Stock Price Prediction using Generic Self-Evolving Takagi-Sugeno-Kang (GSETSK) Fuzzy Neural Network

Ngoc Nam Nguyen and Chai Quek, *Member, IEEE*

Abstract— **This paper analyses stock market price prediction based on a Generic Self-Evolving Takagi-Sugeno-Kang (GSETSK) fuzzy neural network. Stock price prediction is a problem that requires *online adaptive systems* with high accuracy performance. The proposed GSETSK framework uses a novel Multidimensional-Scaling Growing Clustering (MSGC) algorithm which mimics the human cognitive process to flexibly generate fuzzy rules without any *a prior* knowledge. MSGC can quickly generate a compact fuzzy rule base from new incoming data and has strong noise-tolerance capability. It empowers the GSETSK network with the ability to effectively address *adaptive and incremental problems* such as *stock price prediction.* Numerical experiments conducted on real-life stock data confirm the validity of the design and the accuracy performance of the GSETSK system.**

I. Introduction

FINANCIAL Engineering is an increasingly popular research area. Trading systems based on computational intelligence techniques has received substantial interest from both researchers and financial traders. Neural networks have been applied extensively to technical financial forecasting because of their ability to learn complex non-linear mapping and self-adaptation for various statistical distributions [1, 2]. More recently, neuro-fuzzy systems (NFS) have attracted growing interest in financial engineering [3]. The main strength of neuro-fuzzy systems is that they can combine the human-like reasoning style of *fuzzy systems* with the connectionist structure and learning ability of *neural networks* [4]. Neuro-fuzzy systems have been recently applied in forecasting stock price [5, 6]. There are two types of NFS, namely the Mamdani model [7] given in (1), which is focused on interpretability, and the Takagi-Sugeno-Kang (TSK) model [8] given in (2), which is focused on accuracy.

$$R_i: \text{ IF } x_1 \text{ is } A_{i,1} \text{ AND } \ldots \text{AND } x_{N_1} \text{ is } A_{i,N_1} \text{ THEN } y \text{ is } B_i \quad (1)$$

$$R_i: \text{ IF } x_1 \text{ is } A_{i,1} \text{ AND } \ldots \text{AND } x_{N_1} \text{ is } A_{i,N_1} \text{ THEN } \\ y = b_o + b_1x_1 + \ldots + b_{N_1}x_{N_1} \quad (2)$$

in which $\mathrm{x} = [x_1,\ldots,x_{N_1}]$ and y are the input vector and the output value, respectively. $A_{i,k}$ represents the membership function of the input label x_k for the i^{th} fuzzy rule; B_i represents the membership function of the output label y for the i^{th} fuzzy rule, $[b_0,\ldots,b_{N_1}]$ represents a set of consequent parameters of $Rule_i$, N_I represents the number of inputs.

The main advantage of the TSK-model over the Mamdani-model is its ability to achieve *higher level of system modeling accuracy* [8]. Because of its high modeling accuracy, TSK-model is the better choice in dealing with sophisticated problems that require high performance accuracy such as stock data modeling and prediction.

Existing TSK fuzzy systems can be generally categorized into two types. The first type is fundamentally TSK fuzzy systems which employ offline learning algorithms. This results in *their inability to learn in an incremental manner*. A popular system of this type is Adaptive-Network-based Fuzzy Inference System (ANFIS) [9]. The ANFIS structure is fixed, and ANFIS parameters are tuned by an offline gradient descent algorithm. ANFIS's fixed structure restrains it from learning incoming data. In additional, ANFIS lacks a balance between good interpretability and high modeling accuracy as the number of rules grows rapidly when dealing with high-dimensional data. In contrast, the second type of TSK fuzzy networks, with the ability to perform online structure learning from the numerical training data and self-tuning the network parameters, is able to learn in adaptive or incremental systems. A simple process that presupposes an even space partitioning of the linguistic labels is suggested in [10]. However, this method is vulnerable to noisy data and requires the number of clusters to be specified before training. A general Euclidean distance is used as an online rule generation criterion in [11]. However the criterion does not take the width of each fuzzy set into consideration. This might cause the fuzzy rule base to be not compact with many redundant rules which lead to bad interpretability and inconsistency. In general, most of the existing TSK fuzzy systems still encountered one or more of the following major problems despite being good modeling tools for nonlinear estimation. They are: 1) requirements for prior knowledge such as the number of fuzzy linguistic labels; 2) vulnerability to noisy training data; and 3) poor balance between good interpretability and high modeling accuracy (meaning more rules needed to achieve accuracy goals). A novel fuzzy neural network architecture termed as Generic Self-Evolving Takagi-Sugeno-Kang (GSETSK) is proposed in [12] to address the weaknesses of existing TSK fuzzy systems mentioned above. This paper examines the application of GSETSK to the stock prediction problem. Experiments conducted on real-life stock data have confirmed the validity of such a design.

The paper is organized as follows. Section II, III briefly discusses about the GSETSK architecture and its adaptation GSETSK[1]. Section IV evaluates the performance of the GSETSK[1] system with two real-world stock dataset.

The authors are with the Centre for Computational Intelligence, School of Computer Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639798

II. GSETSK

This section introduces the structure of a GSETSK. The GSETSK framework can represent all the TSK fuzzy models [8, 13, 14]. It can be mapped to a zero-order GSETSK model by reducing its functional consequents as in (2) into crisp consequents. It can also be mapped to a first-order GSETSK model (termed as *GSETSK^1*), in which its functional consequents are in form of first-order functions of the input vector.

The training cycle of the GSETSK network consists of two phases which are performed sequentially with a single pass of training data. They are: 1) *structure learning* and 2) *parameter learning*. A novel multidimensional-scaling growing clustering (MSGC) technique is employed to automatically partition the input space from the training data. Initially there is no rule in the rule base of the GSETSK network. The fuzzy rules are subsequently formulated by connecting the appropriate input clusters, the normalization nodes and the TSK consequence nodes during the rule formulation phase of the training cycle. Subsequently, the online recursive least-square learning algorithm [15] is employed to tune the parameters of the GSETSK network.

A. *Structure of GSETSK*

Fig.1 presents the proposed network structure, which has a total six layers. The six layers are: Layer I (the input layer), Layer II (the input linguistic layer), Layer III (the rule-base layer), Layer IV (the normalization layer), and Layer V (the consequence layer) and Layer VI (the summation layer). In a single pass of training data, the input nodes in layer I receive the input vector $\mathrm{x} = [x_1,...,x_{N1}]$ and translate it into singleton fuzzy sets and output to the next layer. Subsequently, the input linguistic layer measures *the matching degree* of each fuzzified input with its corresponding linguistic nodes. Each linguistic node in this layer has a Gaussian membership function with its center and width computed by the MSGC technique which will be described later. The rule-base layer combines all incoming signals from the previous layer to generate the corresponding weights. The normalization layer normalizes these weights. Each node of the consequence layer corresponds to one TSK rule. Finally, the summation layer summarizes all the outputs from the consequence layer. The number of neurons in layer k is labeled by $\mathrm{N_k}$, where $\mathrm{k} \in \{1\ldots6\}$.

Each input node $\mathrm{L_i}$, $\mathrm{i} \in \{1\ldots N_1\}$ has a single input. Each output node $\mathrm{O_n}$, $\mathrm{n} \in \{1\ldots N_6\}$ computes a single output denoted by $\mathrm{y_n}$; $\mathrm{y_n} \in Y=\{y_1\ldots y_n...y_{N6}\}$ in which Y is the outputs of the GSETSK with respect to the input vector X. For clarity of subsequent discussions, the variable *i, j, k, l, m, n* are used to refer to arbitrary nodes in layers 1,2,3,4,5 and 6 respectively. The output of a node is denoted as $\boldsymbol{Z}$ with the superscripts denoting its layer and the subscripts denoting its origin, for example Z_i^I is the output of node *i*th in layer 1. All the outputs of a layer are propagated to the inputs of the connecting nodes at the next layer. All the links in GSETSK are of unity link-weight.

Each input node $\mathrm{L_i}$ might connect to different number of input linguistic nodes $\mathrm{J_i}$. Hence the number of layer 2 nodes is $\mathrm{N_2}$ where $N_2 = \sum_{i=1}^{N_1} Ji$. Layer 3 consists of the rule node $\mathrm{R_k}$, where $k \in \{1\ldots N_3\}$. Each layer 3 node $\mathrm{R_k}$ is directly connected to a layer 4 normalization node $\mathrm{N_l}$. Therefore, the number of nodes in layer 3 and layer 4 are equal, $\mathrm{N_3 = N_4}$. Each layer 4 node $\mathrm{N_l}$ is connected to M number of layer 5 consequent nodes. M is equal to the number of layer 6 node, $\mathrm{M= N_6}$. Hence, the number of nodes in layer 5 is $\mathrm{N_4 \times N_6}$. Each output node at layer 6 is a summation node $\mathrm{O_n}$. Each summation node is connected with $\mathrm{N_4}$ layer 5 consequent nodes. The GSETSK model has TSK-type rules as in (2), thus the trainable parameters are found in layer 5 of the model.

Detailed mathematic functions of each layer of GSETSK are presented as follows.

Layer 1: Input Layer

$$Z_i^I = \mu_i(x_i) = x_i \tag{3}$$

where μ_i is the fuzzy membership function of the input fuzzifier node $\mathrm{L_i}$.

Layer 1 nodes are termed as *linguistic nodes*. Each node receives only one input as one dimension of the vectored data input, and output to several nodes of next layer. In this paper, singleton input is employed.

Layer 2: Input Linguistic Layer

$$Z_{i,j}^{II} = \mu_{i,j}(Z_i^I) \tag{4}$$

where $\mu_{i,j}$ is a fuzzy membership function of the *fuzzy linguistic node* $\mathrm{IL_{i,j}}$.

Layer 2 nodes are termed as input-label nodes. They constitute the antecedent of the fuzzy rules in GSETSK. The label $\mathrm{IL_{i,j}}$ denotes a node that is the *j*th label of the *i*th linguistic variable input. The input linguistic layer measures *the matching degree* of each fuzzified input with its corresponding linguistic nodes. Each linguistic node in this layer has a Gaussian membership function with its center and width dynamically computed in the structure learning phase which will be described later. There are many choices for the types of membership functions for use. However Gaussian membership function is chosen for high accuracy. With the use of Gaussian membership function, (4) becomes

$$Z_{i,j}^{II} = e^{-\frac{(Z_i^I - m_{ij})^2}{\sigma_{ij}^2}} \tag{5}$$

where m_{ij} and σ_{ij} are, respectively, the center and the width of the Gaussian membership function of the *j*th label of the *i*th linguistic variable input $\mathrm{x_i}$.

Layer 3: Rule Layer

$$Z_k^{III} = f_k(Z_{i,j}^{II}), i \in \{1...N_1\} \tag{6}$$

where f_k is a generic activation function for the rule R_k. Each node in rule-base layer represents a single Sugeno-type fuzzy rule and is termed as a *rule node*. A rule node receives inputs from the respective input-label nodes and computes the firing strength of the rule it represents.

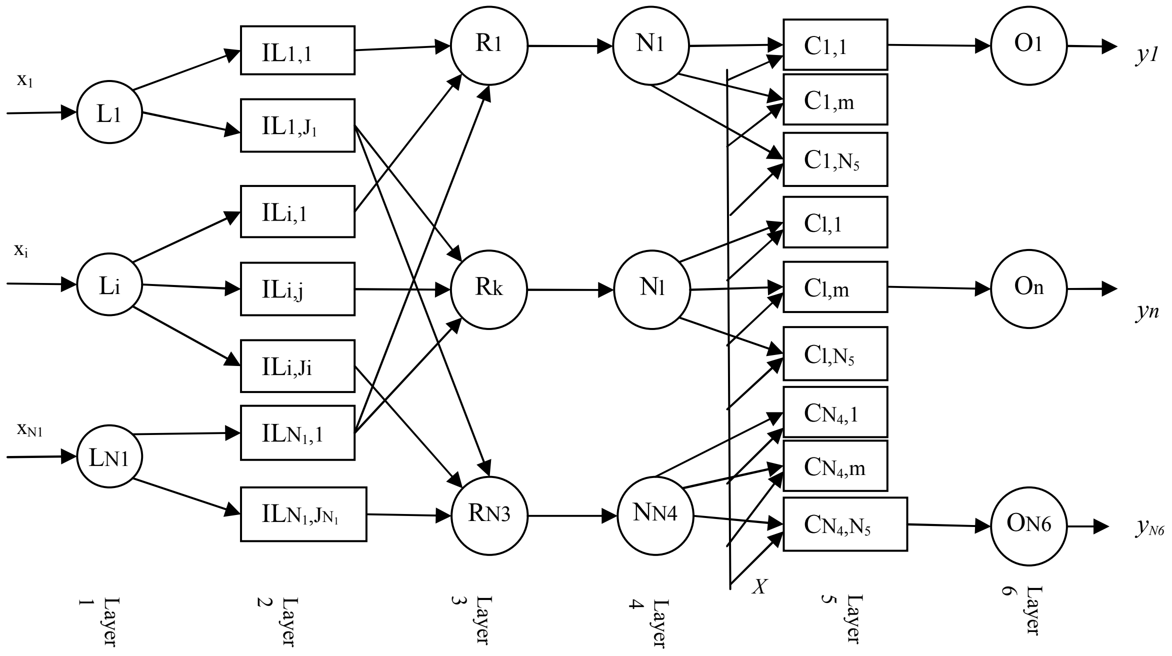


Fig 1. Structure of the GSETSK network

The activation function f_k in (6) is a generic function but in this paper, the AND operation is chosen. Therefore (6) becomes as follow:

$$Z_k^{III} = f_k(Z_{i,j}^{II}) = \prod_i Z_{i,j}^{II} \qquad (7)$$

where $Z_{i,j}^{II}$ is the output of the jth label of the ith linguistic variable input x_i that connects to the kth rule.

Layer 4: Normalization layer

$$Z_l^{IV} = \frac{Z_k^{III}}{\sum_{k=1}^{N_3} Z_k^{III}} = \mu_l \qquad (8)$$

where μ_l is the normalize firing strength.

Layer 5: Consequence Layer

$$Z_{l,m}^{V} = Z_l^{IV} f_{l,m}(X) \qquad (9)$$

where $f_{l,m}(X)$ is a function of consequence node $C_{l,m}$

Layer 6: Summation Layer

$$Z_n^{VI} = \sum_{l=1}^{N_5} Z_{l,m}^{V} \qquad (10)$$

where $Z_{l,m}^{V}$ is the output of consequence node $C_{l,m}$ in layer 5.

GSETSK is a generic framework that can be easily extended to all the TSK models. ANFIS model [9] is a special case of GSETSK which has a product inference scheme at layer 3, a first-order linear function at layer 5, and a single output at layer 6.

B. Structural Learning Phase

GSETSK employs a novel clustering technique known as *Multidimensional-scaling Growing Clustering (MSGC)*. MSGC enables the GSETSK network to grow without *prior* knowledge of the number of clusters and empowers the network the ability to handle noisy/spurious data. The proposed clustering algorithm also helps to minimize the number of rules created and to minimize the number of fuzzy sets on the universe of discourse of each input variable.

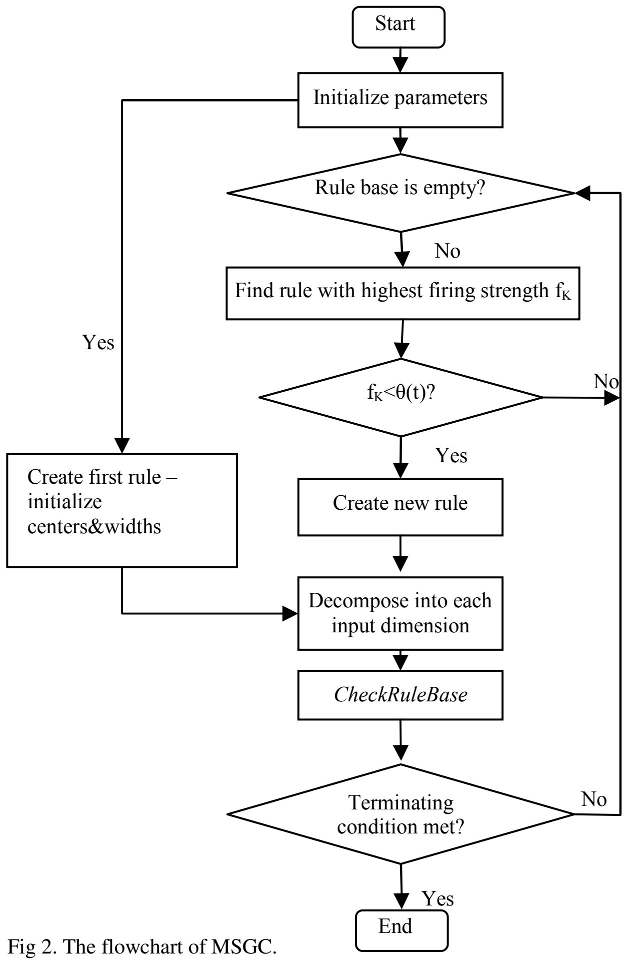
The initial rule base in GSETSK is empty. In a single pass of training data, the GSETSK network models the problem domain by formulating the fuzzy rules by partitioning the input space as well as subsequently deriving the fuzzy sets on the universal of discourse of each input variable using MSGC. In MSGC, each rule in the network corresponding to a cluster is identified in the multidimensional input space. After a cluster is created, the corresponding 1-D membership function for each input variable is determined by decomposing the multidimensional cluster. The multidimensional scaling approach in MSGC technique is inspired by the human cognitive process models [16] in which the spatial representation of a stimulus domain generated by multidimensional scaling facilitates the operation of the fundamental cognitive process within that domain.

The working flowchart of the MSGC is illustrated in Fig. 2. To determine whether a new rule is created or not, the rule firing strength in (7) is used. For an incoming input vector x, the strength a rule is fired is determined by the degree the vector x belongs to the corresponding cluster. The firing strength is computed as in (11). For each input vector x, the maximum firing strength is found as in (12), where u(t) is the number of existing clusters at time t.

$$f_k(x) = \prod_i Z_{i,j}^{II} = e^{\frac{-(x_1-m_{1j})^2}{\sigma_{1j}^2}} \ldots e^{\frac{-(x_i-m_{ij})^2}{\sigma_{ij}^2}} \ldots e^{\frac{-(x_{N1}-m_{N1j})^2}{\sigma_{N1j}^2}} \qquad (11)$$

$$K = \arg \max_{1<k<u(t)} f_k(x) \qquad (12)$$

where K indicates that the Kth rule achieves the maximum firing strength among all the existing fuzzy rules in the rule base.


Start
Initialize parameters
Rule base is empty?
No
Find rule with highest firing strength f_K
Yes
f_K<θ(t)?
No
Yes
Create first rule – initialize centers&widths
Create new rule
Decompose into each input dimension
CheckRuleBase
Terminating condition met?
No
Yes
End
Fig 2. The flowchart of MSGC.

A new rule is created if $f_K < \theta(t)$, in which $\theta(t) \in (0,1)$ is a decaying predefined threshold. $\theta(t)$ also controls the number of rules created. The higher value of $\theta(t)$, the more rules created. Each rule corresponds to a multidimensional cluster in the input space. After a cluster is created, the corresponding 1-D membership function for each input variable is determined by decomposing the multidimensional cluster as in (13) and (14).

$$m[u(t)+1, j] = x_j \qquad (13)$$

$$\sigma[u(t)+1, j] = -\eta \ln(f_K^j) \qquad (14)$$

where $m[u(t)+1, j]$, $\sigma[u(t)+1, j]$ are , respectively, the center and width of the membership function that the new cluster projects into the input dimension *j*th; x_j is the *j*th component of the input vector x; f_K^j is the *j*th component of the maximum firing strength f_K which is $e^{\frac{-(x_i-m_{ij})^2}{\sigma_{ij}^2}}$ and $\eta \geq 0$ is a predefined constant that determines the overlap degree between two arbitrary clusters. The nature logarithm $\ln(f_K^j)$ is directly proportional to the minimum distance from the new input vector x to the existing clusters.

The MSGC technique employed in GSETSK is sufficient to maintain a consistent and compact rule base by performing two procedures in the function *CheckRuleBase*. First, to minimize the number of fuzzy sets of each input variable, the check of similarities between the newly projected membership function and the existing ones in each input dimension is performed. This is to prevent adding a new membership function that highly overlaps with existing membership functions. To determine the similarity measure of two Gaussian fuzzy sets, a fuzzy subsethood measure [17] is computed. The fuzzy subsethood measure defines the degree that fuzzy set A is a subset of fuzzy set B as

$$S(A,B) = \frac{|A \cap B|}{|A|} \qquad (15)$$

If the min operator is used for the intersection operation, then (15) becomes

$$S(A,B) = \frac{\int_{x\in U} \min(\mu_A(x), \mu_B(x))}{\int_{x\in U} \mu_A(x)} \qquad (16)$$

Or can be approximated as (17)

$$S(A,B) = \frac{\max_{x\in U}(\min(\mu_A(x), \mu_B(x)))}{\max_{x\in U}(\mu_A(x))} \qquad (17)$$

Denote $\mu(m_i, \sigma_i)$ as the Gaussian membership with center m_i and width σ_i; $\mu(m_{new-i}, \sigma_{new-i})$ as the membership function of the newly projected membership function in *i*th input dimension. Denote p$_i$ as the current number of partitions of the *i*th input variable. The function *CheckRuleBase* is performed as follow.

```
Function CheckRuleBase
  Begin
    For each input dimension i, perform similarity check as
    follow:
        {
      maxDegree(i,t) = $\max_{1<l<p_i} S[\mu(m_{new-i}, \sigma_{new-i}), \mu(m_{[l,i]}, \sigma_{[l,i]})]$ (18)
          where maxDegree(i,t) is the maximum matching
          degree between the newly created membership
          function with the existing membership functions in
          input dimension i at time t
        IF maxDegree(i,t)>thresA
          Replace the newly created membership function with
          the closest existing one.
        ELSE IF maxDegree(i,t)>thresB
          MergeMembership
        ELSE accept the newly created membership
        function; set $p_i = p_i+1$ }
      Check whether the new rule exists in the rule base.
  End
```

In the above algorithm, *thresA* and *thresB* (*thresA>thresB*) are two predefined similarity criterions.

The second procedure in the function *CheckRuleBase* is implemented in *MergeMembership* function to merge two highly overlapping membership functions into a Gaussian function with larger width. This happens when the newly created membership function is highly similar to one of the existing membership functions. However, to maintain the meaning of a membership function and to prevent a membership function from expanding too many times, the

MergeMembership utilizes the following predefined parameters:

a) Expansion Allowance ε: Any two arbitrary clusters *(m₁,σ₁)* and *(m₂,σ₂)* are allowed to merge if and only if

$$|m_1 - m_2| < \varepsilon \times \min(\sigma_1, \sigma_2) \quad (19)$$

where *ε* (in percent) establishes a relative threshold for the distance between two centers of two clusters. The smaller *ε* is, the smaller the distance between two cluster's centers must be so that two clusters can be merged. *ε* works together with the Willingness Parameter below to maintain the integrity of the input clusters and the fuzzy labels they present. *ε* is set to 50 (percent) as default.

b) Willingness Parameter WP: This parameter indicates the willingness of a membership function to expand/merge with another membership function. WP decreases after each time the membership function performs an expansion. The membership function will not be allowed to merge when its WP reaches 0. The parameter WP maintains the semantic meaning of a membership function by preventing its width from growing overly large. WP is set to 0.5 as default. The rate of decreases of WP depends on the similarity measure between two arbitrary membership functions *(m₁,σ₁)* and *(m₂,σ₂)* as follow

$$WP = WP - (1.5 - WP) \times (1 - SM(\mu(m_1,\sigma_1), \mu(m_2,\sigma_2)) \quad (20)$$

where $SM(\mu(m_1,\sigma_1), \mu(m_2,\sigma_2) \in [0,1)$ is the similarity measure between *(m₁,σ₁)* and *(m₂,σ₂)* as in (17). The smaller the similarity measure between *(m₁,σ₁)* and *(m₂,σ₂)* as in (17) is, the faster WP decreases. It must be noted that the term $(1.5 - WP)$ in (20) ensures that WP always can converge to 0 or less. The smaller WP is, the faster WP will decrease.

Consider that a Gaussian membership function can be approximated by an isosceles triangular with unity height and the length of bottom edge is $2\sigma\sqrt{\pi}$ [18], then the width and center of the new cluster after merging two arbitrary cluster *(m₁,σ₁)* and *(m₂,σ₂)* *(m1>m2)* is as follow:

$$\sigma_{new} = \frac{m_1 - m_2 + (\sigma_1 + \sigma_2)\sqrt{\pi}}{2\sqrt{\pi}} \quad (21)$$

$$m_{new} = [m_1 + m_2 + (\sigma_1 + \sigma_2)\sqrt{\pi}]/2 \quad (22)$$

Merging two membership functions creates a new cluster with larger width which can cover larger region. This lead to fewer rules and fewer fuzzy sets in each dimension. This can also lead to a better interpretability in a neuro-fuzzy system. This online clustering technique MSGC can ensure a consistent and compact rule base in the GSETSK network. It also empowers GSETSK with high noise tolerance capability. Plus it can overcome the main drawback of the traditional clustering techniques which is the requirement of prior knowledge such as the number of classes. Also, it can address the *stability–plasticity dilemma* [19], meaning that new knowledge could be incorporated into the GSETSK system after the system has finished learning. This empowers GSETSK the ability to self-evolve with changing environments.

In the current implementation, the selection of the parameters η, *θ*, *thresA* and *thresB* is heuristic and varies with different simulations. There are several guidelines to help in selecting suitable values for these parameters. A lower value of η means less overlap between rules and a smaller initial width assigned to each membership function. High values of *thresA* and *thresB* result in more membership functions created. The higher the value of θ, the more rules are created.

III. GSETSK¹

GSETSK¹ (first order GSETSK model) is a special case of the GSETSK framework with first-order polynomial functions at layer 5 of the framework. The specific output of an arbitrary consequence node $C_{l,m}$ at layer 5 is described as in (23):

$$Z^{V}_{l,m} = Z^{IV}_{l} f_{l,m}(X)$$
$$f_{l,m}(X) = b_{0,(l,m)} + \ldots + b_{i,(l,m)}x_i + \ldots + b_{N1,(l,m)}x_{N1} \quad (23)$$

where $B_{(l,m)} = [b_{0,(l,m)}, \ldots, b_{i,(l,m)}, \ldots, b_{N1,(l,m)}]^T$ is the parameter vector of the consequence node $C_{l,m}$

The output nodes at layer 6 take the form of (24).

$$y_n = Z^{VI}_{n} = \sum_{l=1}^{N4} Z^{V}_{l,m} = \sum_{l=1}^{N4} \mu_l f_{l,m}(X)$$
$$= \sum_{l=1}^{N4} \mu_l [b_{0,(l,m)} + \ldots + b_{i,(l,m)}x_i + \ldots + b_{N1,(l,m)}x_{N1}] \quad (24)$$

Where μ_l is the normalized firing strength at the *l*th normalization node.

Assuming that the GSETSK¹ network models a system with P number of training samples *(X⁽¹⁾, d⁽¹⁾), …, (X⁽ᵖ⁾, d⁽ᵖ⁾),…, (X⁽ᴾ⁾, d⁽ᴾ⁾)* in which

$X^{(p)}$ *p*th input vector, $p \in \{1 \ldots P\}$

$d^{(p)}$ *p*th desired output value

Then (24) can be arranged in a matrix form as in (25).

$$A\text{x}B = D \quad (25)$$

To visualize (25), assuming that the GSETSK¹ has only one output y, two inputs x_1 and x_2, and only 2 rules, then (25) becomes

$$\begin{bmatrix} \mu_1^{(1)} & \mu_1^{(1)}x_1^{(1)} & \mu_1^{(1)}x_2^{(1)} & \mu_2^{(1)} & \mu_2^{(1)}x_1^{(1)} & \mu_2^{(1)}x_2^{(1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mu_1^{(p)} & \mu_1^{(p)}x_1^{(p)} & \mu_1^{(p)}x_2^{(p)} & \mu_2^{(p)} & \mu_2^{(p)}x_1^{(p)} & \mu_2^{(p)}x_2^{(p)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mu_1^{(P)} & \mu_1^{(P)}x_1^{(P)} & \mu_1^{(P)}x_2^{(P)} & \mu_2^{(P)} & \mu_2^{(P)}x_1^{(P)} & \mu_2^{(P)}x_2^{(P)} \end{bmatrix} \text{x}$$

$$\begin{bmatrix} b_{0,(1,1)} \\ b_{1,(1,1)} \\ b_{2,(1,1)} \\ b_{0,(2,1)} \\ b_{1,(2,1)} \\ b_{2,(2,1)} \end{bmatrix} = \begin{bmatrix} d^{(1)} \\ \vdots \\ d^{(p)} \\ \vdots \\ d^{(P)} \end{bmatrix} \quad (26)$$

In (25) the dimensions of matrices *A*, *B* and *D* are $P \times R$, $R \times 1$ and $P \times 1$ respectively where *R* is the total number of linear parameters and is determined as in (27).

$$R = (N_1 + 1) \times N_3 \quad (27)$$

where N_1 is the number of inputs and N_3 is the number of fuzzy rules (in layer 3 of GSETSK).

Equation (25) is usually an over-determined problem which has the number of training samples greater than the number of parameters to be defined and has no exact solution.

To address the problem, *the recursive linear-least-squares (RLS) algorithm* (a special case of the Kalman filter algorithm [15]) is employed to tune the linear parameters in GSETSK[1] . This iterative linear least-square estimation (LSE) [15] process is able to quickly model the network (identify the linear parameters) online in each single pass of training data.

Denote a_p as the *p*th row of the matrix *A*, then using RLS, *B* can be iteratively estimated as

$$B_{p+1} = B_p + C_{p+1} a_{p+1}^T (d^{(p+1)} - a_{p+1} B_p)$$
$$C_{p+1} = C_p - \frac{C_p a_{p+1}^T a_{p+1} C_p}{1 + a_{p+1} C_p a_{p+1}^T} \qquad (28)$$

With initial condition of $B_0 = 0$ and $C_0 = \lambda\mathbf{I}$ where λ is a large positive number and $\mathbf{I}$ is the identity matrix of dimension $R \times R$.

The RLS empowers GSETSK[1] with high-accuracy online learning performance. However it encounters a major problem, being that it has high computational and space complexities [10]. A first-order TSK fuzzy model of N_1 inputs and N_3 rules will need a *S* matrix of dimension of $(N_1+1)^2 \times N_3^2$ and a computational cost of $O((N_1+1)^2 \times N_3^2)$. In order to reduce the space complexity as well as the computation cost and to enhance the training speed, a localized version of recursive least-squares algorithm is employed in GSETSK[1]. Instead of considering all the fuzzy rules that affect the outputs, only one rule is considered at one time. More details about the localized version can be found in [12].

IV. Experimental Results

This section focuses on investigating the effectiveness of using the GSETSK[1] network in the modeling and forecasting of real-life stock prices. The accuracy performance of GSETSK[1] is benchmarked against other established Mamdani's and TSK's models using the MSE and Pearson correlation coefficient [20]. Two sets of stock price data are used: Singapore Exchange Limited (SGX) and Wells Fargo & Company (WFC).

A. SGX Stock Data

The first experiment was conducted on SGX (Singapore Exchange Limited) stock which is listed on the Mainboard in Singapore Exchange. Daily stock prices from 15 Nov 2005 to 20 October 2009 on the counter S68.SI were collected from the Yahoo Finance website. In this experiment, 500 data samples were used as the training dataset and 500 data samples were used as the testing dataset. The Pearson product-moment correlation value (denoted as $\Re$) [20] is used to compute the accuracy of the predicted stock movement obtained using the proposed GSETSK[1] network. Table I shows the experimental results of the recall on the training data. Table II shows the results of the prediction on the test data. The predictions made by the GSETSK[1] network are shown in Fig 3.

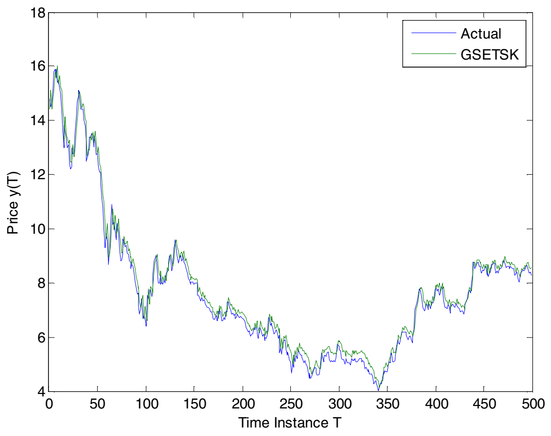


Fig 3. Predictive results of SGX stock test data set using GSETSK[1] (L).

TABLE I
BENCHMARKING AGAINST OTHER MODELS ON RECALLING THE SGX STOCK TRAINING DATA

Network	Recall Result		
	MSE	R	No. of Rules
FCMAC-Yager	0.4377	0.9743	115
ANFIS	0.0426	0.9975	16
DENFIS	0.0736	0.9963	10
GSETSK[1](G)	0.0579	0.9966	10
GSETSK[1] (L)	0.0579	0.9965	10

TABLE II
BENCHMARKING AGAINST OTHER MODELS ON PREDICTING THE SGX STOCK TEST DATA

Network	Predict Result		
	MSE	R	No. of Rules
FCMAC-Yager	0.9800	0.9389	95
ANFIS	3.0300	0.8514	81 (20 epochs)
DENFIS	0.1681	0.9907	10
GSETSK[1] (G)	0.1003	0.9947	10
GSETSK[1] (L)	0.1003	0.9947	10

GSETSK[1](G) denotes the GSETSK[1] network that employs the global version of recursive least-square algorithm. GSETSK[1](L) denotes the GSETSK[1] network that employs the localized version. Other models to be evaluated are FCMAC-Yager [21], ANFIS [9] and DENFIS [11]. FCMAC-Yager is an online Mamdani's model, meanwhile ANFIS and DENFIS are both TSK models. From Fig. 3, it can be observed that GSETSK[1] was able to closely follow the movement of the stock prices. Table I shows that GSETSK[1] and ANFIS yield results that are superior than the remaining in term of accuracy (lowest MSE and $\Re$) in the recall experiment. ANFIS yields slightly better result than GSETSK[1] with the MSE=0.0426 compared to MSE=0.0579 of GSETSK[1]. However ANFIS must be trained with many epochs and it employs offline algorithms therefore it is not suitable for stock price prediction problem. GSETSK[1] is an online learning network but it can yield results that are comparable to ANFIS and yield better results than DENFIS (which is an online learning system). With the same number of rules, GSETSK[1] yields lower MSE and better $\Re$ than

DENFIS. It also outperforms the Mamdani-based network FCMAC-Yager in term of accuracy and the number of rules.

Table II shows that GSETSK[1] outperforms all other models. ANFIS shows good results in the recall experiment but it yields worst results in the prediction experiment. And again, GSETSK[1] yield results that are superior than the rest in term of accuracy and the number of rules.

B. Wells Fargo & Company Stock Data

The second experiment was conducted on WFC (*Wells Fargo & Company Stock*) stock which is listed on the Mainboard in New York Stock Exchange (NYSE). Daily stock prices from 19 July 2006 to 22 September 2009 under the counter NYSE:WFC were collected from the Yahoo Finance website. In this experiment, 400 data samples were used as the training dataset and 400 data samples were used as the testing dataset. The MSE and the Pearson product-moment correlation value (denoted as $\Re$) are used to compute the accuracy of the predicted stock movement obtained using the proposed GSETSK[1]network. The predictions made by the GSETSK[1] network are shown in Fig 4. Table III shows the experimental results of the recall on the training data. Table IV shows the results of the prediction on the test data. Only TSK-models are to be evaluated in this experiment including DENFIS [11], ANFIS [9] and GSETSK[1].

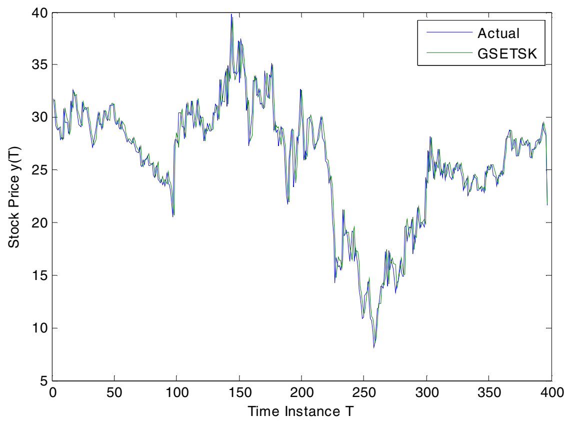


Fig 4. Predictive results of WFC stock test data set using GSETSK[1] (L).

TABLE III
BENCHMARKING AGAINST OTHER MODELS ON RECALLING THE WFC STOCK TRAINING DATA

Network	Recall Result		
	MSE	R	No. of Rules
ANFIS	0.8813	0.9932	32 (20 epochs)
DENFIS	4.4189	0.9742	8
GSETSK(G)	2.085	0.9839	6
GSETSK(L)	2.090	0.9839	6

TABLE IV
BENCHMARKING AGAINST OTHER MODELS ON PREDICTING THE WFC STOCK TEST DATA

Network	Predict Result		
	MSE	R	No. of Rules
ANFIS	15.5189	0.7361	32 (20 epochs)
DENFIS	14.5816	0.8982	7
GSETSK(G)	2.1379	09673	6
GSETSK(L)	2.1383	0.9666	6

Fig.4 shows that GSETSK[1] could closely model and predict the stock price movement. Table III and Table IV shows that ANFIS again yields best results in the recall experiment but worst results in the prediction experiment. This shows that ANFIS is vulnerable to noisy data even though ANFIS employs offline learning algorithms. GSETSK[1]can yield superior results in both experiments in term of accuracy and the number of rules. With lesser number of rules, GSETSK[1]can perform better than DENFIS with lower MSE and better R.

These two experiments show that GSETSK[1] can give promising results in solving stock price prediction problems. GSETSK[1] can deal with new incoming data and achieve high accuracy performance with a compact rule base.

III. CONCLUSION

The GSETSK models are suitable for *adaptive system learning* and *nonlinear system estimation*. The key strength of the GSETSK framework is the flexibility of adopting different formulations and settings to address different issues that most of the other existing TSK neural fuzzy systems encountered. The issues are: 1) requirements for prior knowledge such as the number of fuzzy linguistic labels 2) vulnerability to noisy training data 3) poor balance between interpretability and modeling accuracy.

The GSETSK network employs a novel online clustering technique known as MSGC (Multidimensional-scaling growing clustering) to compute the bell-shaped (Gaussian) fuzzy sets during its structural learning. MSGC mimics the human cognitive process to flexibly generate fuzzy rules without any *a prior* knowledge. MSGC has good noise-tolerance capability and it also helps to *reduce the number of fuzzy sets, fuzzy rules* and *improve the interpretability of the GSETSK network*. In addition, it enables the GSETSK network to *learn in adaptive and incremental systems*. GSETSK can effectively address the stability–plasticity dilemma. The first-order GSETSK model (GSETSK[1]) which assumes the *recursive linear-least-squares (RLS) algorithm.* is appropriate for modeling a system with high accuracy. A localized version of RLS employed in GSETSK[1] significantly reduce computational cost and increase the training speed even further.

Two experiments using real-world stock dataset were conducted to evaluate the performance of the proposed GSETSK[1]. The performance of the GSETSK[1] is evaluated against other published results. The results show that GSETSK[1] can achieve superior results when dealing with stock data modeling and prediction problem which requires online learning abilities and high performance accuracy.

REFERENCE

[1] R. R. Trippi and E. Turban, *Neural Networks in Finance and Investing: Using Artificial Intelligence to Improve Real World Performance*: Chicago: Probus Pub. Co., 1993.

[2] M. Adya and F. Collopy, "How effective are neural networks at forecasting and prediction? A review and evaluation," *Journal of Forecasting,* vol. 17, pp. 481-495, 1998.

[3] R. R. Trippi and J. K. Lee, *Artificial Intelligence in Finance & Investing: State-Of-The-Art Technologies for Securities Selection and Portfolio Management*: Chicago: Irwin Professional Publishing, 1996.

[4] S. Mitra and Y. Hayashi, "Neuro-fuzzy rule generation: survey in soft computing framework," *IEEE Trans. Neural Netw.,* vol. 11, pp. 748-768, 2000.

[5] K. N. Pantazopoulos, L. H. Tsoukalas, N. G. Bourbakis, M. J. Brun, and E. N. Houstis, "Financial prediction and trading strategies using neuro-fuzzy approaches," *IEEE Transactions on Systems, Man and Cybernetics, Part B,* vol. 28, pp. 520-531, 1998.

[6] E. W. Saad, D. V. Prokhorov, and D. C. Wunsch II, "Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks," *IEEE Transactions on Neural Networks,* vol. 9, pp. 1456-1470, 1998.

[7] L. C. S. G. Lin C.T., *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems*. New Jersey: Prentice Hall PTR, 1996.

[8] T. Takagi and M. Sugeno, "Fuzzy Identification of Systems and its Applications to Modeling and Control," *IEEE Trans. Systs., Man, Cybern.,* vol. 15, pp. 116-132, 1985.

[9] R. Jang, "ANFIS: adaptive-network-based fuzzy inference system," *IEEE Trans. Systs., Man, Cybern. B,* vol. 23, pp. 665-685, 1993.

[10] K. H. Quah and C. Quek, "FITSK: online local learning with generic fuzzy input Takagi-Sugeno-Kang fuzzy framework for nonlinear system estimation," *IEEE Trans. Systs., Man, Cybern. B,* vol. 36, pp. 166-178, 2006.

[11] N. K. Kasabov and Q. Song, "DENFIS: Dynamic evolving neural-fuzzy inference system and its application for time-series prediction," *IEEE Trans. Systs., Man, Cybern. B,* vol. 10, pp. 144-154, 2002.

[12] S. Nguyen and C. Quek, "GSETSK - A Generic Self Evolving TSK Fuzzy Neural Network," *IEEE Trans. Neural Netw.,* submitted for publication.

[13] M. Sugeno and G. T. Kang, "Structure Identification of Fuzzy Model," *Fuzzy Sets Syst.,* vol. 28, pp. 15-33, 1988.

[14] T. Takagi and M. Sugeno, "Derivation of fuzzy control rules from human operator's control action," *in: Proc. of IFAC Symp. on Fuzzy Information, Knowledge Representation and Decision Analysis,* pp. 55-60, 1983.

[15] C. T. Lin and C. S. G. Lee, *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems*. New Jersey: Prentice Hall PTR, 1996.

[16] D. L. Michael, "Determining the dimensionality of multidimensional scaling representations for cognitive modeling," *J MATH PSYCHOL,* vol. 45, pp. 149 - 166, 2001.

[17] J. Yen and R. Langari, *Fuzzy Logic: Intelligence, Control and Information*. Englewood Cliffs, NJ: Prentice Hall, 1999.

[18] C. T. Lin and C. S. G. Lee, "Real-time supervised structure/parameter learning for fuzzy neural network," *IEEE Int. Conf. Fuzzy Syst.,* vol. 1283-1291, 1992.

[19] C. Quek and W. L. Tung, "A novel approach to the derivation of fuzzy membership functions using the Falcon-MART architecture," *Pattern Recognit. Lett.,* vol. 22, pp. 941-958, 2001.

[20] R. N. Goldman and J. S. Weinberg, *Statistics: An Introduction*. NJ: Prentice-Hall, 1985.

[21] J. Sim and W. L. Tung, "FCMAC-Yager: A Novel Yager-Inference-Scheme-Based Fuzzy CMAC," *IEEE Trans. Neural Netw.,* vol. 17, pp. 1394-1410, 2006.