Deep Learning Stock Volatilities with Google Domestic Trends

Ruoxuan Xiong¹, Eric P. Nicholas² and Yuan Shen*³

¹Department of Management Science and Engineering, Stanford University
²Google Inc.
³Department of Physics, Stanford University

Abstract

We have applied the Long Short-Term Memory neural network to model S&P 500 volatilities incorporating Google domestic trends as indicators of the public mood and macroeconomic factors. In the 30% testing data, our LSTM model gives a MAPE of 24.2%, outperforming linear Ridge/Lasso and autoregressive Garch benchmarks by at least 31%. This evaluation is done on the optimal observation and normalization scheme which maximizes the mutual information. Our preliminary investigation shows strong potential to better understand stock behaviors using deep learning neural networks structures.

Introduction

One most fundamental question of stock modeling is to determine the timescale, i.e., how frequently one wishes to observe the price. This question should be answered before the evaluation of any quantitative strategies. Jump and drift traders may tend to trade on a daily frequency in which case $\sigma dB$ corrupts price series as noises. Observation of the price at such a frequency can not distinguish the difference between $\mu dt$ and $dJ$. In this work, we denote $\mu (S_t, d_t) dt + dJ (n_t)$ as the stock return $r_t = r (S_t, d_t, n_t)$.

Artificial neural networks (ANNs) are good nonlinear function approximators [1], so they are a natural approach to consider with modeling $r (S_t, d_t, n_t)$ and $\sigma (S_t, d_t, n_t)$ which are suspected to have nonlinear dependence on inputs. It is indeed not new to forecast financial time series using machine learning methods and recurrent neural networks (RNNs) are well-suited for this task. For instance, here is a brief list of some high-impact works using ANNs. This early (1990) work [2] is the first one using RNN to predict stock prices, ref. [3] reported a volatility forecasting model, ref. [4] incorporated public mood data in predicting the market fund DJIA, and ref. [5] modeled investors’ trading behaviors. However, no work has been reported on ANN modeling of either $r_t$ or $\sigma_t$ that takes all components of $S_t$, $d_t$ and $n_t$ into consideration.

Recent advances in ANNs leverage their predicting power by providing more insight into how they operate and systematically avoiding the spectre of overfitting which has plagued highly-parameterized supervised machine learning methods in past decades [6]. Specifically, there are new regularization methods (such as “dropout” [21]) and faster training techniques (such as using piecewise linear activation functions as opposed to transcendental functions [7]), which allow for multiple hidden layers to be trained easily — hence the new term “Deep Learning”. Additionally, new visualization techniques have been demonstrated [8] which give users more insight into how ANNs operate. These advances have also paved the way for more effective train-

---

¹sy0302@stanford.edu


\begin{equation}
\frac{dS_t}{S_t} = \mu (S_t, d_t) dt + \sigma (S_t, d_t, n_t) dB_t + dJ (n_t) + c (u_t - u_{t-1})
\end{equation}

Here $S_t$ is the stock price, $d_t$ is the domestic trend representing public interest and macroeconomics, $n_t$ is the news series of the underlining company, $u_t$ is a representation of the market friction and $c$ is the relative strength which can be measured by the bid-ask bounce.

- High frequency traders. As $dt$ approaches zero, all other terms in equ. die out except the correlation term $c (u_t - u_{t-1})$. A high frequency trader explores the correlation within the price series with strong focus on the market micro-structure and time series methods.

- Jump traders. Positive and negative company news $n_t$ drives the stock price up and down through the jump term $dJ (n_t)$. $dJ$ dissipates into the stock price at an intermediate daily timescale. In some circumstances, $dJ$ can significantly outweigh the other terms and thus provide opportunities for a jump trader.

- Drift traders. At an even longer timescale, $dJ$ and $\sigma dB$ both average out to be zero, leaving the stock price as an integral of the drift term $\mu dt$. A drift trader seeks trading strategies for $S_t$ under the macroeconomic trend $d_t$.

One most fundamental question of stock modeling is to determine the timescale, i.e., how frequently one wishes...
ing, leading to novel architectures such as the Long Short-Term Memory (LSTM), which has shown remarkable results in tasks such as artificial handwriting generation [9] and speech recognition [10].

In this work, we attempt to explore the possibility of learning the functional form of $\sigma$ using the LSTM which incorporates the Google domestic trends together with market data.

**Data Pre-Processing**

In this work, we study the market fund S&P 500 based on publicly available daily data of high, low, open, close and adjusted close prices. Daily returns $r_t$ are evaluated as the log difference of the adjusted close price while volatilities are rather estimated using the high, low, open and close prices in equ. 3 [11].

$$u = \log \left( \frac{H_t}{O_t} \right), \quad d = \log \left( \frac{L_t}{O_t} \right), \quad c = \log \left( \frac{C_t}{O_t} \right) \quad (2)$$

$$\sigma_t = 0.511 (u - d)^2 - 0.019 [c (u + d) - 2ud] - 0.383c^2 \quad (3)$$

![Financial Crisis](image)

**Figure 1:** Relative Google Search Volume on “Bankruptcy” as A Domestic Trend. The data is scaled by the value at the beginning (1-Jan-2004) of the time series. The maximum appears at the time of the great financial crisis which is highlighted by the cyan circle.

Starting from the year 2004, Google collects the daily volume of searches related to various aspects of macroeconomics and the database is open to public as the Google domestic trends $d_t$ [12]. Recent study has shown correlations between Google trends and the equity market [13] and we use them as a representation of the public interest and macroeconomics in this work. Fig. 1 shows an example of bankruptcy scaled by relative fraction in total Google searches on 1-Jan-2004. The marked maximum near the year 2008 corresponds to the financial crisis of 2007-08. For this study, domestic trends include advertising & marketing (advert), air travel (airtvl), auto buyers (auto), auto financing (autofi), business & industrial (bizind), bankruptcy (bnkpt), computers & electronics (comput), credit cards (ccard), durable goods (durable), education (educat), finance & investing (invest), financial planning (finpln), furniture (furtr), insurance (insur), jobs, luxury goods (luxury), mobile & wireless (mobile), mortgage (mortge), real estate (rest), rental, shopping (shop), small business (smallbiz), travel. Together with the return and volatility observed at the present time, they constitute an input $x_\tau, t$ of 25 dimension for the prediction of volatility at the next time stamp.

$$x_t = (r_t, \sigma_t, d_{advert}, \ldots, d_{travel}) \quad (4)$$

Volatilities are studied as the output in this work and the evaluation on returns could also be done.

$$y = \sigma_{t+1} \quad \text{or} \quad r_{t+1} \quad (5)$$

We split the whole data set into training (70%) and test set (30%). The training set starts from 19-Oct-2004 to 9-Apr-2012 while the test set starts from 12-Apr-2012 to 24-Jul-2015. Additionally, it is worth noting here that all these 25 time series are stationary in the sense that their unit-root null hypothesis have p-values less than 0.05 in the Augmented Dickey-Fuller test [14].

As stated in the introduction, preprocessing the time series with different observation and normalization schemes may result in different causality patterns between the input and output. Both the input and output time series should be transformed from the daily data within each scheme. Let $\Delta t$ be the observation interval,

$$r_t^{\Delta t} = \sum_{\tau = (t-1) \Delta t+1}^{t \Delta t} r_t \quad (6)$$

$$d_t^{\Delta t} = \frac{1}{\Delta t} \sum_{\tau = (t-1) \Delta t+1}^{t \Delta t} d_t \quad (7)$$

$$\sigma_t^{\Delta t} = \sqrt{\sum_{\tau = (t-1) \Delta t+1}^{t \Delta t} \sigma_t^2} \quad (8)$$

Normalization can be done by computing the forward z-scores with a sliding window of $k$ days for an time series $A$.

$$Z_{k,i}^A = \frac{A_i - \text{mean}(A_{i-k:i})}{\text{std}(A_{i-k:i})} \quad (9)$$

We would like to note here that $k = \infty$ corresponds to linear transformation of the time series $A$. 

---

2
In the study on the returns and the fact that returns are extremely noisy on the minute-to-daily timescale. Although the noise is unpredictable, how noisy the noise is may be tractable. Fig. 2 also shows that the mutual information finds its maximum value when \( \Delta t \) approaches \( \infty \). This is a timescale that all noises are essentially averaged out and one is left alone with a deterministic drift. Doing the normalization, on the other hand, can either reduce non-stationarity or increase the noise-to-signal ratio. The competition between these two factors results in a local maximum of the mutual information close to \( k = 30 \) in fig. 2b. Finally, we determine the optimal scheme through

\[
(\Delta t, k) = \arg\max_{\Delta t, k} \mathcal{MI}(Z_k^{\lambda t}, Z_k^{\sigma_k^t}).
\]  

(11)

To balance the predicting power and allowance of sufficient data samples, we choose

\[
\Delta t = 3 \text{ days}, \quad k = \infty.
\]  

(12)

It is worth noting that different metrics and scheme spaces can be used to replace equ. 11 for different specific problems. However, the methods of scheme selection used in this work should be widely applicable.

Each combination of \( \Delta t \) and \( k \) should determine an observation and normalization scheme which will all have different predicting power. We denote this scheme as \((\Delta t, k)\). In principle, one could apply learning models on each scheme and evaluate the accuracy of prediction on a validation set such that the optimal scheme can be chosen. Alternatively, an information metric can be set up to select the optimal scheme which maximizes this metric. In this work, we use the mutual information [15] defined as follows for each \((\Delta t, k)\). Assuming conditional independence between the input variables, the mutual information can be broken down into a sum of the individual components of \( x_i \).

\[
\mathcal{MI}(Z_k^{\lambda t}, Z_k^{\sigma_k^t}) = \sum_{\lambda} \mathcal{MI}(Z_k^{\lambda t}, Z_k^{\sigma_k^t}).
\]  

(10)

Fig. 2 shows, using the same color scheme, equ. 10 evaluated for both \( y = r_{t+1} \) (a) and \( y = \sigma_{t+1} \) (b) in the training set. The predicting power for the returns, in the sense of mutual information, is significantly smaller than that for the volatilities. This observation is consistent with our parallel study on the returns and the fact that returns are extremely noisy on the minute-to-daily timescale.
Methods

In our recurrent neural network modeling of the volatility, a single LSTM layer is employed without other hidden layers. The structure of this neural network is shown in fig. 4. It has a dynamic “gating” mechanism. Running through the center is the cell state $I_i$ which we interpret as the information flow of the market sensitivity. $I_i$ has a memory of past time information \[16\] and more importantly it learns to forget \[17\].

$$I_i = f_i \cdot I_{i-1} + c_i \cdot \tilde{I}_i$$ \hspace{1cm} (13)

Here $f_i$ is the fraction of past-time information passed over to the present, $I_i$ measures the information flowing in at the current time and $c_i$ is the weight of how important this current information is. All these three quantities are functions of the input $x_{i,i}$ and last-time’s estimation of the volatility $\hat{o}_i$.

$$f_i = \text{sigmoid} \left( (\hat{o}_i, x_i) \cdot W_f + b_f \right)$$ \hspace{1cm} (14)

$$c_i = \text{sigmoid} \left( (\hat{o}_i, x_i) \cdot W_c + b_c \right)$$ \hspace{1cm} (15)

$$\tilde{I}_i = \tanh \left( (\hat{o}_i, x_i) \cdot W_{\tilde{I}} + b_{\tilde{I}} \right)$$ \hspace{1cm} (16)

To make a prediction of the next volatility value $\hat{o}_{i+1}$, a linear activation function is used.

$$\hat{o}_{i+1} = \alpha + \beta \cdot o_i \cdot \tanh \left[ I_i \right]$$ \hspace{1cm} (17)

Here $o_i$, which is also a function of $x_{i,i}$ and $o_i$ tunes the output.

$$o_i = \text{sigmoid} \left( (\hat{o}_i, x_i) \cdot W_o + b_o \right)$$ \hspace{1cm} (18)

$I_i$ and $\hat{o}_{i+1}$ are passed down to the next time step for continuous predictions. Equ. 13 answers the fundamental question of memory in time series forecasting. It is an equivalent as evaluating autocorrelation and partial autocorrelation functions to determine the $p$ and $q$ maximum lags in the autoregressive moving average model (ARMA($p,q$)) \[18\].

All coefficients here are learned through training with the python deep learning library Keras \[19\]. Specifically, we set up the maximum lag to include 10 continual observations, consistent with the benchmark linear models which we will describe below. The model is trained by the “Adam” method \[20\] with 32 examples in a batch, with mean absolute percent error (MAPE) as the objective loss function and validation fraction as 20%. We have found that tuning the batch size and the validation fraction will change the MAPE in the test set by < 2% once training has reached 20% objective MAPE after roughly 600 epochs. Moreover, data points are shuffled during training, no dropout has been implemented in our work \[21\] and all initial weights are set to be small positive constant terms, similar to the normalized initialization given in \[22\].

To evaluate the performance of the LSTM model, 30% of the observed data is used as the test set. Additionally, we have developed two linear regression models (Ridge and Lasso) and one autoregressive model (Garch) \[18\] as benchmark models.

Garch: $\sigma^2_i = \omega + \alpha^2 \left[ \sigma_{i-1} + \beta \epsilon^2 \right] \sim \mathcal{N}(0,1)$ \hspace{1cm} (19)

Linear: $\sigma_i = \omega + \epsilon_i + \sum_{j=1}^{10} \alpha \cdot \epsilon_{i-j} \sim \mathcal{N}(0,1)$ \hspace{1cm} (20)

While the Garch model is easily trained by the maximum likelihood estimator, the linear model are regularized by $L_p$ norm of the coefficients $\alpha_{i,j}$ thus giving two linear regression benchmarks: Lasso ($p=1$) and Ridge ($p=2$). More specifically, we set up a grid of regularization parameter $C$ from $10^{-2}$ to $10^{-6}$ spaced equally in the log scale, train all of them on the first 80% of the training set by minimizing the following objective function.

$$O_p = C \cdot |\alpha_{i,j}|_p + \sum_i \epsilon_i^2$$ \hspace{1cm} (21)

The linear coefficients are nailed down on the later 20% validation part of the training set. We observe that the coefficients in volatility, return, bankrupt, invest, and jobs are significantly non-zero. This is consistent with the predicting power of each component evaluated by the mutual information.

Results

![Figure 5: Volatility Forecasting Made by the Long Short-Term Memory Model and Comparison with Benchmarks.](image-url)
In fig. 5, we plot the predicted volatility together with the observed values in the test set. The subplot shows two types of error metric for our LSTM model, compared with the benchmark models. The MAPE is used as the loss function in training the neural network. Therefore, the LSTM has significantly lower MAPE (> 31% relatively) than any other benchmark models. In terms of root mean square error (RMSE), the LSTM also outperforms other benchmark models. However, the improvement is not as significant as on the MAPE.

Our LSTM model is fair in the sense that the MAPE in the training set converges to roughly the same value (20%) as the MAPE evaluated in the test set (24.2%). We have further investigated overfitting by reducing the dimensionality of the input vector. Let us denote the LSTM with the full input $x_i$ as LSTM$_{0}$. Let LSTM$_{r}$ be the one which has only a subset of the input vector as listed in fig. 3 including volatility, return, comput, crcard, invest and bnkrpt.

<table>
<thead>
<tr>
<th>RMSE</th>
<th>MAPE (loss function)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM$_{0}$</td>
<td>$2.89 \times 10^{-3}$</td>
</tr>
<tr>
<td>LSTM$_{r}$</td>
<td>$2.88 \times 10^{-3}$</td>
</tr>
<tr>
<td>Garch</td>
<td>$3.13 \times 10^{-3}$</td>
</tr>
</tbody>
</table>

Table 1: Table one

Table. 1 shows that the test set MAPE in the reduced input model LSTM$_{r}$ increases from the original model LSTM$_{0}$. Thus, overfitting in the original model, if there is any, is of a minor issue.

Conclusion

In this work, we understand the Google domestic trends as environmental variables. Together with the market information, they constitute the driving force of daily S&P 500 volatility change. By evaluating the mutual information, we find the optimal observation and normalization scheme for volatility forecasting. Within this scheme, we develop a neural network model which consists of one single long short-term memory layer which is trained on 70% of the entire data set. This model gives a MAPE of 24.2% in the rest 30% of testing data, outperforming other linear and autoregressive benchmark models by at least 31%. This work shows the potential of deep learning of financial time series in the presence of string noises. The methods demonstrated in this work can be directly applicable for other financial quantities at completely different timescale where either correlation or deterministic drift outweigh noises.

References


