Stock Price Prediction via Discovering Multi-Frequency Trading Patterns

Liheng Zhang
University of Central Florida
4000 Central Florida Blvd.
Orlando, Florida 32816
lihengzhang1993@knights.ucf.edu

Charu Aggarwal
IBM T. J. Watson Research Center
New York 10598
charu@us.ibm.com

Guo-Jun Qi*
University of Central Florida
4000 Central Florida Blvd.
Orlando, Florida 32816
guojun.qi@ucf.edu

ABSTRACT

Stock prices are formed based on short and/or long-term commercial and trading activities that reflect different frequencies of trading patterns. However, these patterns are often elusive as they are affected by many uncertain political-economic factors in the real world, such as corporate performances, government policies, and even breaking news circulated across markets. Moreover, time series of stock prices are non-stationary and non-linear, making the prediction of future price trends much challenging. To address them, we propose a novel State Frequency Memory (SFM) recurrent network to capture the multi-frequency trading patterns from past market data to make long and short term predictions over time.

Inspired by Discrete Fourier Transform (DFT), the SFM decomposes the hidden states of memory cells into multiple frequency components, each of which models a particular frequency of latent trading pattern underlying the fluctuation of stock price. Then the future stock prices are predicted as a nonlinear mapping of the combination of these components in an Inverse Fourier Transform (IFT) fashion. Modeling multi-frequency trading patterns can enable more accurate predictions for various time ranges: while a short-term prediction usually depends on high frequency trading patterns, a long-term prediction should focus more on the low frequency trading patterns targeting at long-term return. Unfortunately, no existing model explicitly distinguishes between various frequencies of trading patterns to make dynamic predictions in literature. The experiments on the real market data also demonstrate more competitive performance by the SFM as compared with the state-of-the-art methods.

*G.-J. Qi is the corresponding author.

KDD '17, August 13-17, 2017, Halifax, NS, Canada

ACM ISBN 978-1-4503-4887-4/17/08...$15.00
https://doi.org/10.1145/3097983.3098117

KEYWORDS

Stock price prediction; Multi-frequency trading patterns; State Frequency Memory

ACM Reference format:
Liheng Zhang, Charu Aggarwal, and Guo-Jun Qi. 2017. Stock Price Prediction via Discovering Multi-Frequency Trading Patterns. In *Proceedings of KDD '17, Halifax, NS, Canada, August 13-17, 2017,* 9 pages.
https://doi.org/10.1145/3097983.3098117

1 INTRODUCTION

Stock investors attempt to discover latent trading patterns in stock market to forecast the future price trends for seeking profit-maximization strategies [13, 22]. The prediction of stock prices is a challenging task because of highly volatile and non-stationary nature of market [1]. Even more, predicting the stock prices in short or long-term time range relies on discovering different trading patterns in the security exchange market. For example, the investors like mutual funds and 401(k) managers tend to look for long-term returns and their trading frequencies are relatively low. On the contrary, in high-frequency trading, transactions are processed much more frequently within a short time period, resulting in the high volatility in stock prices.

Therefore, explicitly discovering and separating various frequencies of latent trading patterns should play a critical role in making price predictions for different ranges of time periods. While there are many approaches to stock price prediction [2, 18, 23, 30], none of existing models, to our best knowledge, explicitly decompose trading patterns into various frequency components, and seamlessly integrate the discovered multi-frequency patterns into price predictions. This inspires us to discover and leverage the trading patterns of multiple frequencies, resulting in a novel State Frequency Memory (SFM) recurrent network for stock price prediction.

In signal processing, the frequency domain of input signals is discovered by applying Discrete Fourier Transform (DFT). It decomposes input signals into multiple frequency components as to study the periodicity and volatility of the signals. We will adapt this idea to dynamically decompose the hidden states of trading patterns into multiple frequency components that will in turn drive the evolution of SFM over time. On the other hand, given the nonlinearity and non-stationarity of stock prices, Recurrent Neural

Network (RNN) has emerged as a candidate to learn temporal patterns such as price fluctuations. However, due to the vanishing gradients in the training process, the conventional RNN could fail to learn long-term dependencies in a time series [11]. On the contrary, Long Short Term Memory (LSTM), a variant of RNN, addresses this problem: its network structure is composed of several types of memory gates, which enables it to capture the long-term dependency of stock prices at different moments. Inspired by the gating architecture and multi-frequency decomposition, we present the SFM recurrent network that learns multiple frequency trading patterns as to make short and long-term price predictions.

1.1 A Glance at The Architecture

The State Frequency Memory (SFM) recurrent network attempts to extract and leverage non-stationary trading patterns of multiple frequencies. Compared with most of the existing methods, it is capable of characterizing trading patterns over different cycles, from seconds to minutes to days and even beyond, to infer the future trends of stock prices.

Fig. 1 compares the block diagram of the SFM with the RNN and the LSTM. Like the LSTM, the SFM models the hidden states underlying a time-series with a sequence of memory cells. However, the memory cells in the SFM comprise of state components for multiple frequencies, each of which we call a state-frequency component. Specifically, an input modulation is decomposed into various frequency bands and fed into the memory cell together with the decomposed state-frequencies of the last time. The SFM also uses the memory gates to model the long term dependency. The input and output gates regulate the amount of information flowing into and out of the memory cell across different frequency bands. A joint state-frequency forget gate controls the amount of information across different frequencies should be kept in memory cells. The hidden states of the output are formed from a combination of multiple state-frequency components, which are mapped to predict the future prices over various ranges through a nonlinear regression. In this fashion, trading patterns of various frequencies are learned through the state-frequency decomposition in memory cells, which provide clues on the future stock price trends. Experiments results on fifty stocks across ten sections demonstrate how the SFM can achieve competitive performances by modeling multi-frequency trading patterns as compared with the other state-of-the-art methods.

The remainder of this paper is organized as follows. We first review the related work in Section 2, followed by the presentation of the proposed SFM in Section 3. Experiment results on the stock price dataset are demonstrated in Section 4. The conclusions are made in Section 5.

2 RELATED WORK

One of the most widely used models for stock prediction is the Autoregressive (AR) model for linear and stationary time-series. For example, [17] applied the quantile AR model to analyze the dynamics of stock index returns in China; [31] decomposed the stock return into a couple of components and utilized the AR model to make prediction. However, stock prices are often highly nonlinear and non-stationary, which limits the real-world applicability of the AR model. Hence, a hybrid model combining K-Nearest Neighbor and support vector machine [21] was applied to learn the nonlinear pattern underlying stock price fluctuations. Alternatively, the Hidden Markov Model (HMM) [14] has also been applied to make nonlinear prediction of stock trend.

With the advance of deep learning, it has become promising to exploit and explore deep neural networks [15, 16, 20] for financial prediction. A Harmony Search based neural network [8] is proposed to forecast the stock market; [2] compared capability of neural networks with Autoregressive Integrated Moving Average (ARIMA) to predict stock trends; a Bayesian regularized Artificial Neural Networks [27] was presented to forecast stock indices; [23] presented a combined model desegregating Support Vector Machine, Artificial Neural Networks, and Random Forest for stock price forecasting. Although traditional neural networks have the ability to handle non-linear data, it is inadequate in modeling the long-term dependency in time series. This motivates the use of the gated memory cells to memorize the long-term context of time-series data, leading to the celebrated Long Short-Term Memory (LSTM) network.

The LSTM was first proposed in [12], which is a scalable dynamic model. Since then, many variants of the LSTMs [4, 6, 9] have been proposed. [10] surveyed the performance of various LSTM architectures. There are the existing works applying the LSTM and RNN to financial prediction. [19] comprehensively studied the impact of the LSTM's hyperparameters (e.g., the number of neurons, epochs, and data amount) on the prediction accuracy. A prediction system [18] has also been developed to fuse the Evolino LSTM with Delphi method to make trade decisions. [25] proposed a hybrid model combing RNN and AR to predict the stock returns. In [3], the LSTM is adopted to predict prices with historical numerical and textual data. [5] simulated a stock trading strategy with the forecast of the LSTM.

However, none of these works reveal the multi-frequency characteristics of the stock price time-series. Actually, a short-term prediction relies on the high frequency patterns to model the high volatility of the price time-series. On the other hand, a long-term prediction is more relevant to patterns with low frequencies. To discover the multi-frequency trading patterns, we develop a State Frequency Memory network to reveal dynamics of the stock price time-series. With the learnt frequency components, the proposed model can track the trend of periodic and generate a reliable prediction on stock prices.

3 FORMULATION AND MODEL STRUCTURE

In this section, we will first introduce the conventional LSTM architecture. Then we propose the architecture of the

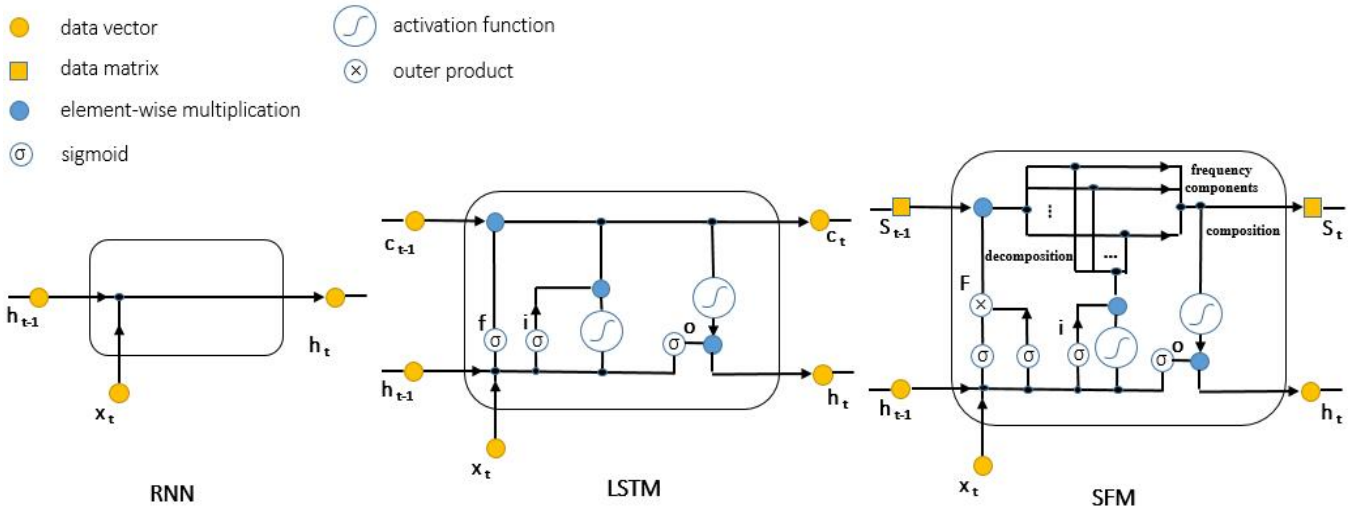


Figure 1: Comparison between the cell structures of the RNN (left), the LSTM (middle) and the SFM (right).

State Frequency Memory (SFM) recurrent neural networks. Finally we apply the recurrent neural networks (LSTM or SFM) to stock price prediction with historical prices.

3.1 Architecture of standard LSTM

Long Short Term Memory (LSTM) [12] network is a variant of Recurrent Neural Network (RNN). Different from the feed-forward neural networks, the RNN contains hidden states which evolve themselves over time. When trained with Back-Propagation Through Time (BPTT) [29], however, the conventional RNN suffers from vanishing gradients, making it unable to handle long-term dependency in a time series. Consequently, the LSTM was proposed to address this problem. Additional gating units in the LSTM make it capable of maintaining the long-term memory of the trading patterns from the historical prices.

Formally, the LSTM can be formulated as follows. At each time t, $\boldsymbol{x_t}$ is an input vector (e.g., stock prices), $\boldsymbol{c_t}$ denotes the memory state vector, and $\boldsymbol{h_t}$ is the hidden state vector output from $\boldsymbol{c_t}$. Then we have:

$$\boldsymbol{i_t} = \text{sigmoid}(\boldsymbol{W_i x_t} + \boldsymbol{U_i h_{t-1}} + \boldsymbol{b_i}) \quad (1)$$

$$\boldsymbol{f_t} = \text{sigmoid}(\boldsymbol{W_f x_t} + \boldsymbol{U_f h_{t-1}} + \boldsymbol{b_f}) \quad (2)$$

$$\tilde{\boldsymbol{c_t}} = \tanh(\boldsymbol{W_c x_t} + \boldsymbol{U_c h_{t-1}} + \boldsymbol{b_c}) \quad (3)$$

$$\boldsymbol{c_t} = \boldsymbol{i_t} \circ \tilde{\boldsymbol{c_t}} + \boldsymbol{f_t} \circ \boldsymbol{c_{t-1}} \quad (4)$$

$$\boldsymbol{o_t} = \text{sigmoid}(\boldsymbol{W_o x_t} + \boldsymbol{U_o h_{t-1}} + \boldsymbol{V_o c_t} + \boldsymbol{b_o}) \quad (5)$$

$$\boldsymbol{h_t} = \boldsymbol{o_t} \circ \tanh(\boldsymbol{c_t}) \quad (6)$$

where $\boldsymbol{W_*}$ and $\boldsymbol{U_*}$ denote the weight matrices, and $\boldsymbol{b_*}$ are bias vectors. The sigmoid sigmoid$(\cdot)$ is adopted as the activation function for three types of gating units – the input gate $\boldsymbol{i_t}$, forget gate $\boldsymbol{f_t}$ and output gate $\boldsymbol{o_t}$. The input modulation $\tilde{\boldsymbol{c_t}}$ and output $\boldsymbol{h_t}$ usually use the hyperbolic tangent $\tanh(\cdot)$ as the activation functions, and "$\circ$" denotes point-wise multiplication.

Three types of gating units control the stock-trading information entering and leaving a memory cell at each time. The input gate regulates the allowed amount of new information (e.g., new stock prices) flowing into the memory cell; the forget gate controls how much information should be kept in the cell; and the output gate defines the amount of information that can be output. The gating architecture of the LSTM enables it to balance between the short and long term dependency over the stock prices in a time series.

3.2 State Frequency Memory Recurrent Networks

In stock market, stock exchange and trading activities are performed at different paces and cycles, yielding multi-frequency trading patterns underlying stock prices. Price predictions over various ranges rely on different frequencies of trading patterns to provide useful clues on the future trends: short-term prediction depends more on high-frequency price data while the long-term prediction should focus more on low-frequency data. Thus, inspired by the Discrete Fourier Transform (DFT), we propose the State Frequency Memory (SFM) recurrent network to enable the discovery and modelling of the latent trading patterns across multiple frequency bands underlying the fluctuation of stock prices.

In particular, the SFM models the dynamics of an input time-series $\{\boldsymbol{x_t}|t = 1, 2, ..., T\}$ with a sequence of memory cells. We denote the output hidden state of SFM at time t as $\boldsymbol{h}_t \in \mathbb{R}^D$. However, to enable the capability of learning patterns across various frequencies, the memory states of SFMs are decomposed into a set of K discrete frequencies $\{w_k = \frac{2\pi k}{K} | k = 1, 2, ..., K\}$, which are evenly spaced on $[0, 2\pi]$.

This constructs a joint decomposition of states and frequencies to capture the temporal context of the input time-series. In this way, we represent the memory states of the SFM as a matrix $\boldsymbol{S_t} \in \mathbb{C}^{D\times K}$ at time t, with rows and columns corresponding to D states and K frequencies. Then, the state-frequency memory of the SFM evolves as a combination of the gated past memory and the current input like the LSTM. However, unlike the LSTM, the recurrent update of the SFM depends on different state-frequency components, reflecting the objectives to make a short or a long term prediction over stock prices.

In particular, the updating rule for the state-frequency matrix is formulated below.

$$\boldsymbol{S_t} = \boldsymbol{F_t} \circ \boldsymbol{S_{t-1}} + (\boldsymbol{i_t} \circ \tilde{\boldsymbol{c_t}}) \begin{bmatrix} e^{j\omega_1 t} \\ e^{j\omega_2 t} \\ ... \\ e^{j\omega_K t} \end{bmatrix}^T \in \mathbb{C}^{D\times K} \quad (7)$$

where $j = \sqrt{-1}$ and $\left[e^{j\omega_1 t}, e^{j\omega_2 t}, ..., e^{j\omega_K t}\right]$ are the Fourier basis of K frequency components of the state sequence.

Here, to balance the short and long-term dependency in a time-series, we adopt the memory gating architecture. The input gate $\boldsymbol{i_t} \in \mathbb{R}^D$ regulates the amount of new information flowing into the current memory cell. We define a joint state-frequency forget gate matrix $\boldsymbol{F_t} \in \mathbb{R}^{D\times K}$ to control how much information on various states and frequencies should be kept in the memory cell.

The input modulation $\tilde{\boldsymbol{c_t}}$ aggregates the current input information, which is then decomposed into a set of frequency bands. This multi-frequency decomposition of input information enables the SFM to discover the trading patterns across various frequencies.

The updating rule can be separated into the real and imaginary parts of the state-frequency matrix $\boldsymbol{S_t}$:

$$Re\boldsymbol{S_t} = \boldsymbol{F_t} \circ Re\boldsymbol{S_{t-1}} + (\boldsymbol{i_t} \circ \tilde{\boldsymbol{c_t}}) \left[cos\omega_1 t, ...cos\omega_K t\right] \quad (8)$$

$$Im\boldsymbol{S_t} = \boldsymbol{F_t} \circ Im\boldsymbol{S_{t-1}} + (\boldsymbol{i_t} \circ \tilde{\boldsymbol{c_t}}) \left[sin\omega_1 t, ...sin\omega_K t\right] \quad (9)$$

It is well known that complex numbers can be uniquely represented by its amplitude and phase. To encode the state-frequency matrix $\boldsymbol{S_t}$, we represent its amplitude $\boldsymbol{A_t}$ and the phase $\angle\boldsymbol{S_t}$ as:

$$\boldsymbol{A_t} = |\boldsymbol{S_t}| = \sqrt{(Re\boldsymbol{S_t})^2 + (Im\boldsymbol{S_t})^2} \in \mathbb{R}^{D\times K} \quad (10)$$

$$\angle\boldsymbol{S_t} = \arctan(\frac{Im\boldsymbol{S_t}}{Re\boldsymbol{S_t}}) \in [-\frac{\pi}{2}, \frac{\pi}{2}]^{D\times K} \quad (11)$$

where $\arctan(\cdot)$ is an element-wise inverse tangent function. Each entry $|\boldsymbol{S_t}|_{d,k}$ of $\boldsymbol{A_t}$ captures the amplitude of the dth state on the kth frequency. The state phase $\angle\boldsymbol{S_t}$ models the phase shift of each frequency component.

The amplitude will be fed into the memory cell gate and its frequency components will be composed to obtain the output hidden state $\boldsymbol{h_t} \in \mathbb{R}^D$. We ignore the phase $\angle\boldsymbol{S_t}$ as we found it has no significant impact on the results in our experiments but incurs extra computational and memory overheads.

To control how much of the past information should be kept in the memory cell, we define a state forget gate $\boldsymbol{f_t^{ste}}$ and a frequency forget gate $\boldsymbol{f_t^{fre}}$ to regulate the information on multi-states and multi-frequencies respectively. They are formulated as:

$$\boldsymbol{f_t^{ste}} = \text{sigmoid}(\boldsymbol{W_{ste}x_t} + \boldsymbol{U_{ste}h_{t-1}} + \boldsymbol{b_{ste}}) \in \mathbb{R}^D \quad (12)$$

$$\boldsymbol{f_t^{fre}} = \text{sigmoid}(\boldsymbol{W_{fre}x_t} + \boldsymbol{U_{fre}h_{t-1}} + \boldsymbol{b_{fre}}) \in \mathbb{R}^K \quad (13)$$

Then a state-frequency forget gate $\boldsymbol{F_t}$ is defined as the outer product $\otimes$ between $\boldsymbol{f_t^{ste}}$ and $\boldsymbol{f_t^{fre}}$ to jointly regulate the state and frequency information:

$$\boldsymbol{F_t} = \boldsymbol{f_t^{ste}} \otimes \boldsymbol{f_t^{fre}} \in \mathbb{R}^{D\times K} \quad (14)$$

It can be regarded as a composition gate over different states and frequencies to control the information flowing into the memory cell.

The formulations of the input gate $\boldsymbol{i_t}$ and the input modulation $\tilde{\boldsymbol{c_t}}$ are in the same fashion as the LSTM:

$$\boldsymbol{i_t} = \text{sigmoid}(\boldsymbol{W_i x_t} + \boldsymbol{U_i h_{t-1}} + \boldsymbol{b_i}) \quad (15)$$

$$\tilde{\boldsymbol{c_t}} = \tanh(\boldsymbol{W_c x_t} + \boldsymbol{U_c h_{t-1}} + \boldsymbol{b_c}) \quad (16)$$

The input modulation feeds the current observation $\boldsymbol{x_t}$ and the output hidden state $\boldsymbol{h_{t-1}}$ of the last step into the memory cell, forming the recurrent structure. As we discussed, it is decomposed into a set of frequency components with the Fourier basis to update the memory states. The input gate $\boldsymbol{i_t}$ controls how much of the new information are allowed to get into the memory cell.

To obtain the output hidden state $\boldsymbol{h_t}$, a state-only memory state $\boldsymbol{c_t}$ is reconstructed to aggregate the information over various frequencies on the state amplitude $\boldsymbol{A_t}$:

$$\boldsymbol{c_t} = \tanh(\boldsymbol{A_t u_a} + \boldsymbol{b_a}) \quad (17)$$

where $\boldsymbol{u_a} \in \mathbb{R}^K$ is an inverse transform vector. The vector composites the frequency components of the memory state. Then the state-only state $\boldsymbol{c_t}$ is obtained as a non-linear mapping of the composition. This process is like the Inverse Fourier Transformation (IFT) which recovers the original signal by combining the frequency components. Rather than adopting the standard Inverse Fourier transformation basis, the model learns the weights $\boldsymbol{u_a}$ for objective learning tasks.

With the output gate $\boldsymbol{o_t}$ regulating the information allowed to output from the memory cell, the output hidden state $\boldsymbol{h_t}$ is computed as:

$$\boldsymbol{o_t} = \text{sigmoid}(\boldsymbol{W_o x_t} + \boldsymbol{U_o h_{t-1}} + \boldsymbol{V_o c_t} + \boldsymbol{b_o}) \quad (18)$$

$$\boldsymbol{h_t} = \boldsymbol{o_t} \circ \boldsymbol{c_t} \quad (19)$$

As a variant of recurrent neural networks, the SFM can also be trained with the BPTT algorithm. The SFM recurrent network keeps the gating architecture of the LSTM, so as to remain the capability of capturing the long term dependency of time-series. Moreover, its unique complex-value memory states model multiple patterns with different frequencies. In the stock price prediction task, the SFM discovers clues from

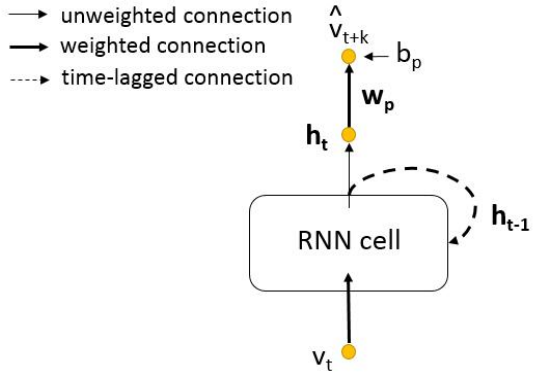


Figure 2: Price prediction with Recurrent Neural Network

the temporal context with a suitable length adjusted by the gate units. On the other hand, it models the trading patterns with various frequencies in stock market. For short term prediction, more high frequency components are needed to capture the high volatility of the time-series. While the long-term prediction is more relevant to the trading patterns with low frequencies. These patterns with various frequencies provide useful hints on the future trend, which is confirmed by the experiments we conducted.

3.3 Price Prediction

Both the SFM and the LSTM are variants of recurrent neural networks. They can be applied to predict the stock price as illustrated in Fig. 2, where a RNN cell can be a SFM cell and LSTM cell. The LSTM-based price prediction model is considered as the state-of-the-art baseline that is compared against the proposed SFM model.

Consider a time series of trading prices $\{p_t|t = 1, 2, ..., T\}$ of a stock. Our goal is to make a n-step prediction on p_{t+n} based on the prices up to p_t, where $n \geq 1$. Formally, the n-step prediction is defined below.

Definition 3.1. (n-step prediction)
Given prices $\{p_t|t = 1, 2, ..., T\}$, n-step prediction on the price p_{t+n} at time $t + n$ can be seen as a function:

$$\hat{p}_{t+n} = f(p_t, p_{t-1}, ..., p_1) \tag{20}$$

where f denotes the model mapping from the history prices to the price of n-step ahead.

As the scales of prices varies for different stocks, without loss of generality, we normalize the prices $\{p_t|t = 1, 2, ..., T\}$ of each stock to $\{v_t|t = 1, 2, ..., T\}$ on the range $[-1, 1]$.

Here we adapt a RNN variant, the SFM or the LSTM, as such a mapping f for n-step prediction. The initial hidden state $\boldsymbol{h_0}$ and memory state (if any) are set to zero, while the normalized price v_t is taken as an input at each step. The chosen RNN variant produces a hidden vector $\boldsymbol{h_t}$ that is used for price prediction. Specifically, we apply a matrix transformation on the hidden vector to make the n-step prediction:

$$\hat{v}_{t+n} = \boldsymbol{w_p}\boldsymbol{h_t} + b_p \tag{21}$$

where $\boldsymbol{w_p}$ is a weight vector, and b_p is the bias. Note that although this is a linear transformation, the nonlinearity of this price predictor arises from the nonlinear hidden vector $\boldsymbol{h_t}$.

This architecture differs from an AR model that uses immediately preceding data in a sliding window of a fixed size w for price prediction, i.e., using $\{p_{t-r}|0 \leq r < w\}$ to predict p_{t+n}. For an AR model, the data in this window can be treated as the context. Although increasing the size of the sliding window can incorporate the longer-term context of stock prices, it significantly increases the model complexity as the window size grows, often leading to an increased overfitting risk into history prices.

We do not encounter this problem in the LSTM or SFM. Because the memory state has the capacity of keeping the long-term dependency between the input prices. Therefore, it is unnecessary to involve the past prices in a time window as in an AR model. Indeed, feeding those past prices into memory cells could result in redundant memory of the past data, which could even degenerate its prediction accuracy due to an increased model complexity.

To learn general trading patterns in the stock market, prices of multiple stocks are used to train the regression network. Denote the prices as $\{p_t^m|t = 1, 2, ..., T; m = 1, 2, ..., M\}$ with M stocks, the model is trained by minimizing the sum of square errors between the predicted and true normalized prices in the training set:

$$\mathcal{L} = \sum_{m=1}^{M}\sum_{t=1}^{T}(v_{t+n}^m - \hat{v}_{t+n}^m)^2 \tag{22}$$

All the model parameters are updated through the BPTT algorithm. In this way, the weights of the model are shared among all the stocks.

The RNN layer in the regression network is flexible to be replaced by any RNN variant. We replace the RNN with the LSTM and the SFM for comparison. In both architectures, the internal memory states are maintained to summarize the internal factors of trading patterns up to the time t. These internal factors are supposed to capture the long-term dependency between stock prices with the help of the memory gates.

Moreover, the memory states in the SFM decompose the states into multiple frequency components. Multi-frequency patterns are summarized within the memory cell. Composing these frequency memory components, it outputs the hidden state $\boldsymbol{h_t}$ that can be used to predict the future prices. With the state-frequency memory states, it enables the SFM to capture the multi-frequency trading patterns as well as the internal dependency in the time-series of stock prices.

4 EXPERIMENTS

In this section, we conduct experiments to test the proposed SFM network. We first provide the details of the dataset

Table 1: Stock symbols of the selected corporations in the dataset. Five corporations with top market capitalization in each sector are selected.

basic materials	cyclicals [1]	energy	financials	healthcare	industrials	non-cyclicals [2]	technology	telecommunications [3]	utilities
BHP	AMZN	CVX	BAC	JNJ	BA	KO	AAPL	CHL	D
DOW	CMCSA	PTR	BRK-B	MRK	GE	MO	GOOGL	DCM	DUK
RIO	DIS	RDS-B	JPM	NVS	MA	PEP	INTC	NTT	EXC
SYT	HD	TOT	SPY	PFE	MMM	PG	MSFT	T	NGG
VALE	TM	XOM	WFC	UNH	UPS	WMT	ORCL	VZ	SO

we collected for experiments. Then, the comparison between the SFM and the other methods is presented. Finally we will discuss the parameter sensitivity of the model performance.

4.1 Dataset

We retrieve the stock prices from Yahoo! Finance. Specifically, we collect the daily opening prices of 50 stocks among 10 sectors from 2007 to 2016 for the experiments. For each sector, corporations with top 5 market capitalization are selected. Tab. 1 presents the stock symbols as well as the sectors of the selected corporations.

All the corporations are listed in the market before 2007. In history, there are several share splits for some stocks. In this situation, the prices are normalized according to the most recent split. The length of the historical prices is 2518 days. Models are trained with the daily prices with 2014 days from 2007 to 2014. Daily prices during 2015 and 2016 are used to validate and test respectively. The length of both validation and test time-series is 252. With prices of 50 corporations during the past ten years, we expect the models to learn general patterns in the market over time.

4.2 Comparison with other methods

To test our model, we conduct experiments to compare the proposed SFM network with the other state-of-the-art methods, including Autoregressive model and the conventional LSTM.

Autoregressive (AR) model is a classical method that is widely used for time-series forecasting. It predicts the future market price from the prices of several past steps. Specifically, its prediction is made as a linear combination of the past prices subject to a Gaussian noise term. The parameter w of an AR model specifies the number of past prices involved in the prediction, reflecting the degree of the dependence on the history data. The best w can be found by minimizing the Akaike Information Criterion (AIC) or Bayesian information criterion (BIC) [24]. Compared with the proposed model, an AR model is a linear regression model and can only model stationary time-series. Through the comparison with the AR model, we aim to test the capability of SFM to capture non-linear and non-stationary dependency, as well as the multi-frequency trading patterns in the market.

[1]Cyclical consumer goods & services.
[2]Non-cyclical consumer goods & services.
[3]Telecommunications services.

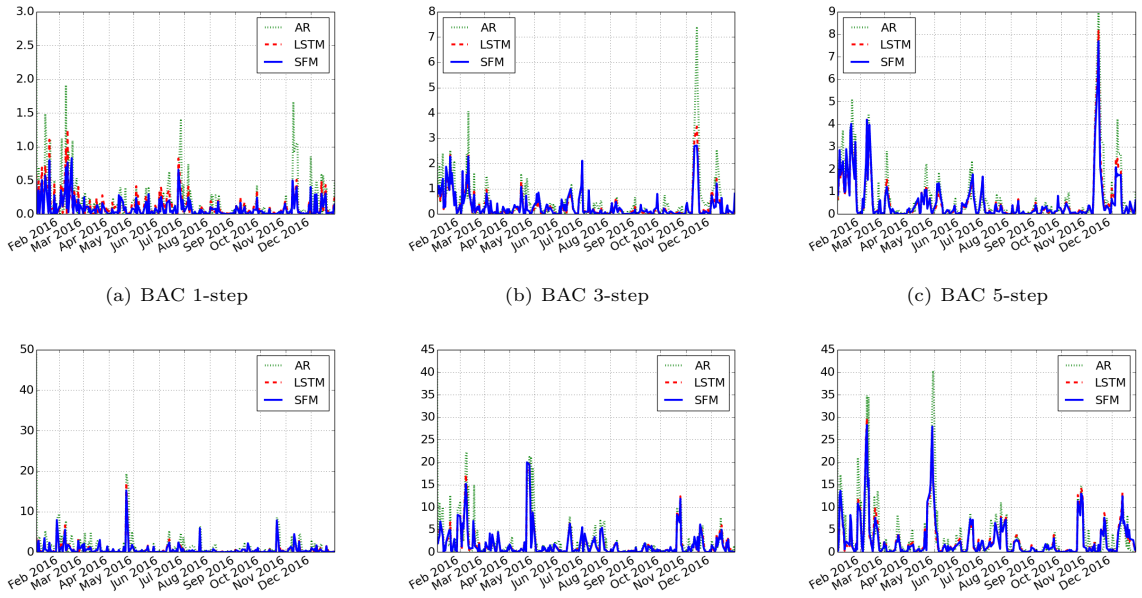
Table 2: The average square error of three models for 1-step, 3-step and 5-step prediction.

	1-step	3-step	5-step
AR	6.01	18.58	30.74
LSTM	5.93	18.38	30.02
SFM	**5.57**	**17.00**	**28.90**

We also compare with the conventional LSTM. It only captures the internal dependency of history prices. By the comparison with the conventional LSTM, we can evaluate the ability of the SFM in exploring trading patterns with various frequencies to predict the stock prices. Both LSTM and SFM are trained with a same procedure. The RMSprop optimizer [28] is used to train the models, with a fixed learning rate set as 0.01. The weights $\boldsymbol{W}_*$ and $\boldsymbol{U}_*$ are initialized by Xavier uniform initializer [7] and Orthogonal initializer [26] respectively. The bias vectors $\boldsymbol{b}_*$ are initialized as zeros. All the training sequences are fed to update the weights and bias per iteration. Both of the models converge after 4K iterations.

We choose the average square error per day per stock as our evaluation metric to compare the performances. We compare the 1-step, 3-step and 5-step prediction accuracy. Among them, the 1-step prediction implies the trend of the next day, which is a short-term prediction. The 3-step prediction indicates the half-week trend. Since the stock market opens only on the weekdays, 5-step prediction implies the trend in the next week. In other words, 5-step prediction typically covers one weekend, usually a much challenging task. The comparison of the test errors are reported in Tab. 2. To explicitly illustrate the performance, we select Bank of America (BAC) in the financials sector and Microsoft (MSFT) in the technology sector to plot the error curves of the test time series in Fig. 3.

From Tab. 2, we can see that the SFM outperforms both the AR and the LSTM models, whereas the LSTM works better than AR model. For each model, as we expected, the performance becomes worse as the prediction step increases. However, the SFM degenerates more slowly than the other two models. This is attributed to its ability of modeling multi-frequency trading patterns with adjusted context of the price time-series.



(a) BAC 1-step (b) BAC 3-step (c) BAC 5-step

(d) MSFT 1-step (e) MSFT 3-step (f) MSFT 5-step

Figure 3: Square error curves of AR, LSTM and SFM. The figure (a)(b)(c) show the results on BAC stock for different period of prediction. Figure (d)(e)(f) illustrate the error curves of models on MSFT stock.

On the other hand, the number w of past prices involved in the AR model can be regarded as the size of the memory of the price data. We also conduct experiments with the AR model of different w. The results are shown in Tab. 3. The test error does not show any significant improvements with an increasing number of w. In fact, the test error becomes even worse when w is larger. This could be caused by the increased model complexity with more parameters, leading to overfitting to the training sequence of price data. On the contrary, both the LSTM and the proposed SFM reach a better test error than the AR model, showing that they are more capable of capturing the long-term dependency without running an overfitting issue facing the AR model.

In addition, with the involvement of trading patterns with multiple frequencies, the SFM shows a more precise prediction than the LSTM that only models the internal dependencies. We note that the frequency components account for trading activities at different paces and cycles. Regulating information of different frequencies with the gate units, the SFM filters the irrelevant frequency components and retains the relevant components to provide a guidance for the prediction of future trend. In this way, it prevents the states from over-dominating the price prediction by trapping the model into the local pattern of price changes. The experiment results also confirm it with the competitive performance achieved by the SFM.

From Fig. 3, we can see that all the three models can capture the trend of stock prices. As expected, the error curves of the SFM are lower than those of the LSTM and the AR model. Moreover, the AR tends to simply repeat the trajectory of history prices with a serious delay. This causes a very poor prediction accuracy when the stock price changes abruptly. In contrast, the LSTM and the SFM perform much better than the AR model, since they are less likely to be affected by the local pattern of price changes in a short term.

Table 3: The test errors of the AR model versus different numbers w of involved past prices.

w	3	5	10	15	20
1-step	**6.01**	6.10	6.07	6.16	6.26
3-step	**18.58**	18.71	18.98	19.26	19.86
5-step	31.06	**30.74**	31.2	32.05	32.35

4.3 Impact of Model Complexity

The complexity of the proposed SFM network has a direct impact on the performance. In theory, the training error can be arbitrarily low with a sufficiently complex model. However, a low prediction error on the training set does not imply a low test error. A complex model can increase the overfitting risk. In particular, the complexity of the SFM can be measured in terms of two metrics – the number of states and the number of frequencies. In the following two parts, we study their impacts on the prediction performance, revealing some insight into how the model works with different configurations.

Table 4: The average square error versus the number of states in LSTM

# of states	10	20	40	50
1-step	**5.93**	6.60	6.91	6.89
3-step	**18.38**	19.05	18.60	18.68
5-step	30.30	30.58	**30.02**	31.07

It is worth noting that the proper hyper-parameters regarding the model complexity are chosen based on the performance on the validation set. In our experiments, we use the prices from 2007 to 2014 to learn the model, and the prices during 2015 as the validation set. Once the hyper-parameters are chosen, the trained model is evaluated on the test set.

4.3.1 Impact of the number of states. For the LSTM, the complexity is estimated by the number of states, i.e., the dimension of the memory states. However, for the SFM, the number of states indicates the number of rows of the memory state matrices, which implies the number of patterns expected to learn from the price time-series. The more expected patterns the network learns, the more complex the model is. We conduct experiments with varying number of states in both the LSTM and the SFM to study its impact on the performance.

Tab. 4 illustrates the results of the LSTM. For 1-step and 3-step prediction, the LSTM achieves the best performance with 10-dimensional memory states. We observe the overfitting with the increasing number of states. Simply increasing the number of states cannot decrease the test error anymore. Even worse, too large number of states (i.e., more than 50) in LSTM can even worsen the test error, causing an excessive fitting into the training prices. Without the decomposition and regulation of information over different frequencies, the LSTM blindly blends the information of all frequencies, including the irrelevant frequency components. Thus, it tends to over-dominate the price prediction by trapping the model into the local pattern. Note that since the 5-step prediction is more challenging, the LSTM needs more states (i.e., 40) to achieve the best performance.

Fixing the number of frequencies as 10, the results of the SFM is presented in Tab. 5. By introducing the decomposition and regulation of the states, the overfitting problem is not as serious as the LSTM. Especially, the SFM achieves its best performance with 50 states for the 3-step and 5-step. Although in 1-step prediction, the performance starts to get worse with increasing number of states, the SFM degenerates more slowly than the LSTM. We will illustrate that the number of frequencies instead of states is more critical for 1-step prediction, which is a short-term prediction. The results indicate that the overfitting problem is relieved with the regulation of multiple frequency components. Actually, the gate units automatically filter the irrelevant frequency components. Relevant frequency components are retained to provide useful hints on the future trend of the stock market.

Table 5: The average square error versus the number of states in SFM. The number of frequencies is fixed as 10.

# of states	10	20	40	50
1-step	**5.57**	5.79	6.15	5.91
3-step	18.48	19.20	17.25	**17.00**
5-step	29.48	29.84	31.30	**28.90**

Table 6: The average square error versus the number of frequencies in SFM. The number of states is fixed as 50.

# of frequencies	5	10	15	20
1-step	6.69	5.91	5.91	**5.88**
3-step	18.39	**17.00**	19.15	19.52
5-step	30.95	**28.9**	30.57	31.22

With the decomposition and regulation of multiple frequencies, involving more hidden states becomes meaningful and yields a more accurate prediction.

4.3.2 Impact of number of frequencies. Besides the states, the number of frequencies is also an important hyperparameter in the SFM. Fixing the number of states as 50, we explore the impacts of the number of frequencies on the performance. The results are shown in Tab. 6.

From Tab. 6, we can see that the SFM achieves the best performance for 1-step prediction with 20 frequency components. In this case, more frequency components are needed to model the high volatility in such a short term. As we introduced in Sec. 3.2, the frequencies are evenly distributed on $[0, 2\pi]$. It is clear that with more frequencies, the memory states involve more information on the high frequency in short-terms. And the components of high frequency account for the short-term trading activities. It indicates the potential of the SFM to model the high frequency trading in the stock market. For 3-step and 5-step prediction, the best performance is achieved with 10 frequencies. As their volatility is not as high as the 1-step prediction, components with low frequency become more relevant to the prediction. The results indicate that with varying number of frequencies, the SFM is capable of capturing patterns in trading activities at different paces.

5 CONCLUSION

In this paper, we present a State Frequency Memory regression network learning trading patterns with multiple frequencies to predict the trend of stock prices. Actually, the stock price reflects the multi-frequency patterns due to the trading activities performed at different paces. Discovering relevant frequency patterns provides useful clues on the future price trend. However, to the best of our knowledge, none of existing methods of price prediction model distinguish between

the multi-frequency patterns underlying the price time-series. With the joint state-frequency memory states, the SFM models the trading patterns as multiple frequency components. The long-term dependencies are as well revealed with the gating structure like the LSTM. Experiment results on the real price data demonstrate that the SFM can discover and regulate the multi-frequency patterns in the stock prices, outperforming both the AR and the conventional LSTM models.

REFERENCES

[1] Klaus Adam, Albert Marcet, and Juan Pablo Nicolini. 2016. Stock market volatility and learning. *The Journal of Finance* 71, 1 (2016), 33–82.

[2] Ayodele Ariyo Adebiyi, Aderemi Oluyinka Adewumi, and Charles Korede Ayo. 2014. Comparison of ARIMA and artificial neural networks models for stock price prediction. *Journal of Applied Mathematics* 2014 (2014).

[3] Ryo Akita, Akira Yoshihara, Takashi Matsubara, and Kuniaki Uehara. 2016. Deep learning for stock prediction using numerical and textual information. In *Computer and Information Science (ICIS), 2016 IEEE/ACIS 15th International Conference on*. IEEE, 1–6.

[4] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[5] Qiyuan Gao. 2016. *Stock market forecasting using recurrent neural network*. Ph.D. Dissertation. University of Missouri–Columbia.

[6] Felix A Gers and Jürgen Schmidhuber. 2000. Recurrent nets that time and count. In *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, Vol. 3. IEEE, 189–194.

[7] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks.. In *Aistats*, Vol. 9. 249–256.

[8] Mustafa Göçken, Mehmet Özçalıcı, Aslı Boru, and Ayşe Tuğba Dosdoğru. 2016. Integrating metaheuristics and Artificial Neural Networks for improved stock price prediction. *Expert Systems with Applications* 44 (2016), 320–331.

[9] Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks* 18, 5 (2005), 602–610.

[10] Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. 2015. LSTM: A Search Space Odyssey. *arXiv preprint arXiv:1503.04069* (2015).

[11] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. 2001. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. (2001).

[12] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[13] Yong Hu, Bin Feng, Xiangzhou Zhang, EWT Ngai, and Mei Liu. 2015. Stock trading rule discovery with an evolutionary trend following model. *Expert Systems with Applications* 42, 1 (2015), 212–222.

[14] G Kavitha, A Udhayakumar, and D Nagarajan. 2013. Stock Market Trend Analysis Using Hidden Markov Models. *arXiv preprint arXiv:1311.4771* (2013).

[15] Kyoung-Jae Kim and Hyunchul Ahn. 2012. Simultaneous optimization of artificial neural networks for financial forecasting. *Applied Intelligence* 36, 4 (2012), 887–898.

[16] Leonel A Laboissiere, Ricardo AS Fernandes, and Guilherme G Lage. 2015. Maximum and minimum stock price forecasting of Brazilian power distribution companies based on artificial neural networks. *Applied Soft Computing* 35 (2015), 66–74.

[17] Lili Li, Shan Leng, Jun Yang, and Mei Yu. 2016. Stock Market Autoregressive Dynamics: A Multinational Comparative Study with Quantile Regression. *Mathematical Problems in Engineering* 2016 (2016).

[18] Nijolė Maknickienė and Algirdas Maknickas. 2013. Financial market prediction system with Evolino neural network and Delphi method. *Journal of Business Economics and Management* 14, 2 (2013), 403–413.

[19] Nijolė Maknickienė, Aleksandras Vytautas Rutkauskas, and Algirdas Maknickas. 2011. Investigation of financial market prediction by recurrent neural network. *Innovative Technologies for Science, Business and Education* 2, 11 (2011), 3–8.

[20] Jibendu Kumar Mantri, P Gahan, and Braja B Nayak. 2014. Artificial neural networks–an application to stock market volatility. *Soft-Computing in Capital Market: Research and Methods of Computational Finance for Measuring Risk of Financial Instruments* (2014), 179.

[21] Rudra Kalyan Nayak, Debahuti Mishra, and Amiya Kumar Rath. 2015. A Naïve SVM-KNN based stock market trend reversal analysis for Indian benchmark indices. *Applied Soft Computing* 35 (2015), 670–680.

[22] Wijnand Nuij, Viorel Milea, Frederik Hogenboom, Flavius Frasincar, and Uzay Kaymak. 2014. An automated framework for incorporating news into stock trading strategies. *IEEE transactions on knowledge and data engineering* 26, 4 (2014), 823–835.

[23] Jigar Patel, Sahil Shah, Priyank Thakkar, and K Kotecha. 2015. Predicting stock market index using fusion of machine learning techniques. *Expert Systems with Applications* 42, 4 (2015), 2162–2172.

[24] David Posada and Thomas R Buckley. 2004. Model selection and model averaging in phylogenetics: advantages of Akaike information criterion and Bayesian approaches over likelihood ratio tests. *Systematic biology* 53, 5 (2004), 793–808.

[25] Akhter Mohiuddin Rather, Arun Agarwal, and VN Sastry. 2015. Recurrent neural network and a hybrid model for prediction of stock returns. *Expert Systems with Applications* 42, 6 (2015), 3234–3241.

[26] Andrew M Saxe, James L McClelland, and Surya Ganguli. 2013. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120* (2013).

[27] Jonathan L Ticknor. 2013. A Bayesian regularized artificial neural network for stock market forecasting. *Expert Systems with Applications* 40, 14 (2013), 5501–5506.

[28] Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* 4, 2 (2012).

[29] Paul J Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proc. IEEE* 78, 10 (1990), 1550–1560.

[30] Yi Xiao, Jin Xiao, John Liu, and Shouyang Wang. 2014. A multiscale modeling approach incorporating ARIMA and ANNs for financial market volatility forecasting. *Journal of Systems Science and Complexity* 27, 1 (2014), 225–236.

[31] Haibin Xie, Jiangze Bian, Mingxi Wang, and Han Qiao. 2014. Is technical analysis informative in UK stock market? Evidence from decomposition-based vector autoregressive (DVAR) model. *Journal of Systems Science and Complexity* 27, 1 (2014), 144–156.