Interpretability in deep learning for finance: a case study for the Heston model

Alessandro Trenta

Scuola Normale Superiore

Calibration using neural networks

- A typical task is to find the correct parameters for a model, given some benchmark data such as option prices or volatilities.
- This problem can be really slow to solve: sometimes there are no analytic solutions and finding the global minimum of an optimization problem can be very time consuming.
- Using neural network for model calibration purposes is becoming popular and widely used as it is fast, robust and accurate.
- In this experiment we use the NN for the whole calibration process, from the market quotes into the model parameters.

Interpretability models

- Interpretability models aim to answer the following question: which input features affect the most our model parameters?
- Two main advantages: for models we have complete knowledge on we can verify if what the algorithm does matches our interpretation or even reject the NN architecture. On the other hand, they can be useful to better understand how a model works.
- Interpretability is still an active research: we don't know much about this concept as it differs from the simple understanding of an algorithm. We want to look at how the model works on data and how the data affect this model an the output parameters.
- Sometimes certain procedures give better accuracy but lack in interpretability.

Local interpretability

Local interpretability models aim to interpret single prediction based on some input features.
Let f be the function defining the original model we want to understand and let $\hat{f}$ be the prediction model obtained calibrating the NN. We want to interpret a single prediction $y = \hat{f}(x)$.
It is often used a simplified input x' defined by a mapping $x = h_x(x')$ which usually converts the input into a binary vector of features.

- a 0 component, means that the feature is taken as the mean of the data set. It indicates that the input is "standard" on this component.
- a 1 component indicates a feature taken as a particular value, different from the mean of the data set.

$$x = [m_1, v_2, m_3, m_4, v_5] \mapsto x' = [0, 1, 0, 0, 1] \tag{1}$$

so that

$$h_x([0, 1, 0, 0, 1]) = [m_1, v_2, m_3, m_4, v_5] \tag{2}$$

Then we call a model g **local** if $\hat{f}$ can be locally approximated by g, so that $z' \sim x'$, then $g(z') \sim \hat{f}(h_x(z'))$.
We suppose the model g has an *additive feature attribution*, so that

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z_i' \tag{3}$$

where M is the number of simplified features, ϕ_i is the contribution of a feature (ϕ_0 is the contribution of the standard vector).

Procedure

- Select an instance x whose model prediction is to be explained.
- Perturb the data set and generate prediction for these new data points.
- Set weights for the new samples based on their proximity to x.
- Train the model g with weights on the data set with variations.
- Interpret the obtain prediction by explaining the local model g.

- Local interpretable model-agnostic explanations.
- We want to minimize the objective function

$$\xi = \arg\min_{g \in \mathcal{G}} L(f, g, \pi_{x'}) + \Omega(g) \tag{4}$$

where g is the explanation model for x' that minimizes the loss function L, $\Omega(g)$ penalises the complexity of g and $\pi_{x'}$ is a set of weights for L for the samples in the neighborhood of x'.

- Solved using a penalised linear regression.

DeepLIFT and LRP

- **Deep-learning important features**: for each input x_i, the attribution $C_{\Delta x_i \Delta y}$ is computed by the effect of the input x_i with respect of a reference value in a recursive way.
- We want the condition to be satisfied $\sum_{i=1}^{n} C_{\Delta x_i \Delta t} = \Delta t$ where $t = f(x)$ is the model output and $\Delta t = f(x) - f(a)$ where a is reference value.
- $z_{ij} = w_{ji}^{(l+1,l)} x_i^{(l)}$ is the weighted activation of neuron j in layer $l+1$ by neuron i in layer l. The reference value $\bar{z}_{ij}$ is computed running the NN with the reference input $\bar{x}$.

- We proceed backwards by setting $r_i^{(L)} = S_i(x) - S_i(\bar{x})$ for unit i of the output (S_i is the output function).
- Then the backwards pass is given by

$$r_i^{(l)} = \sum_j \frac{z_{ji} - \bar{z}_{ji}}{\sum_k z_{jk} - \sum_k \bar{z}_{jk}} r_j^{(l+1)} \tag{5}$$

- Then, $\phi_i^c(x) = r_i^{(1)}$.
- **Layer-wise relevance propagation**: is a DeepLIFT where the reference activation for all neurons is set to 0.

Global interpretability

- The main method is Shapley values.
- It is based on game theory and the idea is to compute the average marginal contribution of each player i to the total game payout.
- Consider the prediction task as a game and each feature value as a player. The gain is calculated as the difference between the prediction for a particular instance and the average prediction for all instances.
- We proceed by adding a feature at each step. The values ϕ_i are calculated as

$$\phi_i(\hat{f}, x) = \sum_{z' \subseteq x' \setminus \{i\}} \frac{|z'|!(M - |z'| - 1)!}{M!} (\hat{f}(h_x(z' \cup \{i\})) - \hat{f}(h_x(z')))$$

We require some conditions

- Efficiency: the total gain is recovered

$$\sum_i \phi(\hat{f}, x) = \hat{f}(x) - \mathbb{E}[\hat{f}(x)] \tag{6}$$

- Dummy: if j never adds value, then $\phi_j = 0$.

$$\forall S \;\; f(S \cup \{x_j\}) = f(S) \implies \phi_j = 0 \tag{7}$$

- Simmetry: if two features contribute equally then they have the same value

$$\forall S \;\; f(S \cup \{x_k\}) = f(S \cup \{x_j\}) \implies \phi_k = \phi_j \tag{8}$$

- Additivity: if $f = f_1 + f_2$ then $\phi(f) = \phi(f_1) + \phi(f_2)$

- Local accuracy: given $x = h_x(x')$

$$f(x) = g(x') = \phi_0 + \sum_i \phi_i x'_i \tag{9}$$

- Missingness: feature missing in x should have no impact, so $x'_i = 0$ implies $\phi_i = 0$.
- Consistency: if we have two models, the change in values should be consistent

$$f_2(z') - f_2(z' \setminus \{i\}) \geq f_1(z') - f_1(z' \setminus \{i\}) \implies \phi_i(f_2, x) \geq \phi_i(f_1, x)$$

It is claimed but not proved that the formula for ϕ_i shown before is the only possible.

Procedure

- We add one by one the features, setting $\phi_i(f, x) = \mathbb{E}[f(z)|x_1 = z_1], \dots$
- In general, $f_x(z') = f(h_x(z')) = \mathbb{E}[f(z)|z_A]$ where A is the set of non zero features of z'.
- The exact value can be hard to compute, we can use an approximation

$$\begin{aligned} f(h_x(z')) &= \mathbb{E}[f(z)|z_A] \\ &= \mathbb{E}_{z_{\bar{A}}|z_A}[f(z)] \\ &\sim \mathbb{E}_{z_{\bar{A}}}[f(z)] \\ &\sim f([z_A, E[z_{\bar{A}}]]) \end{aligned} \tag{10}$$

The Heston model

The model is defined as

$$\begin{aligned} dS_t &= \sqrt{v_t} S_t dW_{1t}, \quad S_0 \\ dv_t &= \kappa(\theta - v_t)dt + \sigma\sqrt{v_t} dW_{2t}, \quad v_0 \\ dW_{1t} dW_{2t} &= \rho dt \end{aligned} \tag{11}$$

We set the dividen yield to zero, as well as the risk free rate. From now we assume $S_0 = 1$.
The free parameters we aim to find for generated data are $\psi = \{v_0, \rho, \sigma, \theta, \kappa\}$.

Descriptions and conditions

- $v_0 > 0$ is the initial value for the variance.
- $\rho \in [-1, 1]$ is the correlations between the two Brownian motions.
- $\sigma \geq 0$ is the volatility of volatility.
- $\theta > 0$ is the long-term mean of variance.
- $\kappa \geq 0$ is the mean-reverting speed.

We also require the Feller condition $2\kappa\theta > \sigma^2$ so that the instantaneous variance is always strictly positive.

Calibration via neural networks

Usually we aim to find the parameters which minimize the errors between the model market prices and the actual for a set of strikes and maturities.
If we consider the option prices

$$C(\psi; K, T) = \mathbb{E}^{Q(\psi)}[(S_T - K)^+] \tag{12}$$

and the mean squared error

$$L(\psi; C^{mkt}) = \frac{1}{n}\sum_{i=1}^{n}(C(\psi; K_i, T_i) - C_i^{mkt})^2 \tag{13}$$

we want to compute

$$C^{mkt} \mapsto \psi^* = \arg\min_{\psi} L(\psi; C^{mkt}) \tag{14}$$

Synthetic data sets

The input we consider will be the volatilities of market option.

- First, we generate the model parameters to be learned (train ~ 8500, test ~ 1500), selecting the ones that respect the Feller condition. The distribution is supposed to be uniform and independent between parameters with the following bounds

parameter	v_0	ρ	σ	θ	κ
lower bound	0.0001	-0.95	0.01	0.01	1.0
upper bound	0.04	-0.1	1.0	0.2	10.0

- We then calculate the volatilities for a set of strikes and maturities:

$$\begin{aligned} K &= \{0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5\} \\ T &= \{0.1, 0.3, 0.6, 0.9, 1.2, 1.5, 1.8, 2.0\} \end{aligned} \tag{15}$$

- The input dimension is then $8 \times 11 = 88$.

Some used quantities and functions

- Mean squared logarithmic error:

$$L(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} \left(\log \left(\frac{y+1}{\hat{y}+1} \right) \right)^2 \tag{16}$$

- ELU activation function:

$$\begin{cases} x & : x \geq 0 \\ \alpha(e^x - 1) & : x < 0 \end{cases} \tag{17}$$

- Hard sigmoid activation function:

$$\max \left(0, \min \left(1, \frac{1+x}{2} \right) \right) \tag{18}$$

https://gitlab.com/Alexthirty/QF_seminar

Thank you!

Temporary page!

LaTeX was unable to guess the total number of pages correctly. A
there was some unprocessed data that should have been added t
the final page this extra page has been added to receive it.
If you rerun the document (without altering it) this surplus page
will go away, because LaTeX now knows how many pages to expe
for this document.