Non-linear system modelling via online clustering and fuzzy support vector machines

Julio César Tovar and Wen Yu*
Departamento de Control Automático,
CINVESTAV-IPN,
Av.IPN 2508, México D.F, 07360, México
E-mail: jtovar@ctrl.cinvestav.mx
E-mail: yuw@ctrl.cinvestav.mx
*Corresponding author

Abstract: This paper describes a novel non-linear modelling approach by online clustering, fuzzy rules and support vector machine. Structure identification is realised by an online clustering method and fuzzy support vector machines, and the fuzzy rules are generated automatically. Time-varying learning rates are applied for updating the membership functions of the fuzzy rules. Finally, the upper bounds of the modelling errors are proven.

Keywords: identification; clustering; fuzzy systems; support vector machines.


Biographical notes: Julio César Tovar received his BS in Electronic Engineering in 2001. He obtained his MSc in Automatic Control from the Research Center of Advanced Studies of the National Polytechnic Institute of México (CINVESTAV-IPN) in 2004. He is currently working toward his PhD in the Department of Automatic Control of the CINVESTAV-IPN. Since 2006, he has been with the Department of Communication and Electronic Engineering from the Higher School of Mechanic and Electrical Engineering of the IPN. His research interests include non-linear systems, neural networks, and fuzzy control.

Wen Yu received his BS from Tsinghua University, Beijing, China in 1990 and his MS and PhD, both in Electrical Engineering, from Northeastern University, Shenyang, China, in 1992 and 1995, respectively. In 1996, he joined CINVESTAV-IPN, México, where he is currently a Professor in the Departamento de Control Automático. He serves as an Associate Editor of Neurocomputing. He is a member of the Mexican Academy of Science.

1 Introduction

Both neural networks and fuzzy logic are universal estimators. They can approximate any non-linear function to any prescribed accuracy, provided that sufficient hidden neurons and fuzzy rules are available (Brown and Harris, 1994; Lin and Lee, 1991). The fusion of some intelligent technologies with fuzzy systems seems to be very effective for non-linear systems modelling. Recent results show that neural networks and fuzzy logic seem to be very effective to identify a wide class of complex non-linear systems when we have no complete model information, or even when we consider the controlled plant as a black box. Fuzzy modelling for non-linear systems is based on a set of IF-THEN rules which use linguistic propositions of human thinking (Zadeh, 1998). The key problem of the fuzzy modelling is extraction of the fuzzy rules; it can be divided into two classes (Leski, 2005):

1. obtaining fuzzy rules from experts
2. obtaining fuzzy rules automatically from observed data.

The expert method uses the un-bias criterion (Rivals and Personnaz, 2003) and the trial-and-error technique; it can only be applied off-line.

The process of fuzzy rule extraction for non-linear systems modelling is called structure identification. A common method is to partition the input and the output data, also called fuzzy grid (Jang, 1993). Most of structure identification approaches are based on offline data clustering, such as fuzzy C-means clustering (Mitra and Hayashi, 2000), mountain clustering (Mitra and Hayashi, 2000), and subtractive clustering (Chiu, 1994). These approaches require that the data is ready before the modelling. There are a few online clustering methods in the literature. A combination of online clustering and genetic algorithm for fuzzy systems is proposed in Juang (2005). In Tzafestas and Zikidis (2001), the input space was automatically partitioned into fuzzy subsets by adaptive resonance theory. Online clustering with a recursively calculated spatial proximity measure was given in Angelov (2004). There is one weakness for the above online
clustering methods: the partitioning of the input (precondition) and the output (consequent) do not take into account a time mark. They use whole data to train each rule. In this paper, a novel online clustering approach is proposed. The time relationship in the input and the output spaces is considered.

Besides clustering approaches, fuzzy rule extraction can also be realised by a neural networks method (Jang, 1993), genetic algorithms (Rivals and Personnaz, 2003), singular value decomposition (SVD-QR) (Chiang and Hao, 2004) and support vector machines (SVM) technique (Cristianini and Shawe-Taylor, 2000). SVM was first used for solving the pattern classification problem. Vapnik defined it as a structure risk minimisation which minimises the upper bound of the modelling error. The basic idea of SVM modelling is to map the inputs into higher dimensional feature space, then solve quadratic programming (QP) with an appropriate cost function (Mueller et al., 2001). SVM has one important property: the solution vector is sparse. Only the non-zero solutions which are called support vectors are useful for the model. In this paper, we use the support vectors to extract the fuzzy rules in each cluster after the online clustering.

In this paper, a three-phase fuzzy SVM technique with online clustering is proposed, see Figure 1. First, we use an online clustering method which divides the input and the output data into several clusters in the same temporal interval, so the structure of fuzzy systems is automatically established. Second, fuzzy SVM is applied to generate support vectors in each cluster. With these support vectors, fuzzy rules are constructed. Here, we use two fuzzy techniques to modify the standard SVM, the kernel is changed as fuzzy membership function and a fuzzy factor is added to the performance index of SVM, and the corresponding fuzzy systems are made. The fuzzy SVM provides an adaptive local representation for SVM, and this takes advantages of some properties of a fuzzy system, such as adaptive learning and economic structure (Hong-Sen and Xu, 2007).

Finally, the membership functions of the precondition and the consequent will be updated by the data in each cluster. It is a process of parameter identification. We use the data to modify the membership functions of each fuzzy rule. Stability of learning algorithms for the parameter identification is very important in applications. It is well known that normal identification algorithms (for example, gradient descent and least squares) are stable in ideal conditions. They might become unstable with respect to unmodelled dynamics; therefore, some robust modification techniques are needed (Ioannou and Sun, 1996). By using passivity theory, we successfully proved that neural networks with time-varying learning rates are stable and robust to any bounded uncertainties (Yu and Li, 2001). Does the parameter identification of the fuzzy modelling have similar characteristics? In this paper, we give a time-varying learning rate for the commonly used back propagation algorithm, and prove that the identification errors are bounded.

2 Online clustering for the input/output data

The following discrete time non-linear system will be identified by our fuzzy modelling method:

\[ y(k) = f \left( x(k), k \right) \]  

where

\[ x(k) = [y(k-1), y(k-2), \ldots, y(k-d), u(k-d-1), \ldots]^T \]  

\( f(\cdot) \) is an unknown slow time-varying non-linear function, representing the plant dynamics, \( u(k) \) and \( y(k) \) are measurable scalar input and output of the non-linear plant, \( d \) is time delay, \( x(k) \in \mathbb{R}^n \) can be regarded as new input to the non-linear function \( f(\cdot) \) it is a NARMAX model (Chen and Billings, 1992).

The objective of structure identification is to partition the input and the output data \( \left[ y(k), x(k) \right] \) of the non-linear system (1) and extract fuzzy rules. We use the following simple example to explain the importance of the online clustering proposed in this paper. We consider a non-linear function as (1).

The non-linear mapping from \( x(k) \) to \( y(k) \) is shown in Figure 2. By the normal online clustering method proposed in Angelov (2004) the input and output may be partitioned into 4 clusters, \( A_1-A_4 \) and \( B_1-B_4 \). From there, four rules can be formulated as ‘IF \( x(k) \) is \( A_j \) THEN \( y(k) \) is \( B_j \), \( j = 1 \ldots 4 \). For example, the third rule is ‘IF \( x(k) \) is \( A_3 \) THEN \( y(k) \) is \( B_4 \)’, this fuzzy rule does not represent relation \( y(k) = f(x(k)) \) well, because the precondition \( x(k) \) and the consequent \( y(k) \) do not occur at the same time for this rule.
In this paper, the input and the output spaces are partitioned in the same temporal interval. There are two reasons. Firstly, non-linear system modelling is to find a suitable mapping between the input and the output. Only when the input and the output of the fuzzy system occur in the same time interval, the fuzzy rules can correspond to the non-linear mapping. Secondly, we consider an online modelling approach. When a new cluster (or a new rule) is created, we do not want to use all data to train it as in Angelov (2004). If the data have time marks, only the data in the corresponding time interval should be used. So clustering with temporal intervals will simplify parameter identification and help to make the task of online modelling.

The basic idea of the proposed online clustering scheme is, if the distance from a point to the centre is less than a required length, the point is in this cluster. When new data comes, the centre and the cluster should be changed according to the new data. We give the following algorithm. The Euclidean distance at time $k$ is defined as

$$d_{k,x} = \left( \sum_{j=1}^{n} \left( \frac{x(k) - \bar{x}}{x_{\text{max}} - x_{\text{min}}} \right)^2 \right)^{1/2}$$

$$d_{k,y} = \frac{y(k) - \bar{y}}{y_{\text{max}} - y_{\text{min}}}$$

$$d_k = \alpha d_{k,x} + (1 - \alpha) d_{k,y}$$

where $x_{\text{max}} = \max_i \{x_i(k)\}$, $x_{\text{min}} = \min_i \{x_i(k)\}$, $y_{\text{max}} = \max_i \{y_i(k)\}$, $y_{\text{min}} = \min_i \{y_i(k)\}$, $\bar{x}/(k)$ and $\bar{y}/(k)$ are the centres of the cluster, $i$ is the dimension of input, $\alpha$ is a positive factor, normally we can choose $\alpha = 1/2$. For the cluster $j$ the centres are updated as

$$\bar{x}_j = \frac{1}{l_j - l_{j-1} + 1} \sum_{l=1}^{l_j} x_i(l)$$

$$\bar{y}_j = \frac{1}{l_j - l_{j-1} + 1} \sum_{l=1}^{l_j} y_i(l)$$

where $l_j^1$ is the first time index of the cluster $j$, $l_j^j$ is the last time index of the cluster $j$. The length of cluster $j$ is $m_j = l_j^j - l_j^1 + 1$. The time interval of cluster $j$ is $\left[l_j^1, l_j^j\right]$. The process of the structure identification can be formed as the following steps:

1. For the first data $G_1$, $k=1$, $\alpha(1), x(1)$ are the centres of the first cluster, $\bar{x}_1 = x_1(1), \bar{y}_1 = y_1(1), l_1^1 = l_1^j = 1, j = 1$.

2. If a new data $[\alpha(k), x(k)]$ comes, increment $l_j^j$ by 1, we use () and () to calculate $d_k$. If no any new data comes, go to 5.

3. If $d_k \leq L$ then $[\alpha(k), x(k)]$ is still in cluster $G_j$, go to 2.

4. If $d_k > L$ then $[\alpha(k), x(k)]$ is in a new cluster, increment $j$ by 1 the centre of the $G_j$ is $\bar{x}_j = x_j(k), \bar{y}_j = y_j(k)$, $l_j^j = l_j^j = k$, go to 2.

5. Check the distances between all centres $\bar{x}_j, \bar{y}_j$, if

$$\sum_{i=1}^{n} \left( \frac{x_i - \bar{x}}{x_{\text{max}} - x_{\text{min}}} \right)^2 \leq L$$

, the two clusters $G_p$ and $G_q$ are combine into one cluster.

There are two design parameters $\alpha$ and $L$. $\alpha$ is regarded as the weight on the input space. If the input dominates the dynamic property, we should increase $\alpha$. Usually we select $\alpha = 0.5$ such that the input and the output have equal influence. If we let $\alpha = 1$, it becomes the normal online clustering (Angelov, 2004; Juang, 2005; and Tzafestas and Zikidis, 2001). $L$ is the threshold of creating new rules; it is the lowest possible value of similarity. How to choose the user defined threshold is a trade-off problem. If the threshold value $L$ is too small, there are many clusters and some of them will be singletons. Conversely, if the threshold $L$ is too large, many objects that are not similar may be partitioned in the same cluster. Since $d_k = \alpha d_{k,x} + (1 - \alpha) d_{k,y}$, so

$$d_{\text{max}} = \alpha \left( x_{\text{max}} - x_{\text{max}} \right) + (1 - \alpha) \left( y_{\text{max}} - y_{\text{min}} \right)$$

. If we want several clusters, we should let $L < d_{\text{max}}$ otherwise there is only one cluster.

### 3 Fuzzy rules extraction by fuzzy support vector machines

An SVM can separate the data into two classes with a maximum margin hyperplane (Cristianini and Shawe-Taylor, 2000). If the training is detachable for the hyperplane, the function is chosen as $h(x) = (w \cdot x) + b$. The margin is defined as the minimum distance from a sample to the surface of resolution. We can measure this margin by the longitude of the vector $w$. In this way the near points to the hyperplane satisfies $|w \cdot x + b| = 1$. 

In this paper we use SVM for function estimation. In order to find support vectors in the cluster \( j \), we use the input/output data \( \{ \mathbf{x}_k, y(k) \} \), \( k \in [l_l, l_u] \) to approximate a non-linear function. Consider a non-linear function regression \( y = f(x) \) can be estimated as the following form,

\[
h(\mathbf{x}_k) = \mathbf{w}^T \varphi(\mathbf{x}_k) + b
\]

(6)

where \( \mathbf{w} \) is the weight vector, \( \varphi(\mathbf{x}_k) \) is a known non-linear function, \( b \) is a threshold, \( \mathbf{x}_k \) is the input vector at time \( k \). Kernel trick is defined as \( k(\mathbf{x}, \mathbf{x}_j) = \varphi(\mathbf{x})^T \varphi(\mathbf{x}_j) \), \( \mathbf{x} \) is the input vector at any time.

### 3.1 Fuzzy kernel

There are many possible choices for the kernel \( k(\mathbf{x}, \mathbf{x}_j) \) we only require \( k(\mathbf{x}, \mathbf{x}_j) \) satisfies the Mercer condition (Cristianini and Shawe-Taylor, 2000). For example the linear kernel \( k(\mathbf{x}, \mathbf{x}_j) = \mathbf{x}_j \mathbf{x}^T \), the MLP kernel \( k(\mathbf{x}, \mathbf{x}_j) = \tanh(a_1 \mathbf{x}^T + a_2) \), \( a_1 \) and \( a_2 \) are constants. RBF kernel

\[
k(\mathbf{x}, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_j\|^2}{\sigma^2}\right).
\]

In this paper, we use the fuzzy kernel, which is defined as

\[
k(\mathbf{x}, \mathbf{x}_j) = \begin{cases} \mu(\mathbf{x}_j) u_i(\mathbf{x}_j) & \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ are both in the } j\text{th cluster} \\ 0 & \text{otherwise} \end{cases}
\]

(7)

where \( M \) is total time, \( u_i(\mathbf{x}_j) \) is the membership function.

Let the training set be

\[
\mathcal{S} = \{(\mathbf{x}_1, y(1)), (\mathbf{x}_2, y(2)), \ldots, (\mathbf{x}_M, y(M))\}
\]

Assume the training samples are partitioned into \( l \) clusters. We can group the training samples into \( l \) clusters as follows:

- cluster 1 = \( \{(\mathbf{x}_1^1, y_1^1), \ldots, (\mathbf{x}_1^k, y_1^k)\} \)
- cluster 2 = \( \{(\mathbf{x}_2^1, y_2^1), \ldots, (\mathbf{x}_2^k, y_2^k)\} \)
- \( \ldots \)
- cluster \( l \) = \( \{(\mathbf{x}_l^1, y_l^1), \ldots, (\mathbf{x}_l^k, y_l^k)\} \)

where \( \mathbf{x}_l^k \) and \( y_l^k \) are the first input/output of the cluster \( g \), \( \mathbf{x}_l^k \), \( y_l^k \) are the last input/output of the cluster \( g \). \( g = 1, 2, \ldots, l \) is the number of points belonging to the \( g \)th cluster, so that we have \( \sum_{g=1}^{l} k_g = M \).

Then the fuzzy kernel can be calculated by using the training set in (7), and the resulting kernel matrix \( \mathbf{K} \) can be rewritten as the following form (Lin and Wang, 2002):  

\[
\mathbf{K} = \begin{bmatrix} \mathbf{K}_1 & 0 & \ldots & 0 \\ 0 & \mathbf{K}_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & \mathbf{K}_l \end{bmatrix} \in \mathbb{R}^{M \times M}
\]

(8)

How to choose the membership function \( u_i(\mathbf{x}_j) \) is another problem. Gaussian function and triangle function are the most popular functions for the membership function of fuzzy systems. When \( u_i(\mathbf{x}_j) \) is Gaussian function, the kernel function is

\[
G = \varphi(\mathbf{x}_1) \cdot \varphi(\mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_1 - \mathbf{x}_j\|^2}{2\sigma^2}\right)
\]

(9)

and the fuzzy kernel is

\[
\mathbf{K}_g = \begin{bmatrix} G(\mathbf{x}_1, \mathbf{x}_1) & G(\mathbf{x}_1, \mathbf{x}_2) & \cdots & G(\mathbf{x}_1, \mathbf{x}_M) \\ G(\mathbf{x}_2, \mathbf{x}_1) & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ G(\mathbf{x}_M, \mathbf{x}_1) & \cdots & G(\mathbf{x}_M, \mathbf{x}_2) & G(\mathbf{x}_M, \mathbf{x}_M) \end{bmatrix} \in \mathbb{R}^{M \times M}
\]

(10)

### 3.2 Fuzzy support vector machines

The normal cost function of non-linear regression is defined as

\[
R_{opt}(\theta) = \frac{1}{N} \sum_{k=1}^{N} \left| y_k - \left( \mathbf{w}^T \varphi(\mathbf{x}_k) + b \right) \right|_\varepsilon
\]

where \( g \) is the Vapnik’s insensitive loss function, defined as

\[
\left| y_k - f(\mathbf{x}) \right| = \begin{cases} 0 & \left| y_k - f(\mathbf{x}) \right| \leq \varepsilon \\ \left| y_k - f(\mathbf{x}) \right| - \varepsilon & \text{otherwise} \end{cases}
\]

\( \varepsilon \) can be regarded as the accuracy of approximation.

In this paper, we introduce a fuzzy factor \( s_i \) to the above performance index (Lin and Wang, 2002). Here \( \sigma \leq s_i \leq 1 \) where \( \sigma \) is a sufficient small positive number, \( s_i \) denotes the important degree of sample \( x_i \) for learning the optimal hyperplane in SVM. We select \( s_i \) as bell shape function (11), see Figure 3.

\[
s_i = \frac{1}{1 +\left(\frac{1-s_i}{a}\right)^2b}
\]

(11)

The optimal hyperplane problem is regarded as the solution to the following problem (primal problem)

\[
\min J_p = \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{k=1}^{n} s_k \xi_k
\]

(12)

subject to \( \left| y_k - \left( \mathbf{w}^T \varphi(\mathbf{x}_k) + b \right) \right| \leq \varepsilon \)

where \( \xi_k \geq 0 \), \( k = 1, 2, \ldots, n \). \( C \) is constant, the smaller \( s_i \) is, the smaller the effect that the sample \( x_i \) to optimal hyperplane. If \( \varepsilon \) is too small, certain points will be outside of this \( \varepsilon \) tube. Therefore additional slack variables \( \xi_k \) are introduced (it is somewhat similar to the classifier case with overlapping). To solve the optimal problem above, we can construct the following Lagrangian function with inequality constraint and slack variables.
\[ L(w, b, \zeta, \alpha, \beta) = \frac{1}{2}w^T w + C \sum_{k=1}^{n} \alpha_k \zeta_k \]
\[ - \sum_{k=1}^{n} \alpha_k \left( y_k - \left( w^T \phi(x_k) + b \right) - 1 + \zeta_k \right) - \sum_{k=1}^{n} \beta_k \zeta_k \]

where \( \alpha_k, \beta_k \geq 0 \) are the Lagrangian multipliers. The primal problem (12) can be transformed into the dual problem by differentiating \( L \) with respect to \( w, \zeta \) and \( b \).

Figure 3 Generalised bell shape built-in membership function (see online version for colours)

From Kuhn-Tucker conditions, necessary and sufficient conditions for the optimal \((w^*, \alpha^*, \beta^*)\) are

\[ \frac{\partial L}{\partial w} = w - \sum_{k=1}^{n} \alpha_k y_k \phi(x_k) = 0 \]
\[ \frac{\partial L}{\partial b} = -\sum_{i=1}^{n} \alpha_i y_i = 0 \]
\[ \frac{\partial L}{\partial \zeta} = s_k C - \alpha_k - \beta_k = 0 \]

\[ y^*_k - \left( w^T \phi(x_k) + b \right) \leq \epsilon, \quad \alpha_k, \beta_k \geq 0 \]

and

\[ \alpha_k \left[ y_k \left( w^T \phi(x_k) + b \right) - 1 + \zeta_k \right] = 0 \] (15)

Apply these conditions into the Lagrangian (13), the primal problem (12) can be changed into the following dual problem (quadratic programming)

\[ \max W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \]

subject to \( \sum_{i=1}^{n} \alpha_i y_i = 0, \quad \alpha_i \geq 0, K(x_i, x_j) \geq 0 \)

where \( 0 \leq \alpha_i \leq s_i C, \quad i = 1, 2, \ldots, n, \)

\( K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j) \) is fuzzy kernel.

By standard QP software package we obtain the solution \( \alpha^*_k, b \) which can be solved by (15). The resulting function is

\[ f(x) = \sum_{k=1}^{n} \left( \alpha_k - \alpha^*_k \right) K(x_k, x) + b \] (17)

Since many \( \alpha^*_k = 0 \), the solution vector is sparse, the sum should be taken only over the non-zero \( \alpha_k \) (support vector), so the final result is

\[ f(x) = \sum_{k=1}^{sv} \left( \alpha_k - \alpha^*_k \right) K(x_k, x) + b \]

where \( sv \) is the number of the support vectors, \( sv < \sqrt{i_n - i_l + 1} \). We define the positions of the support vectors as \( \left[ x^*_i, y^*_i \right] \), \( i = 1, \ldots, sv \).

The point \( x_i \) with the corresponding \( 0 \leq \alpha_k \leq s_i C \) is called a support vector. For fuzzy SVM, there are two types of support vectors, the one corresponding to \( 0 \leq \alpha_k \leq s_i C \) lies on the margin of the hyperplane, and the one corresponding to \( \alpha_k = s_i C \) is misclassified. So the points with the same value of \( \alpha_k \) in fuzzy SVM may indicate a different type of support vector in fuzzy SVM due to the factors \( s_i \).

From (17) we know the support vectors are enough to represent the nonlinearity in each cluster. We use these support vectors to construct fuzzy rules. For the cluster \( j \) we extract fuzzy product rules in the following form (Mamdani fuzzy model, Mamdani, 1976)

\[ R^j : \text{IF} \, x_1(k) \text{is} \, A^j_1 \text{and} \, x_2(k) \text{is} \, A^j_2 \cdots \text{and} \, x_n(k) \text{is} \, A^j_n \, \text{THEN} \, y(k) \text{is} \, B^j \]

here \( j = 1, sv, A^j_1, \ldots, A^j_n \) and \( B^j \) are standard fuzzy sets. We use \(妖 \) fuzzy IF-THEN rules to perform a mapping from an input vector \( x = [x_1 \cdots x_n] \in \mathbb{R}^n \) to an output \( y(k) \) The Gaussian membership function is

\[ \mu_i(x_j) = \exp \left( -\frac{(x_j - c_{ji})^2}{\sigma_{ji}^2} \right) \] (18)

where \( i \) is the condition in the part ‘IF’, \( i = 1 \ldots n, j \) is rule number, \( j = 1 \ldots sv \). If \( \left[ x^*_i, y^*_i \right] \) is the point of the support vector, we let \( x^* \) and \( y^* \) as the centre of the Gaussian functions.

From Wang, (1994) we know, by using product inference, centre average and singleton fuzzifier, the output of the fuzzy system in cluster \( j \) can be expressed as

\[ \hat{y} = \sum_{j=1}^{sv} w_j \left[ \prod_{i=1}^{n} \mu_{A^j_i} \right] / \sum_{j=1}^{sv} \left[ \prod_{i=1}^{n} \mu_{A^j_i} \right] \] (19)

where \( \mu_{A^j_i} \) are the membership functions of the fuzzy sets \( A^j_i \), \( w_j \) is the point at which \( \mu_{B_j} = 1 \). If we define
\[ \phi_i = \frac{n}{j=1} \mu_{A_j} / \sum_{i=1}^{n} \mu_{A_j} \]

(19) can be expressed in matrix form

\[ \hat{y}(k) = W(k) \Phi \left[ x(k) \right] \]  

(20)

where the parameter \( W(k) = [w_1, \ldots, w_n] \in \mathbb{R}^{n \times 1} \), data vector \( \Phi \left[ x(k) \right] = [\phi_1, \ldots, \phi_n]^T \in \mathbb{R}^{n \times 1} \).

For Takagi-Sugeno fuzzy model (Takagi and Sugeno, 1985),

\[ R^j: \text{If } x_1 \text{ is } A_{j1} \text{ and } x_2 \text{ is } A_{j2} \text{ and } \ldots \text{ and } x_n \text{ is } A_{jn} \text{ then } y = p_0 + p_1 x_1 + \ldots + p_n x_n \]

(21)

where \( j = 1 \) sv. The output of the fuzzy logic system can be expressed as

\[ \hat{y} = \sum_{i=1}^{sv} (p_0^i + p_1^i + \ldots + p_n^i) \phi_i \]

(22)

(22) can be also expressed in the form of the Mamdani-type (20),

\[ W(k) = [p_0, p_0^1, \ldots, p_n^1, \ldots, p_0^s, p_0^s \ldots, p_n^s] \]

(23)

**4 Membership functions training**

For the cluster \( j \), we use the input-output data \( [y(k), x(k)] \), \( k \in [l_1, l_2] \) to train the membership functions \( A_{ji} \) (\( i = 1 \) n) and \( B_{ji} \), i.e., the parameter identification of the membership functions are performed in the corresponding time interval found in the structure identification. We discuss two cases: consequence membership functions training and premise membership functions training.

First, we assume the premise membership functions \( A_{j1} \ldots A_{jn} \) are given by prior knowledge, i.e.,

\[ \hat{y} = \frac{n}{j=1} \mu_{A_j} / \sum_{i=1}^{j} \mu_{A_j} \] is known, (see Juang, 2005 and Wang, 1994). Mamdani (20) and the TSK (23) models have the same forms because \( \Phi \left[ x(k) \right] \) is known. We select \( W(1) = y_j \) as initial conditions. We use the data pair \( [x(k), y(k)] \) to find some suitable membership functions in the zone \( k \in [l_1, l_2] \). This can be transformed into a modelling problem of determining the parameters \( \mu_j \), \( \bar{x}_j \) and \( \sigma_j \) such that \( \hat{y} \rightarrow y \).

Let us define identification error \( e(k) \) as

\[ e(k) = \hat{y}(k) - y(k) \]  

(24)

According to function approximation theories of fuzzy logic (Wang, 1994), the identified nonlinear process \( (1) \) can be represented as

\[ y(k) = W^\ast \Phi \left[ x(k) \right] - \mu(k) \]  

(25)

where \( W^\ast \) is unknown weights which can minimise the unmodelled dynamic \( \mu(k) \). The identification error can be represented by (24) and (25).

\[ e(k) = \hat{W}(k) \Phi \left[ x(k) \right] + \mu(k) \]

(26)

where \( \hat{W}(k) = W(k) - W^\ast \). In this paper we are only interested in open loop identification, we assume that the plant \( (i) \) is bounded input and bounded output (BIBO) stable, i.e., \( y(k) \) and \( u(k) \) in \( (1) \) are bounded. By the bound of the membership function \( \Phi, \mu(k) \) in (25) is bounded. The following theorem gives a stable gradient descent algorithm for fuzzy neural modelling.

**Theorem 1**: If we use the fuzzy system (19) to identify non-linear plant \( (1) \) in cluster \( j \), the following gradient descent algorithm with a time varying learning rate can make identification error \( e(k) \) bounded

\[ W(k+1) = W(k) - \eta_k e(k) \Phi^T \left[ x(k) \right] \]  

(27)

where the scalar \( \eta_k = \frac{\eta}{1 + \| \Phi \left[ x(k) \right] \|^2} \) satisfies the following average performance

\[ \lim \sup_{T \rightarrow \infty} \frac{1}{T} \sum_{k=1}^{T} \| e(k) \|^2 \leq \bar{\mu} \]

(28)

where \( \bar{\mu} = \max_k \left[ \| \mu(k) \|^2 \right] \).

**Remark 1**: Generally, a fuzzy system cannot match a non-linear system exactly. The parameters of the fuzzy system will not converge to its optimal values. The online identification proposed in this paper is to force the output of the fuzzy system to follow the output of the plant. Although the parameters cannot converge to their optimal values, (28) shows that the normalised identification error will converge to a ball radius \( \bar{\mu} \). If the fuzzy system (29) can match the non-linear plant \( (1) \) exactly \( (\mu(k) = 0) \), i.e., we can find the best membership function \( \mu_{A_j} \) and \( W^\ast \) such that the non-linear system can be written as \( y(k) = W^\ast \Phi \left[ x(k) \right] \). Since \( \| e(k) \|^2 > 0 \), the same learning...
The normalising learning rate $\eta_k$ in (27) is time varying to assure the stability of identification error. These learning rates are easier to be chosen than Jang (1993) without requiring any prior information, for example we may select $\eta = 1$. Time varying learning rates can be found in some standard adaptive schemes (Ioannou and Sun, 1996). But they need robust modifications to guarantee stability of the identification. (27) is similar to the results of Mandic et al. (2001). In this paper the algorithm is derived from stability analysis (or ISS-Lyapunov function), the algorithm of Mandic et al. (2001) was obtained from minimisation of the cost function. We focus on the bound of the identification error, Mandic et al. (2001) focused on convergence analysis. It is interested to see that the two different methods can get similar results.

Now we discuss the case of training both consequence and premise membership functions. The initial conditions are $c_{ji}(1) = x_j^*$, $W_j(1) = y_j^*$, $\sigma_{ji}(1)$ is random in (0,1).

Since the membership functions are Gaussian functions, the output of the fuzzy system can be expressed as

$$\hat{y} = \frac{\sum_{i=1}^{sv_j} W_i \prod_{j=1}^{n} \exp\left(-\frac{(x_j-c_{ji})^2}{\sigma_{ji}^2}\right)}{\sum_{i=1}^{sv_j} \prod_{j=1}^{n} \exp\left(-\frac{(x_j-c_{ji})^2}{\sigma_{ji}^2}\right)}$$

Let us define

$$z_i = \prod_{j=1}^{n} \exp\left(-\frac{(x_j-c_{ji})^2}{\sigma_{ji}^2}\right) a = \sum_{i=1}^{sv_j} W_i z_i \quad b = \sum_{i=1}^{sv_j} z_i$$

So $\hat{y} = \frac{a}{b}$. Similar as (25), the non-linear plant (1) can be represented as

$$\hat{y} = \frac{\sum_{i=1}^{sv_j} W_i^* \prod_{j=1}^{n} \exp\left(-\frac{(x_j-c_{ji})^2}{\sigma_{ji}^2}\right)}{\sum_{i=1}^{sv_j} \prod_{j=1}^{n} \exp\left(-\frac{(x_j-c_{ji})^2}{\sigma_{ji}^2}\right)} - \mu$$

where $W_i^*$, $c_{ji}^*$ and $\sigma_{ji}^2$ are unknown parameters which may minimise the unmodelled dynamic $\mu$. In the case of three independent variables, a smooth function $f$ has Taylor formula as

$$f(x_1, x_2, x_3) = \sum_{k=0}^{l-1} \frac{f^{(l)}}{l!} x_1^{(l-1)} x_2^{(l-2)} x_3^{(l-3)} + \frac{f^{(l)}}{l!} x_1^{(l-1)} x_2^{(l-2)} x_3^{(l-3)} + \cdots + f + R_l$$

where $R_l$ is the remainder of the Taylor formula. If we let $x_1, x_2, x_3$ correspond $W_i^*$, $c_{ji}^*$ and $\sigma_{ji}^2$, $x_1^0, x_2^0, x_3^0$ correspond $W_i$, $c_{ji}$ and $\sigma_{ji}$,

$$y + \mu = \hat{y} + \sum_{i=1}^{n} s_{ji} z_i / b + \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial f}{\partial x_j} (x_j - c_{ji}) - \sum_{i=1}^{n} \frac{\partial f}{\partial x_j} (x_j - c_{ji}) + \frac{R_l}{b}$$

Remark 2: Let us define

$$\eta_k = \sum_{i=1}^{n} s_{ji} z_i$$

which may minimise the unmodelled dynamic $\mu$. Theorem 2: If we use Mandani-type fuzzy neural network (30) to identify non-linear plant (1), the following back propagation algorithm makes identification error $e(k)$ bounded

$$W_{k+1} = W_k - \eta_k e(k) Z(k)^T$$

$$c_{ji}(k+1) = c_{ji}(k) - 2\eta_k z_i \frac{W_{k}}{b} \frac{\sigma_{ji}}{\sigma_{ji}} (\hat{y} - y)$$

$$\sigma_{ji}(k+1) = \sigma_{ji}(k) - 2\eta_k z_i \frac{W_{k}}{b} \frac{\sigma_{ji}}{\sigma_{ji}} (\hat{y} - y)$$

Using (33) we have

$$e(k) = Z(k) W + D_k \tilde{C}_k + D_k \tilde{B}_k + R_k$$

By the bound of the Gaussian function $\phi$ and the plant is BIBO stable, $\mu$ and $R_k$ in (31) and (32) are bounded. The following theorem gives a stable algorithm for discrete time Mamdani-type fuzzy neural networks.

**Theorem 2:** If we use Mandani-type fuzzy neural network (30) to identify non-linear plant (1), the following back propagation algorithm makes identification error $e(k)$ bounded
where \( \eta_k = \frac{\eta}{1 + \|z_i\|^2 + 2\|\theta_i\|^2} \), \( 0 < \eta \leq 1 \). The average of the identification error satisfies

\[
J = \lim_{T \to \infty} \frac{1}{T} \sum_{k=1}^{T} \|\tilde{e}(k)\|^2 \leq \frac{\eta}{\pi} \zeta
\]

where

\[
\pi = \frac{\eta}{|x^e|^2}, \quad \kappa = \max_k \left[ |z_i|^2 + 2|\theta_i|^2 \right], \quad \zeta = \max_j \left[ z_j(k) - \bar{z}_j \right] \in [k A_x k A_y k B]
\]

Proof it is similar to Theorem 1.

If we use TSK-type fuzzy model (22), and \( A_{ij} \) is used as Gaussian function. The output of the fuzzy logic system can be expressed as

\[
\hat{y} = \frac{1}{\sum_{i=1}^{s_f} \left( \sum_{k=0}^{s_f} p_{ik}^a - p_{ik}^b \right) x_k} \frac{\sum_{i=1}^{s_f} \left( \sum_{k=0}^{s_f} p_{ik}^a \right) x_k}{b} \]

A stable algorithm for TSK-type fuzzy neural networks is as following

\[
p_{ik}^a(k+1) = p_{ik}^a(k) - \eta_k (\hat{y} - y) \frac{z_i}{b} x_k
\]

\[
c_{ij}(k+1) = c_{ij}(k) - 2\eta_k z_i \frac{w_j - z_j}{b} \frac{x_k - c_{ij}(k)}{\sigma_j^2} (\hat{y} - y)
\]

\[
\sigma_{ij}(k+1) = \sigma_{ij}(k) - 2\eta_k z_i \frac{w_j - z_j}{b} \frac{x_k - c_{ij}(k)}{\sigma_j^2} (\hat{y} - y)
\]

where \( \eta_k = \frac{\eta}{1 + \|z_i\|^2 + 2\|\theta_i\|^2} \), \( 0 < \eta \leq 1 \). One can see that the difference between the TSK model and the Mamdani model is the basic functions, it is easy to extend Theorem 1 and Theorem 2 to TSK model.

Now, each cluster has a fuzzy model (31) or (37). This model is valid in the time intervals of the cluster. So (31) or (37) is piecewise non-linear function. For example, in cluster 3 the time interval is \( k \in \left[ \frac{3}{2}, \frac{3}{2} \right] \), the fuzzy model for the cluster is \( \hat{y} = f_3(x) \). This means that

\[
\hat{y} = f_3(x(k)), \quad k \in \left[ \frac{3}{2}, \frac{3}{2} \right]
\]

We can use the idea of Takagi-Sugeno model (Takagi and Sugeno, 1985) to combine these local models into a global model. We define the following fuzzy rules which are corresponded to each cluster

\[
R^j: \text{IF } x_{ij} \leq x_j \leq x_{ij}^+ \text{, and } \cdots \text{, THEN } \hat{y}(k) = f_j(x(k))
\]

where \( j = 1, p \), and \( p \) is the cluster number by the online clustering. The membership function for \( x_j \) is defined as in Figure 4. This is a trapezoidal function

\[
\mu_{A_{ij}}(x) = \begin{cases} \frac{a - x_j}{x_{ij} - x_{ij}^+} & x \in \left[ x_{ij}, x_{ij}^+ \right] \\ 0 & \text{otherwise} \end{cases}
\]

The final fuzzy model is

\[
\hat{y} = \left( \sum_{j=1}^{p} f_j(x(k)) \right) \left( \sum_{i=1}^{n} \prod_{j=1}^{p} \mu_{A_{ij}}(x(k)) \right) \left( \sum_{i=1}^{n} \prod_{j=1}^{p} \mu_{A_{ij}}(x(k)) \right)
\]

Figure 4 Membership function of \( x_i \)

5 Comparisons with other fuzzy modelling algorithms

In this section, we discuss a typical problem which is also discussed by (Jang, 1993; Narendra and Mukhopadhyay, 1997; Wang et al., 2001). The identified plant is

\[
y(k) = \frac{y(k-1)y(k-2) + 2.5}{1 + y(k-1)^2 + y(k-2)^2} + u(k-1)
\]

The training input signal is selected as random number in the interval \([0,1] \).

\[
X(k) = \begin{bmatrix} y(k-1), y(k-2), u(k-1) \end{bmatrix}^T = \begin{bmatrix} x_1(k), x_2(k), x_3(k) \end{bmatrix}^T
\]

We use the following Mamdani fuzzy model, for \( j \)th rule

\[
R^j: \text{IF } x_{ij} \in A_{ij}^j \text{ and } x_j \in A_{ij}^j \text{ and } x_j \in A_{ij}^j \text{ THEN } y(k) = B_{ij}^j
\]

1 Firstly, we use online clustering. We select \( \alpha = 0.4 \). As a prior knowledge, we know the maximum changes in the input and the output are about 3 and 1, 

\[
\|x_{\max} - x_{\min}\| \approx 3, \quad \|y_{\max} - y_{\min}\| \approx 1.8. \text{ So } L \text{ should be chosen such that } L < 1.8 \text{ in this application we select } L = 1.5. \text{ Figure 5 shows the online clustering results for two months data. Here '*' represents the centre of each}
\]
Non-linear system modelling via online clustering and fuzzy support vector machines

cluster, + is the boundary between the clusters. There are five clusters in $0 < t \leq 100$.

Figure 5  Online clustering (see online version for colours)

Secondly, we use support vector machines (12) to obtain the support vectors as in (17). The numbers of support vector number for the five clusters are 7, 5, 8, 6 and 8. So we have 34 rules as in (40). The centres of the Gaussian membership functions of these rules are the positions of the support vectors.

Figure 6  34 membership functions of $x_i$ (see online version for colours)

Now we change the design parameter $\alpha$. $\alpha = 0.1$. We know the maximum changes in the input and the output are about 3 and 1,

$$\alpha \|x_{\text{max}} - x_{\text{min}}\| + (1 - \alpha) \|y_{\text{max}} - y_{\text{min}}\| = 1.2.$$ 

So $L$ should be chosen such that $L < 1.2$, in this application we select $L=1$. There are six clusters and 40 rules as in (40). The modelling results are worse than that of $\alpha = 0.4$ because there are not enough data in each cluster to update the membership functions and the semantics of fuzzy rules may lose with so many rules.

Finally, we compare the above approach with the following fuzzy modelling methods.

1 Adaptive fuzzy modelling approach (Jang, 1993; Wang, 1994) may be the most popular method. We also use 33 fuzzy rules in the form of (40). The Gaussian membership functions are selected randomly at first, and then we use the same training data to update these membership functions. The testing data is also the same as the above.

2 Fuzzy modelling via online clustering can reduce the number of rules, we consider the clustering without considering the temporal interval and each cluster training uses all of data as in (Juang, 2005; Tzafestas and Zikidis, 2001; Angelov, 2004). If we use the output and the input thresholds as 1.5. 30 fuzzy rules are constructed, and then all data are used to train them.

3 Another popular fuzzy modelling method is fuzzy logic with data clustering (Jang, 1993; Mitra and Hayashi, 200; Chiu, 1994). The input is partitioned. With the thresholds 1.5, we have eight clusters in the input space. So we construct eight fuzzy rules. The remaining parts are the same as the method 1.

Table 1 presents the concrete comparisons between our approaches (online clustering with SVM) with the other

After the training phase is finished, we use the test input as

$$u(k) = \sin(k/30) + \cos(k/20) + 2 \sin(k/10)$$

The final modelling results are shown in Figure 7.

Figure 7  Fuzzy modelling (see online version for colours)
three fuzzy modelling methods. All of the four methods use fuzzy models; they are simple if we know the fuzzy rules. Our online clustering with SVM and the adaptive fuzzy modelling can be used to change the model online; the other two approaches are used off line. Our algorithm is most complex because we use the optimisation in SVM, so the computational time is longer and the accuracy is highest. Although the adaptive fuzzy modelling is very simple, its structure (fuzzy rules) should be known, so the testing error is biggest.

Table 1  Comparison of four fuzzy modelling methods

<table>
<thead>
<tr>
<th>Model</th>
<th>Adaptive fuzzy modelling</th>
<th>Fuzzy modelling via online clustering</th>
<th>Fuzzy modelling via clustering</th>
<th>Online with SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Online</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Algorithm</td>
<td>very simple</td>
<td>simple</td>
<td>simple</td>
<td>no</td>
</tr>
<tr>
<td>Speed</td>
<td>very fast</td>
<td>fast</td>
<td>fast</td>
<td>normal</td>
</tr>
<tr>
<td>Structure</td>
<td>no-adaptive</td>
<td>adaptive</td>
<td>adaptive</td>
<td>adaptive</td>
</tr>
<tr>
<td>Parameter</td>
<td>adaptive</td>
<td>adaptive</td>
<td>adaptive</td>
<td>adaptive</td>
</tr>
<tr>
<td>Accuracy</td>
<td>normal</td>
<td>high</td>
<td>normal</td>
<td>very</td>
</tr>
</tbody>
</table>

6 Conclusions

In this paper, we propose an efficient approach for non-linear system modelling using fuzzy rules. Several techniques are combined for the new approach. First we propose an online clustering method which divides the input/output data into several clusters in a same temporal interval. Then we apply fuzzy support vector machines which generate support vectors in each cluster. With these support vectors, fuzzy rules are constructed and the corresponded fuzzy system is made. After the structure identification, a time-varying learning rate is applied for the parameters identification. The contributions of the paper are:

1 online clustering method and support vectors machines are used for the fuzzy rules extraction

2 the upper bound of modelling error and stability are proved for the fuzzy modelling.

References


Appendix

Proof, we selected a positive defined scalar $L_k$ as

$$L_k = \|\hat{W}(k)\|^2$$

By the updating law (27), we have

$$\hat{W}(k+1) = \hat{W}(k) - \eta_k e(k)\Phi^T[x(k)]$$

Using the inequalities

$$\|a - b\| \geq \|a\| - \|b\| - 2\|a - b\| \leq a^2 + b^2$$

for any a and b. By using (26) and $0 \leq \eta_k \leq \eta \leq 1$, we have

$$\Delta_k = L_{k+1} - L_k = \|\hat{W}(k) - \eta_k e(k)\Phi^T(x(k)) - \hat{W}(k)\|^2$$

$$= \|\hat{W}(k)\|^2 - 2\eta_k \|e(k)\|\hat{W}(k)\| + \eta_k^2 \|e(k)\|^2 - \|\hat{W}(k)\|^2$$

$$= \eta_k^2 \|e(k)\|^2 - 2\eta_k \|e(k)\|\hat{W}(k)\| + \eta_k \|e(k)\|^2 + \eta_k \|\mu(k)\|^2$$

$$\leq \eta_k \|e(k)\|^2 (1 - \eta_k \|\Phi(x(k))\|^2) + \eta_k \|\mu(k)\|^2$$

$$= -\eta_k \eta_k \|e(k)\|^2 \|\Phi(x(k))\|^2 + \eta_k \|\mu(k)\|^2$$

Since $\eta_k = \frac{\eta}{1 + \|\Phi(x(k))\|^2}$,

$$\eta_k \left(1 - \eta_k \|\Phi(x(k))\|^2\right) = \eta_k \left(1 - \frac{\eta}{1 + \|\Phi(x(k))\|^2}\|\Phi(x(k))\|^2\right)$$

$$\geq \eta_k \left(1 - \frac{\max_i \|\Phi(x(k))\|^2}{1 + \max_i \|\Phi(x(k))\|^2}\right) \geq \eta_k \left(1 - \frac{\|\Phi(x(k))\|^2}{\max_i \|\Phi(x(k))\|^2}\right)$$

$$= \eta_k \left(1 + \max_i \|\Phi(x(k))\|^2\right) \geq \eta_k $$

So

$$\Delta_k \leq \pi \|e(k)\|^2 + \eta \|\mu(k)\|^2$$

where $\pi$ is defined as

$$\pi = \frac{\eta}{\left[1 + \max_i \|\Phi(x(k))\|^2\right]^2}$$

Because

$$n \min \{\hat{W}_k^2\} \leq L_k \leq n \max \{\hat{W}_k^2\}$$

where $n \min \{\hat{W}_k^2\}$ and $n \max \{\hat{W}_k^2\}$ are $K_\infty$ -functions, and $\pi \|e(k)\|^2$ is an $K_\infty$ -function, $\eta \|\mu(k)\|^2$ is a $K$ -function. So $L_k$ admits an ISS-Lyapunov function as in Definition 2. By Theorem 1, the dynamic of the identification error is input-to-state stable. From (26) and (41) we know $L_k$ is the function of $e(k)$ and $\mu(k)$. The INPUT corresponds to the second term of (41), i.e., the modelling error $\mu(k)$. The STATE corresponds to the first term of (43), i.e., the identification error $e(k)$. Because the INPUT $e(k)$ is bounded and the dynamic is ISS, the STATE $e(k)$ is bounded. (43) Can be rewritten as

$$\Delta_k \leq -\eta \|e(k)\|^2 + \eta \|\mu(k)\|^2$$

Summing (44) from 1 up to $T$, and by using $L_T > 0$ and $L_I$ is a constant, we obtain

$$L_T - L_1 \leq -\eta \sum_{k=1}^{T} \|e_N(k)\|^2 + T \eta \bar{\mu}$$

$$\eta \sum_{k=1}^{T} \|e_N(k)\|^2 \leq L_T - L_1 \leq L_T + T \eta \bar{\mu}$$

(28) is established.