COMPUTING THE IMPLIED VOLATILITY THROUGH NEURAL NETWORKS WITH ASYMPTOTIC REGIMES

Samira Amiriyan[1] And Youness Boutaib [1]

[1]Department of Mathematical Sciences, University of Liverpool, UK

Introduction

The Black-Scholes model, introduced by Fischer Black and Myron Scholes in their seminal paper [1] has profoundly influenced modern financial theory. Among their model's parameters for the price of a European call $C(S_0, K, T, r, \sigma)$, one parameter (namely the volatility parameter σ) has attracted considerable attention. One can observe that the Black-Scholes pricing function is monotonically increasing with respect to its volatility parameter, σ. This property enables the definition of the concept of implied volatility, which can be interpreted as the inverse of the Black-Scholes formula with respect to σ, when we put it equal to real market price. The normalized Black and Scholes call option pricing formula with Strike K and expiration time T on the underlying asset S_t is as follows:

$$C_{\mathrm{BS}}(A, B) = \frac{C(S_0, K, T, r, \sigma)}{S_0} = \Phi\left(-\frac{A}{B} + \frac{B}{2}\right) - \Phi\left(-\frac{A}{B} - \frac{B}{2}\right) e^A,$$

where $A = \log\left(\frac{Ke^{-rT}}{S_0}\right)$ and $B = \sigma\sqrt{T}$.

Due to the complex, non-linear structure of the Black-Scholes formula, an explicit closed-form solution for implied volatility does not exist. Consequently, the computation of implied volatility requires the application of numerical methods, which presents both theoretical and practical challenges in quantitative finance. Therefore, in this work, inspired by the industry's gold standard (Jäckel's method) [2] and the revolution introduced by neural networks, we propose a novel type of neural networks with asymptotic regimes to compute implied volatilities. In this setting, the (log-moneyness, price) space is partitioned into three regions based on the level of volatility, which itself depends on the log-moneyness. For each, a neural network learns a flexible asymptotic weight function (called modulating function) and a corresponding pricing function.

Review of the analysis of the Black and Scholes formula

In this section, we recall some properties of the Black and Scholes formula that inspire the design of the neural network architectures in this work. Our purpose is to find, given $A \geq 0$ and $C \in (0, 1)$, the value of the total variance B_{inv} that satisfies the equation $C = C_{\mathrm{BS}}(A, B_{\mathrm{inv}})$.

In the problem of inverting the Black and Scholes formula, the most successful methods divide the space (A, B) into essentially three regions

$$B \leq B_l(A) \quad , \quad B_l(A) \leq B \leq B_u(A) \quad \text{and} \quad B \geq B_u(A), \tag{1}$$

due to the map $C_{\mathrm{BS}}(A, .)$ having a different growth rate near and far from the inflection point $\sqrt{2A}$, as can be seen from the partition into 3 regions illustrated in Figure 1.

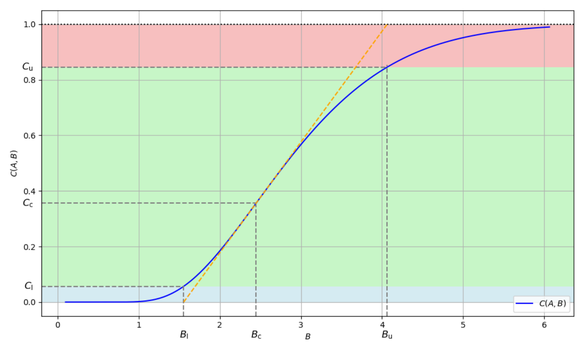


Figure 1. The partition of the space (B, C) into 3 regions defined by the asymptotic to the curve $C_{\mathrm{BS}}(A, \cdot)$. Here $A = 3$.

For instance, [2] champions the use of interpolation methods using rational functions for each given value of A (in particular, the rational cubic Hermite interpolants from [3]), while [4] suggests a uniform approximation across the grid (A, C_{BS}) using Chebyshev polynomials.

Inspired by [2] and [4], we take $B_u(A)$ to be the point of intersection of the asymptotic to $B \to C_{\mathrm{BS}}(A, B)$ at its inflection point $\sqrt{2A}$ with the value $C_{\mathrm{BS}} = 1$, and $B_l(A)$ to be the point of intersection of the asymptotic to $B \to C_{\mathrm{BS}}(A, B)$ at its inflection point $\sqrt{2A}$ with the value $C_{\mathrm{BS}} = 0$.

The analysis yields a partition of the space of values for (A, C) into regions corresponding to different behaviours of the implied volatility function. These are illustrated in Figure 2.

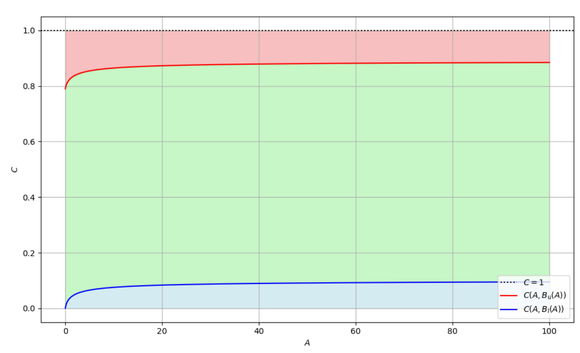


Figure 2. The partition of the space (A, C) into 3 regions defined by the inequalities (1).

Choices of architecture

Having highlighted in the previous section the different behaviours of the implied volatility in the three regions of the domain of A and C, we present now the idea behind our design based on neural network. The idea is to compute an expansion of the implied volatility function in the following form (which we will label as "Free"),

$$\textbf{Free:} \quad \mathrm{IV}(A, C) = f_0(A, C)g_0(A, C) + f_1(A, C)g_1(A, C) + g_2(A, C). \tag{2}$$

In the above expression, (f_0, f_1) plays the role of a trainable parametric partition of unity on $\mathbb{R}_+ \times \mathbb{R}_+$ (but will not be one for all possible values of the parameters). We will call these the modulating functions. The functions g_0, g_1, g_2 approximate the implied volatility function in the corresponding regions. More explicitly, g_0 will be the function that dominates the approximation of the implied volatility function in the region $C \in [0, C_{\mathrm{BS}}(A, B_l(A))]$. Similarly g_1 dominates the approximation in the region $C \in [C_{\mathrm{BS}}(A, B_u(A)), 1]$, while g_2 dominates the approximation in the region $C \in [C_{\mathrm{BS}}(A, B_l(A)), C_{\mathrm{BS}}(A, B_u(A))]$.

We compare our networks with two standard neural networks, NN-Exp and NN-Simple, both learning the mapping $(A, C) \to B$. The key difference is in the output: NN-Exp applies an exponential to ensure positivity, while NN-Simple uses a plain linear layer.

Choices for the implied volatility functions

There are several ways in which one can choose to compute the functions g_i, $i \in \{1, 2, 3\}$. The two most natural ones are to either compute the g_i's using a feed-forward neural network (we label this choice of implementation by "Gen"), or -as implied volatilities are positive- as the exponentials of the realisation of such a neural network (labelled as "Exp").

Choices for the modulating functions

A possible way to model a flexible asymptotic weight function for the region $C \to 1$ is

$$f_1(A, C) = \frac{1}{1 + \sum_{i=1}^{N} \alpha_i (A + \varepsilon_i)^{\beta_i} \left(\frac{1}{C} - 1\right)^{\gamma_i}},$$

where all the parameters α_i, β_i, γ_i and ε_i are trainable positive parameters. We chose to compare two possible implementations for f_1 labelled as follows:

Inv: We learn positive parameters α, β, γ and ε (as exponentials of unconstrained learnable parameters) and compute

$$f_1(A, \widetilde{C}) = \frac{1}{1 + \alpha(A + \varepsilon)^{\beta} \widetilde{C}^{\gamma}}. \tag{3}$$

where $\widetilde{C}$ is the auxiliary variable $\widetilde{C} := \frac{1}{C} - 1$.

Sig: Given parameters to learn $\alpha \in \mathbb{R}$, $\beta > 0$ and $\gamma > 0$ (where the strictly positive parameters are learnt as exponentials of unconstrained parameters), we first compute

$$t_1 = \alpha - \beta \widetilde{A}_{\log} - \gamma \widetilde{C}_{\log},$$

where $\widetilde{A}_{\log} := \log(A)$ and $\widetilde{C}_{\log} := \log\left(\frac{1}{C} - 1\right)$. The output is then obtained by the application of the sigmoid activation function σ_{sig} on t_1. Similary for $C \to 0$ we have,

$$f_0(A, C) = \frac{1}{1 + \sum_{i=1}^{N} \alpha_i A^{-\beta_i} \left(\frac{1}{C} - 1\right)^{-\gamma_i}}.$$

Numerical Results

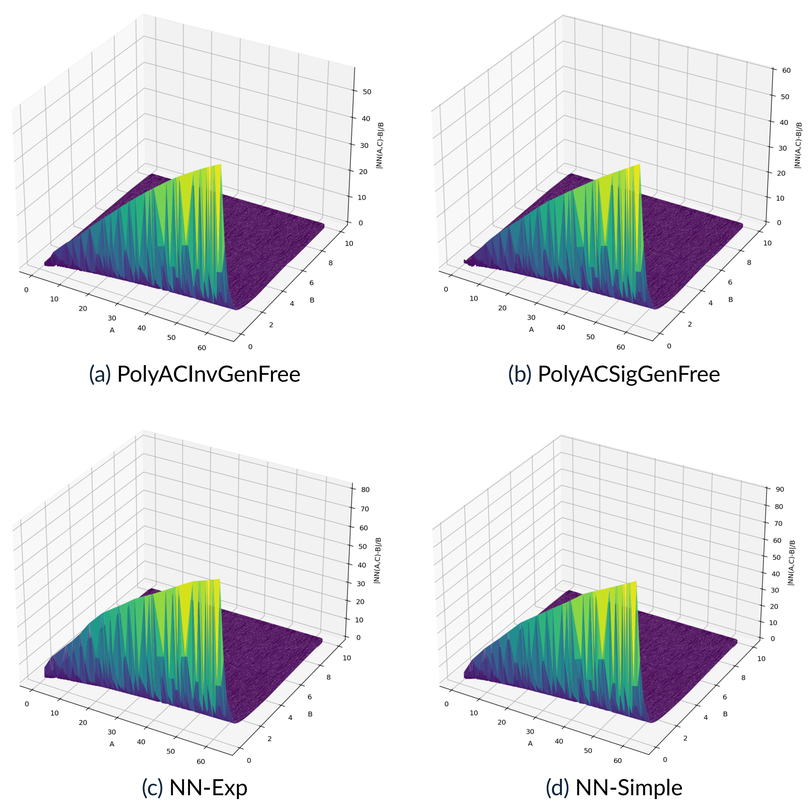
We first build a dataset of values $\{(A_i, B_i, C_i)\}_{i=1}^{m}$ (where $A \in (0, 64]$, $B \in (0, 10]$, and $C = C_{\mathrm{BS}}(A, B)$) across a regular grid of values for (A, B) that will serve to learn the map $(A, C) \mapsto B$ (with $m_1 = 0.8m$ points from the dataset dedicated for learning, with the remaining $m_2 = 0.2m$ points serving for testing.) The architectures were compared across the learning dataset and the testing dataset using the following metrics: (IV denoting the learnt map $(A, C) \mapsto B$ and $k \in \{1, 2\}$)

1. Mean Square Error: $L_k = \frac{1}{2m_k} \sum_{i=1}^{m_k} |\mathrm{IV}(A_i, C_i) - B_i|^2$.
2. Absolute residual error: $\Delta_k = \max_{1 \leq i \leq m_k} |\mathrm{IV}(A_i, C_i) - B_i|$.
3. Relative residual error: $\delta_k = \max_{1 \leq i \leq m_k} \frac{|\mathrm{IV}(A_i, C_i) - B_i|}{B_i}$.

The table 1 clearly shows that our networks, especially "PolyACInvGenFree" outperform the standard neural networks, with significantly lower error. This observation is also evident in Figure 3.

Networks/Errors	Test Mean Square	Test Absolute residual	Test Relative residual
PolyACInvGenFree	**0.551887**	**2.871986**	**57.152523**
PolyACSigGenFree	0.583861	2.976826	59.238846
NN-Exp	2.160205	4.292743	81.073875
NN-Simple	2.182633	4.477299	88.905853

Table 1. Comparison of Network Architectures Based on Test Error Metrics



(a) PolyACInvGenFree (b) PolyACSigGenFree

(c) NN-Exp (d) NN-Simple

Figure 3. Relative Errors Vs (A, B)

Ongoing Research

In future, in addition to implementing a more numerically accurate version of the Black and Scholes function, possible improvements include more machine-learning friendly and more expressive implementations of the modulating functions f_0 and f_1. We also aim to extend our approach to a model-dependent framework, where the asymptotic behavior of option prices under a specific model is first derived. The implied volatility surface is then decomposed into four regions: large maturity T, large strike, and small strike K, and an intermediate zone. A tailored neural network architecture will be trained to learn this composite structure.

References

[1] F. Black and M. Scholes, "The pricing of options and corporate liabilities," *Journal of political economy*, vol. 81, no. 3, pp. 637–654, 1973.

[2] P. Jäckel, "Let's be rational," *Wilmott*, vol. 2015, no. 75, pp. 40–53, 2015.

[3] R. Delbourgo and J. A. Gregory, "Shape preserving piecewise rational interpolation," *SIAM journal on scientific and statistical computing*, vol. 6, no. 4, pp. 967–976, 1985.

[4] K. Glau, P. Herold, D. B. Madan, and C. Pötz, "The chebyshev method for the implied volatility," *arXiv preprint arXiv:1710.01797*, 2017.

samira.amiriyan@liverpool.ac.uk