

Splitting K-means Generated Neural Fuzzy System with Support Vector Regression

Cheng-Da Hsieh
Department of Electrical Engineering,
Hsiuping Institute of Technology
Taichung, Taiwan, R.O.C
Chengda.hsieh@msa.hinet.net

Chia-Feng Juang
Department of Electrical Engineering,
National Chung-Hsing University
Taichung, Taiwan, R.O.C
cfjuang@dragon.nchu.edu.tw

Abstract—This paper proposes a Splitting K-means generated Neural Fuzzy System with Support Vector Regression (SKNFS-SVR). The consequent layer in SKNFS-SVR is a Takagi-Sugeno-Kang (TSK)-type consequent. For structure learning, a splitting K-means algorithm clusters the input training data and determines the rule number. For parameter learning, a linear support vector regression (SVR) algorithm is proposed to tune free parameters in the consequent part. The motivation for using SVR for parameter learning is to improve the SKNFS-SVR generalization ability. This paper demonstrates the capabilities of SKNFS-SVR by conducting simulations in clean and noisy function approximations. This paper also compares simulation results from the SKNFS-SVR with Gaussian kernel-based SVR.

Keywords—Fuzzy modeling, splitting K-means, support vector regression, support vectors, function approximation, fuzzy neural network.

I. INTRODUCTION

Fuzzy systems (FSs) and support vector regressions (SVRs) have been applied to solve the regression estimation and signal prediction problems in these years. FS is a well-known as a powerful tool for modeling nonlinear functions. SVR learning formulation [1, 2] is based on the principle of structural risk minimization and attempts to minimize a bound on the generalization error. The SVR learning approach also exhibits desirable properties such as generalization and over-fitting prevention capabilities.

Researchers have proposed several different neural learning techniques for constructing FSs [3]-[8], such as the self-constructing neural fuzzy inference network (SONFIN) [3] and the self-organizing fuzzy neural network (SOFNN) [7], and the Fuzzified Takagi-Sugeno-Kang (TSK)-type Neural Fuzzy Inference Network (FTNFIN) [8]. However, these neural fuzzy systems (NFS) are based on the minimization of training error (or empirical risk), they do not possess high generalization ability. For noisy regression problems, training error minimization may overtrain the noise into the learning model, leading to an unsatisfied test result. To address this problem, combination of fuzzy set theory and SVR has been proposed for regression problems [9]-[11]. In [9], the authors applied FCM clustering algorithm to obtain fuzzy rule that transforms input data to feature space. Then using SVR solves the weighting parameters. The authors in [10] introduced fuzzy SVM approach to seek a fuzzy linear function that model weights and bias term are fuzzy sets. The authors in [11]

proposed a TSK FS-based SVR (TSFS-SVR) that combines the idea of TSK-type fuzzy system and SVR. The function of TSFS-SVR is a linear combination of TS-type kernel functions, where kernel and weight parameters are learned by clustering and SVR. However, there is only one free consequent parameter in each rule of the TS-kernel-based FS due to the constraint of a TS-kernel. This paper proposes an SVR-FS, whose rules are also TSK type. Different from the study in [11], all of the free consequent parameters in each TSK-type fuzzy rule are tunable.

This paper proposes a Splitting K-means generated Neural Fuzzy System with Support Vector Regression (SKNFS-SVR). The main purpose of the SKNFS-SVR is to introduce a new regression model utilizing FS and SVR. The SKNFS-SVR introduces a splitting K-means algorithm to cluster the input training data and determine the number of rules. We applied SVR to tune the consequent parameters in order to obtain a global regression model with good test performance. Representative simulation results show that the approach achieves better performance over the Gaussian kernel based SVR.

This paper is organized as follows. Section II introduces the SKNFS-SVR structure. Section III introduces splitting K-means algorithm and SKNFS-SVR structure learning. Section IV briefly reviews the basic theory of SVR and introduces SKNFS-SVR parameter learning using a linear SVR learning approach. Section V shows some representative simulations of function approximation. Section VI draws conclusions.

II. SKNFS-SVR STRUCTURE

This section introduces the structure of the SKNFS-SVR. SKNFS-SVR has a TSK-type consequent. The output is a linear combination of network inputs, i.e., $\sum_{j=1}^n a_{ij}x^j + a_{i0}$, where a_{ij} are the consequent part parameters of rule i . For input variable x^j , the following Gaussian membership function is used

$$M_{ij}(x^j) = \exp \left\{ -\frac{(x^j - m_{ij})^2}{\sigma_{ij}^2} \right\}, \quad i = 1, \dots, r \text{ and } j = 1, \dots, n \quad (1)$$

where m_{ij} and σ_{ij} denote the center and width of the Gaussian membership function, respectively, and r is the number of

rules. Using an algebraic product operation to obtain a spatial firing strength $u^i(\bar{x})$ that is determined by

$$u^i(\bar{x}) = \prod_{j=1}^n M_{ij}(\bar{x}^j) = \exp \left\{ -\frac{\|\bar{x} - \bar{m}_i\|^2}{\sigma_{ij}^2} \right\}. \quad (2)$$

Based on the simple weighted sum, the system output can be written as

$$y = \sum_{i=1}^r [u^i(\bar{x}) \cdot (\sum_{j=1}^n a_{ij} x^j + a_{i0})] + b \quad (3)$$

where b is a bias to account for the learning of a linear SVR (see (11)).

III. SKNFS-SVR STRUCTURE LEARNING

The primary task of SKNFS-SVR structure learning is determining the fuzzy rule number r . To generate a suitable number of clusters, a self-splitting K -means algorithm is proposed. In this algorithm, we first calculate the mean of all samples then the variance $\bar{\sigma}^2$ of a cluster is used as the criterion to decide whether or not a cluster should be split into two. The variance of a cluster i is defined as

$$\bar{\sigma}_i^2 = \sum_{j=1}^n \sigma_{ij}^2 \quad (4)$$

where σ_{ij} is the variance of the cluster in the j th input dimension, i.e.,

$$\sigma_{ij}^2 = \frac{1}{N_i} \sum_{\bar{x} \in C_i} (x^j - m_{ij})^2 \quad (5)$$

where $\bar{x} \in C_i$ denotes the samples belonging to cluster i , m_{ij} is the mean of samples and N_i is the number of samples in the cluster. For the existing r clusters, the algorithm finds

$$I = \arg \max_{1 \leq i \leq r} \sigma_i^2 \quad (6)$$

where σ_i^2 is the variance of clusters i . If σ_i^2 is larger than a predefined threshold V_{th} , then cluster I is split into two clusters, resulting in a total of $r+1$ clusters. Centers of the two new clusters are initially set as $\bar{m}_I + \bar{\epsilon}$ and $\bar{m}_I - \bar{\epsilon}$, where $\bar{m}_I = (m_1^I, \dots, m_n^I)$ is the center I and $\bar{\epsilon}$ is a small value. Then, new centers of these $r+1$ clusters are recomputed by splitting K -means algorithm. The whole process recurs until the maximum variance is less than V_{th} . The whole algorithm is summarized as follows:

Begin Algorithm

- **Initialization.** Set cluster number $r=1$. Find $\bar{m}_1$.
- **Step 1.** Calculate (5) and (6).

- **Step 2.** If $(\sigma_i^2 < V_{th})$

{Stop}

Else

{ Split cluster I into two new clusters with centers $\bar{m}_I + \bar{\epsilon}$ and $\bar{m}_I - \bar{\epsilon}$, Set $r \leftarrow r+1$ }

- **Step 3.** Find centers of these r clusters by K -means clustering
Go to Step 1.

End Algorithm

IV. SKNFS-SVR PARAMETER LEARNING

A. Basic SVR Concepts

This section briefly reviews the theory of SVR [12]. Suppose there is a labeled training set

$$S = \{(\bar{x}_1, y_1), (\bar{x}_2, y_2), \dots, (\bar{x}_N, y_N)\} \quad (7)$$

where $\bar{x}_k \in \mathbb{R}^n$ and $y_k \in \mathbb{R}$, $k=1, \dots, N$. The main goal of $\mathcal{E}$ -SVR regression is to find a function $f(\bar{x})$ that has no more than $\mathcal{E}$ deviation from the actually obtained targets y_k for all the training data, and is as close as possible at the same time. First, consider the case of linear function $y'(\bar{x})$, taking the form

$$y'(\bar{x}) = \bar{w}^T \bar{x} + b \quad (8)$$

where $\bar{w} \in \mathbb{R}^n$ is a high dimensional weight vector and the bias $b \in \mathbb{R}$ is a scalar.

The convex optimization problem is feasible when a function $y'(\bar{x})$ exists that approximates all pairs $(\bar{x}_k, y_k)$ with $\mathcal{E}$ precision in the above constraint. If the problem is infeasible or some errors are allowed, then the problem can be defined as

$$\begin{aligned} \text{Min}_{\bar{w}, \xi_k, \hat{\xi}_k} \quad & \frac{1}{2} \bar{w}^T \bar{w} + C \sum_{k=1}^N (\xi_k + \hat{\xi}_k) \\ \text{Subject to} \quad & y_k - (\bar{w}^T \bar{x}_k + b) \leq \mathcal{E} + \hat{\xi}_k \\ & \bar{w}^T \bar{x}_k + b - y_k \leq \mathcal{E} + \xi_k \\ & \hat{\xi}_k, \xi_k \geq 0, k=1, 2, \dots, N \end{aligned} \quad (9)$$

This formulation introduces two slack variables, $\hat{\xi}$ and ξ : one for exceeding the target value by more than $\mathcal{E}$, and the other for being more than $\mathcal{E}$ below the target. The constant $C > 0$ is a user regulated positive parameter that controls the trade-off between the error and margin.

By applying the Karush-Kuhn-Tucker theorem, the constrained optimization problem above can be solved by defining a Lagrangian L [8]:

$$L(\bar{w}, b, \xi_k, \hat{\xi}_k, \alpha_k, \hat{\alpha}_k) = \frac{1}{2} \bar{w}^T \bar{w} + C \sum_{k=1}^N (\xi_k + \hat{\xi}_k) - \sum_{k=1}^N (\eta_k \xi_k + \hat{\eta}_k \hat{\xi}_k) \quad (10)$$

$$- \sum_{k=1}^N \alpha_k (\varepsilon + \xi_k - y_k + \bar{w}^T \bar{x}_k + b)$$

$$- \sum_{k=1}^N \hat{\alpha}_k (\varepsilon + \hat{\xi}_k + y_k - \bar{w}^T \bar{x}_k - b)$$

where $\eta_k, \hat{\eta}_k, \alpha_k$ and $\hat{\alpha}_k$ are Lagrange multipliers. Minimizing $L(\bar{w}, b, \xi_k, \hat{\xi}_k, \alpha_k, \hat{\alpha}_k)$ requires that the gradient of $L(\bar{w}, b, \xi_k, \hat{\xi}_k, \alpha_k, \hat{\alpha}_k)$ with respect to $\bar{w}$, b , ξ_k and $\hat{\xi}_k$ must vanish, i.e.,

$$\frac{\partial L(\bar{w}, b, \xi_k, \hat{\xi}_k, \alpha_k, \hat{\alpha}_k)}{\partial \bar{w}} = \bar{w} - \sum_{k=1}^N (\alpha_k - \hat{\alpha}_k) \bar{x}_k = 0 \quad (11)$$

$$\Rightarrow \bar{w} = \sum_{k=1}^N (\alpha_k - \hat{\alpha}_k) \bar{x}_k$$

$$\frac{\partial L(\bar{w}, b, \xi_k, \hat{\xi}_k, \alpha_k, \hat{\alpha}_k)}{\partial b} = \sum_{k=1}^N (\hat{\alpha}_k - \alpha_k) = 0 \quad (12)$$

$$\frac{\partial L(\bar{w}, b, \xi_k, \hat{\xi}_k, \alpha_k, \hat{\alpha}_k)}{\partial \xi_k} = C - \alpha_k - \eta_k = 0 \Rightarrow \eta_k = C - \alpha_k \quad (13)$$

$$\frac{\partial L(\bar{w}, b, \xi_k, \hat{\xi}_k, \alpha_k, \hat{\alpha}_k)}{\partial \hat{\xi}_k} = C - \hat{\alpha}_k - \hat{\eta}_k = 0 \Rightarrow \hat{\eta}_k = C - \hat{\alpha}_k \quad (14)$$

Thus, the primal problem (9) can be converted into the following equivalent dual problem,

$$\text{Max}_{\alpha_k} L(\alpha_k) = -\frac{1}{2} \sum_{k,j=1}^N (\alpha_k - \hat{\alpha}_k)(\alpha_j - \hat{\alpha}_j) \bar{x}_k \cdot \bar{x}_j$$

$$- \varepsilon \sum_{k=1}^N (\alpha_k + \hat{\alpha}_k) + \sum_{k=1}^N y_k (\alpha_k - \hat{\alpha}_k) \quad (15)$$

$$\text{Subject to } \sum_{k=1}^N (\alpha_k - \hat{\alpha}_k) = 0, \quad \alpha_k, \hat{\alpha}_k \in [0, C]$$

In dual problem (15), the dual variables η_k and $\hat{\eta}_k$ are eliminated through conditions (13) and (14). By substituting (11) into (8), one has

$$y'(\bar{x}) = \sum_{k=1}^N (\alpha_k - \hat{\alpha}_k) \bar{x}_k \cdot \bar{x} + b \quad (16)$$

where the training samples for which $(\alpha_k - \hat{\alpha}_k) \neq 0$ are called support vectors (SVs).

The above linear SVR is employed to learn free parameters in a SKNFS-SVR. The advantage of using SVR instead of neural learning is that SVR is based on structural risk minimization (or minimizing the test error), so it can improve the SKNFS-SVR generalization ability.

B. SKNFS-SVR Parameter Learning

The objective of parameter learning is to tune the parameter a_{ij} of the constructed SKNFS-SVR to an optimal extent. Learning of parameter a_{ij} using a linear SVR algorithm is introduced as follows.

The output function of SKNFS-SVR in (3) can be written as follows

$$y = \sum_{i=1}^r u^i \cdot (\sum_{j=0}^n a_{ij} x^j) + b = \sum_{i=1}^r \sum_{j=0}^n a_{ij} u^i x^j + b \quad (17)$$

where $x_0(k) \triangleq 1$. Eq. (17) shows that the output y is a linear function of $u^i x^j$ with weights a_{ij} . Therefore, linear SVR can be employed to learn parameters a_{ij} . After structure learning, the number of rules r and rule antecedent part parameters are determined. Each input data $\bar{x}_k$ at time step k is transformed to the following vector

$$\bar{\phi}(\bar{x}_k) = [u_k^1 x^0, \dots, u_k^1 x^n, u_k^2 x^0, \dots, u_k^2 x^n, \dots, u_k^r x^0, \dots, u_k^r x^n] \in \mathbb{R}^{r(n+1)} \quad (18)$$

The vector $\bar{\phi}$ is fed as input to a linear SVR, and the training data pairs in (8) are represented by

$$S = \{(\bar{\phi}(\bar{x}_1), y_1), (\bar{\phi}(\bar{x}_2), y_2), \dots, (\bar{\phi}(\bar{x}_N), y_N)\} \quad (19)$$

According to (17), the optimal linear regression function is

$$y(\bar{x}) = \sum_{k=1}^N (\alpha_k - \hat{\alpha}_k) \langle \bar{\phi}(\bar{x}), \bar{\phi}(\bar{x}_k) \rangle + b \quad (20)$$

where α_k and $\hat{\alpha}_k$ are solved by the linear SVR in Subsection IV-A. Eq. (20) can be represented as follows

$$y(\bar{x}) = \sum_{k=1}^N (\alpha_k - \hat{\alpha}_k) \sum_{m=1}^{r(n+1)} \phi^m(\bar{x}) \phi^m(\bar{x}_k) + b$$

$$= \sum_{m=1}^{r(n+1)} [\sum_{k=1}^N (\alpha_k - \hat{\alpha}_k) \phi^m(\bar{x}_k)] \phi^m(\bar{x}) + b \quad (21)$$

$$= \sum_{i=1}^r \sum_{j=1}^n [\sum_{k=1}^N (\alpha_k - \hat{\alpha}_k) u_k^i x_k^j] u_k^i x_k^j + b$$

$$= \sum_{i=1}^r \sum_{j=0}^n a_{ij} u_k^i x^j + b$$

where

$$a_{ij} = \sum_{k=1}^N (\alpha_k - \hat{\alpha}_k) u_k^i x_k^j, \quad i = 1, \dots, r \quad \text{and} \quad j = 0, \dots, n \quad (22)$$

Eq. (21) is equivalent to the SKNFS-SVR output function in Eq. (16). That is, Equation (22) solves parameters a_{ij} in a SKNFS-SVR.

V. SIMULATION RESULTS

This section uses computer simulations to illustrate the performance and capabilities of the proposed SKNFS-SVR in a function approximation problem. First, training data without noise was used to examine SKNFS-SVR performance. Training data with noise was then used to evaluate the robustness of the SKNFS-SVR. To determine the cost parameter C in a Gaussian kernel-based SVR and a linear SVR, different values ($C = 10^{-3}, 10^{-2}, \dots, 10^3$) were simulated, and only the ones that achieved relatively good performance were shown in the example.

Example 1. The approximated function is defined as follows:

$$f(x_1, x_2) = \frac{(5 - x_2)^2}{3(5 - x_1)^2 + (5 - x_2)^2}, x_1, x_2 \in [0, 10] \quad (23)$$

A clean training set containing 200 samples was randomly generated from the input domain. The testing set had 200 points randomly generated by the same way. For SKNFS-SVR training, the parameters in the clustering algorithm were set to $V_{th}=0.024$, and 9 clusters were generated. For the linear SVR for weight training, different insensitive values ($\varepsilon=0.07$, and 0.02) and cost parameters C were studied to determine their influence on performance. Table 1 shows the performance of SKNFS-SVR for the sets of parameters that achieve relatively good performance. The results show that with the same SKNFS-SVR structure, a smaller ε value generates smaller test error. The parameter set $(\varepsilon, C)=(0.02, 10)$ achieves the minimum RMSE of 0.0641724.

A Gaussian kernel-based SVR was designed using the same set of training data. For Gaussian kernel-based SVR, different insensitive values ($\varepsilon=0.07$, and 0.02), cost parameters, C , and kernel width parameters, g , were studied. Table 1 shows the sets of parameters that achieve relatively good performance. The results indicate the parameter set $(\varepsilon, g, C)=(0.02, 20, 10)$ achieves the minimum test RMSE of 0.0774.

Example 2. To compare robustness, a noisy training set was generated by adding white Gaussian noise with mean zero and standard deviation 0.05 to the output of the clean training set. The original clean test set was also used for testing. The SKNFS-SVR contains 9 clusters when $V_{th}=0.024$ as same as using the training data. Table 2 shows the test RMSE, and the total number of parameters in SKNFS-SVR with different ε and ε values, where it can be found that a minimum test RMSE of 0.0605 is achieved when $(\varepsilon, C)=(0.02, 100)$. The test error may increase as ε decreases. This phenomenon is the opposite of training with clean data. The reason is that as ε decreases, more support vectors are generated and the SVR

learns more noise information. This example demonstrates that SKNFS-SVR is more robust to noise than the Gaussian kernel-based SVR.

VI. CONCLUSION

This paper proposes a SKNFS-SVR for regression problems. The overall model size can be reduced because the size of SKNFS-SVR depends on the number of clusters instead of support vectors. Consequent parameters in the SKNFS-SVR are learned by a linear SVR, aiming to minimize structural risk instead of empirical risk. Such novel design provides the SKNFS-SVR with high generalization ability and noise resistance ability. Current simulation results show that the SKNFS-SVR achieves better performance over the Gaussian kernel based SVR for both noiseless and noisy data. Moreover, the SKNFS-SVR model size is relatively much smaller. In the future, comparisons with other types of regression models in different regression problems will be studied to verify the performance of the SKNFS-SVR.

Table 1. Test Results of Different Methods Using Clean Training Data .

	Gaussian SVR		SKNFS-SVR	
ε	0.07	0.02	0.07	0.02
C	10	10	10	10
g	20	20	--	--
Structure	54 SVs	97 SVs	9 rules	9 rules
Parameter number	163	293	55	55
Test RMSE	0.0752	0.0774	0.06653	0.06417

Table 2. Test Results of Different Methods Using Noisy Training Data..

	Gaussian SVR		SKNFS-SVR	
ε	0.07	0.02	0.07	0.02
C	100	100	100	100
g	50	40	--	--
Structure	75 SVs	158 SVs	9 rules	9 rules
Parameter number	277	476	55	55
Test RMSE	0.0719	0.0762	0.0557	0.0605

REFERENCES

- [1] V. Vapnik, The Nature of Statistical Learning Theory. New York : Springer - Verlag, 1995.
- [2] C. Cortes, V. Vapnik. Support vector networks. Machine Learning Journal, vol. 20, no. 3, pp. 1-25, 1995.
- [3] C. F. Juang and C. T. Lin, "An on-line self-constructing neural fuzzy inference network and its applications," *IEEE Trans. Fuzzy Systems*, vol. 6, no. 1, pp. 12-32, Feb. 1998.
- [4] N. Kasabov, "Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning," *IEEE Trans. Syst., Man and Cyber., Part B: Cyber.*, vol. 31, no. 6, pp. 902-918, Dec. 2001.
- [5] N. K. Kasabov and Q. Song, "DENFIS: Dynamic evolving neural-fuzzy inference system and its application for time-series prediction," *IEEE Trans. Fuzzy Systems*, vol. 10, no. 2, pp. 144 -154, April 2002.
- [6] P.P. Angelov and D. Filev, "An approach to online identification of Takagi-Sugeno fuzzy models," *IEEE Trans. Syst., Man and Cyber., Part B: Cyber.*, vol. 34, no. 1, pp. 484-498, Feb. 2004.

- [7] G. Leng, T. M. McGinnity, G. Prasad, "An approach for on-line extraction of fuzzy rules using a self-organizing fuzzy neural network," *Fuzzy Sets and Systems*, vol. 150, no. 2, pp. 211-243, 2005.
- [8] C. F. Juang and C. I. Lee, "A fuzzified neural fuzzy inference network for handling both linguistic and numerical information simultaneously," *Neurocomputing*, vol. 71, no. 1-3, pp. 342-352, Dec. 2007.
- [9] A. Celikyilmaz and I. B. Turksen, "Fuzzy function with support vector machines," *Information Sciences*, vol. 177, pp. 5163-5177, 2007.
- [10] P. Y. Hao and J. H. Chiang, "Fuzzy regression analysis by support vector Learning Approach," *IEEE Trans. Fuzzy Systems*, vol. 16, no. 2, pp. 428-441, 2008.
- [11] C. F. Juang and C. D. Hsieh, "TS-fuzzy system based support vector regression," *Fuzzy Sets and Systems*, vol. 160, no. 17, pp. 2486-2504, Sep. 2009.
- [12] N. Cristianini and J. S.-Taylor, *An Introduction to Support Vector Machines And Other Kernel-based Learning Methods*, Cambridge University Press, 2000.
- [13] A. J. Smola and B. Scholkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol.14, no. 3, pp. 199-222, 2004.