# An Efficient LS-SVM-Based Method for Fuzzy System Construction

Wanqing Zhao, *Member, IEEE*, Jingjing Zhang, and Kang Li, *Senior Member, IEEE*

***Abstract*—This paper proposes an efficient learning mechanism to build fuzzy rule-based systems through the construction of sparse least-squares support vector machines (LS-SVMs). In addition to the significantly reduced computational complexity in model training, the resultant LS-SVM-based fuzzy system is sparser while offers satisfactory generalization capability over unseen data. It is well known that the LS-SVMs have their computational advantage over conventional SVMs in the model training process; however, the model sparseness is lost, which is the main drawback of LS-SVMs. This is an open problem for the LS-SVMs. To tackle the nonsparseness issue, a new regression alternative to the Lagrangian solution for the LS-SVM is first presented. A novel efficient learning mechanism is then proposed in this paper to extract a sparse set of support vectors for generating fuzzy IF–THEN rules. This novel mechanism works in a stepwise subset selection manner, including a forward expansion phase and a backward exclusion phase in each selection step. The implementation of the algorithm is computationally very efficient due to the introduction of a few key techniques to avoid the matrix inverse operations to accelerate the training process. The computational efficiency is also confirmed by detailed computational complexity analysis. As a result, the proposed approach is not only able to achieve the sparseness of the resultant LS-SVM-based fuzzy systems but significantly reduces the amount of computational effort in model training as well. Three experimental examples are presented to demonstrate the effectiveness and efficiency of the proposed learning mechanism and the sparseness of the obtained LS-SVM-based fuzzy systems, in comparison with other SVM-based learning techniques.**

***Index Terms*—Efficient learning, fuzzy rules, fuzzy systems, least-squares support vector machines (LS-SVMs), sparseness.**

## I. Introduction

FUZZY rule-based systems, with their origins from ancient Greek philosophy and at the leading edge of computational intelligence, have been successfully applied to many areas, such as regression estimation, decision making, and pattern recognition [1]–[4]. The main thrust lies on their excellent learning capability and that the resultant fuzzy IF–THEN rules can provide a linguistic model interpretable to the users. The key stage in constructing fuzzy systems usually involves the rule extraction and the associated parameter learning. It is desirable to find a sparse set of fuzzy rules, which provides a concise interpretable explanation of the behavior of the system under investigation. As a result, a variety of rule extraction methods have been proposed in the literature, including heuristic, adaptive, evolutionary, and statistical learning methods.

Among various rule extraction methods, the grid partition method was proposed to divide the input space into rectangular subspaces based on a uniform partitioning of each input variable into fuzzy sets [5]. To cope with the curse-of-dimensionality issue caused by grid partitioning, various clustering methods were devised for fuzzy rule generation [6]–[8], where the number of fuzzy sets employed for each input variable is equal to the number of fuzzy rules used for the whole fuzzy system. Moreover, rank-revealing methods like SVD-QR and Pivoted QR decomposition [9]–[11] are used to determine the effective rank of the matrix constructed from all the rule premises (i.e., the normalized rule firing strength matrix) according to its singular values. However, these methods only work in the input space; thus, the selected rules may not necessarily be related to the output; therefore, the final model performance may not be as good as expected. Orthogonal least-squares (OLS) is another well-researched method [12], [13], which is also used to perform rule base reduction on both the input and output spaces. It is worth mentioning that the fast recursive algorithm (FRA) developed recently by Li *et al.* [14] is a useful alternative to OLS, which avoids any matrix decomposition during the subset selection process. The gradient descent and evolutionary optimization are also used in fuzzy rule extraction and parameter learning to find better global solutions [15]–[18], but they are still very time-consuming. Recently, the approach to use the support vector machine (SVM) methodologies to extract support vectors (SVs) for generating IF–THEN rules and thus to describe the fuzzy system in terms of kernel functions has attracted a lot of research interest in the rule extraction and hereby constitutes the main topic of this paper.

SVMs [19] are new techniques that aim to solve pattern classification problems, based on the principle of structural risk minimization instead of mean squared-error minimization, thus minimizing the upper bound on the model's generalization error. Based on this, fuzzy rule extraction incorporating SVM or support vector regression (SVR) has attracted a lot of interest [20]–[23]. Chiang and Hao [20] first introduced fuzzy model construction using SVM techniques, where the kernel function in an SVM is related to the fuzzy basis function (FBF) to fuse the two mechanisms into a fuzzy rule-based modeling method.

Manuscript received October 2, 2013; revised January 26, 2014; accepted March 19, 2014. Date of publication May 2, 2014; date of current version May 29, 2015. This work was supported in part by the Engineering and Physical Sciences Research Council (U.K.) under Grant EP/L001063/1 and the National Natural Science Foundation of China under Grant 61271347, Grant 61273040, and Grant 51077022, the Shanghai Rising Star programme 12QA1401100, the Shanghai Science and Technology Committee under Grant 11ZR1413100, and the China Scholarship Council.

W. Zhao is with the School of Engineering, Cardiff University, Cardiff, CF24 3AA, U.K. (e-mail: w.q.zhao@ieee.org).

J. Zhang is with the School of Engineering, Cardiff University, Cardiff, CF24 3AA, U.K. (e-mail: ZhangJ68@cardiff.ac.uk).

K. Li (Corresponding author) is with the School of Electronics, Electrical Engineering and Computer Science, The Queen's University of Belfast, Belfast, BT9 5AH, U.K. (e-mail: k.li@qub.ac.uk).

Digital Object Identifier 10.1109/TFUZZ.2014.2321594

The fuzzy rules are generated using the learning mechanism for extracting SVs, where the number of fuzzy rules is then equal to the number of SVs. To further decrease the number of fuzzy rules, a Takagi–Sugeno (T–S) fuzzy system based on support vector regression (TSFS-SVR) was proposed [23]. In the TSFS-SVR, the number of fuzzy rules was determined by a one-pass clustering algorithm, and a new T–S kernel corresponding to a T–S-type fuzzy rule was constructed from the product of a cluster output and a linear combination of input variables.

However, apart from the fact that a large number of SVs may be generated by the SVM learning mechanism, another issue is the high computational complexity involved in solving a dual quadratic programming (QP) problem, which leads to the development of least-squares SVMs (LS-SVMs). The LS-SVMs were thus proposed by modifying the inequality constraints in the two-norm SVMs, resulting in solving a linear Karush–Kuhn–Tucker (KKT) system rather than solving the QP problem in the traditional SVM. Unfortunately, a major drawback of an LS-SVM model is its nonsparseness [24], where all the training patterns are used as SVs in the final classifier. The complexity of the final classifier after learning from data thus is extremely high. Therefore, despite the computational advantage of LS-SVMs, their nonsparseness issue still restricts the development of LS-SVM-based fuzzy systems as the final rule base can be extremely large where the number of fuzzy rules is equal to the number of training patterns. It is worth noting that a conventional strategy to overcome this drawback is to impose sparseness by pruning [25], where a series of LS-SVMs are continuously trained, and each time, a small fraction (for example, 5%) of the instances in the training dataset with smallest support values are discarded. However, this procedure inevitably increases the computational burden, and the resultant model performance cannot be guaranteed. Two fast sparse approximation schemes (i.e., FSALS-SVM and PFSALS-SVM) were also proposed for training LS-SVMs [26]. They are based on the greedy algorithm with the aid of viewing the Wolfe dual problem of LS-SVMs as a regularized loss function induced by reproducing Kernel–Hilbert space (RKHS). Based on these observations, this paper mainly concerns the sparseness issue as well as the computational demand associated with the development of LS-SVM-based fuzzy systems.

The main contribution of this paper is the proposal of an efficient learning mechanism for the construction of sparse LS-SVM-based fuzzy systems with significantly reduced computational demand. The novel techniques employed are summarized as follows. First, the LS-SVM learning mechanism is employed to provide a framework to extract SVs for generating fuzzy IF–THEN rules and to formulate the fuzzy rule-based system in the form of a series expansion of FBFs. To deal with the nonsparseness issue for a conventional LS-SVM, a new regression solution to the Lagrangian one for solving the LS-SVM is presented. This regression solution is obtained by optimizing the same objective function defined in the LS-SVM and has a better objective value compared with the conventional one. Second, a novel learning mechanism is then proposed to extract a sparse set of SVs for generating fuzzy IF–THEN rules from the training instances. The novel mechanism works in stepwise subset selection manner, where in each step, it includes a forward expansion phase to select the most significant SVs and a backward exclusion phase to reevaluate the least insignificant SVs that are selected previously, and both phases work in a regularized least-squares sense. Finally, a few key techniques are proposed to completely avoid the matrix inverse operations and to accelerate the training process, leading to the proposal of the efficient learning algorithm with low computational complexity. It is also worth mentioning that the second-stage technique [27] used to refine a subset of fixed size has shown to be extremely effective when applied to improve the results produced by stepwise forward subset selection approaches. However, its computational demand is still high, and furthermore, the original second-stage algorithm was used to select a subset of terms of a fixed size. In this paper, the second-stage idea is also implemented in the proposed algorithm to demonstrate that the outstanding performance can be achieved by our method. With all these key technologies, the proposed approach can thus achieve both computation reduction and model sparseness in developing the LS-SVM-based fuzzy systems, and either of the two advantages surpasses the respective strength inherent from the conventional SVMs or LS-SVMs. Three simulation and real-world examples on modeling, prediction, and classification problems are presented, respectively, to demonstrate the efficiency of the novel learning mechanism and the sparseness of the constructed LS-SVM-based fuzzy systems.

This paper is organized as follows. Section II gives a brief description of the fuzzy rule-based systems. The mathematical formulation of the LS-SVMs and the new regression solution are then presented in Section III. Section IV proposes the efficient learning mechanism for the construction of sparse LS-SVM-based fuzzy systems. Results from three applications on nonlinear system modeling, melt pressure prediction in polymer extrusion, and mammographic masses diagnosis are presented in Section V. Finally, Section VI concludes this paper.

## II. Fuzzy Rule-Based Systems

This section describes the mathematical formulation of the fuzzy rule-based systems. As indicated in [10] and [20], the spirit of fuzzy rule-based systems applies the strategy of "divide and conquer," in which by using a number of interpretable fuzzy rules, their premise part is first used to partition the original input space into a set of small fuzzy input regions, and the consequent part is then employed to describe the system behavior within that small fuzzy region via various constituents. Therefore, the most common fuzzy rule-based system consists of a set of linguistic fuzzy rules, the $i$th rule being represented by

$$R_i\text{: IF } x_1(t) = A_{i,1} \text{ AND } x_2(t) = A_{i,2} \text{ AND } \ldots \text{ AND } x_n(t) = A_{i,n}, \text{ THEN } \hat{y}_i(t) = \theta_i, i = 1, \ldots, m \quad (1)$$

where $t$ denotes the sampling instant, $i$ is the rule index with a total of $m$ fuzzy rules, $\mathbf{x}(t) = [x_1(t), \ldots, x_n(t)] \in \Re^n$ is an $n$-dimensional input vector for the system of interest, $A_{i,j}$ is the fuzzy set associated with the $i$th rule corresponding to the input variable $x_j(t)$, $\theta_i$ is the constant constituent for the $i$th rule consequent, and $\hat{y}_i(t)$ is the output variable for the $i$th rule in

the fuzzy system. The Gaussian membership function defined as

$$\mu_{i,j}(x_j(t);c_{i,j};\sigma_{i,j}) = \exp\left\{-\frac{1}{2}\left(\frac{x_j(t)-c_{i,j}}{\sigma_{i,j}}\right)^2\right\} \quad (2)$$

is commonly employed for the fuzzy set $A_{i,j}$ in the input space, where $c_{i,j}$ and $\sigma_{i,j}$ denote, respectively, the center and standard deviation of the $i$th membership function with regard to the $j$th input ($j = 1, \ldots, n$). To infer the fuzzy system output, the T-norm operators are applied to compute the $i$th rule firing strength

$$\mu_i(\mathbf{x}(t);\mathbf{c}_i;\boldsymbol{\sigma}_i) = \prod_{j=1}^{n} \exp\left\{-\frac{1}{2}\left(\frac{x_j(t)-c_{i,j}}{\sigma_{i,j}}\right)^2\right\} \quad (3)$$

where $\mathbf{c}_i = [c_{i,1}, \ldots, c_{i,n}]^{\mathrm{T}} \in \Re^n$ and $\boldsymbol{\sigma}_i = [\sigma_{i,1}, \ldots, \sigma_{i,n}]^{\mathrm{T}} \in \Re^n$. Then, the degree of fulfillment (normalized firing strength) of the $i$th rule is given by

$$N_i(\mathbf{x}(t);\mathbf{W}) = \frac{\mu_i(\mathbf{x}(t);\mathbf{c}_i;\boldsymbol{\sigma}_i)}{\sum_{i=1}^{m}\mu_i(\mathbf{x}(t);\mathbf{c}_i;\boldsymbol{\sigma}_i)} \quad (4)$$

where $\mathbf{W} = [\mathbf{c}_1^{\mathrm{T}}, \boldsymbol{\sigma}_1^{\mathrm{T}}, \ldots, \mathbf{c}_m^{\mathrm{T}}, \boldsymbol{\sigma}_m^{\mathrm{T}}]^{\mathrm{T}}$ denotes the premise parameter vector. The weighted-average-defuzzification method can then be employed to calculate the overall output of the fuzzy rule-based system, such that

$$f(\mathbf{X}(t);\mathbf{W};\boldsymbol{\Theta}) = \sum_{i=1}^{m} N_i(\mathbf{x}(t);\mathbf{W})\,\theta_i \quad (5)$$

where $\boldsymbol{\Theta} = [\theta_1, \ldots, \theta_m]^{\mathrm{T}}$ denotes the consequent parameters vector. Note that $N_i(\mathbf{x}(t);\mathbf{W})$ is also called as the FBF. In this circumstance, the fuzzy rule-based system can be viewed as a series of FBF expansions. This linear combination of FBFs is capable of approximating any continuous nonlinear function on a compact set to arbitrary accuracy, provided that sufficient fuzzy rules are made available.

## III. Least-Squares Support Vector Machine and Its New Regression Solution

SVM [19], [28] is a recently proposed technique that aims to solve pattern classification problems, where it is used to find a hyperplane $\mathbf{h}\cdot\mathbf{x}$ ($\mathbf{h}$ is a vector consisting of the associated unknown parameters) that can separate two-class patterns with the maximum margin. This is because maximizing the two-class margin is equivalent to minimizing the upper bound on the model's generalization error (i.e., structural risk minimization). Due to the high computational complexity generally involved in solving the QP problems in the dual space in SVM, LS-SVM was proposed by modifying the inequality constraints in a conventional two-norm SVM. The LS-SVM takes the form of $\mathbf{h}\cdot\boldsymbol{\phi}(\mathbf{x}(t))$, in which the nonlinear function $\phi(\mathbf{x}(t))$ maps the original input data into some high-dimensional feature space, i.e., $\mathbf{x}(t) \in \Re^n \rightarrow \boldsymbol{\phi}(\mathbf{x}(t)) \in \Re^H$, aiming to cope with the linear unseparated problem. Given a set of training patterns $\{\mathbf{x}(t), y(t)\}_{t=1}^{N} \in \Re^n \times \{\pm 1\}$, the classification problem in an LS-SVM is now defined as

$$\min_{\mathbf{h},\varepsilon(t)} \frac{1}{2}\|\mathbf{h}\|^2 + \frac{1}{2\mu}\sum_{t=1}^{N}\varepsilon(t)^2 \quad \text{subject to} \quad \varepsilon(t) = y(t) - \mathbf{h}\cdot\boldsymbol{\phi}(\mathbf{x}(t)) \quad (6)$$

where $\mu$ is a regularization parameter that determines the bias-variance tradeoff. Its solution can be obtained by introducing the Lagrangian

$$\mathcal{L}(\mathbf{h},\boldsymbol{\varepsilon},\boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{h}\|^2 + \frac{1}{2\mu}\sum_{t=1}^{N}\varepsilon(t)^2 - \sum_{t=1}^{N}\alpha_t\{\mathbf{h}\cdot\boldsymbol{\phi}(\mathbf{x}(t)) + \varepsilon(t) - y(t)\} \quad (7)$$

where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_N) \in \Re^N$ is the vector of Lagrange multipliers. The minimum value with respect to $\mathbf{h}$, $\varepsilon(t)$, and $\alpha_t$ is obtained by solving the following well-known KKT system:

$$\begin{aligned}
&\frac{\partial\mathcal{L}}{\partial\mathbf{h}} = 0 \Rightarrow \mathbf{h} = \sum_{t=1}^{N}\alpha_t\boldsymbol{\phi}(\mathbf{x}(t))\\
&\frac{\partial\mathcal{L}}{\partial\varepsilon(t)} = 0 \Rightarrow \alpha_t = \frac{\varepsilon(t)}{\mu} \qquad \forall t \in \{1,2,\ldots,N\}\\
&\frac{\partial\mathcal{L}}{\partial\alpha_t} = 0 \Rightarrow \mathbf{h}\cdot\boldsymbol{\phi}(\mathbf{x}(t)) + \varepsilon(t) - y(t) = 0 \qquad \forall t \in \{1,2,\ldots,N\}.
\end{aligned} \quad (8)$$

These equations can be rewritten concisely in a matrix form as

$$\mathbf{M}\boldsymbol{\alpha} = \mathbf{y} \quad (9)$$

where $\mathbf{M} = \mathbf{K} + \mu\mathbf{I}$ is a definite symmetric matrix, and $\mathbf{K}_{i,j}(\mathbf{x}(i),\mathbf{x}(j)) = \boldsymbol{\phi}(\mathbf{x}(i))\cdot\boldsymbol{\phi}(\mathbf{x}(j))$ is known as the kernel function. By using (8), the LS-SVM classifier can now be rewritten as

$$f(\mathbf{x}) = \sum_{i=1}^{N}\alpha_i\mathbf{K}(\mathbf{x}(i),\mathbf{x}). \quad (10)$$

It is observed from (9) and (10) that the mapping function $\boldsymbol{\phi}(\cdot)$ involved in solving the KKT system and in producing the final model output does not have to be known exactly. Instead, the value of interest is the kernel function $K_{i,j} = \langle\boldsymbol{\phi}(\mathbf{x}(i))\cdot\boldsymbol{\phi}(\mathbf{x}(j))\rangle$, which is vividly referred to as the well-known kernel trick. The linear KKT system in (9) can now be efficiently solved by using direct methods, such as Cholesky decomposition as $\mathbf{M}$ is positive definite. However, a major drawback of an LS-SVM model lies in its nonsparseness [24]. It can be shown in the second equation of (8) that the values of $\alpha_t$ ($t = 1, \ldots, N$) shall never be zero because $\varepsilon(t)$ ($t = 1, \ldots, N$) are nonzero. All training patterns are supposed to contribute to the final model, the importance of each being indicated by its support value. As a result, the LS-SVM obtained will lose sparseness, and the size of the resultant model can be extremely large. This is perhaps the main reason that limits the development of LS-SVM-based fuzzy systems. In this paper, a sparse

LS-SVM learning mechanism will be proposed and integrated into the compact fuzzy rule extraction.

To deal with the nonsparseness issue in the LS-SVM, a new regression solution to the Lagrangian one to solve the LS-SVM is first given. In the aforementioned conventional solution of the LS-SVM presented in (9) and (10), the kernel trick is adopted to deal with the linear inseparable cases in classification. As a result, the necessity of knowing the exact mapping function used to map the input data into some high-dimensional feature space is no longer required. The authors have recently proposed a method [29] by first assuming that the mapping function $\boldsymbol{\phi}(\mathbf{x}(t))$ is already known and given by

$$\begin{aligned}&\boldsymbol{\phi}(\mathbf{x}(t)) = [\varphi_1(\mathbf{x}(t)), \varphi_2(\mathbf{x}(t)), \ldots, \varphi_m(\mathbf{x}(t))]^{\mathrm{T}}\\ &\text{with}\quad \varphi_i(\mathbf{x}(t)) = \exp\left\{-\frac{1}{2}(\mathbf{x}(t)-\mathbf{s}_i)^{\mathrm{T}}\boldsymbol{\Gamma}_i^{-1}(\mathbf{x}(t)-\mathbf{s}_i)\right\}\\ &\qquad\qquad i = 1, \ldots, m \qquad (11)\end{aligned}$$

where $\mathbf{s}_i \in \Re^n (i = 1, 2, \ldots, m)$ are some data vectors from input space, which can be chosen from the training patterns or otherwise. This way, the original input space $\mathcal{F}^n$ is thus transformed into another high-dimensional feature space $\mathcal{F}^m$.

Accordingly, the primal optimization problem of the LS-SVM defined in (6) can thus be reformulated as

$$\min_{\mathbf{h}} J(\mathbf{h}) = \frac{1}{2}\|\mathbf{h}\|^2 + \frac{1}{2\mu}\sum_{t=1}^{N}(y(t) - \mathbf{h}\cdot\boldsymbol{\phi}(\mathbf{x}(t)))^2. \quad (12)$$

This constitutes a regularized least-squares problem, which is also called ridge regression in statistics. The primal optimization problem in the LS-SVM has thus been successfully transformed into a regularized least-squares one, avoiding the KKT problem described in (8). Considering that the gradient of the cost function (12) with respect to the parameter vector $\mathbf{h}$ has to be zero, the estimated optimal parameter vector is then given by

$$\hat{\mathbf{h}} = (\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \mu\mathbf{I})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{y} \quad (13)$$

where $\boldsymbol{\Phi} = [\boldsymbol{\varphi}_1, \boldsymbol{\varphi}_2, \ldots, \boldsymbol{\varphi}_m] \in \Re^{N\times m}$ is the regression matrix, with $\boldsymbol{\varphi}_i = [\varphi_i(\mathbf{x}(1)), \varphi_i(\mathbf{x}(2)), \ldots, \varphi_i(\mathbf{x}(N))]^{\mathrm{T}} \in \Re^N$, $i = 1, \ldots, m$. Each row in the whole mapping matrix $\boldsymbol{\Phi}$ denotes a high-dimensional mapping space for an input vector, while each column denotes one dimension for a subspace of all the input data. The LS-SVM classifier can thus be written as follows for a new test vector $\mathbf{x}$ from the input space:

$$f(\mathbf{x}) = \mathbf{h}\cdot\boldsymbol{\phi}(\mathbf{x}) = \sum_{i=1}^{m} h_i\varphi_i(\mathbf{x}). \quad (14)$$

Similar to the definition of SVs in an SVM and in the conventional solution of a LS-SVM, these $\mathbf{s}_i$ that here correspond to $h_i$ (having nonzero values) that contribute to the final model output are the SVs. As in the conventional solution to an LS-SVM where all the training patterns themselves act as SVs, the regression matrix $\boldsymbol{\Phi} \in \Re^{N\times m}$ $(m = N)$ produced from using all the training patterns as SVs in our proposed solution turns out to be

$$\boldsymbol{\Phi} \in \Re^{N\times N} = \begin{bmatrix} \varphi_1(\mathbf{x}(1)) & \varphi_2(\mathbf{x}(1)) & \cdots & \varphi_N(\mathbf{x}(1)) \\ \varphi_1(\mathbf{x}(2)) & \varphi_2(\mathbf{x}(2)) & \cdots & \varphi_N(\mathbf{x}(2)) \\ \cdots & \cdots & \cdots & \cdots \\ \varphi_1(\mathbf{x}(N)) & \varphi_2(\mathbf{x}(N)) & \cdots & \varphi_N(\mathbf{x}(N)) \end{bmatrix}. \quad (15)$$

This is identical to the kernel matrix $\mathbf{K}(\mathbf{x}(i), \mathbf{x}(j))$ presented previously for the conventional solution to LS-SVM. By using the conventional solution and our new solution to the primal objective problem (6), both objective values can be obtained, assuming that all the training patterns are viewed as SVs. The superiority of the new regression solution to the LS-SVM was compared with the conventional one in [29]. It can be observed that the kernel matrix $\mathbf{K}(\mathbf{x}(i), \mathbf{x}(j)) \in \Re^{N\times N}$ in the conventional solution is a special case of the regression matrix $\boldsymbol{\Phi} \in \Re^{N\times m}$ in our solution. However, both ours and the conventional solutions do not possess the sparseness property at this stage, which in fact represents the main drawback of the LS-SVM models. It is interesting to observe that the compulsory square property of the matrix $\mathbf{K}(\mathbf{x}(i), \mathbf{x}(j))$ in the KKT system (8) is no longer required in our regression matrix. Changes in the value of $m$ indicate how many SVs will be included in the final LS-SVM classifier and, in turn, determine the sparseness and the scale of the classifier. This is a very important characteristic for the novel learning mechanism to be presented in the next section. In the proposed algorithm, since every column in the matrix $\boldsymbol{\Phi}$ corresponds to one dimension of the mapped high-dimensional space, a subset of the training patterns can thus be chosen as the SVs in the LS-SVMs.

## IV. Novel Efficient Learning Mechanism

The aim of this paper is to develop a new fuzzy rule-based system based on a sparse LS-SVM learning mechanism with the model structure shown in Fig. 1. Similar as in SVM-based fuzzy systems [20] (where the kernel function in SVMs is related to the FBF), the FBF (4) is chosen as the mapping function (11) in our proposed solution of LS-SVM, i.e., $\varphi_i(\mathbf{x}(t)) = N_i(\mathbf{x}(t); \mathbf{W})$, to fuse the two systems into a new LS-SVM-based fuzzy rule-based system. Note that as usual, the denominator of the FBF is removed since the number of fuzzy rules is unknown in advance. There is no violation of the spirit of a fuzzy inference system as described in [20], where the rule premises determine the confidence values for all rules, while the rule consequents assign the consequence of the inference system with the confidence values for the corresponding rules. As a result, the SVs extracted from the LS-SVM learning mechanism can be applied in generating the fuzzy IF–THEN rules that correspond to the FBFs. In this manner, the fuzzy systems produced can provide satisfactory generalization capability over unseen data as in the case of LS-SVM. Different from the conventional LS-SVM where all training patterns serve as the SVs (thus causing nonsparseness), a novel sparse LS-SVM learning mechanism is proposed in this paper to produce rule selection in a fuzzy rule-based system.

The global optimization based on the new regression solution (13) of LS-SVMs still leads to the nonsparseness results, as in the conventional solution (9). To tackle this problem, an

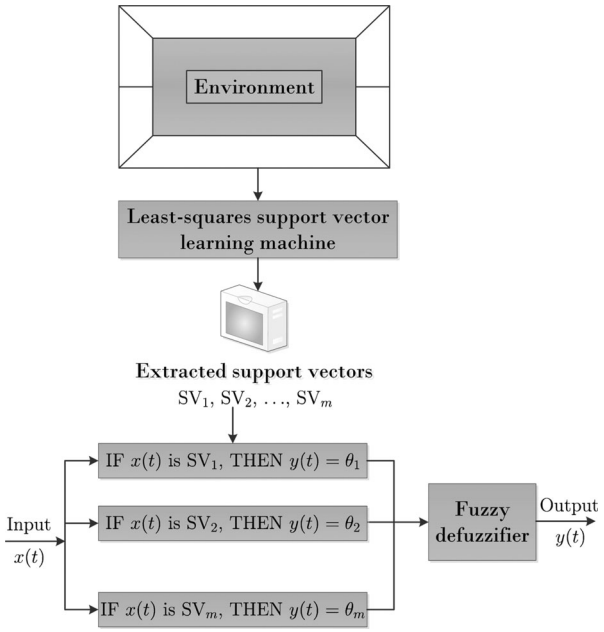



Fig. 1. LS-SVM-based fuzzy system.

efficient learning mechanism based on the subset selection approach is proposed here to find a small subset of SVs. This is, however, an NP-hard problem, which is widely acknowledged as being extremely difficult to solve in terms of algorithm performance and running time. It is generally impractical to find the global optimal subset by performing exhaustive search due to the huge computational burden (where the evaluation of all the possible combinations of subsets from a total number of $N$ candidate SVs is needed). This is also reflected in the experiment section. The novel learning mechanism proposed in this paper works in a stepwise subset selection manner, including a forward expansion phase and a backward exclusion phase on each selection step. The fast recursive algorithm presented in [14] is basically a fast and stable version of forward stepwise subset selection method working in the least-squares sense. It performs conditional optimization at each step under a given number of regressors that have been included in the subset, and the corresponding models are, therefore, usually suboptimal. Unlike the fast recursive algorithm, the novel learning mechanism consists of not only a forward expansion phase but a backward exclusion phase at each subset selection step as well, both also working in a new regularized least-squares sense. It is also different from the previously proposed second-stage algorithm [27], [30], which initially targets a subset of fixed size. The forward expansion phase at each step performs in the same way as in the fast recursive algorithm but within a regularized least-squares framework, instead of the least-squares approach. Here, each time, the most significant item from the candidate pool is added to the selected pool based in an efficient manner. The backward exclusion phase is devised to assess the least insignificant item that has been selected previously and, then, to determine whether or not to remove it from the current selected subset and return it to the candidate pool in order to determine a subset containing the most significant items.

For notation convenience, a similar residue matrix as in [14] is first defined as

$$\mathbf{R}_k \triangleq \mathbf{I} - \mathbf{\Phi}_k(\mathbf{\Phi}_k^{\mathrm{T}}\mathbf{\Phi}_k + \mu\mathbf{I})^{-1}\mathbf{\Phi}_k^{\mathrm{T}}, k = 1, \ldots, m \qquad (16)$$

where $\mathbf{\Phi}_k = [\mathbf{p}_1, \ldots, \mathbf{p}_k]$ represents the selected pool, which is a subset of the regression matrix $\mathbf{\Phi}$ and $\mathbf{R}_0 = \mathbf{I} \in \Re^{N\times N}$. If there is no prior knowledge about the system of interest, the number of initial regressors (equivalently SVs or fuzzy rules) can be set as $m = N$. It is not difficult to find that $\mathbf{R}_k = \mathbf{R}_k^{\mathrm{T}}$, and any changes in the order of the selected regressors $\mathbf{p}_1, \ldots, \mathbf{p}_k$ (i.e., column vectors in the regression matrix $\mathbf{\Phi}_k$) do not affect the value of $\mathbf{R}_k$. Based on the way in which the forward expansion and backward exclusion phases are performed, two basic theorems related to the residue matrix $\mathbf{R}_k$ are given below to facilitate the required sparseness learning for LS-SVM-based fuzzy systems.

*Theorem 1:* Assume $\mathbf{\Phi}_k = [\mathbf{p}_1, \ldots, \mathbf{p}_k]$ is of full-column rank and $\mathbf{R}_k$ is known *a priori*; then, the value of $\mathbf{R}_{k+1}$ computed from $\mathbf{R}_k$ by adding $\boldsymbol{\varphi}_i$ into $\mathbf{\Phi}_k$ is given as

$$\mathbf{R}_{k+1}([\mathbf{\Phi}_k; +\boldsymbol{\varphi}_i]) = \mathbf{R}_k - \frac{\mathbf{R}_k\boldsymbol{\varphi}_i\boldsymbol{\varphi}_i^{\mathrm{T}}\mathbf{R}_k^{\mathrm{T}}}{\boldsymbol{\varphi}_i^{\mathrm{T}}\mathbf{R}_k\boldsymbol{\varphi}_i + \mu} \quad k = 0, 1, \ldots, m-1, \quad i = k+1, \ldots, m. \qquad (17)$$

*Theorem 2:* Assume that $\mathbf{\Phi}_{k+1} = [\mathbf{p}_1, \ldots, \mathbf{p}_{k+1}]$ is of full-column rank and that $\mathbf{R}_{k+1}$ is known *a priori*; then, the value of $\mathbf{R}_k$ computed from $\mathbf{R}_{k+1}$ by removing $\mathbf{p}_i$ from $\mathbf{\Phi}_{k+1}$ is given as

$$\mathbf{R}_k([\mathbf{\Phi}_{k+1}; -\mathbf{p}_i]) = \mathbf{R}_{k+1} + \frac{\mathbf{R}_{k+1}\mathbf{p}_i\mathbf{p}_i^{\mathrm{T}}\mathbf{R}_{k+1}^{\mathrm{T}}}{\mu - \mathbf{p}_i^{\mathrm{T}}\mathbf{R}_{k+1}\mathbf{p}_i} \quad k = 0, 1, \ldots, m-1, \quad i = 1, \ldots, k+1. \qquad (18)$$

In the above two theorems, $[\mathbf{\Phi}_k; +\boldsymbol{\varphi}_i]$ denotes adding a new regressor $\boldsymbol{\varphi}_i$ from the candidate pool $\mathbf{\Psi}_k$ into the selected pool $\mathbf{\Phi}_k$, and $[\mathbf{\Phi}_{k+1}; -\mathbf{p}_i]$ denotes removing a selected regressor $\mathbf{p}_i$ from the selected pool $\mathbf{\Phi}_{k+1}$. The proofs of these two theorems are given in Appendix A. In addition, note that the initial candidate pool is set as $\mathbf{\Psi}_0 = [\boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_m]$.

According to the solution given in (13), the optimal objective function (12) to the LS-SVM is computed as

$$\begin{aligned} J(\hat{\mathbf{h}}) &= \frac{1}{2\mu}\{\mu\hat{\mathbf{h}}^{\mathrm{T}}\hat{\mathbf{h}} + (\mathbf{y} - \mathbf{\Phi}\hat{\mathbf{h}})^{\mathrm{T}}(\mathbf{y} - \mathbf{\Phi}\hat{\mathbf{h}})\} \\ &= \frac{1}{2\mu}\{\mu\mathbf{y}^{\mathrm{T}}\mathbf{\Phi}(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi} + \mu\mathbf{I})^{-2}\mathbf{\Phi}^{\mathrm{T}}\mathbf{y} \\ &\quad + \mathbf{y}^{\mathrm{T}}[\mathbf{I} - \mathbf{\Phi}(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi} + \mu\mathbf{I})^{-1}\mathbf{\Phi}^{\mathrm{T}}]^2\mathbf{y}\} \\ &= \frac{1}{2\mu}\mathbf{y}^{\mathrm{T}}\{\mu\mathbf{\Phi}(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}+\mu\mathbf{I})^{-2}\mathbf{\Phi}^{\mathrm{T}} + \mathbf{\Phi}(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}+\mu\mathbf{I})^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi} \\ &\quad \times (\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi} + \mu\mathbf{I})^{-1}\mathbf{\Phi}^{\mathrm{T}} + \mathbf{I} - 2\mathbf{\Phi}(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi} + \mu\mathbf{I})^{-1}\mathbf{\Phi}^{\mathrm{T}}\}\mathbf{y} \\ &= \frac{1}{2\mu}\mathbf{y}^{\mathrm{T}}\{\mathbf{\Phi}[\mu(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi} + \mu\mathbf{I})^{-1} + (\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi} + \mu\mathbf{I})^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}] \\ &\quad \times (\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi} + \mu\mathbf{I})^{-1}\mathbf{\Phi}^{\mathrm{T}} + \mathbf{I} - 2\mathbf{\Phi}(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi} + \mu\mathbf{I})^{-1}\mathbf{\Phi}^{\mathrm{T}}\}\mathbf{y} \\ &= \frac{1}{2\mu}\mathbf{y}^{\mathrm{T}}\{\mathbf{I} - \mathbf{\Phi}(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi} + \mu\mathbf{I})^{-1}\mathbf{\Phi}^{\mathrm{T}}\}\mathbf{y}. \end{aligned} \qquad (19)$$

Considering the residue matrix defined in (16), the optimal value of the objective function (12) by using $\mathbf{\Phi}_k$ becomes

$$J_k = \frac{\mathbf{y}^{\mathrm{T}}\mathbf{R}_k\mathbf{y}}{2\mu}. \tag{20}$$

Thus, on adding one new regressor, say $\boldsymbol{\varphi}_i$ $(i = k+1, \ldots, m)$, into the selected pool in the forward expansion phase at the $(k+1)$th subset selection step, the objective value is correspondingly decreased by

$$\begin{aligned}\Delta\overrightarrow{J}_{k+1}(\boldsymbol{\varphi}_i) &= \frac{\mathbf{y}^{\mathrm{T}}(\mathbf{R}_k - \mathbf{R}_{k+1}([\mathbf{\Phi}_k; +\boldsymbol{\varphi}_i]))\mathbf{y}}{2\mu} \\ &= \frac{1}{2\mu}\frac{\mathbf{y}^{\mathrm{T}}\mathbf{R}_k\boldsymbol{\varphi}_i\boldsymbol{\varphi}_i^{\mathrm{T}}\mathbf{R}_k^{\mathrm{T}}\mathbf{y}}{\boldsymbol{\varphi}_i^{\mathrm{T}}\mathbf{R}_k\boldsymbol{\varphi}_i + \mu}.\end{aligned} \tag{21}$$

On the contrary, deleting one such regressor, say $\mathbf{p}_i$ $(i = 1, \ldots, k+1)$, from the selected pool in the backward exclusion phase at the $(k+1)$th subset selection step, the objective value is correspondingly increased by

$$\begin{aligned}\Delta\overleftarrow{J}_{k+1}(\mathbf{p}_i) &= \frac{\mathbf{y}^{\mathrm{T}}(\mathbf{R}_k([\mathbf{\Phi}_{k+1}; -\mathbf{p}_i]) - \mathbf{R}_{k+1})\mathbf{y}}{2\mu} \\ &= \frac{1}{2\mu}\frac{\mathbf{y}^{\mathrm{T}}\mathbf{R}_{k+1}\mathbf{p}_i\mathbf{p}_i^{\mathrm{T}}\mathbf{R}_{k+1}^{\mathrm{T}}\mathbf{y}}{\mu - \mathbf{p}_i^{\mathrm{T}}\mathbf{R}_{k+1}\mathbf{p}_i}.\end{aligned} \tag{22}$$

In summary, at the $(k+1)$th subset selection step, the forward expansion phase is first executed, where the regressor producing the largest objective reduction is chosen as the $(k+1)$th regressor and is involved in the selected pool, i.e., $\mathbf{p}_{k+1} = \arg\max_{i=k+1}^{m}\Delta\overrightarrow{J}_{k+1}(\boldsymbol{\varphi}_i)$. When the forward expansion phase is completed, the backward exclusion phase is executed to review the contribution of all previously selected regressors. This is done by excluding the regressor with the smallest contribution from the selected pool and, meanwhile, returning it to the candidate pool, i.e., $\mathbf{p}_r = \arg\min_{i=1}^{k+1}\Delta\overleftarrow{J}_{k+1}(\mathbf{p}_i)$. Note that if $\mathbf{p}_r$ is exactly the regressor $\mathbf{p}_{k+1}$ just selected at the forward expansion phase, then the backward exclusion phase is neglected. In this circumstance, it means that all the regressors in the current selected pool are significant and, thus, that no backward exclusion is needed. Since $\Delta\overleftarrow{J}_{k+1}(\mathbf{p}_{k+1}) = \max_{i=k+1}^{m}\Delta\overrightarrow{J}_{k+1}(\boldsymbol{\varphi}_i)$, one can use $\min_{i=1}^{k}\Delta\overleftarrow{J}_{k+1}(\mathbf{p}_i) < \max_{i=k+1}^{m}\Delta\overrightarrow{J}_{k+1}(\boldsymbol{\varphi}_i)$ as the criterion to determine whether to remove a regressor from the selected pool or not. To efficiently compute the regressor contributions based on (21) and (22), the following two sections give the efficient learning mechanism for producing sparse LS-SVM-based fuzzy systems.

### A. Forward Expansion Phase

In each forward expansion phase, the net contribution of a regressor from the candidate pool to the objective function is expressed in (21). Suppose that the $k$th regressor has just been added into the selected pool; an intermediate matrix $\mathbf{A} \in \Re^{k\times m}$ is thus generated with the $k$th row calculated as

$$a_{k,i} = \begin{cases}\mathbf{p}_k^{\mathrm{T}}\mathbf{R}_{k-1}\mathbf{p}_i, & i = 1, \ldots, k \\ \mathbf{p}_k^{\mathrm{T}}\mathbf{R}_{k-1}\boldsymbol{\varphi}_i, & i = k+1, \ldots, m.\end{cases} \tag{23}$$

Note that the first $k-1$ rows are, therefore, generated in the same way each time a new regressor is included into the model. Thus, the number of rows in matrix $\mathbf{A}$ increases by one as the selection procedure proceeds. By successively using (17), the following can be inferred for efficient computation:

$$a_{k,i} = \begin{cases}\mathbf{p}_k^{\mathrm{T}}\mathbf{p}_i - \sum_{j=1}^{k-1} a_{j,k}a_{j,i}/(a_{j,j}+\mu), & i = 1, \ldots, k \\ \mathbf{p}_k^{\mathrm{T}}\boldsymbol{\varphi}_i - \sum_{j=1}^{k-1} a_{j,k}a_{j,i}/(a_{j,j}+\mu), & i = k+1, \ldots, m.\end{cases} \tag{24}$$

To continue decreasing the computational complexity of the left-hand side entries in the $k$th row, it follows that

$$a_{k,i} = \begin{cases}\mu a_{i,k}/(a_{i,i}+\mu) - \sum_{j=i+1}^{k-1} a_{j,k}a_{j,i}/(a_{j,j}+\mu) & \\ \qquad\qquad i = 1, \ldots, k-1 & \\ \mathbf{p}_k^{\mathrm{T}}\mathbf{p}_k - \sum_{j=1}^{k-1} a_{j,k}^2/(a_{j,j}+\mu), & i = k \\ \mathbf{p}_k^{\mathrm{T}}\boldsymbol{\varphi}_i - \sum_{j=1}^{k-1} a_{j,k}a_{j,i}/(a_{j,j}+\mu), & i = k+1, \ldots, m.\end{cases} \tag{25}$$

Further define a vector $\mathbf{b}^{k+1)} \in \Re^m$, where its entries at the $(k+1)$th step are calculated as

$$b_i^{k+1)} = \begin{cases}\mathbf{p}_i^{\mathrm{T}}\mathbf{R}_{i-1}\mathbf{y}, & i = 1, \ldots, k \\ \boldsymbol{\varphi}_i^{\mathrm{T}}\mathbf{R}_k\mathbf{y}, & i = k+1, \ldots, m\end{cases} \tag{26}$$

and another vector $\mathbf{d}^{k+1)} \in \Re^m$, where

$$d_i^{k+1)} = \begin{cases}\mathbf{p}_i^{\mathrm{T}}\mathbf{R}_{i-1}\mathbf{p}_i, & i = 1, \ldots, k \\ \boldsymbol{\varphi}_i^{\mathrm{T}}\mathbf{R}_k\boldsymbol{\varphi}_i, & i = k+1, \ldots, m.\end{cases} \tag{27}$$

The values for $i = 1, \ldots, k$ are kept unchanged from previous selection steps, and then, using (17) for $i = k+1, \ldots, m$ yields

$$b_i^{k+1)} = \begin{cases}b_i^{k)}, & i = 1, \ldots, k \\ b_i^{k)} - b_k^{k)}a_{k,i}/(a_{k,k}+\mu), & i = k+1, \ldots, m\end{cases} \tag{28}$$

$$d_i^{k+1)} = \begin{cases}d_i^{k)}, & i = 1, \ldots, k \\ a_i^{k)} - a_{k,i}^2/(a_{k,k}+\mu), & i = k+1, \ldots, m.\end{cases} \tag{29}$$

With the aid of the matrix $\mathbf{A}$ and the vectors $\mathbf{b}$ and $\mathbf{d}$, the contribution of regressor $\boldsymbol{\varphi}_i$ $(i = k+1, \ldots, m)$ from the candidate pool at the $(k+1)$th step can be reexpressed as

$$\Delta\overrightarrow{J}_{k+1}(\boldsymbol{\varphi}_i) = \frac{1}{2\mu}\frac{(b_i^{k+1)})^2}{d_i^{k+1)} + \mu}, \quad i = k+1, \ldots, m. \tag{30}$$

As a result, the one with the largest objective reduction is selected as the $(k+1)$th regressor to be included into the system, i.e., $\mathbf{p}_{k+1} = \arg\max_{i=k+1}^{m}\Delta\overrightarrow{J}_{k+1}(\boldsymbol{\varphi}_i)$. As long as this new regressor is included in the selected pool, the next phase is to review the significance of all the previously selected regressors.

### B. Backward Exclusion Phase

*1) Model Review:* Continuing from the forward expansion phase, a total of $k+1$ regressors have now been included in the selected pool. Thus, the intermediate matrix/vectors $\mathbf{A} \in \Re^{(k+1)\times m}$, $\mathbf{b}^{k+2)} \in \Re^m$, and $\mathbf{d}^{k+2)} \in \Re^m$ have been updated correspondingly. A backward exclusion phase is then to be performed, in which the significance of each selected regressor in terms of the objective function is reevaluated as in (22). Two vectors $\mathbf{c}^{k+1)} \in \Re^m$ and $\mathbf{h}^{k+1)} \in \Re^m$ are first defined with their entries at the $(k+1)$th step given by

$$c_i^{k+1)} = \begin{cases} \mathbf{p}_i^{\mathrm{T}} \mathbf{R}_{k+1} \mathbf{y}, & i = 1, \ldots, k+1 \\ \boldsymbol{\varphi}_i^{\mathrm{T}} \mathbf{R}_{k+1} \mathbf{y}, & i = k+2, \ldots, m \end{cases} \tag{31}$$

$$h_i^{k+1)} = \begin{cases} \mathbf{p}_i^{\mathrm{T}} \mathbf{R}_{k+1} \mathbf{p}_i, & i = 1, \ldots, k+1 \\ \boldsymbol{\varphi}_i^{\mathrm{T}} \mathbf{R}_{k+1} \boldsymbol{\varphi}_i, & i = k+2, \ldots, m. \end{cases} \tag{32}$$

Using (17) and comparing with the entries in $\mathbf{b}^{k+2)}$ and $\mathbf{d}^{k+2)}$, the following results can be obtained:

$$c_i^{k+1)} = \begin{cases} c_i^{k)} - a_{k+1,i} b_{k+1} / (a_{k+1,k+1} + \mu) \\ \qquad\qquad i = 1, \ldots, k+1 \\ b_i^{k+2)}, \qquad i = k+2, \ldots, m \end{cases} \tag{33}$$

$$h_i^{k+1)} = \begin{cases} h_i^{k)} - a_{k+1,i}^2 / (a_{k+1,k+1} + \mu) \\ \qquad\qquad i = 1, \ldots, k+1 \\ d_i^{k+2)}, \qquad i = k+2, \ldots, m. \end{cases} \tag{34}$$

This way, the significance of a selected regressor $\mathbf{p}_i$ given in (22) can be computed as

$$\Delta \overleftarrow{J}_{k+1}(\mathbf{p}_i) = \frac{1}{2\mu} \frac{(c_i^{k+1)})^2}{\mu - h_i^{k+1)}}, \quad i = 1, \ldots, k+1. \tag{35}$$

As discussed before, if the criterion $\min_{i=1}^{k} \Delta \overleftarrow{J}_{k+1}(\mathbf{p}_i) < \max_{i=k+1}^{m} \Delta \overrightarrow{J}_{k+1}(\boldsymbol{\varphi}_i)$ is satisfied, then the previously selected regressor with the least significance to the objective function, say $\mathbf{p}_r$, will be excluded from the selected pool and returned into the candidate pool. An efficient process for removing this regressor from the selected pool is now detailed as follows.

*2) Regression Context Reconstruction:* All the intermediate matrix and vectors used in the aforementioned forward expansion and backward exclusion phases, such as $\mathbf{A} \in \Re^{(k+1)\times m}$, $\mathbf{b}^{k+2)} \in \Re^m$, $\mathbf{c}^{k+1)} \in \Re^m$, $\mathbf{d}^{k+2)} \in \Re^m$, and $\mathbf{h}^{k+1)} \in \Re^m$, are the key ideas behind the proposed algorithm and are referred to as the *regression context* as in [27]. However, if one selected regressor needs to be removed from the selected pool, as described in Section IV-B.1, the *regression context* has to be updated. The new *regression context* can be obtained by only again performing the forward expansion procedure using the current selected order of regressors. Unfortunately, this is computationally inefficient. Based on the techniques introduced in [27], a computationally more efficient algorithm is presented for reordering the selected regressors. In more detail, suppose a previously selected regressor $\mathbf{p}_r$ is going to be removed from the current selected pool; then, we first have to shift $\mathbf{p}_r$ to the $(k+1)$th position (the last position) in the regression matrix $\boldsymbol{\Phi}_{k+1}$ as if it was the last selected regressor. This shifted regressor $\mathbf{p}_r$ is then deleted from the last position in $\boldsymbol{\Phi}_{k+1}$.

To shift $\mathbf{p}_r$ to the last position in regression matrix $\boldsymbol{\Phi}_{k+1}$, a series of basic interchanges between two adjacent regressors have to be performed such that

$$\bar{\mathbf{p}}_q = \mathbf{p}_{q+1}, \bar{\mathbf{p}}_{q+1} = \mathbf{p}_q, \quad q = r, \ldots, k. \tag{36}$$

Noting that any changes in the selected order of the regressor terms do not affect the value of $\mathbf{R}_{k+1}$, it is obvious that $\bar{\mathbf{R}}_i$ $(i = 1, \ldots, k+1)$ only changes when $i = q$

$$\begin{aligned} \bar{\mathbf{R}}_q &= \mathbf{R}([\mathbf{p}_1, \ldots, \mathbf{p}_{q-1}, \mathbf{p}_{q+1}]) \\ &= \mathbf{R}_{q-1} - \frac{\mathbf{R}_{q-1} \mathbf{p}_{q+1} \mathbf{p}_{q+1}^{\mathrm{T}} \mathbf{R}_{q-1}^{\mathrm{T}}}{\mathbf{p}_{q+1}^{\mathrm{T}} \mathbf{R}_{q-1} \mathbf{p}_{q+1} + \mu}. \end{aligned} \tag{37}$$

With respect to matrix $\mathbf{A}$, only the $q$th and the $(q+1)$th columns, as well as the $q$th and $(q+1)$th rows, are changed. For both the $q$th and $(q+1)$th columns, the following holds for $i = 1, \ldots, q-1, q+2, \ldots, k+1$:

$$\begin{cases} \bar{a}_{i,q} = \bar{\mathbf{p}}_i^{\mathrm{T}} \bar{\mathbf{R}}_{i-1} \bar{\mathbf{p}}_q = \mathbf{p}_i^{\mathrm{T}} \mathbf{R}_{i-1} \mathbf{p}_{q+1} = a_{i,q+1} \\ \bar{a}_{i,q+1} = \bar{\mathbf{p}}_i^{\mathrm{T}} \bar{\mathbf{R}}_{i-1} \bar{\mathbf{p}}_{q+1} = \mathbf{p}_i^{\mathrm{T}} \mathbf{R}_{i-1} \mathbf{p}_q = a_{i,q}. \end{cases} \tag{38}$$

In the $q$th row, entries from columns 1 to $m$ are changed according to

$$\begin{aligned} \bar{a}_{q,i} &= \bar{\mathbf{p}}_q^{\mathrm{T}} \bar{\mathbf{R}}_{q-1} \bar{\mathbf{p}}_i = \mathbf{p}_{q+1}^{\mathrm{T}} \left( \mathbf{R}_q + \frac{\mathbf{R}_{q-1} \mathbf{p}_q \mathbf{p}_q^{\mathrm{T}} \mathbf{R}_{q-1}}{\mathbf{p}_q^{\mathrm{T}} \mathbf{R}_{q-1} \mathbf{p}_q + \mu} \right) \bar{\mathbf{p}}_i \\ &= \begin{cases} a_{q,q+1}, & i = q+1 \\ a_{q+1,q+1} + a_{q,q+1}^2 / (a_{q,q} + \mu), & i = q \\ a_{q+1,i} + a_{q,q+1} a_{q,i} / (a_{q,q} + \mu), & i = 1, \ldots, q-1 \\ & \quad q+2, \ldots, m. \end{cases} \end{aligned} \tag{39}$$

Similarly, in the $(q+1)$th row, entries from columns 1 to $m$ are changed as

$$\begin{aligned} \bar{a}_{q+1,i} &= \bar{\mathbf{p}}_{q+1}^{\mathrm{T}} \bar{\mathbf{R}}_q \bar{\mathbf{p}}_i = \mathbf{p}_q^{\mathrm{T}} (\mathbf{R}_{q-1} - \frac{\mathbf{R}_{q-1} \mathbf{p}_{q+1} \mathbf{p}_{q+1}^{\mathrm{T}} \mathbf{R}_{q-1}}{\mathbf{p}_{q+1}^{\mathrm{T}} \mathbf{R}_{q-1} \mathbf{p}_{q+1} + \mu}) \bar{\mathbf{p}}_i \\ &= \begin{cases} a_{q,q} - a_{q,q+1} \bar{a}_{q,q+1} / (\bar{a}_{q,q} + \mu), & i = q+1 \\ \mu a_{q,q+1} / (\bar{a}_{q,q} + \mu), & i = q \\ a_{q,i} - a_{q,q+1} \bar{a}_{q,i} / (\bar{a}_{q,q} + \mu), & i = 1, \ldots, q-1 \\ & \quad q+2, \ldots, m. \end{cases} \end{aligned} \tag{40}$$

With respect to the vector $\mathbf{b}^{k+2)}$, only the $q$th and the $(q+1)$th entries are changed using (17) and (37) as before:

$$\begin{cases} \bar{b}_q^{k+2)} = \bar{\mathbf{p}}_q^{\mathrm{T}} \bar{\mathbf{R}}_{q-1} \mathbf{y} = b_{q+1}^{k+2)} + a_{q,q+1} b_q^{k+2)} / (a_{q,q} + \mu) \\ \bar{b}_{q+1}^{k+2)} = \bar{\mathbf{p}}_{q+1}^{\mathrm{T}} \bar{\mathbf{R}}_q \mathbf{y} = b_q^{k+2)} - a_{q,q+1} \bar{b}_q^{k+2)} / (\bar{a}_{q,q} + \mu). \end{cases} \tag{41}$$

For the vector $\mathbf{d}^{k+2)}$, only the $q$th and $(q+1)$th entries are changed using the corresponding diagonal entries of matrix $\mathbf{A}$:

$$\bar{d}_q^{k+2)} = \bar{a}_{q,q}, \bar{d}_{q+1}^{k+2)} = \bar{a}_{q+1,q+1}. \tag{42}$$

Finally, for the two vectors $\mathbf{c}^{k+1)}$ and $\mathbf{h}^{k+1)}$, only the $q$th and the $(q+1)$th entries are interchanged:

$$\begin{cases} \bar{c}_q^{k+1)} = \bar{\mathbf{p}}_q^{\mathrm{T}} \bar{\mathbf{R}}_{k+1} \mathbf{y} = \mathbf{p}_{q+1}^{\mathrm{T}} \mathbf{R}_{k+1} \mathbf{y} = c_{q+1}^{k+1)} \\ \bar{c}_{q+1}^{k+1)} = \bar{\mathbf{p}}_{q+1}^{\mathrm{T}} \bar{\mathbf{R}}_{k+1} \mathbf{y} = \mathbf{p}_q^{\mathrm{T}} \mathbf{R}_{k+1} \mathbf{y} = c_q^{k+1)} \end{cases} \tag{43}$$

$$\begin{cases} \bar{h}_q^{k+1)} = \bar{\mathbf{p}}_q^{\mathrm{T}} \bar{\mathbf{R}}_{k+1} \bar{\mathbf{p}}_q = \mathbf{p}_{q+1}^{\mathrm{T}} \mathbf{R}_{k+1} \mathbf{p}_{q+1} = h_{q+1}^{k+1)} \\ \bar{h}_{q+1}^{k+1)} = \bar{\mathbf{p}}_{q+1}^{\mathrm{T}} \bar{\mathbf{R}}_{k+1} \bar{\mathbf{p}}_{q+1} = \mathbf{p}_q^{\mathrm{T}} \mathbf{R}_{k+1} \mathbf{p}_q = h_q^{k+1)}. \end{cases} \tag{44}$$

After performing $k-r+1$ interchanges between two adjacent selected regressors as described above, the $r$th regressor $\mathbf{p}_r$ can be finally moved to the $(k+1)$th position of $\mathbf{\Phi}_{k+1} = [\bar{\mathbf{p}}_1, \ldots, \bar{\mathbf{p}}_{k+1}]$, i.e, $\mathbf{p}_r = \bar{\mathbf{p}}_{k+1}$, to produce a new *regression context*.

*3) Regressor Exclusion:* Once the regressor $\mathbf{p}_r$ in the last position of the regression matrix has been determined for removal, the new *regression context* then just requires some small changes. First, the selected pool is temporarily updated by $\mathbf{\Phi}_k = [\bar{\mathbf{p}}_1, \ldots, \bar{\mathbf{p}}_k]$ and the candidate pool by $\mathbf{\Psi}_k = [\boldsymbol{\varphi}_{k+1}, \ldots, \boldsymbol{\varphi}_m]$, where $\boldsymbol{\varphi}_{k+1}$ (or equivalently $\bar{\mathbf{p}}_{k+1}$ or $\mathbf{p}_r$) is the regressor that was removed from $\mathbf{\Phi}_{k+1}$. Thus, the residue matrix $\bar{\mathbf{R}}_k$ after removing $\bar{\mathbf{p}}_{k+1}$ is given by

$$\bar{\mathbf{R}}_k = \mathbf{R}([\bar{\mathbf{p}}_1, \ldots, \bar{\mathbf{p}}_k]) = \bar{\mathbf{R}}_{k+1} + \frac{\bar{\mathbf{R}}_k \bar{\mathbf{p}}_{k+1} \bar{\mathbf{p}}_{k+1}^{\mathrm{T}} \bar{\mathbf{R}}_k^{\mathrm{T}}}{\bar{\mathbf{p}}_{k+1}^{\mathrm{T}} \bar{\mathbf{R}}_k \bar{\mathbf{p}}_{k+1} + \mu}. \tag{45}$$

Using this formula, the vector $\mathbf{b}^{k+1)}$ is updated by

$$\bar{b}_i^{k+1)} = \begin{cases} \bar{b}_i^{k+2)}, & i = 1, \ldots, k+1 \\ \bar{b}_i^{k+2)} + \bar{b}_{k+1}^{k+2)} \bar{a}_{k+1,i} / (\bar{a}_{k+1,k+1} + \mu), & i = k+2, \ldots, m \end{cases} \tag{46}$$

and the vector $\mathbf{d}^{k+1)}$ is updated by

$$\bar{d}_i^{k+1)} = \begin{cases} \bar{d}_i^{k+2)}, & i = 1, \ldots, k+1 \\ \bar{d}_i^{k+2)} + \bar{a}_{k+1,i}^2 / (\bar{a}_{k+1,k+1} + \mu), & i = k+2, \ldots, m. \end{cases} \tag{47}$$

Since the removed regressor $\bar{\mathbf{p}}_{k+1}$ may be again selected in a subsequent forward expansion phase at the next selection step, then this regressor should not be excluded from the selected pool. In this case, there is no need to perform any changes on the *regression context* obtained in Section IV-B2. To further reduce the computation time, by just employing the values of $\bar{b}_i^{k+1)}$ and $\bar{d}_i^{k+1)}$ obtained and using (30), it is now ready to determine which one is the new $(k+1)$th regressor to be selected at the next forward expansion phase. This way, $\Delta \overleftarrow{J}_{k+1}(\mathbf{p}_r) = \min_{i=1}^{k} \Delta \overleftarrow{J}_{k+1}(\mathbf{p}_i) > \max_{i=k+2}^{m} \Delta \overrightarrow{J}_{k+1}^{*}(\boldsymbol{\varphi}_i)$ is applied to avoid removing the right regressor that has been selected previously, where $\Delta \overrightarrow{J}_{k+1}^{*}(\boldsymbol{\varphi}_i) = (\bar{b}_i^{k+1)})^2 / (2\mu(\bar{d}_i^{k+1)} + \mu))$, $i = k+2, \ldots, m$. If the regressor $\bar{\mathbf{p}}_{k+1}$ has been marked for removal from the selected pool, then the vector $\mathbf{b}^{k+1)}$ is assigned with entries $\tilde{b}_i^{k+1)} = \bar{b}_i^{k+1)}$ and the vector $\mathbf{d}^{k+1)}$ with entries $\tilde{d}_i^{k+1)} = \bar{d}_i^{k+1)}$ $(i = 1, \ldots, m)$, and the following two vectors $\mathbf{c}^{k)}$ and $\mathbf{h}^{k)}$ are updated:

$$\tilde{c}_i^{k)} = \begin{cases} \bar{c}_i^{k+1)} + \bar{b}_{k+1}^{k+2)} \bar{a}_{k+1,i} / (\bar{a}_{k+1,k+1} + \mu), & i = 1, \ldots, k \\ \tilde{b}_i^{k+1)}, & i = k+1, \ldots, m \end{cases} \tag{48}$$

$$\tilde{h}_i^{k)} = \begin{cases} \bar{h}_i^{k+1)} + \bar{a}_{k+1,i}^2 / (\bar{a}_{k+1,k+1} + \mu), & i = 1, \ldots, k \\ \bar{d}_i^{k+1)}, & i = k+1, \ldots, m. \end{cases} \tag{49}$$

In the case of the matrix $\mathbf{A}$, only the $(k+1)$th row is removed with the others remain unchanged. Obviously, the selected pool $\mathbf{\Phi}_k$ and the candidate pool $\mathbf{\Psi}_k$ are updated using $\tilde{\mathbf{p}}_i = \bar{\mathbf{p}}_i$ for $(i = 1, \ldots, k)$ and $\tilde{\boldsymbol{\varphi}}_i = \boldsymbol{\varphi}_i$ for $(i = k+1, \ldots, m)$. Thus far, the *regression context* $\mathbf{A} \in \Re^{k \times m}$, $\mathbf{b}^{k+1)} \in \Re^m$, $\mathbf{c}^{k)} \in \Re^m$, $\mathbf{d}^{k+1)} \in \Re^m$, and $\mathbf{h}^{k)} \in \Re^m$ is ready for use in the following forward expansion phase at the next selection step.

### C. Computation of Model Parameters

Assuming that a total of $M$ rules have finally been selected by the proposed method, and using the definition of $\mathbf{R}_k$ defined in (16), the model parameters are computed from (13)

$$\begin{aligned} \hat{\boldsymbol{h}}_M &= (\mathbf{\Phi}_M^{\mathrm{T}} \mathbf{\Phi}_M + \mu \mathbf{I})^{-1} \mathbf{\Phi}_M^{\mathrm{T}} \mathbf{y} = \frac{1}{\mu} \mathbf{\Phi}_M^{\mathrm{T}} \mathbf{R}_M \mathbf{y} \\ &= \frac{1}{\mu} \begin{pmatrix} \mathbf{p}_1^{\mathrm{T}} \mathbf{R}_M \mathbf{y} \\ \cdots \\ \mathbf{p}_M^{\mathrm{T}} \mathbf{R}_M \mathbf{y} \end{pmatrix} = \frac{1}{\mu} \begin{pmatrix} c_1^{M)} \\ \cdots \\ c_M^{M)} \end{pmatrix} \end{aligned} \tag{50}$$

where $c_i^{M)}$, $i = 1, \ldots, M$, are the first $M$ entries obtained from the final value of the vector $\mathbf{c}^{M)} \in \Re^M$. Note that if only the forward expansion phase is considered in the selection procedure, then the related model parameters are computed as $\hat{\boldsymbol{h}}_M = [\hat{h}_{M,1}, \ldots, \hat{h}_{M,M}]^{\mathrm{T}}$, in which $\hat{h}_{M,i}$ is given as follows from (17) and (50):

$$\hat{h}_{M,i} = \frac{b_i^{M)}}{a_{i,i} + \mu} - \frac{1}{\mu} \sum_{j=i+1}^{M} \frac{a_{j,i} b_j^{M)}}{a_{j,j} + \mu}, i = 1, \ldots, M. \tag{51}$$

### D. Algorithm: Construction of Sparse least-Squares Support-Vector-Machine-Based Fuzzy Systems

The efficient learning mechanism of the sparse LS-SVM-based fuzzy systems is shown in the flowchart in Fig. 2 and is detailed as follows.

*Step 1) Initialization:* To start the learning process, the candidate pool $\mathbf{\Psi}_0 = [\boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_m]$ is first generated by using all the training patterns as the potential rules/SVs. Note that the initially selected pool $\mathbf{\Phi}_0$ is an empty matrix. The number of selected regressors is set to $k = 0$, and the two vectors $\mathbf{b}^{1)} = [\boldsymbol{\varphi}_1^{\mathrm{T}} \mathbf{y}, \ldots, \boldsymbol{\varphi}_m^{\mathrm{T}} \mathbf{y}]$ and $\mathbf{d}^{1)} = [\boldsymbol{\varphi}_1^{\mathrm{T}} \boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_m^{\mathrm{T}} \boldsymbol{\varphi}_m]$ are initialized.

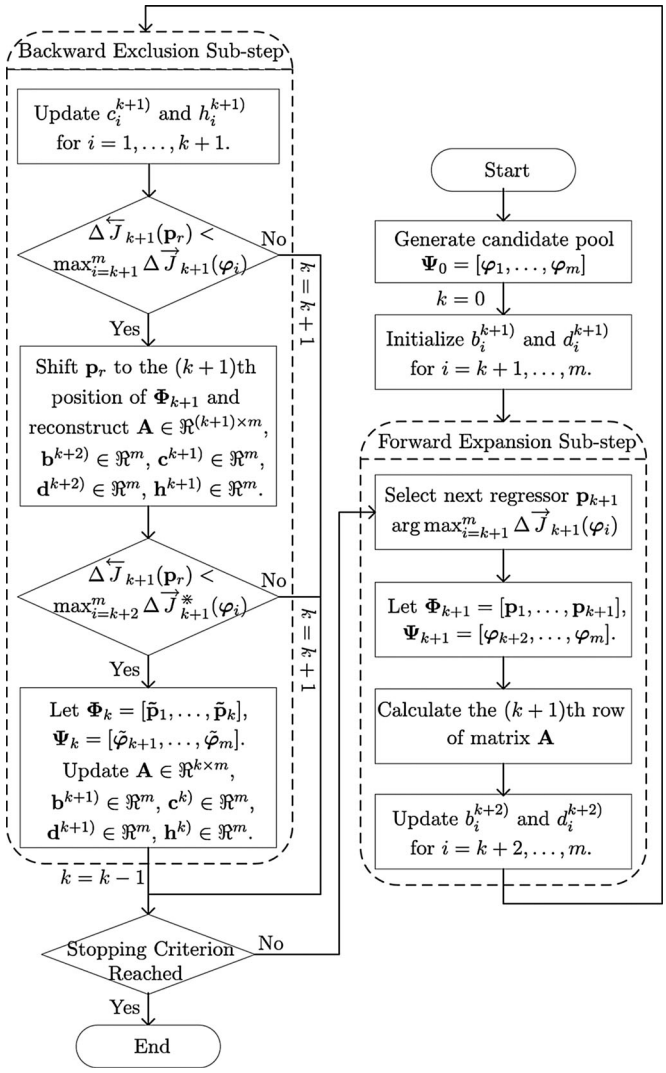



Fig. 2. Flowchart of the proposed efficient learning mechanism for constructing sparse LS-SVM-based fuzzy systems.

*Step 2) Forward expansion phase:* The main task here is to select the most significant regressor from the candidate pool and to update the corresponding variables for the operations ahead.

1) According to the contribution of each candidate regressor computed from (30), the one with the largest objective reduction is selected as the next regressor to be added into the regression matrix $\boldsymbol{\Phi}_{k+1} = [\mathbf{p}_1, \ldots, \mathbf{p}_{k+1}]$, i.e., $\mathbf{p}_{k+1} = \arg\max_{i=k+1}^{m} \Delta \overrightarrow{J}_{k+1}(\boldsymbol{\varphi}_i)$. The corresponding regressor $\mathbf{p}_{k+1}$ is then removed from the candidate pool and $\boldsymbol{\Psi}_{k+1} = [\boldsymbol{\varphi}_{k+2}, \ldots, \boldsymbol{\varphi}_m]$ set.
2) The $(k+1)$th row of matrix $\mathbf{A}$ is calculated using (25), while all the previous $k$ rows remain unchanged.
3) The two vectors $\mathbf{b}^{k+2)}$ and $\mathbf{d}^{k+2)}$ are updated with entries from $k+2$ to $m$ by using (28) and (29) and are employed for selecting the $(k+2)$th regressor from the candidate pool.

*Step 3) Backward exclusion phase:* The main purpose of this phase is to reevaluate the contribution of each of the previously selected regressors.

1) The entries from 1 to $k+1$ for the two vectors $\mathbf{c}^{k+1)}$ and $\mathbf{h}^{k+1)}$ are updated using (33) and (34), while the correspondingly remaining values in the two vectors are inherited from $\mathbf{b}^{k+2)}$ and $\mathbf{d}^{k+2)}$.
2) The criterion $\Delta \overleftarrow{J}_{k+1}(\mathbf{p}_r) = \min_{i=1}^{k} \Delta \overleftarrow{J}_{k+1}(\mathbf{p}_i) < \max_{i=k+1}^{m} \Delta \overrightarrow{J}_{k+1}(\boldsymbol{\varphi}_i)$ is used to decide whether to remove a regressor from the selected pool or not, and to determine which one is to be removed. If the criterion is not met, then set $k = k+1$ and go to Step 4. Otherwise, move to the next step.
3) The regressor $\mathbf{p}_r$ is shifted to the last column of $\boldsymbol{\Phi}_{k+1}$ using a total of $k-r+1$ interchanges between two adjacent previously selected regressors. Thus, a new *regression context* of $\mathbf{A} \in \Re^{(k+1)\times m}$, $\mathbf{b}^{k+2)} \in \Re^m$, $\mathbf{c}^{k+1)} \in \Re^m$, $\mathbf{d}^{k+2)} \in \Re^m$, and $\mathbf{h}^{k+1)} \in \Re^m$ is produced as if $\mathbf{p}_r$ was the last selected regressor in the regression matrix $\boldsymbol{\Phi}_{k+1}$.
4) The criterion $\Delta \overleftarrow{J}_{k+1}(\mathbf{p}_r) < \max_{i=k+2}^{m} \Delta \overrightarrow{J}^{*}_{k+1}(\boldsymbol{\varphi}_i)$ is used to decide whether to remove a regressor from the selected pool or not. If none has to be removed, then set $k = k+1$ and the algorithm moves to Step 4. Otherwise, go to the next step.
5) The regressor $\mathbf{p}_r$ is removed from the selected pool and returned to the candidate pool, i.e., $\boldsymbol{\Phi}_k = [\tilde{\mathbf{p}}_1, \ldots, \tilde{\mathbf{p}}_k]$ and $\boldsymbol{\Psi}_k = [\tilde{\boldsymbol{\varphi}}_{k+1}, \ldots, \tilde{\boldsymbol{\varphi}}_m]$. The *regression context* $\mathbf{A} \in \Re^{k\times m}$, $\mathbf{b}^{k+1)} \in \Re^m$, $\mathbf{c}^{k)} \in \Re^m$, $\mathbf{d}^{k+1)} \in \Re^m$, and $\mathbf{h}^{k)} \in \Re^m$ are then updated and the index $k$ is set to $k-1$ as described in Section IV-B3.

*Step 4)* The learning process will terminate if some stopping criterion is met, such as a certain number of regressors have been selected or some tolerance value has been met. Similar to the stopping criterion commonly used in training neural networks and SVMs [13], [26], the tolerance for the maximum ratio of objective value reduction is used here. In detail, if the ratio $(J_k - \min_{i=k+1}^{m} J_{k+1}(\boldsymbol{\varphi}_i))/J_k$ is less than a very small positive tolerance value ($\rho$), the generalization performance of the fuzzy systems will not be greatly improved by adding a new regressor. It should be noted that the stopping criterion used here is an important measure for the tradeoff between the training accuracy (performance) and the model complexity (sparseness and interpretability) of the obtained fuzzy systems. If the stopping criterion is not met, the algorithm returns to Step 2.

### *E. Convergence and Computational Complexity*

For the convergence, it is obvious that the objective value continuously decreases each time a new regressor is included into the selected pool (i.e., where only the forward expansion phase is applied), with a decrement amount of $\Delta \overrightarrow{J}_{k+1}(\boldsymbol{\varphi}_i)$ at the $(k+1)$th subset selection step if $\boldsymbol{\varphi}_i$ $(i = k+1, \ldots, m)$ is added as defined in (21) and (30). To reassess the contribution of all the previously selected regressors, the backward exclusion phase is performed to exclude the most insignificant regressor with the smallest contribution to the objective function from the selected pool. Thus, the introduction of this backward exclusion phase can cause a small amount of increase $\Delta \overleftarrow{J}_{k+1}(\mathbf{p}_i)$ to the objective value, which is defined in (22) and (35) if a selected regressor, say $\mathbf{p}_i$ $(i = 1, \ldots, k+1)$, is removed from the

TABLE I
OPERATIONS INVOLVED IN THE PROPOSED METHOD

| Algorithms/Operations | | Additions/Subtractions | Multiplications/Divisions | Total Operations |
|---|---|---|---|---|
| | Forward | $M(N^2+N+3)+2N(N-1)$ | $M(N^2+4N+2)-M(M+1)+2N^2$ | $M(2N^2+5N+5)-M(M+1)+2N(2N-1)$ |
| Backward | Constant | $M(N^2+N+2)+M(M+1)/2+2N(N-1)$ | $M(N^2+4N+1)+M(M+1)/2+2N^2$ | $M(2N^2+5N+3)+M(M+1)+2N(2N-1)$ |
| | Shifting | $(k-n_k+1)(2N+7)$ | $(k-n_k+1)(2N+8)$ | $(k-n_k+1)(4N+15)$ |
| | Removing | $2N+2$ | $3N-1$ | $5N+1$ |

selected pool at the $(k+1)$th subset selection step. However, as the criterion $\min_{i=1}^{k}\Delta\overleftarrow{J}_{k+1}(\mathbf{p}_i) < \max_{i=k+1}^{m}\Delta\overrightarrow{J}_{k+1}(\boldsymbol{\varphi}_i)$ is used to determine whether or not a regressor is removed and assuming that the objective value on the $k$th subset selection step at some point is $J_k$, the new objective value $J_k^{*}$ obtained after a forward expansion being followed by a backward exclusion is given by $J_k^{*} = J_k - \max_{i=k+1}^{m}\Delta\overrightarrow{J}_{k+1}(\boldsymbol{\varphi}_i) + \min_{i=1}^{k}\Delta\overleftarrow{J}_{k+1}(\mathbf{p}_i) < J_k$. Thus, the objective value is reduced each time a new subset of $k$ regressors is selected. Obviously, the extreme case is that a nonsparse fuzzy system corresponding to the solution of (13) can be obtained if all the regressors are selected as the SVs with a tolerance value $\rho = 0$. In summary, the convergence of the proposed method composed of iterative forward expansion and backward exclusion phases is guaranteed.

With respect to the computational complexity, the basic arithmetic operations involved in the construction of sparse LS-SVM-based fuzzy systems are additions/subtractions and multiplications/divisions. Assuming that a total of $N$ data samples are used for training and that a total of $M$ rules have been extracted by the proposed learning mechanism, the number of additions/subtractions and multiplications/divisions and overall total of operations from only using the forward expansion phase are listed in the first row of Table I. By introducing the backward exclusion phase, the overall computational complexity then varies with the different numbers of regressors removed at each selection step and the different position of the removed regressor in the selected pool.

The details of the computational complexity, including both the constant part and the variable part (shifting operations and removing operations), are listed in the last three rows of Table I. The first part constant operations involving (33)–(35) and the forward expansion phase are listed in the second row of Table I. Suppose the forward expansion at the $(k+1)$th step is just completed and a previously selected regressor at the $n_k$th position in $\boldsymbol{\Phi}_{k+1}$ is to be removed from current selected pool; then, the operations involved in shifting this regressor to the last position in $\boldsymbol{\Phi}_{k+1}$ and removing it are given in the third and fourth rows of Table I. Due to the fact that $N >> M$, the computation mainly comes from the term $2MN^2$. In practice, the proposed method is usually dominated by the forward expansion phase, while the backward exclusion phase works on revising the selected regression pool. Thus, considering $M > k \geq n_k$, the computational demand of the proposed algorithm does not increase too much, compared with the forward expansion phase. In addition, as described in Section III, it generally needs a computational complexity of $N^3/3 + O(N^2)$ by using the efficient Cholesky decomposition to solve the KKT system (9) only for nonsparse LS-SVMs. Therefore, the computational advantage of our learning mechanism is significant especially when the training dataset consists of a larger number of patterns. If the pruning method [25] discussed in Section I is used for imposing the sparseness for the conventional LS-SVM, its computational complexity can also be extremely large. Thus, the computational demand of the proposed learning mechanism in this paper can be dramatically decreased, meanwhile achieving the model sparseness. These will further be demonstrated in the following experimental examples.

## V. NUMERICAL EXAMPLES

Three simulation and real-world problems are investigated to validate the efficiency and effectiveness of the proposed learning mechanism and the sparseness of LS-SVM-based fuzzy systems constructed. The resulting performances are also compared with other SVM-based fuzzy learning approaches in terms of model sparseness, running time, and model accuracy. The first example is a nonlinear dynamic identification problem [31], the second involves melt pressure prediction in polymer extrusion process [32], and the third is to diagnose the severity of mammographic masses [33]. All the experiments were conducted on an Intel Core$^{\mathrm{TM}}$2 Duo Processor E8135 2.40 GHz, running the Windows 7 operating system, with programs compiled by MATLAB.

### A. Identification of the Nonlinear Dynamic System

The first example [31] involves identifying the following nonlinear dynamic system:

$$y(t) = \frac{y(t-1)}{1.5 + y^2(t-1)} - 0.3y(t-2) + 0.5u(t-1) + \varepsilon(t) \tag{52}$$

where $\varepsilon(t)$ represents a noise sequence [$\varepsilon(t) \sim N(0, 0.01^2)$]. A total of 400 simulated data points were then generated. The first 200 samples of training data were obtained by stimulating the system with a random input signal $u(t)$ uniformly distributed in $[-1, 1]$, while the remaining 200 samples of test data were produced under using a sinusoidal input signal $u(t) = \sin(2\pi t/25)$. Thus, $[u(t-1), y(t-1), y(t-2)]$ and $y(t)$ constituted the input and output variables for the LS-SVM-based fuzzy models to be developed.

The Gaussian width $\sigma$ was set to 3, and the regularization parameter $\mu$ was set to 1/(2 × 1000), as is common. To assess the effectiveness of the proposed algorithm in finding better values

TABLE II
OBJECTIVE VALUES FOUND BY USING DIFFERENT SUBSET SELECTION ALGORITHMS IN EXAMPLE 1

| Subsets/Algorithms | | Forward | Forward+Backward | Forward+Backward+Second-stage | Exhaustive Search* |
|---|---|---|---|---|---|
| $M=1$ | Objective value | 2.4435e+04 | 2.4435e+04 | 2.4435e+04 | 2.4435e+04 |
| | Running time (s) | 6.33e-04 | 6.86e-04 | 7.57e-04 | 1.73e-02 (2.00e+02) |
| $M=3$ | Objective value | 202.9056 | 120.6424 | 120.6424 | 109.9651 |
| | Running time (s) | 9.77e-04 | 3.03e-03 | 3.71e-03 | 1.73e+02 (1.31e+06) |
| $M=5$ | Objective value | 134.4815 | 103.6682 | 103.6682 | $>75.9960$ |
| | Running time (s) | 1.50e-03 | 3.99e-03 | 5.24e-03 | 5.96e+05 (2.54e+09) |
| $M=7$ | Objective value | 104.8279 | 96.9045 | 86.6511 | $>75.9960$ |
| | Running time (s) | 1.82e-03 | 5.52e-03 | 2.62e-02 | 7.86e+08 (2.28e+12) |
| $M=9$ | Objective value | 96.6179 | 87.7518 | 87.7518 | $>75.9960$ |
| | Running time (s) | 2.42e-03 | 1.01e-02 | 1.44e-02 | 6.32e+11 (1.18e+15) |
| $M=11$ | Objective value | 92.0498 | 82.9424 | 82.8894 | $>75.9960$ |
| | Running time (s) | 2.93e-03 | 1.88e-02 | 4.09e-02 | 2.45e+14 (3.88e+17) |
| $M=13$ | Objective value | 87.1243 | 81.8847 | 81.8847 | $>75.9960$ |
| | Running time (s) | 3.40e-03 | 1.93e-02 | 2.80e-02 | 7.09e+16 (8.83e+19) |
| $M=15$ | Objective value | 83.0997 | 80.9598 | 80.8961 | $>75.9960$ |
| | Running time (s) | 4.02e-03 | 1.98e-02 | 6.02e-02 | 1.48e+19 (1.46e+22) |

*1.73e-02 (2.00e+02) denotes running time (all possible combinations). Since it is unable to realize the exhaustive search method when the number of fuzzy rules becomes larger, those objective values with "$>$" represent the one computed by including all the candidate fuzzy rules into the rule base and their running times were estimated based on executing the method for a small number of possible combinations. Similarly afterwards in Tables VI and VIII.

of the objective function, several experiments were carried out, as shown in Table II, given that the same number of fuzzy rules was selected. In the meantime, different sizes of subsets were also tested, and the results are listed in the corresponding rows. The first column lists the values of the objective function together with the running time obtained by only using the forward expansion algorithm, while the results from using a mixture of forward expansion and backward exclusion on each subset selection step are given in the second column. Apparently, as demonstrated in the first row, the two approaches produced the same objectives when only one rule was included in the rule base since this is the global optimum value and the backward exclusion phase was certainly not needed. The superiority of the mixed one over the forward expansion is evident when the selection process continues. It can also be seen that as the selection proceeded, the values of the objective function did not further decrease significantly when a certain number of fuzzy rules had already been selected, which means that the redundant rules with little contribution to the final fuzzy system were later included into the rule base. To further demonstrate the superiority of our proposed algorithm, the idea of the second-stage algorithm proposed in [27], which is used to refine a fixed size subset of regressors, was also applied on the results obtained by the proposed algorithm, as shown in the third column of Table II. Here, the model size was unchanged during the second-stage refinement procedure, and the contribution of each previously selected fuzzy rule in the first stage was reviewed. It is obvious that there was no big improvement after introducing the second-stage optimization, which in turn reflects that the outstanding performance can be achieved by our proposed algorithm. Alternatively, efforts were also made to search for the global optimum results that can theoretically be found by the exhaustive search method. However, it turned out to be unrealistic if the number of fuzzy rules was larger than five in this example due to the huge amount of running time needed. Suppose that seven rules are currently considered to be added into the rule base; there are $200!/(7!(200-7)!) = 2.28\text{e}+12$ possible combinations, approximately needing 24.92 years to find the optimum result (! denotes the factorial operator). If this works on different numbers of fuzzy rules, the running time can be extremely inconceivable. Assuming that all the candidate fuzzy rules are included in the rule base where they collectively produce the minimum objective value, the global value of the objective function produced by the exhaustive search method under each size of subset should be greater than this value (75.9960 in this example). In conclusion, the proposed learning mechanism is able to select a small-size subset of fuzzy rules with acceptable objective value in a short running time.

It is also clear that the changes in the objective values became very small after a certain number of fuzzy rules had been selected; the stopping criterion with a tolerance value 0.02 was then used in this example to terminate the learning. The resultant number of rules, the number of model parameters, the training and test RMSEs (root mean-squared errors), and the running time by performing only the forward expansion phase and both the forward expansion and backward exclusion phases are shown in the first two columns in Table III. Here, the former method found a total of nine rules with a test RMSE of 1.76e-02, while the latter produced better results with eight rules and a smaller RMSE of 1.42e-02 as it is more capable of finding smaller objective values.

For comparison purposes, Table IV also lists the results of various SVM-based trained fuzzy models. The insensitive value used in the SVM-based fuzzy model and the TSFS-SVR was assigned as 0.03. A direct use of the conventional LS-SVM-based learning mechanism to construct a corresponding fuzzy model was also adopted, in which the KKT system defined in (9) was efficiently solved by the Cholesky decomposition as usual. For the TSFS-SVR, the Gaussian width is determined by the aligned clustering algorithm where the initial width, the

TABLE III
RESULTS OF THE PROPOSED LS-SVM-BASED LEARNING MECHANISM IN CONSTRUCTING FUZZY SYSTEMS FOR THE THREE EXAMPLES

| Example | Example 1 | | Example 2 | | Example 3 | |
|---|---|---|---|---|---|---|
| Algorithms | F | F+B | F | F+B | F | F+B |
| # Rules | 9 | 8 | 18 | 15 | 25 | 16 |
| # Parameters | 37 | 33 | 109 | 91 | 151 | 97 |
| Training RMSE/accuracy | 2.00e-02 | 1.94e-02 | 8.44e-02 | 8.25e-02 | 82.80% | 83.00% |
| Test RMSE/accuracy | 1.76e-02 | 1.42e-02 | 2.11e-01 | 2.07e-01 | 84.55% | 84.85% |
| Running time (s) | 2.71e-03 | 9.80e-03 | 5.92e-02 | 1.30e-01 | 5.03e-02 | 1.21e-01 |

* Here, "F" represents the results obtained from only using forward expansion and "F"+"B" is the results from both the forward expansion and backward exclusion.

TABLE IV
COMPARISON RESULTS OF VARIOUS SVM-BASED LEARNING TECHNIQUES IN CONSTRUCTING FUZZY SYSTEMS IN EXAMPLE 1

| Methods | SVM-based | | | LS-SVM-based | | | TSFS-SVR | | | Proposed | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $1/(2\mu)$ | 10 | 100 | 1000 | 10 | 100 | 1000 | 10 | 100 | 1000 | 10 | 100 | 1000 |
| # Rules | 27 | 25 | 16 | 200 | 200 | 200 | 37 | 37 | 37 | 25 | 14 | 8 |
| # Parameters | 110 | 102 | 66 | 801 | 801 | 801 | 186 | 186 | 186 | 101 | 57 | 33 |
| Training RMSE | 1.95e-2 | 1.85e-2 | 1.67e-2 | 2.01e-2 | 1.84e-2 | 1.56e-2 | 4.22e-2 | 4.21e-2 | 4.20e-2 | 2.24e-2 | 1.98e-2 | 1.94e-2 |
| Test RMSE | 1.83e-2 | 2.00e-2 | 1.88e-2 | 1.51e-2 | 1.39e-2 | 1.34e-2 | 2.82e-2 | 3.02e-2 | 3.18e-2 | 1.77e-2 | 1.47e-2 | 1.42e-2 |
| Running time (s) | 0.54 | 0.54 | 0.53 | 0.11 | 0.11 | 0.11 | 1.25 | 1.57 | 2.04 | 1.50e-2 | 1.37e-2 | 9.80e-3 |

TABLE V
FUZZY RULES OBTAINED FROM THE PROPOSED LEARNING APPROACH IN EXAMPLE 1

$R_1$: If $x_1$ is close to 0.3560 and $x_2$ is close to $-0.3775$ and $x_3$ is close to 0.2831, then $y_1$ is close to 1.6518
$R_2$: If $x_1$ is close to $-0.4326$ and $x_2$ is close to $-0.0889$ and $x_3$ is close to $-0.0902$, then $y_2$ is close to $-1.7856$
$R_3$: If $x_1$ is close to $-0.4244$ and $x_2$ is close to $-0.0983$ and $x_3$ is close to $-0.0163$, then $y_3$ is close to $-1.7375$
$R_4$: If $x_1$ is close to 0.2143 and $x_2$ is close to $-0.3717$ and $x_3$ is close to 0.9183, then $y_4$ is close to 1.7617
$R_5$: If $x_1$ is close to $-0.1834$ and $x_2$ is close to 0.5619 and $x_3$ is close to $-0.9942$, then $y_5$ is close to $-2.0003$
$R_6$: If $x_1$ is close to 0.7123 and $x_2$ is close to 0.2143 and $x_3$ is close to $-0.6880$, then $y_6$ is close to 2.1157
$R_7$: If $x_1$ is close to $-0.1218$ and $x_2$ is close to 0.5619 and $x_3$ is close to $-0.8827$, then $y_7$ is close to $-1.9111$
$R_8$: If $x_1$ is close to 0.4892 and $x_2$ is close to $-0.1725$ and $x_3$ is close to 0.2676, then $y_8$ is close to 1.8918

threshold, and the overlap coefficient were set as 0.3, 0.78, and 1.6 separately, and all the remaining parameters used in determining its weighting parameters were the same as in the other SVM-based fuzzy models. As a common strategy for model training [20], [23], [26], different values of regularization were also examined with a suitable Gaussian width as defined before, which gave good generalization performance. It is clear that the LS-SVM-based fuzzy model trained by our proposed learning mechanism required the least amount of running time for all the $\mu$ values, while the test performance was comparable with the other SVM-based techniques. The sparseness of the fuzzy models produced is reflected in the number of model parameters being used, as shown in the third row of Table IV, and fewer model parameters used in a sparser model. As mentioned earlier, the conventional LS-SVM-based learning mechanism used all the training instances as SVs, thus always resulting an extremely complex fuzzy model. While achieving an acceptable model performance, our LS-SVM-based fuzzy model also proved to be capable of producing a significantly sparser solution for all different $\mu$ values as expected. The best result is given by $1/(2\mu) = 1000$ with a total of eight rules obtained. In general, the decrease of $\mu$ can enhance the training accuracy. However, according to the tradeoff between the training accuracy and the regularization defined in the objective function (6), this could cause the overfitting problem. Fortunately, overfitting did not occur here, and the test results are quite acceptable. It should be noted that the sparse fuzzy systems obtained by our method, which consist of fewer fuzzy rules, also help to avoid overfitting. Similar to [20], the nonlinear system in this example is thus represented as a combination of a series expansion of FBFs (assigned by SVs), and this corresponds to a set of fuzzy IF–THEN rules shown in Table V ($x_1$, $x_2$, $x_3$, and $y_i$ denote the three input variables $u(t-1), y(t-1), y(t-2)$ and the output for the $i$th rule, respectively). Since a larger number of system inputs and fuzzy rules are involved in the next two examples, the similar rule representation will not be listed in this paper. Fig. 3 also shows the training and test outputs of the LS-SVM-based fuzzy model learnt by our algorithm. In conclusion, the proposed learning mechanism has shown to be able to produce sparser fuzzy models in terms of rule numbers and parameter numbers, while obtaining comparable model performance within a short period of running time.

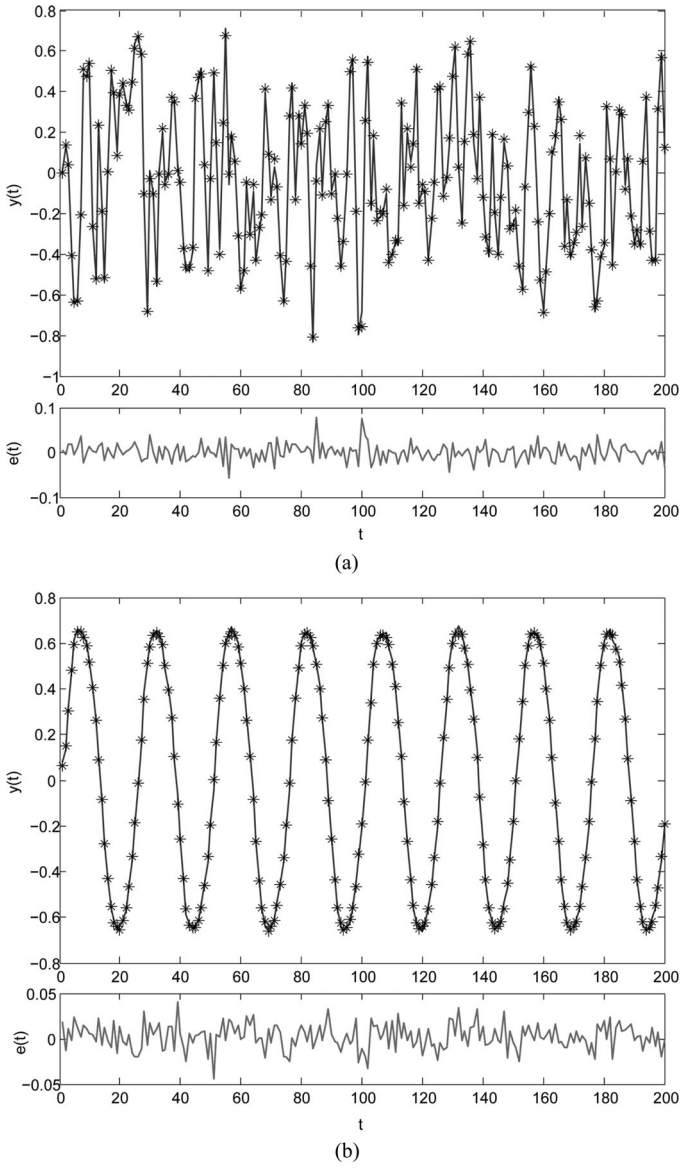



Fig. 3. Fuzzy training and test outputs for $u(t-1), y(t-1), y(t-2)$, and $y(t)$ by using our approach in example 1. (The sign "$*$" denotes the model output, the solid line is the original data, and the bottom curve is the error between the upper two values.) (a) Training output. (b) Test output.

### *B. Melt Pressure Prediction in Polymer Extrusion*

Polymer processing is a major manufacturing sector. An extrusion process is used to melt and then form the raw polymeric materials into continuous profiles [32]. In more detail, polymer material in the form of pellets is first fed into a fixed hopper. The material is then conveyed forward by a rotating screw and discharged through a die before being converted to a continuous polymer product. The product emerging from the die is cooled by blown air or in a water bath. Melt pressure at the end of extruder is one of the most important process parameters in the extruders, and it is closely related to the quality of polymer product produced. It is useful to understand the processing behavior and that the melt pressure is affected by many factors, such as screw speed, motor current, process operating conditions,

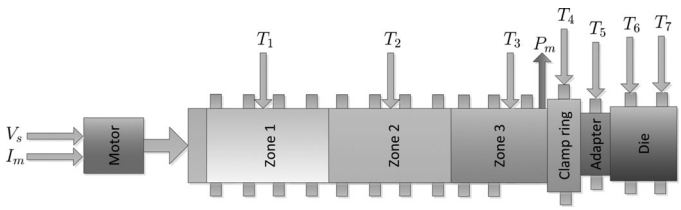



Fig. 4. Signals measured from the single-screw extruder.

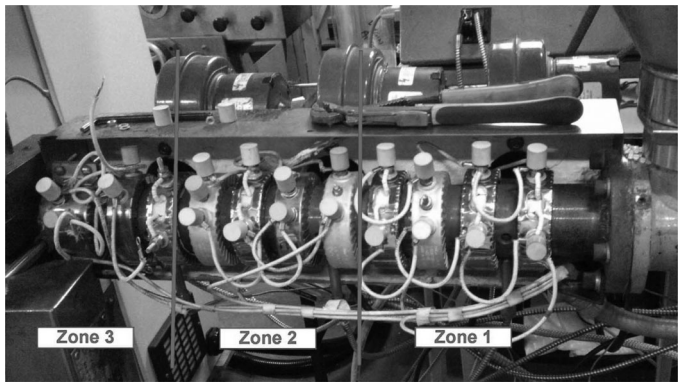



Fig. 5. Heating bands mounted in the three zones.

machine geometry, and material properties. Of these, the prediction of the effects of screw speed, motor current, and process operating conditions on melt pressure is important for a given machine, particular screw geometry, and polymer materials.

The experiment was conducted on a Killion KTS-100 single-screw extruder in the Queen's University of Belfast. A total of seven heaters were located along the barrel, each controlled by a Eurotherm 808 PID temperature controller. The actual location of these heaters is shown in Fig. 4. The temperatures related to the melt pressure are located at zone 1, zone 2, and zone 3, in which four heating bands were mounted in the first two zones and three in the last one, as illustrated in Fig. 5. The experimental trials were conducted using a virgin low-density polyethylene (Dow LD150R, density: 0.921 g/cm$^3$, and MFI: 0.25g/10 min). With a down-sampled frequency of 0.2 Hz, a total of 1154 data points were collected, from which 600 were used for training, the remaining 554 being used as the prediction dataset. These data points were processed by a second-order low-pass digital Butterworth filter with a normalized cutoff frequency of 0.01. The input vector to the melt pressure fuzzy models was set as $[V_s, I_m, T_1, T_2, T_3]$.

In this example, the Gaussian width for all the SVM-based fuzzy models was set to 30 and the regularization parameter involved in the model training processes were the same as that used in Example 1. A series of experiments were conducted to verify the effectiveness and efficiency of the proposed algorithm based on different numbers of fuzzy rules being included in the fuzzy systems. Table VI confirms the superiority of the proposed learning mechanism in terms of finding better objective values compared with the forward expansion algorithm. It can also be found that after some iterations, there were no significant improvements by performing an additional second-stage algorithm, comparing the objective values in the second and

TABLE VI
OBJECTIVE VALUES FOUND BY USING DIFFERENT SUBSET SELECTION ALGORITHMS IN EXAMPLE 2

| Subsets/Algorithms | | Forward | Forward+Backward | Forward+Backward+Second Stage | Exhaustive Search |
|---|---|---|---|---|---|
| $M=1$ | Objective value | 1.2510e+06 | 1.2510e+06 | 1.2510e+06 | 1.2510e+06 |
| | Running time (s) | 5.93e-03 | 5.95e-03 | 5.95e-03 | 5.14e-02 (6.00e+02) |
| $M=3$ | Objective value | 2.3050e+05 | 2.1476e+05 | 2.1476e+05 | 1.0097e+05 |
| | Running time (s) | 1.20e-02 | 1.69e-02 | 1.74e-02 | 5.15e+03 (3.58e+07) |
| $M=5$ | Objective value | 1.8486e+05 | 5.4048e+04 | 1.7402e+04 | $>$3.4383e+03 |
| | Running time (s) | 1.87e-02 | 3.80e-02 | 1.21e-01 | 1.79e+08 (6.37e+11) |
| $M=7$ | Objective value | 6.7776e+04 | 1.0051e+04 | 9.4991e+03 | $>$3.4383e+03 |
| | Running time (s) | 2.55e-02 | 5.42e-02 | 6.80e-02 | 2.15e+12 (5.36e+15) |
| $M=9$ | Objective value | 2.6339e+04 | 6.9254e+03 | 6.0776e+03 | $>$3.4383e+03 |
| | Running time (s) | 3.26e-02 | 6.43e-02 | 1.83e-01 | 1.44e+16 (2.62e+19) |
| $M=11$ | Objective value | 1.1069e+04 | 6.3752e+03 | 5.8740e+03 | $>$3.4383e+03 |
| | Running time (s) | 3.77e-02 | 8.05e-02 | 2.41e-01 | 5.88e+19 (8.29e+22) |
| $M=13$ | Objective value | 6.8825e+03 | 5.5882e+03 | 5.5385e+03 | $>$3.4383e+03 |
| | Running time (s) | 4.53e-02 | 1.13e-01 | 1.98e-01 | 1.73e+23 (1.84e+26) |
| $M=15$ | Objective value | 5.7281e+03 | 5.2928e+03 | 5.2913e+03 | $>$3.4383e+03 |
| | Running time (s) | 5.06e-02 | 1.30e-01 | 1.85e-01 | 3.33e+26 (3.01e+29) |
| $M=17$ | Objective value | 5.4462e+03 | 5.1279e+03 | 5.1120e+03 | $>$3.4383e+03 |
| | Running time (s) | 5.85e-02 | 1.38e-01 | 2.91e-01 | 4.89e+29 (3.79e+32) |

TABLE VII
COMPARISON RESULTS OF VARIOUS SVM-BASED LEARNING TECHNIQUES IN CONSTRUCTING FUZZY SYSTEMS IN EXAMPLE 2

| Methods | SVM-based | | | LS-SVM-based | | | TSFS-SVR | | | Proposed | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $1/(2\mu)$ | 10 | 100 | 1000 | 10 | 100 | 1000 | 10 | 100 | 1000 | 10 | 100 | 1000 |
| # Rules | 32 | 23 | 27 | 600 | 600 | 600 | 513 | 513 | 513 | 21 | 19 | 15 |
| # Parameters | 194 | 140 | 164 | 3601 | 3601 | 3601 | 3592 | 3592 | 3592 | 127 | 115 | 91 |
| Training RMSE | 9.42e-2 | 9.02e-2 | 8.92e-2 | 7.67e-2 | 5.78e-2 | 4.39e-2 | 1.04e-1 | 1.04e-1 | 1.04e-1 | 1.41e-1 | 9.76e-2 | 8.25e-2 |
| Prediction RMSE | 2.24e-1 | 1.93e-1 | 2.71e-1 | 1.94e-1 | 1.93e-1 | 2.79e-1 | 1.76 | 1.76 | 1.76 | 2.81e-1 | 2.11e-1 | 2.07e-1 |
| Running time (s) | 6.68 | 6.27 | 6.38 | 1.26 | 1.25 | 1.27 | 14.30 | 13.76 | 15.14 | 0.17 | 0.17 | 0.13 |

the third columns. This, in turn, means that it is unnecessary to use a second-stage algorithm to refine the fuzzy rules produced by our algorithm. The descending rate of the objective values became negligible as redundant fuzzy rules were then subsequently added in the system after a certain number of fuzzy rules had been selected. Similar to Example 1, the results from the exhaustive search method are also listed in the last column of Table VI. It is impossible to run this algorithm for a long period of 1.06e+19 years if 15 rules are required for constructing a fuzzy system in this case. The stopping tolerance was set to $\rho = 0.02$ in this example. The final results obtained by using the proposed learning mechanism are shown in the middle two columns in Table III, where a small number of fuzzy rules can be obtained by combining the backward exclusion and forward expansion algorithm.

For the comparison purposes, the number of rules, the number of model parameters, the training and prediction errors, and the running time of SVM-based, conventional LS-SVM-based, TSFS-SVR, and our learnt fuzzy models are all listed in Table VII. The insensitive value used in the SVM-based model and TSFS-SVR was assigned as 0.15. For the TSFS-SVR, the initial width, the threshold, and the overlap coefficient in the aligned clustering algorithm were set as 3, 0.97, and 3.8 respectively. The other parameters were the same as that used in Example 1. It is shown that the TSFS-SVR was always incapable of producing good generalization results, and it also had a large number of model parameters. It can be seen that the SVM-based and conventional LS-SVM-based learning mechanisms both generally produced accurate fuzzy models with acceptable prediction RMSEs for all $\mu$ values, while the LS-SVM became extremely complex and the SVM provided less number of fuzzy rules. Apart from the model sparseness issue, this example also confirmed that the LS-SVM-based learning mechanism saved more running time than the SVM-based one. As expected, the proposed approach can further reduce the running time and provide the smallest number of fuzzy rules for all $\mu$ values with comparable generalization performance. The performance of the final sparse LS-SVM-based fuzzy model constructed by the novel learning mechanism on the training and prediction datasets is illustrated in Fig. 6.

### C. Mammographic Masses Diagnosis

Mammography is the most effective method among various breast cancer screening techniques. However, about 70% unnecessary biopsies with benign outcomes are generally performed because of the low positive predictive value of breast biopsy resulting from mammogram interpretation. To reduce the high number of unnecessary breast biopsies, it is important to develop a diagnosis system that can help physicians in their decision to perform a breast biopsy on a suspicious lesion seen in a mammogram image or to perform a short term follow-up examination

TABLE VIII
OBJECTIVE VALUES FOUND BY USING DIFFERENT SUBSET SELECTION ALGORITHMS IN EXAMPLE 3

| Subsets/Algorithms | | Forward | Forward+Backward | Forward+Backward+Second Stage | Exhaustive Search |
|---|---|---|---|---|---|
| $M=1$ | Objective value | 4.5067e+05 | 4.5067e+05 | 4.5067e+05 | 4.5067e+05 |
| | Running time (s) | 3.89e-03 | 4.25e-03 | 4.31e-03 | 4.43e-02 (5.00e+02) |
| $M=3$ | Objective value | 3.8270e+05 | 3.8270e+05 | 3.8270e+05 | 3.1491e+05 |
| | Running time (s) | 7.67e-03 | 1.05e-02 | 1.08e-02 | 2.97e+03 (2.07e+07) |
| $M=5$ | Objective value | 3.5237e+05 | 3.5237e+05 | 3.5237e+05 | $>$2.4105e+05 |
| | Running time (s) | 1.25e-02 | 1.44e-02 | 1.47e-02 | 7.14e+07 (2.55e+11) |
| $M=7$ | Objective value | 3.4708e+05 | 3.3104e+05 | 3.2830e+05 | $>$2.4105e+05 |
| | Running time (s) | 1.57e-02 | 2.52e-02 | 4.61e-02 | 5.81e+11 (1.49e+15) |
| $M=9$ | Objective value | 3.2534e+05 | 3.0456e+05 | 3.0306e+05 | $>$2.4105e+05 |
| | Running time (s) | 2.00e-02 | 3.47e-02 | 6.45e-02 | 2.60e+15 (5.01e+18) |
| $M=11$ | Objective value | 3.1845e+05 | 2.6941e+05 | 2.6486e+05 | $>$2.4105e+05 |
| | Running time (s) | 2.54e-02 | 5.22e-02 | 1.04e-01 | 7.44e+18 (1.10e+22) |
| $M=13$ | Objective value | 3.0093e+05 | 2.6103e+05 | 2.5565e+05 | $>$2.4105e+05 |
| | Running time (s) | 2.84e-02 | 6.91e-02 | 2.66e-01 | 1.45e+22 (1.68e+25) |
| $M=15$ | Objective value | 2.9786e+05 | 2.5587e+05 | 2.5503e+05 | $>$2.4105e+05 |
| | Running time (s) | 3.26e-02 | 1.13e-01 | 2.36e-01 | 1.98e+25 (1.89e+28) |
| $M=17$ | Objective value | 2.8150e+05 | 2.5441e+05 | 2.5390e+05 | $>$2.4105e+05 |
| | Running time (s) | 3.56e-02 | 1.31e-01 | 3.36e-01 | 2.05e+28 (1.63e+31) |

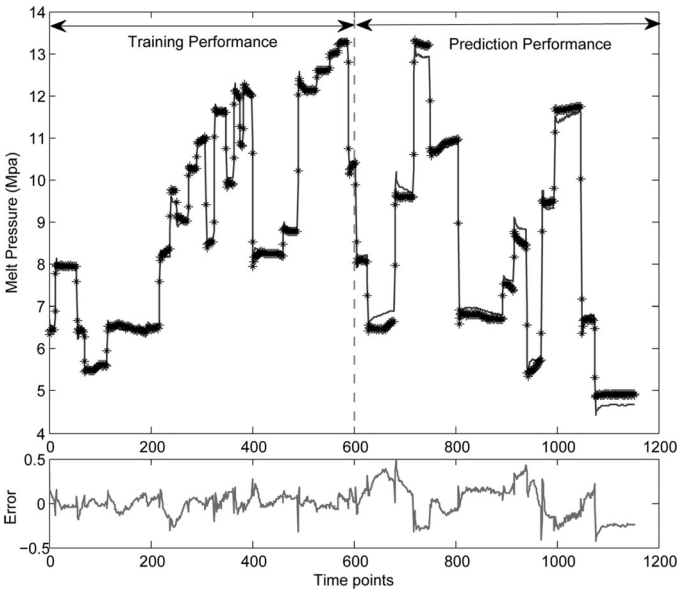



Fig. 6. Training and prediction performances of our LS-SVM-based fuzzy model in melt pressure development. (The sign "$\ast$" denotes the model output, the solid line is the original data, and the bottom curve stands for the corresponding error between the upper two curves.)

[33]. The purpose of this example is to design a fuzzy classifier to increase the ability of physicians in determining the severity (benign or malignant) of a mammographic mass lesion from breast imaging reporting and data system (BI-RADS) assessment, the patient's age, and three BI-RADS attributes (mass shape, mass margin, and mass density). The dataset was obtained from University of California, Irvine Machine Learning Repository, whereas it was collected at the Institute of Radiology of the University Erlangen-Nuremberg between 2003 and 2006, containing a total of 961 instances consisted of the ground truth (the severity field) for 516 benign and 445 malignant masses. After removing 131 instances associated with missing values, a total of 830 instances were finally obtained, from which 500 were used as training samples and 330 as test samples.

The Gaussian width for all the SVM-based fuzzy classifiers was set to 10 in this example, while the remaining parameters were the same as in the previous two examples. Table VIII again demonstrates the superiority of our proposed algorithm in terms of finding better objective values according to the objective function defined in (12). The use of both the backward exclusion and forward expansion algorithm can further decrease the objective value after some number of fuzzy rules had been included in the fuzzy system, while the second-stage refinement algorithm cannot produce significant improvements. As a matter of fact, in some cases, the results performed by our proposed algorithm were very close to the global optimum assuming that all candidate fuzzy rules were selected. The stopping tolerance was set to 0.002, and the final results produced by the proposed learning mechanism are shown in the last two columns in Table III with a small number of fuzzy rules being found by combining both the backward exclusion and forward expansion algorithm. As in the previous two examples, the results produced by our approach are listed in the last three columns of Table IX and compared with that by SVM-based, conventional LS-SVM-based, and TSFS-SVR models. Comparing the SVM-based with the conventional LS-SVM-based ones, both produced good test accuracies for all $\mu$ values, while the former is much sparser along with longer running time and the situation was exactly opposite for the latter one. The TSFS-SVR in this example produced the worst performance on the test data also with a highly complex model. However, for our proposed approach, it is evident that the LS-SVM-based fuzzy classifiers trained by the novel learning mechanism were able to provide the most sparse model together with the least amount of running time while producing comparable test accuracies.

## VI. CONCLUSION

This paper has investigated the construction of fuzzy rule-based systems by building sparse LS-SVMs. To achieve a sparse solution, a new regression solution to the primal optimization problem of LS-SVM has been presented first, which avoids

TABLE IX
COMPARISON RESULTS OF VARIOUS SVM-BASED LEARNING TECHNIQUES IN CONSTRUCTING FUZZY SYSTEMS IN EXAMPLE 3

| Methods | SVM-based | | | LS-SVM-based | | | TSFS-SVR | | | Proposed | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $1/(2\mu)$ | 10 | 100 | 1000 | 10 | 100 | 1000 | 10 | 100 | 1000 | 10 | 100 | 1000 |
| # Rules | 240 | 215 | 205 | 500 | 500 | 500 | 338 | 338 | 338 | 15 | 16 | 16 |
| # Parameters | 1442 | 1292 | 1232 | 3001 | 3001 | 3001 | 2367 | 2367 | 2367 | 91 | 97 | 97 |
| Training accuracy | 84.20% | 84.00% | 86.60% | 83.80% | 85.40% | 87.20% | 94.00% | 94.20% | 94.20% | 81.60% | 82.20% | 83.00% |
| Test accuracy | 84.24% | 84.85% | 80.61% | 83.33% | 81.21% | 80.30% | 71.52% | 71.21% | 71.52% | 83.64% | 84.24% | 84.85% |
| Running time (s) | 2.18 | 3.07 | 5.06 | 0.78 | 0.80 | 0.78 | 1.04e+3 | 961.91 | 1.26e+3 | 0.13 | 0.11 | 0.12 |

solving the KKT system in its conventional solution, which may result in all training patterns that are being used as the SVs. A novel learning mechanism has then been proposed, which efficiently works in a stepwise subset selection approach, consisting of a forward expansion phase and a backward exclusion phase at each selection step. The execution of the algorithm is extraordinarily fast, and a few key techniques have been introduced to avoid inverse operations and to accelerate training process, confirmed with the detailed computational complexity analysis. As a result, a sparse set of SVs for generating the fuzzy IF–THEN rules from the training instances can be obtained easily. Three examples, including a nonlinear dynamic modelling, melt pressure prediction, and mammographic masses diagnosis, have been presented to demonstrate the efficiency and effectiveness of the proposed learning mechanism. The superiorities of the LS-SVM-based fuzzy systems developed by the proposed method over other SVM-based learning techniques, in terms of the model sparseness and the computational demand, have been well demonstrated and verified.

## APPENDIX
## PROOF OF THE TWO THEOREMS

According to the definition of $\mathbf{R}_k$ in (16), it follows that

$$\mathbf{R}_{k+1}([\mathbf{\Phi}_k;+\varphi_\mathbf{i}]) = \mathbf{I} - [\mathbf{\Phi}_k \quad \varphi_\mathbf{i}]([\mathbf{\Phi_k} \quad \varphi_\mathbf{i}]^{\mathrm{T}}[\mathbf{\Phi_k} \quad \varphi_\mathbf{i}]+\mu\mathbf{I})^{-\mathbf{1}} \times [\mathbf{\Phi}_k \quad \varphi_\mathbf{i}]^{\mathrm{T}}. \tag{53}$$

Applying the well-known matrix inverse equality

$$\begin{cases} \mathbf{I} - \mathbf{B}[\mathbf{DB} + \mathbf{C}^{-1}]^{-1}\mathbf{D} = [\mathbf{I} + \mathbf{BCD}]^{-1} \\ [\mathbf{A} + \mathbf{BCD}]^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}[\mathbf{DA}^{-1}\mathbf{B} + \mathbf{C}^{-1}]^{-1}\mathbf{DA}^{-1} \end{cases} \tag{54}$$

(53) turns out to be

$$\begin{aligned} \mathbf{R}_{k+1}([\mathbf{\Phi}_k;+\boldsymbol{\varphi}_i]) &= \left(\mathbf{I} + \frac{1}{\mu}[\mathbf{\Phi}_k \quad \boldsymbol{\varphi}_i][\mathbf{\Phi}_k \quad \boldsymbol{\varphi}_i]^{\mathrm{T}}\right)^{-1} \\ &= \left(\mathbf{I} + \frac{1}{\mu}\mathbf{\Phi}_k\mathbf{\Phi}_k^{\mathrm{T}} + \frac{1}{\mu}\boldsymbol{\varphi}_i\boldsymbol{\varphi}_i^{\mathrm{T}}\right)^{-1} \\ &= \mathbf{R}_k - \frac{\mathbf{R}_k\boldsymbol{\varphi}_i\boldsymbol{\varphi}_i^{\mathrm{T}}\mathbf{R}_k^{\mathrm{T}}}{\boldsymbol{\varphi}_i^{\mathrm{T}}\mathbf{R}_k\boldsymbol{\varphi}_i + \mu} \end{aligned} \tag{55}$$

where $\mathbf{R}_k = (\mathbf{I} + \mathbf{\Phi}_k\mathbf{\Phi}_k^{\mathrm{T}}/\mu)^{-1}$ is obtained by using (16) and (54). Thus, Theorem 1 has been proved. In addition, it follows from (16) that

$$\begin{aligned} &\mathbf{R}_k([\mathbf{\Phi}_{k+1};-\mathbf{p}_i]) \\ &\quad= \mathbf{I} - [\mathbf{\Phi}_{k+1};-\mathbf{p}_i]([\mathbf{\Phi}_{k+1};-\mathbf{p}_i]^{\mathrm{T}}[\mathbf{\Phi}_{k+1};-\mathbf{p}_i] + \mu\mathbf{I})^{-1} \\ &\qquad\times [\mathbf{\Phi}_{k+1};-\mathbf{p}_i]^{\mathrm{T}} \\ &\quad= \left(\mathbf{I} + \frac{1}{\mu}[\mathbf{\Phi}_{k+1};-\mathbf{p}_i][\mathbf{\Phi}_{k+1};-\mathbf{p}_i]^{\mathrm{T}}\right)^{-1} \\ &\quad= \left(\mathbf{I} + \frac{1}{\mu}[\mathbf{\Phi}_{k+1};-\mathbf{p}_i][\mathbf{\Phi}_{k+1};-\mathbf{p}_i]^{\mathrm{T}} + \frac{1}{\mu}\mathbf{p}_i\mathbf{p}_i^{\mathrm{T}} - \frac{1}{\mu}\mathbf{p}_i\mathbf{p}_i^{\mathrm{T}}\right)^{-1} \\ &\quad= \left(\mathbf{I} + \frac{1}{\mu}\mathbf{\Phi}_{k+1}\mathbf{\Phi}_{k+1}^{\mathrm{T}} - \frac{1}{\mu}\mathbf{p}_i\mathbf{p}_i^{\mathrm{T}}\right)^{-1} \\ &\quad= \mathbf{R}_{k+1} + \frac{\mathbf{R}_{k+1}\mathbf{p}_i\mathbf{p}_i^{\mathrm{T}}\mathbf{R}_{k+1}^{\mathrm{T}}}{\mu - \mathbf{p}_i^{\mathrm{T}}\mathbf{R}_{k+1}\mathbf{p}_i} \end{aligned} \tag{56}$$

where $\mathbf{R}_{k+1} = (\mathbf{I} + \mathbf{\Phi}_{k+1}\mathbf{\Phi}_{k+1}^{\mathrm{T}}/\mu)^{-1}$. Thus, Theorem 2 has been proved as well.

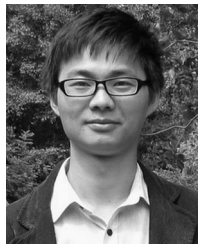
## ACKNOWLEDGMENT

The authors would like to thank the editors, the anonymous reviewers, and Prof. G. W. Irwin for their most constructive comments and suggestions to improve the quality of this paper.

## REFERENCES

[1] B. Pizzileo, K. Li, G. Irwin, and W. Zhao, "Improved structure optimization for fuzzy-neural networks," *IEEE Trans. Fuzzy Syst.*, vol. 20, no. 6, pp. 1076–1089, Dec. 2012.
[2] W. Zhao, K. Li, and G. Irwin, "A new gradient descent approach for local learning of fuzzy neural models," *IEEE Trans. Fuzzy Syst.*, vol. 21, no. 1, pp. 30–44, Feb. 2013.
[3] J.-Q. Wang and H.-Y. Zhang, "Multicriteria decision-making approach based on Atanassov's intuitionistic fuzzy sets with incomplete certain information on weights," *IEEE Trans. Fuzzy Syst.*, vol. 21, no. 3, pp. 510–515, Jun. 2013.
[4] J. Sanz, A. Fernandez, H. Bustince, and F. Herrera, "IVTURS: A linguistic fuzzy rule-based classification system based on a new interval-valued fuzzy reasoning method with tuning and rule selection," *IEEE Trans. Fuzzy Syst.*, vol. 21, no. 3, pp. 399–411, Jun. 2013.
[5] M. Z. Wei, B. J. Bai, A. H. Sung, Q. Liu, J. Wang, and M. E. Cather, "Predicting injection profiles using ANFIS," *Inf. Sci.*, vol. 177, no. 20, pp. 4445–4461, Oct. 2007.
[6] A. Gómez-Skarmeta, M. Delgado, and M. Vila, "About the use of fuzzy clustering techniques for fuzzy model identification," *Fuzzy Sets Syst.*, vol. 106, no. 2, pp. 179–188, Sep. 1999.
[7] N. K. Kasabov and Q. Song, "DENFIS: Dynamic evolving neural-fuzzy inference system and its application for time-series prediction," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 2, pp. 144–154, Apr. 2002.
[8] A. Esfahanipour and W. Aghamiri, "Adapted neuro-fuzzy inference system on indirect approach TSK fuzzy rule base for stock market analysis," *Expert Syst. Appl.*, vol. 37, no. 7, pp. 4742–4748, Jul. 2010.

[9] Y. Yam, P. Baranyi, and C.-T. Yang, "Reduction of fuzzy rule base via singular value decomposition," *IEEE Trans. Fuzzy Syst.*, vol. 7, no. 2, pp. 120–132, Apr. 1999.
[10] J. Yen and L. Wang, "Simplifying fuzzy rule-based models using orthogonal transformation methods," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 29, no. 1, pp. 13–24, Feb. 1999.
[11] M. Setnes and R. Babuska, "Rule base reduction: Some comments on the use of orthogonal transforms," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 31, no. 2, pp. 199–206, May 2001.
[12] L. X. Wang and J. M. Mendel, "Fuzzy basis functions, universal approximation, and orthogonal least-squares learning," *IEEE Trans. Neural Netw.*, vol. 3, no. 5, pp. 807–814, Sep. 1992.
[13] X. Hong, C. J. Harris, and S. Chen, "Robust neurofuzzy rule base knowledge extraction and estimation using subspace decomposition combined with regularization and D-optimality," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 1, pp. 598–608, Feb. 2004.
[14] K. Li, J. X. Peng, and G. W. Irwin, "A fast nonlinear model identification method," *IEEE Trans. Autom. Control*, vol. 50, no. 8, pp. 1211–1216, Aug. 2005.
[15] Z. Johanyak, and S. Kovacs, "Sparse fuzzy system generation by rule base extension," in *Proc. 11th Int. Conf. Intell. Eng. Syst.*, Jun. 2007, pp. 99–104.
[16] C.-J. Lin, C.-H. Chen, and C.-T. Lin, "A hybrid of cooperative particle swarm optimization and cultural algorithm for neural fuzzy networks and its prediction applications," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 39, no. 1, pp. 55–68, Jan. 2009.
[17] C. F. Juang, C. M. Hsiao, and C. H. Hsu, "Hierarchical cluster-based multispecies particle-swarm optimization for fuzzy-system optimization," *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 1, pp. 14–26, Feb. 2010.
[18] M. Fazzolari, R. Alcala, Y. Nojima, H. Ishibuchi, and F. Herrera, "A review of the application of multiobjective evolutionary fuzzy systems: Current status and further directions," *IEEE Trans. Fuzzy Syst.*, vol. 21, no. 1, pp. 45–65, Feb. 2013.
[19] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995.
[20] J. H. Chiang and P. Y. Hao, "Support vector learning mechanism for fuzzy rule-based modeling: A new approach," *IEEE Trans. Fuzzy Syst.*, vol. 12, no. 1, pp. 1–12, Feb. 2004.
[21] R. Batuwita and V. Palade, "FSVM-CIL: Fuzzy support vector machines for class imbalance learning," *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 3, pp. 558–571, Jun. 2010.
[22] C. F. Juang and C. D. Hsieh, "A locally recurrent fuzzy neural network with support vector regression for dynamic-system modeling," *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 2, pp. 261–273, Apr. 2010.
[23] C. F. Juang and C. D. Hsieh, "TS-fuzzy system-based support vector regression," *Fuzzy Sets Syst.*, vol. 160, no. 17, pp. 2486–2504, Sep. 2009.
[24] J. A. K. Suykens, L. Lukas, and J. Vandewalle, "Sparse approximation using least squares support vector machines," in *Proc. IEEE Int. Symp. Circuits Syst.*, May. 2000, vol. 2, pp. 757–760.
[25] J. Valyon and G. Horváth, "A sparse least squares support vector machine classifier," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Jul. 2004, vol. 1, pp. 543–548.
[26] L. Jiao, L. Bo, and L. Wang, "Fast sparse approximation for least squares support vector machine," *IEEE Trans. Neural Netw.*, vol. 18, no. 3, pp. 685–697, May 2007.
[27] K. Li, J.-X. Peng, and E.-W. Bai, "A two-stage algorithm for identification of nonlinear dynamic systems," *Automatica*, vol. 42, no. 7, pp. 1189–1197, Jul. 2006.
[28] M. Adankon, M. Cheriet, and A. Biem, "Semisupervised learning using Bayesian interpretation: Application to LS-SVM," *IEEE Trans. Neural Netw.*, vol. 22, no. 4, pp. 513–524, Apr. 2011.
[29] J. Zhang, K. Li, G. W. Irwin, and W. Zhao, "A regression approach to LS-SVM and sparse realization based on fast subset selection," in *Proc. 10th Word Congr. Intell. Contr. Autom.*, Jul. 2012, pp. 612–617.
[30] B. Pizzileo, K. Li, and G. W. Irwin, "A fast method for fuzzy neural modelling and refinement," *Int. J. Modell., Identif. Control*, vol. 8, no. 3, pp. 175–183, Jan. 2009.
[31] Y. H. Chen, B. Yang, A. Abraham, and L. Z. Peng, "Automatic design of hierarchical Takagi-Sugeno type fuzzy systems using evolutionary algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 15, no. 3, pp. 385–397, Jun. 2007.
[32] C. Abeykoon, P. J. Martin, A. L. Kelly, and E. C. Brown, "A review and evaluation of melt temperature sensors for polymer extrusion," *Sens. Actuators A, Phys.*, vol. 182, pp. 16–27, Aug. 2012.
[33] K. Bache and M. Lichman. (2013). UCI machine learning repository. [Online]. Available: http://archive.ics.uci.edu/ml



**Wanqing Zhao** (M'13) received the B.Eng. degree in automation from Anhui Polytechnic University, Anhui, China, in 2006; the M.Eng. degree in control theory and control engineering from Shanghai University, Shanghai, China, in 2009; and the Ph.D. degree from the Intelligent Systems and Control Group, Queen's University Belfast, Belfast, U.K., in 2012.

He is currently a Research Fellow with the School of Engineering, Cardiff University, Cardiff, U.K., and he previously held the position of Research Associate with the Department of Computer Science, Loughborough University, Loughborough, U.K. His research interests include system identification, fuzzy regression, neural networks, machine learning, autonomous system, and heuristic optimization methods.

**Jingjing Zhang** received the B.Eng. degree in automation from the Beijing Institute of Technology, Beijing, China, in 2006; the M.Eng. degree in software engineering from Tsinghua University, Beijing, in 2009; and the Ph.D. degree from the Intelligent Systems and Control Group, Queen's University Belfast, Belfast, U.K., in 2012.

She is currently a Research Associate with the School of Engineering, Cardiff University, Cardiff, U.K. She previously held a research position with the School of Computing, the University of Dundee, Dundee, U.K. Her research interests include support vector machines, pattern recognition, machine learning, computer vision, medical image processing, and heuristic optimization methods.

**Kang Li** (M'05–SM'11) received the B.Sc. degree from Xiangtan University, Hunan, China, in 1989; the M.Sc. degree from the Harbin Institute of Technology, Harbin, China, in 1992; and the Ph.D. degree from Shanghai Jiaotong University, Shanghai, China, in 1995.

He is currently a Professor of intelligent systems and control with the School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, Belfast, U.K. He has published more than 200 papers in his areas of expertise and edited 12 conference proceedings (Springer). His current research interests include nonlinear system modeling, identification and control, bio-inspired computational intelligence, and fault-diagnosis and detection, with applications on the development of a new generation of low-cost intelligent energy saving technologies for decarbonizing the whole energy systems, from conventional thermal power plants, integration of renewable energies, to end user energy efficiency in plastics industry and integration of electric vehicles to the grid. His research interest also include bioinformatics with applications on food safety, healthcare, and biomedical engineering.

Dr. Li serves on the Editorial Boards of *Neurocomputing*; the *Transactions of the Institute of Measurement & Control*; the *International Journal of Modelling, Identification and Control*; and *Cognitive Computation*.