Dynamic Fuzzy Neural Networks—A Novel Approach to Function Approximation

Shiqian Wu and Meng Joo Er

***Abstract*—In this paper, an architecture of dynamic fuzzy neural networks (D-FNN) implementing Takagi–Sugeno–Kang (TSK) fuzzy systems based on extended radial basis function (RBF) neural networks is proposed. A novel learning algorithm based on D-FNN is also presented. The salient characteristics of the algorithm are: 1) hierarchical on-line self-organizing learning is used; 2) neurons can be recruited or deleted dynamically according to their significance to the system's performance; and 3) fast learning speed can be achieved. Simulation studies and comprehensive comparisons with some other learning algorithms demonstrate that a more compact structure with higher performance can be achieved by the proposed approach.**

***Index Terms*—Dynamic structure, function approximation, fuzzy neural networks, hierarchical on-line self-organizing learning, TSK fuzzy reasoning.**

I. Introduction

System modeling has played an important role in many engineering fields such as control, pattern recognition, communications, and so on. The main idea in conventional approaches is to find a global function of systems based on mathematical tools. However, it is well known that these methods have been found to be unsatisfactory in coping with ill-defined and uncertain systems. In order to circumvent these problems, model-free approaches using either fuzzy logic or neural networks have been proposed. Functionally, a fuzzy system or a neural network can be described as a function approximator. More specifically, they aim at obtaining an approximation of an unknown mapping $f: \Re^r \rightarrow \Re^s$ from sample patterns drawn from the function f. Theoretical investigations have revealed that neural networks and fuzzy inference systems are universal approximators [1], [2], i.e., they can approximate any function to any prescribed accuracy provided that sufficient hidden neurons or fuzzy rules are available. Recently, the synergy of the two paradigms has given rise to a rapidly emerging field, fuzzy neural networks (FNN), that is intended to capture the advantages of both fuzzy logic and neural networks. Numerous results have shown that FNN could offer a viable approach for system modeling [3]–[8].

In existing FNN's, almost all these systems are trained by the backpropagation (BP) algorithm [3]–[8]. The major drawback of the BP algorithm is the slow speed of learning because of the entrapment of local minima so that these systems cannot be implemented in real-time processes. Although some special FNN's (fuzzy neurons and fuzzy weights) have been presented, the typical approach of FNN's is to build standard neural networks which are designed to approximate a fuzzy system through the structure of neural networks. The main idea is as follows. Assume that some particular membership functions have been defined and the number of rules is determined *a priori* according to either expert knowledge or trial and error method. Next, the parameters are modified by the hybrid [3] or BP learning algorithm [4]–[8]. Nevertheless, in most FNN's, structure identification is still difficult. In [3], the structure of the adaptive-network-based fuzzy inference system (ANFIS) is mainly determined by expert knowledge. In [4], the author proposed the self-organizing learning scheme for structure learning. A hierarchically self-organizing approach was used in [5]. In [6], the structure is determined by clustering the input space. In [7], a simple and effective method whereby the structure is identified by input–output data pairs rather than by only input data was presented. A new concept fuzzy curve for determining structure of FNN's was proposed in [8]. As this approach is a heuristic one, it is not easy to determine the architecture and the performance.

Manuscript received November 22, 1998; revised January 5, 2000. This work was supported by the Nanyang Technological University Applied Research Grant RG 38/95. This paper was recommended by Associate Editor P. Willett.

The authors are with the Intelligent Machines Research Laboratory, School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798 (e-mail: p144623148@ntu.edu.sg; emjer@ntu.edu.sg).

Publisher Item Identifier S 1083-4419(00)02974-5.

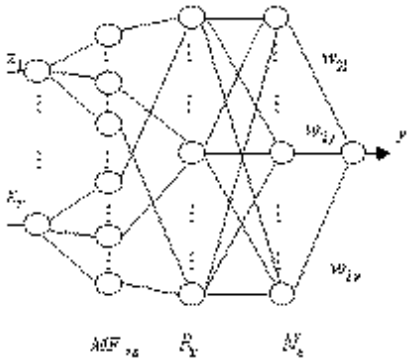

Fig. 1. Architecture of the D-FNN.

This paper presents an architecture of D-FNN performing TSK fuzzy systems based on the extended RBF neural networks and a novel learning algorithm based on the D-FNN. The key idea of the algorithm is the following. The system starts with no hidden units. Then neurons can be recruited or deleted dynamically according to their significance to system performance so that not only the parameters can be adjusted, but also the structure can be self-adapted simultaneously. In the algorithm, a simple hierarchical learning approach is proposed. By using the pruning technology, significant neurons are selected so that a parsimonious structure with high performance can be achieved.

This paper is organized as follows. Section II presents the architecture of the D-FNN. A novel learning algorithm is described in detail in Section III. Section IV presents some simulation results and comparative studies with other learning algorithms. Finally, conclusions are drawn in Section V.

II. Architecture of the Dynamic Fuzzy Neural Networks

The new structure based on extended RBF neural networks to perform TSK model-based fuzzy system is shown in Fig. 1.[1] Comparing with standard RBF neural networks, the term "extended RBF neural networks" implies that: 1) there are more than three layers; 2) no bias is considered; and 3) the weights may be a function instead of a real constant.

Layer 1: Each node in layer 1 represents an input linguistic variable.

Layer 2: Each node in layer 2 represents a membership function (MF) which is in the form of Gaussian functions:

$$\mu_{ij}(x_i) = \exp\left[-\frac{(x_i - c_{ij})^2}{\sigma_j^2}\right] \qquad i = 1, 2, \cdots, r,\ j = 1, 2, \cdots, u \tag{1}$$

[1]For ease of understanding, we shall consider multi-input and single-output systems in the following analysis. However, the results can be readily applied to multi-input and multi-output systems. Also, we shall use small letters to denote scalars and capital letters for matrices and vectors.

where

μ_{ij} jth membership function of x_i;
c_{ij} center of the jth Gaussian membership function of x_i;
σ_j width of the jth Gaussian membership function of x_i;
r number of input variables;
u number of membership functions.

Layer 3: Each node in layer 3 represents a possible IF-part for fuzzy rules. For the jth rule R_j, its output is

$$\phi_j = \exp\left[-\frac{\sum_{i=1}^{r}(x_i - c_{ij})^2}{\sigma_j^2}\right] = \exp\left[-\frac{\|X - C_j\|^2}{\sigma_j^2}\right] \qquad j = 1, 2, \cdots, u \tag{2}$$

where $X = (x_1 \cdots x_r)$ and C_j is the center of the jth RBF unit. From (2) we can see that each node in this layer also represents an RBF unit.

Layer 4: We refer to these nodes as N (normalized) nodes. Obviously, the number of N nodes is equal to that of R nodes. The output of the N_j node is

$$\psi_j = \frac{\phi_j}{\sum_{k=1}^{u}\phi_k} = \frac{\exp\left[-\frac{\|X - C_j\|^2}{\sigma_j^2}\right]}{\sum_{k=1}^{u}\exp\left[-\frac{\|X - C_k\|^2}{\sigma_k^2}\right]} \qquad j = 1, 2, \cdots, u. \tag{3}$$

Layer 5: Each node in this layer represents an output variable as the summation of incoming signals

$$y(X) = \sum_{k=1}^{u} w_{2k} \cdot \psi_k = \frac{\sum_{k=1}^{u} w_{2k}\ \exp\left[-\frac{\|X - C_k\|^2}{\sigma_k^2}\right]}{\sum_{k=1}^{u}\exp\left[-\frac{\|X - C_k\|^2}{\sigma_k^2}\right]} \tag{4}$$

where y is the value of an output variable and w_{2k} is the weight of each rule.

For the TSK model

$$w_{2j} = k_{j0} + k_{j1}x_1 + \cdots k_{jr}x_r \qquad j = 1, 2\ \cdots, u. \tag{5}$$

When the weights are considered as real constants, i.e.,

$$w_{2j} = c_j \qquad j = 1, 2\ \cdots, u \tag{6}$$

we have the simplified fuzzy model or the S model.

III. A Novel Learning Algorithm of the D-FNN

As indicated in Section II, each node in layer 3 represents an RBF unit. If the number of RBF units is required to be identified, we cannot choose the D-FNN structure *a priori*. This leads us to develop a new type of learning algorithm for the D-FNN which is capable of automatically determining the RBF units for system performance.

A. Criteria of Neuron Generation

1) Factors of Neuron Generation:

a) System Errors: The error criterion can be described as follows: for the ith observation $(X_i,\ t_i)$, where X_i is the input vector and t_i is the desired output, compute the overall D-FNN output y_i on the existing structure according to (4).

Define

$$\|e_i\| = \|t_i - y_i\|. \tag{7}$$

If

$$\|e_i\| > k_e \tag{8}$$

an RBF unit should be considered. Here, the value k_e is chosen *a priori* according to the desired accuracy of the D-FNN.

b) Accommodation Boundary: The accommodation criterion is described as follows: for the ith observation $(X_i,\ t_i)$, calculate the distance $d_i(j)$ between the observation X_i and the center C_j of the existing RBF units, i.e.,

$$d_i(j) = \|X_i - C_j\| \qquad j = 1,\ 2\ \cdots, u \tag{9}$$

where u is the number of existing RBF units.

Find

$$d_{\min} = \arg\ \min(d_i(j)). \tag{10}$$

If

$$d_{\min} > k_d \tag{11}$$

an RBF unit should be considered. Otherwise, the observation $(X_i,\ t_i)$ can be represented by the existing nearest RBF unit. Here, k_d is the effective radius of the accommodation boundary.

2) Hierarchical Learning: The concept of "hierarchical learning" was first presented in [9]. The main idea is that the accommodation boundary of each RBF unit is not fixed but adjusted dynamically in the following way: initially, the accommodation boundaries are set large for achieving rough but global learning. Then, they are gradually reduced for fine learning. Inspired by this idea, a simple method based on monotonically decreasing function to reduce both the effective radius of each RBF unit and error index gradually is presented. To be more specific, instead of constant $k_e,\ k_d$ in (8) and (11), we choose $k_e,\ k_d$ as

$$k_e = \max\left[e_{\max} \times \beta^i,\ e_{\min}\right] \tag{12}$$

$$k_d = \max\left[d_{\max} \times \gamma^i,\ d_{\min}\right] \tag{13}$$

where

$e_{\max}$ predefined maximum error;
$e_{\min}$ desired accuracy of an output of D-FNN;
β convergence constant;
$d_{\max}$ largest length of the input space;
$d_{\min}$ smallest length of interest;
γ decay constant.

The key idea of the hierarchical learning is to first find and cover the most troublesome positions, which have large errors between the desired and the actual outputs but are not properly covered by existing RBF units. This is called coarse learning. When k_e and k_d reaches $e_{\min}$ and $d_{\min}$, respectively, fine learning begins.

B. Allocation of RBF Unit Parameters

After a neuron has grown, the problem is how to determine its parameters. Simulation results show that the width of an RBF unit is significant for its generalization. If the width is less than the distance between adjacent inputs (i.e., underlapping), the RBF unit does not generalize well and the D-FNN will not give meaningful outputs in response to inputs for which they are not designed. However, if the width is too large, the output of the RBF node may always be large (near 1.0) irrespective of inputs. Hence, we allocate a new RBF unit with

$$C_i = X_i \tag{14}$$

$$\sigma_i = k \times d_{\min} \tag{15}$$

where k is an overlap factor that determines the overlap of responses of the RBF units.

When the first pattern (X_1, t_1) enters the D-FNN, it is adopted as the first neuron: $C_1 = X_1$, $\sigma_1 = \sigma_0$, where σ_0 is a prespecified constant.

It is worth highlighting that only one case i.e., $\|e_i\| > k_e$, $d_{\min} > k_d$ for neuron generation has been considered so far. For the other three cases, the algorithm is as follows:

Case 1: $\|e_i\| \le k_e$, $d_{\min} \le k_d$*:* This implies that D-FNN can accommodate the observation (X_i, t_i) completely. Nothing should be done or only w_2 should be updated.

Case 2: $\|e_i\| \le k_e$, $d_{\min} > k_d$*:* This indicates that the system has good generalization and only weights should be adjusted.

Case 3: $\|e_i\| > k_e$, $d_{\min} \le k_d$*:* This reveals that although X_i can be clustered to the adjacent generated RBF unit, the significance of the RBF unit is not so great. More precisely, the nearest RBF unit is not so good for generalization. Accordingly, the width of the nearest RBF node and all the weights should be updated simultaneously.

For the nearest kth RBF unit clustering (X_i, t_i):

$$\sigma_k^i = k_w \times \sigma_k^{i-1} \tag{16}$$

where k_w is a predefined constant.

C. Weight Adjustment

The idea of weight adjustment is as follows: assume nth training pattern enters the D-FNN and all these n data are memorized by the D-FNN in the input matrix $P \in \Re^{r\times n}$ and the output matrix $T \in \Re^n$, respectively, upon which the weights are determined.

Suppose u RBF units are generated according to the two criteria stated above for these n observations, the output of N nodes can be obtained according to (3). Denote

$$\psi = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \vdots & \vdots \\ a_{u1} & \cdots & a_{un} \end{pmatrix}. \tag{17}$$

Then, for any input $X_j(x_{1j}, x_{2j}, \cdots, x_{rj})$, the output y_j of the system is

$$y_j = \sum_{i=1}^{u} w_{2i} \times \psi_i. \tag{18}$$

Rewriting (18) in a more compact form

$$W_2 \Psi = Y \tag{19}$$

where for the S model, $W_2 \in \Re^u$, $\Psi \in \Re^{u\times n}$, and for the TSK model, W_2 and Ψ are given by

$$W_2 = (\, k_{10} \ \cdots \ k_{u0} \ \ k_{11} \ \cdots \ k_{u1} \ \cdots \ k_{1r} \ \cdots \ k_{ur} \,) \tag{20}$$

$$\Psi = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \vdots & \vdots \\ a_{u1} & \cdots & a_{un} \\ a_{11}.x_{11} & \cdots & a_{1n}.x_{1n} \\ \vdots & \vdots & \vdots \\ a_{u1}.x_{11} & \cdots & a_{un}.x_{1n} \\ \vdots & \vdots & \vdots \\ a_{11}.x_{r1} & \cdots & a_{1n}.x_{rn} \\ \vdots & \vdots & \vdots \\ a_{u1}.x_{r1} & \cdots & a_{un}.x_{rn} \end{pmatrix}. \tag{21}$$

Our objective is the following: given $\Psi \in \Re^{(r+1)u\times n}$ and $T \in \Re^n$ related by

$$Y = W_2 \Psi \tag{22}$$

$$\tilde{E} = \|T - Y\|. \tag{23}$$

Find an optimal coefficient vector $W_2^* \in \Re^{(r+1)u}$ such that the error energy $\tilde{E}^T\tilde{E}$ is minimized. This problem can be solved by the well-known linear least squares (LLS) method by approximating

$$W_2^* \Psi = T. \tag{24}$$

The optimal W_2^* is in the form of

$$W_2^* = T\Psi^+ \tag{25}$$

where Ψ^+ is the pseudoinverse of Ψ

$$\Psi^+ = \left(\Psi^T\Psi\right)^{-1}\Psi^T. \tag{26}$$

It has been shown that the LLS method provides a computationally simple but efficient procedure for determining the weights so that it can be computed very quickly and can be used for real-time control.

D. Pruning Technology

In the above algorithm, once a hidden unit is created, it can never be removed regardless of whether it is significant. Sometimes, a neuron may be active initially, but eventually contributes little to the system. Furthermore, pruning becomes imperative for identification of nonlinear systems with time-varying dynamics. If inactive hidden units can be detected and removed as learning progresses, a more parsimonious network topology can be achieved. Here, a new pruning technology called error reduction ratio (ERR) method is proposed.

1) Error Reduction Ratio Method: Rewrite (22) and (23) as

$$D = H\theta + E \tag{27}$$

where

$$D = T^T \in \Re^n$$

$$H = \Psi^T = (h_1 \cdots h_v) \in \Re^{n\times v}$$

$$\theta = W_2^T \in \Re^v$$

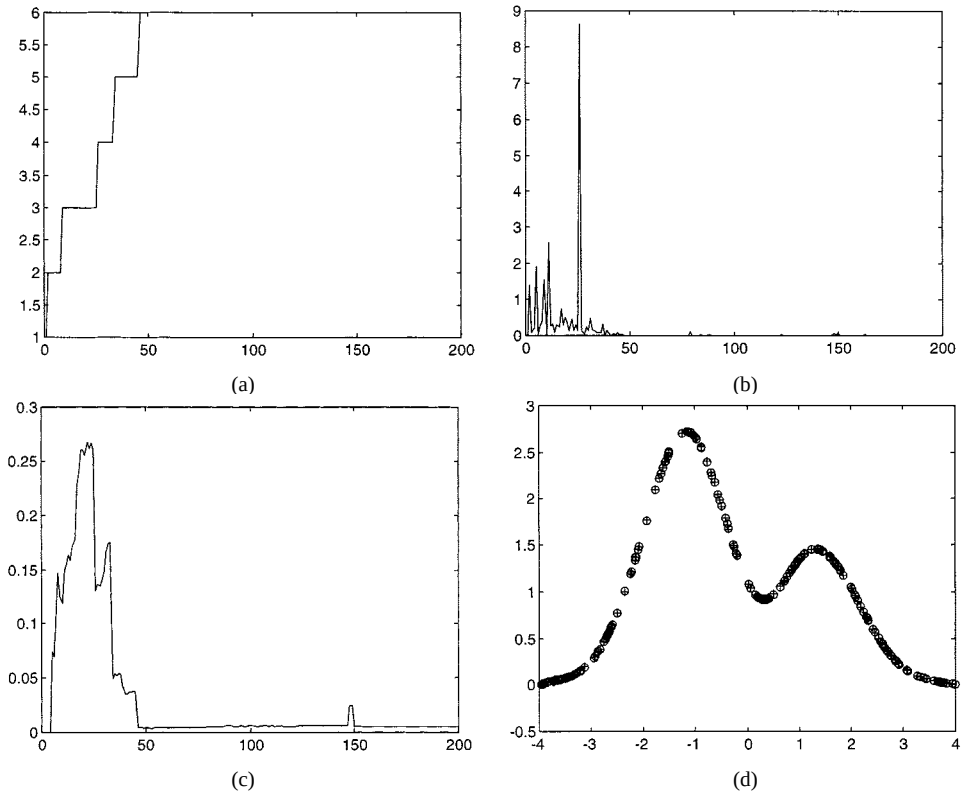

Fig. 2. Static function approximation: (a) growth of neurons, (b) actual output error during training, (c) root mean squared error during training, and (d) the hermite function $(+)$ and its approximation (o).

and

$$v = u \times (r + 1).$$

For any matrix H, if its row number is larger than the column number, we can transform H into a set of orthogonal basis vectors and thus make it possible to calculate an individual contributions to the desired output energy from each basis vector [10]. Thus, H is decomposed into

$$H = WA \tag{28}$$

where $W = (w_1 \cdots w_v)$ is an $n \times v$ matrix (of the same dimension as H) with orthogonal columns and A is a $v \times v$ upper triangular matrix. Note that the space spanned by the set of orthogonal basis vectors w_i is the same space spanned by the set of h_i.

Substituting (28) into (27) yields

$$D = WA\theta + E = WG + E. \tag{29}$$

The LLS solution of G is given by $G = (W^T W)^{-1} W^T D$, or

$$g_i = \frac{w_i^T D}{w_i^T w_i} \qquad i = 1, 2 \cdots, v. \tag{30}$$

The quantities G and θ satisfy the following equation

$$A\theta = G. \tag{31}$$

As w_i and w_j are orthogonal for $i \neq j$, the sum of squares or energy of D is as follows:

$$D^T D = \sum_{i=1}^{v} g_i^2 w_i^T w_i + E^T E. \tag{32}$$

If D is the desired output vector after its mean has been removed, the variance of D is given by [10]

$$n^{-1} D^T D = n^{-1} \sum_{i=1}^{v} g_i^2 w_i^T w_i + n^{-1} E^T E. \tag{33}$$

It is seen that $\sum_{i=1}^{v} g_i^2 w_i^T w_i / n$ is the part of the desired output variance which can be explained by the regressor w_i and $E^T E / n$ is the unexplained variance of D. Thus, $g_i^2 w_i^T w_i / n$ can be regarded as the increment to the explained desired output variance introduced by w_i and an ERR due to w_i can be defined as

$$\mathrm{err}_i = \frac{g_i^2 w_i^T w_i}{D^T D} \qquad 1 \leq i \leq v. \tag{34}$$

Substituting g_i by (30), we have

$$\mathrm{err}_i = \frac{\left(w_i^T D\right)^2}{w_i^T w_i D^T D} \qquad i \leq v. \tag{35}$$

The ERR in (35) offers a simple and effective means of seeking a subset of significant regressors. The significance of (35) is that err_i reflects the similarity of w_i and D or the inner product of w_i and D.

If err_i assumes the largest value, the similarity of w_i and D will be greatest and w_i is the most significant factor to the output.

2) Determination of RBF Units: Define the ERR matrix $\Delta = (\delta_1, \delta_2, \cdots, \delta_u) \in \Re^{(r+1)\times u}$ whose elements are obtained from (35) and the ith column of Δ is the total ERR corresponding to the ith RBF unit. Furthermore, define

$$\eta_i = \sqrt{\frac{\delta_i^T \delta_i}{r+1}} \tag{36}$$

then, η_i represents the significance of the ith RBF unit. The larger the value of η_i is, the more important the ith RBF unit is.

If

$$\eta_i < k_{\mathrm{err}}$$

where k_{err} is a prespecified threshold, then the ith RBF unit is deleted.

IV. Simulation Studies

In Section IV, the effectiveness of the proposed algorithm is demonstrated in static function approximation, nonlinear dynamic system identification and Mackey–Glass time-series prediction. Some comparisons are made with other learning algorithms such as RBF-AFS [5], OLS [10], RANEKF [11], and M-RAN [12].

Example 1: Static Function Approximation: Here, the underlying function to be approximated is the Hermite polynomial:

$$f(x) = 1.1(1 - x + 2x^2)\ \exp\left(-\frac{x^2}{2}\right). \tag{37}$$

The parameters, which are commonly used, are chosen to be the same as those in [11] and [12]:

$$\begin{aligned} d_{\max} &= 2 & e_{\max} &= 1.1 \\ d_{\min} &= 0.2 & e_{\min} &= 0.02 \\ \gamma &= 0.977 & k &= 1.1 \\ \beta &= 0.9 & k_w &= 1.1 \\ \sigma_0 &= 2 & k_{\mathrm{err}} &= 0.0015. \end{aligned}$$

Random sampling of the interval $[-4, 4]$ is used in obtaining the 200 input–output data for the training set. The results are shown in Fig. 2.

A comparison of structure and performance based on root mean squared error (RMSE) is shown in Table I.

In order to evaluate whether the neurons generated by the D-FNN are reasonable, the distribution of the neurons is shown in Table II and compared with that generated by the OLS.

It can be seen from Table II that the RBF units in the D-FNN are automatically distributed wherever the output has sharp changes with different widths so as to accommodate local fields to a great extent. On the contrary, very nearby RBF nodes, for example, $C_1 = -1.0228$ and $C_3 = -1.119$ in the OLS, will be generated because of fixed width for each neuron so that more neurons are needed.

Example 2: Nonlinear Dynamic System Identification: The plant to be identified in [5] is described as:

$$y(t+1) = \frac{y(t)y(t-1)[y(t)+2.5]}{1+y^2(t)+y^2(t-1)} + u(t). \tag{38}$$

TABLE I
Comparison of Structure and Performance of Different Algorithms

	D-FNN	OLS	M-RAN	RANEKF
Number of neurons	6	7	7	13
RMSE	0.0056	0.0095	0.009	0.0262

TABLE II
Distribution of RBF Units

Neuron	D-FNN		OLS	
	Center C	Width σ	Center C	Width σ
1	0.5164	3.8974	-1.0228	1.2011
2	-1.5806	4.0865	1.5925	1.2011
3	2.7701	4.8311	-1.1190	1.2011
4	-3.8253	3.9765	2.0755	1.2011
5	1.6810	1.1980	-0.6863	1.2011
6	-0.4198	1.0298	2.3174	1.2011
7			-1.6848	1.2011

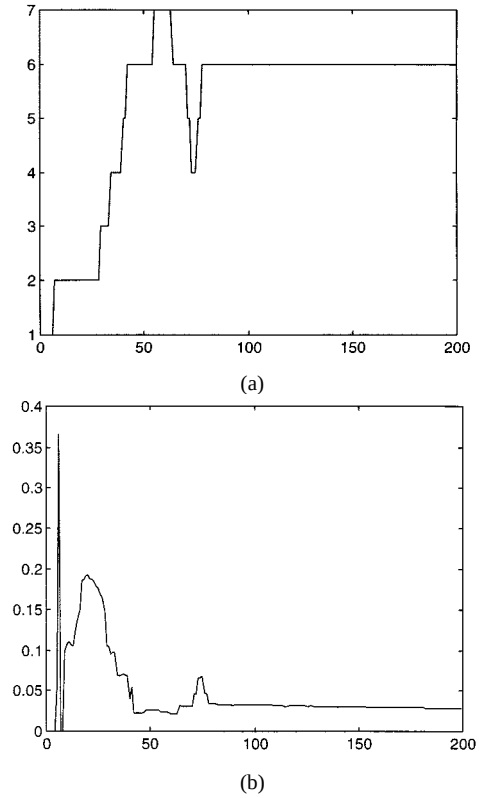

Fig. 3. Nonlinear dynamic system identification: (a) growth of neurons and (b) root mean squared error during training.

TABLE III
Comparison of Structure and Performance of Different Algorithms

	D-FNN	OLS	RBF- AFS
Number of neurons	6	65	35
RMSE	0.0283	0.0288	0.1384

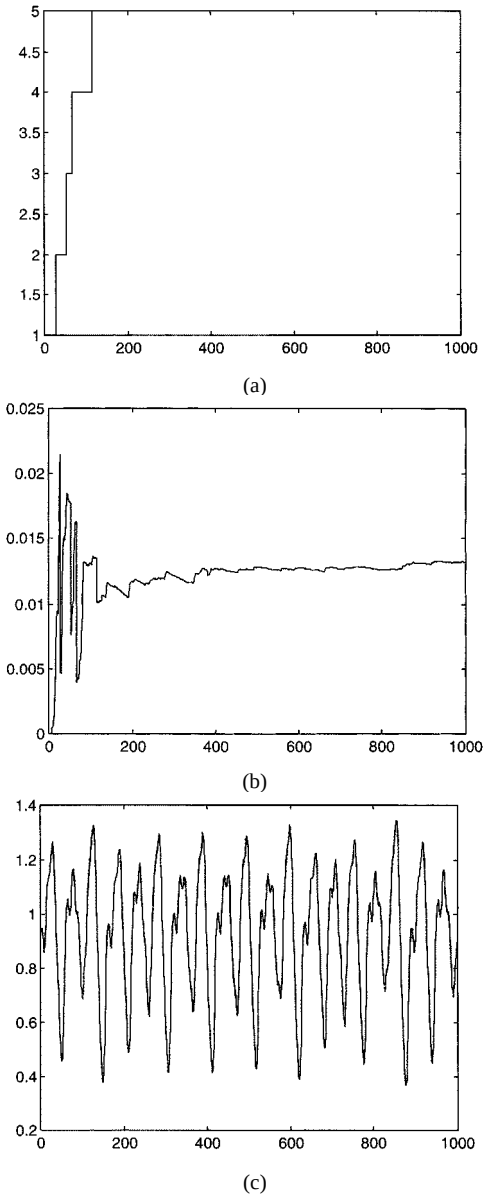

Fig. 4. Mackey-Glass time-series prediction: (a) growth of neurons, (b) root mean squared error during training, and (c) training result [desired (—) and actual (—) data].

The equilibrium states of the unforced system are (0, 0) and (2, 2) on the state space. If a series-parallel identification model is used for identifying the plant, the model can be described by the equation

$$\hat{y}(t+1) = f(y(t),\ y(t-1),\ u(t)) \tag{39}$$

where f is the D-FNN with three inputs and one output. Using the same input $u(t) = \sin(2\pi t/25)$, the results are shown in Fig. 3 and comparisons with [5] and [10] are shown in Table III.

Example 3: Mackey–Glass Time-Series Prediction: The Mackey–Glass time-series prediction in [5] is generated by

$$x(t+1) = (1-a)x(t) + \frac{bx(t-\tau)}{1+x^{10}(t-\tau)}. \tag{40}$$

Choosing the same parameters as in [5] i.e., $a = 0.1$, $b = 0.2$, $\tau = 17$ and the initial condition as $x(0) = 1.2$, the prediction model

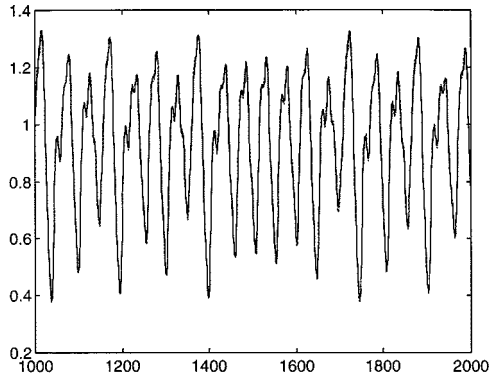

Fig. 5. Testing result [original (—) and predicted (—) data].

TABLE IV
COMPARISON OF STRUCTURE AND PERFORMANCE OF DIFFERENT ALGORITHMS

Technique	Number of neurons	RMSE of training	RMSE of testing
D-FNN	5	0.0132	0.0131
OLS	13	0.0158	0.0163
RBF-AFS	21	0.0107	0.0128

is given by

$$x(t+6) = f[x(t),\ x(t-6),\ x(t-12),\ x(t-18)]. \tag{41}$$

For the purpose of training, we extract 1000 data points between $t = 124$ and $t = 1123$ and use them for preparing the input and output sample data in the structure of (41). The results are shown in Fig. 4.

In order to demonstrate the capability of prediction by the learned D-FNN, another 1000 data points between $t = 1124$ and $t = 2123$ are tested. The result is shown in Fig. 5.

Comparisons of D-FNN, OLS,and RBF-AFS are shown in Table IV.

V. CONCLUSIONS

In this paper, an architecture of D-FNN performing TSK fuzzy systems based on extended RBF neural networks is proposed. Based on the D-FNN, a hierarchical on-line self-organizing learning algorithm, whereby both the structure and parameter identification can be achieved quickly and simultaneously without iterative learning and initialization of structure and parameters, is presented. Simulation studies and comprehensive comparisons show that the integration of TSK fuzzy model with neural networks could offer tremendous advantages in approximating both static functions and nonlinear dynamic systems. A parsimonious structure with high performance can be achieved by the proposed approach.

ACKNOWLEDGMENT

The authors would like to express sincere thanks to the anonymous reviewers for their valuable comments.

REFERENCES

[1] K. Hornic, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, 1989.
[2] L. Wang, "Fuzzy systems are universal approximators," in *Proc. Int. Conf. Fuzzy Syst.*, 1992.
[3] J.-S. R. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Trans. Syst. Man. Cybern.*, vol. 23, pp. 665–684, 1993.
[4] C. T. Lin, "A neural fuzzy control system with structure and parameter learning," *Fuzzy Sets and Systems*, vol. 70, pp. 183–212, 1995.

[5] K. B. Cho and B. H. Wang, "Radial basis function based adaptive fuzzy systems and their applications to system identification and prediction," *Fuzzy Sets and Systems*, vol. 83, pp. 325–339, 1996.
[6] C. F. Juang and C. T. Lin, "An on-line self-constructing neural fuzzy inference network and its applications," *IEEE Trans. Fuzzy Syst.*, vol. 6, pp. 12–32, 1998.
[7] R. P. Li and M. Mukaidono, "A new approach to rule learning based on fusion of fuzzy logic and neural networks," *IEICE Trans. Inf. Syst.*, vol. E78-d, pp. 1509–1514, 1995.
[8] Y. H. Lin and G. A. Cunningham, "A new approach to fuzzy-neural system modeling," *IEEE Trans. Fuzzy Syst.*, vol. 3, pp. 190–197, 1995.
[9] S. Lee and R. M. Kil, "A Gaussian potential function network with hierarchically self-organizing learning," *Neural Networks*, vol. 4, pp. 207–224, 1991.
[10] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function network," *IEEE Trans. Neural Networks*, vol. 2, pp. 302–309, 1991.
[11] V. Kadirkamanathan and M. Niranjan, "A function estimation approach to sequential learning with neural networks," *Neural Comput.*, vol. 5, pp. 954–975, 1993.
[12] Y. Lu, N. Sundararajan, and P. Saratchandran, "A sequential learning scheme for function approximation by using minimal radial basis function networks," *Neural Comput.*, vol. 8, pp. 461–478, 1997.

Tracking Control of a Rolling Disk

C. Frangos and Y. Yavin

***Abstract*—The tracking control of a disk rolling without slipping on the horizontal (X, Y)-plane is considered. The motion of the disk can be controlled via a tilting torque and a pedaling torque. The concept of path controllability of the disk is introduced and then used to calculate control laws such that the disk tracks a given path in the (X, Y)-plane.**

***Index Terms*—Input–output linearization, nonholonomic constraints, path controllability, rolling disk, tracking control.**

I. Introduction

In this work, we deal with the tracking control of a disk rolling on a horizontal plane. The dynamic model describing the disk's motion can be seen as a simplified model of a riderless unicycle. The motion of the disk can be controlled by a tilting torque and a pedaling torque. The problem addressed here is to compute feedback control laws for the pedaling torque and tilting torque such that the disk tracks a prescribed path in the (X, Y)-plane. The disk can start from various initial positions that are not necessarily on the path. If the initial position is not on the path then the feedback control laws should steer the disk onto the path and keep it moving on the path. The desired path is generated by using the motion of a fictitious point mass in the (X, Y)-plane. Note that the motion of the disk is subject to nonholonomic constraints [1], [7].

Manuscript received November 19, 1994; revised August 27, 1996, October 29, 1997, and January 5, 2000. This paper was recommended by Associate Editor W. A. Gruver.

C. Frangos is with the Department of Statistics, The Rand Afrikaans University, Johannesburg, South Africa.

Y. Yavin is with the Laboratory for Decision and Control, Department of Electrical and Electronic Engineering, University of Pretoria, Pretoria 0002, South Africa.

Publisher Item Identifier S 1083-4419(00)02975-7.

The tracking control problem posed here is of basic importance for solving more complex vehicle path planning and control problems [4]. One example is the control of a mobile robot with a rotating or retractable arm [16], [17] that has to move along a very narrow passage and perform some task. The present work is part of a conceptual study in autonomous robots and vehicles and the hardware implementation issues will be considered at a later stage. However, a rolling disk with an overhead rotor has been proposed for implementing the tilting torque [13].

The application of optimal control theory to nonlinear systems [3] leads in general only to open loop control laws. Since we are interested in obtaining feedback control laws for the disk, optimal control is not appropriate in this instance. The required feedback control laws are obtained by using the concept of path controllability. Let (x, y, z) denote the coordinates of the center of the disk in the inertial (X, Y, Z) coordinate system. Denote $\boldsymbol{v}(t) = (x(t), y(t), dx(t)/dt, dy(t)/dt) \in R^4$. If for any two points $\boldsymbol{Q}_1$ and $\boldsymbol{Q}_2$ in R^4 and for any finite time interval $[0, t_f]$, a tilting torque and pedaling torque can be found such that $\boldsymbol{v}(\cdot)$ will move from $\boldsymbol{v}(0) = \boldsymbol{Q}_1$ to $\boldsymbol{v}(t_f) = \boldsymbol{Q}_2$ then we say that the disk's motion is *path controllable*.

The concept of path controllability introduced here is a kind of input–output linearization [8], [9] and has been successfully applied to nonlinear systems in R^8 [11], [12], R^9 [5], [10], R^{10} [15], and R^{11} [13], [14], [16], [17].

II. Dynamical Model of a Rolling Disk

In this work, we consider the control of a disk rolling on the horizontal (X, Y)-plane (see Fig. 1). Let $\boldsymbol{I}$, $\boldsymbol{J}$, and $\boldsymbol{K}$ be unit vectors along the inertial (X, Y, Z) coordinate system. The Euler angles θ, ϕ, ψ are defined as follows: θ is the leaning angle, that is, the angle between the disk axis and the Z-axis, where for $\theta = \pi/2$ the plane of the disk is vertical to the (X, Y)-plane; the angle ϕ represents the direction of the disk; and ψ represents the angle of rotation of the disk around its axis. Denote by $\boldsymbol{k}$

$$\boldsymbol{k} = \sin\theta\cos\phi\boldsymbol{I} + \sin\theta\sin\phi\boldsymbol{J} + \cos\theta\boldsymbol{K} \tag{1}$$

a unit vector along the axis of the disk. Then the vectors

$$\boldsymbol{i}_\theta = \frac{\partial\boldsymbol{k}}{\partial\theta} = \cos\theta\cos\phi\boldsymbol{I} + \cos\theta\sin\phi\boldsymbol{J} - \sin\theta\boldsymbol{K} \tag{2}$$

and

$$\boldsymbol{i}_\phi = \left(\frac{1}{\sin\theta}\right)\frac{\partial\boldsymbol{k}}{\partial\phi} = -\sin\phi\boldsymbol{I} + \cos\phi\boldsymbol{J} \tag{3}$$

are at all times in the plane of the disk. Also, we introduce the vectors $\boldsymbol{i}$ and $\boldsymbol{j}$

$$\boldsymbol{i} = \cos\psi\boldsymbol{i}_\theta + \sin\psi\boldsymbol{i}_\phi, \quad \boldsymbol{j} = -\sin\psi\boldsymbol{i}_\theta + \cos\psi\boldsymbol{i}_\phi \tag{4}$$

which are always in the plane of the disk. Then, the angular velocity vector $\boldsymbol{\omega} = \omega_1\boldsymbol{i} + \omega_2\boldsymbol{j} + \omega_3\boldsymbol{k} = \omega_x\boldsymbol{I} + \omega_y\boldsymbol{J} + \omega_z\boldsymbol{K}$ is given by (see [1])

$$\omega_1 = \dot\theta\sin\psi - \dot\phi\sin\theta\cos\psi,\ \omega_2 = \dot\theta\cos\psi + \dot\phi\sin\theta\sin\psi$$
$$\omega_3 = \dot\psi + \dot\phi\cos\theta \tag{5}$$