Proposal for machine learning project

Charlie Browne, Head of Risk, Quant & Market Data Solutions, GoldenSource

Lecturers:
Dr Bernard Butler
Dr Kieran Murphy

Department of Computing & Mathematics,
Waterford Institute of Technology

Data quality is about continuous improvement and is much more than a 'fit-for-purpose' check at a moment in time on a particular data set.

1 Introduction

Although there has been extensive research on derivative pricing models, no study to date has systematically reviewed the risk factor inputs to these models. My PHD research seeks to address this research gap. It investigates the nature of the relationship between derivative pricing models and their risk factor inputs. Four separate classes of risk factors are proposed. The first is market risk factors which are the set of market rates (prices, rates, volatilities) that are the building blocks for the second set, the parametric risk factors – i.e., the risk factors generated by parametric derivative pricing models. The third set are non-parametric risk factors. These are risk factors generated by models that are trained using Machine Learning (ML) techniques to re-produce the input market data of the first two categories. The fourth set is the regulatory-defined risk factors.

This essay forms part of the background research for my PhD. I was tasked with coming up with a proposal for a machine learning project related to the third, Machine Learning (non-parametric) set of risk factors. This paper contains and explains the proposal.

Background

The over-the-counter (OTC) derivatives market plays a central role in the global economy. The notional amount outstanding of OTC derivatives globally in 2021 was estimated to be $610 trillion , over six times the size of estimated global GDP . Firms face two categories of risk in the everyday running of their businesses. The first is business risk and includes commercial risks such as falling demand, increased market competition and supply chain failure. Acceptance and management of these risks is part of the role of the firm's management. Rewarding business owners for the acceptance of business risk is one of the tenets of capitalism. The second category of risks, and the category that this study will be examining, is financial market risks. These are the risks that changes in variables such as interest rates, exchange rates, stock and commodity prices negatively impact the profits of the firm. These financial market risks are also the risk factor inputs into the derivative pricing models that built upon the famous Black-Scholes-Merton (BSM) model that was first published in 1973. Building on state preference theory, the efficient market hypothesis and no-arbitrage principles, the BSM model gave investment banks the opportunity to take a leading role in offering OTC derivative products which allowed firms to more accurately manage the financial market risks they were exposed to.

My thesis commences with a review of the finance theory underpinning the BSM model. A comparison between standard finance theory and some of the pertinent behavioural finance concepts is then undertaken. A description of BSM implied volatility, one of the main risk factor categories, that has become an industry standard for quoting and pricing traded options then follows. The nature of the relationship

between derivative pricing models and their risk factor inputs follows next. As noted in Section 1, my study proposes that there are four separate classes of risk factors.

2 Derivative pricing: Parametric models

Andreou, Charalambous and Martzoukos (2008) describe the Black-Scholes-Merton (BSM) model and its various derivations as the "parametric models" as they rely on the specification of parameters for the underlying diffusion processes. In a similar context, Amilon (2003) refers specifically to the parameterization of the diffusion processes. His research compares the BSM model to a neural network approach to pricing derivatives and includes a discussion on the background to the BSM model. The parameterized diffusion process for the underlying stock price that BSM is based on is referred to as Geometric Brownian Motion (GBM). A remarkable aspect of the BSM model is the fact that a closed-form solution, generally known as the Black-Scholes formula, exists for it. Most derivative pricing models that assume diffusion processes for their input parameters do not have closed-form solutions. Numerical techniques such as Monte Carlo simulation are required to solve them.

Amilon (2003) describes the breakthrough role that the BSM model played in derivative pricing despite the unrealistic assumptions that the model is based on. Since the early days of BSM much research has been done on models that relax some of its assumptions. The most notable being the development of models, known as stochastic volatility models, which allow the volatility parameter to be non-constant. In such models a transform of the volatility parameter is described by a stochastic diffusion process. A minimum of two diffusion processes are therefore assumed by stochastic volatility models: one for the underlying price and another for the volatility of that underlying price. Stochastic volatility models achieve a better approximation to traded market prices than the BSM model does (Bakshi, Cao, and Chen, 1997).

Andreou et al (2018) argue that the various models that were developed to account for the flaws in BSM (e.g. stochastic volatility models, jump diffusion models, and models that assume interest rates have a term structure and are stochastic) have not been successful in doing so – despite their improvements on the original BSM model. Derivative pricing models today still do not produce prices that are consistent with prices observed in financial markets. They offer a number of reasons as to why this is the case. They have poor out-of-sample pricing performance, model parameters are sometimes implausible and inconsistent with market data (Bakshi et al, 1997), and the complexity of the models often makes them difficult to implement. For these reasons and others, the BSM model, despite its many flawed assumptions, remains ubiquitous in financial markets.

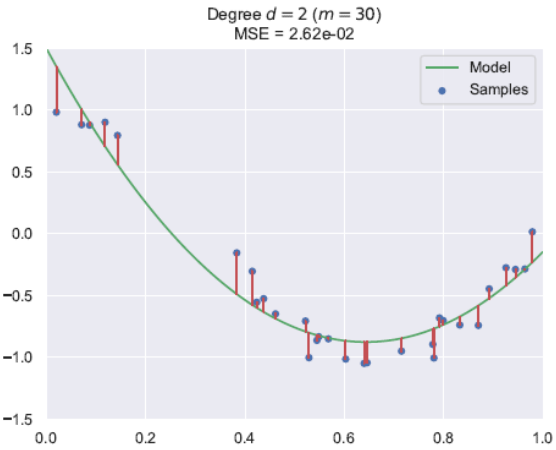
3 Derivative pricing: Non parametric models

In their seminal work on pricing derivatives using learning networks Hutchinson, Lo and Poggio (1994) refer to their approach as non-parametric. Since that time ML approaches to derivative pricing have been referred to as the "non-parametric" models (Amilon, 2003; Andreou et al, 2008; Park, Kim and Lee, 2014; Culkin and Das, 2017; Ye and Zhang, 2019). Ye and Zhang (2019) describe the specific approach proposed in Hutchinson et al (1994) as brute-force curving fitting. Pearl (2019) and Bishop (2021), in reference to this relationship between curve fitting and ML, argue that much of the recent progress that has been made in ML is founded on curve-fitting approaches that have been around for decades.

Andreou et al (2008) argue that non-parametric techniques such as Artificial Neural Networks (ANNs) are effective alternatives to the parametric derivative pricing models. While finance theories such as the Efficient Market Hypothesis (Samuelson, 1965; Fama, 1970), No Arbitrage (Ross, 1976b) and the Capital Asset Pricing Model (Sharpe, 1964) underpin much of the literature on parametric derivative pricing models, ANNs on the other hand are not required to sit on any particular theoretical framework. ANN estimates calculate the derivative's price inductively using historical or implied input variables or option transactions data. Ye and Zhang (2019) refer to the extensive research on the use of ML techniques on financial market variables. A natural consequence of introducing ML and artificial intelligence (AI) to the finance industry is the desire to use its tools to predict market prices for variables like stock prices, FX rates and interest rates. Aydogdu (2018), for example, uses neural networks to forecast stock price returns.

Given the fundamental role that curve-fitting plays in ML, there is a certain irony in the terminology that has evolved to describe ML approaches to derivative pricing. These "non-parametric" ML approaches to pricing options and other derivatives are in fact often based on highly parameterised functions. In ML, "given data and a parameterised function, curve fitting is the problem of determining function parameters so that the function output matches desired output as closely as possible. " The parameterised function in this sentence refers to the mathematical models that underlie the ML algorithm. "Determining function parameters" is the mathematical description of the process for training the ML model and the "function output" refers to the predictions of the ML model. Figure 1 is an illustration of a quadratic curve-fitting process for 30 observations (rows of data). Models are trained by fitting a curve to the data using techniques that minimise the loss functions associated with distances between observations and the curve.

Figure 1: Curve fitting processes in machine learning



Culkin and Das (2017) attribute much of the success of ML to the improvements in mathematical methods underpinning Artificial Neural Networks (ANNs). The improvements allowed for extremely fast training of the models. ANNs are trained by minimizing a loss function using the appropriate optimization technique. Large ANNs require the fitting of high volumes of parameters. Using gradients for optimization is computationally very expensive. Gradient approaches are numerical in nature and take too long to run for most applications. However, the discovery the backpropagation technique meant that parameters of the neural network could be solved for without using numerical approaches. Optimization times sped up dramatically as a result. Schmidhuber (2014) credits the discovery of the backpropagation technique to several researchers including Kelley (1960) and Bryson (1961). According to Culkin and Das (2017), without the backpropagation technique deep learning approaches would not exist. The availability of analytical solutions via backpropagation enabled the advancement of deep learning.

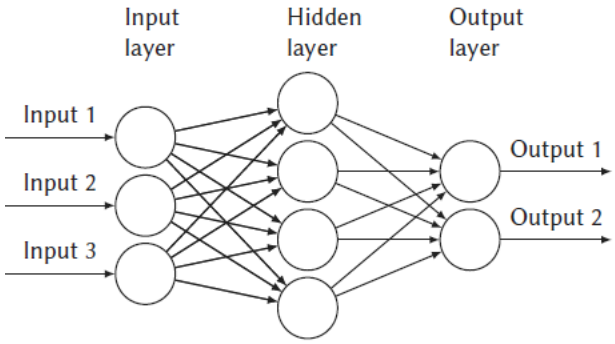
Table 1 contains the five variants of ML algorithms as described in Domingos (2015). He refers to these variants as the five tribes of ML. Artificial Neural Network (ANN) approaches belong to the Connectionists tribe and have origins in neuroscience.

Table 1: The five tribes of Machine Learning (Domingos, 2015)

Tribe	Origins	Learning Algorithm
Symbolists	Logic, Philosophy	Inverse Deduction
Connectionists	Neuroscience	Backpropagation
Evolutionaries	Mathematical Biology	Genetic Programming
Bayesians	Statistics	Probabilistic Inference
Analogizers	Psychology	Kernel Machines

Pierrot (2020) states that ANNs are models inspired by biological neural networks that can be described using three functional abstractions: neurons, layers, and global architecture. Figure 2 illustrates the Multi-Layer Perceptron (MLP), a fundamental concept in neural networks that shows the relationship between these components. Andreou et al (2018) describe a neural network as a set of processing functions that exist in multiple layers that are connected via arcs.

Figure 2: The Multi-Layer Perceptron (MLP), Pierrot (2020)



Amilon (2003) describes neurons as the central commodity of all ANNs. They exist in layers within the MLP. There is an input layer, several hidden layers and an output layer. The neurons in the hidden layers are used to generate a representation of the data. The role of each neuron is to sum the signals that it receives and then, after adding a bias term, perform a transformation using a monotonically increasing function such as a sigmoid or the logistic function. In this way the signals are passed through the layers of the MLP and the process is repeated. The connections between the neurons are represented by weights. The outputs of the ANN are validated against known targets. Differences between the outputs and the target are registered and used in a loss function such as the sum of squared errors.

Pierrot (2020) discusses the weights associated with the connections between neurons. He describes how each neuron is connected to a neuron in the next layer by a learnable weight, a bias term and an activation function. Backpropagation works backwards from the output error, adjusting weights and biases so that the

loss function – which is generally a gradient descent algorithm - is minimized. The problem is often specified as a supervised regression problem with the output loss being the Mean Squared Error (MSE).

Pierrot (2020) discusses the use of hyperparameters in an ANN. Hyperparameters are settings that define the model architecture and control the way that the learning process works. Examples of hyperparameters are the number of neurons, the number of layers, the learning rate, the number of principle components where Principal Component Analysis (PCA) is being used as part of the modelling process, and the batch size. They should be set so that the performance of the model is optimized. Amilon (2003) discusses approaches for optimizing model performance. Using the input data for both fitting the model and evaluating it risks generating optimistic (and therefore erroneous) estimates of model performance with the sub-optimality of the model only revealing itself when it is asked to predict on new unseen data

Amilon (2003) states that a popular approach to avoiding such performance issues is to split the data into a training set - used for estimating weights, and a test set – used for evaluating the model. Figure 3 illustrates the concept. Pierrot (2020) describes the use of a more advanced cross-validation approach to train and test the model. The training data set is divided into k subsets of data called folds. The model iterates through the folds and is trained on the data from all folds except the current one. Predictions are generated for the data from that fold. The hyperparameters that obtained the best score are used to train the final model. Figure 4 illustrates the K-fold cross-validation approach.

Figure 3: Splitting the data set

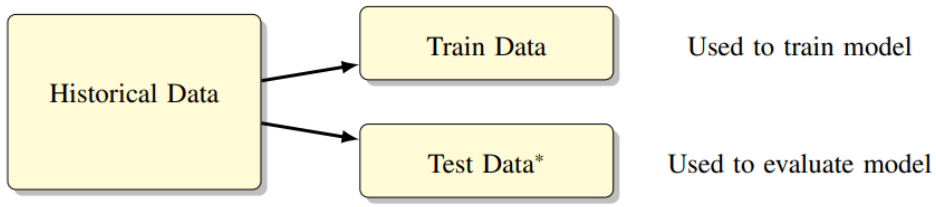
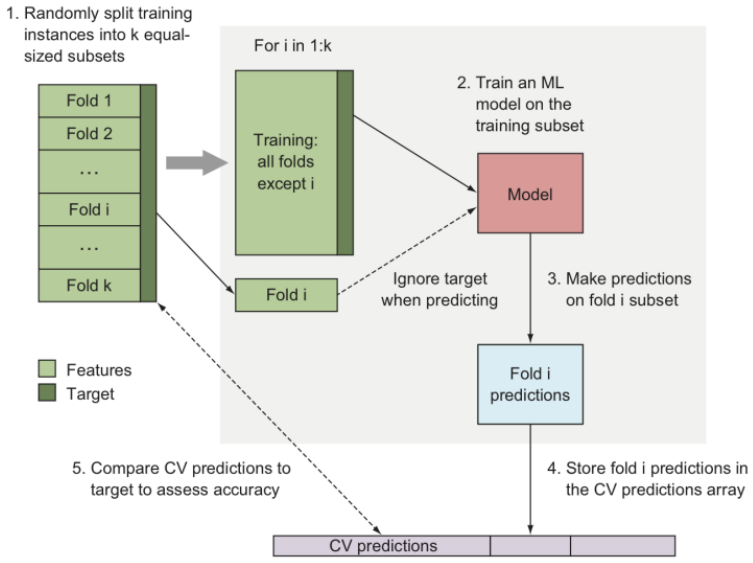
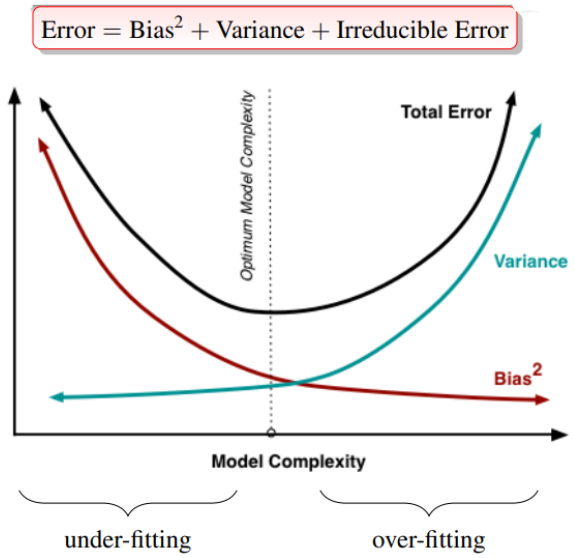


Figure 4: K-fold cross validation



Amilon (2003) discusses the MLP's ability to model functions of any type and refers to the description by Hornik, (1989) of the MLP as a universal approximator. However, where too many layers are added, overfitting will result as a consequence of outliers. The modelling process is therefore a trade-off between high model complexity which increases accuracy (reduces bias) but risks over-fitting, and low model complexity which reduces accuracy (increases bias) but risks under-fitting. Figure 5 provides a graphical illustration of this trade-off.

Figure 5: Under versus Over fitting



4 The Machine Learning Approach for training derivative pricing models

Andreou et al (2018) analyse multiple ANN approaches for pricing derivatives and compare these to the BSM-based parametric and semi-parametric approaches described in Section 2. They describe the hybrid neural network option pricing approach by Lajbcygier et al. (1997). It is based on the original ANN target function by Watson and Gupta (1996). In these hybrid models the target function calculates the residuals between market prices of options and the option prices calculated using the parametric approach. They use the prices of European call options on the S&P 500 stock index to analyse differences between the different methods and conclude that the BSM-based hybrid ANN models outperform both the BSM-based parametric models and the regular ANN models.

Pierrot (2020) describes five broad set of steps required to train a derivative pricing model using ML. The first step is the generation of model parameters using random sampling techniques. The second step is the use of derivative pricing models to derive the price of the option for each parameter in the model. The third step involves the splitting of the data created by the second step into a training sample – consisting of 70% of the data, and a testing sample – consisting of 30% of the data. The fourth step is to use the training sample to train the ML model to map the model parameters, generated in the first step, to the option prices from the second step. The last step is to use the trained model on the test set to validate that it generates option prices of the required accuracy.

5 Machine learning project

My data mining project will use an ANN to generate an Implied Volatility (IV) surface for IBM, an S&P 500 stock. The approach will broadly follow the five steps outlined in Pierrot (2020) in Section 4 but with two modifications. For the first step I will use the tick history of the stock as the sample inputs to create a distribution of prices that will be used in the generation of derivative pricing model parameters . For the last step, instead of the final output of the deep learning process being BSM option prices for the IBM stock, the output will be the BSM implied volatility (IV) surface for IBM. I choose this approach because my PHD is focused on derivative pricing risk factors – and the BSM IV surface is exactly that, a derivative pricing risk factor.

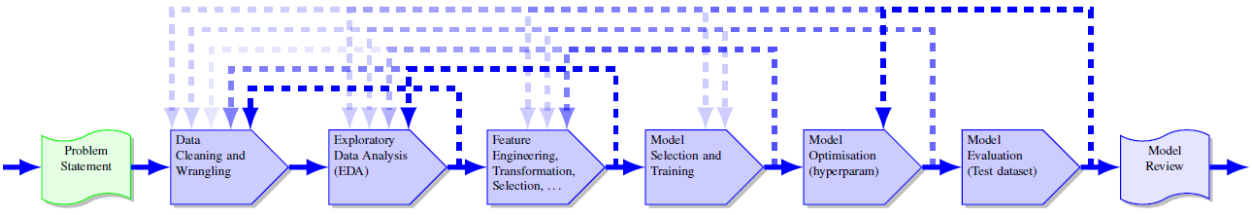
The derivative pricing model I will use to generate the BSM IV surface will be the Heston model (Heston, 1993) which is an example of a stochastic volatility model described in Section 2. The Heston parameters, generated in Step 1 of the Pierrot approach are standard per the Heston model, i.e., they are initial variance, long-term variance, reversion speed, gamma, and correlation. The IBM BSM IV surface will be calculated by transforming Heston option prices, calculated using the Heston parameters, into points on the IBM IV surface. Each option price is transformed into an IV surface point using a root-finding method. The points on the IV surface points are identified by moneyness and maturity. Per Pierrot (2020), the MLP architecture will have three layers and the hyper-parameters will be optimised using the Bayesian search approach.

Highly traded, liquid stocks like IBM generate stock price history (i.e., tick history) on a second-by-second or more frequent basis. I will use the last 1,000,000 ticks available from the relevant tick history database as the tick history time-series. This time-series will be used as the stock price distribution required for the generation of Heston parameters. The tick history is expected to have a number of data quality issues including missing prices, outliers and stale prices.

The tick history will be read into a Pandas data frame resulting in a labelled data set of one million rows and five or six columns. The columns will include the various price types that typically comes with tick history columns including bid, ask and traded price as well as trade volume and trade/quote time.

I will follow a standard data mining workflow such as the one described in Figure 6 to achieve my overall data mining objective. Other potential data mining workflows that could be followed are Knowledge Discovery in Data (KDD) or Cross Industry Standard for Data Mining (CRISP-DM).

Figure 6: Data Mining Workflow



The first step in the data mining workflow is data cleaning and wrangling. Table 2 describes the proposed sub-sets required to clean the tick history.

Table 2: Data cleaning and wrangling on the tick history

Step	Sub Step
Data Cleaning & Wrangling	Sense-check the data. Check its shape and view the first few rows.
	Check for missing values. Identify the time-series window, e.g., 1-minute, as the largest expected time-gap that should elapse without a price tick. Missing ticks should be filled with a single price inserted into the middle of the price-less 1-minute time-series window. The process is referred to as gap-filling. Three options will be available for gap-filling the tick history: linear interpolation, copy-forward or proxy. The proxy method will identify an expected change in price (in percentage terms) of a proxy security. The proxy security needs to be highly correlated (Pearson correlation) with IBM. The expected change in price will then be applied to derive the tick prices missing for IBM.
	Check for outliers. An outlier will be defined as a price change between one tick and the next that is greater than one standard deviation. Standard deviation will be calculated using the last 60 tick price changes. Where an outlier is detected, it will be replaced with a price calculated using one of the three approaches described above (linear interpolation, copy-forward or proxy).
	Check for stale prices. Pandas commands will be run to check for the occurrence of potential stale prices. A stale price will be defined as a price that is the same as the previous price in the time-series. It is difficult to know whether stale prices are actually stale (i.e., erroneous) or whether the price has just coincidentally ticked to the same price as the previous price. Unless the level of stale prices is material stale prices will not be replaced using any of the methods described above.

Once data cleaning and wrangling is done, the next step in the data mining workflow is the EDA and feature engineering. Table 4 outlines the sub-steps involved here.

Table 4: EDA and feature engineering on the tick history

Step	Sub Step
EDA	Dtale, Pandas Profiling are two potential tools that could be used for the EDA. However, both are likely to be very slow processing the tick history given the volume of data.
	Fix data types. Most if not all fields will be numerical. Check that the data types for these are all float64.
	Check the statistical distributions of the price fields (bid, ask, etc). The price fields should be lognormally distributed. The returns (percentage or log returns) should be normally distributed
	Bid-ask spread analysis. Calculate a new bid-ask spread field [Ask price – Bid price] using Pandas. Generate a histogram to view the distribution of the bid-ask spread. Use the Seaborn library to create a boxplot of the distribution. Check for outliers using the 1.5 inter-quartile range rule.
	Examine the correlations of potential proxy instruments. Load tick history for other instruments that are expected to be positively correlated with the tick history of the IBM stock, e.g., other instruments that belong to the S&P 500 index and that are in the technology sector. If correlations are not strongly positive, investigate the reasons.

Once data wrangling and EDA are complete, then model selection/training, model optimisation and model evaluation will be done – as per the data mining workflow These latter steps will follow the process outlined in Sections 3 and 4 of this document.