LV models can be calibrated in a independent and simultaneous way to market data to get the value of model parameters. Then, it is possible to obtain LSV volatility surface. Whatever process V_t and local volatility is, the standard steps to calibrate the model are:

- Calibration of local volatility σ_{loc};

- Calibration of stochastic volatility V_t;
- Calibration of Leverage function.

It is common to use the Dupire volatility, seen in 1.2.1, as local volatility, with all the difficulties we highlighted in previous section.
In the models we treat, the discounted price process $(S_t)_{t\geq 0}$ of an asset satisfies

$$dS_t = S_t L\left(S_t, t\right) V_t dW_t \tag{1.34}$$

where the volatility process $(V_t)_{t\geq 0}$ can be of Heston type or SABR type. However, V_t can also be very general and could for instance be chosen as rough volatility model. We point out that this model correspond to the choice of $\sigma_1\left(S_t, V_t, t\right) = S_t V_t$ in (1.31).

The leverage function L is the crucial part in this model. It allows in principle to perfectly calibrate the implied volatility surface seen on the market. To achieve this goal L must satisfy

$$L^2(t,s) = \frac{\sigma^2_{\mathrm{Dup}}(t,s)}{\mathbb{E}\left[V_t^2 \mid S_t = s\right]} \tag{1.35}$$

where σ_{Dup} denotes Dupire's local volatility function.
This is a important result that follow directly from Gyöngy theorem [A.3.6], an important result that establishes a link between local volatility and stochastic volatility models.
Please note that (1.35) is an implicit equation for L as it is needed for the computation of $\mathbb{E}\left[V_t^2 \mid S_t = s\right]$. This in turn means that the SDE for the price process $(S_t)_{t\geq 0}$ is actually a McKean Vlasov SDE, since the law of (S_t, V_t) enters in the equation.
Different approaches have been presented to solve in efficient ways this problem, such as Monte Carlo methods, PDE methods based on non-linear Fokker-Planck equations and inverse problem techniques. Between these, the Monte Carlo approach with particle approximation method [18] for the McKean-Vlasov SDE works impressively well, as very few paths have to be simulated

in order to achieve very accurate calibration results.

In this thesis we propose an alternative, fully data-driven approach circumventing in particular the interpolation of the volatility surface, being necessary in several other approaches in order to compute Dupire's local volatility. This means that we only take the available discrete data into account and do not generate a continuous surface interpolating between the given market option prices. Indeed, we just learn or *train* the leverage function L to generate the available market option prices accurately and to do this we use a **deep learning** approach, namely we will parametrize the leverage function with a feed-forward **neural network**, able to be trained to market data in order to calibrate the model.
Therefore we need to introduce in the next chapter deep learning and artificial neural networks.

Chapter 2

Introduction to Deep Learning

In this chapter we want to introduce **deep learning theory** and most important results concerning artificial neural networks and their training. For a more in-depth study of deep learning theory, we refer to the exhaustive book *Deep Learning* by I. Goodfellow, Y. Bengio and A. Courvill [16].

Modern deep learning provides a very powerful framework for supervised learning. By adding more layers and more units within a layer, a deep network can represent functions of increasing complexity. Most tasks that consist of mapping an input vector to an output vector, and that are easy for a person to do rapidly, can be accomplished via deep learning, given sufficiently large models and sufficiently large datasets of labeled training examples.
We will now introduce two core concepts in deep learning, namely artificial neural networks and stochastic gradient descent, where the latter is a widely used optimization method for optimization problems involving the first. In standard machine learning terminology, the optimization problem is usually referred to as **"training"** and in the sequel we will use both terminologies interchangeably.

2.1 Artificial Neural Networks

Feedforward neural networks, also often called **deep feedforward networks**, or **multilayer perceptrons** (MLPs), are the quintessential deep learning models. The goal of a feedforward network is to approximate some function f^*. For example, for a classifier, $y = f^*(x)$ maps an input x to a category y. A feedforward network defines a mapping $y = f(x; \theta)$ and learns the value of the parameters θ that result in the best function approximation. These models are called **feedforward** because information flows through the function being evaluated from $\boldsymbol{x}$, through the intermediate computations used to define f, and finally to the output $\boldsymbol{y}$.

Feedforward networks are of extreme importance to machine learning and deep learning practitioners. They form the basis of many important commercial applications. For example, the convolutional networks used for object recognition from photos are a specialized kind of feedforward network. Feedforward networks are a conceptual stepping stone on the path to recurrent networks, which power many natural language applications.

2.1.1 Network architecture

Feedforward neural networks are called **networks** because they are typically represented by composing together many different functions. The model is associated with a directed acyclic graph describing how the functions are composed together. For example, we might have three functions $f^{(1)}$, $f^{(2)}$, and $f^{(3)}$ connected in a chain, to form $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$. These chain structures are the most commonly used structures of neural networks. In this case, $f^{(1)}$ is called the **first layer** of the network, $f^{(2)}$ is called the **second layer**, and so on. The overall length of the chain gives the **depth** of the model. It is from this terminology that the name "deep learning" arises.

The final layer of a feedforward network is called the **output layer**. During neural network training, we drive $f(\mathbf{x})$ to match $f^*(\mathbf{x})$. The training data provides us with noisy, approximate examples of $f^*(x)$ evaluated at different training points. Each example $\boldsymbol{x}$ is accompanied by a label $y \approx f^*(\mathbf{x})$. The training examples specify directly what the output layer must do at each point $\boldsymbol{x}$; it must produce a value that is close to y. The behavior of the other layers is not directly specified by the training data. The learning algorithm must decide how to use those layers to produce the desired output, but the training data does not say what each individual layer should do. Instead, the learning algorithm must decide how to use these layers to best implement an approximation of f^*. Because the training data does not show the desired output for each of these layers, these layers are called hidden layers.

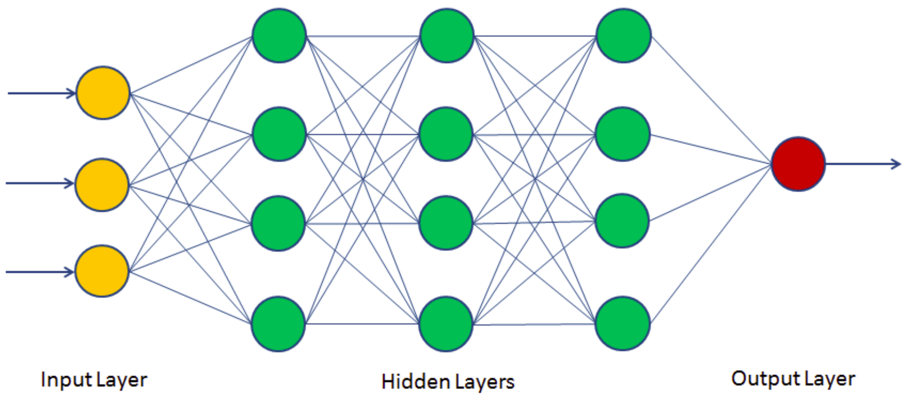

Figure 2.1: Example of the structure of a simple neural network

Finally, these networks are called **neural** because they are loosely inspired by neuroscience. Each hidden layer of the network is typically vector-valued. The dimensionality of these hidden layers determines the **width** of the model. Each element of the vector may be interpreted as playing a role analogous to a neuron. Rather than thinking of the layer as representing a single vector-to-vector function, we can also think of the layer as consisting of many **units** that act in parallel, each representing a vector-to-scalar function. Each unit resembles a neuron in the sense that it receives input from many other units and computes its own activation value. The idea of using many layers of vector-valued representation is drawn from neuroscience. The choice of the functions $f^{(i)}(\mathbf{x})$ used to compute these representations is also loosely guided by neuroscientific observations about the functions that biological neurons compute.

Let's now define in a rigorous mathematical way a feedforward neural network:

Definition 2.1 (Feedforward Neural Network)**.** Let $L, N_0, N_1, \dots N_L \in \mathbb{N}$, $\sigma : \mathbb{R} \to \mathbb{R}$ and for any $\ell \in \{1, \dots, L\}$, let $w_\ell : \mathbb{R}^{N_{\ell-1}} \to \mathbb{R}^{N_\ell}$, $\boldsymbol{x} \mapsto \boldsymbol{W}^{(\ell)}\boldsymbol{x}+\boldsymbol{b}^{(\ell)}$ be an affine function with $\boldsymbol{W}^{(\ell)} \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ and $\boldsymbol{b}^{(\ell)} \in \mathbb{R}^{N_\ell}$ and additionally $\boldsymbol{b}_L = 0$. A function $\mathbb{R}^{N_0} \to \mathbb{R}^{N_L}$ defined as

$$F = w_L \circ F_{L-1} \circ \cdots \circ F_1, \qquad \text{with } F_\ell = \sigma \circ w_\ell \qquad \text{for } \ell \in \{1, \dots, L-1\}$$

is called a **feed forward neural network**. L denotes the number of layers and $N_1, \dots, N_{L-1}$ denote the dimensions of the hidden layers and N_0 and N_L the dimension of the input and output layers. The function σ is called **activation function** and it is applied componentwise. The importance of this function will be highlighted in next section.

2.1.2 Activation function

An **activation function** is a function that is added into an artificial neural network in order to help the network learn complex patterns in the data. When comparing with a neuron-based model that is in our brains, the activation function is at the end deciding what is to be fired to the next neuron. That is exactly what an activation function does in an ANN as well. It takes in the output signal from the previous cell and converts it into some form that can be taken as input to the next cell. The comparison can be summarized in the figure below.

The presence of a **non-linear** activation function is important due to his capability to add non-linearity into the neural network and, more over, because it help in keeping the value of the output from the neuron restricted to a certain limit. This is important because input into the activation function is $\boldsymbol{W}^{(\ell)}\boldsymbol{x} + \boldsymbol{b}^{(\ell)}$ and its value is not bounded. Without the activation

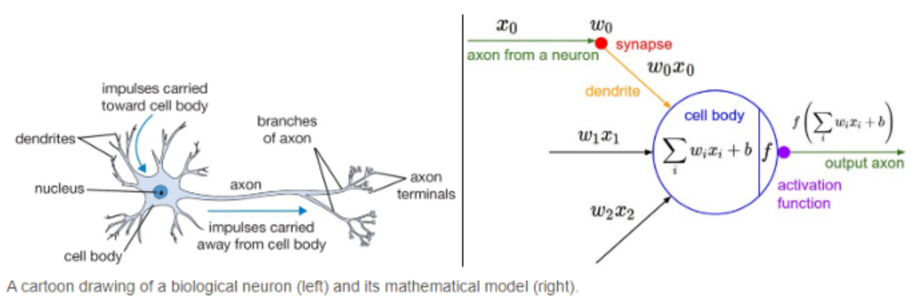

Figure 2.2: Comparison between human neuron and artificial neuron

function it could go very high in magnitude, especially in case of very deep neural networks that have millions of parameters, and therefore cause computational issues. As we will see in the next sections, usually neural networks are trained through gradient based algorithms. In order to be able to utilize these algorithms, there are some desirable features for the choice of activation function:

- **Differentiable:** In order to calculate gradient, layers in the model need to be differentiable or at least differentiable in parts.
- **Zero-Centered:** Output of the activation function should be symmetrical at zero so that the gradients do not shift to a particular direction.
- **Computational Inexpensive:** Activation functions are applied after every layer and need to be calculated millions of times in deep networks.

We present some of most popular activation function:

- **Sigmoid:** defined as $\sigma(x) = \frac{1}{1+e^{-x}}$, is important only for historical reasons and never used in real models due to his computational expense and its not being zero-centered.
- **Tanh:** compared to sigmoid, it solves the zero-center problem.
- **ReLu** (**Rectified Linear Unit**): is defined as $f(x) = max(0, x)$ and is widely used, especially with Convolutional Neural networks. It is

easy to compute and has other desirable features, with just one issue of not being zero centred.

We conclude this section presenting an important result know as **universal approximation theorem**, achieved in 1991 by Hornik [26]. For its formulation we denote the set of all feed forward neural networks with activation function σ, input dimension N_0 and output dimension N_L by $\mathcal{NN}^{\sigma}_{\infty,N_0,N_L}$.

Theorem 2.1.1 (Universal approximation theorem).
Suppose σ is bounded and non-constant. Then the following statements hold:

1. *For any finite measure μ on $\left(\mathbb{R}^{N_0}, \mathcal{B}\left(\mathbb{R}^{N_0}\right)\right)$ and $1 \leq p < \infty$, the set $\mathcal{NN}^{\sigma}_{\infty,N_0,1}$ is dense in $L^p\left(\mathbb{R}^{N_0}, \mathcal{B}\left(\mathbb{R}^{N_0}\right), \mu\right)$;*
2. *If in addition $\sigma \in C(\mathbb{R},\mathbb{R})$, then $\mathcal{NN}^{\sigma}_{\infty,N_0,1}$ is dense in $C\left(\mathbb{R}^{N_0}, \mathbb{R}\right)$ for the topology of uniform convergence on compact sets.*

Since each component of an $\mathbb{R}^{N_L}$ valued neural network is an $\mathbb{R}$-valued neural network, this result easily generalizes to $\mathcal{NN}^{\sigma}_{\infty,N_0,N_L}$ with $N_L > 1$.

For the rest of this thesis we will denote by $\mathcal{NN}_{N_0,N_L}$ the set of all neural networks in $\mathcal{NN}^{\sigma}_{\infty,N_0,N_L}$ with a fixed architecture, i.e. a fixed number of layers L, fixed input and output dimensions N_ℓ for each hidden layer $\ell \in \{1,\ldots,L-1\}$ and a fixed activation function σ. This set can be described by

$$\mathcal{NN}_{N_0,N_L} = \{F(\cdot \mid \theta) \mid F \text{ feed forward neural network and } \theta \in \Theta\}$$

with parameter space $\Theta \subset \mathbb{R}^q$ for some $q \in \mathbb{N}$ and $\boldsymbol{\theta} \in \Theta$ corresponding to the entries of the matrices $\boldsymbol{W}^{(\ell)}$ and the vector $\boldsymbol{b}^{(\ell)}$ for $\ell \in \{1,\ldots,L\}$

2.2 ANN training

Once we have chosen the network architecture (number of hidden layers and number of neurons for each layer), we must adapt the optimal weights $\boldsymbol{W}^{(\ell)}$ and $\boldsymbol{b}^{(\ell)}$ for $\ell \in \{1,\ldots,L\}$ by training the learning system.

2.2.1 Backpropagation

In this section we introduce the most widely used method for neural network training, the **backpropagation algorithm**. For a thorough study of the subject we refer to the 1992 paper of R. Hecht-Nielsen [22].

Applying an optimization algorithm like Gradient, Conjugate Gradient or a Quasi-Newton method to Neural Networks it can be quite difficult, especially when the network is very deep (it owns many hidden layers). This algorithm was built ad hoc for Neural Networks and allow to calculate, using the **chain rule**, derivatives of cost function with respect to weights of the network, in order to use a **gradient based** algorithm to minimize the cost function and find a local minimum.

We consider a Deep Neural Network with n input and m output. Let

$$\left\{(\boldsymbol{x}_1, \boldsymbol{y}_1), \dots, (\boldsymbol{x}_p, \boldsymbol{y}_p)\right\}$$

be the training set. This consists of p ordered pairs of vectors in $\mathbb{R}^n \times \mathbb{R}^m$. We denote by $\hat{y}_h$ with $h = 1, \dots, p$ the set of network outcomes with respect to the training set elements. The aim is to minimize the cost function

$$C = \sum_{h=1}^{p} \|\boldsymbol{y}_h - \hat{\boldsymbol{y}}_h\|^2 .$$

Other choices can be made for the cost function, depending on the situation. For the sake of simplicity, we explain the algorithm fixing as activation function the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$ presented in the previous section and a Neural Network with only one hidden layer of dipension q, but the method can be naturally extended to more general cases.
At first, synaptic weights of the network are randomly chosen.
Then, the algorithm consist of 4 steps:

1. **Feedforward computation:**
   The input vector $\boldsymbol{x} = (x_i)_{1\le i\le n}$ is presented to the network. The vectors $\hat{\boldsymbol{y}}^{(1)} = (\hat{y}_j^{(1)})_{1\le j\le q}$ and $\hat{\boldsymbol{y}}^{(2)} = (\hat{y}_k^{(2)})_{1\le k\le m}$ are, respectively, the output

vector produced by the first layer and the output vector produced by the second layer. Namely, we have

$$\begin{aligned}\hat{\boldsymbol{y}}^{(2)} &= \boldsymbol{W}^{(2)}\hat{\boldsymbol{y}}^{(1)} \\ \hat{\boldsymbol{y}}^{(1)} &= \sigma(\hat{\boldsymbol{z}}^{(1)}) \\ \hat{\boldsymbol{z}}^{(1)} &= \boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}\end{aligned} \tag{2.1}$$

so that the complete structure is given by

$$\hat{\boldsymbol{y}}^{(2)} = \boldsymbol{W}^{(2)}\left(\sigma\left(\boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}\right)\right) \tag{2.2}$$

We write it also component wise:

$$\hat{y}_k^{(2)} = \sum_{j=1}^{q} W_{kj}^{(2)}\left(\sigma\left(\sum_{i=1}^{n} W_{ji}^{(1)}x_i + b_i^{(1)}\right)\right) \tag{2.3}$$

In this step $\hat{\boldsymbol{y}}^{\mathbf{(1)}}$ and $\hat{\boldsymbol{y}}^{\mathbf{(2)}}$ are computed and stored and also the evaluated derivatives of the activation functions are also stored in each neuron.

2. **Backpropagation to the output layer:**
We are looking for the value of partial derivatives $\partial C/\partial W_{kj}^{(2)}$. The weight $W_{kj}^{(2)}$ is the synaptic connection between the j-th neuron of the hidden layer and the k-th neuron of the output layer. We apply now the chain rule for the computation of the derivative and we get

$$\frac{\partial C}{\partial W_{kj}^{(2)}} = \sum_{k=1}^{m} \frac{\partial C}{\partial \hat{y}_k^{(2)}}\frac{\partial \hat{y}_k^{(2)}}{\partial W_{kj}^{(2)}} = \sum_{k=1}^{m}\left(\hat{y}_k^{(2)} - y_k\right)\hat{y}_j^{(1)}$$

3. **Backpropagation to the hidden layer:** Now we are looking for the value of partial derivatives $\partial C/\partial W_{ij}^{(1)}$ and $\partial C/\partial b_i^{(1)}$. By the definition of derivative of sigmoid function, we have $\sigma'(x) = \sigma(x)(1-\sigma(x))$. Analogously to the previous step, by the chain rule we get

$$\begin{aligned}\frac{\partial C}{\partial W_{ji}^{(1)}} &= \sum_{k=1}^{m} \frac{\partial C}{\partial \hat{y}_k^{(2)}}\frac{\partial \hat{y}_k^{(2)}}{\partial \hat{y}_j^{(1)}}\frac{\partial \hat{y}_j^{(1)}}{\partial \hat{z}_j^{(1)}}\frac{\partial \hat{z}_j^{(1)}}{\partial W_{ji}^{(1)}} \\ &= \sum_{k=1}^{m}\left(\hat{y}_k^{(2)} - y_k\right)W_{kj}^{(2)}\hat{y}_j^{(1)}\left(1-\hat{y}_j^{(1)}\right)x_i\end{aligned}$$

and

$$\begin{aligned}
\frac{\partial C}{\partial b_i^{(1)}} &= \sum_{k=1}^{m} \frac{\partial C}{\partial \hat{y}_k^{(2)}} \sum_{j=1}^{q} \frac{\partial \hat{y}_k^{(2)}}{\partial \hat{y}_j^{(1)}} \frac{\partial \hat{y}_j^{(1)}}{\partial \hat{z}_j^{(1)}} \frac{\partial \hat{z}_j^{(1)}}{\partial b_i^{(1)}} \\
&= \sum_{k=1}^{m} \left(\hat{y}_k^{(2)} - y_k\right) \sum_{j=1}^{q} W_{kj}^{(2)} \hat{y}_j^{(1)} \left(1 - \hat{y}_j^{(1)}\right)
\end{aligned}$$

4. **Weights update:** After computing all partial derivatives, the network weights must b updated, making use of some gradient based optimization algorithm. There are a lot of variants of the classical gradient descent, so in the next section we present some of the most popular algorithm used in neural networks training. by using a Gradient Descent method.

2.2.2 Optimization algorithms

The backpropagation algorithm works in collaboration with an optimization method for minimizing the cost function. At first, synaptic weights of the network are randomly chosen. Then the backpropagation allow us to calculate the partial derivatives with respect to the weights, that we call

$$\begin{aligned}
\delta_{kj}^{(2)} &= \frac{\partial C}{\partial W_{ji}^{(2)}}; \\
\delta_{ji}^{(1)} &= \frac{\partial C}{\partial W_{ji}^{(1)}}; \\
\delta_{i}^{(1)} &= \frac{\partial C}{\partial b_i^{(1)}}.
\end{aligned}$$

The central idea of **gradient descent** method is that the weights correction takes place along the negative direction of the gradient of the cost function. The weights update is given by:

$$\begin{aligned}
\Delta W_{kj}^{(2)} &= -\gamma \delta_{kj}^{(2)}, &&\text{for } k = 1, \ldots, m; \quad j = 1, \ldots, q \\
\Delta W_{ji}^{(1)} &= -\gamma \delta_{ji}^{(1)}, &&\text{for } j = 1, \ldots, q; \quad i = 1, \ldots, n \\
\Delta b_{i}^{(1)} &= -\gamma \delta_{i}^{(1)}, &&\text{for } i = 1, \ldots, n
\end{aligned}$$

The step length γ is also called the **learning rate**. A correct choice of this parameter is fundamental for the convergence of the algorithm. The learning rate can be fixed or can be adaptive, in order to improve the algorithm's performances. An important features of gradient descent is the convergence to a local minimum, making necessary a good first guess to ensure a good performance of the algorithm. As in optimization theory, the **early stopping** methods are concerned with the problem of choosing a time to stop the process. Here are some examples:

- The sequence is interrupted when the last value is in the neighborhood of a local minimum. In other words, when the Euclidean norm of the function gradient is less than a fixed threshold value.
- When the error variation percentage between two consecutive epochs is sufficiently small.
- The learning algorithm is stopped when it reaches the maximum number of iterations.

We now present some popular variants of gradient descent method, overviewed in 2017 by Sebastian Ruder [39].

- **Stochastic Gradient Descent (SGD):** it is one of the most used methods in practical applications. It is very simple to implement and the computational cost is quite low. The idea is to not use the whole dataset to calculate the gradient in each point, but instead to consider only a subset (mini-batch) of size r. It follows that

  $$\begin{aligned} g &= \frac{1}{r}\sum_{i=1}^{r} \nabla_w C\left(f\left(x^{(i)};\theta\right), y^{(i)}\right) \\ \Delta\theta &= -\gamma g \end{aligned}$$

  where θ represent the network weights and f the network function. The objective function is the loss function C which is the difference between estimated and true values for a sample of data. The learning rate is heuristically fixed at 0.01. We observe that, if the step length is too big ($\gamma >> 0.01$), then the method may not be converge. Instead, a learning rate that is too small ($\gamma << 0.01$) leads to slow convergence.

- **Momentum:** SGD has trouble descending ravines, i.e. areas where the surface curves much more steeply in one dimension than in another, that are common around local minima. In this scenario, SGD oscillates across the slopes of the ravine while only making hesitant progress along the bottom towards the local minimum. The momentum method accelerates SGD in the relevant direction and dampens oscillations. The method uses the momentum α, which depends on previous iterations. Let g_t be the gradient of the objective function at iteration t.

$$v_{t+1} = \alpha v_t - \gamma g_t$$
$$\theta_{t+1} = \theta_t + v_{t+1}$$

  Usually, α is equal to 0.5 or 0.9 .

- **RMSProp:** The **Root Mean Square Propagation** method is an adaptive algorithm. Hence, the learning rate γ is adapted for each of the parameters. It provides good performance in practice. The running average is calculated in terms of mean squared,

$$E\left[g^2\right]_t = \eta E\left[g^2\right]_{t-1} + (1-\eta)\langle g, g\rangle$$

  where $\eta \in [0,1]$ is the exponential decaying factor (forgetting factor). Usually, $\eta = 0.9$. Intuitively, the choice of η defines how the previous iteration memory is important in the running average computation. The weights update is given by

$$\Delta w_t = -\frac{\gamma g_t}{\sqrt{E\left[g^2\right]_t + \epsilon}}$$

  We observe that the root square to the denominator indicates the mean square (RMS, root mean square). In this case the learning rate γ is dynamically controlled by the root mean square of the gradient norm. It has been added to the denominator the factor ϵ, in order to prevent it from tending to 0.

- **Adam:** The **Adaptive Moment Estimation** method, proposed in 2017 by D. P. Kingma and J. Ba [29], is the most popular today and it can be seen as a combination of RMSProp and Momentum method. Adam uses the running average of the objective function gradient and its second momentum. The parameters update follows the below scheme:

$$M_{t+1} = \beta_1 M_t + (1 - \beta_1)\, g_t$$
$$v_{t+1} = \beta_2 v_t + (1 - \beta_2)\, \langle g_t, g_t \rangle$$

and the bias correction

$$\hat{M} = \frac{M_{t+1}}{1 - (\beta_1)^{t+1}}$$

$$\hat{v} = \frac{v_{t+1}}{1 - (\beta_2)^{t+1}}$$

The weight correction is

$$w_{t+1} = w_t - \gamma \frac{\hat{M}}{\sqrt{\hat{v} + \epsilon}}$$

The term ϵ is used to ensure numerical stability. The parameters β_1 and β_2 are used to control the exponential decay of the gradient and its second momentum. Usually we set $\epsilon = 10^{-8}, \beta_1 = 0.9$ and $\beta_2 = 0.999$

Chapter 3

Pricing and Hedging techniques

In this chapter we focus on pricing and hedging techniques that will be necessary to explain to detail the algorithm of calibration of a LSV model. We choose to follow Pascucci [34] for Monte Carlo method introduction and Glasserman [15] for variance reduction techniques.

3.1 Monte Carlo method

The **Monte Carlo** method is a simple technique of numerical approximation of the mean of a random variable X. It is used in many circumstances in mathematical finance and in particular in the pricing problem. More generally, the Monte Carlo method allows approximating the value of an integral numerically: indeed we recall that, if $Y \sim \text{Unif}_{[0,1]}$ is uniformly distributed on [0,1] and $X = f(Y)$, then we have

$$\mathbb{E}[X] = \int_0^1 f(x)dx$$

The Monte Carlo method is based on the **strong law of large numbers**[A.1.4], which states that the average of the results obtained from a large number of trials should be close to the expected value and will tend to become closer to the expected value as more trials are performed: if (X_n) is a sequence of integrable i.i.d. random variables and such that $E[X_1] = \mathbb{E}[X]$, then

$$\lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} X_k = \mathbb{E}[X]$$

Consequently, if we are able to draw samples $\bar{X}_1, \dots, \bar{X}_n$ from X in an independent way, then the mean

$$\frac{1}{n}\sum_{k=1}^{n} \bar{X}_k$$

gives an a.s. approximation of $\mathbb{E}[X]$. In order to analyze some of the main features of this technique, we consider the problem of numerical approximation of the following integral over the unitary cube in $\mathbb{R}^d$:

$$\int_{[0,1]^d} f(x)dx$$

The most natural way to approximate the value of the integral consists in considering a discretization by Riemann sums: for fixed $n \in \mathbb{N}$, on $[0,1]^d$ we build a grid of points with coordinates of the form $\frac{k}{n}$, $k = 0, \dots, n$. Then we rewrite the integral in the form

$$\int_{[0,1]^d} f(x)dx = \sum_{k_1=0}^{n-1} \cdots \sum_{k_d=0}^{n-1} \int_{\frac{k_1}{n}}^{\frac{k_1+1}{n}} \cdots \int_{\frac{k_d}{n}}^{\frac{k_d+1}{n}} f\left(x_1, \dots, x_d\right) dx_1 \cdots dx_d$$

and we approximate the right-hand side by

$$\begin{aligned}\sum_{k_1=0}^{n-1} \cdots \sum_{k_d=0}^{n-1} \int_{\frac{k_1}{n}}^{\frac{k_1+1}{n}} \cdots \int_{\frac{k_d}{n}}^{\frac{k_d+1}{n}} f\left(\frac{k_1}{n}, \dots, \frac{k_d}{n}\right) dx_1 \cdots dx_d \\ = \frac{1}{n^d}\sum_{k_1=0}^{n-1} \cdots \sum_{k_d=0}^{n-1} f\left(\frac{k_1}{n}, \dots, \frac{k_d}{n}\right) =: S_n(f)\end{aligned} \tag{3.1}$$

If f is Lipschitz continuous, with Lipschitz constant L, then

$$\left|\int_{[0,1]^d} f(x)dx - S_n(f)\right| \le \frac{L}{n}$$

Further, if $f \in C^q\left([0,1]^d\right)$, we can easily obtain an n^{-q}-order scheme, by substituting $f\left(\frac{k_1}{n}, \dots, \frac{k_d}{n}\right)$ in (3.1) with the q -th order Taylor expansion of f with initial point $\left(\frac{k_1}{n}, \dots, \frac{k_d}{n}\right)$

In principle, this kind of approximation gives better results than the Monte Carlo method. However, the convergence of the scheme depends heavily on the regularity of f. For example, the measurable function $f(x) = 1_{[0,1]^d \setminus \mathbb{Q}^d}$ has integral equal to 1, but $S_n(f) = 0$ for every $n \in \mathbb{N}$.
Moreover, the computation of the approximation term $S_n(f)$ necessary to get an error of the order of $\frac{1}{n}$ involves the valuation of f in n^d points; so the number of points increases exponentially with the dimension of the problem. It follows that, in practice, only if d is small enough it is possible to implement the method in an effective way.

Now we consider the approximation with the Monte Carlo method. If (Y_n) is a sequence of i.i.d random variables with uniform distribution on $[0,1]^d$, we have

$$\int_{[0,1]^d} f(x)dx = E\left[f\left(Y_1\right)\right] = \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} f\left(Y_k\right) \tag{3.2}$$

We observe that, in order for the integral to converge, it suffices that f is integrable on $[0,1]^d$ and no further regularity assumption is required.

Concerning the computational complexity, we can give a first estimate of the error of the Monte Carlo method directly by the Markov inequality [A.1.3], that gives an upper bound for the probability that a non-negative function of a random variable is greater than or equal to some positive constant. We consider a sequence of real i.i.d. random variables (X_n) with $\mu = E\left[X_1\right]$ and $\sigma^2 = \operatorname{var}\left(X_1\right)$ finite. Furthermore, we set

$$M_n = \frac{1}{n}\sum_{k=1}^{n} X_k$$

By Markov's inequality, for every $\varepsilon > 0$, we have

$$P\left(\left|M_n - \mu\right| \geq \varepsilon\right) \leq \frac{\operatorname{var}\left(M_n\right)}{\varepsilon^2} =$$

(by the independence)

$$= \frac{n \operatorname{var}\left(\frac{X_1}{n}\right)}{\varepsilon^2} = \frac{\sigma^2}{n\varepsilon^2}$$

that can be rewritten in a more appealing way as follows:

$$P\left(|M_n - \mu| \leq \varepsilon\right) \geq p, \qquad \text{where } p := 1 - \frac{\sigma^2}{n\varepsilon^2} \tag{3.3}$$

First of all we note that, being the technique based on the generation of random numbers, the result and the error of the Monte Carlo method are random variables. Formula (3.3) gives an estimate of the error in terms of the number of samples n, the maximum approximation error ε and p, that is the minimum probability that the approximated value M_n belongs to the confidence interval $[\mu - \varepsilon, \mu + \varepsilon]$. According to (3.3), for fixed $n \in \mathbb{N}$ and $p \in]0, 1[$, the maximum approximation error of the Monte Carlo method is

$$\varepsilon = \frac{\sigma}{\sqrt{n(1-p)}} \tag{3.4}$$

Therefore, the error is of the order of $\frac{1}{\sqrt{n}}$ regardless of the dimension of the problem. Summing up, if the dimension is low and some suitable regularity assumptions are verified, then it is not difficult to implement deterministic algorithms performing better than Monte Carlo. However, when the dimension of the problem increases, these deterministic algorithms become burdensome and the Monte Carlo method is, for now, the only viable alternative.

We also observe that, by (3.4), the standard deviation σ is directly proportional to the approximation error: as a matter of fact, from a computational point of view σ is a crucial parameter which influences significantly the efficiency of the approximation. Typically σ is not known; nevertheless it is possible to use the random numbers that we have generated to construct an estimator of σ :

$$\sigma_n^2 := \frac{1}{n-1}\sum_{k=1}^{n}\left(X_k - \mu_n\right)^2, \quad \mu_n := \frac{1}{n}\sum_{k=1}^{n} X_k$$

Usually, in order to improve the effectiveness of the Monte Carlo method, **variance-reduction** methods are used. These techniques, elementary in some cases, employ the specific features of the problem to reduce the value of σ_n and consequently increase the speed of convergence. We will present two of most pupular techniques in section

3.1.1 Simulation

The first step to approximate $\mathbb{E}[X]$ by the Monte Carlo method consists in generating n independent realizations of the random variable X : this poses some practical problems.

First of all n must be large enough and so the generation of the simulations cannot be made by hand (for example, by tossing a coin): therefore we must use the power of a computer to perform the computation. This rather obvious remark introduces the first serious problem: a computer can generate "random" values only by using deterministic algorithms. So, in order to implement the Monte Carlo method, actually we have at our disposal only "pseudo-random" numbers, i.e. numbers that have the same statistical properties as the actual random values but, when the number of times we simulate increases, are not generated in a really independent way. This translates into an additional error that cannot be easily estimated in the approximated result. Therefore it should always be borne in mind the fact that the quality of the random-number generator influences the numerical result significantly. After shedding some light on this first matter, for the vast majority of the well-known distributions, and in particular for the Normal standard distribution, it is not difficult to find a pseudo-random number generator. Having this at our disposal, pricing of a European option with payoff F is indeed an easy task. For example, in the Black-Scholes model, where the final price of the underlying asset is

$$S_T = S_0 \exp\left(\sigma W_T + \left(r - \frac{\sigma^2}{2}\right) T\right)$$

the procedure is as follows:

(A.1) We draw n independent samples $Z_1, \dots, Z_n$, from the standard Normal distribution;

(A.2) We consider the corresponding realizations of the final value of the underlying asset

$$\bar{S}_T^{(k)} = S_0 \exp\left(\sigma\sqrt{T}\bar{Z}_k + \left(r - \frac{\sigma^2}{2}\right)T\right)$$

(A.3) We compute the approximation of the price of the derivative

$$\frac{e^{-rT}}{n}\sum_{k=1}^{n} F\left(\bar{S}_T^{(k)}\right) \approx e^{-rT} E\left[F\left(S_T\right)\right]$$

Since the vast majority of models used in financial engineering for option pricing doesn't allow analytical form for the distribution of the payoff random variables, we need to simulate the whole **trajectories** of the underlying. More over, in this way it's easy to price exotic options, whose value depends on the trajectory, with Monte Carlo methods. To do this, we now present the **Euler scheme** and the higher-order **Milstein scheme**.

- **Euler Scheme:** Let consider a local-volatility model in which the dynamics of the underlying asset under the EMM is given by

$$dS_t = rS_t dt + \sigma\left(t, S_t\right) dW_t \tag{3.5}$$

In this case the distribution of the final price S_T is not known explicitly. We discretize the equation (3.5) obtaining

$$\bar{S}_{t_i} = \bar{S}_{t_{i-1}}\left(1 + r\left(t_i - t_{i-1}\right)\right) + \sigma\left(t_{i-1}, \bar{S}_{t_{i-1}}\right)\left(W_{t_i} - W_{t_{i-1}}\right) \tag{3.6}$$

We remark that random variable $W_{t_i} - W_{t_{i-1}}$ has normal distribution with 0 mean and $t_i - t_{i-1}$ variance. Therefore, the procedure to obtain some realizations of S_T is as follows:

(B.1) We produce nm independent realizations $Z_{k,i}$, for $k = 1, \ldots, n$ and $i = 1, \ldots, m$, of the Normal standard distribution $\mathcal{N}_{0,1}$

(B.2) Using the iterative formula

$$\bar{S}_{t_i}^{(k)} = \bar{S}_{t_{i-1}}^{(k)}\left(1 + r\left(t_i - t_{i-1}\right)\right) + \sigma\left(t_{i-1}, \bar{S}_{t_{i-1}}^{(k)}\right)\sqrt{t_i - t_{i-1}}\bar{Z}_{k,i}$$

we determine the corresponding realizations of the final value of the underlying asset $\bar{S}_T^{(1)}, \ldots, \bar{S}_T^{(n)}$

(B.3) we compute the approximation of the price of the derivative as in (A.3).

- **Milstein Scheme:** Analogously to the deterministic case, it is possible to introduce higher-order schemes for the discretization of stochastic equations. One of the simplest is the Milstein scheme, which is similar to Euler scheme with a first-order approximation of the diffusion term with respect to the variable x :

$$\int_{t_{i-1}}^{t_i} \sigma\left(t, S_t\right) dW_t \sim \int_{t_{i-1}}^{t_i} \left(\sigma\left(t_{i-1}, S_{t_{i-1}}\right) + \partial_x \sigma\left(t_{i-1}, X_{t_{i-1}}\right)\left(W_t - W_{t_{i-1}}\right)\right) dW_t$$

By simple computation we get

$$\int_{t_{i-1}}^{t_i} \left(W_t - W_{t_{i-1}}\right) dW_t = \frac{\left(W_{t_i} - W_{t_{i-1}}\right)^2 - \left(t_i - t_{i-1}\right)}{2}$$

Then, putting $\delta = t_i - t_{i-1}$ and denoting a standard Normal random variable by Z, we get the natural extension of the iterative scheme in (B.2)

$$\bar{S}_{t_i} = \bar{S}_{t_{i-1}} + \mu\left(t_{i-1}, \bar{S}_{t_{i-1}}\right)\delta + \sigma\left(t_{i-1}, \bar{S}_{t_{i-1}}\right)\sqrt{\delta} Z + \partial_x \sigma\left(t_{i-1}, \bar{S}_{t_{i-1}}\right)\frac{\delta\left(Z^2 - 1\right)}{2}$$

By way of example, for the discretization of a geometric Brownian motion

$$dS_t = \mu S_t dt + \sigma S_t dW_t$$

we have

$$S_{t_i} = S_{t_{i-1}}\left(1 + \delta\left(\mu + \frac{\sigma^2}{2}\left(Z^2 - 1\right)\right) + \sigma\sqrt{\delta} Z\right)$$

3.2 Variance reduction

We present the two simplest and popular methods for increasing the efficiency of Monte Carlo simulation by reducing the variance of simulation estimates. These methods draw on two broad strategies for reducing variance: taking advantage of tractable features of a model to adjust or correct simulation outputs, and reducing the variability in simulation inputs. We discuss antithetic variates and control variates.

3.2.1 Antithetic variates

The method of antithetic variates attempts to reduce variance by introducing negative dependence between pairs of replications. The method can take various forms; the most broadly applicable is based on the observation that if U is uniformly distributed over $[0,1]$, then $1-U$ is too. Hence, if we generate a path using as inputs $U_1,\dots,U_n$, we can generate a second path using $1-U_1,\dots,1-U_n$ without changing the law of the simulated process. The variables U_i and $1-U_i$ form an antithetic pair in the sense that a large value of one is accompanied by a small value of the other. This suggests that an unusually large or small output computed from the first path may be balanced by the value computed from the antithetic path, resulting in a reduction in variance.

These observations extend to other distributions through the inverse transform method: $F^{-1}(U)$ and $F^{-1}(1-U)$ both have distribution F but are antithetic to each other because F^{-1} is monotone. For a distribution symmetric about the origin, $F^{-1}(1-u)$ and $F^{-1}(u)$ have the same magnitudes but opposite signs. In particular, in a simulation driven by independent standard normal random variables, antithetic variates can be implemented by pairing a sequence $Z_1,Z_2,\dots$ of i.i.d. $N(0,1)$ variables with the sequence $-Z_1,-Z_2,\dots$ of i.i.d. $N(0,1)$ variables, whether or not they are sampled through the inverse transform method.

To analyze this approach more precisely, suppose our objective is to estimate an expectation $\mathrm{E}[Y]$ and that using some implementation of antithetic sampling produces a sequence of pairs of observations $\left(Y_1,\tilde{Y}_1\right),\left(Y_2,\tilde{Y}_2\right),\dots$ (Y_n,Y_n). The key features of the antithetic variates method are the following:

- the pairs $\left(Y_1,\tilde{Y}_1\right),\left(Y_2,\tilde{Y}_2\right),\dots,\left(Y_n,\tilde{Y}_n\right)$ are i.i.d.;
- for each i, Y_i and $\tilde{Y}_i$ have the same distribution, though ordinarily they are not independent.

We use Y generically to indicate a random variable with the common distribution of the Y_i and $\tilde{Y}_i$

The antithetic variates estimator is simply the average of all $2n$ observations,

$$\hat{Y}_{\text{AV}} = \frac{1}{2n}\left(\sum_{i=1}^{n} Y_i + \sum_{i=1}^{n} \tilde{Y}_i\right) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_i + \tilde{Y}_i}{2}\right) \tag{3.7}$$

The rightmost representation in (3.7) makes it evident that $\hat{Y}_{\text{AV}}$ is the sample mean of the n independent observations

$$\left(\frac{Y_1 + \tilde{Y}_1}{2}\right), \left(\frac{Y_2 + \tilde{Y}_2}{2}\right), \ldots, \left(\frac{Y_n + \tilde{Y}_n}{2}\right)$$

The central limit theorem therefore applies and gives

$$\frac{\hat{Y}_{\text{AV}} - \text{E}[Y]}{\sigma_{\text{AV}}/\sqrt{n}} \Rightarrow N(0,1)$$

with

$$\sigma_{\text{AV}}^2 = \text{Var}\left[\frac{Y_i + \tilde{Y}_i}{2}\right]$$

Under what conditions is an antithetic variates estimator to be preferred to an ordinary Monte Carlo estimator based on independent replications? To make this comparison, we assume that the computational effort required to generate a pair $\left(Y_i, \tilde{Y}_i\right)$ is approximately twice the effort required to generate Y_i. Under this assumption, the effort required to compute $\hat{Y}_{\text{AV}}$ is approximately that required to compute the sample mean of $2n$ independent replications, and it is therefore meaningful to compare the variances of these two estimators. Using antithetics reduces variance if

$$\text{Var}\left[\hat{Y}_{\text{AV}}\right] < \text{Var}\left[\frac{1}{2n}\sum_{i=1}^{2n} Y_i\right]$$

i.e., if

$$\text{Var}\left[Y_i + \tilde{Y}_i\right] < 2\,\text{Var}\left[Y_i\right]$$

The variance on the left can be written as

$$\begin{aligned}\text{Var}\left[Y_i + \tilde{Y}_i\right] &= \text{Var}\left[Y_i\right] + \text{Var}\left[\tilde{Y}_i\right] + 2\,\text{Cov}\left[Y_i, \tilde{Y}_i\right] \\ &= 2\,\text{Var}\left[Y_i\right] + 2\,\text{Cov}\left[Y_i, \tilde{Y}_i\right]\end{aligned}$$

using the fact that Y_i and $\bar{Y}_i$ have the same variance if they have the same distribution. Thus, the condition for antithetic sampling to reduce variance becomes

$$\operatorname{Cov}\left[Y_i, \tilde{Y}_i\right] < 0$$

Put succinctly, this condition requires that negative dependence in the inputs (whether U and $1-U$ or Z and $-Z$) produce negative correlation between the outputs of paired replications. A simple sufficient condition ensuring this is monotonicity of the mapping from inputs to outputs defined by a simulation algorithm.

3.2.2 Control variates

The method of control variates is among the most effective and broadly applicable techniques for improving the efficiency of Monte Carlo simulation. It exploits information about the errors in estimates of known quantities to reduce the error in an estimate of an unknown quantity. To describe the method, we let $Y_1, \ldots, Y_n$ be outputs from n replications of a simulation. For example, Y_i could be the discounted payoff of a derivative security on the i-th simulated path. Suppose that the Y_i are independent and identically distributed and that our objective is to estimate $\mathrm{E}\left[Y_i\right]$. The usual estimator is the sample mean $\bar{Y} = \left(Y_1 + \cdots + Y_n\right)/n$. This estimator is unbiased and converges almost surely as $n \to \infty$.

Suppose, now, that on each replication we calculate another output X_i along with Y_i. Suppose that the pairs $\left(X_i, Y_i\right), i = 1, \ldots, n$, are i.i.d. and that the expectation $\mathrm{E}[X]$ of the X_i is known. (We use (X, Y) to denote a generic pair of random variables with the same distribution as each $\left(X_i, Y_i\right)$) Then for any fixed b we can calculate

$$Y_i(b) = Y_i - b\left(X_i - \mathrm{E}[X]\right)$$

from the i-th replication and then compute the sample mean

$$\bar{Y}(b) = \bar{Y} - b(\bar{X} - \mathrm{E}[X]) = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - b\left(X_i - \mathrm{E}[X]\right)\right) \tag{3.8}$$

This is a control variate estimator; the observed error $\bar{X} - \mathrm{E}[X]$ serves as a control in estimating $\mathrm{E}[Y]$.

As an estimator of $\mathrm{E}[Y]$, the control variate estimator (3.8) is unbiased because

$$\mathrm{E}[\bar{Y}(b)] = \mathrm{E}[\bar{Y} - b(\bar{X} - \mathrm{E}[X])] = \mathrm{E}[Y] = \mathrm{E}[Y]$$

and it is consistent because, almost surely,

$$\begin{aligned}\lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} Y_i(b) &= \lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} \left(Y_i - b\left(X_i - \mathrm{E}[X]\right)\right) \\ &= \mathrm{E}[Y - b(X - \mathrm{E}[X])] \\ &= \mathrm{E}[Y]\end{aligned}$$

Each $Y_i(b)$ has variance

$$\begin{aligned}\operatorname{Var}\left[Y_i(b)\right] &= \operatorname{Var}\left[Y_i - b\left(X_i - \mathrm{E}[X]\right)\right] \\ &= \sigma_Y^2 - 2b\sigma_X\sigma_Y\rho_{XY} + b^2\sigma_X^2 =: \sigma^2(b)\end{aligned} \tag{3.9}$$

where $\sigma_X^2 = \operatorname{Var}[X], \sigma_Y^2 = \operatorname{Var}[Y]$, and ρ_{XY} is the correlation between X and Y. The control variate estimator $\bar{Y}(b)$ has variance $\sigma^2(b)/n$ and the ordinary sample mean $\bar{Y}$ (which corresponds to $b = 0$) has variance σ_Y^2/n. Hence, the control variate estimator has smaller variance than the standard estimator if $b^2\sigma_X < 2b\sigma_Y\rho_XY$ The optimal coefficient b^* minimizes the variance (3.9) and is given by

$$b^* = \frac{\sigma_Y}{\sigma_X}\rho_{XY} = \frac{\operatorname{Cov}[X,Y]}{\operatorname{Var}[X]}$$

Substituting this value in (3.9) and simplifying, we find that the ratio of the variance of the optimally controlled estimator to that of the uncontrolled estimator is

$$\frac{\operatorname{Var}\left[Y - b^*(X - \mathrm{E}[X])\right]}{\operatorname{Var}[\bar{Y}]} = 1 - \rho_{XY}^2 \tag{3.10}$$

A few observations follow from this expression:

- With the optimal coefficient b^*, the effectiveness of a control variate, as measured by the variance reduction ratio (3.10) is determined by the strength of the correlation between the quantity of interest Y and the control X. The sign of the correlation is irrelevant because it is absorbed in b^*;

- If the computational effort per replication is roughly the same with and without a control variate, then (3.10) measures the computational speed-up resulting from the use of a control. More precisely, the number of replications of the Y_i required to achieve the same variance as n replications of the control variate estimator is $n/\left(1-\rho_{XY}^2\right)$;
- The variance reduction factor $1/\left(1-\rho_{XY}^2\right)$ increases very sharply as $|\rho_{XY}|$ approaches 1 and, accordingly, it drops off quickly as $|\rho_{XY}|$ decreases away from 1. For example, whereas a correlation of 0.95 produces a ten-fold speedup, a correlation of 0.90 yields only a five-fold speed-up; at $|\rho_{XY}| = 0.70$ the speed-up drops to about a factor of two. This suggests that a rather high degree of correlation is needed for a control variate to yield substantial benefits.

These remarks and equation (3.10) apply if the optimal coefficient b^* is known. In practice, if $\mathrm{E}[Y]$ is unknown it is unlikely that σ_Y or ρ_{XY} would be known. However, we may still get most of the benefit of a control variate using an estimate of b^*. For example, replacing the population parameters in (3.9) with their sample counterparts yields the estimate

$$\hat{b}_n = \frac{\sum_{i=1}^{n}\left(X_i-\bar{X}\right)\left(Y_i-\bar{Y}\right)}{\sum_{i=1}^{n}\left(X_i-\bar{X}\right)^2}$$

Dividing numerator and denominator by n and applying the of large numbers shows that $\hat{b}_n \to b^*$ almost surely. This suggests using the estimator $\bar{Y}\left(\hat{b}_n\right)$, the sample mean of $Y_i\left(\hat{b}_n\right) = Y_i - \hat{b}_n\left(X_i - \mathrm{E}[X]\right), i = 1, \ldots, n$.

3.3 Hedging

Hedging is a technique that consist in finding a strategy that replicate the value of a derivative financial instrument. Usually, the objective is to mitigate the market risk deriving from the payoff randomness of the derivative. For our purposes instead, a hedging strategy is necessary in combination with the control variate method presented in the previous section.
We now present the most popular method, called Delta Hedge, that comes from the Black Scholes framework, and also a recent proposal that concern the parametrization of the hedging strategy via neural networks.

3.3.1 Black Scholes Delta Hedging

We present an alternative ways to obtain the Black-Scholes equation (1.5). The following approaches is heuristic; its good point is that it is intuitive, while its flaw is that it is not completely rigorous. Furthermore it assume the no-arbitrage principle as a starting point, rather than a result.

Let us consider the point of view of a bank that sells an option and wants to determine a hedging strategy by investing in the underlying asset. Let us consider a portfolio consisting of a certain amount of the risky asset S_t and of a short position on a derivative with payoff $F(S_T)$ whose price, at the time t, is denoted by $f(t, S_t)$. The value of the portfolio is then given by

$$V(t, S_t) = \alpha_t S_t - f(t, S_t)$$

In order to determine α_t, we want to render V neutral with respect to the variation of S_t, or, in other terms, V immune to the variation of the price of the underlying asset by imposing the condition

$$\partial_s V(t, s) = 0$$

By the equality $V(t, s) = \alpha_t s - f(t, s)$, we get

$$\alpha_t = \partial_s f(t, s) \tag{3.11}$$

and this is commonly known as the **Delta hedging** strategy. By the self-financing condition and applying Ito Lemma (A.3.4), we have

$$\begin{aligned} dV(t,S_t) &= \alpha_t dS_t - df(t,S_t) \\ &= \left((\alpha_t - \partial_s f)\,\mu S_t - \partial_t f - \frac{\sigma^2 S_t^2}{2}\partial_{ss} f \right) dt + (\alpha_t - \partial_s f)\,\sigma S_t dW_t \end{aligned}$$

Therefore the choice (3.11) wipes out the riskiness of V, represented by the term in dW_t, and cancels out also the term containing the return μ of the underlying asset. Summing up we get

$$dV(t,S_t) = -\left(\partial_t f + \frac{\sigma^2 S_t^2}{2}\partial_{ss} f\right) dt \tag{3.12}$$

Now since the dynamics of V is deterministic, by the no-arbitrage principle V must have the same return of the non-risky asset:

$$dV(t,S_t) = rV(t,S_t)\,dt = r\left(S_t \partial_s f - f\right) dt \tag{3.13}$$

so, equating formulas (3.12) and (3.13) we obtain again the Black-Scholes equation.

We now present the rigorous theorem that provide us the exact self-financing hedging strategy in the Black Scholes framework.

Theorem 3.3.1.
The Black-Scholes market model is complete and arbitrage free, this meaning that every European derivative $F(S_T)$, with F verifying opportune hypothesis, is replicable in a unique way. Indeed there exists a unique strategy $h = (\alpha_t, \beta_t) \in \mathcal{A}$ replicating $F(S_T)$, that is given by

$$\alpha_t = \partial_s f(t,S_t), \quad \beta_t = e^{-rt}\left(f(t,S_t) - S_t \partial_s f(t,S_t)\right) \tag{3.14}$$

where f is the lower bounded solution of the Cauchy problem

$$\frac{\sigma^2 s^2}{2}\partial_{ss} f + rs\partial_s f + \partial_t f = rf, \quad in \quad \left[0,T\right[\times \mathbb{R}^+ \tag{3.15}$$

$$f(T,s) = F(s), \qquad s \in \mathbb{R}^+ \tag{3.16}$$

By definition, $f(t,S_t) = V_t^{(\alpha,\beta)}$ is the arbitrage price of $F(S_T)$

From a theoretical point of view the Delta-hedging strategy (3.11) guarantees a perfect replication of the payoff. So there would be no need to further study the hedging problem. However, in practice the Black-Scholes model poses some problems: first of all, the strategy (3.14) requires a continuous rebalancing of the portfolio, and this is not always possible or convenient, for example because of transition costs. Secondly, the Black-Scholes model is commonly considered too simple to describe the market realistically: the main issue lies in the hypothesis of constant volatility that appears to be definitely too strong if compared with actual data (see section 1.1.2).

The good point of the Black-Scholes model is that it yields explicit formulas for plain vanilla options. Furthermore, even though it has been severely criticized, it is still the reference model. At a first glance this might seem paradoxical but, as we are going to explain, it is not totally groundless.

Robustness of the model

Following Pascucci [34], we now show how the assumption of Black Scholes framework and related delta hedging technique can work well, even if the actual dynamic of the underlying differ from BS.

We assume the Black-Scholes dynamics for the underlying asset

$$dS_t = \mu S_t dt + \sigma S_t dW_t \tag{3.17}$$

where μ, σ are constant parameters and we denote by r the short-term rate. Then the price $f(t, S_t)$ of an option with payoff $F(S_T)$ is given by the solution of the Cauchy problem

$$\frac{\sigma^2 s^2}{2}\partial_{ss} f + rs\partial_s f + \partial_t f = rf, \qquad \text{in } \left[0, T\right[\times \mathbb{R}^+ \tag{3.18}$$

$$f(T, s) = F(s), \qquad s \in \mathbb{R}^+ \tag{3.19}$$

Moreover

$$f(t, S_t) = \alpha_t S_t + \beta_t B_t$$

is the value of the Delta-hedging strategy given by $\alpha_t = \partial_s f(t, S_t)$ and $\beta_t = f(t, S_t) - S_t \partial_s f(t, S_t)$. Let us suppose now that the actual dynamics of the underlying asset is different from (3.17) and are described by an Itô process of the form

$$d\bar{S}_t = \mu_t \bar{S}_t dt + \sigma_t \bar{S}_t dW_t \tag{3.20}$$

with $\mu_t \in \mathbb{L}^1_{\text{loc}}$ and $\sigma_t \in \mathbb{L}^2_{\text{loc.}}$ On the basis of the final condition (3.19), the Delta-hedging strategy replicates the payoff $F\left(\bar{S}_T\right)$ on any trajectory of the underlying asset. However the fact that the actual dynamics (3.20) is different from the Black-Scholes ones causes the loss of the self-financing property: in practice, this means that hedging has a different cost (possibly greater) with respect to the Black-Scholes price $f\left(0, \bar{S}_0\right)$. Indeed we have

$$df\left(t, \bar{S}_t\right) = \partial_s f d\bar{S}_t + \left(\partial_t f + \frac{\sigma_t^2 \bar{S}_t^2}{2} \partial_{ss} f\right) dt =$$

(by (3.18))

$$\begin{aligned} &= \partial_s f d\bar{S}_t + \left(rf - r\bar{S}_t \partial_s f - \frac{\left(\sigma^2 - \sigma_t^2\right) \bar{S}_t^2}{2} \partial_{ss} f\right) dt \\ &= \partial_s f d\bar{S}_t + \left(f - \bar{S}_t \partial_s f\right) dB_t - \frac{\left(\sigma^2 - \sigma_t^2\right) \bar{S}_t^2}{2} \partial_{ss} f dt. \end{aligned} \tag{3.21}$$

More explicitly we have the following integral expression of the payoff

$$F\left(\bar{S}_T\right) = f\left(T, \bar{S}_T\right) = I_1 + I_2 + I_3$$

where

$$I_1 = f\left(0, \bar{S}_0\right)$$

is the Black-Scholes price,

$$I_2 = \int_0^T \partial_s f\left(t, \bar{S}_t\right) d\bar{S}_t + \int_0^T \left(f\left(t, \bar{S}_t\right) - \bar{S}_t \partial_s f\left(t, \bar{S}_t\right)\right) dB_t$$

is the gain of the Delta-hedging strategy,

$$I_3 = -\frac{1}{2} \int_0^T \left(\sigma^2 - \sigma_t^2\right) \bar{S}_t^2 \partial_{ss} f\left(t, \bar{S}_t\right) dt$$

is a correction term due to the erroneous specification of the model for the underlying asset. Clearly $I_3 = 0$ if $\sigma = \sigma_t$ and only in that case the strategy is self-financing.

We remark that I_3 depends only on the misspecification of the volatility term and not on the drift. More precisely I_3, which also represents the replication error of the Delta-hedging strategy, depends on the Vega which measures the convexity of the Black-Scholes price as a function of the price of the underlying asset. In particular the error is small if $\partial_{\mathrm{as}} f$ is small. Furthermore, if the price is convex, $\partial_{ss} f \geq 0$, as in the case of Call and Put options, then the Black-Scholes strategy (whose final value is $I_1 + I_2$) super-replicates the derivative for any dynamics of the underlying asset as long as we choose the volatility sufficiently large, $\sigma \geq \sigma_t$, since in this case $I_3 \leq 0$

In this sense the Black-Scholes model is robust and, if used with all due precautions, can be effectively employed to hedge derivatives.

3.3.2 Deep Hedging

We want now to present a use of neural networks in hedging. In particular, when the number of hedging instruments becomes higher, one can learn the hedging strategy by parametrizing it via neural networks. The idea of using a neural network as a hedging strategy is the basis of the work of H. Bühler, L. Gonon, J. Teichmann and B. Wood [6], from which we take inspiration to provide an introduction to the method.

Let the payoff be a function of the terminal values of the hedging instruments, i.e. $C = g\left(Z_T\right)$. Then in Markov models it makes sense to specify the hedging strategy via a function

$$h : \mathbb{R}^{+} \times \mathbb{R}^{r} \to \mathbb{R}^{r}, \quad h_t = h(t, z)$$

which in turn will correspond to an artificial neural network

$$(t, z) \to h(t, z, \delta) \in \mathcal{NN}_{r+1,r}$$

with weights denoted by δ in some parameter space Δ. Following the approach in [6], an optimal hedge for the claim C with given market price π^{mkt} can be computed via

$$\inf_{\delta \in \Delta} \mathbb{E}\left[u\left(-C + \pi^{mkt} + \left(h\left(\cdot, Z_{\cdot} | \delta\right) \bullet Z_{\cdot}\right)_T\right)\right]$$

for some convex loss function $u : \mathbb{R} \to \mathbb{R}^+$. Recall that $(h \bullet Z)_T$ denotes the stochastic integral with respect to Z at time T. If $u(x) = x^2$, which is often used in practice, this then corresponds to a quadratic hedging criterion.

To tackle this optimization problem, we can apply stochastic gradient descent. Indeed, the stochastic objective function $Q(\delta)(\omega)$ is given by

$$Q(\delta)(\omega) = u\left(-C(\omega) + \pi^{mkt} + \left(h\left(\cdot, Z_{\cdot}|\delta\right)(\omega) \bullet Z_{\cdot}(\omega)\right)_T\right)$$

Chapter 4

LSV model calibration

We have presented in the previous sections the option pricing framework and the importance of a general class of model like local stochastic volatility models. Furthermore we have discussed about the technical necessary instrument to our scope, that is showing a calibration method for the LSV SABR model. We will now make a precise formalization of the considered calibration problem and the method we apply.
The calibration algorithm we proposed is based on works of C. Cuchiero, W. Khosrawi and J. Teichmann [10] and of S. Ben Hamida and R. Cont [3].

4.1 Leverage function as neural network

Consider the LSV model

$$\begin{aligned} dS_t &= V_t L\left(S_t, t\right) S_t dW_t \\ dV_t &= \nu V_t dZ_t \\ dW_t dZ_t &= \rho dt \end{aligned} \tag{4.1}$$

defined on $\left(\Omega, \left(\mathcal{F}_t\right)_{t\geq 0}, \mathcal{F}, \mathbb{Q}\right)$, some filtered probability space, where $\mathbb{Q}$ is a risk neutral measure. We assume the stochastic process V to be fixed. This can for instance be achieved by first calibrating the pure stochastic volatility model with $L \equiv 1$ (e.g. SABR) and by fixing the corresponding parameters.

Our main goal is to determine the leverage function L in perfect accordance with market data. Due to the universal approximation properties outlined in Section 2.1.2 (Theorem 2.1.1), we choose to parametrize L via neural networks. More precisely, let $0 = T_0 < T_1 \cdots < T_n = T$ denote the maturities of the available European call options to which we aim to calibrate the LSV model. We then specify the leverage function $L(t,s)$ via a family of neural networks, i.e.

$$L(t,s) = 1 + F_i(s) \quad t \in [T_{i-1}, T_i), \quad i \in \{1, \ldots, n\} \tag{4.2}$$

where $F_i \in \mathcal{NN}_{1,1}$. We denote the parameters of F_i by θ_i and the corresponding parameter space by Θ_i. For each maturity T_i, we assume to have J_i options with strikes K_{ij}, $j \in \{1, \ldots, J_i\}$.
The calibration functional for the i -th maturity is then of the form

$$\operatorname{argmin}_{\theta_i \in \Theta_i} \sum_{j=1}^{J_i} w_{ij} u \left(\pi_{ij}^{\text{mod}} \ (\theta_i) - \pi_{ij}^{\text{mkt}} \right), \quad i \in \{1, \ldots, n\} \tag{4.3}$$

where $\pi_{ij}^{\text{mod}} \ (\theta_i) \left(\pi_{ij}^{\text{mkt}} \right.$ respectively) denotes the model (market resp.) price of an option with maturity T_i and Strike $K_{ij}, u : \mathbb{R} \to \mathbb{R}^+$ is some (positive, nonlinear, convex) function (e.g. square or absolute value) measuring the distance between market and model prices.

The adversarial part of the algorithm is represented by variable weights w_{ij}, that can be for example of vega type, which allows to match implied volatility data rather then pure prices, our actual goal very well.
In fact, choosing weight of vega type proposed by S. Ben Hamida and R. Cont in [3]

$$w_{ij} = \max \left(\frac{1}{\text{Vega}\,(T_i, K_{ij})}, 100 \right) \tag{4.4}$$

it is possible to “converts” errors in price into errors in implied volatility.
This occurs because Vega is the derivative of the option value with respect to the volatility of the underlying asset, thus it measures sensitivity to volatility. What happens is that the calibration functional gives a greater weight and hence a greater ”importance” in the calibration task to those strikes and

maturities that are more sensitive to variations in volatility. Thresholding by 100 in (4.4) avoids overweighting of options very far from the money.

We solve the minimization problems (4.3) iteratively: we start with maturity T_1 and fix θ_1. This then enters in the computation of $\pi_{2j}^{\mathrm{mod}}(\theta_2)$ and thus in (4.3) for maturity T_2, etc. To simplify the notation in the sequel, we shall therefore leave the index i away so that for a generic maturity $T > 0$, (4.3) becomes

$$\mathrm{argmin}_{\theta\in\Theta} \sum_{j=1}^{J} w_j u\left(\pi_j^{\mathrm{mod}}(\theta) - \pi_j^{\mathrm{mkt}}\right) \tag{4.5}$$

Since the model prices are given by

$$\pi_j^{\mathrm{mod}}(\theta) = \mathbb{E}\left[(S_T(\theta) - K_j)^+\right] \tag{4.6}$$

we have $\pi_j^{\mathrm{mod}}(\theta) - \pi_j^{\mathrm{mkt}} = \mathbb{E}\left[Q_j(\theta)\right]$ where

$$Q_j(\theta)(\omega) := (S_T(\theta)(\omega) - K_j)^+ - \pi_j^{\mathrm{mkt}} \tag{4.7}$$

Note that S_T depends via (4.2) on θ. The calibration task then amounts to finding a minimum of

$$f(\theta) := \sum_{j=1}^{J} w_j u\left(\mathbb{E}\left[Q_j(\theta)\right]\right) \tag{4.8}$$

4.1.1 Gradient based algorithm

In light of Theorem 2.1.1, it is clear that neural networks can serve as function approximators and the goal is to find the "correct" parameters. Usually, the situation is such that the unknown function is expressed as an expectation. Probably the most prolific training method for such a setup is stochastic gradient descent, and we will shortly recall the most basis facts about this optimization/training method.

The structural properties of neural networks allow to solve minimization problems of the type

$$\min_{\theta\in\Theta} f(\theta) \quad \text{with} \quad f(\theta) = \mathbb{E}[Q(\theta)] \tag{4.9}$$

for some stochastic objective function $Q : \Omega \times \Theta \to \mathbb{R}, (\omega, \theta) \mapsto Q(\theta)(\omega)$ that depends on parameters θ in some space Θ very efficiently via **stochastic gradient** descent and **backpropagation**.

The classical method how to solve generic optimization problems for some differentiable objective function f (not necessarily of the expected value form as in (4.9)) is to apply a gradient descent algorithm: starting with an initial guess $\theta^{(0)}$, one iteratively defines

$$\theta^{(k+1)} = \theta^{(k)} - \eta_k \nabla f^{(k)} \left(\theta^{(k)}\right) \tag{4.10}$$

for some learning rate η_k and $f^{(k)} = f$. Under suitable assumptions, $\theta^{(k)}$ converges for $k \to \infty$ to a local minimum of the function f.

One of the key insights of deep learning is that stochastic gradient descent methods are much more efficient. To apply this, it is crucial that the objective function f is linear in the sampling probabilities. In other words, f needs to be of the expected value form as in (4.9). In the simplest form of stochastic gradient descent, under the assumption that

$$\nabla f(\theta) = \mathbb{E}[\nabla Q(\theta)]$$

the true gradient of f is approximated by a gradient at a single sample $Q(\theta)(\omega)$ which reduces the computational cost considerably. In the updating step for the parameters θ as in (4.10), ∇f is then replaced by $\nabla Q(\theta)(\omega)$, hence

$$\theta^{(k+1)} = \theta^{(k)} - \eta_k \nabla Q^{(k)} \left(\theta^{(k)}\right)(\omega) \tag{4.11}$$

with $Q^{(k)} = Q$. The algorithm passes through all samples ω of the so-called training data set and performs the update for each element, several times until an approximate minimum is reached.

A compromise between computing the true gradient of f and the gradient at a single $Q(\theta)(\omega)$ is to compute the gradient of a subsample of size N_{batch} , called (mini)-batch, so that $Q^{(k)}$ used in the update (4.11) is now given by

$$Q^{(k)}(\theta) = \frac{1}{N_{\text{batch}}} \sum_{n=1}^{N_{\text{batch}}} Q(\theta) \left(\omega_{n+kN_{\text{batch}}}\right), \quad k \in \{0, 1, \dots, \lfloor N/N_{\text{batch}} \, , \rfloor - 1\} \tag{4.12}$$

where N is the size of the whole training data set. Any other unbiased estimators of $\nabla f(\theta)$ can of course also be applied in (4.11).

In typical applications of machine learning, the data set available is limited, i.e. $N < \infty$. In our situation, we apply machine learning in a simulated environment, allowing us to generate data at will. This corresponds to $N = \infty$, meaning that we can generate completely new data in each step k.

Since u in equation (4.8) is a general non-linear function, this is clearly not of the expected value form of problem (4.9). In the following section we illustrate two possibilities how to deal with this non-linearity and the fact that stochastic gradient descent is not directly applicable.

4.2 Minimize the functional calibration

The goal of this section is to specify two methods for minimizing (4.8). It is possible to consider linearized versions of (4.8) such that classical stochastic gradient descent with potentially small batch-size is possible. In this thesis however, we consider two approaches that both amount to use classical gradient descent and a technical stratagem to use stochastic gradient descent when the function u in (4.8) has a particular form.

4.2.1 Standard gradient descent

The most obvious choice (which however does not work in practice) is to use a standard Monte Carlo estimator for $\mathbb{E}\left[Q_j(\theta)\right]$ so that (4.8) is estimated by

$$\widehat{f}(\theta) = \sum_{j=1}^{J} w_j u\left(\frac{1}{m}\sum_{l=1}^{m} Q_j(\theta)\left(\omega_l\right)\right) \tag{4.13}$$

for i.i.d samples $\{\omega_1, \ldots, \omega_m\} \in \Omega$.
Since the Monte Carlo error decreases as $\frac{1}{\sqrt{m}}$, the number of simulation m has to be chosen large ($\approx 10^8$) in order to approximate well the true model prices in (4.6). Note that implied volatility to which we actually aim to

calibrate is even more sensitive. As it is not obvious how to apply stochastic gradient descent due to the non-linearity of u, it seems necessary at first sight to compute the gradient of the whole function $\hat{f}(\theta)$. As $m \approx 10^8$, this is however computationally very expensive and does not allow to find a minimum in the usually high dimensional parameter space Θ in a reasonable amount of time.

4.2.2 Standard gradient descent with control variates

One possible remedy is to apply hedging control variates as introduced in Section 3.2.2 as variance reduction technique. This allows to reduce the number of samples m in the Monte Carlo estimator drastically so that usual (non-stochastic) gradient descent is enough to achieve accurate calibration results.

Assume that we have r hedging instruments (including the price process S) denoted by $(Z_t)_{t>0}$ which are σ -martingale under $\mathbb{Q}$ and take values in $\mathbb{R}^r$. Consider strategies $h_j : [0,T] \times \mathbb{R}^r \to \mathbb{R}^r$ and some constant c. Define

$$X_j(\theta)(\omega) := (S_t(\theta)(\omega) - K_j)^+ - c\left(h_j(\cdot, Z_\cdot(\theta)(\omega)) \bullet Z_\cdot(\theta)(\omega)\right)_t - \pi_j^{\mathrm{mkt}} \quad (4.14)$$

where $(h_j \bullet Z)_t$ denotes (a discretized version of) the stochastic integral with respect to Z. The calibration functionals (4.8) and (4.13) can then simply be defined by replacing $Q_j(\theta)(\omega)$ by $X_j(\theta)(\omega)$

Analogously as in Section 3.3.2 we can parametrize the hedging strategies via neural networks and find the optimal weight δ by computing

$$\operatorname{argmin}_{\delta\in\Delta} \frac{1}{N}\sum_{n=1}^{N} \ell\left(-X_j(\theta,\delta)(\omega_n)\right)$$

for i.i.d samples $\{\omega_1, \ldots, \omega_N\} \in \Omega$ and some loss function ℓ when θ is fixed. Here:

$$X_j(\theta,\delta)(\omega) = (S_T(\theta)(\omega) - K_j)^+ - \left(h_j(\cdot, Z_\cdot(\theta)(\omega) \mid \delta) \bullet Z_\cdot(\theta)(\omega)\right)_T - \pi_j^{mkt}$$

This means to iterate the two optimization procedures, one for θ and the other one for δ. Clearly the Black-Scholes hedge approach of Section 3.3.1 works as well, in this case without additional optimization with respect to the hedging strategies.

Estimator compatible with stochastic gradient descent

We show at least in a special case of the nonlinear function ℓ an application of stochastic gradient descent to the calibration functional (4.8). This means that we must cast (4.8) into expected value form. We focus on the case when $u(x)$ is given by $u(x) = x^2$ and write $f(\theta)$ as

$$f(\theta) = \sum_{j=1}^{J} w_j \mathbb{E}\left[Q_j(\theta)\tilde{Q}_j(\theta)\right]$$

for some independent copy $\tilde{Q}_j(\theta)$ of $Q_j(\theta)$, which is clearly of the expected value form required in (4.9). A Monte Carlo estimator of $f(\theta)$ is then constructed by

$$\widehat{f}(\theta) = \frac{1}{N} \sum_{n=1}^{N} \sum_{j=1}^{J} w_j Q_j(\theta)\left(\omega_n\right) \widetilde{Q}_j(\theta)\left(\omega_n\right)$$

for independent draws $\omega_1, \ldots, \omega_N$ (the same N samples can be used for each strike K_j). Equivalently we have

$$\widehat{f}(\theta) = \frac{1}{N} \sum_{n=1}^{N} \sum_{j=1}^{J} w_j Q_j(\theta)\left(\omega_n\right) Q_j(\theta)\left(\omega_{n+m}\right)$$

for independent draws $\omega_1, \ldots, \omega_{2N}$. The analog of (4.12) is then given by

$$Q^{(k)}(\theta) = \frac{1}{N_{\text{batch}}} \sum_{l=1}^{N_{\text{batch}}} \sum_{j=1}^{J} w_j Q_j(\theta)\left(\omega_{l+2kN_{\text{batch}}}\right) Q_j(\theta)\left(\omega_{l+(2k+1)N_{\text{hateh}}}\right)$$

for $k \in \{0, 1, \ldots, \lfloor N/N_{\text{batch}} \rfloor - 1\}$.
Clearly we can now modify and improve the estimator by using again hedge control variates and replace $Q_j(\theta)$ by $X_j(\theta)$ as defined in (4.14).

4.3 Numerical implementation

We consider a SABR type model with one dimensional price-process and one dimensional variance process V for which the dynamics are given by

$$\begin{aligned} dS_t &= S_t L^d(t, S_t) V_t dW_t \\ dV_t &= \nu V_t dB_t \\ dW_t dB_t &= \rho dt \end{aligned} \tag{4.15}$$

for two Brownian motions W, B with correlation $\rho \in [-1, 1]$.
Since to compute model prices shoud be used a Monte Carlo method via an Euler-discretization of the model, it will be preferable to work in log-price coordinates for S. In particular, it's possible to parametrize L^d with $X := \log S$ rather then S.
By denoting this parametrization again with L^d where d stands for "data", we therefore have $L^d(t, X)$ instead of $L^d(t, S)$ and the model dynamics read

$$\begin{aligned} dX_t &= V_t L^d(t, X_t) dW_t - \frac{1}{2} V_t^2 L^d(t, X_t)^2 dt \\ dV_t &= \nu V_t dB_t \\ dW_t dB_t &= \rho dt \end{aligned}$$

Note that V is a geometric Brownian motion, in particular, the closed form solution for V is available and given by

$$V_t = V_0 \exp\left(-\frac{\nu^2}{2} t + \nu B_t\right)$$

Recall that we specify the leverage function $L(t, x)$ via a family of neural networks, i.e.,

$$L(t, x) = 1 + F_i(x) \quad t \in [T_{i-1}, T_i), \quad i \in \{1, \ldots, n = 4\}$$

where $F_i \in \mathcal{NN}_{1,1}$. Each F_i can be specified, for example, as a 3-hidden layer feed forward net work where the dimension of each of the hidden layers is 50 . As activation function can be chosen $\sigma = \tanh$. As before we denote the parameters of F_i by θ_i and the corresponding parameter wpace by Θ_i.

Since closed form pricing formulas are not available for such an LSV model, let us briefly specify our pricing method. For the variance reduced Monte Carlo estimator as of (4.13) can be always used a standard Euler-SDE discretization with step size $\Delta_t = 1/100$. As variance reduction method, can be implemented the running Black - Scholes Delta hedge with instantaneous running volatility of the price process, i.e., $L(t, X_t) V_t$ is plugged in the formula for the Black - Scholes Delta. The only parameter that remains to be specified, is the number of trajectories used for the Monte Carlo estimator which is done in Algorithm 4.3.1 and Algoritm 4.3.2 below.

As a first calibration step, it's necessary to calibrate the SABR model (i.e., (4.15) with $L \equiv 1$) to the market prices and fix the calibrated SABR parameters ν, ϱ and V_0. For the remaining parameters θ_i, $i = 1, \ldots, 4$, are applied the following algorithm until all parameters are calibrated.

Algorithm 4.3.1.
In the subsequent pseudo code, the index i stands for the maturities, N for the number of samples used in the variance reduced Monte Carlo estimator as of (4.13) *and k for the updating step in the gradient descent:*

```
# Initialize the network parameters
initialize $\theta_1, \ldots, \theta_4$
# Define initial number of trajectories and initial step
$N$, $k$ = 400, 1
# The time discretization for the MC simulations and the
# abort criterion
$\Delta_t$, tol = 0.01, 0.0045

for $i$ = 1,...,4:
    nextslice = False
```

```
# Compute the initial normalized vega weights
# for this slice:
$w_j = \bar{w}_j / \sum_{l=1}^{20} \tilde{w}_l$ with $\bar{w}_j = 1/v_{ij}$, where $v_{ij}$ is
the Black-Scholes vega for strike $K_{ij}$, the corresponding
market implied volatility and the maturity $T_i$.

while nextslice == False
    do:
        Simulate N trajectories of the SABR-LSV
        process up to time $T_i$, compute the payoffs.
    do:
        Compute the stochastic integral of the
        Black-Scholes Delta hedge against
        these trajectories for maturity $T_i$
    do:
        Compute the calibration functional as of
        $(4.13)$ with $\ell(x) = x^2$ and weights
        $w_j$.
    do:
        Make an optimization step from $\theta_i^{(k-1)}$ to $\theta_i^{(k)}$,
        similarly as in $(4.10)$ but with the more
        sophisticated ADAM-optimizer with learning rate $10^{-3}$.
    do :
        Update the parameter $N$, the condition
        nextslice and compute model prices
        according to Algorithm $4.3.2$.
    do :
        $k$ = $k$ + 1
```

Algorithm 4.3.2.
We update the parameters in Algorithm 4.3.1 according to the following rules:

```
if $k$ == 500:
     $N$ = 2000
if $k$ == 1500:
     $N$ = 10000
else if $k$ == 4000:
     N = 50000

if $k$ >= 5000 and $k$ mod 1000 == 0:
     do:
          Compute model prices $\pi_{model}$ for slice $i$
          via MC simulation using $10^7$ trajectories.
          Apply the Black-Scholes Delta
          hedge for variance reduction.
     do :
          Compute implied volatilities $iv_{model}$
          from the model prices $\pi_{model}$.
     do :
          Compute the maximum error of model implied
          volatilities against market implied volatilities:
          err_cali = $\|$ iv_model - iv_market $\|_{max}$
          if err_cali $\leq$ tol or $k$ == 12000:
               $nextslice$ = True
          else:
               Adjust the weights w_{j } according to:
               for j = 1,...,20:
                    $w_j$ = $w_j$ + 0.1 * |iv_model$_j$ - iv_market$_j$|
               This puts higher weights on the options
               where the fit can still be improved
               Normalize the weights:
               for j = 1,...,20:
                    $w_j = u_j$ / $\sum_{\ell=1}^{20} w_l$
```

4.4 Conclusion

We have proposed an algorithm that show how the parametrization by means of neural networks can be used to calibrate local stochastic volatility models to implied volatility data. We make the following remarks:

1. The method we presented does not require any form of interpolation for the implied volatility surface since we do not calibrate via Dupire's formula. As the interpolation is usually done ad hoc, this might be a desirable feature of our method.

2. It is possible to "plug in" any stochastic variance process such as rough volatility processes as long as an efficient simulation of trajectories is possible.

3. The multivariate extension is straight forward.

4. As showed by C. Cuchiero, W. Khosrawi and J. Teichmann in [10], the level of accuracy of this algorithm is of a very high degree, making the presented method already of interest by this feature alone.

5. The method can be significantly accelerated by applying distributed computation methods in the context of multi-GPU computational concepts.

6. The presented algorithm is further able to deal with path-dependent options since all computations are done by means of Monte Carlo simulations.

Appendix A

Stochastic process

Definition A.1. A stochastic process is a family $(X_t)_{t\geq 0}$ of random variables with values in $\mathbb{R}$ such that the map

$$X : I \times \Omega \mapsto \mathbb{R} \quad X(t,\omega) = X_t(\omega)$$

is a function of both time t and randomness ω. For each ω, the trajectory

$$X(\omega) \mapsto X_t(\omega)$$

defines a function of time, called the sample path of process.

Definition A.2. A filtration on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ is an increasing family of $\sigma-$ algebras $(\mathcal{F}_t)_{t\geq 0}$ such that $\forall t \geq s \geq 0$ we have $\mathcal{F}_s \subseteq \mathcal{F}_t$

A.1 Brownian motion

Definition A.3. Let $\left(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\geq 0} \mathbb{P}\right)$ be a filtered probability space. A standard (one-dimensional) **Wiener process** or **Brownian motion**, is a stochastic process $W = (W_t)_{t\geq 0}$ in $\mathbb{R}$ such that:

- $W_0 = 0$ a.s.

- W is $\mathcal{F}$ -adapted and continuos

- for $t > s \geq 0$ the random variable $W_t - W_s$ has normal distribution i.e. $W_t - W_s \sim \mathcal{N}_{0,t-s}$ and is independent of $\mathcal{F}$

Definition A.4. Let $\left(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\geq 0} \mathbb{P}\right)$ be a filtered probability space. A stochastic process M is a martingale if:

- $M_t \in \mathbb{L}^1(\Omega) \forall t \geq 0$
- $\mathbb{E}\left[M_t \mid \mathcal{F}_s\right] = M_s$ for $s \leq t$

Definition A.5. Let $\left(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\geq 0} \mathbb{P}\right)$ be a filtered probability space. A stochastic process $M = (M_t)_{t\in[0,T]}$ is a $\mathcal{F}_t-$ local martingale if there exists an increasing sequence (τ_n) of $\mathcal{F}_t$ - stopping times, called localizing sequence for M, such that

$$\lim_{n\to\infty} \tau_n = T, \qquad \text{a.s.}$$

and the stochastic process $M_{t\wedge\tau_n}$ is a $\mathcal{F}_t$ - martingale for all $n \in \mathbb{N}$.

Definition A.6. Let $\left(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\geq 0} \mathbb{P}\right)$ be a filtered probability space and $(W_t)_{t\in[0,T]}$ a d-dimensional Brownian motion. Given a d-dimensional process $\theta \in \mathbb{L}^2_{\text{loc}}$, we define the exponential martingale associated to θ as:

$$Z^\theta_t = \exp\left(-\int_0^t \theta_s dW_s - \frac{1}{2}\int_0^t |\theta_s|^2 ds\right), \quad t \in [0,T]$$

For Itô's formula we have:

$$dZ^\theta_t = -Z^\theta_t \theta_t dW_t$$

and so Z^θ is a local martingale.

Theorem A.1.1 (Girsanov theorem).
Let Z^θ the exponential martingale associated to the process $\theta \in \mathbb{L}^2_{loc.}$ We assume that Z^θ is a $\mathbb{P}-$ martingale and we consider the measure $\mathbb{Q}$ defined by

$$\frac{d\mathbb{Q}}{d\mathbb{P}} = Z^\theta_T$$

Then the process

$$W^\theta_t := W_t + \int_0^t \theta_s ds, \quad t \in [0,T]$$

is a Brownian motion on $(\Omega, \mathcal{F}, \mathbb{Q}, \mathcal{F}_t)$

Theorem A.1.2.
Let X an Itô process in $\mathbb{R}^n$ defined as

$$X_t = X_0 + \int_0^t b_s ds + \int_0^t \sigma_s dW_s, \quad t \in [0,T]$$

with $b \in \mathbb{L}^1_{loc}$ and $\sigma \in \mathbb{L}^2_{loc}$. Given $r = \left(r^1, \cdots, r^N\right) \in \mathbb{L}^1_{loc}$, we assume that there exists a process $\theta = \left(\theta^1, \cdots, \theta^d\right) \in \mathbb{L}^2_{loc}$ such that:

- *it holds*

$$\sigma_t \theta_t = b_t - r_t, \quad t \in [0,T]$$

- *the process Z^θ is a $\mathbb{P}-$ martingale.*

Then we have that

$$X_t = X_0 + \int_0^t r_s ds + \int_0^t \sigma_s dW_s^\theta, \quad t \in [0,T]$$

where W^θ is the $\mathbb{Q}$ - martingale defined by Girsanov theorem.

Proposition A.1.3 (Markov inequality)**.**
Let X be a random variable and let $\lambda \in \mathbb{R}^+, 1 \leq p < +\infty$. Then

$$P(|X| \geq \lambda) \leq \frac{E\left[|X|^p\right]}{\lambda^p} \tag{A.1}$$

In particular, if X is a integrable real r.v., we have

$$P(|X - \mathbb{E}[X]| \geq \lambda) \leq \frac{\mathrm{var}(X)}{\lambda^2}$$

Theorem A.1.4 (Strong law of large numbers)**.**
Let (X_n) be a sequence of i.i.d. integrable random variables. Let $\mu = E\left[X_1\right]$ and

$$M_n = \frac{X_1 + \cdots + X_n}{n}$$

then we have

$$\lim_{n \to \infty} M_n = \mu$$

almost surely and in L^1 -norm.

A.2 Equivalent martingale measure

Definition A.7. An equivalent martingale measure $\mathbb{Q}$ is a probability measure on the space $(\Omega, \mathcal{F}, \mathbb{P})$ such that

- $\mathbb{P}$ and $\mathbb{Q}$ are equivalent measures i.e.

$$\mathbb{P}(A) = 0 \Longleftrightarrow \mathbb{Q}(A) = 0 \quad \text{for every } A \in \mathcal{F}$$

- the Radon-Nikodym derivative $\frac{dQ}{dP}$ belongs to $\mathbb{L}^2(\Omega, \mathcal{F}, \mathbb{P})$

- the discounted asset price process is a $\mathcal{F} - mg$.

Let suppose market model consists of a risk-free asset B_t and N risky assets $A_1^t, \cdots, A_N^t$. The risk-free asset B_t is a numeraire process governed by

$$dB_t = r(t)B_t dt$$

where $r(t)$ is locally deterministic interest rate. From this, we know that $B_t = \exp\left[\int_0^t r(s)ds\right]$

Theorem A.2.1 (First fundamental theorem).
The market model is free of arbitrage if and only if there exists a martingale measure $\mathbb{Q}$ such that the processes

$$\frac{B_t}{B_t}, \frac{A_t^1}{B_t}, \cdots, \frac{A_t^n}{B_t}$$

are martingales under $\mathbb{Q}$.

The martingale measure $\mathbb{Q}$ is usually called the risk neutral measure, under which the price of an option is unique such that there is no arbitrage opportunity.

Theorem A.2.2 (Second fundamental theorem).
Assuming free of arbitrage, the market model is complete if and only if the martingale measure $\mathbb{Q}$ is unique.

Proposition A.2.3 (Martingale pricing formula)**.**
To avoid arbitrage, a contingent claim must be priced by

$$\Pi(t;X) = B_t \cdot \mathbb{E}\left[\frac{X}{B_T} \mid \mathcal{F}_t\right] = \mathbb{E}\left[e^{-\int_t^T r(s)ds} X \mid \mathcal{F}_t\right]$$

under the risk neutral measure $\mathbb{Q}$*, given* $\Pi(T) = X$*.*

A.3 Stochastic differential equations

We assume that an $\mathbb{R}^n$ valued stochastic process $X_t = \left(X_1^t, \cdots, X_n^t\right)^T$ for $t > 0$ follows a stochastic differential equation (SDE)

$$dX_t = \mu\left(X_t, t\right) dt + \sigma\left(X_t, t\right) dW_t \tag{A.2}$$

where $W = (W_t^1, \cdots, W_t^m) \in \mathbb{R}^m$ is a $m-$ dimensional Brownian motion, $\mu\left(X_t, t\right) = \left(\mu_1, \cdots, \mu_n\right)^T \in \mathbb{R}^n$ is an $n-$ dimensional vector, and $\sigma\left(X_t, t\right) \in \mathbb{R}^{n\times m}$ is an $n \times m$ matrix as

$$\boldsymbol{\sigma}\left(\boldsymbol{X_t}, \boldsymbol{t}\right) = \begin{pmatrix} \sigma_{11} & \cdots & \sigma_{1m} \\ \cdots & \cdots & \cdots \\ \sigma_{n1} & \cdots & \sigma_{nm} \end{pmatrix}$$

Definition A.8. A process X_t is called a strong solution of the SDE (A.2) if for all $t > 0$ the integrals $\int_0^t \mu\left(X_s, s\right) ds$ and $\int_0^t \sigma\left(X_s, s\right) dW_s$ exist, and

$$X_t = X_0 + \int_0^t \mu\left(X_s, s\right) ds + \int_0^t \sigma\left(X_s, s\right) dW_s$$

Definition A.9. If there exists a probability space with a filtration, a Brownian motion $\widehat{W_t}$ and a process $\widehat{X_t}$ adapted to that filtration such that $\widehat{X_0}$ has the given distribution, and for all $t > 0, \widehat{X_t}$ satisfies

$$\widehat{X_t} = \widehat{X_0} + \int_0^t \mu\left(\widehat{X_s}, s\right) ds + \int_0^t \sigma\left(\widehat{X_s}, s\right) d\widehat{W_s}$$

then $\widehat{X_t}$ is called a weak solution of the SDE (A.2).

The first fundamental result for SDEs is the existence and uniqueness of the solution.

Theorem A.3.1 (Existence and uniqueness of strong solution for SDE with Lipschitz coefficients)**.**
Suppose that X_t follows (A.2)*. If the following conditions are satisfied:*

1. *the coefficients μ and σ are Lipschitz continuous;*
2. *the coefficients μ and σ satisfy the linear growth condition;*
3. *X_0 is independent of W_t and $\mathbb{E}\left[|X_0|^2\right] < \infty$*

then there exists a unique strong solution X_t of the (A.2)*.*

Theorem A.3.2 (Existence and uniqueness of strong solution for SDE with non-Lipschitz coefficients)**.**
Suppose that X_t follows (A.2)*. If the following conditions are satisfied:*

1. *μ is Lipschitz continuous;*
2. *there is an increasing function $\varrho(u), u \in (0, \infty)$ such that $\int_0^\varepsilon \varrho^{-2}(u)du = \infty$ for some $\varepsilon > 0$, and*

$$|\sigma(x,t) - \sigma(y,t)| \leq \varrho(|x-y|)$$

then there exists a unique strong solution X_t of the SDE (A.2)*.*

Theorem A.3.3.
If μ is Lipschitz continuous and σ is Hölder continuous of order $\alpha \geq \frac{1}{2}$, then there exists a unique strong solution.

Theorem A.3.4 (Itô's lemma)**.**
Suppose that X_t follows (A.2)*. Let $f(X_t, t)$ be a twice differentiable function. Then the process $f(x,t)$ follows*

$$df = \frac{\partial f}{\partial t}dt + \sum_{i=1}^{n} \frac{\partial f}{\partial X^i} dX_t^i + \frac{1}{2}\sum_{i,j=1}^{n} \frac{\partial^2 f}{\partial X^i \partial X^j} dX_t^i \cdot dX_t^j$$

Theorem A.3.5 (Feynman-Kač theorem).
Suppose that X_t follows (A.2). *Consider the PDE for $F(X_t, t)$*

$$\frac{\partial F}{\partial t} + \sum_{i=1}^{n} \mu_i \frac{\partial F}{\partial X^i} + \frac{1}{2} \sum_{i,j=1}^{n} \left(\sigma\sigma^T\right)_{ij} \frac{\partial^2 F}{\partial X^i \partial X^j} - r\left(X_t, t\right) F = 0$$

with terminal condition $F(X_T, T) = g(X_T)$. If the following conditions are satisfied:

1. *the coefficients $\mu, \sigma\sigma^T$ and r are bounded and satisfy Hölder condition;*
2. *$\sigma\sigma^T$ is uniformly positive definite;*
3. *the function $g(X_T)$ satisfies the polynomial growth condition,*

then there exists a unique solution $F(X_t, t)$ of the previous PDE. Given an initial condition $X_t = x = (x^1, \ldots, x^n)$, the solution $F(X_t, t)$ can be expressed as

$$F(x,t) = \mathbb{E}\left[e^{-\int_t^T r(X_u,u)du} g\left(X_T\right) \mid X_t = x\right] = e^{-\int_t^T r(X_u,u)du} \int_{\mathbb{R}^n} g(y) p(y, T \mid x) dy$$

where $p(y, T \mid x)$ is the transition density given $X_t = x$.

The Feynman-Kač formula shows a connection between stochastic processes and deterministic partial differential equations (PDEs).

Lemma A.3.6 (Gyöngy).
Let $(X_t)_t$ be an n-dimensional Itô's process, satisfying the following SDE :

$$\begin{aligned} dX_t &= \beta_t dt + \nu_t dW_t, \quad t \in [0,T] \\ X_0 &= x_0 \end{aligned}$$

where $(W_t)_t$ is a d-dimensional Brownian motion on the filtered probability space $(\Omega, \mathcal{F}, \mathbb{P}, \mathcal{F}_t)$, β_t and ν_t are stochastic processes, n-dimensional and $n \times d$-dimensional respectively, $\mathcal{F}_t$-adapted and bounded, such that $\nu_t \nu_t^T$ is uniformly positive definite. Hence defining:

$$b(t,x) = \mathbb{E}\left[\beta_t \mid X_t = x\right]$$

$$\sigma^2(t,x) = \mathbb{E}\left[\nu_t \nu_t^T \mid X_t = x\right]$$

the SDE with non-random coefficients given by b and σ :

$$\begin{aligned} dY_t &= b\left(t, Y_t\right) dt + \sigma\left(t, Y_t\right) dW_t, \quad t \in [0,T] \\ Y_0 &= X_0 \end{aligned}$$

admits a weak solution $\left(Y_t\right)_t$ having the same one-dimensional probability y distribution as $\left(X_t\right)_t$ for all $t \in [0,T]$ (i.e. for all $t \in [0,T]$, the random variables Y_t, X_t have the same distribution).

List of Figures

Bibliography

[1] F. Abergel and R. Tachet, *A nonlinear partial integro-differential equation from mathematical finance*, Discrete and Continuous Dynamical Systems - Discrete Ccontin Dyn Syst, 27 (2010), pp. 907–917.

[2] C. Alexander and L. Nogueira, *Stochastic Local Volatility*, ICMA Centre Discussion Papers in Finance icma-dp2008-02, Henley Business School, Reading University, Sept. 2004.

[3] S. Ben Hamida and R. Cont, *Recovering volatility from option prices by evolutionary optimization*, The Journal of Computational Finance, 8 (2005), pp. 43–76.

[4] T. Bjork, *Arbitrage Theory in Continuous Time*, 3 ed., 2009.

[5] F. Black and M. S. Scholes, *The Pricing of Options and Corporate Liabilities*, Journal of Political Economy, 81 (1973), pp. 637–654.

[6] H. Bühler, L. Gonon, J. Teichmann, and B. Wood, *Deep hedging*, 2018.

[7] J. C. Cox, *Notes on option pricing i: Constant elasticity of diffusions*, Unpublished Draft, Stanford University, (1975).

[8] J. C. Cox, J. E. Ingersoll, and S. Ross, *A theory of the term structure of interest rates*, Econometrica, 53 (1985), pp. 385–407.

[9] A. Cozma, M. Mariapragassam, and C. Reisinger, *Calibration of a hybrid local-stochastic volatility stochastic rates model with a control*

variate particle method, SIAM Journal on Financial Mathematics, 10 (2019), pp. 181–213.

[10] C. Cuchiero, W. Khosrawi, and J. Teichmann, *A generative adversarial network approach to calibration of local stochastic volatility models*, Risks, 8 (2020), p. 101.

[11] E. Derman and I. Kani, *Riding on a smile*, Risk, 7 (1994).

[12] B. Dupire, *Pricing with a smile*, Risk Magazine, (1994).

[13] B. Engelmann, F. Koster, and D. Oeltz, *Calibration of the heston stochastic local volatility model: A finite volume scheme*, 2011.

[14] J. Gatheral, *Jim Gatheral: The volatility surface, a practitioner's guide*, Financial Markets and Portfolio Management, 22 (2008), pp. 93–94.

[15] P. Glasserman, *Monte Carlo methods in financial engineering*, Springer, New York, 2004.

[16] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, The MIT Press, 2016.

[17] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, *Generative adversarial networks*, 2014.

[18] J. Guyon and P. Henry-Labordere, *The smile calibration problem solved*, SSRN Electronic Journal, (2011).

[19] J. Guyon and P. Henry-Labordère, *Being particular about calibration*, Risk, 25 (2012), p. 88.

[20] J. Guyon and P. Henry-Labordere, *Nonlinear Option Pricing*, Chapman and Hall/CRC Financial Mathematics Series, Chapman and Hall/CRC, 2013.

[21] P. Hagan, D. Kumar, A. Lesniewski, and D. Woodward, *Managing smile risk*, Wilmott Magazine, 1 (2002), pp. 84–108.

[22] R. Hecht-Nielsen, *Theory of the backpropagation neural network*, in Neural Networks for Perception, H. Wechsler, ed., Academic Press, 1992, pp. 65 – 93.

[23] P. Henry-Labordere, *Calibration of local stochastic volatility models to market smiles: A monte-carlo approach*, RISK, September, (2009).

[24] A. Hernández, *Model calibration with neural networks*, Neuroeconomics eJournal, (2016).

[25] S. L. Heston, *A closed-form solution for options with stochastic volatility with applications to bond and currency options*, Review of Financial Studies, 6 (1993), pp. 327–343.

[26] K. Hornik, *Approximation capabilities of multilayer feedforward networks*, Neural Networks, 4 (1991), pp. 251 – 257.

[27] J. HULL and A. WHITE, *The pricing of options on assets with stochastic volatilities*, The Journal of Finance, 42 (1987), pp. 281–300.

[28] M. Jex, R. C. Henderson, and D. Wang, *Pricing exotics under the smile*, 1999.

[29] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017.

[30] A. Lipton, *The vol smile problem*, Risk, 15 (2002), pp. 61–65.

[31] S. Liu, A. Borovykh, L. A. Grzelak, and C. W. Oosterlee, *A neural network-based framework for financial model calibration*, Journal of Mathematics in Industry, 9 (2019).

[32] M. Lorig, S. Pagliarani, and A. Pascucci, *Explicit implied volatilities for multifactor local-stochastic volatility models*, 2014.

[33] R. Merton, *The theory of rational option pricing*, Bell J Econ Manage Sci, 4 (1973), pp. 141–183.

[34] A. Pascucci, *PDE and Martingale Methods in Option Pricing*, Springer-Verlag, Milan, Italy, 2010.

[35] S. R. Pliska, *Introduction to Mathematical Finance: Discrete Time Models*, Blackwell, Malden, MA, 1997.

[36] A. Quarteroni, R. Sacco, and F. Saleri, *Matematica numerica*, (1998).

[37] R. Rebonato, *Volatility and Correlation: The Perfect Hedger and the Fox*, 01 2004.

[38] Y. Ren, D. Madan, and M. Q. Qian, *Calibrating and pricing with embedded local volatility models*, Risk, (2007).

[39] S. Ruder, *An overview of gradient descent optimization algorithms*, 2017.

[40] Y. F. Saporito, X. Yang, and J. P. Zubelli, *The calibration of stochastic local-volatility models: An inverse problem perspective*, Computers & Mathematics with Applications, 77 (2019), pp. 3054 – 3067.

[41] E. M. Stein and J. C. Stein, *Stock price distributions with stochastic volatility: An analytic approach*, Review of Financial Studies, 4 (1991), pp. 727–752.

[42] Y. Tian, Z. Zhu, G. Lee, F. Klebaner, and K. Hamza, *Calibrating and pricing with a stochastic-local volatility model*, The Journal of Derivatives, 22 (2015), pp. 21–39.