Evolving Least Squares Support Vector Machines for Stock Market Trend Mining *

Lean Yu [1 2 †]***, Huanhuan Chen*** [3]***, Shouyang Wang*** [1]***, Kin Keung Lai*** [2]

[1] Institute of Systems Science, Academy of Mathematics and Systems Science,
Chinese Academy of Sciences, Beijing 100190, China
[2] Department of Management Sciences, City University of Hong Kong,
Tat Chee Avenue, Kowloon, Hong Kong
[3] School of Computer Science, University of Birmingham,
Edgbaston, Birmingham, B15 2TT, United Kingdom

Abstract — In this study, an evolving least squares support vector machine (LSSVM) learning paradigm with a mixed kernel is proposed to explore stock market trends. In the proposed learning paradigm, a genetic algorithm (GA), one of the most popular evolutionary algorithms (EAs), is first used to select input features for LSSVM learning, i.e., evolution of input features. Then another GA is used for parameters optimization of LSSVM, i.e., evolution of algorithmic parameters. Finally, the evolving LSSVM learning paradigm with best feature subset, optimal parameters and a mixed kernel is used to predict stock market movement direction in terms of historical data series. For illustration and evaluation purposes, three important stock indices, S&P 500 Index, Dow Jones Industrial Average Index, and New York Stock Exchange Index, are used as testing targets. Experimental results obtained reveal that the proposed evolving LSSVM can produce some forecasting models that are easier to be interpreted by using a small number of predictive features and are more efficient than other parameter optimization methods. Furthermore, the produced forecasting model can significantly outperform other forecasting

* This work is partially supported by NSFC, CAS, AMSS, SRG of City University of Hong Kong.
† Corresponding Author. Tel.: 8610-62565817, Fax: 8610-62541823. Email: yulean@amss.ac.cn.

models listed in this study in terms of the hit ratio. These findings imply that the proposed evolving LSSVM learning paradigm can be used as a promising approach to stock market tendency exploration.

Index Terms — Least squares support vector machine, evolutionary algorithms, feature selection, parameter optimization, mixed kernel, genetic algorithm, statistical models, artificial neural networks, stock market trend mining

I. INTRODUCTION

Mining stock market trend or predicting stock price movement direction is regarded as a rather challenging task due to its high volatility, irregularity and noisy environment in stock markets. Usually, the difficulty in predicting stock price movement direction is attributed to the limitations of many conventional linear forecasting models. For example, some researchers found that many standard econometric models are unable to produce significantly better predictions than the random walk model [1], which has also encouraged academic researchers and business practitioners to develop more predictable models. Recent studies reveal that nonlinear models are able to simulate the volatile stock markets well and produce better predictive results than traditional linear models in stock market tendency exploration [2]. Of various nonlinear models, the artificial neural networks (ANNs) are considered as a class of strong alternatives to predicting stock price movement direction. As claimed by Grudnitski and Osburn [3], ANNs are particularly well suited for finding accurate solutions in an environment characterized by complex, noisy, irrelevant or partial information. Furthermore, ANNs have been proved to be a class of universal function approximators that can map any nonlinear function without any *a priori* assumption about the data [4]. For these reasons, ANNs have been widely applied to stock

market prediction [5].

In all neural network applications, multi-layer feedforward neural network (MLFNN) is used most frequently for stock market price prediction [1, 6-12]. Besides the MLFNN, other neural network types, such as probabilistic neural network (PNN) [13-15] recurrent neural network (RNN) [16-17] are also applied to stock market price prediction. Particularly, when predicting stock market price, some researchers attempt to hybridize some novel factors into the neural network learning process to improve the prediction performance. For example, Kohara et al. [17] incorporated prior knowledge into the neural network learning to improve the performance of stock market prediction. Tsaih et al. [18] integrated the rule-based technique and ANN to predict the S&P 500 stock index future price based on daily data. Similarly, Kim and Han [19] proposed a genetic algorithm (GA) approach to discretizing input features and determining connection weights for ANN to predict the stock price index. They suggested that their approach reduced the dimensionality of the feature space and enhanced the prediction performance. A recent good survey about stock market prediction with ANN can refer to Huang et al. [5] for more literature.

Numerous successful applications have shown that ANN is a very useful tool for stock market modeling and forecasting, but some studies have also revealed that ANN often exhibits inconsistent results due to the limitations of ANN itself [20]. Furthermore, in some practical applications, local minima and overfitting are often encountered in ANN modeling. To overcome these shortcomings, support vector machine (SVM) first proposed by Vapnik [21] was introduced in the 1990s. Compared with the ANN, the main advantages of the SVM reflect two-fold. On the one hand, the SVM is implemented by the structural risk minimization (SRM) principle, which searches to minimize an upper bound of generalization error. Thus, the solution of the SVM may be a global optimum rather than a local optimum. On the other hand, the SVM can minimize the risk of overfitting by choosing the maximal margin hyperplane in feature space

[22]. Due to these characteristics, the SVM models are attracting more and more attention. Typical examples are Huang et al. [2], Kim [23] and Yu et al. [24].

However, a predictive SVM model often suffers from much difficulty in improving computational efficiency, optimizing model parameters and selecting relevant input features. First of all, the Vapnik's SVM model requires solving a quadratic programming (QP) problem and thus it is very slow when a large-scale practical problem is given. Second, in the SVM modeling, some important parameters such as upper bound parameter and kernel parameters are not optimized, which may affect the generalization performance of SVM. Third, a predictive SVM model also encounters much difficulty in selecting some important features. Furthermore, the problem may become more intractable when the model interpretability is important. For example, in stock market trend exploration, it is critical for decision-makers to understand the key drivers of affecting stock price movement. However, a predictive SVM model that is essentially a "black box" is not helpful for developing comprehensive forecasting models [25].

In order to provide a good solution to the above problems, this study proposes an evolving least squares support vector machine (LSSVM) [26-27] learning paradigm with the best feature subset, optimal model parameters and a mixed kernel for stock market trend mining. The main reasons for proposing the evolving LSSVM reflect three-fold. First of all, the LSSVM has excellent generalization performance and low computational cost [26] relative to the standard SVM proposed by Vapnik [21]. Second, some empirical experiments [26-28] also confirmed the efficiency of LSSVM. Third, some evolving strategies [29-34] are usually efficient to solve some difficult optimization problems. Concretely speaking, the proposed evolving LSSVM learning paradigm consists of the following two main components: (1) GA-based feature selection, i.e., feature evolution component and (2) GA-based parameter optimization, i.e., parameter evolution component.

In the feature evolution component, a standard genetic algorithm (GA) [35], the most popular type of evolutionary algorithm (EA), is used to select important input features for LSSVM learning. In this component, two key goals, forecasting performance (or predictive accuracy) and model complexity (or model interpretability) constitute the evaluation fitness function of the GA. Note that these two goals are often in conflict. In this study, we try to arrive at a trade-off between performance and complexity from the following two aspects.

On the one hand, we will build a simplified predictive model that integrates LSSVM with a GA to improve the prediction performance. In this component, we first utilize GA to identify some key variables of affecting stock price movement and then use the selected variables to train LSSVM. This can be done by learning linear or possibly nonlinear relationships between the given input variables and the dependent variable. Because the original input variables may contain some redundant information, reducing some redundant variables may improve the prediction performance.

On the other hand, we enhance the interpretability of the predictive model by reducing the data dimensionality using GA. In this component, GA is used to select a subset of original features thus simplifying the LSSVM model and increasing the model interpretability. Usually, data dimensionality reduction can be done via feature selection. Generally, feature selection algorithms such as principal component analysis (PCA) have been often used for this purpose. However, in this study, the PCA is not appropriate because our goal is not only to reduce the data dimensionality, but also to obtain highly accurate predictive models. But the PCA does not consider the relationship between the response variable and other input variables in the process of data reduction and thus it is difficult to produce a highly accurate model. Furthermore, the resulting principal components from the PCA can be difficult to be interpreted when the dimensionality of input variables is huge. On the contrary, the GA has proved to have superior

performance to other algorithms for dataset with high dimensionality [36]. In the feature evolution process, if we extract as much information as possible from a given data set while using the smallest number of features, we can not only save much computational cost, but also build a simplified LSSVM model with better generalization. Furthermore, feature selection can also significantly improve the comprehensibility of the resulting models. Even a complicated model can be more easily understood if constructed from only a few variables.

In the parameter evolution component, another GA is used to optimize parameters of LSSVM. Usually, the LSSVM generalization ability is controlled by kernel type, kernel parameters and upper bound parameter. Every kernel type has its advantages and disadvantages and thus a mixed kernel [37-38] is introduced into the LSSVM learning paradigm in this study. In this component, kernel combination coefficients, kernel parameters and upper bound parameter are also evolved and optimized. In this component, the forecasting performance or predictive accuracy is used as the evaluation fitness function of the GA. Note that the forecasting performance is the average predictive accuracy via k-fold cross validation.

In sum, the proposed evolving LSSVM learning paradigm makes full use of the desirable characteristics of GA and LSSVM models to achieve two principal goals: model interpretability and predictive accuracy. The detailed process is as follows. A standard GA is used to select the possible combination of features. The input features selected by GA are used to train LSSVM. The trained LSSVM is tested on an evaluation set, and a proposed model is evaluated in terms of two evaluation criteria: prediction accuracy (which is maximized) and model complexity (which is minimized). This process is repeated many times as the algorithm searches for a desirable trade-off between predictive accuracy and model complexity. The final results obtained is a highly accurate predictive model that uses only a subset of initial features, thus simplifying the model and providing some useful information on future data collection work.

The main motivation of this study is to propose a new evolving LSSVM learning paradigm integrating LSSVM with GA for exploring stock market tendency and to test the predictability of the proposed learning paradigm by comparing it with statistical models and neural network models. The rest of the study is organized as follows. The next section gives a brief introduction of SVM and LSSVM. The new evolving LSSVM learning paradigm is described in Section III in detail. In Section IV the research data and comparable forecasting models are presented. The experimental results are reported in Section V. Section VI concludes the paper.

II. SVM AND LSSVM

Support vector machine (SVM) is a highly-competitive learning paradigm originally proposed by Vapnik [21] in the 1990s. It is based on the structural risk minimization (SRM) principle from computational learning theory. The basic idea of SVM is to maximize the margin hyperplane in the feature space. Similar to other supervised learning methods, an underlying theme of the SVM is to learn from data. Suppose that there is an input space, denoted by X, $X \in R^n$, an output space, denoted by Y, and a training dataset $D = \{(x_1, y_1), \cdots, (x_i, y_i), \cdots, (x_N, y_N)\} \subseteq (X \times Y)^N$, N is the size of the training data. The overall assumption for learning is the existence of a hidden function $Y=f(X)$, and learning task is to construct a heuristic function $g(X)$, such that $g \to f$ on the prediction of Y. The nature of the output space Y decides the learning type. $Y=\{1, -1\}$ leads to a binary classification problem, $Y=\{1, 2,\ldots, M\}$ leads to a multi-class classification problem, and $Y \subseteq R^n$ leads to a regression problem. Here only binary classification problem is discussed.

As earlier noted, SVM belongs to the type of maximal margin classifier, in which the classification problem can be represented as an optimization problem, as shown in Eq. (1).

$$\begin{cases} \min & \phi(w,b,\xi) = (1/2)\, w^T w + C\sum_{i=1}^{N} \xi_i \\ \text{s.t.} & y_i[\varphi(x_i)\cdot w + b] \geq 1 - \xi_i \\ & \xi_i \geq 0, i = 1,2,..., N \end{cases} \tag{1}$$

where w is the normal vector of the hyperplane, b is the bias that is a scalar, $\varphi(x)$ is a mapping function, ξ_i is a tolerable classification error, C is an upper bound parameter controlling the trade-off between margin maximization and tolerable classification errors. When C is large, the error term will be emphasized. Small C means that the large classification margin is encouraged.

Vapnik [21] showed how training a SVM for classification leads to a quadratic programming (QP) problem with bound constraints and linear equality constraints, as shown in Eq. (2).

$$\begin{cases} \max & J(\alpha) = \sum_{i=1}^{N} \alpha_i - (1/2)\sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \varphi(x_i)^T \varphi(x_j) \\ & \quad = \sum_{i=1}^{N} \alpha_i - (1/2)\sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \\ \text{s.t.} & \sum_{i=1}^{N} \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C, i = 1, 2, ..., N \end{cases} \tag{2}$$

where α_i is the so-called Lagrange multipliers, which can be obtained in Eq (2), $K(x_i, x_j)$ is defined as a kernel function with $K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$. The elegance of using the kernel function is that one can deal with feature spaces of arbitrary dimensionality without having to compute the map function $\varphi(x)$ explicitly. Any function that satisfies Mercer's condition [21] can be used as the kernel function. Typical examples of the kernel function are the polynomial kernel $K_{poly}(x_i, x_j) = (x_i^T x_j + 1)^d$, radial basis function (RBF) kernel: $K_{rbf}(x_i, x_j) = \exp(-\left\|x_i - x_j\right\|_2^2 / \sigma^2)$, and sigmoid kernel $K_{sig}(x_i, x_j) = \tanh(\rho x_i^T x_j + \theta)$, where d, σ, ρ, and θ are kernel parameters. From the implementation point of view, training SVM is actually equivalent to solve the linearly constrained QP problem. For this purpose, some traditional QP algorithms such as interior point algorithms [39] and active set method [40] can be utilized. However, for some large-scale

problems, these traditional algorithms are not suitable due to the following reasons. First, these algorithms require that the kernel matrix be computed and stored in memory which may need extremely large memory for the large size problems. Second, these algorithms involve expensive matrix operations such as the Cholesky decomposition of a large submatrix of the kernel matrix. Third, for practitioners who would like to develop their own implementation of an SVM classifier, coding these algorithms is very difficult [41].

Some attempts have been made to develop methods that overcome some or all of these problems. Vapnik [42] presented the chunking algorithm to enhance the computational efficiency by removing some zero vectors. But this algorithm is only suitable for the case that the number of support vectors is small. If the number of support vectors itself is large, the chunking algorithm is inappropriate [41].

Recently Platt suggested a sequential minimal optimization (SMO) algorithm [43] to speedup the training of SVM. Generally the SMO is a carefully organized algorithm which has excellent computational efficiency. However, due to its way of computing and using a single threshold value, it can get into a confused end state and can also become inefficient [41].

To overcome the above problems, a new technique, least squares SVM (LSSVM) proposed by Suykens and Vandewalle [26] is introduced. In LSSVM, the classification problem can be formulated as:

$$\begin{cases} \min & \phi(w,b,\xi) = (1/2)\, w^T w + C\sum_{i=1}^{N} \xi_i^2 \\ \text{s.t.} & y_i[\varphi(x_i)\cdot w + b] = 1 - \xi_i, i = 1,2,..., N \end{cases} \tag{3}$$

In order to solve the above optimization problem in Eq. (3), a Lagrangian function can be constructed below:

$$\zeta(w,b,\xi_i;\alpha_i) = (1/2)\, w^T w + c\sum_{i=1}^{N} \xi_i^2 - \sum_{i=1}^{N} \alpha_i[y_i(w^T \varphi(x_i) + b) - 1 + \xi_i] \tag{4}$$

where α_i are Lagrangian multipliers. Differentiating (4) with w, b, ξ_i, and α_i, we can obtain

$$\begin{cases} \dfrac{d}{dw}\zeta(w,b,\xi_i;\alpha_i) = w - \sum_{i=1}^{N}\alpha_i y_i \varphi(x_i) = 0 \\ \dfrac{d}{db}\zeta(w,b,\xi_i;\alpha_i) = -\sum_{i=1}^{N}\alpha_i y_i = 0 \\ \dfrac{d}{d\xi_i}\zeta(w,b,\xi_i;\alpha_i) = 2c\xi_i - \alpha_i = 0, i = 1,\cdots,N \\ \dfrac{d}{d\alpha_i}\zeta(w,b,\xi_i;\alpha_i) = y_i(w^T\varphi(x_i)+b) - 1 + \xi_i = 0, i = 1,\cdots,N \end{cases} \tag{5}$$

By simple substitutions, we get the following linear equations.

$$\begin{cases} \sum_{i=1}^{N}\alpha_i y_i = 0 \\ \sum_{i=1}^{N}\alpha_i y_i y_j \varphi(x_i)\varphi(x_j) + \dfrac{1}{2c}\alpha_i + b y_i = 1, j = 1,\ldots,N \end{cases} \tag{6}$$

Using the above (N+1) linear equations, we can obtain the solution of LSSVM. Compared with the standard SVM [22], the solution of LSSVM can be achieved from a set of linear equations instead of a QP problem although there are more efficient SVMs than the standard one. In some real-world experiments, the LSSVM has been found with excellent generalization performance and low computational cost [26-27]

It is worth noting that the generalization ability of SVM and LSSVM is controlled by kernel types, kernel parameters and upper bound C [37-38]. As previously mentioned, three typical kernels, polynomial, RBF and sigmoid kernel, are often used. However, every kernel has its advantages and disadvantages and it is therefore hard to say which one is the best in all problems. Recent studies [37-38] found that the mixture or hybridization of these kernel functions can improve the generalization performance of SVM. In this study, we also use a mixed kernel to train LSSVM, which is different from Suykens' LSSVM [27-28]. The mixed kernel is a convex combination of the above three kernels. The convex combination kernel can be written by

$$\begin{cases} K = \lambda_1 K_{poly} + \lambda_2 K_{rbf} + \lambda_3 K_{sig} \\ \text{where} \quad \lambda_1 + \lambda_2 + \lambda_3 = 1 \\ \qquad\quad 0 \le \lambda_1, \lambda_2, \lambda_3 \le 1 \end{cases} \tag{7}$$

According to the previous kernel description of SVM, kernel function must satisfy Mercer's Theorem [21]. In the proposed mixed kernel, since K_{poly} and K_{rbf}, satisfy Mercer's Theorem, a convex combination of them also satisfy Mercer's Theorem. Although sigmoid kernel K_{sig} does not always satisfy Mercer's condition, it is provided since people have historically found it useful. Furthermore, practical applications have proved that it is suitable for a kernel. Thus K_{sig} is added into our proposed mixed kernel for LSSVM modeling.

III. EVOLVING LSSVM LEARNING PARADIGM

In this section, the proposed evolving LSSVM learning paradigm is described in detail. First of all, a general framework of the evolving LSSVM learning paradigm is presented. Then each component of the proposed evolving LSSVM learning paradigm is described in detail.

A. General Framework of Evolving LSSVM Learning Paradigm

A great number of empirical experiments demonstrated that the LSSVM is an efficient learning algorithm for regression and classification problems [26-27]. However, there are still two main problems for LSSVM applications. On the one hand, when the input space dimensions, i.e., input features, are rather large, the interpretability of the LSSVM-based predictive model will be poor. It is therefore necessary for LSSVM to preprocess the input features. On the other hand, LSSVM generalization ability is often controlled by kernel type, kernel parameters and upper bound parameter. Although this study uses a mixed kernel function to overcome the influence of kernel types, the choice of many parameters, such as convex combination coefficients (λ_1, λ_2, and λ_3),

kernel parameters (d, σ, ρ, and θ) and upper bound parameter C, depends in many aspects on the art of the researchers. For these two problems, evolutionary algorithm (EA) is used. Particularly, for the first problem, a standard genetic algorithm (GA) is used for input feature selection to increase the model interpretability and avoid "curse of dimension". For the second problem, another GA is used to optimize the parameters of LSSVM to improve the generalization ability. Based upon the two evolutionary procedures, the evolving LSSVM learning paradigm is formulated, which is illustrated in Fig. I.

<Insert Fig. I Here>

As can be seen from Fig. I, it is easy to find that the evolving LSSVM learning paradigm consists of two main components responding to the above two main problems. In the first component, GA searches the exponential space of feature variable subsets and passes one subset of features to a LSSVM model. The LSSVM extracts predictive information from each subset and learns the patterns. Once a LSSVM learns the data patterns, the trained LSSVM is evaluated on a hold-out dataset not used for training, and returns the evaluation criteria as a fitness function to the GA. In terms of the fitness values, the GA biases its search direction to optimize the evaluation objective. This routine continues for a fixed number of generations. Among all the evolved models over the generations, we select the best feature subset in terms of fitness function values. It is worth noting that in the training and evaluation procedures, the LSSVM only uses the selected feature variables. In the second component, GA is used to optimize the eight undetermined parameters, λ_1, λ_2, λ_3, d, σ, ρ, θ, and C, as listed above. In this component, the eight undetermined parameters are first defined as a chromosome. The fitness of each chromosome is determined by the GA fitness function, i.e., predictive performance of LSSVM using the chromosome as its parameters. The chromosomes are processed by several evolution procedures, i.e., crossover, mutation and selection, to produce the optimal solution. The detailed contents of

every component are described in the following subsections. Through the above two evolutionary procedures, an evolving LSSVM learning paradigm with best feature subset and optimal parameters are produced for generalization purpose.

It is worth noting that the two evolution components can be combined. That is, the feature evolution and parameter evolution can be optimized simultaneously. But in this study two main goals, model interpretability and model accuracy, should be tested. Therefore, we use two individual evolution components for this purpose. In addition, the sequence of the two evolution components can be exchangeable from the theoretic viewpoint. That is, in the evolving LSSVM learning paradigm, the parameter optimization procedure can be performed before the feature selection procedure. But in practice it is unsuitable for large input feature dimension because it will lead to too much computational workload. In this sense, performing feature selection before parameter evolution is more rational.

B. GA-based Input Features Evolution

For many practical problems, the possible input variables may be quite large. Among these input variables, there may be some redundancy. Furthermore, too many input features may lead to "curse of dimension". In addition, a large number of input variables will increase the size of LSSVM and thus require more training data and longer training times in order to obtain a reasonable generalization ability [44]. Therefore input feature reduction should be done with feature selection procedure. Generally, feature selection is defined as the process of selecting a subset of the original features by eliminating redundant features or some features with little information [25]. In the proposed evolving LSSVM learning paradigm, the first task is to select important features for LSSVM learning. The main aims of feature selection reflect two-fold. The first is to discard some unimportant features with less information and thus reducing the input

feature dimensions and improving the model prediction performance. The second is to identify some key features of affecting model performance and therefore reducing the model complexity. In this study, we use standard GA to extract input feature subset for LSSVM modeling.

To date, GA, the most popular type of evolutionary algorithm (EA), has become an important stochastic optimization method as they often succeed in finding the best optimum in contrast to most common optimization algorithms. GA imitates the natural selection process in biological evolution with selection, mating reproduction (crossover) and mutation, and the sequence of the different operations of a genetic algorithm is shown in the left part of Fig. II. The objects to be optimized are represented by a chromosome whereby each object is encoded in a binary string called a gene. Thus, a chromosome consists of as many genes as parameters to be optimized [45]. Interested readers can refer to Goldberg [35] for more details about GA. In the following the GA-based feature variable selection is discussed in detail.

<Insert Fig. II Here>

First of all, a population, which consists of a given number of chromosomes, is initially created by randomly assigning "1" and "0" to all genes. In the case of variable selection, a gene contains only a single bit string for the presence and absence of a variable. The top right part of Fig. II shows a population of four chromosomes for a three-variable selection problem. In this study, the initial population of the GA is randomly generated except of one chromosome, which was set to use all variables. The binary string of the chromosomes has the same size as variables to select from whereby the presence of a variable is coded as "1" and the absence of a variable as "0". Consequently, the binary string of a gene consists of only one single bit [45].

The subsequent work is to evaluate the chromosomes generated by previous operation by a so-called fitness function, while the design of the fitness function is a crucial point in GA, which determines what a GA should optimize. In order to guide a GA toward more highly fit regions of

the search space of chromosomes, one or more fitness values must be assigned to each string. Usually, a standard GA is designed for optimization problems with only one objective. Each fitness value can be determined by a fitness function that we want to optimize. In this work, the fitness value for a string is determined by a LSSVM model. Because our goal is to find a small subset of input variables from many candidate variables, the evaluation of the fitness starts with the encoding of the chromosomes into LSSVM model whereby "1" indicates that a specific variable is selected and "0" that a variable is not selected by the LSSVM model. Note that the LSSVM is used here for modeling the relationship between the input variables and the response variable. Then the LSSVM models are trained with a training data set and after that, the trained LSSVM is tested on a hold-out dataset, and the proposed model is evaluated both on the model generalization performance (e.g., hit ratio, which is maximized) and the modeling complexity (i.e., number of selected feature variables, which is minimized) of the solution. Finally, these two evaluation criteria are combined into one so-called fitness function f and used to evaluate the quality of each string. For a classification problem, for example, our fitness function for the GA variable selection can use the following form:

$$f = E_{accuracy} - \alpha E_{complexity} \tag{8}$$

where $E_{accuracy}$ is the classification accuracy or hit ratio, representing the model predictive power, $E_{complexity}$ is the model complexity, α is a parameter. Parameter α is aimed at adjusting the number of variables used by the evolved LSSVM in terms of users' preference. Usually, a high value of the fitness function results in only few variables selected for each LSSVM model whereas a small value of fitness function results in more variables being selected. We expect that lower complexity will lead to easier interpretability of solutions as well as better generalization.

From Eq. (8), it is not hard to find that two different goals, model accuracy and model

complexity, are combined into one fitness value for candidate solutions. Usually, model accuracy of classification problems can be represented by hit ratio with the following form:

$$E_{accuracy} = \frac{\text{The number of correct classification}}{\text{The number of all evaluation sample}} \tag{9}$$

In Eq. (8), the aim of the first part is to favor feature variable subsets with higher discriminative power to classification problems, while the second part is aimed at finding parsimonious solutions by minimizing the number of selected features as follows.

$$E_{complexity} = \frac{n_v}{N_{tot}} \tag{10}$$

where n_v is the number of variables used by the LSSVM models, N_{tot} is the total number of variables.

After evolving the fitness of the population, the best chromosomes with the highest fitness value are selected by means of the roulette wheel. Thereby, the chromosomes are allocated space on a roulette wheel proportional to their fitness and thus the fittest chromosomes are more likely selected. In the following mating step, offspring chromosomes are created by a crossover technique. A so-called one-point crossover technique is employed, which randomly selects a crossover point within the chromosome. Then two parent chromosomes are interchanged at this point to produce two new offspring. After that, the chromosomes are mutated with a probability of 0.005 per gene by randomly changing genes from “0” to “1” and vice versa. The mutation prevents the GA from converging too quickly in a small area of the search space. Finally, the final generation will be judged. If yes, then the optimized subsets are selected. If no, then the evaluation and reproduction steps are repeated until a certain number of generations, a defined fitness or a convergence criterion of the population are reached. In the ideal case, all chromosomes of the last generation have the same genes representing the optimal solution [45].

C. GA-based Parameters Evolution

As earlier noted, the LSSVM generalization ability is often affected by kernel types, kernel parameters and upper bound parameters. To reduce the effect of kernel types, we use a mixed kernel in the LSSVM model, as shown in Eq. (7). As a result it brings three additional coefficients, λ_1, λ_2, and λ_3. These three coefficients along with kernel parameters (d, σ, ρ, θ) and upper bound parameter C constitute the objects of evolution because the choice of these parameters depends heavily on the experience of researchers. Therefore a general representation of parameters using in a LSSVM training process are describe as a vector as follows:

$$\Theta = (\lambda_1, \lambda_2, \lambda_3, d, \sigma, \rho, \theta, C) \tag{11}$$

To perform parameter search process using GA, the model generalization performance is used as fitness function. In order to increase the robustness, k-fold cross validation method is used. That is, the average model performance is used as the fitness function. For a regression or classification problem, we define the fitness function f by the following steps.

(1) Randomly split the samples data into $D_{training}$ and $D_{validation}$ using k-fold cross validation technique;

(2) Use $D_{training}$ to train the LSSVM model with the parameters P and obtain a predictor or classifier;

(3) Use the predictor or classifier to predict or classify the samples in $D_{validation}$;

(4) Compute the average prediction or classification performance of the k-fold cross validation, $\bar{E}_{accuracy}$;

(5) The value of the fitness function f is model generalization performance, it is the average value, i.e., $f(\Theta) = \bar{E}_{accuracy}$.

Now the goal is to maximize the fitness function $f(\Theta)$. Together with the previous parameter constraints, we can formulate the following optimization problem:

$$\begin{cases} \max & f(\Theta) = \overline{E}_{accuracy} \\ \text{s.t.} & \lambda_1 + \lambda_2 + \lambda_3 = 1 \\ & 0 \le \lambda_1, \lambda_2, \lambda_3 \le 1 \\ & \lambda_1, \lambda_2, \lambda_3, d, \sigma, \rho, \theta, C \ge 0 \end{cases} \tag{12}$$

where Θ is a vector representing the LSSVM parameters. The GA is used to solve the above optimization problem.

First of all, the parameter vector $\Theta = (\lambda_1, \lambda_2, \lambda_3, d, \sigma, \rho, \theta, C)$ of LSSVM is defined as a chromosome. Then the fitness of each chromosome is determined by the fitness function: average generalization performance of LSSVM using the chromosome as its parameters. The chromosomes are processed by evolution steps, i.e., selection, crossover and mutation, as illustrated in Fig. II, to produce the optimal solution. The detailed evolutionary procedure can be written as following:

(1) Generate the initial chromosome randomly within the initial range;

(2) For each chromosome, use fitness function *f* to calculate the fitness value;

(3) Perform the basic genetic operations: select parents based on their fitness values, produce children from the parents by crossover and mutation. These basic genetic operations are similar to the previous description in Section III.B. Replace the current chromosome with the children to formulate a new chromosome;

(4) Repeat the second and third steps until the stop criteria are met.

(5) Report the best chromosome with the largest fitness value as the optimal solution. This best chromosome corresponds to the optimal parameters by GA process. Return these parameters.

In Steps (1) and (3), the initialized and updated chromosomes must be satisfied the constraints in Eq. (12). Note that the GA uses the following five stop criteria [46]:

(1) When the number of generations reaches the predefined generations;

(2) After running for an amount of time in seconds equal to the pre-specified time;

(3) When the fitness value for the best point in the current chromosome is less than or equal to the predefined fitness limit;

(4) If there is no improvement in the fitness function for a sequence of consecutive generations of length;

(5) If there is no improvement in the fitness function during an interval of time in seconds equal to the pre-specified value.

Through the above evolutionary procedures, the evolving LSSVM learning paradigm with a mixed kernel, best input features and optimal parameters are produced. For illustration and evaluation purpose, the produced evolving LSSVM learning paradigm can be further applied to stock market trend mining problems in the following section.

IV. RESEARCH DATA AND COMPARABLE MODELS

In this section, the research data and their input features are first described. Then the comparable forecasting models are brief reviewed.

A. Research Data and Input Variables Description

In this section, three typical stock indices, S&P 500 index, Dow Jones Industrial Average (DJIA) index, and New York Stock Exchange (NYSE) index are used to test the effectiveness of the proposed evolving LSSVM learning paradigm. The historical data are monthly and are obtained from Wharton Research Data Service (WRDS), provided by Wharton School of the University of Pennsylvania. The entire data set covers the period from January 1926 to December 2005 with a

total of 960 observations. Although the Dow Jones Industrial Average Index started from January 1896, we only use the data covered from January 1926 to December 2005 for consistency purpose. The data sets are divided into two periods: the first period covers January 1926 to December 1989 with 768 observations, while the second period is from January 1990 to December 2005 with 192 observations. The first period, which is assigned to in-sample estimation, is used for network learning, i.e., training set. The second period, which is reserved for out-of-sample evaluation, is used for validation, i.e., testing set. Note that this data split is only for final validation and testing purpose, which is determined by the later experiments. For feature selection and parameter optimization procedure, *k*-fold cross validation split method is used, which will be described later. For space limitation, the original data are not listed in this paper, and detailed data can be obtained from the WRDS.

The main aim of this study is to mine and explore the movement trend of stock index. They are categorized as "1" and "-1" in the research data. "1" represents that the next month's index is higher than this month's index, and "-1" means that the next month's index is lower than this month's index. In this sense, stock market trend mining is actually a classification problem. Therefore we use hit ratio to measure the prediction performance, which is defined as

$$\text{Hit ratio} = \frac{1}{N}\sum_{i=1}^{N} R_i \tag{13}$$

where $R_i = 1$ if $MO_i = AO_i$; $R_i = 0$ otherwise. MO_i is the model output, and AO_i is the actual output, N is the number of the testing examples.

Since we attempt to mine the stock price index movement direction, technical indicators and some factors affecting stock market price index are used as input variables. In this study, we select 20 technical indicators and 6 fundamental macro variables to make up the initial input features, as determined by the review of domain experts and prior studies [2, 23]. The

descriptions of initially selected attributes or features are presented in Table I. Note that the data of technical indicators can be obtained through computation with corresponding computational formula shown in Table I and the data of fundamental macro variables can also be obtained directly from the WRDS.

<Insert Table I Here>

B. Overview of Other Comparable Forecasting Models

In order to evaluate the forecasting ability of the proposed evolving LSSVM learning paradigm, we compare its performance with those of conventional methods, such as statistical and time series model as well as typical nonlinear intelligent models: neural network model, standard SVM model, individual LSSVM model without GA-based input feature selection, as well as individual LSSVM model without GA-based parameter optimization. Typically, we select the auto-regressive integrated moving average (ARIMA) model, linear discriminant analysis (LDA) model, individual back-propagation neural network (BPNN) model and standard SVM model with full feature variables as the benchmarks. For further comparison, individual LSSVM model with polynomial kernel ($LSSVM_{poly}$), individual LSSVM model with RBF kernel ($LSSVM_{rbf}$), individual LSSVM model with sigmoid kernel ($LSSVM_{sig}$), individual LSSVM model with a mixed kernel ($LSSVM_{mix}$), individual LSSVM model with GA-based input feature selection only ($LSSVM_{gafs}$) and individual LSSVM model with GA-based parameter optimization ($LSSVM_{gapo}$) are also conducted. We do not compare our proposed learning paradigm to a standard logit regression model because a logit regression model is a special case of single BPNN model with one hidden node.

For ARIMA models, only the stock index price P is used. In the ARIMA model [47], the future value of a variable is assumed to be a linear function of several past observations and

random errors. That is, the underlying process that generates the time series takes the form:

$$\phi(B)y_t = \theta(B)e_t \tag{14}$$

where y_t and e_t are the actual value and random error at time t respectively; B denotes the backward shift operator, i.e. $By_t = y_{t-1}$, $B^2y_t = y_{t-2}$ and so on; and $\phi(B) = 1 - \phi_1 B - \cdots - \phi_p B^p$, $\theta(B) = 1 - \theta_1 B - \cdots - \theta_q B^q$, where p, q are integers and often referred to as orders of the model. Random errors, e_t, are assumed to be independently and identically distributed with a mean of zero and a constant variance of σ^2, i.e. $e_t \sim IID(0, \sigma^2)$. If the dth difference of $\{y_t\}$ is an ARIMA process of order p and q, then y_t is called an ARIMA (p, d, q) process. Because ARIMA is a time series model, the final stock index price movement direction needs to use the following equation to calculate with the predicted value:

$$z = \begin{cases} +1, & (y_{t+1} - y_t)(\hat{y}_{t+1} - y_t) \geq 0; \\ -1, & \text{otherwise.} \end{cases} \tag{15}$$

where y_t is the actual value in current period t and $\hat{y}_{t+1}$ is the predicted value in the next period.

LDA [2] can handle the case in which the within-class frequencies are unequal and its performance has been examined on randomly generated test data. This method maximizes the ratio of between-class variance to the within-class variance in any particular data set, thereby guaranteeing maximal separability. Usually, a LDA model with d-dimension inputs takes the following form:

$$z(x) = \text{sgn}(\hat{y}) = \text{sgn}(a_0 + \sum_{i=1}^{d} a_i x_i) \tag{16}$$

where a_0 is the intercept, x_i are various factors affecting stock price movement, and a_i are the coefficients of related factors. Each of the ARIMA and LDA models is estimated by in-sample data. The model selection process is then followed by using an empirical evaluation, e.g., RMSE,

which is based on the out-of-sample data.

The BPNN model [4] is widely used and produces successful learning and generalization results in various research areas. Usually, a BPNN can be trained by the historical data. The model parameters (connection weights and node biases) will be adjusted iteratively by a process of minimizing the forecasting errors. For prediction purposes, the final computational form of the BPNN model can be written as

$$z(x) = \mathrm{sgn}(\hat{y}) = \mathrm{sgn}(a_0 + \sum_{j=1}^{q} w_j f(a_j + \sum_{i=1}^{p} w_{ij} x_i) + \xi_t) \tag{17}$$

where a_j (j = 0, 1, 2, …, q) is a bias on the jth unit, w_{ij} (i = 1, 2, …, p; j = 1, 2, …, q) is the connection weight between layers of the model, x_i (i = 1, 2,…, p) are the input variable factors, $f(\bullet)$ is the transfer function of the hidden layer, p is the number of input nodes and q is the number of hidden nodes. In our study, the BPNN has 26 input nodes because 26 input variables are employed. By trial and error, we set the number of training epochs is 500 and heuristically determine the number of hidden nodes using the formula ($2\times node_{in}\pm 1$) where $node_{in}$ represents the number of input nodes. The learning rate is 0.25 and the momentum factor is 0.30. The hidden nodes use sigmoid transfer function and the output node uses the linear transfer function.

For standard SVM model [21] with all input features, the SVM model has also 26 input variables, the radial basis function (RBF) is used as the kernel function of SVM. In standard SVM model with RBF kernel, there are two parameters, i.e., upper bound C and kernel parameter σ, to tune. By trial and error, the kernel parameter σ is 10 and the upper bound C is 70. More details about standard SVM, please refer to Vapnik [21]. SVM and LSSVM training code are the modification of LIBSVM [48] and LS-SVMlab [49].

V. EXPERIMENT RESULTS

In this section, we first show how to determine the key determinants of affecting stock price movement using the LSSVM guided by GA. Then the parameters evolutionary experiments are carried out. Finally, we compare the performance of the proposed evolving LSSVM learning paradigm with some comparable forecasting models.

A. Empirical Analysis of GA-based Input Features Evolution

In order to show the robustness of the GA-based feature selection method and determine which features are the key driver of stock index price movement, we use a *k*-fold cross-validation (CV) estimation procedure to perform eight independent experiments for each stock index in terms of eight different data partitions. In the *k*-fold CV estimation procedure, the training data is divided into *k* non-overlapping groups. We train a LSSVM using the first (*k*-1) groups of training data and test the trained LSSVM on the *k*th group. We repeat this procedure until each of the groups is used as a test set once. We then take the average of the performance measurements over the *k* folds. In this experiment, *k* is set to 5. This is a reasonable compromise considering the computational complexity and modeling robustness. Furthermore, an estimate from 5-fold CV is likely to be more reliable than an estimate from a common practice only using a single testing set. Note that the LSSVM in the experiments uses the mixed kernels. For the S&P 500 index, the parameter vector $\Theta_{S\&P500} = (\lambda_1, \lambda_2, \lambda_3, d, \sigma, \rho, \theta, C) = (0.1,0.6,0.3,1.95,2.42,36.33,72.56,223.55)$, For the DJIA index, $\Theta_{DJIA} = (\lambda_1, \lambda_2, \lambda_3, d, \sigma, \rho, \theta, C) = (0.2,0.5,0.3,1.78,6.23,82.35,99.03,126.21)$, for the NYSE index, $\Theta_{NYSE} = (\lambda_1, \lambda_2, \lambda_3, d, \sigma, \rho, \theta, C) = (0.1,0.6,0.3,1.79,2.98,40.28,78.43,250.52)$.

As previously mentioned, we will perform the eight independent experiments for each stock index. For this purpose different training data partitions are carried out. In this study, we randomly select *i*%(*i* = 20, 30, …, 90) training data to perform the feature selection. Here we

assume that the LSSVM training with less than 20% training data is inadequate. Each testing experiment includes three steps. First of all, we randomly select some input variable using GA and pass them to LSSVM. Second, the dataset is randomly separated two parts, training samples and validation samples, in terms of i%, respectively. Third, we select i% training data to train LSSVM model and then test the trained LSSVM with (100–i)% validation data in terms of 5-fold CV procedure. In addition, the crossover rate of GA is 0.8, the mutation rate is 0.005, the maximum generation is set to 150, and the number of independent GA runs is also set to 150 in each experiment. Accordingly, the selected features using GA-based feature selection procedure are shown in Table II. Note that the selected key features are the best feature sets from all the experiments, i.e., they are selected by eight independent experiments simultaneously through different dataset produced by data partition method.

<Insert Table II Here>

As can be seen from Table II, it is easy to see which features are key determinants of stock price movement. For each data partition, different predictive features are clearly highlighted. We partially attribute this finding to the strong correlation among price movement related features. However, note that one feature, monthly trading volume (20), is selected by all data partitions and all tested three stock indices. This makes considerable sense because the trading volume is significantly related to future stock price movement from this analysis. Therefore the investment decision-makers will add more weight to this feature for future predictions.

With reference to Table II, we also find that the key determinants for different stock indices are different except the monthly trading volume. For the S&P 500 index, One-month T-bill (21), MACD (11) and one-month excess return (19), can be considered to be some key driver of S&P 500 stock price movement because they appear many times in eight experiments. For the Dow Jones Industrial Average index, industrial production (23) and one-month T-bill (21) can be seen

as two important key drivers for Dow Jones Industrial Average index movement. While for the New York Stock Exchange index, the consumer price index (22), government consumption (24) and private consumption (25) are the key drivers for NYSE index movement direction. The reason leading to this difference is that different stock indices include different stock members from different industries. For example, S&P 500 is a composite index covering different industrial stocks, while Dow Jones Industrial Average index is an important industrial index closely related to industrial production. In terms of these key determinants, we can reduce data collection and storage requirements and thus reducing labor and data transmission costs. Generally, the results obtained are consistent with previous studies, such as Campbell et al. [50] and Hiemstra and Jones [51].

Through the analysis of these key determinants, we also find that most key determinants of affecting stock index movement are some fundamental macro factors, such as one-month T-bill (21), consumer price index (22) and industrial production (23). On the contrary, only one indicator, monthly trading volume (20) is chosen as the key driver among the 20 technical indicators, which implies that the stock index movement usually depends on the change of fundamental macro variables. This finding is very meaningful for investors and decision-makers because this finding can tell them that investing stock market should depend mostly on the fundamental analysis rather than technical analysis.

At the same time, the GA-based feature selection procedure reduces data dimensions for different stock indices. For S&P 500 index, most data partitions choose at most six features except one that selects eight features at data partition i = 20% (The main reason may be that too less data are used). On average, the GA-based feature selection method selects six features. This reduces the data dimensionality by (26-6)/26 ≈ 76.92% for S&P 500 index. For the Dow Jones Industrial Average index and New York Stock Exchange index, the GA-based feature selection

procedure chooses five features on the average. The amount of data reduction reaches (26-5)/26 = 80.77%. This indicates that we can reduce data collection, computation and storage costs considerably through the GA-based feature selection procedure.

Although we only select several typical features to predict stock price movement tendency, the prediction performance of the model is still promising, as illustrated in Fig. III. Note that the value in Fig. III is the average prediction performance over the 5-fold cross validation experiments.

<Insert Fig. III Here>

As can be seen form Fig. III, several finding can be observed. First of all, before 70%~80% partition rate, the prediction performance generally improves with the increase of data partition. The main reason is that too less training data (e.g., 20%~30% partition rates) is often insufficient for LSSVM learning. Second, when 90% partition rate is used, the predictions show the worse performance relative to 80% partition rate. The reason is unknown and it is worth exploring further with more experiments. Third, in the three testing cases, the performance of the DJIA index is slightly better than the other two indices. The possible reason is that the DJIA index has a relatively smaller volatility than other two indices and real reason is worth further exploring in the future. In summary, the evolving LSSVM learning paradigm with GA-based feature selection is rather robust. The hit ratios of almost all data partition are above 75% except that the data partition is set to 20% for NYSE index (74.69%). Furthermore, the variance of five-fold cross validation experiments is small at 1%~8%. These findings also imply that the feature evolved LSSVM model can effectively predict stock market movement direction.

B. Empirical Analysis of GA-based Parameter Optimization for LSSVM

As is known to all, different parameters selection often results in different model performance for

different practical applications. For the LSSVM model, the generalization performance is often affected by kernel parameters and upper bound parameter, as earlier noted. In this study, the utilization of the mixed kernel also brings more uncertain parameters. Therefore the LSSVM model used in this study has eight parameters, which can be represented as a vector, i.e., $\Theta = (\lambda_1, \lambda_2, \lambda_3, d, \sigma, \rho, \theta, C)$. In order to search their optimal combination, a series of data experiments are conducted. For saving the computational cost, we only use an LSSVM model for each stock index. According to the previous subsection, the LSSVM with 70% partition rate and five typical features (8, 11, 20, 21, and 26) is used for S&P 500 index modeling. For Dow Jones Industrial Average index, we use the LSSVM with 80% partition rate and four key features (1, 19, 20, and 23). While for New York Stock Exchange index, the LSSVM with 80% partition rate and five important features (11, 16, 20, 22, and 25) is utilized. In addition, the crossover rate of GA is 0.8, the mutation rate of GA is 0.005 and the maximum generation of GA is 150. Here the generation of GA represents the number of independent GA runs. Although we set the number of independent GA runs is 150, all of best solutions can be obtained before the iterations arrive the number of independent GA runs (i.e., maximum generation).

According to the above setting, the evolutionary experiments on different parameters selections are conducted, the corresponding results are reported in Table III. It is worth noting that the parameters reported is the best parameter set from each experiment.

<Insert Table III Here>

As can be seen from Table III, some important and interesting conclusions can be found. First of all, observing the kernel mixed coefficients, it is easy to find that for the all testing cases the proportion of the RBF kernel (i.e., the coefficient of λ_2) is the highest. This demonstrates that a RBF kernel has good ability to increase the generalization capability. Second, when the coefficient of λ_1 arrives at 0.5, the prediction performance is the worst of all the three testing

cases. This implies that the polynomial kernel cannot dominate the kernel for LSSVM learning when complex learning task is assigned. Third, Similar to RBF kernel, the sigmoid kernel can also increase the generalization ability of LSSVM or decrease the error rate of LSSVM generalization through observing the change of λ_3. Fourth, in terms of fitness function value, we can roughly estimate the rational range of kernel parameters. For example, for the parameter d, the range [1.5, 2.5] seems to generate good generalization performance. Fifth, for the upper bound parameter, a value larger than 100 seems to be suitable for practical classification applications. Perhaps such a value may give an appropriate emphasis on misclassification rate.

In addition, an important problem in parameter evolution is computational time complexity. It is well known to all that the GA is a class of stochastic search algorithm and it is a time-consuming algorithm. Although GA-based input feature evolution has reduce the modeling complexity to some extent, but the parameters search is still a time-consuming process. For comparison purpose, two commonly used parameter search methods, the grid search (GS) algorithm [52] and direct search (DS) algorithm [46, 53] are used.

In grid search (GS) algorithm, each parameter in the grid is first defined by the grid range. Then a unit grid size is evaluated by an objective function f. The point with best f value corresponds to the optimal parameter. The grid search may offer some protection against local minima but it is not very efficient. The optimal results depend on the initial grid range. Usually a large grid range can provide more chance to achieve the optimal solution but it takes more computational time. Interested readers can refer to [52] for more details about grid search.

Direct search (DS) algorithm [46, 53] is a simple and straightforward search method and can be applied to many nonlinear optimization problems. Suppose the search space dimension is n, a point $\boldsymbol{p}$ in this space can be denoted by $(z_1, z_2, \ldots, z_n)$, the objective function is f, and pattern $\boldsymbol{v}$ is a collection of vectors that is used to determine which points to search in terms of a current point,

i.e., $\boldsymbol{v} = [v_1, v_2, \ldots, v_{2n}]$, $v_1 = [1, 0, \ldots, 0]$, $v_2 = [0, 1, \ldots, 0]$, …, $v_n = [0, 0, \ldots, 1]$, $v_{n+1} = [-1, 0, \ldots, 0]$, $v_{n+2} = [0, -1, \ldots, 0]$, …, $v_{2n} = [0, 0, \ldots, -1]$, $v_i \in R^n$, $i = 1, 2, \ldots, 2n$. The points set M = $\{m_1, m_2, \ldots, m_{2n}\}$ around current point $\boldsymbol{p}$ to be searched are defined by the mesh which multiple the pattern vector $\boldsymbol{v}$ by a scalar r, called the mesh size. If there is at least one point in the mesh whose objective function value is better than that of the current point $\boldsymbol{p}$, we replace this old point with the new point until the best point is found. For more details, please refer to [46, 53].

In this study, the computational time of GA is compared with grid search algorithm and direct search algorithm. In our experiments, the initial parameter range is determined by Table III. In the grid search method, the unit grid size is 0.5. In the direction search algorithm, the maximum iteration is 100. The parameter setting of GA is similar to the above experiment, as shown before Table III. The program is run on an IBM T60 Notebook with Pentium IV CPU running at 1.66GHz with 512MB RAM. All the programs are implemented with Matlab language. The experiments of three different parameter search methods used the identical training and testing sets with five-fold cross validation. The average classification accuracy of the three methods and computational time are shown in Table IV.

<Insert Table IV Here>

From Table IV, we can find that for different stock indices, the performance obtained from parameter search is similar to some extent. Furthermore, there is no significant difference among the three parameter search methods for the average prediction performance according to two-tail *t*-test. However, the computational time of each parameter search algorithm is distinctly different. In the three methods, the CPU time of the grid search algorithm is the longest for the three testing stock indices, followed by the genetic algorithm. The shortest CPU time is direct search methods. Although the computational time of the GA is slightly worse than that of direct search,

we only use a standard GA procedure. Perhaps some improved GA procedures (e.g., replacing binary encoding with decimal encoding, using self-adaptive cross rates and mutation rates) can speedup the computation process, thus making the evolving LSSVM learning paradigm become one of the most promising learning techniques.

In addition, to fully measure the prediction and exploration power of the proposed evolving LSSVM learning paradigm, it is required to further compare with other forecasting models, which is performed in the following subsection.

C. Comparisons with Other Forecasting Models

According to the previous experiment design presented in Section IV, each of the comparable forecasting models described in the previous section is estimated by in-sample data. The model estimation selection process is then followed by an empirical evaluation based on the out-of-sample data. At this stage, the prediction performance of the models is measured by hit ratio. Similar to the prior experiments, five-fold cross validation experiments are conducted and the corresponding results are reported in Table V. Note that the value in bracket is the standard deviation of five-fold cross validation experiments.

<Insert Table V Here>

As can be see from Table V, the following several findings are observed.

First of all, the differences between the different models are very significant. For example, for the S&P 500 testing case, the hit ratio for the ARIMA model is 55.78%, for the LDA model and BPNN model the hit ratios are 61.43% and 67.56%, respectively, and for the standard SVM model it is only 72.61%; while for the proposed evolving LSSVM forecasting model, the hit ratio reaches 82.66%, which is significantly higher than the individual SVM, BPNN and other statistical models, implying that the proposed evolving LSSVM learning has a significant

improvement on SVM model in mining and exploring stock market trend.

Second, by comparing two conventional statistical and time series models (i.e., ARIMA and LDA) with the latter nine intelligent mining models (i.e., BPNN, SVM and seven LSSVM variants), it is clear that the intelligent mining models consistently outperform the conventional statistical models. The main reasons reflect two aspects. On the one hand, the intelligent mining models can easily capture the nonlinear patterns in the stock market because they adopt nonlinear mapping functions, while the two conventional statistical models are all linear models. On the other hand, in the intelligent mining models, more factors may be included. In our study, ARIMA only use historical time series. That is, they only use one stock price factor and other technical indicators and fundamental macro variables are not considered.

Third, in the two conventional models, we can find that the performance of the LDA model is significantly better than that of the ARIMA model. The possible reason is that the LDA uses more factors and increasing the prediction performance. In the BPNN and standard SVM models, the SVM is better than the BPNN. The main reason is that the SVM model can overcome some shortcomings of BPNN, such as overfitting and local minima and thus increasing the generalization performance.

Fourth, among the four LSSVM with different kernel functions, the LSSVM with mixed kernel function shows its predictive capability relative to the other three single kernel functions. The main reason is that the mixed kernel absorbs the advantages and overcome some disadvantages of each single kernel function because each single kernel function has their own advantages and disadvantages.

Fifth, in the last three evolved LSSVM models, the difference among them is insignificant. The main reason is that the input features are evolved and model parameters are optimized. However, these evolving LSSVM models consistently perform better than LSSVM models

without evolution. Particularly, GA-based input feature selection procedure greatly reduces the model input variable and thus increasing model interpretability and effectiveness. This is also the main reason why the evolved LSSVM models outperform the LSSVM without evolution.

Finally, the proposed evolving LSSVM learning paradigm has some comparative advantages relative to individual SVM and BPNN. First of all, the evolving LSSVM can overcome some shortcomings of BPNN, such as overfitting and local minima. Second, the evolving LSSVM used the mixed kernel and thus the LSSVM has stronger generalization ability. Third, the parameter optimization procedure via GA can also increase the generalization performance of the evolving LSSVM. Fourth, the feature evolution in the evolving LSSVM can easily find some key drivers of affecting model performance, and thus increasing the interpretability of LSSVM. Furthermore, the evolving LSSVM model with important input features has better generalization performance than individual SVM model.

D. Further Discussions

The above subsection in this section verified the effectiveness of the proposed evolving LSSVM learning paradigm. Through hit ratios and their standard deviations, we can judge which model is the best and which model is the worst. However, it is unclear what exactly the differences between good learning models and bad ones are. For this purpose, we conducted McNemar's test [54] to examine whether the proposed evolving LSSVM learning paradigm significantly outperforms the other ten models listed in this study. As a non-parametric test for two related samples, it is particularly useful for before-after measurement of the same subjects [55]. Taking the second dataset as an example, Table VI shows the results of the McNemar's test for Done Jones Industrial Average Index to statistically compare the performance in respect of testing data among the ten models. For space consideration, the results on McNemar's test for

other two practical datasets are omitted here. Actually, we can obtain some similar conclusions from the second and third datasets via McNemar's test. Note that the results listed in Table VI are the Chi squared values and *p* values are in brackets.

<Insert Table VI Here>

As shown in Table VI, we can draw the following conclusions:

(1) The proposed evolving LSSVM learning paradigm outperforms the LSSVM with RBF kernel, sigmoidal kernel, polynomial kernel and four single models at 1% statistical significance level. At the same time, the proposed evolving LSSVM learning paradigm also performs better than the LSSVM with a mixed kernel at 10% significant level. However, the proposed evolving LSSVM learning model does not significantly outperform the individual LSSVM model with GA-based input feature selection only and individual LSSVM model with GA-based parameter optimization only, which imply the effects of the evolution on the performance improvement.

(2) For the $\text{LSSVM}_{\text{gapo}}$ and $\text{LSSVM}_{\text{gafs}}$ models, we can find that these two models can significantly outperform almost all the individual models (i.e., individual $\text{LSSVM}_{\text{rbf}}$, $\text{LSSVM}_{\text{sig}}$, $\text{LSSVM}_{\text{poly}}$, SVM, BPNN, LDA, and ARIMA models) at 1% significance level except of single LSSVM with a kernel function. This further indicates that the impacts of evolution on model performance are significant.

(3) Similarly, LSSVM with a mixed kernel can significantly outperform the LSSVMpoly, standard SVM, single BPNN, LDA and ARIMA, which implies the effects of mix kernel on model performance improvement. But it can not conclude that the LSSVMmix can produce more prediction results than the $\text{LSSVM}_{\text{rbf}}$ and $\text{LSSVM}_{\text{sig}}$. The main reason is still unknown and it is worth exploring further.

(4) For the $\text{LSSVM}_{\text{rbf}}$, $\text{LSSVM}_{\text{sig}}$, $\text{LSSVM}_{\text{poly}}$ and SVM models, it can outperform the individual BPNN, LDA and ARIMA models at 1% or 10% significance level However, the

RBFN ensemble model does not outperform the BPNN ensemble model and the individual SVMR model at 10% significance level. Similarly, the BPNN ensemble model leads to a similar finding. All findings are consistent with results reported in Table V. For the first and third datasets, we can draw some similar conclusions when applying to the above procedures.

VI. CONCLUSIONS

In this study, an evolving LSSVM model integrating LSSVM and evolutionary algorithms (EAs) is proposed to predict stock market tendency. In the proposed evolving LSSVM learning paradigm, a standard genetic algorithm (GA) is first used to select possible input feature combination and optimize parameters of LSSVM and then the evolved LSSVM is used to predict the stock market trend. There are two distinct strengths for the evolving LSSVM model: one is its ability to build an interpretable prediction model because smaller numbers of features are used. The other is its ability to build an optimal prediction model because all model parameters are optimized. Particularly, through a series of simulation experiments, we show that the proposed evolving LSSVM model not only maximizes the generalization performance but also selects a most parsimonious model. These indicate that the proposed evolving LSSVM model can be used as a viable alternative solution to stock market trend mining and exploration.

It is worth noting, however, that the proposed evolving LSSVM learning paradigm could be further improved in the future, such as in areas of ensemble learning and ensemble evolution with LSSVM. Furthermore, this proposed method can not only work in the stock market mining but also work in general classification/regression problems. Future studies will look into these important issues.

ACKNOWLEDGEMENTS

The authors would like to thank the editors and the anonymous referees for their valuable comments and suggestions. Their comments helped to improve the quality of the paper immensely. This work describe here is partially supported by the grants from the National Natural Science Foundation of China (NSFC No. 70601029, 70221001), the Knowledge Innovation Program of the Chinese Academy of Sciences and the NSFC/RGC Joint Research Scheme (No. N_CityU110/07).

REFERENCES

[1] A.S. Chen and M.T. Leung, "Regression neural network for error correction in foreign exchange forecasting and trading", *Computers & Operations Research*, vol. 31, no. 7, pp. 1049-1068, 2004.

[2] W. Huang, Y. Nakamori and S.Y. Wang, "Forecasting stock market movement direction with support vector machine", *Computers & Operations Research*, vol. 32, no.10, pp. 2513-2522, 2005.

[3] G. Grudnitski and L. Osburn, "Forecasting S&P and gold futures prices: an application of neural networks", *Journal of Futures Market*, vol. 13, pp. 631-643, 1993.

[4] K. Hornik, M. Stinchocombe and H.White, "Multilayer feedforward networks are universal approximators", *Neural Networks*, vol. 2, pp. 359-366, 1989.

[5] W. Huang, K.K. Lai, Y. Nakamori, S.Y. Wang and L. Yu, "Neural networks in finance and economics forecasting", *International Journal of Information Technology and Decision Making*, vol. 6, no. 1, pp. 113-140, 2007.

[6] Y. Yoon and G. Swales, “Predicting stock price performance: A neural network approach”, *Proceedings of the 24th Annual Hawaii International Conference on System Sciences*, Hawaii, pp. 156-162, 1991.

[7] R.R. Trippi and D. DeSieno, “Trading equity index futures with a neural network”, *Journal of Portfolio Management*, vol. 19, pp. 309-317, 1992.

[8] S.W. Yu, “Forecasting and arbitrage of the Nikkei stock index futures: An application of back-propagation networks”, *Asia-Pacific Financial Markets*, vol. 6, pp. 341–354, 1999.

[9] A. Chen, M. Leung and H. Daouk, “Application of neural networks to an emerging financial market: Forecasting and trading the Taiwan stock index”, *Computers & Operations Research*, vol. 30, no. 6, pp. 901-923, 2003.

[10] R. Sitte and J. Sitte, “Analysis of the predictive ability of time delay neural networks applied to the S&P 500 time series”, *IEEE Transaction on Systems, Man and Cybernetics – Part C: Applications and Reviews*, vol. 30, no. 4, 568-572, 2000.

[11] A. Kanas, “Neural network linear forecasts for stock returns”, *International Journal of Finance and Economics*, vol. 6, pp. 245–254, 2001.

[12] L. Yu, S.Y. Wang and K.K. Lai, “A novel adaptive learning algorithm for stock market prediction”, *Lecture Notes in Computer Science*, vol. 3827, pp. 443 – 452, 2005.

[13] P. Wasserman, *Advanced Methods in Neural Computing*. Van Nostrand Reinhold, New York, 1993.

[14] S.H. Kim and S.H. Chun, “Graded forecasting using an array of bipolar predictions: application of probabilistic neural networks to a stock market index”, *International Journal of Forecasting*, vol. 14, pp. 323–337, 1998.

[15] M.T. Leung, H. Daouk and A.S. Chen, “Forecasting stock indices a comparison of classification and level estimation models”, *International Journal of Forecasting*, vol. 16, pp.

173–190, 2000.

[16] K. Kamijo and T. Tanigawa, “Stock price pattern recognition: A recurrent neural network approach”, *Proceedings of the International Joint Conference on Neural Networks*, San Diego, pp. 215-221, 1990.

[17] K. Kohara, T. Ishikawa, Y. Fukuhara and Y. Nakamura, “Stock price prediction using prior knowledge and neural networks”, *International Journal of Intelligent Systems in Accounting, Finance and Management*, vol. 6, pp. 11–22, 1997.

[18] R. Tsaih, Y. Hsu and C.C. Lai, “Forecasting S&P 500 index futures with a hybrid AI system”, *Decision Support Systems*, vol. 23, pp. 161-174, 1998.

[19] K. Kim and I. Han, “Genetic algorithm approach to feature discretization in artificial neural networks for the prediction of stock price index”, *Expert Systems with Applications*, vol. 19, pp. 125-132, 2000.

[20] T.H. Hann and E. Steurer, “Much ado about nothing? Exchange rates forecasting: neural networks vs. linear models using monthly and weekly data”, *Neurocomputing*, vol. 10, pp. 323-339, 1996.

[21] V.N. Vapnik, *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

[22] N. Cristianini, “Support vector and kernel machines”, *Proceedings of the 18th International Conference on Machine Learning (ICML 2001)*, MA, USA, June 28-July 1, 2001.

[23] K.J. Kim, “Financial time series forecasting using support vector machines”, *Neurocomputing,* vol. 55, pp. 307-319, 2003.

[24] L. Yu, S.Y. Wang and K.K. Lai, “Mining stock market tendency using GA-based support vector machines”, *Lecture Notes in Computer Science*, vol. 3828, pp. 336-345, 2005.

[25] Y. Kim and W.N. Street, “An intelligent system for customer targeting: a data mining approach”, *Decision Support Systems*, vol. 37, pp. 215-228, 2004.

[26] J.A.K. Suykens and J. Vandewalle, “Least squares support vector machine classifiers”, *Neural Processing Letters*, vol. 9, pp. 293-300, 1999.

[27] J.A.K. Suykens, T.V., Gestel, J.D., Brabanter, B.D., Moor and J. Vandewalle, *Least Squares Support Vector Machines*. World Scientific, Singapore, 2002.

[28] K.K. Lai, L. Yu, L.G. Zhou and S.Y. Wang, “Credit risk evaluation with least square support vector machine”, *Lecture Notes in Artificial Intelligence*, vol. 4062, pp. 490-495, 2006.

[29] A. Brabazon and M. O’Neill, *Biologically Inspired Algorithms for Financial Modelling*, Springer, Berlin, 2006

[30] A.L. Garcia-Almanza and E.P.K. Tsang, “Repository method to suit different investment strategies”, *Proceedings of IEEE Congress on Evolutionary Computation (CEC 2007)*, pp. 790-797, 2007.

[31] A.L. Garcia-Almanza, E.P.K. Tsang and E. Galvan-Lopez, “Evolving decision rules to discover patterns in financial data sets”, in E. Kontoghiorghes, B. Rustem, and P. Winker (eds.), *Computational Methods in Financial Engineering*, Springer, Heidelberg, pp. 239-255, 2008.

[32] A.L. Garcia-Almanza, E.P.K. Tsang, “Evolving decision rules to predict investment opportunities”, *International Journal of Automation and Computing*, vol. 5, no. 1, pp. 22-31, 2008.

[33] E.P.K. Tsang and J. Li, “EDDIE for financial forecasting”, in S.H. Chen (ed.), *Genetic Algorithms and Programming in Computational Finance*, Kluwer Series in Computational Finance, pp. 161-174, 2002.

[34] E.P.K. Tsang, S. Markose and H. Er, “Chance discovery in stock index option and future arbitrage”, *New Mathematics and Natural Computation*, vol.1, no.3, pp. 435-447, 2005.

[35] D.E. Goldberg, *Genetic Algorithm in Search, Optimization, and Machine Learning*.

Addison-Wesley, Reading, MA, 1989.

[36] M. Kudo and J. Sklansky, “Comparison of algorithms that select features for pattern classifiers”, *Pattern Recognition*, vol. 33, pp. 25-41, 2000.

[37] G.F., Smits and E.M. Jordaan, “Improved SVM regression using mixtures of kernels”, *Proceedings of the 2002 International Joint Conference on Neural Networks*, vol. 3, pp. 2785-2790, 2002.

[38] A.T. Quang, Q.L. Zhang and X. Li, “Evolving support vector machine parameters”, *Proceedings of the First International Conference on Machine Learning and Cybernetics*, pp. 548-551, 2002.

[39] L. Kaufman, “Solving the quadratic programming problem arising in support vector classification”, in B. Scholkopf, C. Burges and A. Smola (eds.), *Advances in Kernel Methods: Support Vector Machines*, MIT Press, Cambridge, MA, 1998.

[40] A.J. Smola and B. Scholkopf, “A tutorial on support vector regression”, *NeuroCOLT Technical Report TR-1998-030*, Royal Holloway College, London, UK, 1998.

[41] S. Keerthi, S. Shevade, C. Bhattacharyya, and K. Murthy, “Improvements to Platt’s SMO algorithm for SVM classifier design”, *Neural Computation*, vol. 13, no. 3, pp. 637-649, 2001.

[42] V. Vapnik, *Estimation of Dependences based on Empirical Data*. Springer-Verlag, Berlin, 1982.

[43] J. Platt, “Fast training of support vector machines using sequential minimal optimization”, in: B. Scholkopf, C. Burges and A. Smola (eds.), *Advances in Kernel Methods: Support Vector Machines,* MIT Press, Cambridge, MA, 1998.

[44] X. Yao, “Evolving artificial neural networks”, *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423-1447, 1999.

[45] L. Yu, S.Y. Wang and K.K. Lai, "An Integrated Data Preparation Scheme for Neural Network Data Analysis", *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 2, pp. 217-230, 2006.

[46] The Mathworks, "Genetic Algorithm and Direct Search Toolbox: User's Guide". URL: http://www.mathworks.com/.

[47] G.E.P. Box and G. Jenkins, *Time Series Analysis: Forecasting and Control.* Holden-Day, San Francisco, 1970.

[48] C.C. Chang and C.J. Lin, "LIBSVM: a Library for Support Vector Machines". URL: http://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf.

[49] K. Pelckmans, J.A.K. Suykens, T. Van Gestel, J. De Brabanter, L. Lukas, B. Hamers, B. De Moor and J.Vandewalle, "LS-SVMlab: a MATLAB/C toolbox for Least Squares Support Vector Machines". URL: http://www.esat.kuleuven.ac.be/sista/lssvmlab.

[50] J.Y. Campbell, S.J. Grossman and J. Wang, "Trading volume and serial correlation in stock returns", *Quarterly Journal of Economics*, vol. 107, pp. 905–939, 1993.

[51] C. Hiemstra and J.D. Jones, "Testing for linear and nonlinear Granger causality in the stock price–volume relation", *Journal of Finance*, vol. 49, no. 5, 1639–1664, 1994.

[52] J. Kim, "Iterated Grid Search Algorithm on Unimodal Criteria", *PhD Thesis*, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 1997.

[53] R. Hooke and T.A. Jeeves, "Direct search solution of numerical and statistical problems", *Journal of the Association for Computing Machinery*, vol. 8, pp. 212-229, 1961.

[54] Q. McNemar, "Note on the sampling error of differences between correlated proportions and percentages", *Psychometrica*, vol. 12, pp. 153-157, 1947.

[55] D.R. Cooper and C.W. Emory, *Business Research Methods*. Irwin, Chicago, 1995.

BIOGRAPHY

Lean Yu received the Ph.D. degree in Management Sciences and Engineering from the Academy of Mathematics and Systems Science, Chinese Academy of Sciences (CAS). He has published more than 30 papers in journals including IEEE Transactions on Knowledge and Data Engineering, European Journal of Operational Research, International Journal of Intelligent Systems, and Computers & Operations Research. He is currently an associate professor in the Academy of Mathematics and Systems Science of Chinese Academy of Sciences. His research interests include artificial intelligence, computer simulation, decision support systems, knowledge management and financial forecasting.

Huanhuan Chen received the B.Sc. degree from the University of Science and Technology of China (USTC), Hefei, Anhui, China, in 2004. He is currently working toward the PhD degree in School of Computer Science, University of Birmingham, U.K. He was the recipient of Dorothy Hodgkin Postgraduate Award in 2004. His research interests include data mining, machine learning and evolutionary computation.

Shouyang Wang received the Ph.D. degree in Operations Research from Institute of Systems Science, Chinese Academy of Sciences (CAS), Beijing in 1986. He is currently a Bairen distinguished professor of Management Science at Academy of Mathematics and Systems Sciences of CAS and a Lotus chair professor of Hunan University, Changsha. He is the editor-in-chief or a co-editor of 12 journals. He has

published 18 books and over 120 journal papers. His current research interests include financial engineering, e-auctions and decision support systems.

Kin Keung Lai is the Chair Professor of Management Science at City University of Hong Kong, and he is also the Associate Dean of the Faculty of Business. Currently, he is also acting as the Dean of College of Business Administration at Hunan University, China. Prior to his current post, he was a Senior Operational Research Analyst at Cathay Pacific Airways and the Area Manager on Marketing Information Systems at Union Carbide Eastern. Professor Lai received his Ph.D. at Michigan State University, USA. Professor Lai's main research interests include logistics and operations management, computer simulation, AI and business decision modeling.