Deep Learning in Finance MAP548

Project 1 - Deep pricing and calibration

Eduardo Abi Jaber

PLEASE ENTER YOUR FULL NAMES HERE:

- MEMBER 1
- MEMBER 2
- MEMBER 3

DEADLINE: 25/02 (5:00 pm) to be sent by email to eduardo.abi-jaber@polytechnique.edu and stefano.de.marco@cmap.polytechnique.fr

PLease send both pdf ipynb files with name : GroupX_Project1

The two factor Bergomi model

The two factor Bergomi Model (L Bergomi, 2005) under the risk-neutral filtered probability space $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\geq 0}, \mathbb{Q})$ has the following dynamics:

$$
\begin{aligned}
dS_t &= S_t\sqrt{V_t}dB_t, \quad S_0 > 0;\\
X_t &= X_t^1 + X_t^2,\\
V_t &= \xi \exp\left(X_t - \frac{1}{2}\mathbb{V}[X_t]\right),\\
X_t^i &= \eta_i \int_0^t e^{-\kappa_i(t-s)}dW_s,
\end{aligned}
\tag{1}
$$

where $B = \rho W + \sqrt{1-\rho^2}W^{\perp}$ with $(W, W^{\perp})$ a two-dimensional Brownian motion, $\rho \in [-1, 1]$.

X_t^i is a Ornstein–Uhlenbeck (Gaussian) process such that $X_t^i \sim \mathcal{N}(0, \eta_i^2 \frac{(1-e^{-2\kappa_i t})}{2\kappa_i})$. Note that both X^1 and X^2 are driven by the same Brownian motion.

Recall $\mathbb{V}[(X+Y)] = \mathbb{V}[X] + \mathbb{V}[Y] + 2\mathbb{COV}[X, Y]$.

In addition, we fix $\kappa_2 = 2.6$, thus there are in total five calibratable model parameters: $(\xi, \kappa_1, \eta_1, \eta_2, \rho)$.

We are interested in computing the price of European style contingent claims, with payoff $g(S_T)$ for some function g:

$$C_t = \mathbb{E}\left[g(S_T)|\mathcal{F}_t)\right].$$

For European vanilla call options, with $g(x) = (x-K)^+$ with strike K; no closed form formula to compute $g(S_T)$

Suggested range of model parameters for training:

$\xi \in [0.03, 0.25], \kappa_1 \in [10, 60], \eta_1 \in [5, 35], \eta_2 \in [1, 5], \rho \in [-0.9, -0.1]$

Your task

Inspired by the deep pricer for the one factor Bergomi model. Implement a deep pricer for the two factor Bergomi model. You have to generate your own training set using a scheme of your choice that you have to detail (use the fixed grid for strikes and maturities below).

Once your NN is trained, showcase the train and test error, and perform a calibration on the market implied volatility surface that was used in the one factor Bergomi and comment.

Also provide/display the output prices of the NN of the set of parameters set 1, 2 and 3 below.

You have to provide a notebook that compiles, together with the trained weights of your Neural Networks that we can load with the command: *model_iv.load_weights('2FBergomiNNWeights.h5')*.

(!) The actual training of NN is not difficult, the difficult part is to get good data (and lots of it), so be careful about your simulation schemes (training might require a larger dataset than for 1 factor bergomi).

Sample IV surface values

In [40]:

```
#based on the grid:
S0 = 100
strikes=np.array([0.5,0.6,0.7,0.8,0.9,1.0,1.1,1.2,1.3,1.4,1.5 ])*S0
maturities=np.array([0.1,0.3,0.6,0.9,1.2,1.5,1.8,2.0 ])
```

parameter set 1

$\xi = 0.06, \kappa_1 = 41.6, \eta_1 = 18.2428, \eta_2 = 3.43, \rho = -0.7$

parameter set 2

$\xi = 0.1, \kappa_1 = 15.6, \eta_1 = 5.5857, \eta_2 = 2.2867, \rho = -0.4$

parameter set 3

$\xi = 0.2, \kappa_1 = 54.6, \eta_1 = 31.3496, \eta_2 = 4.5733, \rho = -0.8$

In [ ]:

In [ ]:

References

1) Lorenzo Bergomi. Smile dynamics II. Risk Magazine, 2005

2) Blanka Horvath, Aitor Muguruza, and Mehdi Tomas. Deep learning volatility: a deep neural network perspective on pricing and calibration in (rough) volatility models. Quantitativ Finance, 21(1):11–27, 20218

3) Ryan McCrickerd and Mikko S Pakkanen. Turbocharging monte carlo pricing for the rough bergomi model. Quantitative Finance, 18(11):1877–1886, 2018

```
In [ ]:
```