Volatility Forecasting with PINNs

Jialiang Chen
Stern School of Business
New York University
New York, NY 10012, USA

Advisor: Professor João Sedoc and Professor Raghu Sundaram

Abstract

We introduce a new architecture of physics-informed neural networks for the task of volatility forecasting. Physics informed neural networks are neural networks that are trained to solve supervised learning tasks while maintaining certain physical constraints given by a set of general partial differential equations. Broadly, we let the model predict the future realized volatility while satisfying the Black-Scholes PDE constraints. In this way, our model will carry information from the option market into the prediction of future realized volatility. The natural difficulty of volatility forecasting methods provide the model with difficulty. However, by including Black-Scholes physical constraints in the model, we could see the resulting accuracy substantially improved. This provides directions for future research.

1. Introduction

Volatility measures the magnitude of the movements of the stock returns in a given period, and it is widely used in the financial industry. In some works volatility is represented in the form of absolute value of the returns, but our work concerns the standard deviations of returns in a range of future period. Volatility forecasting models can be used in different sections. For example, volatility has been accepted as a risk measure of the assets in the financial market. In the financial world, any returns and gains must be measured and discounted with respect to the volatility it bears. Volatility on the stock indexes also measures the overall risk level in the economy. A portfolio manager must know the probability that his or her holdings will decline in the future. In addition, volatility shed some lights on equity returns of the market. Empirical findings shows that there’s a negative correlation between stock returns and people’s perceptions of its volatility [1]. Therefore, predicting volatility sheds light on the future returns of the assets, which is also an essential concern for both practitioners and academics in finance. The most common used index that captures investors’ sentiments in the market is VIX index, which was invented by Professor Brenner at Stern. The basic idea of the index it taking implied volatilities of 30-day
SP500 options with different strikes, and average them on a weighted basis to get the VIX index. There's always a gap between the realized volatility and people's forecast, which means there's substantial space of improvements for existing volatility forecasting models. Another area that is closely related to volatility is options. Option traders are sometimes called volatility traders because the essential variable for option contracts is the volatility of the underlying asset. Most of the time, people take the prices of options as given by the market and work backwardly with Black-Scholes model to get the implied volatility on the asset. However, there are a few drawbacks on this method. First is the famous volatility smile. Empirically, options with strikes that are further away from the spot price have higher implied volatility, while options with strikes closer to the spot price have lower implied volatility. This doesn't align with the Black-Scholes model since the same spot should have the same volatility. The main explanation is that the assumptions of the Black-Scholes model are flawed, such as lognormal distribution and continuous price change. Given the above implications of volatility, it is critical for financial institutions to have a good volatility model that captures future realized volatilities. There exist a large amount of work in this area, and they can be roughly divided into two groups of method. The first is historical information based. The most famous one being GARCH, invented by Professor Robert Engle at Stern. The second group is option based, and the representative is Black-Scholes model. Nowadays, as the power of data science penetrates nearly all the fields in academia, there's no doubt that some tried to apply machine learning models on volatility forecasting. As shown by Kim Christensen(2021), ML models performed better in out of sample loss comparing to GARCH and HAR mode. He concluded that there are two reasons. First, the ML models benefit from the data-rich environment in the financial market as the feature place has been extended and the dataset enlarged. Second, ML has the ability to capture the highly non-linear structure in the financial market, especially in volatility forecasting. They tested different mainstream ML models and found that Neural Network and Random Forest performed the best over all the ML models. Our work will be based on their findings to start with Neural Networks and using its variant—physics informed neural networks(PINNs). It is used to solve certain physical problems with ML that involves some theoretical boundaries from PDEs as well as data points from experiments. It is a natural analogy that the future realized volatility should satisfy boundaries given by the BS equation and fit the past data points(option prices) we observe. Therefore, we try to apply this PINNs in our project.

2. Methodology

2.1 Overview of volatility models

The tradition volatility model that is widely used formulate the conditional variance directly as a function of observables [1]. Two of the most famous and fundamental of this kind are the ARCH and GARCH model. For example, a GARCH\((p, q)\) model
can be:

\[ h_t = \omega + \sum_{i=1}^{p} \alpha_i (R_{t-i} - \mu)^2 + \sum_{j=1}^{q} \beta_j h_{t-j} \]

where \( R_t \) is the continuously compounded return and \( h_t \) is the volatility. The idea we can get from the model is that, the volatility at time step \( t \) can be modeled as a function of the two sets of inputs. The first set of inputs is past returns, and the second set of inputs is past volatility. This formulation provides insights for us to define our own volatility model. The GARCH and ARCH framework has been proved to fit the empirical results in some sense since they could generate return series that fulfill some stylized facts of the market, including volatility persistence, mean reverting, clustering, etc [1]. Therefore we continue this formulation in our experiment.

2.2 Overview of Deep Neural Networks

The use of neural networks as function approximators and numerical solvers provide a new frontier for machine learning and numerical analysis [3]. For our paper, we need to have a brief understanding of neural networks to proceed.

![A typical Neural Network](image)

Each node in Figure 1 is called a neuron. Each neural is a function of the all the neurons from the previous layers. If we denote the outputs from the previous layers as \( x \), and we denote the function of this neuron as \( f(x) \), we could see that

\[ f(x) = g(w^T x + b) \]

where \( g \) is the non-linear activation function and \( w^T x + b \) is a linear transformation plus a shift. Each neuron in the neural network has the same formulation, but each
has its own weight parameter \( w \) and \( b \). The set of all the parameters is the parameters of the neural network, and our goal is to make the model work by optimizing those parameters. The neurons of the network jointly implement a complex non-linear mapping from the input to the output [4]. The mapping is "learned" by adapting the weights of each neuron through a technique called backpropagation [4]. It goes as follows: We start at the output of the network and move in the graph in reverse direction [4]. The procedure will stop once we hit the input layer[4]. To train the network, we need to have a loss function that takes in the output vector of the neural networks and some labels or correct values and compares the difference between them. In case of the continuous predictions, \( L^2 \) loss is usually used as a loss function. To update each layer, we compute the gradients of the loss function with respect to the weights. This computation requires chain rule:

\[
\frac{\partial f}{\partial x} = \frac{\partial f}{\partial y} \frac{\partial y}{\partial x}
\]

In computer implementation, it is divided into a forward pass and a backward pass. The forward pass goes through the layers and computes the output of each layer, and the backward pass computes the gradient at each layer. The gradient depicts the direction where the loss function decreases at the fastest rate. We control the step size by adjusting a learning rate constant. As the forward-backward loop repeats, we hope that at some point the weights are adjusted to minimize the loss function. Having all these, we can proceed to our discussion of physics-informed neural networks.

2.3 Physics-Informed Neural Networks

Physics-Informed Neural Networks (we refer to as PINNs later in the paper) were initially introduced by Raissi, M., et al. in 2018 [5]. They were used as an alternative to solve PDEs in a numerical way. The resulting neural networks form a new class of data-efficient universal function approximators that naturally encode any underlying physical laws as prior information [5]. It arose due to the fact that in analyzing complex physical, biological or engineering systems, the cost of data acquisition is prohibitive and researchers in these fields have to make decisions under partial information [5]. A special feature about those fields is that there exist a vast amount of prior knowledge that is not being utilized [5]. This construction allows us to tackle a wide range of problems in computation science with a data-efficient solution [5]. We use the example in Raissi’s work [5] to explain. Let’s consider a parametrized and non linear PDE in its general form:

\[
\frac{\partial u}{\partial t} + \mathcal{N}[u; \lambda] = 0
\]

where \( u(t, x) \) is the latent solution and \( \mathcal{N}[u; \lambda] \) is the nonlinear operator parametrized by \( \lambda \) [5]. Using this equation, we can define

\[
f := \frac{\partial u}{\partial t} + \mathcal{N}[u]
\]
The calculation of $\mathcal{N}[u]$ involves derivatives, which can be reached by using autodifferentiation. $f(t, x)$ and $u(t, x)$ share the same parameters. The PINN can be trained by minimizing the mean squared error loss [5]:

$$MSE = MSE_u + MSE_f$$

where

$$MSE_u = \frac{1}{N_u} \sum_{i=1}^{N_u} |u(t^i_{u}, x^i_{u}) - u^i|^2$$

and

$$MSE_f = \frac{1}{N_f} \sum_{i=1}^{N_f} |f(t^i_{f}, x^i_{f})|^2$$

The term $\{t^i_{u}, x^i_{u}, u^i\}_{i=1}^{N_u}$ denotes the initial and boundary training data on $u(t, x)$ and $\{t^i_{f}, x^i_{f}\}_{i=1}^{N_f}$ and $\{t^i_{f}, x^i_{f}\}_{i=1}^{N_f}$ specify the collocation points for $f(t, x)$ [5]. Therefore, we are learning some internal structure of the equation by enforcing the model to satisfy the PDE constraints.

3. Modeling Methodology

3.1 Black-Scholes-Merton Equation

Initially proposed by Black, Scholes, and Merton, the Black-Scholes-Merton equation provides a framework for option pricing [7]. We denote the stock price as $S$, $E$ be the strike price, $t$ be the time $c$ be the option price value, $r$ be the risk-free rate, $\sigma$ be the volatility, and $D_0$ be the continuous dividend yield. The BS equation gives us [7]:

$$\frac{\partial C}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2} + (r - D_0) S \frac{\partial C}{\partial S} - rC = 0, \quad 0 < S < S_f(t), \quad 0 \leq t < T$$

The boundary condition is:

$$C(T, S) = \max(S - E, 0)$$

and

$$C(0, t) = 0 \quad \forall t$$

The formula includes a volatility as the input, which is usually not directly observable from the market. However, the options are publicly traded, so people’s general view is that the price of options reflect people’s perception of its future volatility, and such volatility can be computed from Black Scholes pricing formula and numerical methods. This is called implied volatility. However, it is generally true that options on the same asset with the same time to maturity but different strikes have different implied volatility [8]. There are multiple explanations involving market dynamics or behavioral finance [8].
### 3.2 Constructing the PINNs

![Figure 2: Our PINN for volatility forecasting](image)

We have two sets of inputs, the first set is the option specs, including stock price, strike price, time to maturity, risk free rate, and dividend yield. The other set of inputs is the past information we used to predict future volatility. For example, in the graph we denote the last input to be \( h_9 \), which means there are 10 past information input, the returns of lag 1 day to 10 days. Each layer in the middle is transformed by a ReLU function. After going through multiple layers, our model will output a 2-dimensional vector. The first component is simply the predicted option price, and the second component is the predicted future output. Starting from the predicted option price, we can enforce the two constraints from the Black Scholes PDE. The first one is:

\[
F_t(t, s) + rsF_s(t, s) + \frac{1}{2} s^2 \sigma^2(t, s) F_{ss}(t, s) - rF(t, s) = 0
\]

which is denoted as \( MSE_F \), it can be computed by using the auto-differentiation feature of deep learning packages. The second constraint is

\[
F(T, s) = \Phi(s)
\]

which is denoted as \( MSE_B \). The third term is a loss term measuring the prediction loss for the second component of the output vector and realized future volatility of the corresponding period. We finally add them up to get the total loss of this sample. The intuition behind is that we want the parameters of the model to fit both the market expectation of the future volatility and the realized volatility. The option features can provide information about the implied volatility, while the past information features can provide estimate of the volatility structure, as the GARCH and ARCH model suggest.
4. Performance and Analysis

4.1 Dataset

In the implementation, we focused on predicting single stock volatility. The stock we choose the use is stock of Apple Inc, with ticker AAPL. The reason to choose it is that it has a long history and is a very liquid option market. The data source is Bloomberg. Due to the limitation of the database, we could only access daily level options data. I took the AAPL options data from Jan 1, 2004 to Apr 1, 2022. We chose the strike price to be between 90% to 110%. We would like to forecast the volatility in the next 10 days, but we don’t always have options that mature in 10 days. Therefore, we restrict on options that have time to maturity from 5 to 10 days. For the risk free rate, we used the U.S. 10yr treasury bond yield rate. For the dividend yield, we used the market estimate. Only call options dataset is included. Each layer has 400 neurons. The option data comes from the daily option chain at the end of the day. We have 24080 pieces of data.

4.2 Model Specs

Our model has 7 fully connected layers, each uses ReLU as the activation function. For the input, we used the option specs plus the 10 days returns. We also tested the performance when added the past 10 days volatility.

4.3 Results of Different Models

The two measures that we would use to evaluate our models are Mean Square Percent Error and Mean Absolute Percent Error of the realized future volatility. Separately,

\[
MSPE = \frac{\sum_{i=1}^{N} \left(\frac{\hat{\sigma}_i - \sigma_i}{\sigma_i}\right)^2}{N}
\]

and

\[
MAPE = \frac{\sum_{i=1}^{N} \left|\frac{\hat{\sigma}_i - \sigma_i}{\sigma_i}\right|}{N}
\]

. The results are summarized in the following:

<table>
<thead>
<tr>
<th></th>
<th>M1_{50}</th>
<th>M1_{150}</th>
<th>M2_{50}</th>
<th>M2_{150}</th>
<th>benchmark_{50}</th>
<th>benchmark_{150}</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSPE_train</td>
<td>0.3047</td>
<td>0.2926</td>
<td>0.6182</td>
<td>0.5199</td>
<td>0.4411</td>
<td>0.4409</td>
</tr>
<tr>
<td>MSPE_validation</td>
<td>0.2999</td>
<td>0.287</td>
<td>0.6388</td>
<td>0.5233</td>
<td>0.4319</td>
<td>0.4288</td>
</tr>
<tr>
<td>MAPE_train</td>
<td>41.48</td>
<td>41.59</td>
<td>56.65</td>
<td>51.91</td>
<td>47.52</td>
<td>47.39</td>
</tr>
<tr>
<td>MAPE_validation</td>
<td>41.46</td>
<td>41.46</td>
<td>57.01</td>
<td>51.78</td>
<td>47.46</td>
<td>47.16</td>
</tr>
</tbody>
</table>

M1 and M2 are two models with the following descriptions:

M1: model with option features, 10 day past returns.
M2: model with options features, and 10 day past returns, and 10 day past volatilities.
benchmark: model without adding the PINNs constraints.
4.4 Implication and Further Study

The first thing to notice is that our model doesn’t provide a strong volatility forecasting model. During the raining process, the loss of the neural networks kept decreasing even after 200 epochs. However, we could see that that the MSPE or MAPE doesn’t change a lot between 50 epochs and 150 epochs. We should also notice that M2 does not perform better than M1, which means that the extra past volatility that is supposed to be informative actually only provided noise for the model. Therefore, adding features may not necessarily improve he performance. Moreover, the main conclusion is that the model M1 performs substantially better than the benchmark. Since the extra loss term does not contribute directly to the prediction of future volatility, this means that the extra constraints does help and provide some useful restrictions for the model to refine its prediction. Also, PINNs took far more training epochs to converge since the benchmark model converges after 3 epochs. There are multiple ways to improve the model. First, the formulation of prediction daily level volatility is not promising in itself. The time scale of 10 days is too long and contains too much unexplained information. We need to focus on shorter time scale, like predicting the volatility of the next 10 minutes. This requires far more data. Second, the model is not capturing time dependencies by directly feeding it with returns in the past 10 days. We should use deep learning models that captures sequential dependencies, like RNN or transformers. Finally, predicting an accurate volatility is very difficult. We can reformulate the problem to a classification problem, where we can predict whether there will be a large movement in volatility.
5. Reference


