Dimensionality Reduction in Basis-function Networks:
exploiting the link with fuzzy system

K J Hunt[1], R Haas and R Murray-Smith

Daimler-Benz AG
Systems Technology Research Berlin

Abstract

We address the dimensionality problem in the training of basis function networks of various types. Some results which establish the functional equivalence of basis function networks and fuzzy inference systems are discussed, and this equivalence is exploited in the development of a generalised form of basis function network which has in general a lower dimension than the standard form of network. The functional equivalence result allows the employment of training algorithms from both the fields of neural networks and fuzzy systems. We summarise examples of such algorithms and point to their possible use in fuzzy control systems.

1. Summary

In this section we summarise the main technical points of the paper. The standard basis function network is described and the problems which arise when the input space has high dimension are highlighted. The Takagi-Sugeno model of fuzzy inference (the TS-model) is then introduced and motivates the establishment of the generalised basis function network (GBFN). The basis functions in the GBFN operate on only a subset of the input variables and the dimensionality problems are therefore considerably alleviated.

The standard basis function network is shown in figure 1. It is described by

$$\hat{y} = \hat{f}(\vec{\psi}) = \sum_{i=1}^{n} \rho_i(\vec{\psi}) \cdot \hat{f}_i. \qquad (1)$$

Here, $\hat{y} \in \mathcal{R}$ is the network output and $\vec{\psi} \in \mathcal{R}^{n_\psi}$ is the input vector. The network has n nonlinear processing units and the nonlinearity of the i-th unit is represented by the function $\rho_i(\vec{\psi})$. The output

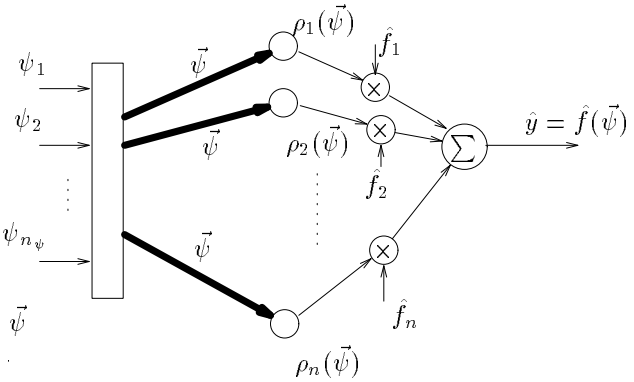

Figure 1: Standard basis function network

of each processing unit is multiplied by the scalar weight $\hat{f}_i$. A normalised form of equation (1) is often used, see [1] for further analysis of the side-effects of normalisation. This has the form

$$\hat{y} = \hat{f}(\vec{\psi}) = \frac{\sum_{i=1}^{n} \rho_i(\vec{\psi}) \cdot \hat{f}_i}{\sum_{i=1}^{n} \rho_i(\vec{\psi})}. \qquad (2)$$

A common form of basis function is the radial Gaussian although many other types including splines are possible. The use of ellipsoidal basis functions is discussed at this Workshop by Murray-Smith and Gollee [2]. For our purposes it suffices to consider the simple Gaussian basis function which has the form

$$\rho_i(\vec{\psi}) = \exp\left[-\frac{\|\vec{\psi} - \vec{c}_i\|^2}{2\sigma_i^2}\right] \qquad (3)$$

where $\|.\|$ denotes the Euclidean norm. $\vec{c}_i \in \mathcal{R}^{n_\psi}$ is the centre of unit i and σ_i its width. These equations explicitly reveal the dimensionality problem in basis function networks. The centre vector has the same dimension as the input vector so that the number of basis function units required to uniformly cover the input space increases exponentially with the dimension. Assuming, for example, a uniform distribution of n_d units per dimension the total number of units is $n_d^{n_\psi}$. Training of basis function networks usually involves a clustering

[1] Correspondence: Daimler-Benz AG, Alt-Moabit 91 B, D-10559 Berlin, Germany. Tel: + 49 30 399 82 275, FAX: + 49 30 399 82 107, Email: x@DBresearch-berlin.de where $x \in$ {hunt, haas, murray}.

approach to set the positions of the basis functions and this clearly becomes a serious problem with increasing dimension of the input space.

The Takagi-Sugeno model of fuzzy inference (see [3]) integrates fuzzy conditions and functional relationships between the input and output spaces. We denote the vector of input variables for the fuzzy system as $\vec{\psi} \in \mathcal{R}^{n_\psi}$. A key point about the TS-model is that each rule in the fuzzy system typically conditions only a subset of the input variables; we denote the vector of variables conditioned in rule i as $\vec{x}_i \in \mathcal{R}^{n_{x_i}}$, and therefore $\vec{x}_i \subset \vec{\psi}$ with $n_{x_i} \leq n_\psi$.

Rules in the TS-model have the canonical form

$$R_i : \textbf{if } x_{i1} \text{ is } A_{i1} \wedge x_{i2} \text{ is } A_{i2} \wedge \ldots \wedge x_{in_{x_i}} \text{ is } A_{in_{x_i}} \textbf{ then } \hat{y} = \hat{f}_i(\vec{\psi}).$$

The number of rules is denoted as n^r so that $i = 1 \ldots n^r$. A_{ij} are linguistic labels of fuzzy sets describing the qualitative state of the input variables, $\wedge$ is a fuzzy conjunction operator (usually of T-Norm-characteristics) and the rule output $\hat{y}$ is a linear or nonlinear function of the input variables. Rule inference is realized by first calculating the fulfilment, or firing strength, of the premise part as

$$\mu_{R_i}(\vec{x}_i) = \mu_{i1}(x_{i1}) \wedge \ldots \wedge \mu_{in_{x_i}}(x_{in_{x_i}}).$$

Here, μ_{ij} is the membership function of fuzzy set A_{ij}. The firing strength is then multiplied with the output function, defining a locally valid model on the support of the cartesian product of the fuzzy sets involved in building the premise. The fulfilment of the premise part can be calculated using multiplication or the minimum operator. The overall output of the TS-model is defined as the sum

$$\hat{y} = \hat{f}(\vec{\psi}) = \sum_{i=1}^{n^r} \mu_{R_i}(\vec{x}_i) \cdot \hat{f}_i(\vec{\psi}), \qquad (4)$$

although a normalised version of this expression can also be used. The normalised version is

$$\hat{y} = \hat{f}(\vec{\psi}) = \frac{\sum_{i=1}^{n^r} \mu_{R_i}(\vec{x}_i) \cdot \hat{f}_i(\vec{\psi})}{\sum_{i=1}^{n^r} \mu_{R_i}(\vec{x}_i)}. \qquad (5)$$

Consideration and comparison of equations (1) and (4) reveals a strong structural similarity between the standard basis function network and the TS-model of fuzzy inference. Indeed, the functional equivalence of the TS-model and radial basis function networks was established, under certain technical conditions, by Jang and Sun [4] and further developed by Hunt *et al* [5]. Hunt *et al* [6] went on to extend the functional equivalence result to spline-based networks. In the latter two references the results were based upon a generalised form of basis function network. This network type is inspired partly from the functional form of the TS-model where only a subset of the input variables are

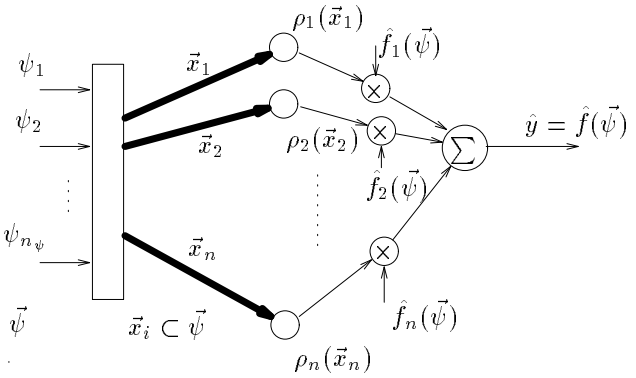

Figure 2: Generalised basis function network

processed by each rule, and where the 'local models' $\hat{f}_i(\vec{\psi})$ are defined at the output of each rule. The firing strength of each rule, as embodied in the non-linear function μ_{R_i}, is clearly equivalent to the output of each processing unit ρ_i in a basis function network. The functional equivalence result is formally summarised in the next section.

The network which is motivated by these considerations is known as the *generalised basis function network* (see figure 2) and is described by

$$\hat{y} = \hat{f}(\vec{\psi}) = \sum_{i=1}^{n} \rho_i(\vec{x}_i) \cdot \hat{f}_i(\vec{\psi}). \qquad (6)$$

The normalised form of this network is

$$\hat{y} = \hat{f}(\vec{\psi}) = \frac{\sum_{i=1}^{n} \rho_i(\vec{x}_i) \cdot \hat{f}_i(\vec{\psi})}{\sum_{i=1}^{n} \rho_i(\vec{x}_i)}. \qquad (7)$$

This network has the desirable feature that each processing unit processes only a subset of the network input vector ($\vec{x}_i \subset \vec{\psi}$), and also that the output weights are functions of the input vector. These two features combine to significantly reduce the dimensionality problem in the training of basis function networks. First, the parameters of each unit (e.g., the centres) are defined on a lower dimensional space than the input vector. Second, the use of local models $\hat{f}_i(.)$ provides a much richer model structure which results in the requirement for fewer units to reach a given degree of model fidelity. Typically, local linear models are used. The interpretation of the GBFN as a 'local model network' has been considered by Johansen [7, 8, 9]. In the local model framework the input to the basis functions are required to capture the non-linear effects in the system. Thus, the $\vec{x}_i$ can be interpreted as operating point vectors. The basis functions use the operating point vectors to partition the space into a set of overlapping regions within which the system behaviour is characterised by the corresponding local model - the basis functions interpolate the linear local models.

2. Functional Equivalence

Here, the functional equivalence results referred to above are summarised. Full details can be found in references [5] and [6].

As mentioned above each processing unit in the BFN, $\rho_i(\vec{x}_i)$, is associated directly with a single rule in the fuzzy system, $\mu_{R_i}(\vec{x}_i)$. We therefore proceed, without any loss of generality, under the following assumptions:

(A.1) The number of basis function processing units is equal to the number of fuzzy if-then rules, i.e., $n = n^r$.

(A.2) Both the basis function network and the fuzzy inference system use the same method to calculate their overall outputs, i.e., either the normalised calculation ((5) and (7)) or the unnormalised calculation ((4) and (6)).

We are now ready to state the functional equivalence theorem:

Theorem 1 *The generalised basis function network is functionally equivalent to the TS-model of fuzzy inference if and only if the following condition is satisfied:*

(C.1) *The activation value of the i-th basis function network processing unit is equal to the firing strength of the i-th fuzzy if-then rule, i.e.,*

$$\rho_i(\vec{x}_i) = \mu_{R_i}(\vec{x}_i).$$

□

Proof

Under assumption (A.1) the TS-model of fuzzy inference can be expressed as

$$\hat{y} = \hat{f}(\vec{\psi}) = \frac{\sum_{i=1}^{n} \mu_{R_i}(\vec{x}_i) \cdot \hat{f}_i(\vec{\psi})}{\sum_{i=1}^{n} \mu_{R_i}(\vec{x}_i)}$$

or in unnormalised form as

$$\hat{y} = \hat{f}(\vec{\psi}) = \sum_{i=1}^{n} \mu_{R_i}(\vec{x}_i) \cdot \hat{f}_i(\vec{\psi}).$$

When (C.1) is satisfied these equations become

$$\hat{y} = \hat{f}(\vec{\psi}) = \frac{\sum_{i=1}^{n} \rho_i(\vec{x}_i) \cdot \hat{f}_i(\vec{\psi})}{\sum_{i=1}^{n} \rho_i(\vec{x}_i)}$$

and

$$\hat{y} = \hat{f}(\vec{\psi}) = \sum_{i=1}^{n} \rho_i(\vec{x}_i) \cdot \hat{f}_i(\vec{\psi}).$$

These equations are exactly equivalent, respectively, to the BFN equations (7) and (6).

□

We recall that the TS-model of fuzzy inference has firing strengths $\mu_{R_i}(\vec{x}_i)$ which are composed from the univariate membership functions $\mu_{ij}(x_{ij})$. This composition is normally done using either the minimum or the product operator. It is clear that composition of corresponding BFN basis functions should be achieved using the same operator. The following statement results.

Corollary 1 *If one paradigm (fuzzy or neural) uses the minimum operator then so should the other. Conversely, if one paradigm uses the product operator then so should the other.*

The implications of this corollary are discussed more fully in the sequel. For the moment it suffices to point out here that basis function networks are usually based upon the product operator (e.g., RBF and spline networks) implementing the tensor product. In the transformation of such networks to the equivalent fuzzy system the product operator is required. On the other hand, transformation in the opposite direction (fuzzy $\rightarrow$ network) may well involve a fuzzy system using the minimum operator. The corresponding network will therefore require also to formulate basis functions based on compositions of univariate basis functions with the minimum operator. This is clearly unconventional for networks but presents no difficulty - the only requirement is to ensure (C.1) in theorem 1. Moreover if the fuzzy partition consists entirely of triangular sets a short algebraic transformation shows that this can be regarded as the establishment of a weighted L_∞-norm, defining a metric structure in the input domain that serves as a basis for the construction of a RBF net with piecewise linear basis functions.

3. Network Weight Optimisation

The training procedure for a generalised basis function network involves a number of steps. First the structure of the network must be established. This involves determining the input variables for the basis functions and then choosing the basis functions' internal parameters (e.g. centres and widths). These steps amount to determining the way in which the input space is locally partitioned. Constructive algorithms for this are described in the companion paper by Murray-Smith and Gollee [2]. The possible use of algorithms from the fuzzy systems domain is discussed in the sequel (section 4). In this paper we restrict attention to the determination of the parameters of the local models $\hat{f}_i$.

We now focus specifically on non-linear dynamic systems modelling. The underlying system under consideration is assumed to have the form

$$\begin{aligned} y(t) &= f(y(t-1),\ldots y(t-n_y),u(t-k),\ldots \\ &\quad u(t-k-n_u))+e(t), \end{aligned} \tag{8}$$

or

$$y(t) = f(\vec{\psi}(t-1))+e(t) \tag{9}$$

with

$$\begin{aligned} \vec{\psi}(t-1) &= [y(t-1),\ldots y(t-n_y),u(t-k),\ldots \\ &\quad u(t-k-n_u)]^T. \end{aligned} \tag{10}$$

If the local models are linear, i.e.

$$\hat{y}_i(t) = \hat{f}_i(\vec{\psi}(t-1)) = \vec{\psi}^T(t-1)\vec{\theta}_i, \tag{11}$$

then the overall global model is

$$\hat{y}(t) = \hat{f}(\vec{\psi}(t-1)) = \sum_{i=1}^{n} \vec{\psi}^T(t-1)\vec{\theta}_i\rho_i(\vec{x}_i). \tag{12}$$

Given a series of input-output measurements $(u(t),y(t))$, $t=1\ldots N$, and a set of model validity functions ρ_i we wish to determine the parameters of the local models $\hat{f}_i$ in (6).

A global criterion for estimation of the parameters of the model is

$$J_N(\vec{\theta}) = \frac{1}{N}\sum_{t=1}^{N}\alpha(t)(y(t)-\hat{y}(t))^2, \tag{13}$$

and the vector containing all the estimated model parameters (see equation (21) below) is obtained as

$$\hat{\vec{\theta}} = \arg\min J_N(\vec{\theta}). \tag{14}$$

In (13) $\alpha(t)$ are observation weights which can be attached to each measurement.

A second possibility is to locally estimate the parameters of the local models (11) by defining a set of local criteria, as in [10]. For the i-th local model the estimation criterion is

$$J_{iN_i}(\vec{\theta}_i) = \frac{1}{N_i}\sum_{t=1}^{N_i}\alpha_i(t)(y(t)-\hat{y}_i(t))^2. \tag{15}$$

In this case the estimate of the local model parameter vector $\vec{\theta}_i$ is given by

$$\hat{\vec{\theta}}_i = \arg\min J_{iN_i}(\vec{\theta}_i). \tag{16}$$

For the global criterion it is possible to set $\alpha(t)=1$ for all t, or to select $\alpha(t)$ to achieve desirable parameter tracking behaviour. For the local criteria, on the other hand, the weights must be chosen to take direct account of the interactions of the validity functions[1]. Our confidence in a given observation regarding its relevance for the i-th local model is directly reflected in the i-th validity function. The local weighting functions should therefore be set as

$$\alpha_i(t) = \rho_i(\vec{x}_i(t)), \tag{17}$$

which results in a set of local criteria

$$J_{iN_i}(\vec{\theta}_i) = \frac{1}{N_i}\sum_{t=1}^{N_i}\rho_i(\vec{x}_i(t))(y(t)-\hat{y}_i(t))^2. \tag{18}$$

The advantages of the local approach are:

- The use of the local criteria is more computationally efficient since the computational load for an optimsation algorithm such as singular value decomposition increases as $(N^2n_\theta + Nn_\theta^2 + \min(N,n_\theta)^3)$, where n_θ is the total number of estimated parameters [11]. In local methods n_θ is obviously far smaller than in global ones, and effort scales linearly with increasing numbers of local models.
- A related point is that only a small fraction of the local models need to be updated with each observation (due to our definition of the validity functions). With global estimation every observation affects all parameters.
- In the global case the regressor vector (20) will have many elements which are close to zero since only a small number of validity functions are significantly non-zero. This can lead to poor conditioning of the optimisation equations and numerical problems.
- Some local models may be pre-defined based on *a priori* knowledge. The parameters of these models need not be updated based on the observations, and this calls for local estimation of the other local models.

We now develop batch and on-line estimation algorithms for the GLMN.

3.1. Estimation in Uncorrelated Noise

Since all the signals in $\vec{\psi}$ are measurable (according to (10)) the global model can be written as a linear regression

$$\hat{y}(t) = \vec{\phi}^T(t-1)\vec{\theta} \tag{19}$$

where the regression vector $\vec{\phi}$ is defined as

$$\begin{aligned} \vec{\phi}(t-1) &= [\rho_1\vec{\psi}^T(t-1),\rho_2\vec{\psi}^T(t-1),\ldots \\ &\quad \rho_n\vec{\psi}^T(t-1)]^T \end{aligned} \tag{20}$$

and the parameter vector simply consists of the local model parameters,

$$\vec{\theta} = [\vec{\theta}_1^T,\vec{\theta}_2^T,\ldots\vec{\theta}_n^T]^T. \tag{21}$$

[1] In the global case the validity functions appear directly in the global model output $\hat{y}$ through definition of the regressor vector - see below.

If instead we use local estimation each local model forms a linear regression as follows:

$$\hat{y}_i(t) = \vec{\psi}^T(t-1)\vec{\theta}_i = \vec{\phi}^T(t-1)\vec{\theta}. \qquad (22)$$

In this case $\vec{\phi}(t-1) = \vec{\psi}(t-1)$ and $\vec{\theta} = \vec{\theta}_i$ for each local estimator.

3.1.1 Batch Estimation: For the global linear regression defined by equations (12) and (19) – (21) together with the global criterion (13) the off-line estimate of the parameters is easily determined to be [12]

$$\begin{aligned}\hat{\vec{\theta}}(N) &= [\sum_{t=1}^{N} \alpha(t)\vec{\phi}(t-1)\vec{\phi}^T(t-1)]^{-1} \cdot \\ &\quad \sum_{t=1}^{N} \alpha(t)\vec{\phi}(t-1)y(t). \qquad (23)\end{aligned}$$

For the local linear regressions defined by equation (22) together with the criteria (18) the least-squares estimate for the local parameters is

$$\begin{aligned}\hat{\vec{\theta}}_i(N_i) &= [\sum_{t=1}^{N_i} \rho_i(\vec{x}_i(t))\vec{\phi}(t-1)\vec{\phi}^T(t-1)]^{-1} \cdot \\ &\quad \sum_{t=1}^{N_i} \rho_i(\vec{x}_i(t))\vec{\phi}(t-1)y(t). \qquad (24)\end{aligned}$$

3.1.2 Recursive Estimation: In the case of uncorrelated noise the optimal unbiased parameter estimate can be asymptotically recovered with the recursive least-squares (RLS) algorithm. The global parameter estimate at t is calculated with the following algorithm:

$$\hat{\vec{\theta}}(t) = \hat{\vec{\theta}}(t-1) + L(t)(y(t) - \vec{\phi}^T(t-1)\hat{\vec{\theta}}(t-1)), \quad (25)$$

$$L(t) = \frac{P(t-1)\vec{\phi}(t-1)}{\frac{1}{\alpha(t)} + \vec{\phi}^T(t-1)P(t-1)\vec{\phi}(t-1)}, \quad (26)$$

$$\begin{aligned}P(t) &= P(t-1) - \\ &\quad \frac{P(t-1)\vec{\phi}(t-1)\vec{\phi}^T(t-1)P(t-1)}{\frac{1}{\alpha(t)} + \vec{\phi}^T(t-1)P(t-1)\vec{\phi}(t-1)}. \qquad (27)\end{aligned}$$

The recursive equations for the local parameter estimators have the same form as above. The $\hat{\vec{\theta}}$, L and P are simply replaced by local values $\hat{\vec{\theta}}_i$, L_i and P_i. The local $\vec{\theta}$ and $\vec{\phi}$ are defined as in (22), and the weights $\alpha(t)$ are set as in the batch case to be equal to the validity functions, $\alpha_i(t) = \rho_i(\vec{x}_i(t))$.

4. Training Algorithms from Fuzzy Systems

The first approaches to the design of learning fuzzy systems were based on the Takagi-Sugeno model. They decomposed the problem into a non-linear optimisation part of generating the fuzzy partition (type, shape and number of fuzzy sets for each dimension, usually done by cluster analysis) and a linear part of fixing local model parameters with standard least squares techniques [3],[13]. Since then many other schemes have been proposed mostly inspired by the fusion of neural network learning and fuzzy inference mechanisms. Other approaches are based on the use of genetic algorithms [14]. Regardless which algorithm is used, a learning technique has to fix the structure of the fuzzy system, i.e. types of membership function (Gaussian, triangular, etc.), condition and consequent part of rules and overall number of rules and the free parameters, i.e shape and centres of fuzzy sets for input and output variables. For both parametric and structural adjustment, *a priori* knowledge is important and can improve convergence properties and the interpretability of the trained system.

4.1. Structural Learning

Starting with a suitable fuzzy partition the process of rule generation can be done automatically with the following algorithm which has been proposed by Wang and Mendel [15] and which scans the observation data. Let $Fuzzy(dom(x_{ij}))$ denote the set of membership functions defined on the domain $dom(x_{ij})$ and assume that the input-output data is partitioned in a training set $D_t = \{(\vec{x}_i, y_i)\}$ $i = 1, \ldots, n_t$ and a verification part $D_v = \{(\vec{x}_k, y_k)\}$ $k = 1, \ldots, n_v$. For each observation $(\vec{x}_i, y_i)$ we generate a rule

$$R_i : A_{i1} \times A_{i2} \times \ldots \times A_{in_i} \longmapsto B_i \qquad (28)$$

where

$$A_{ij} = \arg \max_{A \subset Fuzzy(dom(x_{ij}))} \{\mu_A(x_i)\} \qquad (29)$$

and

$$B_i = \arg \max_{B \subset Fuzzy(dom(y))} \{\mu_B(y)\}. \qquad (30)$$

If there are conflicting outputs connected with the same conditional part we select the rule with maximum output. This learning scheme produces a rulebase $RB = \bigcup_{i=1}^{n_r} R_i$. The number of rules depends on the data distribution and the resolution of the fuzzy partition (i.e. the number of fuzzy sets used to cover the input variables). High dimensional inputs combined with a fine fuzzy partition can lead to very large rulebases which are difficult to handle. An alternative arises from a generalized version of the well known product space clustering algorithm, that has been proposed by Kosko [16]. Product space clustering differs from Wang and

Mendel's algorithm in the use of cluster analysis as preprocessing before rules are generated. It reduces the number of rules significantly. Let us assume a cluster analysis of the observation data has produced n_c cluster centres. The generalized algorithm takes each cluster centre $\vec{c}$ and generates a multi-conditional rule $R_i : A_{i1} \times A_{i2} \times \ldots \times A_{in_i} \longmapsto B_i$ where the fuzzy sets A_{ij} ,B_i satisfy the condition

$$\begin{aligned} A_{ij} &= \arg \max_{A \subset Fuzzy(dom(c_{ij}))} \{\mu_A(c_{ij})\} \\ B_i &= \arg \max_{B \subset Fuzzy(dom(y))} \{\mu_B(y)\}. \end{aligned} \quad (31)$$

Kavli's ASMOD algorithm for the automatic adaptation of spline based basis function models is also very applicable to fuzzy based modelling methods, especially in high dimensional spaces [17].

4.2. Parametric Learning

For the moment, let us consider a fixed fuzzy partition of the input space. If the fuzzy system can be transformed to a Takagi-Sugeno model, local model parameters can be adjusted in the same way as described in section 3. This is a direct consequence of the close relationship between the two modelling techniques. The situation is more difficult if the parameters of fuzzy sets in the input partition should be trained, too. It is possible to use a gradient search algorithm which can be applied to basis function nets and which treats the basis functions (fuzzy rules) as a hidden layer. Consider a fuzzy system of Takagi-Sugeno type with Gaussian membership functions, constant local models $\hat{f}_i = \theta_i \in \mathcal{R}$ and the unnormalized output $f_{TS}(\vec{x}) = \sum_{i=1}^{r} \mu_{R_i}(\vec{x}_i)\,\theta_i$. Let $\sigma_j^{(i)}$ denote the width coefficients of the jth fuzzy set in the ith rule arranged as a matrix $\Delta_i = diag(\sigma_1^{(i)}, \ldots, \sigma_n^{(i)})$. The system is equivalent to an ellipsoidal basis function net [2] with basis functions $\rho_i(\Psi) = \Phi(d_i) = \mu_{R_i}(\vec{x}_i)$, where the distance measure d_i (reflecting the receptive field of a rule's premise) is defined by means of the weighted euclidean metric $\|\cdot\|_{2,\Delta_i}$ as follows:

$$d_i(\vec{x}) = \frac{1}{2}\|\vec{x} - \vec{c}_i\|^2_{2,\Delta_i} = \frac{1}{2}(\vec{x} - \vec{c}_i)\Delta_i(\vec{x} - \vec{c}_i)' \quad (32)$$

The inner product in (32) gives:

$$d_i = \frac{1}{2}\sum_{j=1}^{n} \sigma_j^{(i)}(x_j - c_{ij})^2 \quad (33)$$

Let the instantaneous error be defined as $E_t = \frac{1}{2}e_t^2 = \frac{1}{2}[y_t - \hat{y}_t]^2$ and let $\alpha_1, \ldots, \alpha_3$ denote the learning coefficients for gradient descent. The local model parameters are adjusted by gradient search (for convenience the subscript t in y_t and the vector variable $\vec{x}$ in Φ_i are omitted). Thus, the weight change $\Delta\theta_i$ applied to the ith local model is computed as follows:

$$\begin{aligned} \Delta\theta_i &= -\alpha_1 \frac{\partial E}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \theta_i} \\ &= \alpha_1 (y - \hat{y})\,\Phi_i(d_i) = \alpha_1 e \Phi_i(d_i) \end{aligned} \quad (34)$$

with $e = (y - \hat{y})$ as the instantanous error for the given data sample. The basis functions are reshaped and moved in input space by treating them as a hidden layer. The width parameters are adjusted by gradient descent, yielding

$$\Delta\sigma_j^{(i)} = -\alpha_2 \frac{\partial E}{\partial \sigma_j^{(i)}} = -\alpha_2 \frac{\partial E}{\partial \Phi_i} \frac{\partial \Phi_i}{\partial d_i} \frac{\partial d_i}{\partial \sigma_j^{(i)}}, \quad (35)$$

where the partial derivatives can be easily computed as follows:

$$\begin{aligned} \frac{\partial E}{\partial \Phi_i} &= \frac{\partial E}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \Phi_i} = -e\theta_i \\ \frac{\partial \Phi_i}{\partial d_i} &= -\Phi_i \\ \frac{\partial d_i}{\partial \sigma_j^{(i)}} &= \frac{1}{2}(x_j - c_{ij})^2 \end{aligned} \quad (36)$$

Substituting these results in equation (35), we get:

$$\Delta\sigma_j^{(i)} = -\frac{\alpha_2}{2} e\theta_i \Phi_i(d_i)(x_j - c_{ij})^2 \quad (37)$$

Finally, the centres are adjusted using the following derivation:

$$\begin{aligned} \Delta c_{ij} &= -\alpha_3 \frac{\partial E}{\partial c_{ij}} = -\alpha_3 \frac{\partial E}{\partial \Phi_i} \frac{\partial \Phi_i}{\partial d_i} \frac{\partial d_i}{\partial c_{ij}} \\ &= -\alpha_3 e \theta_i \Phi_i(d_i) \sigma_j^{(i)} (c_{ij} - x_j) \end{aligned} \quad (38)$$

Concrete applications of the techniques described above can be found in [6]. A simple but often effective procedure to design an autonomous controller for processes which are controlled by humans is to copy the human who interacts with the plant. There may be many measurement signals which have to be considered to derive the proper control sequence so that the learning system might be confronted with a high dimensional problem.

Let us assume we want to train a basis function net which is capable of mimicking the behaviour of the human plant driver. Exploiting the link to fuzzy inference systems we can try to collect expert rules which are represented as fuzzy production rules and use this knowledge to pre-structure the basis function net. It is obvious that this knowledge acquisition is imperfect and does not fully reflect the human behaviour. Hence, the observation data help to optimise the controller. The optimisation technique developed in the last section can be used to reconfigure centres and shapes of the basis function without destroying the overall rule structure that has been derived from the expert interrogation.

References

[1] R. Shorten and R. Murray-Smith, "On Normalising Basis Function networks," in *4th Irish Neural Networks Conf., Univ. College Dublin*, Sept. 1994.

[2] R. Murray-Smith and H. Gollee, "A constructive learning algorithm for local model networks," in *Proc. IEEE Workshop on Computer-intensive methods in control and signal processing, Prague, Czech Republic*, 1994.

[3] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Trans. on Systems, Man and Cybernetics*, vol. SMC-15, pp. 116–132, January/February 1985.

[4] J. S. Roger Jang and C. T. Sun, "Functional equivalence between radial basis function networks and fuzzy inference systems," *IEEE Trans. on Neural Networks*, vol. 4, pp. 156–159, January 1993.

[5] K. J. Hunt, R. Haas, and R. Murray-Smith, "Extending the functional equivalence of radial basis function networks and fuzzy inference systems," 1994. Submitted for publication.

[6] K. J. Hunt, R. Haas, and R. Murray-Smith, "On the functional equivalence of fuzzy inference systems and spline-based networks," 1994. Submitted for publication.

[7] T. A. Johansen and B. A. Foss, "A NARMAX model representation for adaptive control based on local models," *Modelling, Identification and Control*, vol. 13, no. 1, pp. 25–39, 1992.

[8] T. A. Johansen and B. A. Foss, "Constructing NARMAX models using ARMAX models," *Int. J. Control*, vol. 58, no. 5, pp. 1125–1153, 1993.

[9] B. A. Foss and T. A. Johansen, "On local and fuzzy modelling," in *Proc. 3rd Int. Conf. on Industrial Fuzzy Control and Intelligent Systems, Houston, Texas*, 1993.

[10] R. Murray-Smith, "Local Model Networks and Local Learning," in *Fuzzy Duisburg, '94*, pp. p404–409, 1994.

[11] B. Noble and J. W. Daniel, *Applied linear algebra*. Prentice–Hall Int., 3rd ed., 1988.

[12] L. Ljung and T. Söderström, *Theory and Practice of Recursive Identification*. London: MIT Press, 1983.

[13] M. Sugeno and T. Yasukawa, "A fuzzy-logic-based approach to qualitative modeling," *IEEE Trans. on Fuzzy Systems*, pp. 7–31, February 1993.

[14] R. Haas and K. J. Hunt, "Genetic based optimisation of a fuzzy-neural vehicle controller," in *Proc. Fuzzy Systems Conference, Munich*, 1994.

[15] L.-X. Wang and J. M. Mendel, "Generating fuzzy rules by learning from examples," in *Proceedings of the IEEE International Symposium on Intelligent Control, Arlington, USA*, pp. 263–268, 1991.

[16] B. Kosko, *Neural Nets and Fuzzy Systems*. Prentice-Hall, 1992.

[17] T. Kavli, *Learning Principles in Dynamic Control*. PhD thesis, University of Oslo, 1992.