Deep Learning Equity Derivatives

Alessio Peroni, Giovanni Amici

August 2020

Abstract

Deep Learning Equity Derivatives is a research thesis consisting in evaluating the application of Deep Learning (a branch of Machine Learning) to financial derivatives valuation, and in particular to equity derivatives (i.e. which take stocks as underlying financial instruments).

The idea of using Deep Learning in pricing comes from the fact that derivatives valuation can require a huge amount of computational time, especially if these derivatives are 'exotic' and if they are priced with a high accuracy. An exotic derivative is a financial instrument with quite a complex structure, usually meaning that there is not a closed form mathematical formula to solve it, but instead one needs to apply some numerical methods as for example a Monte Carlo simulation. Monte Carlo simulation is a technique which, in this case, permits to generate a certain number of random paths of the underlying stock price, according to some pre-defined dynamics, and then to derive the derivative price taking into consideration all of these paths.

To achieve a high pricing accuracy, one needs to apply a Monte Carlo simulation with a huge number of paths, meaning that the time spent will be high, too high to be accepted in derivatives valuation, since financial markets continuously move and even a few seconds can change the decisions to be taken when one trades.

There is a further important, and specific to this project, reason why Deep Learning could be a great alternative to traditional financial pricing. The model used in this work to evaluate equity derivatives is the Heston model, a sophisticated model that takes as input some parameters which are not observable in the market. In general, the estimation of these parameters is obtained through a calibration, which however is a difficult process that requires time.

The way in which Deep Learning can solve these two above mentioned issues is the following. Regarding the computational time, there will be a Neural Network that, after it is well trained on a dataset generated by traditional pricing, learns a model which is able to predict immediately an output value after having taken as input some set of values. The second problem is solved by the fact that the inputs provided to the NN are, instead of the Heston model parameters, just market observable values (specifically, quoted implied volatilities).

What is fundamental in this process is that every combination of parameters needed to price the derivatives is uniquely substituted by some set of NN input values. But most of all, the core evaluation of this work regards the level of pricing accuracy achieved by the NN. If the prices predicted by the NN are almost the same that traditional pricing would have provided, then it means the NN can be used and is a valid (and way better, in terms of speed) alternative of traditional pricing.

The derivatives that will be considered are both plain vanilla and exotic options. Call options (as Put options) can be priced through a semi-closed formula under the Heston model, meaning that a short time is required for their evaluation. NN can be a good pricer also for these vanilla options, given that one would avoid the calibration of Heston parameters, using instead quoted volatilities. But probably the most important role of vanilla options is that they produce the implied volatilities which become part of the NN inputs, which is useful for any kind of derivatives. In fact, the NN learns to price exotic derivatives through the implied volatilities recovered from vanilla options.

Apart from Call options, the other type of vanilla option considered is the Put option (obtained through Put-Call parity). Concerning exotic options, the dataset contains a couple of derivatives with non-path-dependent payoff, which are Digital options (still priced with semi-closed formula) and Collar options, and some with path-dependent payoff, i.e. Knock-Out, Knock-In and Double Barrier options.

Collars and path-dependent options need to be priced with numerical methods. The NN takes the previously described implied volatilities as inputs to price all these derivatives. And specifically, the NN considers a volatility 'smile', i.e. a set of implied volatilities quoted at the same maturity, which usually, when plotted, show a convex curve similar to a smile.

Before the pricing part of the dataset, a large set of input values is generated. The distribution of these values should represent a wide range of real situations, in order for NN to learn from the dataset to price in almost every market circumstances. For every derivative, there is at least a number of inputs equal to the length of the volatility smile considered plus one (the maturity). Then, it depends by the type of derivative: vanilla options and Digital options have a strike price, Collars have a high strike and a low strike, barrier options have a strike and one or two barriers.

Note that while it is important to choose the distributions of the option parameters (i.e. maturity and the specific parameters, as the strike) according to a realistic range of values of these parameters themselves, in the Heston parameters case the logic is a bit different. In fact, it is important to choose their distributions also according to the implied volatilities that are indirectly recovered from them. Once the empirical distribution of the implied volatilities well covers almost the whole space between 0 and 1, it means the distributions of Heston parameters are basically representing every market situations.

All the parameters are generated through the Latin Hypercube Sampling technique. This means that they are not generated with a Monte Carlo simulation, that would mimic randomness, but instead in a way that better covers the whole multidimensional space. Consequently, this also leads to the fact that a lower number of points are needed to well represent the space, meaning a substantial reduction in computational time to generate the dataset (and then also to train a smaller dataset).

After having completed the dataset, the ML part of the project starts, structured as a Supervised Learning problem, in a Regression case. A Deep Neural Network (DNN) is set up and its general architecture, for example in the Call option case, can be summarized as follows: it starts with an input layer having a number of nodes equal to the length of the volatility smile plus 2 (maturity and strike); then it has an arbitrary amount of hidden layers and nodes per hidden layer (numbers to be chosen after some hyper-parameter tuning); finally, an output layer composed by a single node, the Call price.

Once the NN structure is established, an extensive process composed by training, validating and testing is undertaken, along with a careful Hyper-Parameter Tuning. This implies that many different combinations of for example, number of hidden layers, number of nodes per hidden layer, number of epochs, batch size, learning rate, optimizer and loss function are analysed.

As can be noted, the core of the project is the timing. For this reason, it is fundamental to use the proper hardware to undertake the whole work. Through the use of E4 machine, there is the chance to quickly generate the dataset and most of all to train the NN model in a reasonable amount of time. Once one checks that the time spent to generate the dataset and train the model is reasonable and that the prediction of the NN model approaches the values that the Heston model would have produced, then the NN can definitely be used in real situations.