CCIMI & Morgan Stanley Collaborative Project Presentation (shorter version)

Neural network approximation to the SABR option pricing model

Parley Ruogu Yang & Russell Barker

UNIVERSITY OF CAMBRIDGE
Faculty of Mathematics
Morgan Stanley

31 March 2021

This is a shorter version of my previous talk.
See full version at my website:
https://parleyyang.github.io/Cam/index.html

An advertisement about my research group:
Optimal Portfolio Research Group (Cambridge, Oxford, UCD)
https://optimalportfolio.github.io/
We ARE looking for industrial collaborations!

Key reference: Horvath, Blanka, Aitor Muguruza, and Mehdi Tomas (2019). *Deep Learning Volatility*, arXiv:1901.09647v2

The set-up from Horvath et al (2019)

Whiteboard

SABR model

Whiteboard

SABR model — parameters

We fix certain probability distribution on $\mathbb{R}^5$, in particular,

$$P(0) \sim U[0, 200] \tag{1a}$$
$$\alpha(0) \sim U[0, 1] \tag{1b}$$
$$\beta \sim U[0, 1] \tag{1c}$$
$$\rho \sim U[-1, 1] \tag{1d}$$
$$v \sim U[0, 1] \tag{1e}$$

Motivation for computational analysis

Whiteboard

Contribution

- Deepen the understanding of the approximation behaviour towards SABR pricing model
- Computationally observe the trade-off amongst Monte Carlo sample size (M), Price paths per sample (N) and step size (s) given computing constraints.
- Further the observations on NN approximations[1] and potential failures.

[1] Harry and Valeria would have more to talk about on this part.

Algorithm

Algorithm 1: Simulation and getting a sample of the distribution of $F(K;\theta)$, with Monte Carlo standard deviations

Input: Interest rate r, strike price K, terminal time T, incremental time s, number of paths per draw of θ, denoted N, number of draws of θ, denoted M, and the distribution of θ

Output: $\{P_{i,j}(T)\}_{j=1}^{N}$ thus $F_i(K;\theta_i)$ for each θ_i drawn, and get $\{F_i(K;\theta_i)\}_{i=1}^{M}$ with Monte Carlo standard deviation $\{\sigma_i^{MC}\}_{i=1}^{M}$

1. For $i \in \{1, ..., M\}$:
   (a) Sample θ_i
   (b) Repeat below for N times to get $\{P_{i,j}(T)\}_{j=1}^{N}$:
      i. Sample Brownian Motion $W_1(t), W_2(t)$:
         A. Sample $\Delta W_2(t) \sim N(0, s)$
         B. Sample $\Delta W_1(t)$
      ii. Compute $\alpha(t), P(t)$:
         A. Compute $\alpha(t)$
         B. Compute $\Delta P(t) = \alpha(t)P(t)^{\beta}\Delta W_1(t)$
   (c) Obtain $F_i(K; \theta_i), \sigma_i^{MC}$
      i. Compute $P_{i,j}^{call}(K; P_{i,j}) := \max\{P_{i,j}(T; \theta_i) - K, 0\}e^{-rT}$ for all j
      ii. Obtain $F_i(K; \theta_i)$ as the mean of $\{P_{i,j}^{call}(K; P_{i,j})\}_{j=1}^{N}$
      iii. Obtain σ_i^{MC} as the standard deviation of $\{P_{i,j}^{call}(K; P_{i,j})\}_{j=1}^{N}$
      iv. Reject the sample and re-run (i.e. return to 1(a) and keeping the same index i) if $F_i(K; \theta_i) > 400$
2. Therefore obtain $\{F_i(K; \theta_i), \sigma_i^{MC}\}_{i=1}^{M}$

Remarks and computational costs

Skipped

Specifications in this experiment

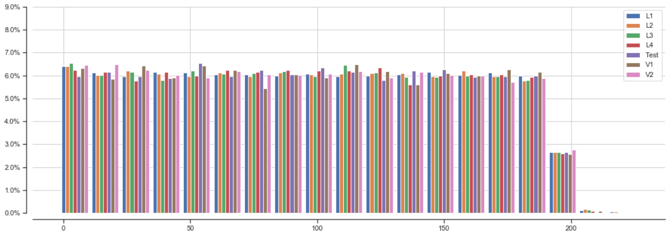
Here we simply fix J and the specification of K_j: we let $K_1, \ldots, K_8$ to be $0.7F(0), \ldots, 1.4F(0)$, respectively.

Choices of combinations for training and validation set:

N	M	J	s^{-1}	Name
50	12K	8	100	V1
500	12K	8	20	V2
50	100K	8	10	L1
25	100K	8	20	L2
50	50K	8	20	L3
100	25K	8	20	L4

Configuration for the fixed test set, which will be used for final evaluation:

N	M	J	s^{-1}	Name
500	10K	8	100	TestData



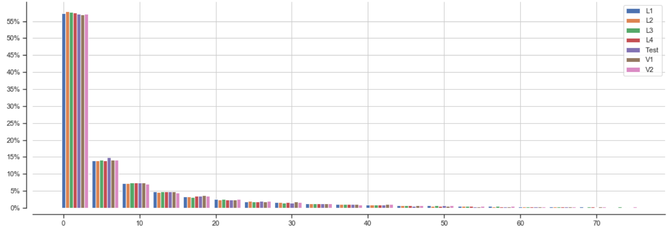


Figure: Histogram of $F_i(K;\theta_i)$ (top) and σ_i^{MC} (bottom)

Aim

Whiteboard

Architecture

Recall that a component-wise Exponential Linear Units (ELU) stands for

$$ELU(x) = \mathbb{1}[x > 0]x + \mathbb{1}[x < 0](e^x - 1)$$

and that a component-wise Rectified ELU (RELU) stands for

$$RELU(x) = \max\{0, x\}$$

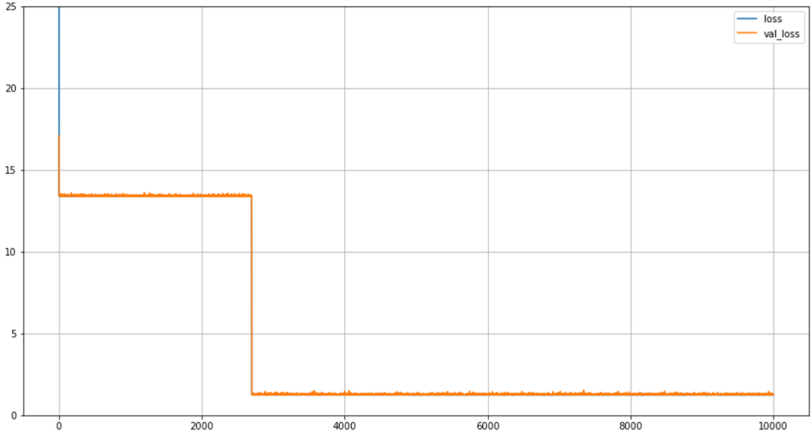
We consider

$$\phi = \sigma_3 \circ W_3 \circ \sigma_2 \circ W_2 \circ \sigma_1 \circ W_1$$

where $\sigma_1 = \sigma_2 = ELU$ and $\sigma_3 \in \{ELU, RELU\}$ and affine maps $W_l : \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l}$. We thus have neuron vector $n = (n_0, n_1, n_2, n_3)$ with $n_0 = \dim(\Theta)$ and $n_3 = J$

Training & Empirical troubles

Training: ADAM with MAE loss. Weight initialisation: the trouble.



Validation — Evaluation function

Consider a function that outputs the percentage of predictions errors that are larger than $k\sigma^{MC}$, where $k \in \{1, 2, 3\}$. This can be mathematically written as:

$$\ell_k^{MC}(\sigma, D, \phi) = 100(|D|J)^{-1} \sum_{(x,y,\sigma^{MC})\in D} \sum_{j=1}^{J} \mathbb{1}[|y_j - (\phi(x))_j| > k\sigma] \quad (2)$$

The choice of σ varies — here we consider the σ_L, σ_S, noting the maximum and minimum, respectively, of the generated σ^{MC}

Validation — Evaluation procedure

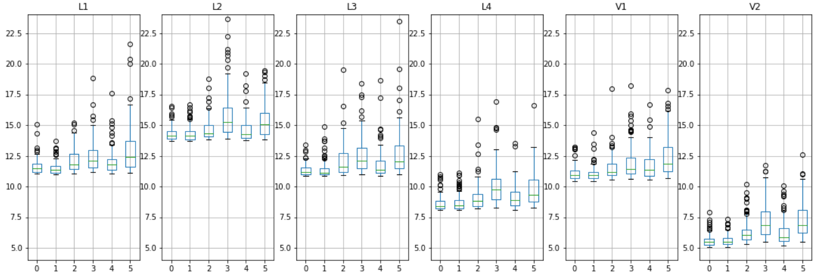
1. For each ϕ_z, for each randomisation $i \in \{1, 2, ..., 99, 100\}$,[2] we obtain $l(D; z, i) = \ell_2^{MC}(\sigma_L, D, \phi_z)$.
2. Summarise them by obtaining $l^{median}(D; z)$ as the median of $\{l(D; z, i)\}_{i=1}^{100}$.
3. Select the top-performing model of each dataset, that is, $z^*(D) = \arg\min_{z \in Z}\{l^{median}(D; z)\}$

[2] This can be completed by the `setseed` in python.

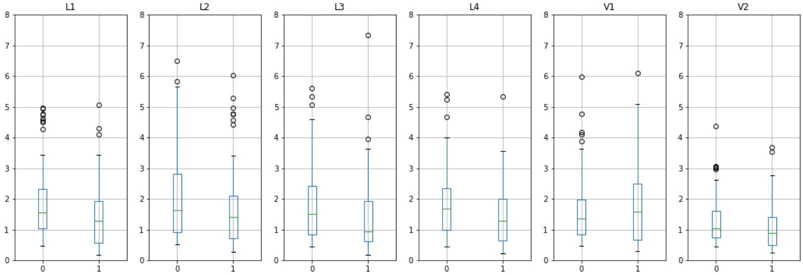
Practical set-ups of the architecture

Label	n	Total number of parameters	σ_3
0	(5,6,7,8)	149	ELU
1	(5,6,7,8)	149	RELU
2	(5,10,40,8)	828	ELU
3	(5,10,40,8)	828	RELU
4	(5,40,40,8)	2208	ELU
5	(5,40,40,8)	2208	RELU

Validation results



Test results



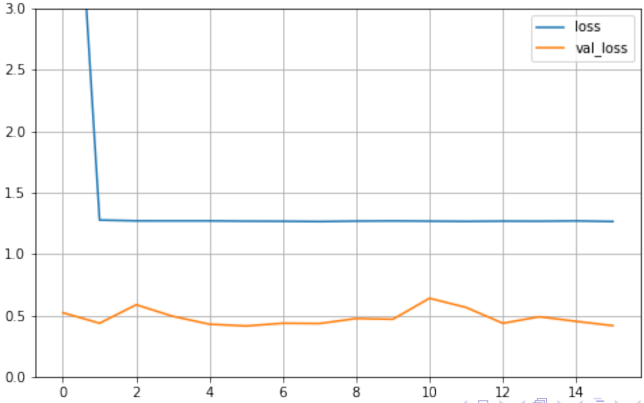
From above, we see more stable output in V2 compared to V1 — this implies that, from the simulation, an increased path size (N) at a cost of increased step size (s) could be beneficial. The rests have no contribution to a determined conclusion.

Interesting observations

MC-NN Error decomposition:

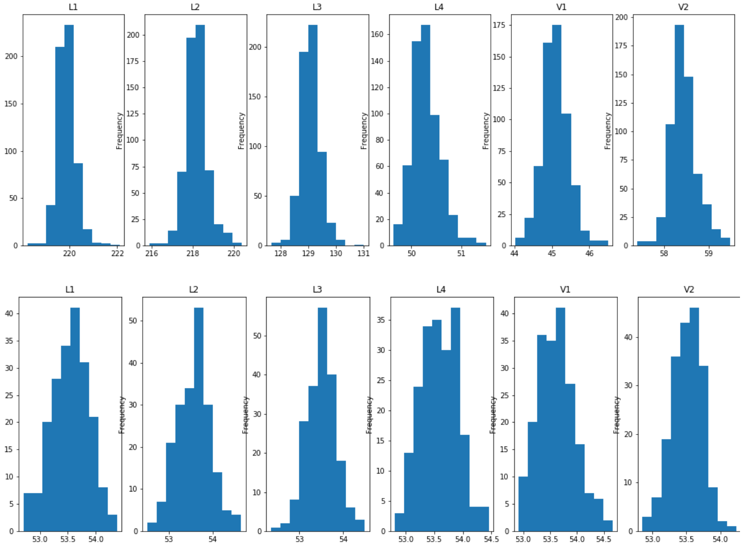
$$F_i(K;\theta_i) - \phi(K;\theta_i) = (F_i(K;\theta_i) - F(K;\theta_i)) + (F(K;\theta_i) - \phi(K;\theta_i))$$

It seems that the MC-Truth error occupies heavily in some dataset, as during the testing stage, the within-data loss could be higher than the test-data loss.



Interesting observations & Future extensions

Irreducible error on the max error — similar to Dr Hansen's arguments.



Future extensions

1. One may try approximating the true option prices via polynomial expansions.
2. Surprised about how fragile and delicate the training of NN can be. Potential extension is to back check on earlystop and potentially other training methods
3. The simulated dataset can be fructified and diversified with put options and grid sample of θ
4. It is a headache doing experiments with Colab / local machines as the computing power was so limited.