Neural Comput & Applic
DOI 10.1007/s00521-014-1550-z

Empirical analysis: stock market prediction via extreme learning machine

Xiaodong Li · Haoran Xie · Ran Wang · Yi Cai · Jingjing Cao · Feng Wang · Huaqing Min · Xiaotie Deng

Received: 11 September 2013 / Accepted: 17 January 2014
© Springer-Verlag London 2014

Abstract How to predict stock price movements based on quantitative market data modeling is an attractive topic. In front of the market news and stock prices that are commonly believed as two important market data sources, how to extract and exploit the hidden information within the raw data and make both accurate and fast predictions simultaneously becomes a challenging problem. In this paper, we present the design and architecture of our trading signal mining platform that employs extreme learning machine (ELM) to make stock price prediction based on those two data sources concurrently. Comprehensive experimental comparisons between ELM and the state-of-the-art learning algorithms, including support vector machine (SVM) and back-propagation neural network (BP-NN), have been undertaken on the intra-day tick-by-tick data of the H-share market and contemporaneous news archives. The results have shown that (1) both RBF ELM and RBF SVM achieve higher prediction accuracy and faster prediction speed than BP-NN; (2) the RBF ELM achieves similar accuracy with the RBF SVM and (3) the RBF ELM has faster prediction speed than the RBF SVM. Simulations of a preliminary trading strategy with the signals are conducted. Results show that strategy with more accurate signals will make more profits with less risk.

Keywords Stock market prediction · Trading signal mining platform · Extreme learning machine

X. Li · H. Xie · R. Wang
Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong

R. Wang
Institute of Biomedical and Health Engineering, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518067, China

Y. Cai (✉) · H. Min
School of Software Engineering, South China University of Technology, Guangzhou 510006, China
e-mail: scut.yi.cai@gmail.com; ycai@scut.edu.cn

J. Cao
School of Logistics Engineering, Wuhan University of Technology, Wuhan 430070, China

F. Wang
State Key Lab of Software Engineering, School of Computer Science, Wuhan University, Wuhan 430072, China

X. Deng
AIMS Lab, Department of Computer Science, Shanghai Jiaotong University, Shanghai 200240, China

X. Deng
Shanghai Key Lab of Intelligent Information Processing and School of Computer Science, Fudan University, Shanghai 200433, China

1 Introduction

Stock market is one of the most important and active financial markets. Market players use different models to analyze the huge market data to make predictions. Many previous works in computer science put the problem into a machine learning framework. Schumaker and Chen [1] make forecasting based on financial news articles by a text classification approach. Yeh et al. [2] use multi-kernel learning model to regress on the stock prices. However, most previous approaches only focus on the prediction accuracy, and rarely analyze the speed that is an important aspect of the online forecasting. Since both the prediction accuracy and speed are highly demanding in stock price predictions, we build up a trading signal mining platform

Published online: 02 February 2014

that could take news and market prices as inputs and predict stock short-term price movements with high accuracy and fast speed.

Prediction accuracy, which is also known as the accuracy of trading signals, is the first requirement of the model. Generally speaking, trading signals that consistently have the accuracy statistically different from random predictions are useful. Execution strategies, such as volume-weighted average price (VWAP) and market on close (MOC) etc., can benefit from accurate predictions by adjusting their trading schedules or tilting their trading volumes according to the directions of the trading signals. Proprietary strategies, such as market making (MM), trend following (TF) etc., could use the trading signals to avoid adverse selection cost, or they could even directly bet on the signals. Therefore, higher accuracy of trading signals indicates greater chance of profitability.

The other requirement is the prediction speed of the system. For intra-day high frequency trading (HFT), the speed of incoming real-time market data is at millisecond level for most stock markets. Thus, in order to adapt to the dynamics of the intra-day market conditions and give enough time for the trading strategies to respond to the trading signals, the prediction speed of the system should be faster than market data speed. The model's training or tuning time, on the other hand, is not critical, since those phases can be arranged as an overnight job and finish before the market opens.

Extreme learning machine (ELM), as an emergent supervised technique [3, 4], is reported to have high accuracy and fast prediction speed while solving various real-life problems. As shown by Huang et al. [3], ELM randomly assigns the input weights and hidden layer biases instead of fully tuning all the internal parameters such as BP-NN. Specific to Single-hidden Layer Feedforward Networks (SLFNs), ELM could analytically determine the output weights. With the proof of theorems (Theorem 2.1 and 2.2 in [3]), it gives the smallest norm of weights, easy implementation and fast prediction speed. Huang and Siew [4] kernelize the basic version of ELM, which further improves the ELM's prediction accuracy. Solid experimental results on UCI Machine Learning Repository data sets (http://www.ics.uci.edu) could be found in [5, 6].

Based on aforementioned good properties and performance, we consider ELM as a candidate with a high priority to match the two key requirements of the stock market prediction problem. In this paper, we build up a trading signal mining platform that uses ELM to integrate two kinds of market information sources, i.e., market news and prices. We compare the performance of ELM with SVM and BP-NN on one year H-share news articles and intra-day tick prices. Experimental comparisons demonstrate the effectiveness of the proposed system.

The rest of this paper is organized as follows. Section 2 reviews the approaches to stock market predictions, the formulation of ELM and its applications. Section 3 presents our proposed system. Section 4 reports the experimental comparisons and discussions. Section 5 gives the conclusion and future work directions.

2 Related work and background

2.1 Information sources in stock market

Market news and stock prices, known as two of the most important sources of market information, are widely adopted by both academic researchers and market practitioners. With the advancement of the HFT, the reporting speed and the volume of market data have been increasingly significant. In particular, useful trading signals are mined by high performance computers, together with algorithms of financial engineering and machine learning. In order to better utilize these signals, how to model and make accurate and fast predictions has become a challenging problem.

In our previous work [7], it was discovered that combining both market news and prices achieves better prediction accuracy. Therefore, in this paper, instead of modeling the mapping from market information to prediction signal separately as

$$\pi : \mathbf{N} \mapsto \mathbb{L} \text{ or } \pi : \mathbf{P} \mapsto \mathbb{L}, \tag{1}$$

where $\mathbf{N}$ denotes the information source of news, $\mathbf{P}$ denotes the information source of prices and $\mathbb{L}$ is the set of prediction labels, we combine $\mathbf{N}$ and $\mathbf{P}$ together and make the model as

$$\pi : \{\mathbf{N}, \mathbf{P}\} \mapsto \mathbb{L}. \tag{2}$$

2.2 Traditional approaches

Analysis of news articles has been reported in the literature of computer sciences. Following the approaches of text mining on news articles, Seo et al. [8] build a multi-agent system for intelligent portfolio management, which can assess the risk associated with companies by analyzing news articles. Yu et al. [9] propose a four-stage SVM based on the multi-agent ensemble learning approach to credit risk evaluation. The AZFinText system, built by Schumaker and Chen [10], makes not only directional predictions but also quantified estimations of prices.

As illustrated in Fig. 1, the processing pipeline of the approaches that use news could be summarized below:

1. *Representation of news articles* A piece of news is usually represented as a term vector by using the

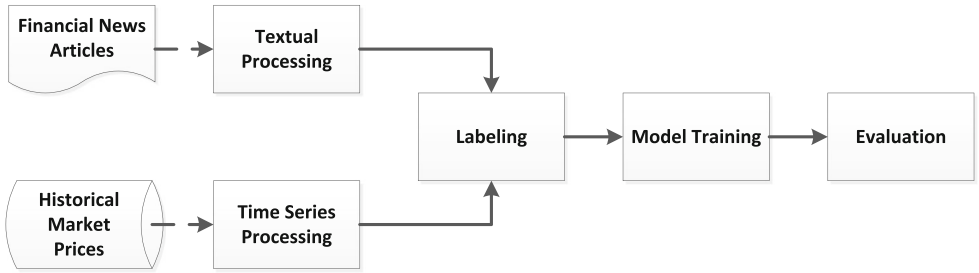

Fig. 1 Architecture of traditional approaches

vector space model after the removal of stop words and feature selection [11–14]. Sentiment analysis is occasionally employed to analyze news at a semantic level [15–18].

2. *Labeling* Each piece of news is then assigned with a label. In a classification model, pieces of news are sorted by their time stamps, and labeled with nominal values, such as *positive/neutral/negative*. While in regression approaches, news is simply labeled with a continuous value.
3. *Model training and evaluation* Machine learning models are employed in this step. Except for evaluating models by means of standard metrics, such as accuracy and mean square error (MSE).

Besides the works on news, there are also many published papers on mining signals from market prices. Gestel et al. [19] use a Bayesian evidence framework and apply it to least-squares SVM regression for price and volatility predictions. Tay et al. [20, 21] and Cao et al. [22] modify the SVM objective function and make C an adaptive parameter for non-stationary time series. For predicting index prices, Kim [23] concludes that the performance of SVM is better than that of BP-NN. By applying SVM to predict S&P 500 daily prices, Cao et al. [24, 25] also find that SVM has the better performance based on the metrics of normalized mean square error and mean absolute error. Huang et al. [26] predict the price directional movement of NIKKEI 225 index using SVM. After comparing SVM with linear discriminant analysis, quadratic discriminant analysis, and BP-NN, they reach the same conclusion. To summarize, the steps of the approaches in this category are: (1) preprocessing raw prices. Since the absolute price level contains very little information, inter-day or intra-day historical prices are usually translated into various indicators and taken as the inputs of a model. (2) patten classification. Patterns of indicators are then classified (or regressed) by SVM into predetermined categories (or estimated values).

2.3 Extreme learning machine

The basic version of ELM is a SLFN with random hidden nodes. Suppose we have N arbitrary distinct instances $(\mathbf{x}_i, \mathbf{t}_i)$, where $\mathbf{x}_i \in \mathbf{R}^n$ and $\mathbf{t}_i \in \mathbf{R}^m$. The standard SLFNs with $\tilde{N}$ hidden neurons and activation function $g(x)$ are mathematically modeled as

$$\sum_{i=1}^{\tilde{N}} \beta_i g(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) = \mathbf{o}_j, \quad j = 1, \ldots, N, \tag{3}$$

where β_i is the output weight, $\mathbf{w}_i$ is the input weight and b_i is the bias of ith hidden neuron. With the ability of zero-error approximation, i.e., $\sum_{j=1}^{\tilde{N}} \|\mathbf{o}_j - \mathbf{t}_j\| = 0$, Eq. (3) could be rewritten as

$$\mathbf{H}\beta = \mathbf{T}. \tag{4}$$

Following the naming convention in Huang et al. [3], $\mathbf{H}$ is named as the hidden layer output matrix of the neural network.

In order to find β_i, $\mathbf{w}_i$ and b_i such that

$$\|\mathbf{H}(\hat{\mathbf{w}}, \hat{b})\hat{\beta} - \mathbf{T}\| = \min_{\mathbf{w},b,\beta} \|\mathbf{H}(\mathbf{w}, b)\beta - \mathbf{T}\|, \tag{5}$$

which is the same as to minimize the error

$$E = \sum_{j=1}^{N} \left(\sum_{i=1}^{\tilde{N}} \beta_i g(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) - \mathbf{t}_j \right)^2. \tag{6}$$

Besides the commonly known gradient-based learning algorithm, Huang et al. [3] propose a solution that does not need to tune all the parameters within the network, which is to solve the linear system, i.e.,

$$\hat{\beta} = \mathbf{H}^{\dagger}\mathbf{T}, \tag{7}$$

to give the smallest norm least-squares solution. They also prove that the solution is unique (Theorem 2.1 in [3]).

Huang and Siew [4] extend the basic ELM to RBF kernel-based ELM. With the ability of infinite differential,

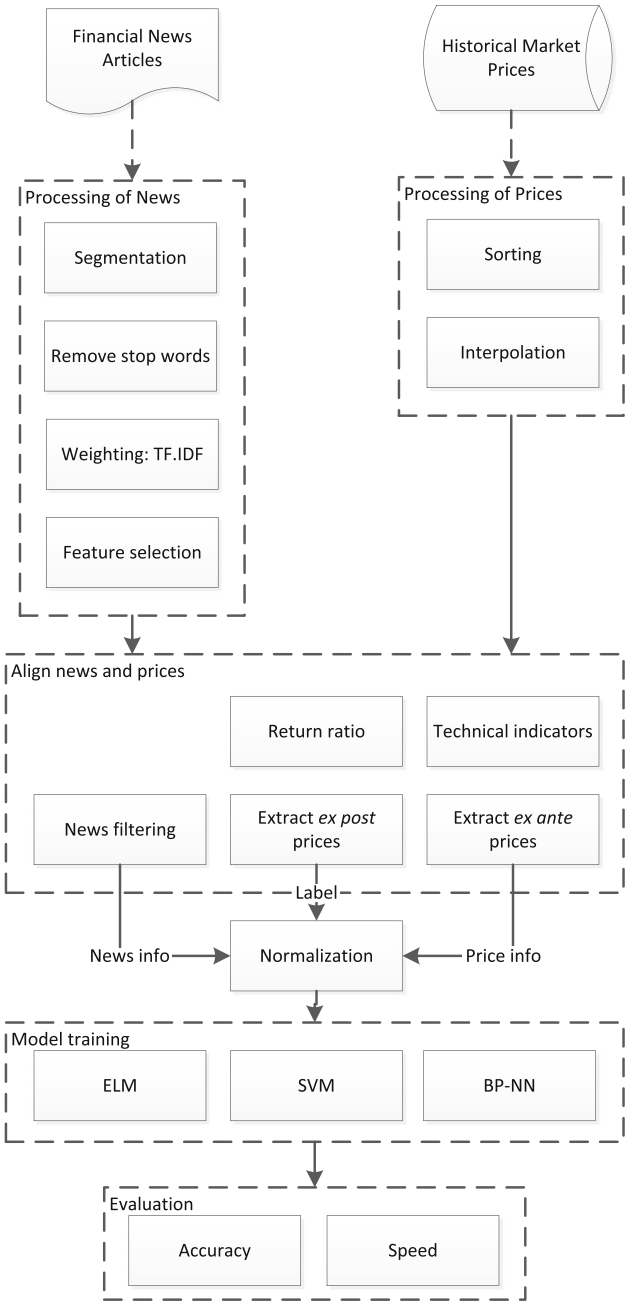

Fig. 2 Architecture of the trading signal mining platform

gaussian kernel is included in ELM, and Eq. (4) could be formulated as

$$\sum_{i=1}^{\tilde{N}} \beta_i exp\left(\frac{\|\mathbf{x}_j - \mu_i\|^2}{\sigma_i}\right) = \mathbf{t}_j, \quad j = 1, \ldots, N. \tag{8}$$

In a similar way as the basic ELM, $\hat{\beta}$ can be solved by Eq. (7), and the algorithms for solving basic and kernel ELM could be compactly written as: given a training data set, (1) arbitrarily assign parameters (for basic ELM, the parameters refer to the weights and bias; for RBF kernel ELM, the parameters refer to the kernel's mean and width); (2) calculate hidden layer output matrix **H**; (3) calculate the output weight β.

ELM has been applied to many real-life problems. Sun et al. [27] apply ELM to fashion retail sales predictions and find the performance better than BP-NN. Sun et al. [28] apply OS-ELM to p2p networks based on an ensemble classification framework. In the field of Bioinformatics, Handoko et al. [29] use ELM to predict peptides binding to the human leukocyte antigens (HLA). Sarawathi et al. [30] use a combination of integer-coded genetic algorithm (ICGA) and particle swarm optimization (PSO) together with ELM for gene selection and classification. In this paper, we apply ELM to another challenging domain—the stock market prediction.

3 Trading signal mining platform

Following the preprocessing steps in [7], we build up our trading signal mining platform, and the work flow is illustrated in Fig. 2. Section 3.1 shows how to preprocess the tick prices and news articles. Section 3.2 illustrates how to construct the instances and their labels. Section 3.3 discusses data normalization.

3.1 Preprocessing of prices and news

After sorting the tick prices by their transaction time, the tick data are represented as an unequally spaced series. Since time intervals between consecutive prices are not the same, if we sample at a predetermined frequency, it is possible that at some points there is no appropriate value. This is the issue of what value should be filled in for the blank periods. To address this issue, we use the nearest previous sampling value to fill in. This method splits tick data series by a fixed window with length w and extracts the last trade price in each w. If there is no record in region of $[w_{i-1}, w_i]$, the extracted price of $[w_{i-2}, w_{i-1}]$ is taken as the price of $[w_{i-1}, w_i]$.

Before processing the raw news articles, some of them are not included for further consideration because of two constrains: (1) market rules and (2) news impact overlap. Take H-share market for example, according to the regulation of H-share market in year 2001, continuous trading sessions are from 10:00 to 12:30, and from 14:30 to 16:00, which are the main trading hours. Since we only consider intra-day short-term predictions in our platform, only the news articles that have time stamps within the main trading

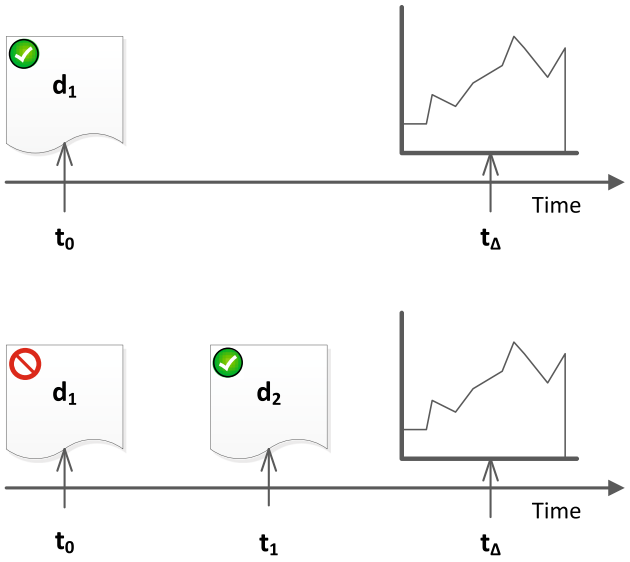

Fig. 3 Example of news filtering

hours are kept for further processing[1]. Besides the trading hour constraints, it is also suggested by Schumaker et al. [31] that we eliminate the first 20 min after the market open of the morning session and the first 20 min after the market open of the afternoon session. This is to absorb news impact in the morning before market open and lunch break. Another issue is the overlap of news impacts. As illustrated in Fig. 3, assume two news articles d_1 and d_2 are released within the same time window Δ, where both of them are related to the same company, it is then hard to determine whether the price movement at time $t_{+\Delta}$ is caused by d_1, or d_2, or both. To avoid this situation, d_1 is purposely eliminated in our platform.

News articles, as they have different nature from the prices, are preprocessed through another approach. The market news we get are all written in Chinese language, the first step in the processing pipeline is to segment the terms in the news. In addition to the Chinese segmentation software[2] we use, we also artificially build up a small market-specific dictionary to improve the segmentation accuracy. Following many previous works [1, 10, 31, 32], we keep the relatively more representative words, such as adjectives, nouns and verbs etc., and remove the less important words, i.e., the stop words, in the second step. Then, we use commonly adopted weighting scheme—$tf \cdot idf$—to calculate the weights of each word. In this way, each news article is projected onto the term space and could be simply represented as a vector. However, the length of the vector is quite long and not all the words are necessarily included in the final feature list. Feldman [33] filters out 90 % of the features and only keeps about top 10 % words as features. In our approach, we do feature selection by using the chi-square (χ^2) method that compares the difference of one vector with another vector, and gives a score for the difference, as shown in Formula (9). To be specific, the χ^2 method calculates the difference score feature-by-feature by comparing the vector of the feature with the vector of the labels. For example, let $p(t_k)$ denote the percentage that word t_k occurs in the articles, and denote $1 - p(t_k) asp(\overline{t_k})$, and $p(+)$ and $p(-)$ are the probability of class labels. Then,

$$\chi^2 = \frac{(ad - bc)^2 N}{(a+b)(a+c)(b+d)(c+d)}, \tag{9}$$

where

$$\begin{aligned} a &= N \cdot p(\overline{t_k}, -), \\ b &= N \cdot p(\overline{t_k}, +), \\ c &= N \cdot p(t_k, -), \\ d &= N \cdot p(t_k, +). \end{aligned} \tag{10}$$

The higher the χ^2 score is, the more informative the feature will be. In our platform, to make the continuous label discretized, we choose thresholds ± 0.3 % (which is usually considered as the transaction cost of the market) as the cutoffs of the future return in order to label each news, and also we choose $\alpha = 0.05$ for χ^2 and keeps top 1,000 χ^2 scored words as features.

3.2 Extract the price context for news

As shown in Eq. (2), our approach uses both the information set from news and also the one from tick prices. The way we combine the two information sources with different characteristics together is to extract the "price context" around each news event, construct features for the "price context" and then combine the features from news and the features from prices.

3.2.1 Extract and process after-event prices

In HFT, the short-term price movements after news release have attracted great attention of investors. Such "price context" is termed as *after-event* prices. Gidofalvi [34] claims that the impact of news reaches the greatest level within 20 min after it is released. To the best of our

[1] To inter-day value traders and news traders, they usually take the overnight news articles into consideration, as they analyze the mid- or long-term impacts of news, such as daily, weekly and even monthly. However, for high frequency trading, they usually make use of intra-day short-term signals. Since before the main trading hours, there is an auction session in H-share market, the impact of the news overnight is assumed to be reflected in the auction prices and *absorbed* before continuous session starts. Therefore, regarding to our platform's prediction frequency, we only keep intra-day news. This approach also follows the method in many previous works in computer science, e.g., Schumaker et al. [31].

[2] Software is downloaded on ictclas.org.

knowledge, there is neither a theoretical nor a practical method that could accurately calculate how long the news impact will last. In order to test more cases, we extract future 5, 10, 15, 20, 25 and 30 min prices after news is released and convert the prices into *simple return*, respectively, which are further used to label news articles. In this way, we extend the work [31] by testing one prediction window to six different ones.

More precisely, assume the time stamp of one piece of news is t_0, and we could find the corresponding price p_0 in the tick price series preprocessed. Similarly, we could also find the future 5, 10, 15, 20, 25 and 30 min prices in the series, denoted as p_{+5}, p_{+10}, p_{+15}, p_{+20}, p_{+25}, and p_{+30}, respectively. Then, we use Formula (11) to convert the *after-event* prices into *simple return*,

$$R = \frac{p_i - p_0}{p_0}, \tag{11}$$

where R is the simple return that we will use to label news articles. As the aforementioned trading hour constraint, there is no price during the lunch break and the market close. Thus, we need to leave a time buffer before each trading session closes. Take t_{0+20} for example, if t_0 is 15:45, which does not have 20 min before market close, we will denote this time stamp as *trading hour constraints violation* and eliminate the corresponding news article.

We use thresholds ± 0.3 % (average transaction costs in the market) as the cutoffs for the labels. In another word, if R is greater than 0.3 %, instance is labeled with positive, and news will be labeled as negative if R is less than -0.3 %.

3.2.2 Extract and process before-event prices

In contrast to the *after-event* prices, the short-term historical prices are named as *before-event* prices in our setting. We sample at the frequency of 1 sample per 1 min on the price series before the news is released. One way to construct features from *before-event* prices is simply to use the 30 sampled prices as 30 features. However, machine learning model, such as ELM and SVM, will assume that features are not dependent on each other. Suppose we permute the position of the 30 values, learning model will treat them the same as before. In other words, the *sequential* information is not kept. To address this issue, we follow Cao and Tay [20, 35] approach, which uses RDP indicators to *summarize* the short price series. The formulae of RDPs are listed in Table 1, we use the same formulae to convert *before-event* prices into indicators.

Follow the thinking of converting prices into indicators, we also adopt some other technical indicators from stock technical analysis in addition to RDPs. The formulae and descriptions of technical indicators are listed in Table 2, where p_i is the price at time point i and q is the order of the formula. In such a way of conversion, 30 *before-event* price points are summarized by 6 RDPs and five technical indicators. We will simply refer them as indicators in the following sections.

Table 1 The formulae of RDPs

RDP	Formula
RDP-5	$100*(p_i - p_{i-5})/p_{i-5}$
RDP-10	$100*(p_i - p_{i-10})/p_{i-10}$
RDP-15	$100*(p_i - p_{i-15})/p_{i-15}$
RDP-20	$100*(p_i - p_{i-20})/p_{i-20}$
RDP-25	$100*(p_i - p_{i-25})/p_{i-25}$
RDP-30	$100*(p_i - p_{i-30})/p_{i-30}$

3.3 Normalization

According to the pipelines proposed in Fig. 2, many instances have been generated: (1) instances of news articles. Each news instance is represented by a row vector and all the vectors form a matrix N. (2) instances of *before-event* prices. Each price instance is represented by a row vector of indicators and all the vectors form a matrix I. (3) A column vector L that contains the labels of all instances. Since each instance in N and I is generated from the same news and its "price context", the numbers of rows in matrix N, I and L are the same as the numbers of news articles.

The normalization step is divided into two parts: (1) For those features that only take non-negative values, such as the $tf \cdot idf$ values of news features, we use Eq. (12) to normalize them,

$$norm(f_{in}) = \frac{f_{in} - \min\{f_{*n}\}}{\max\{f_{*n}\} - \min\{f_{*n}\}}, \tag{12}$$

where f_{in} is the ith value in the nth feature and $\max(f_{*n})$ ($\min(f_{*n})$) is the greatest (smallest) value in the nth feature. After normalization, the range of the value is [0, 1]. (2) For those features that could take both negative and non-negative values, such as some features of the indicators, we use Eq. (13) to normalize them,

$$norm(f_{in}) = \frac{f_{in}}{\max\{|f_{*n}|\}}. \tag{13}$$

After normalization, the range of the value is $[-1, 1]$.

4 Experimental results and discussions

4.1 Data sets

We use two data sets, namely the market news and stock tick prices:

Table 2 Stock technical indicators

Indicator	Description	Formula
RSI(q)	Relative Strength Index	$100*Up/(Up + Down)$ $Up = \sum_{p_i > (\sum_i p_i)/q}(p_i - (\sum_i p_i)/q)$ $Down = \sum_{p_i < (\sum_i p_i)/q}(p_i - (\sum_i p_i)/q)$
RSV(q)	Raw Stochastic Value	$100 * (p_0 - \min_q(p_i))/(\max_q(p_i) - \min_q(p_i))$
R(q)	Williams Index	$100 * (\max_q(p_i) - p_0)/(\max_q(p_i) - \min_q(p_i))$
BIAS(q)	Bias	$100 * (p_0 - (\sum_i p_i)/q)/((\sum_i p_i)/q)$
PSY(q)	Psychological Line	$100 * (\sum 1\{p_i > p_{i-1}\})/q$

Table 3 The numbers of instances after preprocessing

	5 m	10 m	15 m	20 m	25 m	30 m
Total	1,721	1,953	2,035	1,965	1,963	1,906

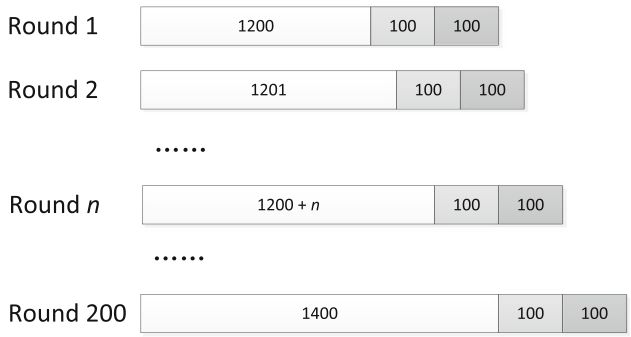

Fig. 4 Rolling window mode of each round

Market news We bought one year market news archive from Caihua (http://www.finet.hk). The news archive tags each news article with a time stamp, where the time is the release time of the news. News are in Traditional Chinese, UTF-16LE coded.

Stock tick prices The stock tick prices contain all the tick prices of all the names listed in H-share market in year 2001.

H-share market lists thousands of stocks. However, only a proportion of them are traded actively, which are mainly the liquid stocks of Hang Seng Index constituents (HSI, http://www.hsi.com.hk). According to the change history of HSI, in year 2001, HSI has 33 stocks and there are two replacements of the constituents within the year, which happened on 1st June and 31th July. Due to the *tyranny of indexing* [36], which observes that during the first few months, the price behaviors of the newly added constituents do not appear rational and they are usually mispriced. Therefore, only the stocks that are the constituents throughout the whole year are selected. As a consequence, the total number of stocks reduces to 23.

There are 28,885 pieces of news in sum. The numbers of instances after preprocessing are listed in Table 3.

As illustrated in Fig. 4, we split the data set for each prediction time horizon into *training*, *validation* and *testing* sets. The size of validation and testing sets is fixed as 100 instances. The size of training set starts from 1,200 instances and increases by 1 instance after each round of running. Note that, the three sets are sequentially located, and there is no overlap between them.

4.2 Model setup and parameter tuning

We setup four state-of-the-art models in the framework for comparison:

Back-propagation neural network (BP-NN) Following Huang et al. [3], we include a fast BP algorithm named Levenberg–Marquardt into our experiments[3]. The parameter to be tuned for BP-NN is the number of hidden nodes, #(*hidden_nodes*)[4]. As required by the convergence condition of BP-NN, the parameter should be greater than or equal to #(*input_nodes*) which are 1,011 features in our problem. After several trials of running, we find it impossible for our server to support that high number of nodes due to the limit of memory. We then empirically set #(*hidden_nodes*) = 10 in order to balance the hardware feasibility and BP-NN.

Support vector machines (SVM) As reviewed in Sect. 2, SVM is widely applied in finance domain. The parameters to be tuned for SVM are the kernel type and related kernel parameters. Since RBF SVM is reported to have better accuracy on text classification problem, RBF kernel is adopted in our setting. Grid search is applied, where kernel's parameter γ searches $\{2^{-17}, 2^{-16}, \ldots, 2^{2}\}$ and penalty parameter C searches $\{2^{-5}, 2^{-4}, \ldots, 2^{14}\}$, and the best combination among the $20 \times 20 = 400$ pairs is selected based on SVM's accuracy on the validation set.

Basic ELM (B-ELM) Similar to BP-NN, B-ELM has only the number of hidden nodes to be tuned. Unlike BP-

[3] Levenberg–Marquardt algorithm has its implementation in MATLAB 7.14 toolbox.

[4] Notation #(*X*) indicates the number of object X.

Table 4 The definition of *tp*, *tn*, *fp* and *fn*

	Predict +	Predict 0	Predict −
True +	t_{++}	f_{+0}	f_{+-}
True 0	f_{0+}	t_{00}	f_{0-}
True −	f_{-+}	f_{-0}	t_{-}

Table 5 Results of the validation and testing accuracy

Validation	5 m		10 m		15 m	
	Avg.	Dev.	Avg.	Dev.	Avg.	Dev.
BP-NN	0.521	0.045	0.538	0.054	0.548	0.059
SVM	**0.625**	0.024	**0.661**	0.023	**0.694**	0.029
B-ELM	0.499	0.054	0.492	0.049	0.503	0.047
K-ELM	0.605	0.021	0.650	0.029	0.680	0.027
Validation	20 m		25 m		30 m	
	Avg.	Dev.	Avg.	Dev.	Avg.	Dev.
BP-NN	0.551	0.051	0.556	0.053	0.520	0.043
SVM	**0.686**	0.029	**0.659**	0.022	**0.643**	0.031
B-ELM	0.505	0.049	0.513	0.048	0.513	0.048
K-ELM	0.667	0.029	0.649	0.034	0.633	0.050
Testing	5 m		10 m		15 m	
	Avg.	Dev.	Avg.	Dev.	Avg.	Dev.
BP-NN	0.499	0.046	0.526	0.055	0.537	0.058
SVM	0.544	0.060	0.573	0.047	**0.644**	0.031
B-ELM	0.493	0.051	0.506	0.047	0.499	0.052
K-ELM	**0.570**	0.049	**0.584**	0.033	0.607	0.044
Testing	20 m		25 m		30 m	
	Avg.	Dev.	Avg.	Dev.	Avg.	Dev.
BP-NN	0.557	0.061	0.553	0.052	0.522	0.049
SVM	**0.625**	0.028	**0.593**	0.040	0.535	0.036
B-ELM	0.500	0.045	0.522	0.046	0.506	0.049
K-ELM	0.597	0.024	0.551	0.043	**0.536**	0.037

NN, B-ELM does not require much memory while running. Therefore, following the guideline that

$$\#(input\_nodes) \leq \#(hidden\_nodes)$$

and

$$\#(hidden\_nodes) \leq \#(instances),$$

we arbitrarily set #(*hidden_nodes*) = 1100 which is in the range [1011, 1200].

Kernel ELM (K-ELM) K-ELM has the similar tuning process as the SVM case. For comparison, the RBF kernel is also adopted for K-ELM. The γ of RBF kernel searches $\{2^{-17}, 2^{-16}, \ldots, 2^{2}\}$ and regulation term searches $\{2^{-5}, 2^{-4}, \ldots, 2^{14}\}$. As illustrated in Fig. 4, the training, validation and testing round advances 1 instance each time, and we test 200 rounds in total.

4.3 Experimental results and findings

We evaluate the models from both the prediction accuracy and speed. The accuracy is measured by

$$acc = \frac{t_{++} + t_{00} + t_{--}}{all}, \tag{14}$$

and

$$all = t_{++} + t_{00} + t_{--} + f_{0+} + f_{-+} + f_{+0} + f_{-0} + f_{+-} + f_{0-}, \tag{15}$$

where t_{++}, t_{00}, t_{-}, f_{0+}, f_{-+}, f_{+0}, f_{-0}, f_{+-} and f_{0-} are defined in Table 4.

Model prediction speed is measured by CPU time.

The accuracy of each model at each time point is listed in Table 5. We calculate the average accuracy (shown in avg. column) over the 200 rounds as well as the standard deviation (shown in dev. column).

From the accuracy results, we can see that

SVM and K-ELM perform better than the other two models in the validation set. Among the 6 time points, SVM achieves 6 the best results, which are marked in bold font. K-ELM achieves 6 the second best results, which are marked by underline. We also use a statistical testing method *t* test to determine the difference significance between SVM and K-ELM, and the result is presented in Table 6, from which we can observe that SVM significantly outperforms K-ELM on validation set.

K-ELM achieves the same level of accuracy as SVM in the testing phase. Among the 6 time points, SVM achieves 3 the best results and 3 the second best results, while K-ELM achieves the other 3 the best results and 3 the second best results. Besides the average and standard deviation, we draw the one-to-one comparison in Fig. 5.

Within each sub-figure, the blue diagonal line denotes that K-ELM has the same accuracy as SVM; red circle means on that specific testing set (while using rolling window mode, there are 200 testing sets in total.), K-ELM has higher accuracy than SVM; on the other hand, black plus means SVM achieves higher accuracy than K-ELM on that testing set. It could be clearly observed that in sub-figures (1), (2) and (6), K-ELM is better than SVM, and in (3), (4) and (5) SVM is better.

Table 6 *p* Value of *t* test between K-ELM and SVM

p-value	5 m	10 m	15 m	20 m	25 m	30 m
SVM v.s. K-ELM	2.8e−42	8.6e−18	1.9e-07	1.6e−13	1.7e−07	6.7e−04
Hypothesis	5 m	10 m	15 m	20 m	25 m	30 m
SVM outperforms K-ELM?	Yes	Yes	Yes	Yes	Yes	Yes

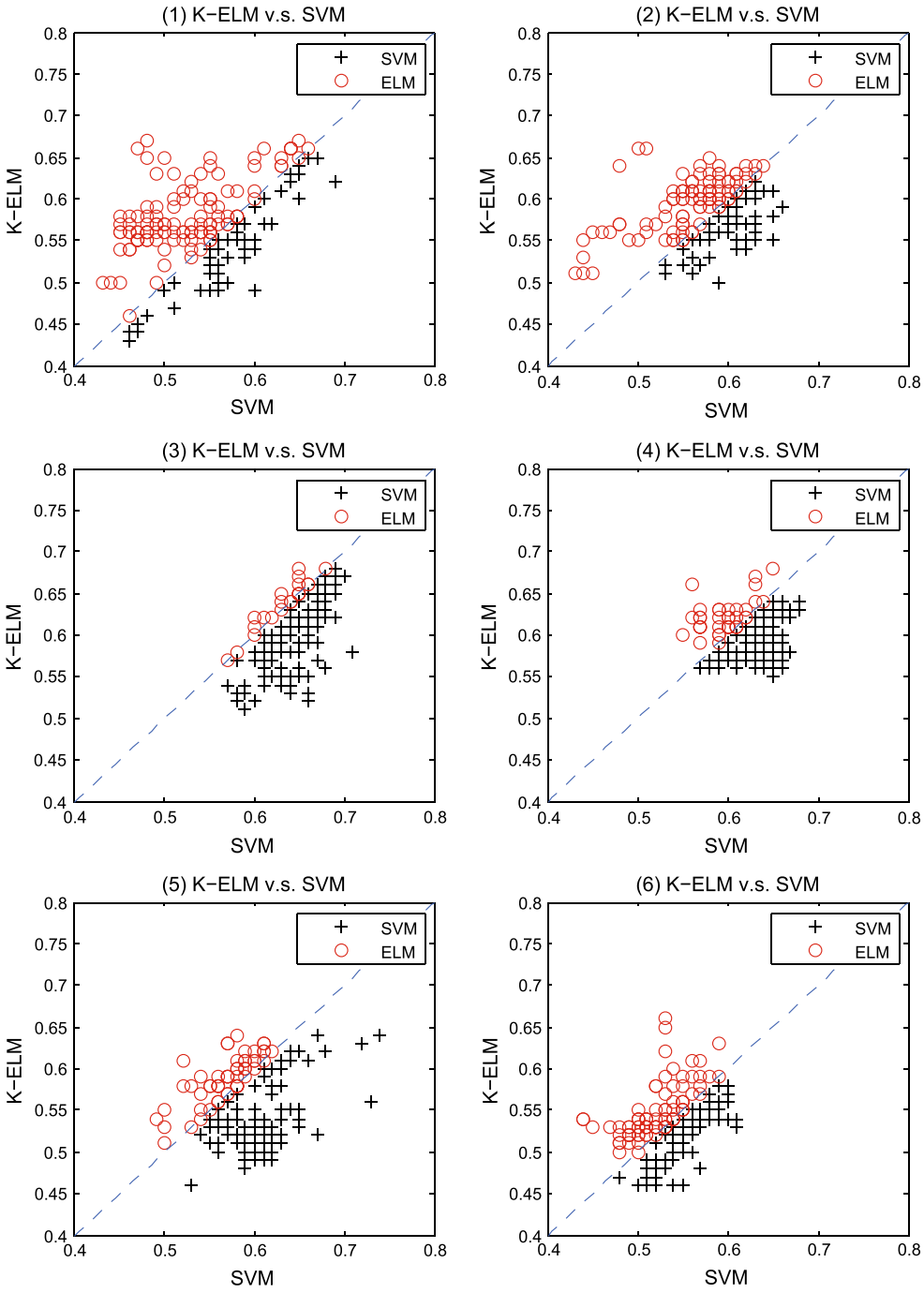

Fig. 5 One-to-one comparison between K-ELM and SVM

Table 7 Results of the prediction speed

Prediction (100 instances)	5 m		10 m		15 m	
	Avg.	Dev.	Avg.	Dev.	Avg.	Dev.
BP-NN	0.200	0.055	0.165	0.063	0.190	0.114
SVM	0.266	0.025	0.261	0.026	0.266	0.030
B-ELM	0.149	0.072	0.149	0.084	0.155	0.094
K-ELM	**0.138**	0.084	**0.148**	0.092	**0.144**	0.089
Prediction (100 instances)	20 m		25 m		30 m	
	Avg.	Dev.	Avg.	Dev.	Avg.	Dev.
BP-NN	0.178	0.068	0.197	0.059	0.192	0.055
SVM	0.252	0.026	0.261	0.046	0.279	0.032
B-ELM	0.160	0.090	0.160	0.094	0.151	0.077
K-ELM	**0.140**	0.070	**0.140**	0.090	**0.131**	0.060

Table 8 Key software and hardware specs of our server

Software	
BP-NN	MATLAB 7.14
SVM	LIBSVM 3.17, MATLAB interface
B-ELM	MATLAB 7.14
K-ELM	MATLAB 7.14
Hardware	
CPU	4 core 8 thread, 2.53 GHz each
Mem	16 GB DDR3 1333 MHz
OS	RedHat Enterprise Linux 5.4

The results of the prediction speed of each model at each time point are listed in Table 7. The key software and hardware specifications of our server that we think may affect the performance of the models are listed in Table 8. Since the training speed is not critical in trading[5], we compare the testing speed of each model over 100 testing instances, 200 rounds.

From the results, we can see that both B-ELM and K-ELM run faster than SVM and BP-NN, among which K-ELM is the fastest on our machine (K-ELM has 6 best results, and B-ELM has 6 second best results). This is due to the advantage of ELM that the model keeps fewer nodes than SVM's support vectors while transferring from the training phase to the testing phase.

We also examine the hardware requirement of K-ELM. We set up a background monitor which captures the runtime CPU and memory usage of one K-ELM process from the training phase to the testing phase. The results are illustrated in Fig. 6.

The monitor samples once per second. It is surprised to see that the CPU usage of K-ELM is high. It reaches 400 % CPU power during the peak time, which means that 2 CPU cores are dedicated to K-ELM. Comparing to SVM, the number is no more than 50 % (using half one core) during the peak time. If the system needs to analyze many stocks in parallel and the hardware resource is not enough, this would be a disadvantage for ELM that processes would race for the CPU and further slow the prediction speed.

We design and implement a preliminary market making strategy that adopts the signals generated by the models, which (1) places limit orders on both best bid and ask prices of the order book; (2) readjusts (cancels and resubmits) the orders when signals change; (3) unwinds the inventory when market closes. Simulations with out-of-sample historical prices and news of 0005.HK are conducted for the strategy. The measurements we use are daily profit and loss (PnL) and Sharpe ratio, where Sharpe ratio is defined as

$$\text{Sharperatio} = \frac{avg(PnL)}{dev(PnL)}. \tag{16}$$

Sharpe ratio that is greater than 1 indicates that the strategy is making profit with less risk. The simulation results are presented in Table 10.

As we can see from the results, strategies with more accurate signals (SVM and K-ELM) can make more profits than the other two models.

5 Conclusion and future work

How to model market data and make both accurate and fast short-term stock price predictions in stock market is an

[5] Online dynamic reconfiguration is not discussed in the paper, since all the four models are not online algorithms and training can be setup as an overnight job. We record the training time of each model and present them in Table 9. As we can see from the results, the training time of four models is much longer than validation and testing time, and the time of BP-NN is much longer than the time of the other three models.

Table 9 Results of the training speed

Training	5 m	10 m	15 m	20 m	25 m	30 m
BP-NN	2,222.738	2,878.992	3,383.253	4,687.365	2,539.905	2,107.953
SVM	234.048	164.011	198.750	180.899	196.266	328.021
B-ELM	13.784	13.974	13.743	13.739	14.028	13.809
K-ELM	55.799	49.958	52.963	53.244	59.636	55.184

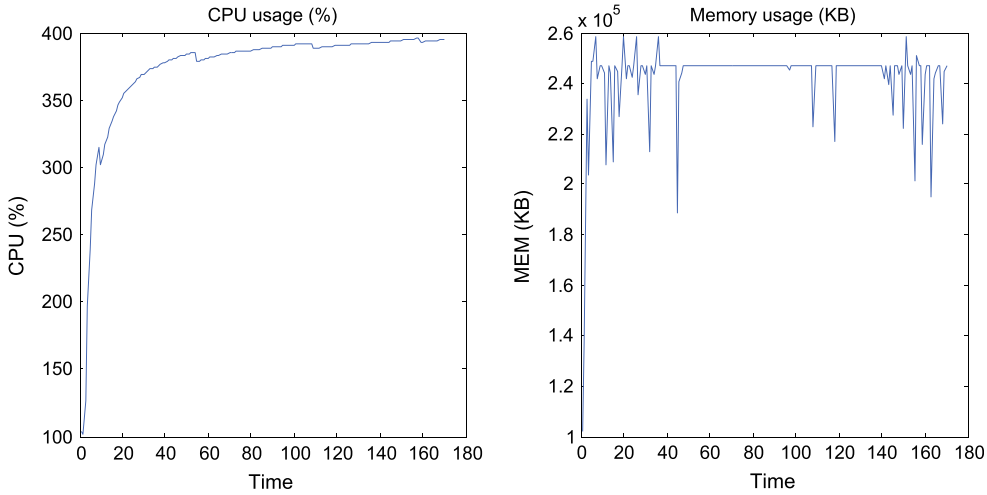

Fig. 6 Runtime CPU and memory usage of K-ELM

Table 10 Simulation results for 0005.HK

0005.HK	PnL avg.	PnL dev.	Sharpe ratio
BP-NN	20.5	51.6	0.4
SVM	40.3	23.2	1.7
B-ELM	21.1	49.9	0.4
K-ELM	41.0	23.3	1.8

attractive problem. In this paper, we build up a trading signal mining platform that makes use of the two most important information sources, i.e., market news articles and stock tick prices. For the sake of high prediction accuracy and fast prediction speed, we adopt B-ELM and K-ELM as the core algorithms in our platform for information integration and price movement predictions. We empirically evaluate the performance of the platform by comparing ELM with another two state-of-the-art learning algorithms, i.e., BP-NN and SVM. Experiments have been conducted on the intra-day tick-by-tick data of 23 stocks in the H-share market and corresponding commercial news archives. From the results, we find that

Both the kernelized ELM and the RBF SVM achieve higher prediction accuracy and faster prediction speed than the BP-NN and the basic version of ELM;
The kernelized ELM achieves similar accuracy with the RBF SVM;
The kernelized ELM has faster prediction speed than the RBF SVM.

A preliminary market making strategy using the signals is back-tested on out-of-sample historical data. The simulation results give supportive evidences that strategies with more accurate signals make more profits with less risk. We also notice that the kernelized ELM requires more CPU resources than the RBF SVM, which would cause running processes racing for hardware resources, and the CPU scheduling will thus become the bottleneck. How to reduce the CPU requirement of the kernelized ELM while keep the fast prediction speed is the problem that we need to investigate in the future.

Acknowledgments This work was partly supported by National Natural Science Foundation of China (Grant No. 61300137); the Guangdong Natural Science Foundation, China (Grant No. S2011040002222); the Fundamental Research Funds for the Central Universities, SCUT (Grant No. 2012ZM0077). This work was also supported by Shenzhen New Industry Development Fund under grant No. JCYJ20120617120716224.

References

1. Schumaker RP, Chen H (2010) A discrete stock price prediction engine based on financial news. Computer 43(1):51–56
2. Yeh C-Y, Huang C-W, Lee S-J (2011) A multiple-kernel support vector regression approach for stock market price forecasting. Expert Syst Appl 38:2177–2186
3. Huang G-B, Zhu Q-Y, Siew C-K (2006) Extreme learning machine: theory and applications. Neurocomputing 70(1):489–501

4. Huang G-B, Siew C-K (2004) Extreme learning machine: RBF network case. In: Control, automation, robotics and vision conference, ICARCV'04, vol 2, pp 1029–1036
5. Wei X-K, Li Y-H, Feng Y (2006) Comparative study of extreme learning machine and support vector machine. In: Advances in neural networks, ISNN'06, vol 3971. Springer, Berlin, Heidelberg, pp 1089–1095
6. Huang G-B, Wang DH, Lan Y (2011) Extreme learning machines: a survey. Int J Mach Learn Cybernet 2(2):107–122
7. Li X, Wang C, Dong J, Wang F, Deng X, Shanfeng Z (2011) Improving stock market prediction by integrating both market news and stock prices. In: Hameurlain A, Liddle S, Schewe K-D, Zhou X (eds) Database and expert systems applications. Lecture notes in computer science, volume 6861. Springer, Berlin, pp 279–293
8. Seo Y-W, Giampapa J, Sycara K (2004) Financial news analysis for intelligent portfolio management. PhD thesis, Robotics Institute, Carnegie Mellon University
9. Yu L, Yue YW, Wang S, Lai KK (2010) Support vector machine based multiagent ensemble learning for credit risk evaluation. Expert Syst Appl 37(2):1351–1360
10. Schumaker RP, Chen H (2009) Textual analysis of stock market prediction using breaking financial news: The azfin text system. ACM Trans Inf Syst 27(2):12:1–12:19
11. Fu T-C, Chung F-L, Ng V, Luk R Pattern discovery from stock time series using self-organizing maps. In: Workshop notes of workshop on temporal data mining, KDD'01
12. Kohonen T (1982) Self-organized formation of topologically correct feature maps. Biol Cybern 43:59–69
13. Kohonen T, Somervuo P (1998) Self-organizing maps of symbol strings. Neurocomputing 21(1–3):19–30
14. Ultsch A (1999) Data mining and knowledge discovery with emergent self-organizing feature maps for multivariate time series. Kohonen Maps 46:33–46
15. Godbole N, Manjunath S, Steven S (2007) Large-scale sentiment analysis for news and blogs. ICWSM'07, Boulder, Colorado, USA
16. Kim S-M, Hovy E (2004) Determining the sentiment of opinions. In: Proceedings of the 20th international conference on computational linguistics, COLING'04, pp 1367–1373, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics
17. Pang B, Lee L (2008) Opinion mining and sentiment analysis. Found Trends Inf Retr 2(1–2):1–135
18. Pang B, Lee L, Vaithyanathan S (2002) Thumbs up?: sentiment classification using machine learning techniques. In Proceedings of the conference on empirical methods in natural language processing, EMNLP'02, pp 79–86, Stroudsburg, PA, USA. Association for Computational Linguistics.
19. Van Gestel T, Suykens JAK, Baestaens D-E, Lambrechts A, Lanckriet G, Vandaele B, De Moor B, Vandewalle J (2001) Financial time series prediction using least squares support vector machines within the evidence framework. IEEE Trans Neural Netw 12(4):809–821
20. Tay FEH, Cao L (2001) Application of support vector machines in financial time series forecasting. Omega 29(4):309–317
21. Tay FEH, Cao L (2002) Modified support vector machines in financial time series forecasting. Neurocomputing 48(1):847–861
22. Cao L, Gu Q (2002) Dynamic support vector machines for non-stationary time series forecasting. Intell Data Anal 6(1):67–83
23. Kyoung-Jae K (2003) Financial time series forecasting using support vector machines. Neurocomputing 55(1–2):307–319
24. Cao L, Tay FEH (2001) Financial forecasting using support vector machines. Neural Comput Appl 10(2):184–192
25. Cao L, Tay FEH (2003) Support vector machine with adaptive parameters in financial time series forecasting. IEEE Trans Neural Netw 14(6):1506–1518
26. Huang W, Nakamori Y, Wang S-Y (2005) Forecasting stock market movement direction with support vector machine. Comput Oper Res 32(10):2513–2522
27. Sun Z-L, Choi T-M, Au K-F, Yu Y (2008) Sales forecasting using extreme learning machine with applications in fashion retailing. Decis Support Syst 46(1):411–419
28. Sun Y, Yuan Y, Wang G (2011) An os-elm based distributed ensemble classification framework in p2p networks. Neurocomputing 74(16):2438–2443
29. Handoko SD, Keong KC, Soon OY, Zhang GL, Brusic V (2006) Extreme learning machine for predicting hla-peptide binding. In: Advances in neural networks, ISNN'06, vol 3973. Springer, Berlin, Heidelberg, pp 716–721
30. Saraswathi S, Sundaram S, Sundararajan N, Zimmermann M, Nilsen-Hamilton M (2011) Icga-pso-elm approach for accurate multiclass cancer classification resulting in reduced gene sets in which genes encoding secreted proteins are highly represented. IEEE/ACM Trans Comput Biol Bioinf 8(2):452–463
31. Schumaker RP, Chen H (2009) A quantitative stock prediction system based on financial news. Inf Process Manag 45(5):571–583
32. Schumaker RP, Chen H (2006) Textual analysis of stock market prediction using financial news articles. In: Americas conference on information systems, AMCIS'06, vol 1, pp 1–20
33. Feldman R, Sanger J (2007) The text mining handbook. Cambridge University Press, Cambridge
34. Gidófalvi G (2001) Using news articles to predict stock price movements. PhD thesis, Department of Computer Science and Engineering, University of California
35. Cao L, Tay FEH (2004) Support vector machine with adaptive parameters in financial time series forecasting. IEEE Trans Neural Netw 14(6):1506–1518
36. Ritter JR (2003) Behavioral finance. Pac Basin Financ J 11(4):429–437