Beyond Black-Scholes: A New Option for Options Pricing

By Balazs Mezofi and Kristof Szabo

Advances in computing and machine learning have made it possible to employ a new, data-driven approach to pricing options.
In 1973, Fischer Black, Myron Scholes and Robert Merton published their now-well-known options pricing formula, which would have a significant influence on the development of quantitative finance. In their model (typically known as Black-Scholes), the value of an option depends on the future volatility of a stock rather than on its expected return. Their pricing formula was a theory-driven model based on the assumption that stock prices follow geometric Brownian motion. Considering that the Chicago Board Options Exchange (CBOE) opened in 1973, the floppy disk had been invented just two years earlier and IBM was still eight years away from introducing its first PC (which had two floppy drives), using a data-driven approach based on real-life options prices would have been quite complicated at the time for Black, Scholes and Merton. Although their solution is remarkable, it is unable to reproduce some empirical findings. One of the biggest flaws of Black-Scholes is the mismatch between the model volatility of the underlying option and the observed volatility from the market (the so-called implied volatility surface).

Today investors have a choice. We have more computational power in our mobile phones than state-of-the-art computers had in the 1970s, and the available data is growing exponentially. As a result, we can use a different, data-driven approach for options pricing. In this article, we present a solution for options pricing based on an empirical method using neural networks. The main advantage of machine learning methods such as neural networks, compared with model-driven approaches, is that they are able to reproduce most of the empirical characteristics of options prices.

**Introduction to Options Pricing**

With the financial derivatives known as options, the buyer pays a price to the seller to purchase a right to buy or sell a financial instrument at a specified price at a specified point in the future. Options can be useful tools for many financial applications, including risk management, trading and management compensation. Not surprisingly, creating reliable pricing models for options has been an active research area in academia.

One of the most important results of this research was the Black-Scholes formula, which gives the price of an option based on multiple input parameters, such as the price of the underlying stock, the market’s risk-free interest rate, the time until the option expiration date, the strike price of the contract and the volatility of the underlying stock. Before Black-Scholes, practitioners used pricing models based on the put-call parity or an assumed risk premium similar to the valuation of investment projects. In corporate finance, one of the most frequently used models for the valuation of companies is the discounted cash flow model (DCF), which calculates the present value of a company as the sum of its discounted future cash flows. The discount rate is based on the perceived risk of investing capital in that company. The revolutionary idea behind Black-Scholes was that it is not necessary to use the risk premium when valuing an option, as the stock price already contains this information. In 1997, the Royal Swedish Academy of Sciences awarded the Nobel Prize in economic sciences to Merton and Scholes for their groundbreaking work. (Black didn’t share in the prize. He died in 1995, and Nobel Prizes are not awarded posthumously.)

If all option prices are available in the market, Black-Scholes can be used to calculate the so-called implied volatility based on option prices, as all the other variables of the formula are known. Based on Black-Scholes, the implied volatility should be the same for all strike prices of the option, but in practice researchers found that the implied volatility for options is not constant. Instead, it is skewed or smile-shaped.

Researchers are actively seeking models that are able to price options in a way that can reproduce the empirically observed implied volatility surface. One popular solution is the Heston model, in which the volatility of the underlying asset is determined using another stochastic process. The model, named after University of Maryland mathematician Steven Heston, is able to reproduce many empirical findings — including implied volatility — but not all of them, so financial engineers have used different advanced underlying processes to come up with solutions to generate empirical findings. As the pricing models evolved, the following difficulties arose:

The revolutionary idea behind Black-Scholes was that it is not necessary to use the risk premium when valuing an option, as the stock price already contains this information.
• The underlying price dynamics got more complex mathematically and became more general — for example, using Lévy processes instead of Brownian motions.
• The pricing of options became more resource intensive. Though the Black-Scholes model has a closed-form solution for pricing European call options, today people usually use more computationally intensive Monte Carlo methods to price them.
• It takes deeper technical knowledge to understand and use the pricing models.

Applying machine learning methods to options pricing addresses most of these problems. There are different algorithms that are able to approximate a function based on the function’s inputs and outputs if the number of data points is sufficiently large. If we see the option as a function between the contracted terms (inputs) and the premium of the option (output), we can simply ignore all of the financial questions related to options or stock markets. Later we will see how adding some financial knowledge back into the model can help improve the accuracy of the results, but on the basic level no finance-related information is needed.

One of these approximation techniques uses artificial neural networks, which have a number of useful properties. For example, some members of artificial neural networks are universal approximators — meaning that if the sample is large enough and the algorithm is complex enough, then the function that the network learned will be close enough to the real one for any practical purpose, as showed by George Cybenko (1989) and Kurt Hornik, Maxwell B. Stinchcombe and Halbert White (1989). Artificial neural networks are suitable for large databases because the calculations can be done easily on multiple computers in parallel. One of their most interesting properties is duality in calculation speed: Although the training can be quite time-consuming, once the process is finished and the approximation of the function is ready, the prediction is extremely fast.

Neural Networks

The essential concept of neural networks is to model the behavior of the human brain and create a mathematical formulation of that brain to extract information from the input data. The basic unit of a neural network is a perceptron, which mimics the behavior of a neuron and was invented by American psychologist Frank Rosenblatt in 1957. But the potential of neural networks was not unleashed until 1986, when David Rumelhart, Geoffrey Hinton and Ronald Williams published their influential paper on the backpropagation algorithm, which showed a way to train artificial neurons. After this discovery, many types of neural networks were built, including the multilayer perceptron (MLP), which is the focus of this article.

The MLP is made up of layers of perceptrons, each of which has an input: the sum of the output of the perceptrons from the previous layer multiplied by their weights; it can be different for each perceptron. The perceptrons use a nonlinear activation function (like the S-shaped sigmoid function) to transform the input signals into output signals and send these signals into the next layer. The first layer (the input layer) is unique; perceptrons in this layer have just an output, which is the input data. The last layer (the output layer) is unique in the sense that in regression problems it usually consists of a single perceptron. Any layers between these two layers are usually called hidden layers. For an MLP with one hidden layer, the visualization is as follows in Figure 1.
Figure 1 can be written mathematically between the hidden layer and the input layer as:

\[ h_{i,t} = f_1 \left( \alpha_{i0} + \sum_{i=1}^{n} \alpha_{ij} \cdot x_{j,t} \right) \]

and between the final output and the hidden layer as:

\[ y_t = f_2 \left( \beta_0 + \sum_{i=1}^{m} \beta_i \cdot h_{i,t} \right) + \epsilon_t \]

where \( f_1 \) and \( f_2 \) are activation functions, \( \alpha \) and \( \beta \) contain weight matrices between layers, and \( \epsilon \) is an error term with 0 mean.

The first step of the calculation is to randomly initialize the weight matrices; this process will be used to transform the input variables to the forecasted output. Using this output, the value of the loss function can be calculated, comparing the real and the forecasted results using the training data. The backpropagation method can be used to calculate the gradients of the model, which then can be used to update the weight matrices. After the weights have been updated, the loss function should have a smaller value, indicating that the forecasting error on the training data has been decreased. The previous steps should be repeated until the model converges and the forecasting error is acceptable.

Although the previous process may seem complicated, there are many off-the-shelf programming packages that allow users to concentrate on the high-level problem instead of the implementation details. The user’s responsibility is to convert the input and output data to the correct form, set the parameters of the neural network and start the learning phase. Typically, the most important parameters are the number of neurons in each layer and the number of layers.

**Pricing Options with Multilayer Perceptrons**

As shown previously, the classical options pricing models are built on an underlying process that reproduces the empirical relationship among option data (strike price, time to maturity, type), underlying data and the premium of the option, which is observable in the market. Machine learning methods do not assume anything about the underlying process; they are trying to estimate a function between the input data and premiums, minimizing a given cost function (usually the mean squared error between the model price and the observed price on the market) to reach good out-of-sample performance.

There is an evolving literature applying other data science methods, such as support vector regression or tree ensembles, but neural networks like multilayer perceptrons generally fit well for options pricing. In most cases, the option premium is a monotonic function of the parameters, so only one hidden layer is needed to deliver high precision and the model is harder to overtrain.

Using machine learning for pricing options is not a new concept; two of the relevant early works were created in the early 1990s to price index options on the S&P 100 and the S&P 500. These methods are convenient nowadays thanks to the availability of several software packages for neural networks. Although pricing options became easier, it is still slightly more complicated than loading the input data (options characteristics, data of the underlying asset) and target data (premiums) and pressing “enter.” One problem remains: designing the architecture of the neural network and avoiding overfitting the model.

Most machine learning methods are based on an iterative process to find the appropriate parameters in a way that minimizes the difference between the results of the model and the target. They usually start by learning meaningful relationships, but after a while they are minimizing only the sample-specific error and reducing the general performance of the model on unseen out-of-sample data. There are many ways to handle this problem; one of the popular ones is early stopping. This method separates the original training data into training and validation samples, instructing the model only on the training data and evaluating it on the validation sample. At the beginning of the learning process, the error of the validation sample decreases synchronously with the error of the training sample, but later the training and validation samples start to diverge; the error decreases only in the training sample and increases in the validation sample. This phenomenon signals the overfitting of the parameters, and the process should be stopped at the end of the synchronously decreasing phase.

Models that have more parameters can be overfitted more easily, so the number of the perceptrons and layers should be balanced between learning the important features and losing some precision because of overfitting. The learning rate determines how much to modify the parameters in each iteration; it is an important setting and must be set manually. Sometimes these
metaparameters are decided based on validation errors; choosing them is more art than science. Picking the “best” parameters can yield better results, but the accuracy gained during fine tuning usually diminishes, so the trained model is good enough to use after just a few trials.

**Improving Performance**

The above-mentioned methods can be generally used for improving neural network models. In many cases, adding problem-specific knowledge (in this case, financial knowledge) can improve the performance of the model. At this point, the MLP has already learned a good approximation of the options pricing formula, but the precision is determined by the sample size (which is usually fixed) and the input variables. From here, there are three ways to further improve performance:

1. Add more input variables that help the model to better understand the options pricing formula.
2. Increase the quality of the input variables by filtering outliers.
3. Transform the function in a way that it is easier to approximate.

The first approach is quite straightforward. Introducing a new variable into the model increases its complexity and makes it easier to overfit. As a result, each new variable has to increase the predictive power of the model to compensate for the increased number of parameters. And because option prices are dependent on the expected volatility of the underlying security in the future, any variable that acts as a proxy for the historical or implied volatility usually makes the MLP more precise. To improve the accuracy, Loyola University Chicago professors Mary Malliaris and Linda Salchenberger suggested adding the delayed prices of the underlying security and the option.

The second method is to increase the quality of the input variables. Because the prices of less liquid options typically contain more noise than do more-liquid ones, filtering out those options should improve the accuracy of the pricing model. But if we would like to estimate the premium for deep-in-the-money or out-of-the-money options, this cleaning method could eliminate a significant part of the used dataset. Thus, it is important for researchers to choose filtering criteria that are the optimal choice between dropping outliers and keeping the maximum amount of useful information.

The third approach — where art takes over methodology — raises an open question: If the neural network can approximate any function, then what should we forecast? This is the point at which there is the least amount of consensus among practitioners.

The problem is clear: We need a final output from the neural network that says how much an option with the input parameters is worth. That, however, does not mean the final price is the best target to aim for. The question is less relevant when we have a large sample size. When the dataset is small, choosing the best way to measure a function can increase the precision further. The most frequently chosen solutions are the following:

**Predict the premium of the option directly, potentially using the information we have from mathematical models — for example, adding implied volatility to the input variables.**

Even if we successfully minimize the error for the function of the premium, that does not mean that after transforming it to the final prediction the errors will still be the best achievable for the premium. By predicting the premium directly, we are forcing the best result.

**Predict the implied volatility of the option, and put it back into the Black-Scholes formula.** That should make the premium readable. The big advantage here is that the different target variable is in the same range of values even if the premium of the option is different in magnitude. Other Black-Scholes variables can be used to try to predict options premiums, but implied volatility is the most popular among them.

**Estimate the ratio between the option premium and the strike price.** If the underlying options prices behave like geometric Brownian motions, that property can be used to reduce the number of input parameters. In this case, the researcher would use the ratio between the underlying price and the strike price as one of the input parameters instead of using the underlying and strike prices separately. This solution can be very
When the dataset is small, choosing the best way to measure a function can increase the precision further.

useful if the size of the dataset is small and you are more exposed to overfitting problems.

While there is disagreement about which function researchers should try to predict, there is a second debate about whether the dataset should be split into subsets based on various qualities. Malliaris and Salchenberger argue that in-the-money and out-of-the-money options should be split into different datasets. From a practical point of view, this approach can be useful because the magnitude of the option premiums can be very different in the two groups. Sovan Mitra, a senior lecturer in mathematical sciences at the University of Liverpool, contends that if the data are split into too many parts, the chance of overfitting increases and the model’s precision on out-of-sample results is reduced.9

The world has come a long way since Black, Scholes and Merton published their seminal papers on options pricing in 1973. The exponential growth in computational power and data, particularly over the past decade, has allowed researchers to apply machine learning techniques to price derivatives with a precision unforeseen in the ’70s and ’80s. Back then, options pricing was driven mainly by theoretical models based on the foundation of stochastic calculus. In this article, we provide an alternative method that uses machine learning, in particular neural networks, to price options with a data-driven approach. We believe that this approach could be a valuable addition to the tool set of financial engineers and may replace traditional methods in many application areas.4

Balazs Mezofi is a Quantitative Researcher at WorldQuant, LLC, and has an MSc in Actuarial and Financial Mathematics from Corvinus University in Budapest.

Kristof Szabo is a Senior Quantitative Researcher at WorldQuant, LLC, and has an MSc in Actuarial and Financial Mathematics from Eötvös Loránd University in Budapest.

ENDNOTES


Thought Leadership articles are prepared by and are the property of WorldQuant, LLC, and are being made available for informational and educational purposes only. This article is not intended to relate to any specific investment strategy or product, nor does this article constitute investment advice or convey an offer to sell, or the solicitation of an offer to buy, any securities or other financial products. In addition, the information contained in any article is not intended to provide, and should not be relied upon for, investment, accounting, legal or tax advice. WorldQuant makes no warranties or representations, express or implied, regarding the accuracy or adequacy of any information, and you accept all risks in relying on such information. The views expressed herein are solely those of WorldQuant as of the date of this article and are subject to change without notice. No assurances can be given that any aims, assumptions, expectations and/or goals described in this article will be realized or that the activities described in the article did or will continue at all or in the same manner as they were conducted during the period covered by this article. WorldQuant does not undertake to advise you of any changes in the views expressed herein. WorldQuant and its affiliates are involved in a wide range of securities trading and investment activities, and may have a significant financial interest in one or more securities or financial products discussed in the articles.