A Hierarchical Fuzzy System with Automatical Rule Extraction

Shuqing Zeng
Department of Computer Science and Engineering
Michigan State University
East Lansing, MI 48824, USA
zengshuq@cse.msu.edu

Yongbao He and Jie Jiang
Department of Computer Science
Fudan University
220 Handan Road
Shanghai 200433, China

Abstract— **In this paper, we propose a hierarchical fuzzy system for high-dimensional data. We introduce a locally weighted scheme to the extraction of Takagi-Sugeno type rules. We apply the sequential least-squares method to estimate the linear model. A hierarchical clustering takes place in the product space of systems inputs and outputs, and each path from the root to a leaf corresponds to a fuzzy IF-THEN rule. Only a subset of the rules is considered, based on the locality of the input query data. At each hierarchy, a discriminating subspace is derived from the high-dimensional input space for a good generalization capability. Both a synthetic data set and a real-world robot collision avoidance problem are considered to illustrate how the algorithm works and the applicability of it.**

I. Introduction

The fuzzy rule-based model represents a simple and powerful tool to model system dynamics by means of IF-THEN rules. Human experts' knowledge is usually used to design those rules. This acquisition process is cumbersome, and in some cases (e.g., unknown system), expert knowledge is not available. Neuro-fuzzy approaches [4] are introduced with the purpose of extracting the fuzzy rules, and the membership-functions automatically from measured input-output pairs. However, in neural-fuzzy approaches, the dimension of the input vector is usually small to be manageable.

In this paper, we propose a hierarchical fuzzy model, which extracts rules from high-dimensional data pairs. The major distinctions of the model include: First, we derive automatically discriminant feature subspaces in a coarse-to-fine manner from the high-dimensional input space. The features are most discriminative in the sense that input variables irrelevant to the output are disregarded to achieve better generalization. Second, we organize the rule base in a hierarchical way. This tree architecture recursively excludes many far-away, unrelated rules from consideration; thus, the time for inference and learning is $O(\log(r))$, where r is the size of the rule base.

The remainder of this paper is organized as follows: Section II presents the fuzzy model. We then extend it to the hierarchical form in Section III. The results of the simulation and real robot experiments are reported in Section IV. Discussions and concluding remarks are given in Section V.

II. Fuzzy Modeling

A. Fuzzy Rule-Based System

A fuzzy rule is a syntactic structure of the form

$$\text{IF } \textit{antecedent} \text{ THEN } \textit{consequent} \tag{1}$$

where *antecedent* and *consequent* are well-formed fuzzy predicates. Given a system with the input vector $\mathbf{x} \in \mathcal{X} \subset R^n$ and the output vector $\mathbf{y} \in \mathcal{Y} \subset R$. In the Takagi-Sugeno (TS) model [7], the base of r fuzzy rules is represented by

$$\begin{array}{l} \text{IF } \mathbf{x} \text{ is } A_1 \text{ THEN } y_1 = \beta_1^T\mathbf{z} + \beta_{0,1} \\ \vdots \\ \text{IF } \mathbf{x} \text{ is } A_r \text{ THEN } y_r = \beta_r^T\mathbf{z} + \beta_{0,r} \end{array} \tag{2}$$

where A_1,...,A_N denote multivariate *antecedents* (fuzzy predicates) defined on the universe R^d. Here the *consequents* are linear function of the dependent vector $\mathbf{z}$ that is the projected vector on a discriminating subspace (see Section III-B). The membership function of the i-th *antecedent* A_i is defined as

$$A_i(\mathbf{x}) : R^d \mapsto [0,1] \tag{3}$$

For a query input $\mathbf{x}$ the output of the rule-base is calculated by aggregating the individual rules contributions

$$y = \frac{\sum_{i=1}^{r} A_i(\mathbf{x}) y_i}{\sum_{i=1}^{r} A_i(\mathbf{x})} \tag{4}$$

where y_i and $A_i(\mathbf{x})$ denote the output and the activation level of the i-th rule, respectively.

B. Incremental Parameter Estimation

Two phases are involved to estimate the parameters of the i-th rule. First, in the structure identification phase the fuzzy predicate A_i is determined, which will be discussed in Section III-C. Second, In the parameter identification phase, we assume the *antecedent* predicate A_i is fixed and apply sequential least-square algorithm to estimate the parameters: β_i and $\beta_{0,i}$ of the rule.

For notation simplicity we neglect the rule index i. Consider a collection of t input-output data pairs $(\mathbf{x}_k, y_k)$, $k = 1, 2, ..., t$ where $\mathbf{x}_k$ is the d dimensional input vector and y_k denotes the scalar target output of the system. Let $\mathbf{z}_k$, $k = 1, ..., t$ be extracted feature vector which we will discuss in Section III-B.

As in locally weighted learning (LWR) [1], we compute the following diagonal activation matrix:

$$W = \text{diag}\begin{bmatrix} A(\mathbf{x}_1) & A(\mathbf{x}_2) & ... & A(\mathbf{x}_t) \end{bmatrix}$$

with each one corresponding to the data pair. Let $\mathbf{y} = \begin{bmatrix} y_1 & ... & y_t \end{bmatrix}^T$ and $Z = \begin{bmatrix} \mathbf{z}_1^T & ... & \mathbf{z}_t^T \end{bmatrix}^T$. For computational and analytical simplicity, let $Z_e = \begin{bmatrix} Z & \mathbf{1} \end{bmatrix}$ and $\theta = \begin{bmatrix} \beta^T & \beta_0 \end{bmatrix}^T$.

We formulate the estimation as finding the parameter θ of the linear model:

$$y = \theta^T \mathbf{z} + n \tag{5}$$

such that the following cost function is minimized

$$(\mathbf{y} - Z_e\theta)^T W (\mathbf{y} - Z_e\theta).$$

where n denotes a white Guassian process.

The solution of this weighted least square problem [6] is

$$\theta = (Z_e^T W Z_e)^{-1} Z_e^T W \mathbf{y} \tag{6}$$

Let the weight of the k-th data pair $(\mathbf{x}_k, y_k)$, $(k = 1, ..., t)$ be the square root of the i-rule's *antecedent* $A(\mathbf{x}_k)$, i.e.,

$$a_k = \sqrt{A(\mathbf{x}_k)}$$

Each row k of X_e and $\mathbf{y}$ is multiplied by the corresponding weight a_k creating new variables $\mathbf{z}_k^* = a_k \mathbf{z}_k$ and $y_k^* = a_k y$. Also, this can be done using matrix notation $Z^* = Z_e W^{\frac{1}{2}}$ and $\mathbf{y}^* = \mathbf{y} W^{\frac{1}{2}}$, where $Z^* = \begin{bmatrix} \mathbf{z}_1^{*T} & \dots & \mathbf{z}_t^{*T} \end{bmatrix}^T$ and $\mathbf{y}^* = \begin{bmatrix} y_1^* & \cdots & y_t^* \end{bmatrix}^T$. Therefore, Eqs. (5) and (6) become respectively

$$y = Z^* \theta + n \tag{7}$$

and

$$\theta = (Z^{*T} Z^*)^{-1} Z^{*T} \mathbf{y}^* \tag{8}$$

in the matrix form.

We note that Eq. (8) assumes the collection of input-output pairs are available before the estimation. This is not suitable for online identification where data pairs arriving sequentially. The inverse operation is too expensive to recompute whenever a new data comes in. However, it is possible to update θ incrementally as new data pairs are acquired.

Let us write the linear *consequent* part of the rule, Eq. (7) as follows

$$\begin{bmatrix} y_1^* \\ \vdots \\ y_{t-1}^* \\ \hline y_t^* \end{bmatrix} = \begin{bmatrix} Z_{t-1}^* \\ \\ \hline \mathbf{z}_t^{*T} \end{bmatrix} \theta + \begin{bmatrix} n_1 \\ \vdots \\ n_{t-1} \\ \hline n_t \end{bmatrix} \tag{9}$$

Let $\hat{\theta}^{(t)}$ denote the estimated θ of Eq. (9) after presenting the data set X_t^*. Let $P_t^{-1} = Z_t^{*T} Z_t^*$. The recursive equations for sequentially estimating θ can be written as [6]

$$\begin{aligned} \hat{\theta}^{(t)} &= \hat{\theta}^{(t-1)} + \gamma_t P_{t-1} \mathbf{z}_t^* (y_t^* - \mathbf{z}_t^* \hat{\theta}^{(t-1)}) \\ P_t &= P_{t-1} - \gamma_t P_{t-1} \mathbf{z}_t^* \mathbf{z}_t^{*T} P_{t-1} \\ \gamma_t^{-1} &= 1 + \mathbf{z}_t^{*T} P_{t-1} \mathbf{z}_t^* \end{aligned} \tag{10}$$

where

$$\begin{aligned} y_t^* &= a_t y_t \\ \mathbf{z}_t^* &= a_t \mathbf{z}_t \end{aligned} \tag{11}$$

The reason that we do not do regression on $\mathbf{x}$ directly is that matrix P_t in Eq. (10) is a $d \times d$ matrix, where d denotes the dimension of the input vector. Computing such a matrix for input $\mathbf{x}$ (e.g., a system with 1000 input variables) is too expensive.

III. Hierarchical Fuzzy Model

In this section, we propose a hierarchical fuzzy model. The fuzzy model presented in the previous section is a flat model. All rules in the base take part simultaneously in the inference process, each to an extent proportionate to the firing value associated with its *antecedent* (e.g., the fuzzy predicate A_i of the i-th rule). This poses a problem for learning since the parameters of the all rules need to be updated when a new pair is presented, which is computationally too expensive. To solve the problem, here we present a rule selection scheme based on locality.

A. Clustering

Locality selection involves successively partitions in the product space of input-output pairs so that at each level of the hierarchy, data pairs within the same region (cluster) are more similar to each other than those in difference regions (clusters). As shown in Fig. 1, each node denotes a cluster, or a rule's *antecedent* in fuzzy terminology. For example, region 1 denotes the support set of A_1 and region 1.1 denotes the support set of A_{11}. A path from the root to a leaf represents a rule, i.e.,

$$\begin{aligned} &\text{IF } A_{k_1}(\mathbf{x}) \wedge A_{k_1 k_2}(\mathbf{x}) \wedge \ldots \wedge A_{k_1 k_2 \ldots k_j}(\mathbf{x}) \\ &\quad \text{THEN } y_i = \beta_i^T \mathbf{x} + \beta_{0,i} \end{aligned}$$

where k_1, ..., k_j are index of child cluster w.r.t. its parent, respectively.

We observe that

$$A_{k_1}(\mathbf{x}) \supset A_{k_1 k_2}(\mathbf{x}) \supset \ldots \supset A_{k_1 k_2 \ldots k_j}(\mathbf{x})$$

Therefore the *antecedent* $A_{k_1}(\mathbf{x}) \wedge A_{k_1 k_2}(\mathbf{x}) \wedge \ldots \wedge A_{k_1 k_2 \ldots k_j}(\mathbf{x}) = A_{k_1 k_2 \ldots k_j}(\mathbf{x})$. In addition, a cluster $A_{k_1 \ldots k_j}(\mathbf{x})$ not fired unless its parent $A_{k_1 \ldots k_{j-1}}(\mathbf{x})$ is activated. Therefore, the tree structure recursively excludes many inapplicable rules from consideration and, thus, the time to retrieve and update the tree for each newly arrived data point $\mathbf{x}$ is $O(\log(r))$, where r is the number of rules. This extremely low time complexity is essential for online learning with a very large rule base.

Let's begin with the available K input-output pairs. The data set to be clustered is represented as a data matrix composed from H and $\mathbf{y}$

$$U^T = \begin{bmatrix} X & \mathbf{y} \end{bmatrix} \tag{12}$$

with each column $\mathbf{u}_k$ represents an input-output pair: $\mathbf{u}_k = \begin{bmatrix} \mathbf{x}_k^T & y_k \end{bmatrix}^T$, $k = 1, ..., K$.

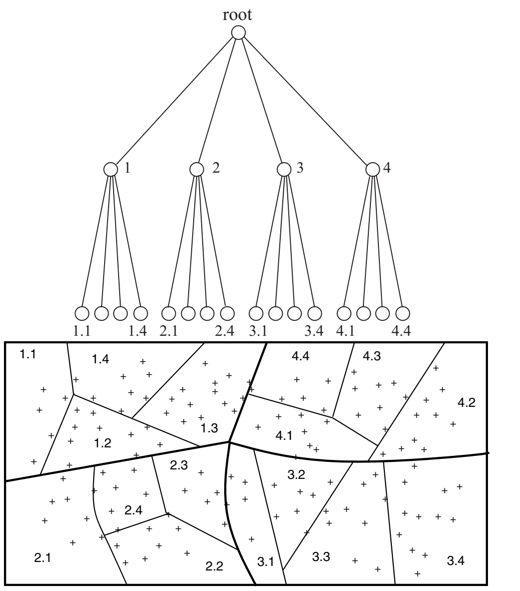

Fig. 1. The hierarchical organization of the rule base. Each path from the root from a leaf corresponds to a rule. The higher level node covers a larger region, and may partition into several smaller regions. Regions in the input space marked $i.j$ where i is the index of the parent region while j is the index of child region.

Definition 1: The dissimilarity measure d *between the two pairs* i *and* j*,* $1 \leq i, k \leq K$ *is defined as*

$$d(\boldsymbol{u}_i, \boldsymbol{u}_k) = w_x \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_k\|^2}{\sigma_x^2} + w_y \frac{\|y_i - y_k\|^2}{\sigma_y^2} \qquad (13)$$

where w_x *and* w_y *are two positive weights that sum to 1:* $w_x + w_y = 1$*;* σ_x *and* σ_y *denote the estimated scalar scatter of* $\boldsymbol{x}$ *and* $\mathbf{y}$ *vectors*[1]*, respectively.*

The hierarchical clustering algorithm begins with the entire data set as a single cluster G, and recursively splits one of the existing clusters into q child clusters: G_1,...,G_q in a top-down fashion. For each iteration, we apply an augmented K-means (see Algorithm 1) to perform the splits. we begin with choosing the q most separated samples in G as the initial means: $\{\bar{\mathbf{u}}_1, ..., \bar{\mathbf{u}}_q\}$ and $\bar{\mathbf{u}}_l = (\bar{\mathbf{x}}_l, \bar{y}_l)$, $l = 1, ..., q$.

The recursive splitting continues until all clusters either becoming a singleton or the dissimilarity within the cluster of each one from another is less than a threshold.

B. Local Feature Extraction

We note that in Eq. (2), the linear model applies on the input vector $\mathbf{x}$ directly, which might contain irrelevant dimensions of the input data ("curse of dimensionality"). Given the empirical observation that the true intrinsic dimensionality of high dimensional data is often very low, we can view informative low-dimensional projections of the data.

[1]For simplicity, we assume the covariance matrix (Σ_x) of a variate $\mathbf{x}$ is equal to $\sigma_x^2 I$, where I denotes an identical matrix. Thus, its corresponding scatter is $\sigma_x = \sqrt{\text{tr}\Sigma_x} = \sqrt{d}\sigma$, where d denotes the dimension of the variate $\mathbf{x}$ (see Marida [5] page 13).

Algorithm 1 Cluster splitting algorithms

```
1: Randomly select $\bar{\mathbf{u}}_1$ from $G$.
2: for $k = 2, ..., q$ do
3:     Select $\bar{\mathbf{u}}_k = \mathbf{u}_j \in G$ s.t. $j = \arg\max_{1 \leq l \leq K} d_k(\mathbf{u}_l)$,
       where $d_k(\mathbf{u}) = \min_{1 \leq m \leq k-1}(d(\mathbf{u}, \bar{\mathbf{u}}_m))$
4: end for
5: For current set of means $\{\bar{\mathbf{u}}_1, ..., \bar{\mathbf{u}}_q\}$, assign each samples
   of $G$ to the closest mean. That is,
       $C(l) = \text{argmin}_{1 \leq k \leq q} \|\mathbf{u}_l - \bar{\mathbf{u}}_k\|, \ l = 1, ..., K$
6: Given the updated cluster assignment $C$, recompute the
   cluster means $\{\bar{\mathbf{u}}_1, ..., \bar{\mathbf{u}}_q\}$.
7: Steps 5 and 6 are iterated until either maximum iteration
   has reached or the assignments do not change.
8: Reassignment each samples of $G$ to the closest fuzzy
   antecedent, i.e.,
       $C(l) = \text{argmin}_{1 \leq k \leq q} \|A(\mathbf{x}_l)_k\|, \ l = 1, ..., K$
   where $A(.)$ will be defined in Eq. (15) of Section III-C.
```

Consider an internal node G of the tree with q child clusters in d-dimensional input space whose means are $\{\bar{\mathbf{x}}_1, ..., \bar{\mathbf{x}}_q\}$. Those clusters lie in an linear manifold of dimension $\leq q-1$, and if d is much larger than q, this will be a considerable drop in dimension.

We consider that two special cases of the dissimilarity measure defined in Eq. (13). First, let $w_x = 0$ and $w_y = 1$ and this is equivalent to supervised learning. Thus each cluster corresponds to a class (i.e., a different output value from that of other clusters). In locating the closest cluster, we can ignore distances orthogonal to this subspace, since they will contribute equally to each class. We might just as well project the X onto this cluster-spanning subspace H_L. There is a fundamental dimension reduction, namely that we need only consider the data in a subspace of dimension at most $q-1$. Second, $w_x = 1$ and $w_y = 0$ corresponding to unsupervised learning, where the clusters do not contain label related information. The clusters spread out as much as possible in term of variance and this amounts to finding principle component (PCA) subspaces of the data. Other configurations of w_x and w_y find a subspace softly combining the LDA and PCA ones.

In summary, finding the optimal subspaces for the most discriminating features (MDF) involves the following steps:

- compute the $q \times d$ matrix of clusters M ($M = \begin{bmatrix}\bar{\mathbf{x}}_1 & \dots & \bar{\mathbf{x}}_q\end{bmatrix}^T$) and the common covariance matrix W (for within-class covariance)
- compute $M^* = MW^{-\frac{1}{2}}$ using the singular value decomposition (SVD) of W (sphering the within-class covariance)
- apply the *Gram-Schmidt* procedure on the columns of M^* yielding $\mathbf{v}_l^*$, $l = 1, ..., L$ and $L \leq q-1$.
- compute the basis of the MDF subspace $\mathbf{v}_l = W^{-\frac{1}{2}}\mathbf{v}_l^*$, $l = 1, ..., L$.

Practically, we assume that the within-class covariance $W =$

$\sigma^2 I$ where I denotes an identity matrix[2]. Thus, applying the *Gram-Schmidt* procedure to M yields the desired basis vectors $\mathbf{v}_l$, $l = 1, ..., q$.

Let $V = \begin{bmatrix}\mathbf{v}_1 & \ldots & \mathbf{v}_L\end{bmatrix}$. The projected vector (feature) $\mathbf{z}$ for the node G is

$$\mathbf{z} = V\mathbf{x} \tag{14}$$

C. Estimating the Antecedent

Each node (cluster) in the tree represents a rule antecedent. Without losing generality, we consider an internal node A_{k_1} and the goal is to estimate the membership function of $A_{k_1 k_2}$, $k_2 = 1, ..., q$. The other lower nodes can be handled similarly.

For notation convenience, we drop the subscript k_1 without causing ambiguity. We assume the *antecedent* A whose fuzzy membership function takes the form, for $k_2 = 1, ..., q$

$$A_{k_2}(\mathbf{z}) = \exp[-(\mathbf{z} - \bar{\mathbf{z}}_{k_2})^T D_{k_2}^{-1}(\mathbf{z} - \bar{\mathbf{z}}_{k_2})] \tag{15}$$

where $\mathbf{z}$ is the projected vector on the MDF subspace; D_{k_2} is a positive determinate matrix and $\bar{\mathbf{z}}_{k_2}$ is the center of the cluster, both needing to be estimated from the data. It is worthy noting that A_{k_2} is defined on the MDF subspace H_L rather than the original input space.

Let $Z_{k_2} = \begin{bmatrix}\mathbf{z}_1^T & \ldots & \mathbf{z}_{n_{k_2}}^T\end{bmatrix}^T$ be projected data matrix that belongs to the cluster A_{k_2}. We then write the equations to compute $\mathbf{z}$ and D as follow

$$\bar{\mathbf{z}}_{k_2} = Z_{k_2}\mathbf{1}/n_{k_2} \tag{16}$$

and

$$\hat{D}_{k_2} = \alpha\hat{\Sigma}_{k_2} + \beta\hat{\Sigma} + \gamma\rho^2 I \tag{17}$$

where $\hat{\Sigma}_{k_2}$ denotes each cluster's covariance matrix; $\hat{\Sigma}$ is the common or within-class covariance matrix; ρ^2 is the shrunken scalar variance. Here weights α, β and γ sums to 1 and allows a continuum of transition among qudractic discriminant (QDA), linear discriminant (LDA) and scalar covariance models. This method is very similar in flavor to [2], [3] whose key point is regularization. By adjusting those weight controlling the model complexity becomes possible. QDA is the most complex model and scalar variance model is the less complex one. In practice, we set those parameters as a function of the number of samples belonging to the cluster.

D. Summary of the Algorithms

Algorithm 2 selects the subset of rules being fired by the input query $\mathbf{x}$. Only rules with the q largest *antecedent* fuzzy measurement in the base are selected. Instead of exhaustive linear search the whole base, we use a greedy heuristic tree search algorithm.

[2]This is reasonable since modeling W needs $d^2/2$ parameters which is huge for a large d. Thus estimating W tends to overfit the data. Moreover, computing SVD of M is computationally expensive.

Algorithm 2 Select rules: *Given a rule base R and an input vector $\mathbf{x}$, return the q rules whose antecedent clusters are the closest to the query input.*

```
1: $c, p \leftarrow$ the root node of $R$.
2: for  $c$ has been split into sub-clusters do
3:     $p \leftarrow c$
4:     $c \leftarrow$ the $m$th child of the node $c$, where $m = \arg\min_{1 \le k_i \le q}(A_{k_i}(\mathbf{x}))$ and $A_{k_i}(\mathbf{x})$, $k_i = 1, ..., q$ are defined in Eq. (15).
5: end for
6: Return $p$ and its child nodes which correspond to $q$ rules.
```

1) Inference: The goal of inference is that given a base B and an input vector $\mathbf{x}$, return the corresponding estimated output $\hat{y}$. This involves the followings steps:

- use Algorithm 2 finding the parent node p and its associated clusters $\{A_i \mid i = 1, ..., q\}$
- compute the projected feature vector $\mathbf{z} = V\mathbf{x}$, where V denotes the projection matrix of the MDF space derived by the node p
- compute the $y_i = \theta_i^T \mathbf{z}$ and $\bar{y} = \frac{\sum_{i=1}^q A_i(\mathbf{x}) y_i}{\sum_{i=1}^q A_i(\mathbf{x})}$ (see Eqs. (5) and (4) respectively)

2) Learning: The learning of the proposed fuzzy model contains two phases: structure identification and parameter identification.

In structure identification, a set of initial hierarchical fuzzy rules are discovered from given a collected labeled sample $G = \{(\mathbf{x}_l, y_l) \mid l = 1, ..., N\}$. The result is the hierarchical organized rule base B. This involves

- recursively split G by using Algorithm 1 until all clusters either the number of members is less than a predefined value N_T or the dissimilarity of all members of each one from one another is less than a threshold d_T.
- estimate parameters θ of Eq. (5) by performing the weighted-least-square on each leaf node using its associated data samples (see Eq. (6)).

In parameter identification, we assume the structure of the rule base is fixed. Unlike the structure learning, this step can be done incrementally, namely, the proposed model is capable of adapting based on new coming data pair $(\mathbf{x}_t, y_t)$. This involves

- as in inference, use Algorithm 2 finding the parent node p and its associated clusters $\{A_i \mid i = 1, ..., q\}$ and compute the projected vector onto the subspace derived by p, i.e., $\mathbf{z} = V\mathbf{x}$
- for all $i = 1, ..., q$ apply the sequential least square equations listed in Eq. (10) to update the parameters in the *consequent* part of the rules

IV. Experiments

A. Simulation

We show experimentally the proposed method's feature extraction and real-time learning capabilities on an artificial dataset. As a first test (2D-cross), we ran the method on noisy training data drawn from a two dimensional function

$$y = \max\{\exp(-10x_1^2), \exp(-50x_2^2), 1.25\exp(-5(x_1^2 + x_2^2))\} + N(0, 0.1^2)$$

as shown in Fig. 2 (a). This function is a combination of nonlinear and linear areas and an interesting test of learning and generalization capabilities of a learning algorithm [8]: learning system with simple model find it hard to capture the nonlinearity well, while more complex models easily overfit the linear areas. A second test (100d-cross) added 98 irrelevant noise features to the input, each having a density $N(0, 0.025^2)$. We thus obtain a 100-dimension input space. A third test (200d-cross) added another 100 irrelevant noise features with the density $N(0, 0.05)$ to the input space of the second test. The learning curves with these data set are illustrated in Fig. 2 (c). In all three cases, the normalized mean square error (nMSE) is reported on an independent test set (1681 points on a 41×41 grid in the unit-square in the input space). As the number of training number increasing, all nMSEs converged to the nice function approximation result whose $nMSE < 0.03$ after 100,000 training data points. Fig. 2 (b) shows the learned surface of the third test after 100,000 training samples presented. Fig. 2 (d) illustrates the execution time of both training and test processes with respect the number of training samples. It is interesting to note that the execution time increases linearly w.r.t. the number of training samples. The reason is that the learning and inference procedures have the logarithmic complexity and, thus, the average time of adding a training sample and retrieving a testing sample does not change much even though the size of tree has grow tremendously. As considering the third case (200d-cross), execution time on a 200-dimension data set takes only about 500 seconds for 100,000 sample, in other words, the average time for a sample is about 5 milliseconds. This is fast and is extremely important for later real-time robot navigation experiment.

In summary, the power of feature extraction is due to finding the MDF subspace. The real-time learning performance is achieved by the hierarchical organization of the rule base.

B. Range-based Collision-avoidance Behavior

In the following experiments, we trained the robot to learn the collision-avoidance behavior. The goal is to learn the mapping from a laser ranger's raw reading **x** ($\dim(\mathbf{x}) = 361$) to the robot's heading and speed $y = (\theta, v)^T$. The robot does not sense and store scene configuration (e.g., global map of the environment) nor the global robot position. That is, we assume that the current range map contains all of the information that is sufficient for robots to derive the next motor control signal. The robot's only goal is to move safely according to the scene: It has no target location. Such a navigation system is useful for applications where a human guides global motion but local motion is autonomous.

Experiment 1. In this simulated experiment, we interactively collected 1917 samples via a simulated robot in 16 scenarios.

TABLE I
THE RESULT OF THE CROSS-VALIDATION TEST.

	Range	Mean error (with attention)
θ	$[0, \pi]$	0.090
v	$[0, 1.0]$	0.005

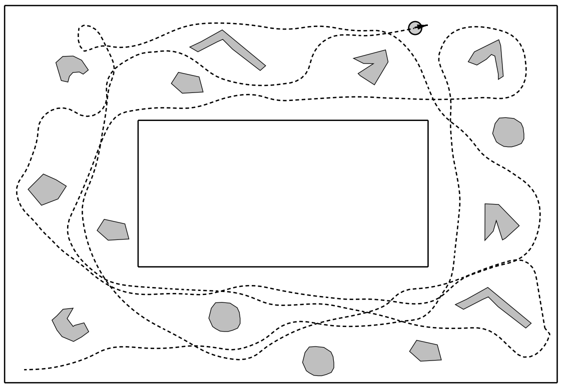

Fig. 3. A 10-minute run by the simulated robot with the attentional module. The solid dark lines denote walls and the small trace circles show the trajectory. Obstacles of irregular shapes are scattered about the corridor.

In order to test the generalization capability of the learning system, we performed a cross-validation test for the learned fuzzy system. The 1917 training samples were divided into 10 bins. Chose 9 bins for training and left one bin for testing. This procedure was repeated ten times, one for each choice of test bin. The average results are shown in Table 1.

The tests were performed in an environment different from the training scenarios. In Fig. 3, with attention, the simulated robot performed successfully a continuous 10-minute run. The robot's trajectory is shown by small trailing circles. Remember that no environmental map was stored across the laser maps and the robot had no global position sensors.

Experiment 2. Our mobile humanoid robot, Dav [9], was used to test our proposed framework. Dav was trained using pre-collected dataset with 4,655 samples. Some training range images around those corners and intersections are shown in Fig. 4. A continuous 15-minute run was performed by Dav in the corridor of the Engineering Building at Michigan State University. The corridor was crowded with high school students, as shown in Fig. 5. Dav successfully navigated in this dynamic changing environment without collisions with moving students. It is worth noting the testing scenarios were not the same as the training scenarios.

V. CONCLUSION

This paper describes a learning fuzzy framework to model system dynamics. The power of the proposed method is to enable the machine to learn a very complex function $y = f(\mathbf{x})$ between the input (e.g., sensory input **x**) and the desired behavior y. Such a function is typically so complex that

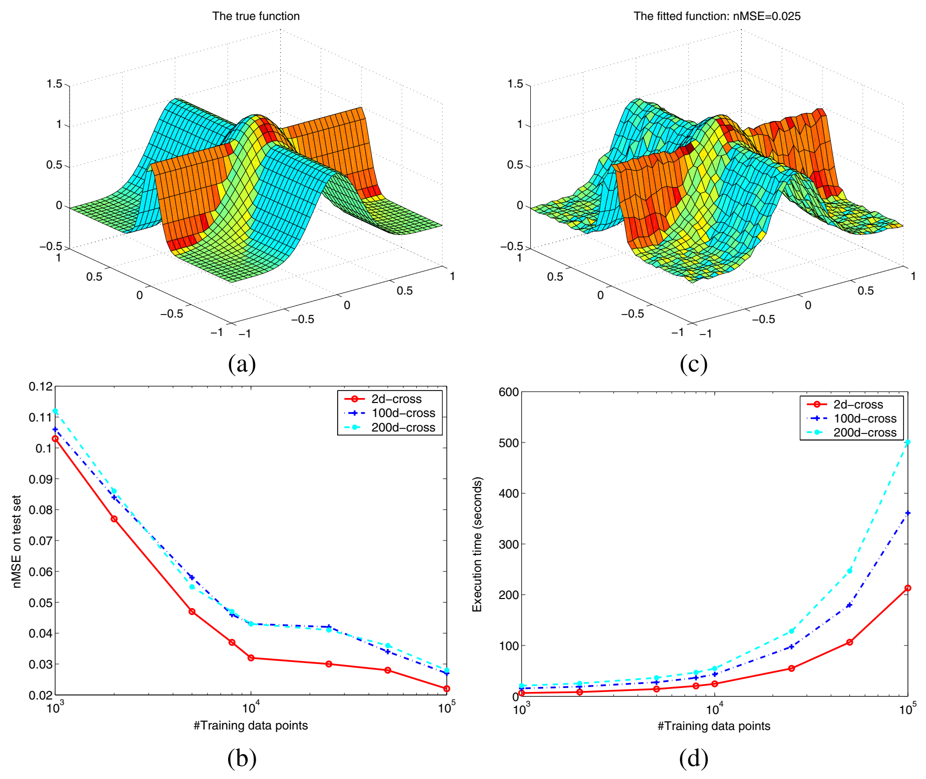

Fig. 2. Experiment results on the 2D-cross data set.

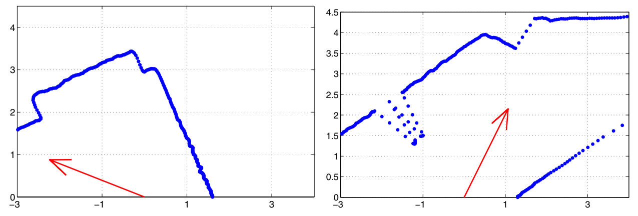

Fig. 4. A subset of training samples. The arrows denote the labeled heading for the robot.

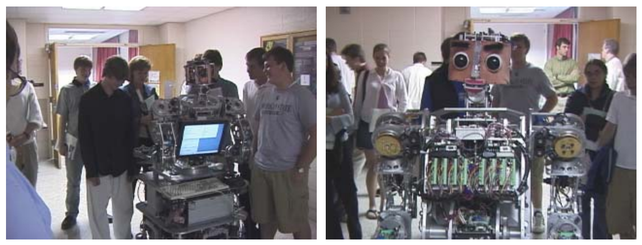

Fig. 5. Dav moved autonomously in a corridor crowded with people.

writing a program to simulate it accurately is not possible. The success of learning for such a high-dimensional input is mainly due to the discriminating feature extraction, and the real-time speed is due to the logarithmic time complexity.

The key idea of this paper is locally model-fitting whose learning algorithm first partitions the space into local regions. In each of those regions, a simple models (e.g., TS model) is used to model the input-output function. The number and the organization of local models account for the nonlinearity and the complexity of the problem. This method is suitable for incremental learning, especially in the situation where limited knowledge exists about the target system with high-dimensional input.

References

[1] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11(1-5):11–73, 1997.

[2] J. Friedman. Regularized discriminant analysis. *Journal of the American Statistical Association*, 84:165–175, 1989.

[3] W. S. Hwang and J. Weng. Hierarchical discriminant regression. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(11):1277–1293, 11 2000.

[4] Chin-Teng Lin and C.S. George Lee. *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems*. Prentice-Hall PTR, 1996.

[5] K. V. Mardia, J. T. Kent, and J. M. Bibby. *Multivariate Analysis*. Academic Press, London, New York, 1979.

[6] L. L. Scharf. *Statistical Signal Processing: Detection, Estimation, and Time Series Analysis*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1991.

[7] T. Takagi and M. Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man and Cybernetics*, 15(1):116–132, 1985.

[8] S. Vijayakumar and S. Schaal. Locally weighted projection regression: An o(n) algorithm for incremental real time learning in high dimensional space. In *Proc. of Seventeenth International Conference on Machine Learning (ICML2000)*, pages 1079–1086, Stanford, CA, June 19-July 2, 2000.

[9] S. Zeng, D. M. Cherba, and J. Weng. Dav developmental humanoid: The control architecture and body. In *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pages 974–980, Kobe, Japan, July 2003.