Data-driven option pricing using single and multi-asset supervised learning

Hcine Turki

July 18, 2023

Outline

1 Definitions

-What is option pricing?

Option pricing theory estimates a value of an options contract by assigning a price, known as a premium, based on the calculated probability that the contract will finish in the money (ITM) at expiration. Essentially, option pricing theory provides an evaluation of an option's fair value, which traders incorporate into their strategies. Models used to price options account for variables such as current market price, strike price, volatility, interest rate, and time to expiration to theoretically value an option.

-Data-driven option pricing

Data-driven option pricing refers to the process of determining the fair value of financial options using empirical data and statistical techniques, as opposed to relying solely on traditional pricing models such as the Black-Scholes model. In this approach, historical market data and other relevant information are used to estimate the probabilities and distributions of future price movements. Traditional option pricing models, like the Black-Scholes model, assume certain idealized market conditions and mathematical formulas to determine option prices. However, these models may not always accurately capture the complex dynamics of real-world financial markets. Data-driven option pricing attempts to address this limitation by incorporating actual market data into the pricing process.

Black-Scholes option pricing

It's a widely used mathematical model for option pricing. The model assumes certain idealized market conditions and provides a formula to calculate the theoretical price of a European-style option. The following assumptions are made:

1. No dividends are paid out during the life of the option.
2. Markets are random (i.e., market movements cannot be predicted).
3. There are no transaction costs in buying the option.
4. The risk-free rate and volatility of the underlying asset are known and constant.
5. The returns of the underlying asset are normally distributed.
6. The option is European and can only be exercised at expiration.

1. Definitions

2. Machine Learning techniques
   - Extreme gradient boosting
   - 2.2 Feed forward Neural Network

3. Data

4. 4 Model I/O

2 Supervised machine learning algorithms

Consider a finite labeled dataset represented as $\{(X_1, Y_1), (X_2, Y_2), \ldots, (X_J, Y_J)\}$, where the vector X_j is associated with a label Y_j. The supervised machine learning algorithms attempt to find a mapping $f : X_j \to Y_j$ such that the mapping obtained is the "best" out of all the possible mappings. A qualitative assessment of the mapping (also referred to as a model) is made possible by an "objective" function (also known as a "loss function"). The specifics of the objective function and the strategy used to create the mappings vary with the choice of the algorithm.

2.1 Extreme gradient boosting

Extreme Gradient Boosting combines two powerful techniques, namely "boosting" and "gradient descent". Gradient boosting involves constructing an ensemble of "weak" learners, which in the case of XGBoost, are decision trees. These "weak" learners are combined in an iterative fashion to obtain a "strong" learner. XGBoost can be used for classification. The XGBoost algorithm, for a set of N output classes, assigns a score $F_i(x)$ to the ith class for the input x. We define $F(x)$ as $(F_1(x), F_2(x), F_3(x), \ldots, F_N(x))$. The softmax function is then used to calculate the probability of x being in class i. $P_i(x) := \frac{e^{F_i(x)}}{\sum_{k=1}^{N} e^{F_k(x)}}$, for $i = 1, 2, \ldots, N$.

The Categorical Cross Entropy loss function is used:
$L(z, F(x)) := -\sum_{i=1}^{N} z_i \log(P_i(x))$, where $z := (z_1, z_2, \ldots, z_N)$ is the probability mass function of the true outputs.
XGBoost tries to minimize the loss function given pairs of input-outputs: $\{(x_j, y_j) \mid j = 1, 2, 3, \ldots, J\}$.
We use the data to compute $z_{(j)i} = \delta(y_j, i)$ so $z_{(j)} = (0, \ldots, 1, \ldots, 0)$.
XGBoost then uses the gradient boosting algorithm to obtain an approximation of the minimizer $F_b(\cdot) := \arg\min F(\cdot)$, where

$$F(\cdot) = \frac{1}{J} \sum_{j=1}^{J} L(z_{(j)}, F(x_j))$$

Step1: The algorithm first computes
$F^{(0)}(\cdot) := \arg\min_{\gamma \in \mathbb{R}^N} \sum_{j=1}^{J} L(z_{(j)}, \gamma)$
Making some simplifications we get:
$L(z_{(j)}, \gamma) = -\sum_{i=1}^{J} z_{(j)i} \log(p_i(\gamma)) = -\log(P_j(\gamma))$
Hence, the model is initialized with
$F^{(0)}(\cdot) := \arg\min_{\gamma \in \mathbb{R}^N} \left(-\sum_{j=1}^{J} \log(P_j(\gamma)) \right)$

$$P_j(\gamma) = \frac{e^{\gamma_j}}{\sum_{k=1}^{J} e^{\gamma_k}}$$

$$\frac{d(\log(p_j(\gamma)))}{d\gamma_i} = \frac{d(\gamma_j)}{d\gamma_i} - \frac{e^{\gamma_i}}{\sum_{k=1}^{N} e^{\gamma_k}} = \delta_{ij} - \text{softmax}(\gamma)_i$$

As a result,

$$\frac{dL(z_{(j)}, \gamma)}{d\gamma_i} = \text{softmax}(\gamma)_i - \delta_{ij}$$

In step 2, a loop is made where M decision trees are generated. Generic gradient boosting at the m-th step fits a decision tree $h_m(x)$ to pseudo-residuals. Let J_m be the number of its leaves. The tree partitions the input space into J_m disjoint regions $R_{1m}, \ldots, R_{J_m m}$ and predicts a constant value in each region. The output of $h_m(x)$ for input x can be written as:

$$h_m(x) = \sum_{j=1}^{J_m} b_{jm} \mathbf{1}_{R_{jm}}(x)$$

The value predicted in the region R_{jm} is denoted by b_{jm}.

This iterative scheme is then carried out over $m = 1, 2, 3, \ldots, M$ iterations. At the mth iteration, the algorithm computes, for each j, the residual $r_j^{(m)}$ as:

$$r_j^{(m)} := -\frac{\partial L(z^{(j)}, \gamma)}{\partial \gamma} = (\delta_{i,j} - \text{softmax}(\gamma)_i), \quad i \in 1, \ldots, N$$

and calculate with $\gamma = F^{(m-1)}(x_j) \in \mathbb{R}^N$.

The weak learner $h^{(m)}(x)$ is then fit to the training dataset $\{(x_j, r_j^{(m)})\}_{j=1}^{J}$. The algorithm then computes the multiplier $\alpha^{(m)}$ using the equation:

$$\alpha^{(m)} = \arg\min_{\alpha \in \mathbb{R}} \sum_{j=1}^{J} L\left(z^{(j)}, F^{(m-1)}(x_j) + \alpha h^{(m)}(x_j)\right).$$

This multiplier, $\alpha^{(m)}$, is then used to update the model/score as given by the scheme:

$$F^{(m)}(\cdot) = F^{(m-1)}(\cdot) + \alpha^{(m)} h^{(m)}(\cdot).$$

The XGBoost algorithm thus results in a strong learner by combining M weak learners which are in this case decision trees in order to obtain a close approximate to the true probability distribution.

2.2 Feed forward Neural Network

It's a special type of Aritficial Neural networks algorithms which try to harness the ability of biological networks to learn patterns within data.

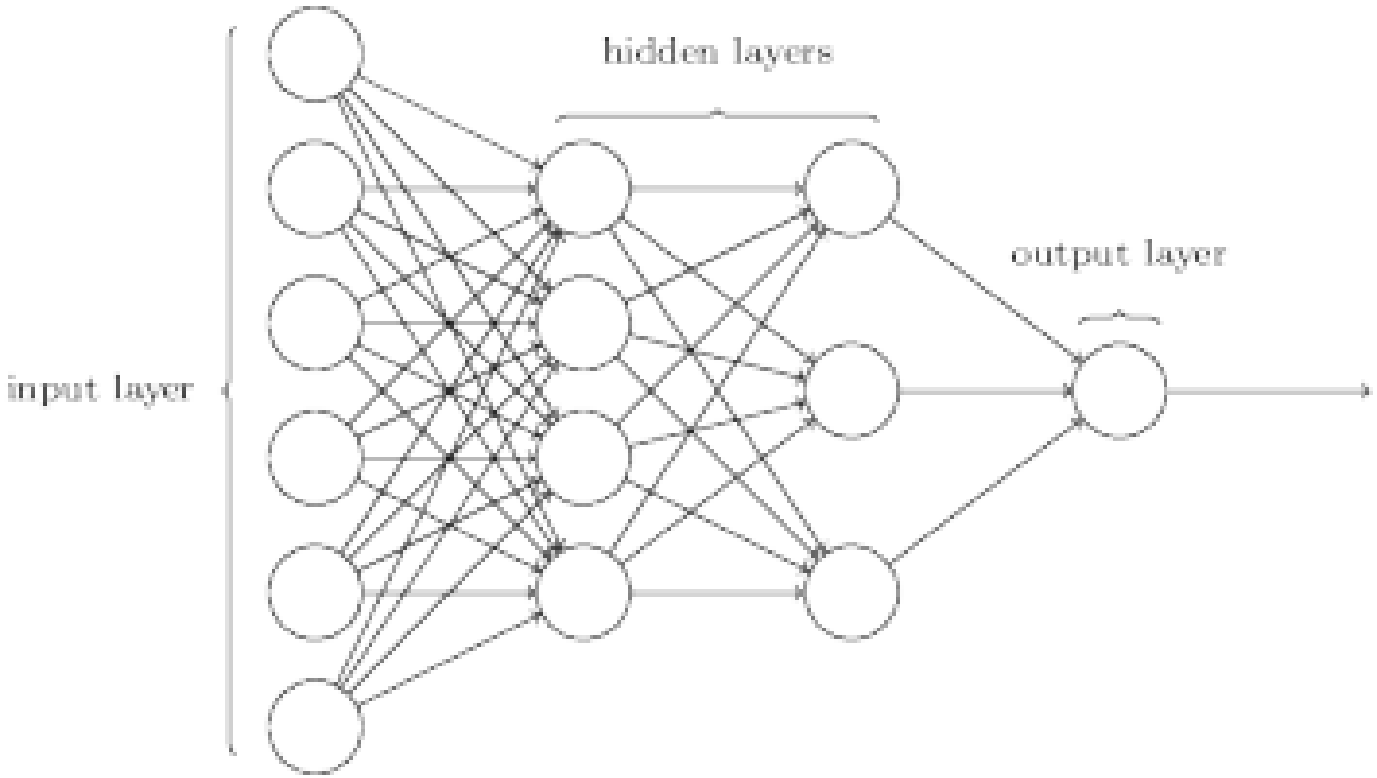

Figure: A representative feed forward Neural Net

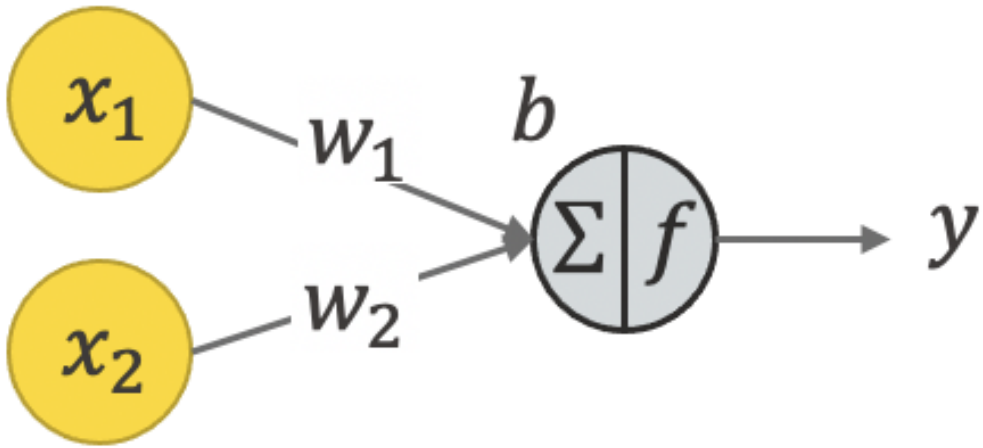

Figure: Components of a single neuron

The optimal number of layers in a neural network and the number of neurons in each layer is to be determined for a given problem. Along with this, it is also necessary to determine the appropriate activation functions for each of the neurons as well as the optimization scheme to be used.

Architecture used

Composition of the ANN used		
	Number of Neurons	*Activation Function*
Layer 1	128	ReLU
Layer 2	64	ReLU
Layer 3	50	softmax

TABLE 1. "Architecture" of the Neural Net used

The ReLU activation function is defined as:

$$\text{ReLU}(x) = \max(0, x)$$

The softmax activation function is defined as:

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}$$

Training

The parameters of the model(weights and biases) are trained using the input outputs pairs so a loss function is needed and in this case Cross entropy loss is used.
The parameters are optimized using Adam optimiser, an advancement of the stochastic gradient descent optimizer.

Adam optimizer

It's a method for efficient stochastic optimization that only requires first-order gradients with little memory requirement and combines advantages from AdaGrad and RMSProp methods. It computes the individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients.
Given a stochastic scalar function that is differentiable with respect to parameters θ, we want to minimize the expected value of $f(\theta)$. The gradient at time t is denoted as $g_t = \nabla f$.
The algorithm updates exponential moving averages of the gradient m_t and the squared gradient v_t, where β_1 and β_2 control the exponential decay rates of these moving averages.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector

$m_0 \leftarrow 0$ (Initialize 1$^{\text{st}}$ moment vector)
$v_0 \leftarrow 0$ (Initialize 2$^{\text{nd}}$ moment vector)
$t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**
  $t \leftarrow t + 1$
  $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
  $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
  $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
  $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
  $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
  $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
end while
return θ_t (Resulting parameters)

1. Definitions

2. Machine Learning techniques
   - Extreme gradient boosting
   - 2.2 Feed forward Neural Network

3. Data

4. 4 Model I/O

3 Data

The daily contract price data of European call options for NIFTY50 and BANKNIFTY indices are considered because they have a high trading volumes for these contract which makes us consider the derivative price as 'fair' price. Data is obtained for the years 2014/2018, from the NSE website's contract wise archive section and then filtered.

Symbol	Date	Expiry	Option Type	Strike Price	Open	High	Low	Close	LTP	Settle Price	No. of contracts	Turnover in Lacs	Premium Turnover in Lacs	Open Int	Change in OI	Underlying Value
NIFTY	01-Jan-2014	30-Jan-2014	CE	7000	0.95	1.10	0.85	1.00	1.10	1.00	2444	8555.16	-	811950	10750	6301.65
NIFTY	01-Jan-2014	30-Jan-2014	CE	7050	0.00	0.00	0.00	0.25	0.25	1.40	0	0.00	-	350	0	6301.65
NIFTY	01-Jan-2014	30-Jan-2014	CE	6800	3.15	3.95	2.85	3.60	3.90	3.60	15039	51157.12	-	1403700	12400	6301.65
NIFTY	01-Jan-2014	30-Jan-2014	CE	6850	0.00	0.00	0.00	2.75	2.75	6.85	0	0.00	-	15350	0	6301.65
NIFTY	01-Jan-2014	30-Jan-2014	CE	6900	1.40	1.85	1.30	1.55	1.85	1.55	9156	31594.85	-	778250	39950	6301.65

Figure: Snapshot of the Unfiltered Option Dataset

The 33 months period, i.e., data from Jan 2015 to Sept 2017 forms the training dataset and the succeeding data i.e., from Oct 2017 to Dec 2018 forms the test dataset for evaluating the proposed models

1. Definitions

2. Machine Learning techniques
   - Extreme gradient boosting
   - 2.2 Feed forward Neural Network

3. Data

4. 4 Model I/O

4. Model I/O

Categorical Output Variable: Scale free output variable:

$$100 \times \left(\frac{C}{K} \right)$$

where K: Strike price and C: Close price
Since no real market is complete and no additional optimization on hedging strategies is performed, the option contracts may possess multiple fair prices which could be anything in a certain interval so the outputs are divided into non-overlapping "bins".

Approach 1

The future dynamics of the underlying security can be anticipated from its present price and the price dynamics followed by it over the past few days. Log returns are used to resolve the scale dependency. Stationarity of log returns is assumed and its order statistics are caluculated by computing the log returns for the past 20 trading days. This approach results in a row of 22 features as given below:

1. The 19 log return order statistics.
2. The time to maturity (τ) of the option contract.
3. The interest rate r: The 3-month sovereign bond yield rates is used as an approximation for the risk-free interest rates.
4. Moneyness: Computed as $\frac{S}{k}$ (The ratio of Spot to Strike prices).

Approach 2

In this approach, the central tendency and the dispertion are measured using the first raw moment aand the covariance matrix of the component-wise log returns of the vector valued series. For a window of the past 20 trading days, the arithmetic mean (of Open, High, Low, and Close) and the covariance matrix are constructed. This approach results in the following 17 features:

1. Means of the log return series.
2. Ten statistics from sigma, namely $\left\{ \frac{\Sigma_{ij}}{\sqrt{|\Sigma_{ij}|}} \mid 1 \leq j \leq i \leq 4 \right\}$ with the convention $\frac{x}{\sqrt{|x|}}$ if $x = 0$.
3. Features (2)-(4) from Approach I.