Neuro-Fuzzy Modeling

Robert Babuška

Delft Center for Systems and Control
Delft University of Technology
The Netherlands

r.babuska@dcsc.tudelft.nl
www.dcsc.tudelft.nl/˜babuska

Outline

- Introduction
- Fuzzy Systems and Neural Networks
  - a brief refresher: fuzzy sets, TS model, FF neural net
- Neuro-Fuzzy Modeling
  - structures, learning, problems
- Examples
  - static function, dynamic system, application example
- Conclusions

Modern Control Applications

- complicated MIMO processes: interactions, nonlinearity
- autonomous control: integration of control, supervision, fault detection, reconfiguration

$\Downarrow$

$\boxed{\text{need for effective nonlinear modeling and control techniques}}$

Tools: $\begin{cases} \text{– mechanistic models} \\ \text{– polynomials, splines, look-up tables} \\ \text{– neural networks, (neuro-)fuzzy systems, ...} \end{cases}$

Requirements on Models

◇ effective use of prior knowledge and data

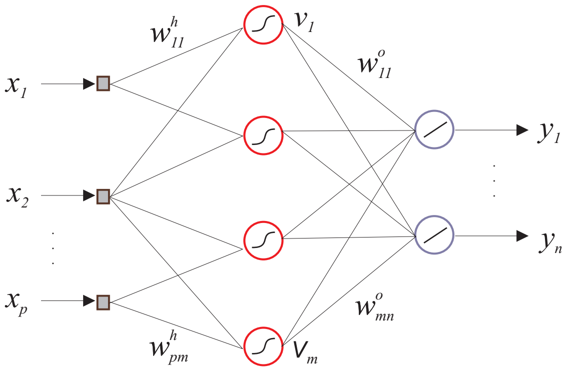
◇ accuracy, reliability (generalization)

◇ transparency, readability, interpretation

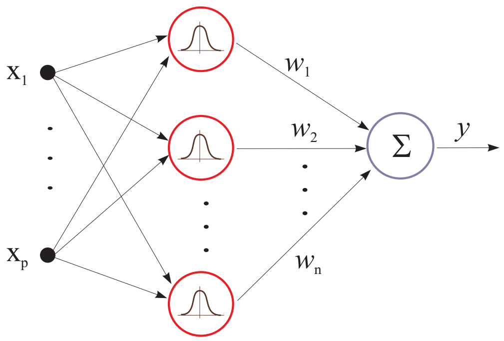
Multi-Layer Neural Networks

– primarily based on learning from data

– black-box model



Radial Basis Function Network

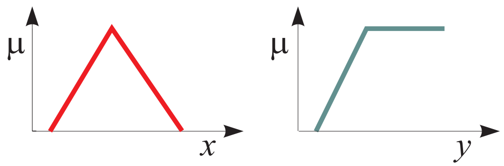


Fuzzy Systems

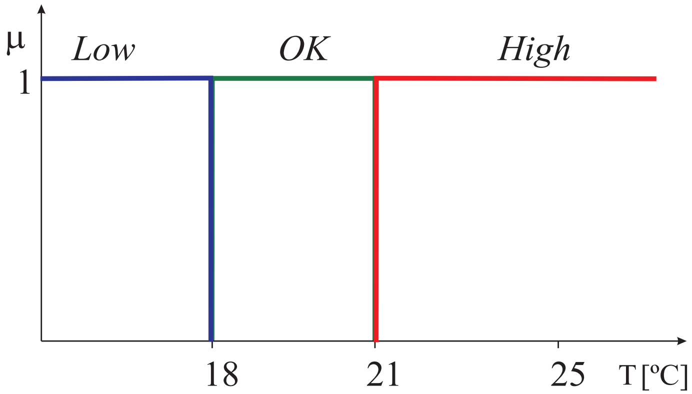
– model represented as a set of rules

– primarily based on expert knowledge

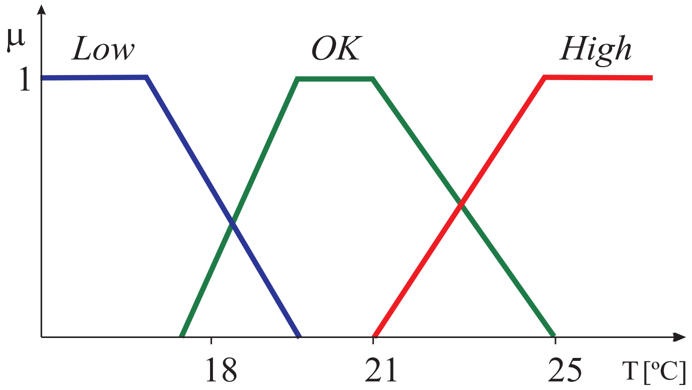
If x is Medium **then** y is Large



Conventional Sets



Fuzzy Sets



Fuzzy Set, Membership Function

$$\mu_A(x)\colon X \to [0, 1]$$

$$\mu_A(x) = \begin{cases} 1 & x \text{ is full member of } A \\ (0, 1) & x \text{ partly belongs to } A \\ 0 & x \text{ is not member of } A \end{cases}$$

element-wise definition: $A = \{\mu_A(x_i)/x_i \mid x_i \in X\}$

Takagi–Sugeno Fuzzy Model

$$\textbf{If } x \textbf{ is Medium then } y = f_m(x)$$

– each rule is a local regression model

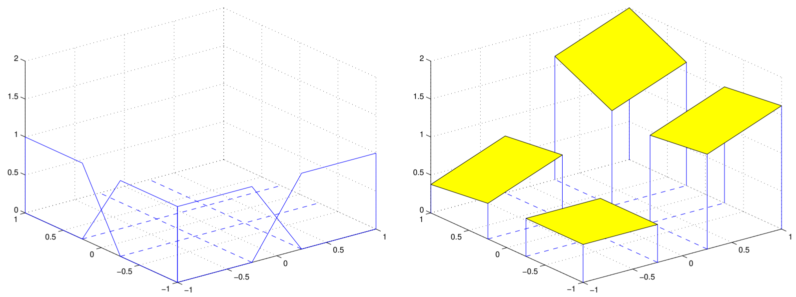
– f_m usually affine linear: $y = a_m x + b_m$

with special cases: $b_m = 0$ (homogeneous)

$a_m = 0$ (singleton, zero-order TS)

TS Model – Example

If x_1 **is** *Negative* **and** x_2 **is** *Negative* **then** $y_1 = [x_1\ x_2]\mathbf{a}_1 + b_1$
If x_1 **is** *Negative* **and** x_2 **is** *Positive* **then** $y_2 = [x_1\ x_2]\mathbf{a}_2 + b_2$
...



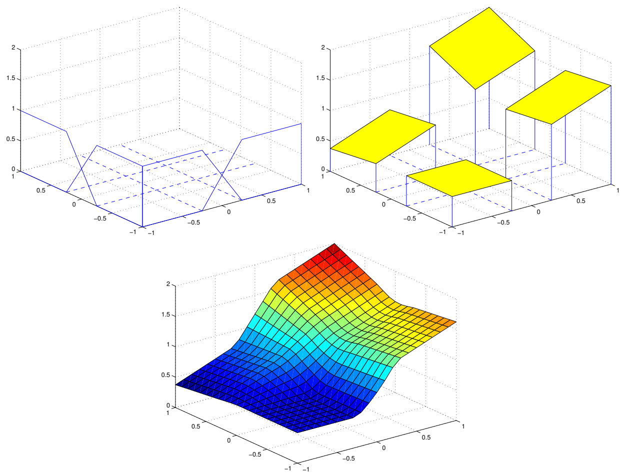
Inference in the TS Model

$R_i: \quad$ **If** x_1 **is** F_1^i **and** x_2 **is** F_2^i **and** $\ldots$ **and** x_q **is** F_q^i
then $y_i = \mathbf{a}_i^T\mathbf{x} + b_i, \qquad i = 1, 2, \ldots, l$

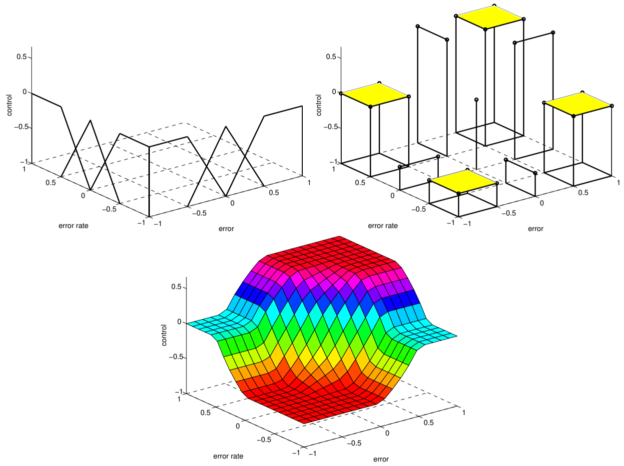
$$\beta_i(x) = \mu^i_{F_1}(x_1) \star \mu^i_{F_2}(x_2) \star \cdots \star \mu^i_{F_q}(x_q)$$

$$w_i(x) = \frac{\beta_i(x)}{\sum_{i=1}^{l} \beta_i(x)} \qquad\qquad y = \sum_{i=1}^{l} w_i(x)(\mathbf{a}_i^T\mathbf{x} + b_i)$$

TS Model – Geometrical Representation



Zero-Order TS model



Obtaining TS models

- Taylor expansion $\mathbf{A}_i = \left.\frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}}\right|_{(\mathbf{x}_i,\mathbf{u}_i)}$
  - local linearization
- Linearizing transformation
  - global parameterization
- Systems identification
  - neuro-fuzzy learning
  - tree-construction, fuzzy clustering, etc.

Linearizing transformation

nonlinear system: $y_t = \sin(x)$

TS representation:

$$\begin{cases} R_1: & \textbf{If } x \textbf{ is } F^1 \textbf{ then } y = a_1 x \\ R_2: & \textbf{If } x \textbf{ is } F^2 \textbf{ then } y = a_2 x \end{cases}$$

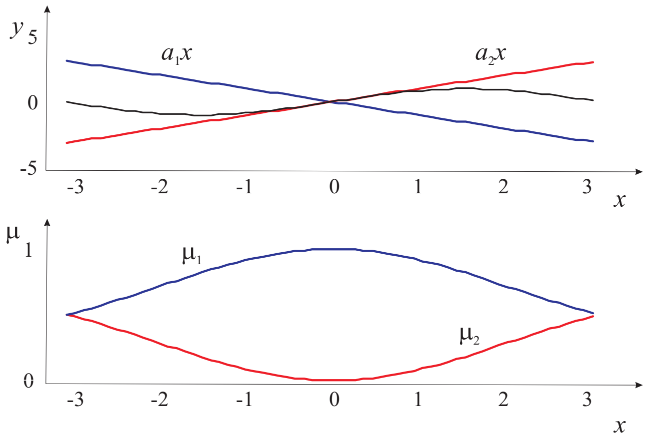
choose $\mu_2(x) = 1 - \mu_1(x)$ $\quad y = \mu_1(x) a_1 x + [1 - \mu_1(x)] a_2 x$

$$\sin(x) = \mu_1(x) a_1 x + [1 - \mu_1(x)] a_2 x$$

$$\Rightarrow \mu_1(x) = \begin{cases} \frac{\sin(x) - a_2 x}{a_1 x - a_2 x} & x \neq 0 \\ 0 & x = 0 \end{cases}$$

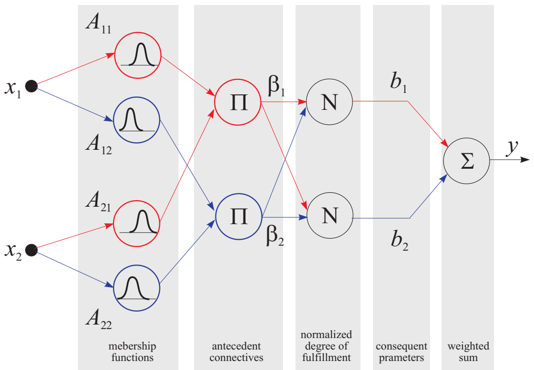
$$x \in [-\pi, \pi] \quad \Rightarrow \quad a_1 = -1, \quad a_2 = 1, \quad \mu_1(x) = \frac{x - \sin(x)}{2x}$$

Linearizing transformation – cont'd



Neuro-Fuzzy Network (Zero-Order TS)

If x_1 **is** A_{11} **and** x_2 **is** A_{21} **then** $y = b_1$
If x_1 **is** A_{12} **and** x_2 **is** A_{22} **then** $y = b_2$



In Terms of Formulas . . .

Gaussian membership functions:

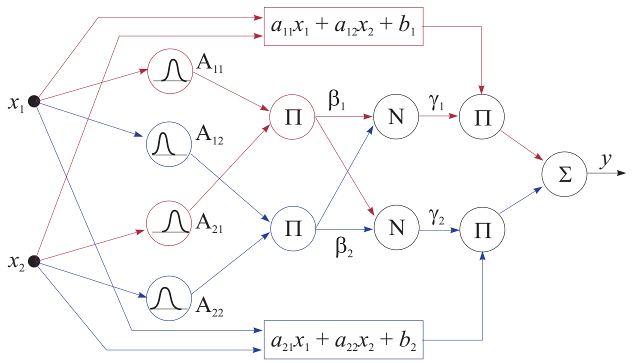
$$\mu_{A_{ij}}(x_j; c_{ij}, \sigma_{ij}) = \exp\left(-\frac{(x_j - c_{ij})^2}{2\sigma_{ij}^2}\right)$$

Input–output equation:

$$y = \frac{\sum_{i=1}^{K} \beta_i(x) b_i}{\sum_{i=1}^{K} \beta_i(x)} = \frac{\sum_{i=1}^{K} \prod_{j=1}^{p} \exp\left(-\frac{(x_j - c_{ij})^2}{2\sigma_{ij}^2}\right) b_i}{\sum_{i=1}^{K} \prod_{j=1}^{p} \exp\left(-\frac{(x_j - c_{ij})^2}{2\sigma_{ij}^2}\right)}$$

Neuro-Fuzzy Network (Affine TS)

If x_1 **is** A_{11} **and** x_2 **is** A_{21} **then** $y_1 = a_{11}x_1 + a_{12}x_2 + b_1$
If x_1 **is** A_{12} **and** x_2 **is** A_{22} **then** $y_2 = a_{21}x_1 + a_{22}x_2 + b_2$



In Terms of Formulas . . .

Gaussian membership functions:

$$\mu_{A_{ij}}(x_j; c_{ij}, \sigma_{ij}) = \exp\left(-\frac{(x_j - c_{ij})^2}{2\sigma_{ij}^2}\right)$$

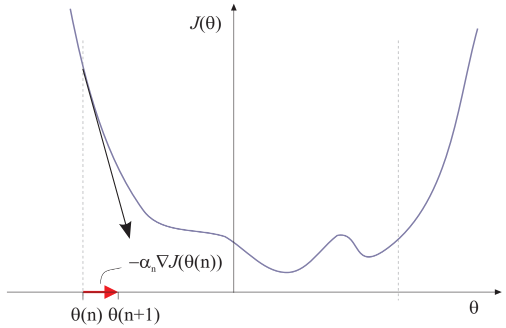
Input–output equation:

$$y = \frac{\sum_{i=1}^{K} \prod_{j=1}^{p} \exp\left(-\frac{(x_j - c_{ij})^2}{2\sigma_{ij}^2}\right)\left(a_i^T x + b_i\right)}{\sum_{i=1}^{K} \prod_{j=1}^{p} \exp\left(-\frac{(x_j - c_{ij})^2}{2\sigma_{ij}^2}\right)}$$

Backpropagation Learning

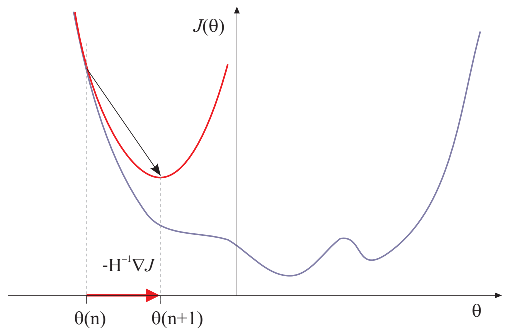
$$\theta(n+1) = \theta(n) - \alpha(n)\nabla J(\theta(n))$$

$$\nabla J(\theta) = \left[\frac{\partial J(\theta)}{\partial \theta_1}, \frac{\partial J(\theta)}{\partial \theta_2}, \ldots, \frac{\partial J(\theta)}{\partial \theta_M}\right]^T$$



Second-Order Gradient Methods

$$J(\theta) \approx J(\theta(n))(\theta - \theta(n)) + \frac{1}{2}(\theta - \theta(n))^T \mathbf{H}(\theta(n))(\theta - \theta(n))$$



faster convergence, higher memory requirements

Randomized Optimization Techniques

- ⋄ Genetic algorithms
- ⋄ Evolutionary computation
- ⋄ Simulated annealing
- ⋄ Ant colony optimization, etc.

Next to the membership functions, also the fuzzy logic operators and rule antecedent variables can be optimized.

Representation of Dynamics

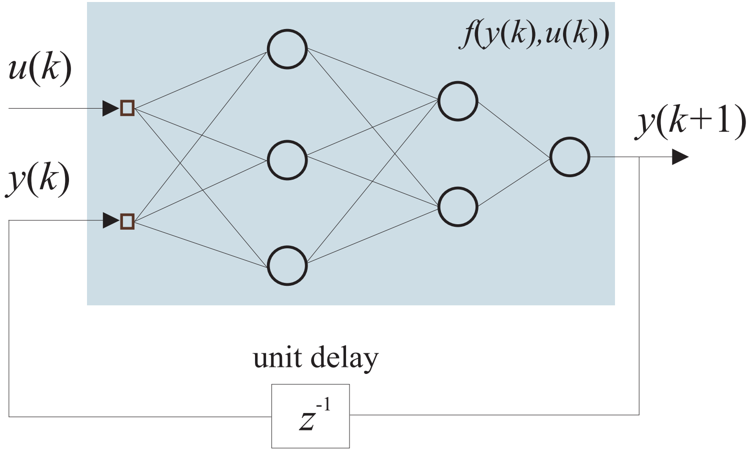
Nonlinear input–output (regression) model:

$$y(k+1) = f\left(y(k), \dots, y(k-n_y+1), u(k), \dots, u(k-n_u+1)\right)$$

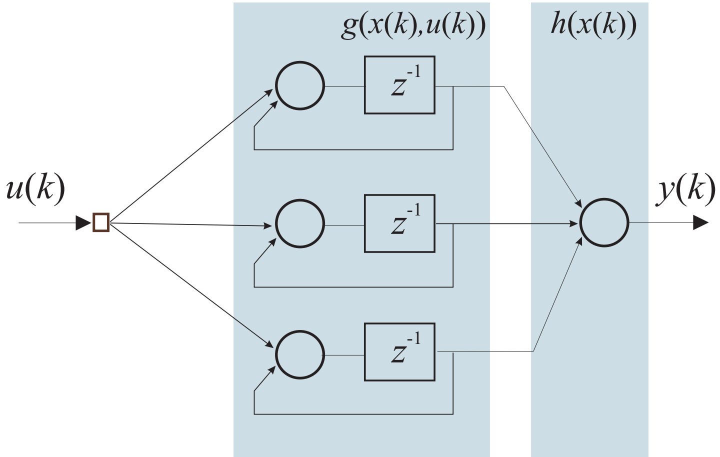
Nonlinear state-space model:

$$\begin{aligned} x(k+1) &= g\left(x(k), u(k)\right) \\ y(k) &= h(x(k)) \end{aligned}$$

Input–Output Neural Network



State–Space Neural Network (Elman)



Dynamic Takagi–Sugeno Model

Input–output form:

If $y(k)$ is A_i **and** $u(k)$ is B_i

$$\textbf{then } y(k+1) = \sum_{j=1}^{n_y} a_{ij} y(k-j+1) + \sum_{j=1}^{n_u} b_{ij} u(k-j+1) + c_i$$

State-space form:

If $x(k)$ is A_i **and** $u(k)$ is B_i

$$\textbf{then } \begin{cases} x_i(k+1) = \mathbf{\Phi}_i x(k) + \mathbf{\Gamma}_i u(k) + a_i \\ y_i(k) = \boldsymbol{C}_i x(k) + \boldsymbol{D}_i u(k) + c_i \end{cases}$$

Dynamic TS Model

$$y = \sum_{i=1}^{K} w_i(z) \mathcal{M}_i(u) \quad \text{with} \quad \sum_{i=1}^{K} w_i(z) = 1 \quad \forall z$$

$\mathcal{M}_i$: local affine model (input–output or state-space)

$$\begin{aligned} y_i(k+1) &= \mathcal{A}_i(q^{-1}) y(k) + \mathcal{B}_i(q^{-1}) u(k) + c_i \\ &= A_i x(k) + B_i u(k) + c_i \end{aligned}$$

$w_i(\cdot)$: weighting function for operating region i

$z(k)$: scheduling variable

Degrees of Freedom

Local models:

- shared vs. local states
- same or different structure (linearity, order, delays)
- represent nominal or failure modes

Weighting (membership) functions:

- discrete vs. continuous
- fixed vs. adaptive

Scheduling variables $z \in Z$:

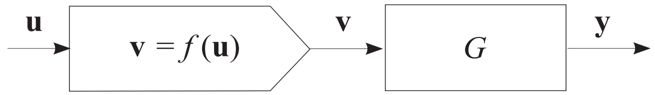
- $Z \supseteq X \times U, \qquad Z \not\subset X \times U, \qquad Z \subset U, \qquad Z \subset Y, \qquad Z \subset X$

Structured Nonlinear Models: Special Case

- $z(k) = u(k)$, $\mathcal{A}_i(q^{-1}) = \mathcal{A}_j(q^{-1}), \quad \forall i, j$:

$$y(k+1) = \mathcal{A}(q^{-1}) y(k) + \sum_{i=1}^{K} w_i(u(k)) [\mathcal{B}_i(q^{-1}) u(k) + c_i]$$

Hammerstein system



Structured Nonlinear Models: Special Case

- $z_i(k) \neq u(k), \quad \forall i$:

$$y(k+1) = \sum_{i=1}^{K} w_i(y(k), \ldots, u(k-1), \ldots)[\mathcal{A}_i(q^{-1})y(k) + \mathcal{B}_i(q^{-1})u(k) + c_i]$$

input-affine system

$$y(k+1) = g(\mathbf{x}(k)) + h(\mathbf{x}(k)) \cdot u(k)$$

Advantages – Application's Viewpoint

Flexible tool for approximating nonlinear processes

- both abrupt and smooth nonlinearities
- structural choices
- systems with variable structure (mode changes, failures)

Related Techniques / Terminology

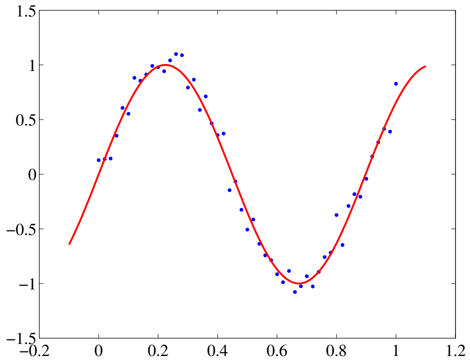
- gain-scheduling (control systems; Shamma)
- composite linear models (identification; Ljung)
- phase regression (nonlinear regression; Seber and Wild)
- expert mixtures (statistical learning; Jordan)

Constructing Neuro-Fuzzy Systems

– Expert knowledge $\longrightarrow$ if–then rules (initial model). Fine-tune parameters by using data.

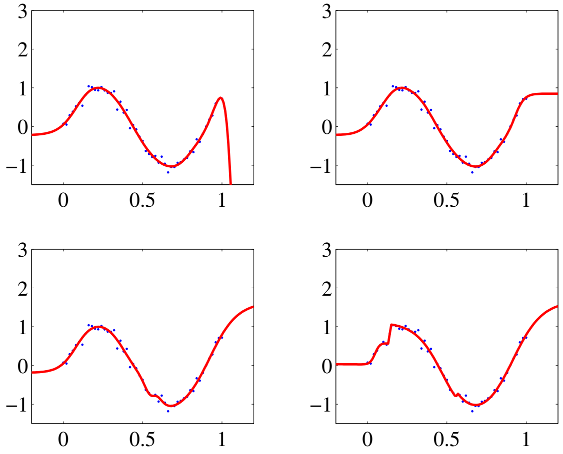
- ◇ robust initialization
- ◇ improved generalization

Example

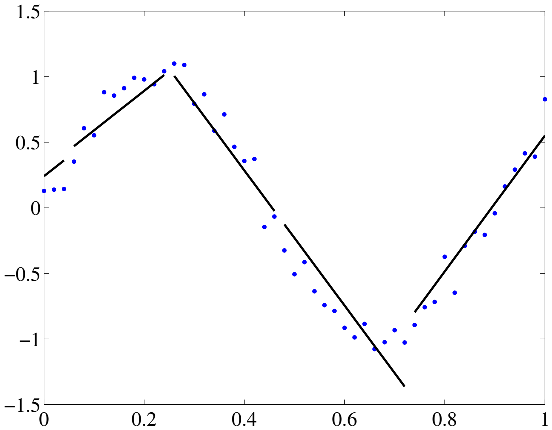


- noisy data
- domain not entirely covered

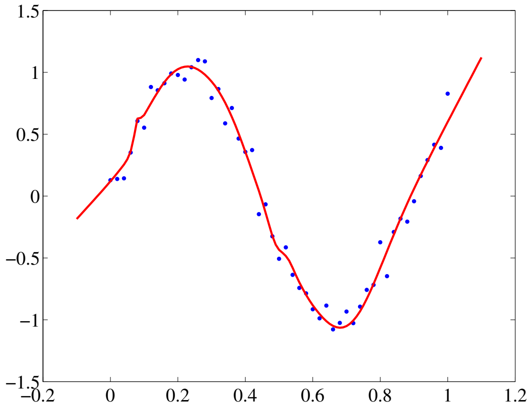
Feedforward Neural Network (5 Neurons)



Initial Neuro–Fuzzy TS Model (5 rules)



Final Neuro–Fuzzy TS Model



Remark: there are other ways to improve initialization/generalization

Constructing Neuro-Fuzzy Systems

– Expert knowledge ⟶ if–then rules (initial model). Fine-tune parameters by using data.

◇ robust initialization

◇ improved generalization

– Fuzzy rules constructed from scratch from data. Expert validation, modification.

◇ model inspection

◇ iterative experiment design

Note: fine-tuning parameters improves performance but may reduce the transparency/interpretability of the model!

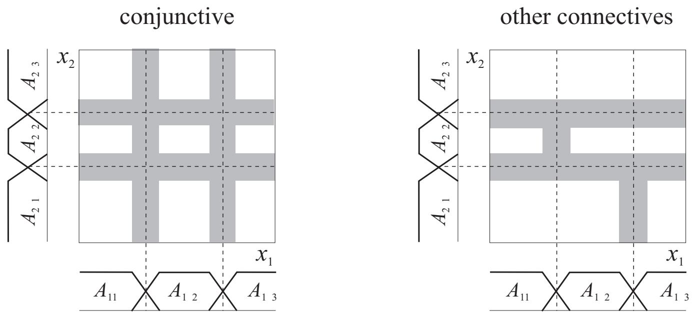
Construction of Membership Functions

◇ templates (grid partitioning)

◇ discrete search, tree-construction

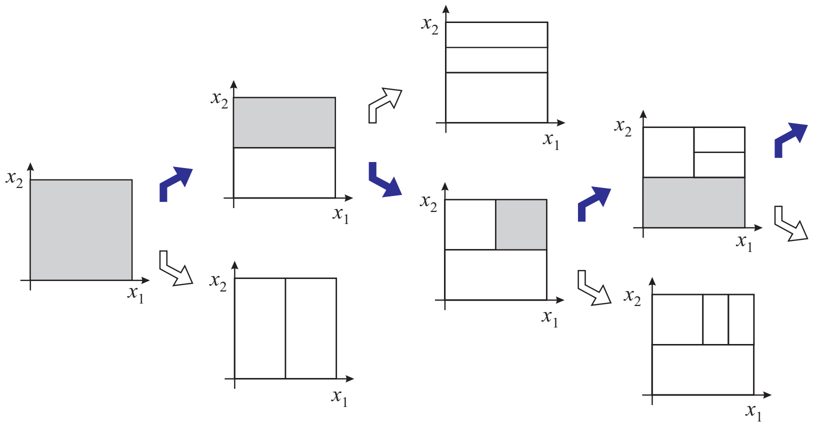
◇ fuzzy clustering

Grid Partitioning

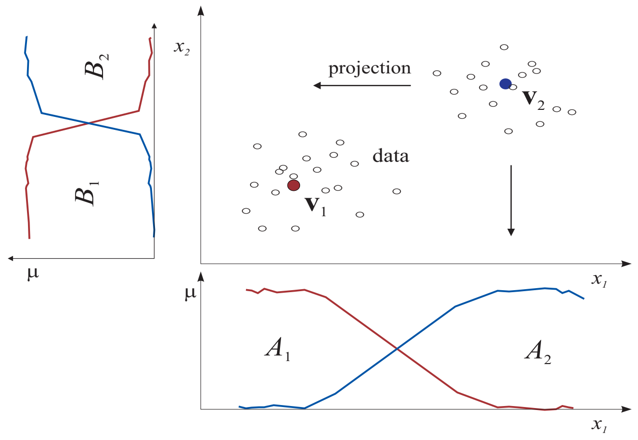


- Good for small problems (1 or 2 inputs).
- Transparency "guaranteed".

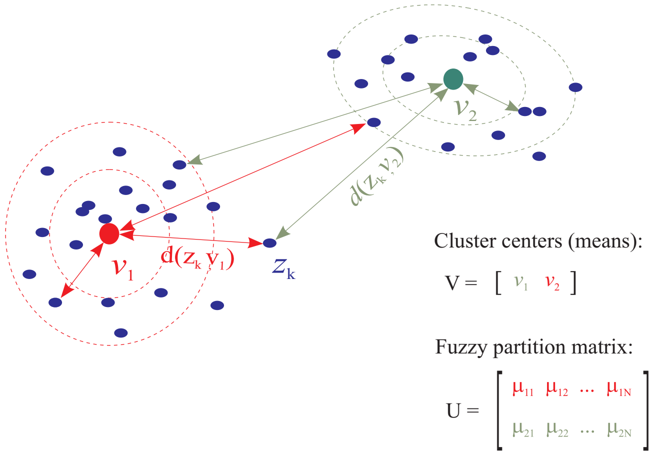
Tree-Construction Methods (Lolimot)



Extraction of Rules by Fuzzy Clustering



Fuzzy Clustering



Fuzzy Clustering Problem

Given the data:

$$\mathbf{z_k} = [z_{1k}, z_{2k}, \dots, z_{nk}]^T \in \mathbb{R}^n, \quad k = 1, \dots, N$$

Find:

the fuzzy partition matrix:

$$\mathbf{U} = \begin{bmatrix} \mu_{11} & \dots & \mu_{1k} & \dots & \mu_{1N} \\ \vdots & \dots & \vdots & \dots & \vdots \\ \mu_{c1} & \dots & \mu_{ck} & \dots & \mu_{cN} \end{bmatrix}$$

and the cluster centers:

$$\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_c\}, \quad \mathbf{v}_i \in \mathbb{R}^n$$

Fuzzy Clustering: an Optimization Approach

Objective function (least-squares criterion):

$$J(Z; V, U, A) = \sum_{i=1}^{c} \sum_{j=1}^{N} \mu_{i,j}^m d_{A_i}^2(\mathbf{z}_j, \mathbf{v}_i)$$

subject to constraints:

$0 \le \mu_{i,j} \le 1,$	$i = 1, \dots, c, \ j = 1, \dots, N$	membership degree
$0 < \sum_{j=1}^{N} \mu_{i,j} < 1,$	$i = 1, \dots, c$	no cluster empty
$\sum_{i=1}^{c} \mu_{i,j} = 1,$	$j = 1, \dots, N$	total membership

Fuzzy c-Means Algorithm

Repeat:

1. Compute cluster prototypes (means): $v_i = \frac{\sum_{k=1}^{N} \mu_{i,k}^m \mathbf{z}_k}{\sum_{k=1}^{N} \mu_{i,k}^m}$

2. Calculate distances: $d_{ik} = (\mathbf{z}_k - \mathbf{v}_i)^T (\mathbf{z}_k - \mathbf{v}_i)$

3. Update partition matrix: $\mu_{ik} = \frac{1}{\sum_{j=1}^{c} (d_{ik}/d_{jk})^{1/(m-1)}}$

until $\|\Delta \mathbf{U}\| < \epsilon$

Distance Measures

- Euclidean norm:

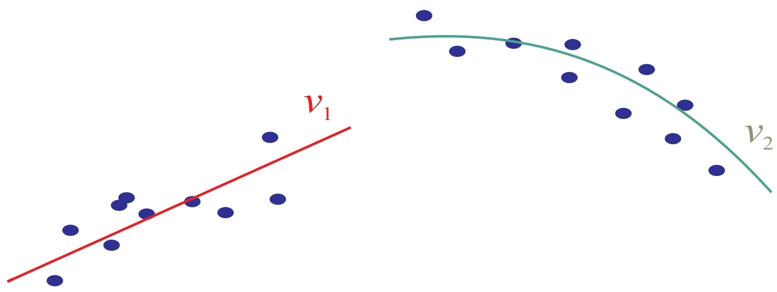
$$d^2(\mathbf{z}_j, \mathbf{v}_i) = (\mathbf{z}_j - \mathbf{v}_i)^T (\mathbf{z}_j - \mathbf{v}_i)$$

- Inner-product norm:

$$d^2_{A_i}(\mathbf{z}_j, \mathbf{v}_i) = (\mathbf{z}_j - \mathbf{v}_i)^T A_i (\mathbf{z}_j - \mathbf{v}_i)$$

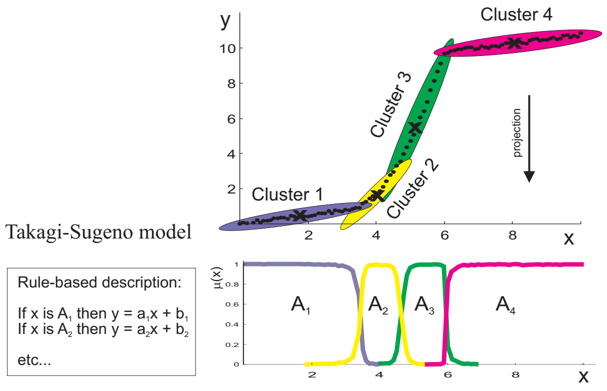
- Many other possibilities ...

Generalized Prototypes

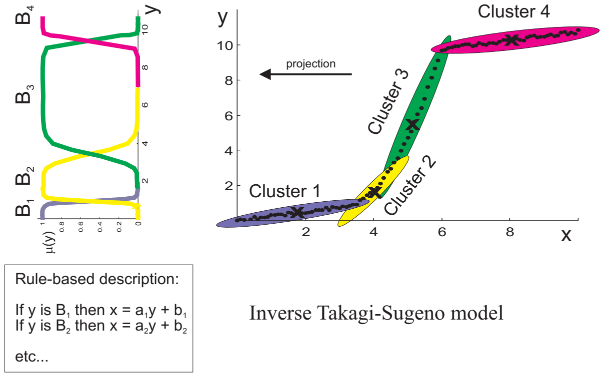


lines, circles, ellipses, functions, etc.

Extraction of Rules by Fuzzy Clustering



Extraction of Rules by Fuzzy Clustering



Inverse Takagi-Sugeno model

Estimation of Consequent Parameters

$$\hat{y}(k+1) = \sum_{i=1}^{K} w_i(z(k))\left(A_i x(k) + B_i u(k) + c_i\right) = \sum_{i=1}^{K} w_i(z(k))\theta_i^T \varphi(k)$$

- **Global LS criterion:**

$$[\theta_1 \ldots \theta_K]^T = \arg\min \left(\mathbf{y} - \sum_{i=1}^{K} \boldsymbol{w}_i(\mathbf{z})\theta_i^T \boldsymbol{\phi}\right)^2 = \left(\boldsymbol{\phi}^T\boldsymbol{\phi}\right)^{-1}\boldsymbol{\phi}^T\mathbf{y}$$

with $\boldsymbol{\phi} = [\boldsymbol{w}_1(\mathbf{z})\boldsymbol{\varphi} \quad \ldots \quad \boldsymbol{w}_K(\mathbf{z})\boldsymbol{\varphi}]$

- **Weighted local LS criterion:**

$$\theta_i = \arg\min \left(\boldsymbol{w}_i(\mathbf{z})\left(\mathbf{y} - \theta_i^T\boldsymbol{\varphi}\right)^2\right) = \left(\boldsymbol{\varphi}^T\boldsymbol{w}_i(\mathbf{z})\boldsymbol{\varphi}\right)^{-1}\boldsymbol{\varphi}^T\boldsymbol{w}_i(\mathbf{z})\mathbf{y}$$

Example: Local vers. Global Estimation

$$y = f_s(u) = 3u^2 - 5u + 6$$

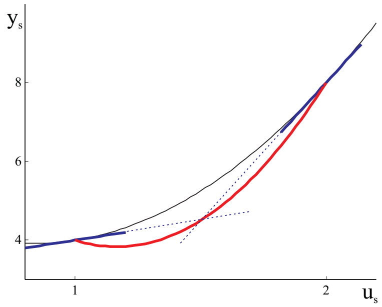
choose points t_1 and t_2, use triangular membership functions:

$$\mu_{A_1}(u) = \frac{u - t_1}{t_2 - t_1}, \quad \mu_{A_2} = 1 - \mu_{A_1}$$

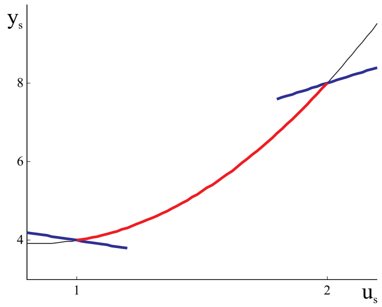
TS model (2 rules) gives a 2nd-order polynomial:

$$y_s = \frac{a_1 - a_2}{t_2 - t_1}u^2 + \frac{t_2 a_2 - t_1 a_1 + b_1 - b_2}{t_2 - t_1}u + \frac{t_2 b_2 - t_1 b_1}{t_2 - t_1}$$

Local Validity vs. Global Fit



locally valid



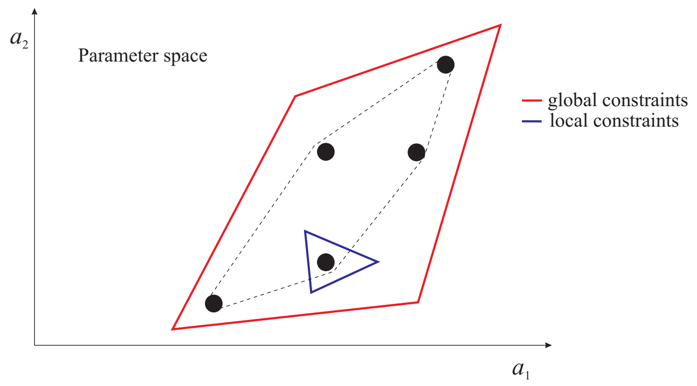
good global fit

Possible Solutions

- use different models for different purposes
- constraints on local model behavior
- multi-objective optimization - penalty on local behavior
- other interpolation methods

Constrained Estimation

$$A(x) = \sum_{i=1}^{K} w_i(x) A_i, \quad \sum_{i=1}^{K} w_i(x) = 1 \quad \forall z$$



Example – Constraints on Model Gain

$K_{\min}$ and $K_{\max}$ . . . minimal and maximal process gain

$$K_{\min} \leq \frac{\sum_{i=1}^{n_b} b_i}{1 + \sum_{i=1}^{n_a} a_i} \leq K_{\max}$$

this gives:

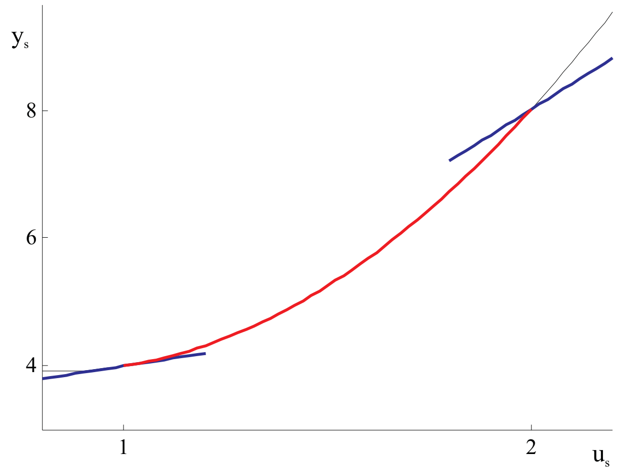
$$K_{\min}\left(1 + \sum_{i=1}^{n_a} a_i\right) - \sum_{i=1}^{n_b} b_i \leq 0$$

$$-K_{\max}\left(1 + \sum_{i=1}^{n_a} a_i\right) + \sum_{i=1}^{n_b} b_i \leq 0$$

Advantages of Constraints

- Imposing a certain structure (e.g., Hammerstein model)
- Use of *a priori* knowledge:
  - poles of sampled systems
  - stability
  - bounds on settling time
  - bounds on gain, nature of nonlinearity
- Improve model in case of poor experimental data

Result of Constrained Estimation



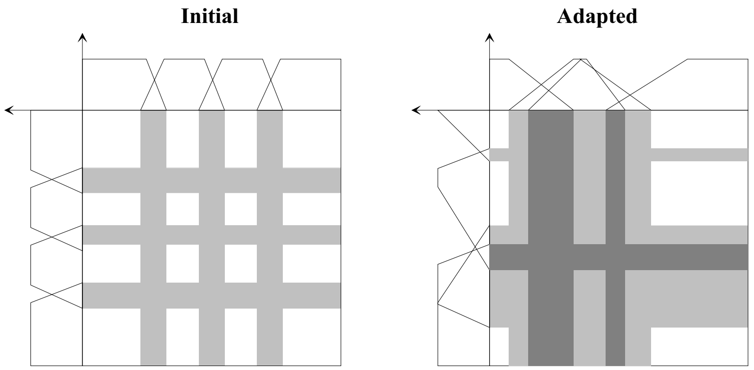
Multi-Objective Identification

$$\hat{\theta} = \arg\ \min \left(V(\theta) + \sum_{i=1}^{K} \beta_i V_i(\theta_i) \right)$$

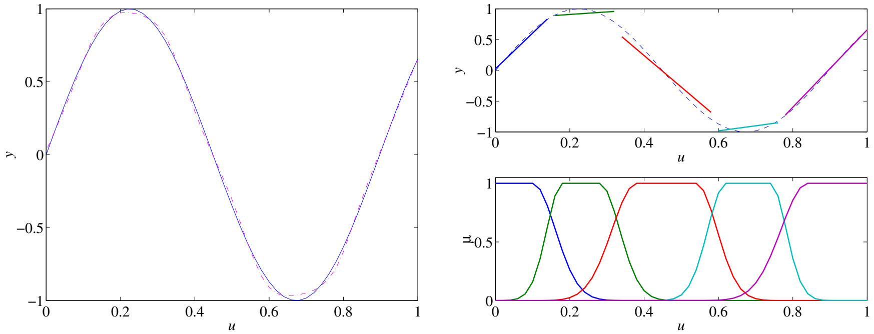
Karush-Kuhn-Tucker conditions:

$$\frac{\partial V}{\partial \theta_i}(\hat{\theta}(\beta)) + \beta_i \frac{\partial V_i}{\partial \theta_i}(\hat{\theta}(\beta)) = 0$$

Hard to Interpret Membership Functions

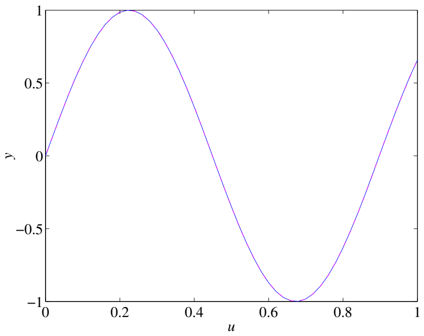


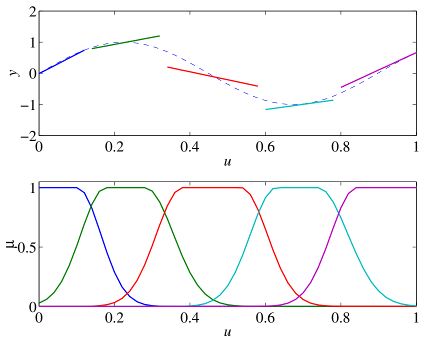
Static Example: Initial Model



obtained by fuzzy clustering + least squares, RMS = 0.0258

Static Example: Fine-Tuned Model





trained by the ANFIS function of Matlab, RMS = 0.0011

Summary

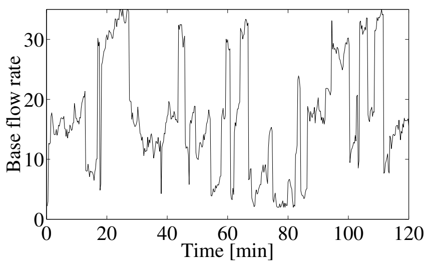
	RMS	$\|\theta - \theta_0\|$
initial	0.0258	1.81
final	0.0011	5.30

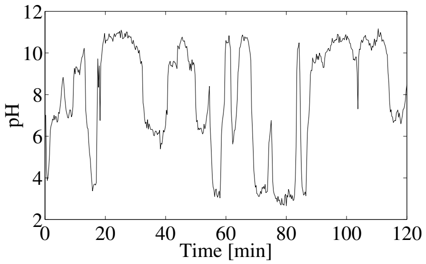
Some of the rules:

Initial: If u is A_3 then $y = -5.302u + 2.380$

Final: If u is A_3' then $y = -3.206u + 1.405$

Dynamic Example: pH Neutralization

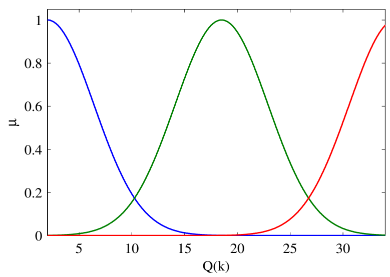


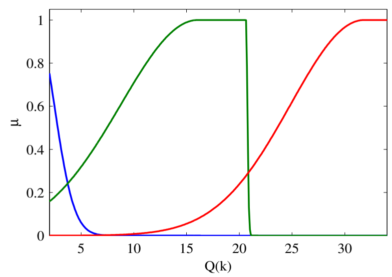


simulated process, strongly nonlinear

neuro-fuzzy model structure: $\mathrm{pH}(k+1) = f(\mathrm{pH}(k), Q(k))$

Rules and Membership Functions



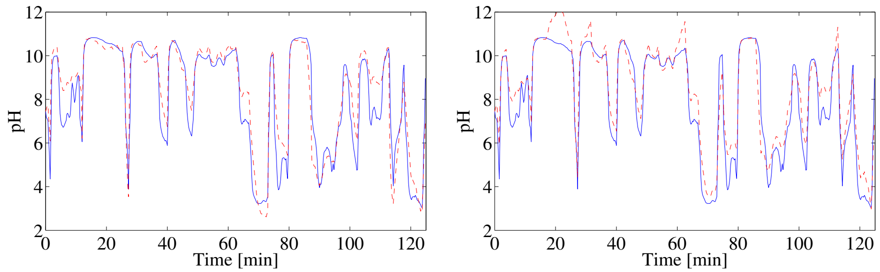


Some of the rules:

If $Q(k)$ is *Low* then $\mathrm{pH}(k+1) = 0.83\mathrm{pH}(k) + 0.09Q(k) + 0.03$

If $Q(k)$ is *Low'* then $\mathrm{pH}(k+1) = 0.37\mathrm{pH}(k) - 0.05Q(k) + 2.14$

Validation



	before training	after training
training data set	0.90	0.82
validation data set	0.81	0.89

Various Real-World Applications

- Turbidity prediction in a rapid sand filter
  - outperforms linear and polynomial models
  - adaptive variant (recursive parameter estimation)
- Estimation of respiratory parameters
  - detailed information about the form of nonlinearity
  - diagnosis → ventilator settings, therapy
- Engine load prediction for a combine harvester
  - multi-objective optimization
  - transparent results obtained from data

Combine Harvester Engine Load Prediction

Model Variables

Signal	Symbol	Units
Engine Load	$y(k)$	%
Feedrate	$u_F(k-4)$	Volt
Slope	$u_S(k-1)$	degrees
Handle Position	$u_H(k-1)$	%

Inputs selected by means of genetic algorithm search.

Fuzzy Model Structure

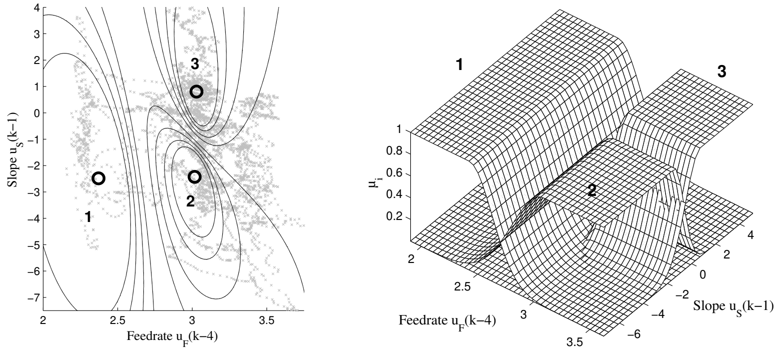
$$\textbf{IF}\ \big(u_F(k-4) \in A_{F,i}\big)\ \textbf{AND}\ \big(u_S(k-1) \in A_{S,i}\big)$$

$$\textbf{THEN}$$

$$y(k) = a_{1,i}y(k-1) + a_{2,i}y(k-2) +$$

$$b_{F,i}u_F(k-4) + b_{S,i}u_S(k-1) + b_{H,i}u_H(k-1) + d_i$$

Antecedent Partition (Fuzzy Clustering)



Consequent Parameters (Global vs. Balanced)

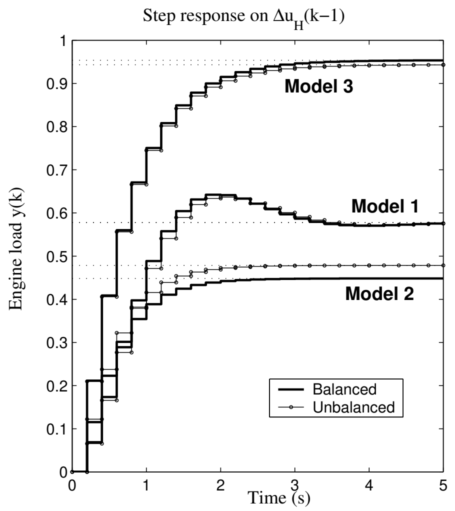
i	$a_{1,i}$	$a_{2,i}$	$b_{F,i}$	$b_{S,i}$	$b_{H,i}$	d_i
1	1.57	−0.686	2.86	−0.011	0.070	−5.05
2	0.93	−0.203	8.50	0.853	0.128	−15.3
3	0.93	−0.164	8.18	0.946	0.237	−23.7

i	$a_{1,i}$	$a_{2,i}$	$b_{F,i}$	$b_{S,i}$	$b_{H,i}$	d_i
1	1.51	−0.625	2.07	0.109	0.065	−2.84
2	0.94	−0.186	7.36	0.676	0.125	−13.5
3	0.92	−0.139	7.79	0.814	0.264	−24.7

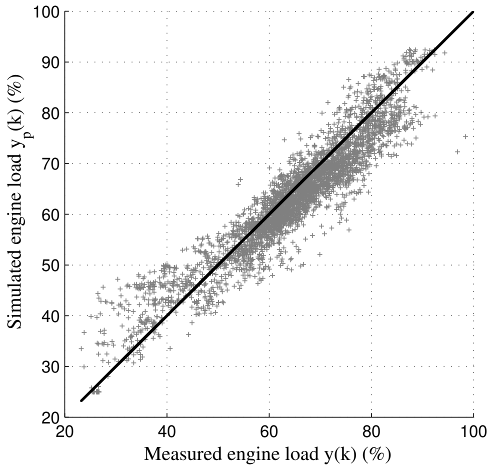
Interpretation of Local Models – Gains

i	K_F^{stat}	K_S^{stat}	K_H^{stat}
1	17.54 %/V	0.50 %/deg	0.57
2	29.89 %/V	2.87 %/deg	0.44
3	33.82 %/V	3.52 %/deg	0.94

Interpretation of Local Models – Dynamics



Performance on Validation Data



Concluding Remarks

- Neuro-fuzzy modeling – powerful, versatile framework
  - flexible choice of model structure ("user friendly")
  - straightforward parameter estimation (consequents)
  - integration *a priori* knowledge and data

- Problems, limitations
  - large, over-parameterized models
  - care must be taken with local interpretation
  - convex sum – compromise between local and global fit
  - use of constraints and regularization help
  - be self-critical