Adaptive Neural Fuzzy Inference Systems (ANFIS) MODULE 11

Dr. R.B.Ghongade,
VIIT, Pune-411048

Agenda

- Objective
- Fuzzy Control
  - Background, Technology & Typology
- ANFIS:
  - as a Type III Fuzzy Control
  - Characteristics
  - Pros and Cons
  - Opportunities
  - Applications

ANFIS Objective

- To integrate the best features of Fuzzy Systems and Neural Networks:
  - From FS: Representation of prior knowledge into a set of constraints (network topology) to reduce the optimization search space
  - From NN: Adaptation of backpropagation to structured network to automate FC parametric tuning
- ANFIS application to synthesize:
  - controllers (automated FC tuning)
  - models (to explain past data and predict future behavior)

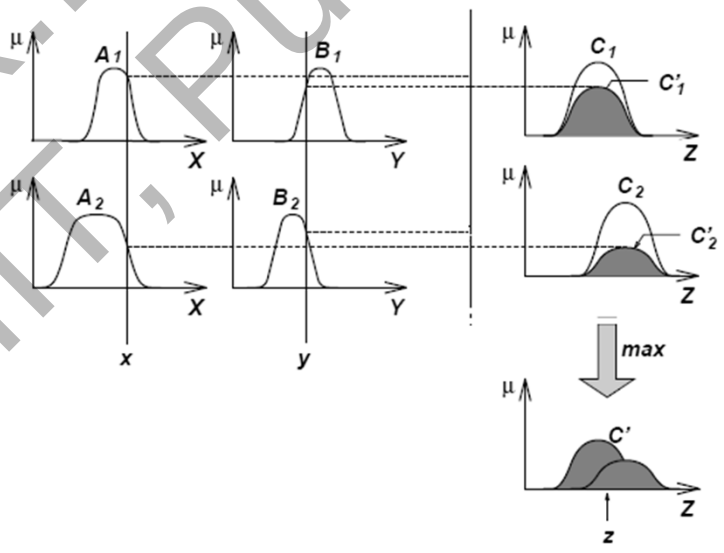
FC Technology & Typology

- Fuzzy Control
  - A high level representation language with local semantics and an interpreter/compiler to synthesize non-linear (control) surfaces
  - A Universal Functional Approximator
- FC Types
  - Type I: RHS is a monotonic function
  - Type II: RHS is a fuzzy set
  - Type III: RHS is a (linear) function of state

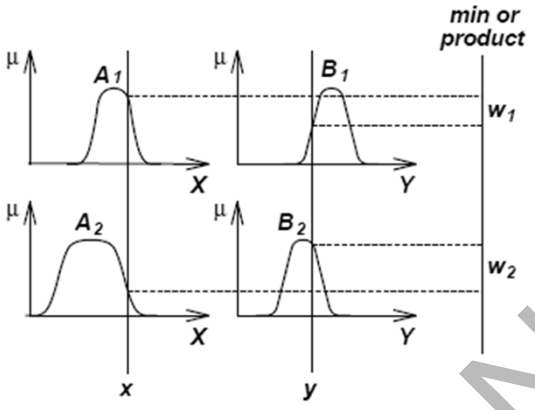
FC Technology (Background)

- Fuzzy KB representation
  - Scaling factors, Termsets, Rules
- Rule inference (generalized *modus ponens*)
- Development & Deployment
  - Interpreters, Tuners, Compilers, Run-time
  - Synthesis of control surface
- FC Types I, II, III

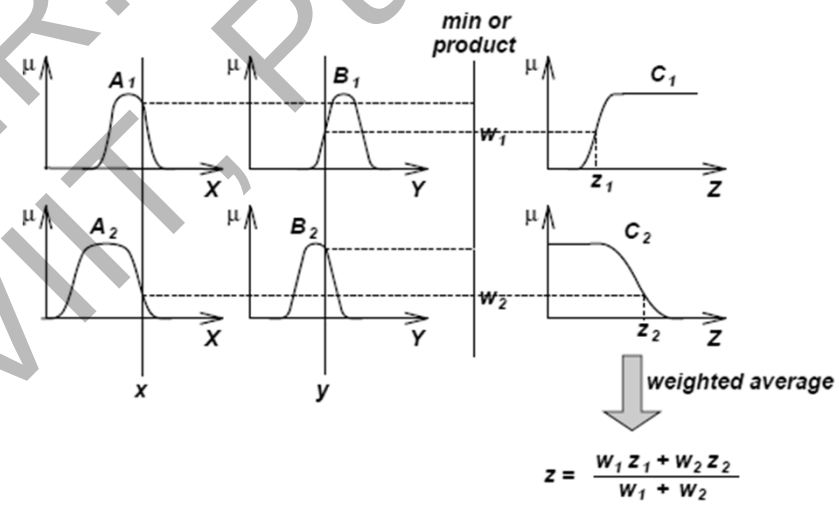
The Mamdani fuzzy inference system using min and max for fuzzy AND and OR operators respectively



The Sugeno fuzzy model



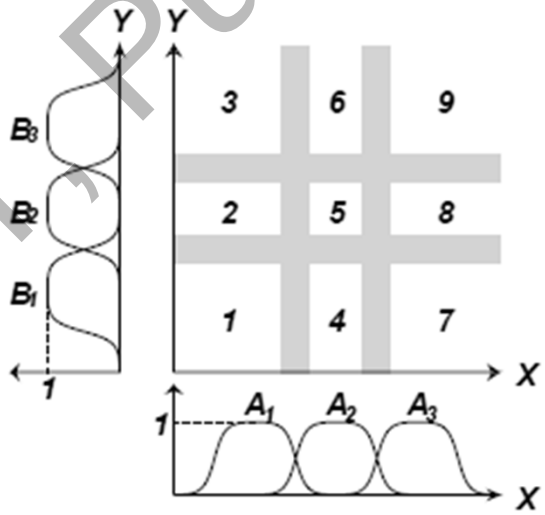
The Tsukamoto fuzzy model



Partition Styles for Fuzzy Models

- Grid partition
  - This partition method is often chosen in designing a fuzzy controller, which usually involves only several state variables as the inputs to the controller
  - This partition strategy needs only a small number of MFs for each input
  - However, it encounters problems when we have a moderately large number of inputs

Grid partition



Tree partition

- Each region can be uniquely specified along a corresponding decision tree
- The tree partition relieves the problem of an exponential increase in the number of rules
- However more MFs for each input are needed to define these fuzzy regions, and these MFs do not usually bear clear linguistic meanings such as “small”, “big” and so on

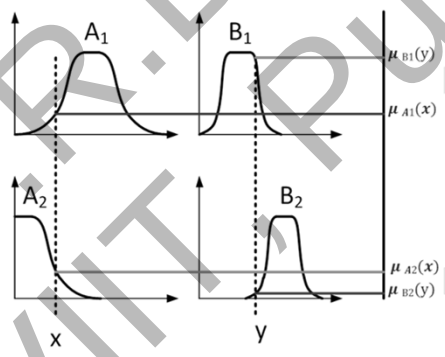
Scatter partition

- By covering a subset of the whole input space that characterizes a region of possible occurrence of the input vectors, the scatter partition can also limit the number of rules to a reasonable amount

FC of Type II, III, and ANFIS

- Type II Fuzzy Control must be tuned manually
- Type III Fuzzy Control (Takagi-Sugeno type)have an automatic Right Hand Side (RHS) tuning
- ANFIS will provide both:
  - RHS tuning, by implementing the TSK controller as a network
  - and LHS tuning, by using back-propagation

(a)A two input 1st order Sugeno fuzzy model with two rules



$w_1=\mu_{A1}(x)\mu_{B1}(y)$

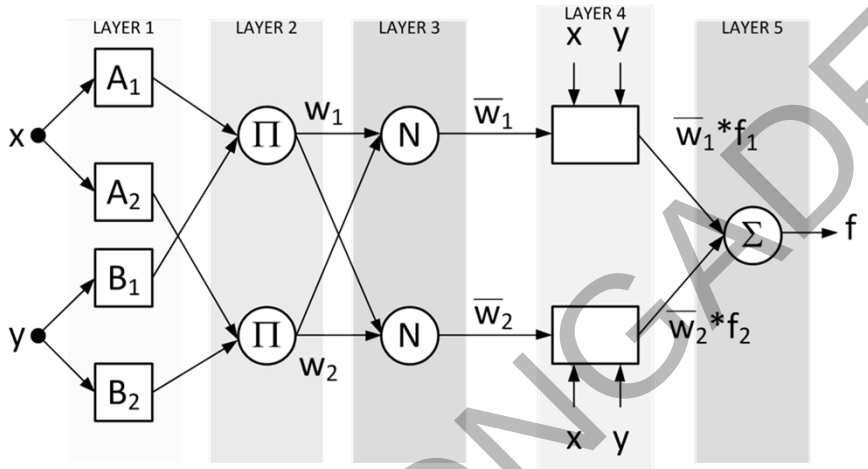
$\bar{w}_1 = \frac{w_1}{w_1+w_2}$ $\quad f_1=p_1x+q_1y+r_1$

$f=\bar{w}_1 f_1+\bar{w}_2 f_2$

$\bar{w}_2 = \frac{w_2}{w_1+w_2}$ $\quad f_2=p_2x+q_2y+r_2$

$w_2=\mu_{A2}(x)\mu_{B2}(y)$

(b)Equivalent ANFIS Network



ANFIS Neurons: Clarification note

- Note that neurons in ANFIS have different structures:
  - Values [Membership function defined by parameterized soft trapezoids (Generalized Bell Functions)]
  - Rules [Differentiable T-norm - usually product]
  - Normalization [Sum and arithmetic division]
  - Functions [Linear regressions and multiplication with $\overline{w_i}$, i.e., normalized weights w_i,]
  - Output [Algebraic Sum]

Typical rule set with two fuzzy if-then rules

- IF $(x\ is\ A_1) AND\ (y\ is\ B_1)\ THEN\ \ f_1 = p_1x + q_1y + r_1$
- IF $(x\ is\ A_2) AND\ (y\ is\ B_2)\ THEN\ \ f_2 = p_2x + q_2y + r_2$
- We denote the output node i in layer l as $O_{l,i}$
- **Layer 1**: Every node i in this layer is an adaptive node with a node output defined by:

$$\boldsymbol{O_{1,i} = \mu_{A_i}(x), \ldots for\ i = 1, 2}$$

or

$$\boldsymbol{O_{1,i} = \mu_{B_{i\_2}}(y), \ldots for\ i = 3, 4}$$

  - where x $(or\ y)$ is the input to the node and A_i or B_{i_2} is a fuzzy set associated with this node
  - In other words, outputs of this layer are the membership values of the premise part
  - A_i can be characterized by the generalized bell function:

$$\mu_A(x_1) = \frac{1}{1 + \left|\frac{x_1 - c_i}{a_i}\right|^{2b_i}}$$

  - where (a_i, b_i, c_i) is the parameter set
  - Parameters in this layer are referred to as *premise parameters*

Layer 1

- We denote the output node i in layer l as $O_{l,i}$
- **Layer 1**: Every node i in this layer is an adaptive node with a node output defined by:

$$\boldsymbol{O_{1,i} = \mu_{A_i}(x), \ldots for\ i = 1, 2}$$

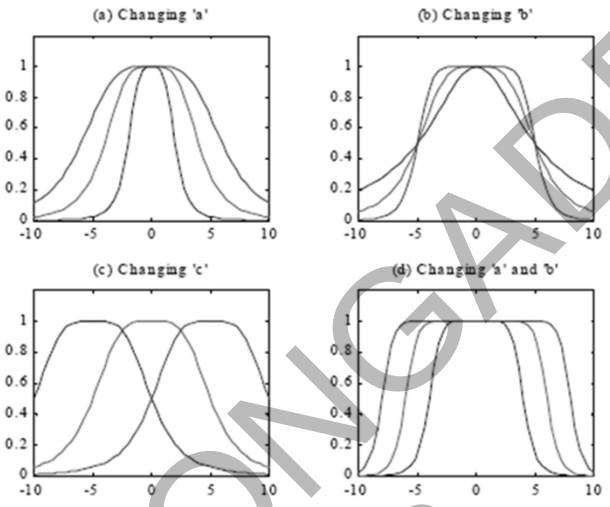
or

$$\boldsymbol{O_{1,i} = \mu_{B_{i\_2}}(y), \ldots for\ i = 3, 4}$$

  - where x $(or\ y)$ is the input to the node and A_i or B_{i_2} is a fuzzy set associated with this node
  - In other words, outputs of this layer are the membership values of the premise part
  - Where A is a linguistic label (small, large, …)
  - A_i can be characterized by the generalized bell function:
  - where (a_i, b_i, c_i) is the parameter set
  - Parameters in this layer are referred to as *premise parameters*
  - Node output : ***membership value of input***

Effect of changing Parameters {a,b,c}

$$\mu_A(x_1) = \frac{1}{1 + \left|\frac{x_1 - c_i}{a_i}\right|^{2b_i}}$$



Layer 2: Firing Strength of Rule

- Every node in this layer is a fixed node labeled Π which multiplies the incoming signals and outputs the product:

$$O_{2,i} = w_i = \mu_{A_i}(x) \times \mu_{B_i}(y) \ldots for\ i = 1, 2$$

  - Each node output represents the *firing strength* of a rule
  - Any other T-norm operators that perform fuzzy AND can be used as the node function in this layer
  - Node output: ***firing strength of rule***

Layer 3: Normalize Firing Strength

- Every node in this layer is a fixed node labeled N
  - The i-th node calculates the ratio of the i-th rule's firing strength to the sum of all rules firing strengths

$$O_{3,i} = \overline{w_i} = \frac{w_i}{w_1 + w_2} \dots for\ i = 1, 2$$

  - This layer is called as normalized firing strengths
  - Node output: ***Normalized firing strengths***

Layer 4: Consequent Parameters

- Every node i in this layer is an adaptive node with a node function

$$O_{4,i} = \overline{w_i} \cdot f_1 = \overline{w_i}\,(p_i x + q_i y + r_i)$$

  - where $\overline{w_i}$ is the output of layer 3 and (p_i, q_i, r_i) is the parameter set referred to as *consequent parameters*
  - Node output: ***Evaluation of Right Hand Side Polynomials***

Layer 5: Overall Output

- The single node in this layer is a fixed node labeled Σ which computes the overall output as the summation of all incoming signals

$$O_{5,i} = \textit{overall output} = \sum \overline{w_i} \cdot f_i = \sum_i \frac{w_i \cdot f_i}{w_i}$$

  - Output is linear in consequent parameters *p,q,r*:

$$O_{5,i} = \frac{w_1 \cdot f_1}{w_1 + w_2} + \frac{w_2 \cdot f_2}{w_1 + w_2}$$

$$= \overline{w_1}\,(p_1 x + q_1 y + r_1) + \overline{w_2}\,(p_2 x + q_2 y + r_2)$$

$$= (\overline{w_1} x)p_1 + (\overline{w_1} y)q_1 + (\overline{w_1} r_1) + (\overline{w_2} x)p_2 + (\overline{w_2} y)q_2 + (\overline{w_2} r_2)$$

  - Node output: ***Weighted Evaluation of RHS Polynomials***

ANFIS Computational Complexity

Layer #	L-Type	# Nodes	# Param
L_1	Values	(p•n)	3•(p•n)=|S1|
L_2	Rules	p^n	0
L_3	Normalize	p^n	0
L_4	Lin. Funct.	p^n	(n+1)•p^n=|S2|
L_5	Sum	1	0

- n=number of inputs
- p=number of membership functions

ANFIS Parametric Representation

- ANFIS uses two sets of parameters: S1 and S2
  - S1 represents the fuzzy partitions used in the rules LHS

$$S1 = \{\{a_{11}, b_{11}, c_{11}\}, \{a_{12}, b_{12}, c_{12}\}, \ldots, \{a_{1p}, b_{1p}, c_{1p}\}, \ldots, \{a_{np}, b_{np}, c_{np}\}\}$$

  - S2 represents the coefficients of the linear functions in the rules RHS

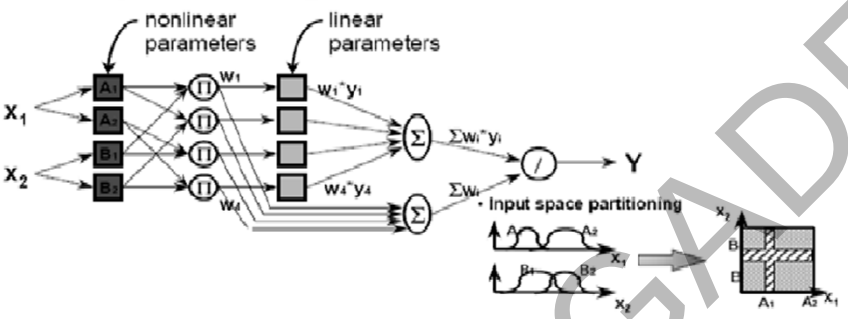
$$S2 = \{\{c_{10}, c_{11}, \ldots, c_{1n}\}, \ldots, \{c_{p^n 0}, c_{p^n 1}, \ldots, c_{p^n n}\}\}$$

ANFIS Learning Algorithms

- ANFIS uses a two-pass learning cycle
  - Forward pass:
    - S1 is fixed and S2 is computed using a Least Squared Error (LSE) algorithm (Off-line Learning)
  - Backward pass:
    - S2 is fixed and S1 is computed using a gradient descent algorithm (usually Back-propagation)

Structure ID & Parameter ID

Hybrid training method



	Forward Pass	Backward Pass
Premise Parameters	Fixed	Gradient Descent
Consequent Parameters	Least-Squares Estimate	Fixed
Signals	Node Outputs	Error Signals

Solving for consequent parameters

- For given values of S1, using K training data, we can transform the above equation into $\boldsymbol{B} = \boldsymbol{AX}$, where X contains the elements of S2
- This is solved by: $(\boldsymbol{A^T A})^{-1}\boldsymbol{A^T}\,\boldsymbol{B} = \boldsymbol{X}^*$ where $(\boldsymbol{A^T A})^{-1}\boldsymbol{A^T}$ is the pseudo-inverse of A (if $\boldsymbol{A^T A}$ is nonsingular)

$$\begin{bmatrix} \overline{w_1}x_1 & \overline{w_1}y_1 & \overline{w_1} & \overline{w_2}x_1 & \overline{w_2}y_1 & \overline{w_2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \overline{w_1}x_K & \overline{w_1}y_K & \overline{w_1} & \overline{w_2}x_K & \overline{w_2}y_K & \overline{w_2} \end{bmatrix} \begin{bmatrix} p_1 \\ q_1 \\ r_1 \\ p_2 \\ q_2 \\ r_2 \end{bmatrix} = \begin{bmatrix} f_{[1]} \\ \vdots \\ f_{[K]} \end{bmatrix}$$

$$AX = B$$
$$X = A^{-1}B$$

- Since A is never a square matrix direct inverse is cannot be computed hence we use pseudo-inverse

$$X = (A^T A)^{-1} A^T B$$

ANFIS Least Squares (LSE) Batch Algorithm

- LSE used in Forward Stroke:
  - Parameter Set: $S = \{S1 \cup S2\}, and \{S1 \cap S2 = \varnothing\}$

    $Output = F(\bar{I}, S), where\ I\ is\ the\ input\ vector$

    $H(Output) = H \circ F(\bar{I}, S), where H \circ F\ is\ linear\ in\ S2$
- The LSE minimizes the error $||AX - B||2$ by approximating X with X*

ANFIS LSE Batch Algorithm (cont.)

- Rather than solving directly $(\boldsymbol{A^T A})^{-1} \boldsymbol{A^T B} = \boldsymbol{X}^*$, we resolve it iteratively (from numerical methods):

$$\left.\begin{aligned} S_{i+1} &= S_i - \frac{S_i a_{(i+1)} a_{(i+1)}^T S_i}{1 + a_{(i+1)}^T S_i a_{(i+1)}}, \\ X_{i+1} &= X_i + S_{(i+1)} a_{(i+1)} (b_{(i+1)}^T - a_{(i+1)}^T X_i) \end{aligned}\right\} for\ i = 0,1,\ldots,K-1$$

where:

$X_0 = \bar{0}$,

$S_0 = \gamma I$, (where γ is a large number)

$a_i^T = i^{th}$ line of matrix A

$b_i^T = i^{th}$ element of vector B

$X^* = X_k$

ANFIS Back-propagation

- Error measure E_k (for the kth $(1 \leq k \leq K)$ entry of the training data)

$$E_k = \sum_{i=0}^{N(L)} (t_i - o_{L,i})^2$$

where:
$N(L) = number\ of\ nodes\ in\ layer\ L$
$t_i = i^{th}\ component\ of\ desired\ output\ vector$
$o_{L,i} = i^{th}\ component\ of\ actual\ output\ vector$

- Let us simplify the error equation (assuming single dimension output vector)to:

$$E_k = \frac{1}{2}(t_k - o_k)^2$$

- Overall Error measure E is:

$$E = \sum_{k=1}^{K} E_k$$

- For each premise parameter the update formula is:

$$\Delta\beta_i = -\eta \frac{\partial^+ E}{\partial \beta_i}$$

Where,

$$\eta = \frac{\kappa}{\sqrt{\sum_i \left(\frac{\partial E}{\partial \beta_i}\right)^2}}, is\ the\ learning\ rate$$

$\kappa\ is\ step\ size$

$\frac{\partial^+ E}{\partial \beta_i}\ is\ the\ ordered\ derivative$

Derivation of the Partial Derivatives

- It is necessary to calculate the partial derivatives of the error function with respect to the parameters of the fuzzy system, which need to be tuned
- In other words, for every parameter β_i we have to calculate:

$$\frac{\partial E_k}{\partial \beta_i} = \frac{\partial E_k}{\partial o_k}\frac{\partial o_k}{\partial \beta_i} = -(t_k - o_k)\frac{\partial o_k}{\partial \beta_i}$$

- Using the chain rule error rate for ***consequent*** parameters can be calculated as:

$$\frac{\partial E_k}{\partial \beta_c} = \frac{\partial E}{\partial o_5}\frac{\partial o_5}{\partial o_4}\frac{\partial o_4}{\partial \beta_c}$$

where β_c is the consequence parameter and o_i is the output of the i^{th} layer

- The error rate for ***premise*** parameters can be calculated as follows:

$$\frac{\partial E_k}{\partial \beta_p} = \frac{\partial E}{\partial o_5}\frac{\partial o_5}{\partial o_4}\frac{\partial o_4}{\partial o_3}\frac{\partial o_3}{\partial o_2}\frac{\partial o_2}{\partial o_1}\frac{\partial o_1}{\partial \beta_p}$$

- We have:

$$\frac{\partial E}{\partial o_5} = -(t_k - o_k)$$

$$\frac{\partial o_5}{\partial o_4} = \frac{\partial\left(\sum f_j \overline{w_j}\right)}{\partial(f_i \overline{w_i})} = 1$$

  - where $\overline{w_i}$ is the normalized firing strength of the i^{th} rule

$$\frac{\partial o_4}{\partial o_3} = \frac{\partial f_i \overline{w_i}}{\partial \overline{w_i}} = f_i$$

  - where i is the number of the corresponding rule (or, alternatively, the number of the unit in the 3rd layer)

- Next:

$$\frac{\partial o_3}{\partial o_2} = \frac{\partial}{\partial w_i}\left(\frac{w_i}{\sum_{j=1}^{n} w_j}\right) = \frac{\sum_{j=1}^{n} w_j - w_i}{\left(\sum_{j=1}^{n} w_j\right)^2}$$

  - where i is the number of the corresponding rule (or, alternatively, the number of the unit in the 2nd layer) and n is the total number of rules in the system

$$\frac{\partial o_2}{\partial o_1} = \frac{\partial}{\partial A_m}\left(\prod_{A_j \in \Re(A_m)} A_j\right) = \prod_{A_j \in \Re(A_m), A_j \neq A_m} A_j$$

  - where $A_j \in \Re(A_m)$ denotes the fuzzy sets, which make the premise part of the rule containing fuzzy set A_m

- For the consequence parameter β_c :

$$\frac{\partial E}{\partial \beta_c} = -(t - o)\frac{\partial o_4}{\partial \beta_c}$$

- Calculating the partial derivatives of the output functions w.r.t. their parameters:

$$\frac{\partial o_4}{\partial p_i} = \frac{\partial}{\partial p_i}(f_i \overline{w_i}) = \frac{\partial}{\partial p_i}(\overline{w_i}\,(pix + qiy + r_i)\,) = \overline{w_i}x$$

$$\frac{\partial o_4}{\partial q_i} = \frac{\partial}{\partial q_i}(f_i \overline{w_i}) = \frac{\partial}{\partial q_i}(\overline{w_i}\,(pix + qiy + r_i)\,) = \overline{w_i}y$$

$$\frac{\partial o_4}{\partial r_i} = \frac{\partial}{\partial r_i}(f_i \overline{w_i}) = \frac{\partial}{\partial r_i}(\overline{w_i}\,(pix + qiy + r_i)\,) = \overline{w_i}$$

  - where i is the number of the corresponding rule

- For the premise parameter β_p of the membership function of linguistic label A_m :

$$\frac{\partial E}{\partial \beta_p} = -(t - o) f_i \left[\frac{\sum_{j=1}^{n} w_j - w_i}{\left(\sum_{j=1}^{n} w_j\right)^2}\right]\left[\prod_{A_j \in \Re(A_m), A_j \neq A_m} A_j\right]\frac{\partial o_1}{\partial \beta_p}$$

- Let us now calculate $\frac{\partial o_1}{\partial \beta_p}$:

$$\frac{\partial o_1}{\partial a_{ij}} = \frac{\partial}{\partial a_{ij}}\left[exp\left[-\left(\frac{x - c_{ij}}{a_{ij}}\right)^{2b_{ij}}\right]\right]$$

$$= 2\frac{\left(\frac{x - c_{ij}}{a_{ij}}\right)^{2b_{ij}} b_{ij}\left[exp\left[-\left(\frac{x - c_{ij}}{a_{ij}}\right)^{2b_{ij}}\right]\right]}{a_{ij}}$$

$$\frac{\partial o_1}{\partial b_{ij}} = \frac{\partial}{\partial b_{ij}}\left[exp\left[-\left(\frac{x-c_{ij}}{a_{ij}}\right)^{2b_{ij}}\right]\right]$$

$$= -2\left(\frac{x-c_{ij}}{a_{ij}}\right)^{2b_{ij}} ln\left(\frac{x-c_{ij}}{a_{ij}}\right)\left[-\left(\frac{x-c_{ij}}{a_{ij}}\right)^{2b_{ij}}\right]$$

$$\frac{\partial o_1}{\partial c_{ij}} = \frac{\partial}{\partial c_{ij}}\left[exp\left[-\left(\frac{x-c_{ij}}{a_{ij}}\right)^{2b_{ij}}\right]\right]$$

$$= 2\frac{\left(\frac{x-c_{ij}}{a_{ij}}\right)^{2b_{ij}} b_{ij}\left[exp\left[-\left(\frac{x-c_{ij}}{a_{ij}}\right)^{2b_{ij}}\right]\right]}{x-c_{ij}}$$

– where i is the number of the corresponding rule and j is the number of the corresponding linguistic variable in the rule

$$\frac{\partial o_1}{\partial a_{ij}} = \begin{cases} \dfrac{2\left|\frac{x-c_{ij}}{a_{ij}}\right|^{2b} b_{ij}\left(x-c_{ij}\right)}{\left(1+\left|\frac{x-c_{ij}}{a_{ij}}\right|^{2b_{ij}}\right)^2 a_{ij}^2\left|\frac{x-c_{ij}}{a_{ij}}\right|}, & \text{if } \dfrac{x-c_{ij}}{a_{ij}} > 0 \\ \dfrac{-2\left|\frac{x-c_{ij}}{a_{ij}}\right|^{2b} b_{ij}\left(x-c_{ij}\right)}{\left(1+\left|\frac{x-c_{ij}}{a_{ij}}\right|^{2b_{ij}}\right)^2 a_{ij}^2\left|\frac{x-c_{ij}}{a_{ij}}\right|}, & \text{if } \dfrac{x-c_{ij}}{a_{ij}} < 0 \\ 0, & \text{if } \dfrac{x-c_{ij}}{a_{ij}} = 0 \end{cases}$$

$$\frac{\partial o_1}{\partial b_{ij}} = \begin{cases} \dfrac{-2\left|\dfrac{x-c_{ij}}{a_{ij}}\right|^{2b_{ij}} \ln\left(\left|\dfrac{x-c_{ij}}{a_{ij}}\right|\right)}{\left(1+\left|\dfrac{x-c_{ij}}{a_{ij}}\right|^{2b_{ij}}\right)^2}, & \text{if } \dfrac{x-c_{ij}}{a_{ij}} \neq 0 \\ 0, & \text{if } \dfrac{x-c_{ij}}{a_{ij}} = 0 \end{cases},$$

$$\frac{\partial o_1}{\partial c_{ij}} = \begin{cases} \dfrac{2\left|\dfrac{x-c_{ij}}{a_{ij}}\right|^{2b} b_{ij}}{\left(1+\left|\dfrac{x-c_{ij}}{a_{ij}}\right|^{2b_{ij}}\right)^2 a_{ij}\left|\dfrac{x-c_{ij}}{a_{ij}}\right|}, & \text{if } \dfrac{x-c_{ij}}{a_{ij}} > 0. \\ \dfrac{-2\left|\dfrac{x-c_{ij}}{a_{ij}}\right|^{2b} b_{ij}}{\left(1+\left|\dfrac{x-c_{ij}}{a_{ij}}\right|^{2b_{ij}}\right)^2 a_{ij}\left|\dfrac{x-c_{ij}}{a_{ij}}\right|}, & \text{if } \dfrac{x-c_{ij}}{a_{ij}} < 0. \\ 0, & \text{if } \dfrac{x-c_{ij}}{a_{ij}} = 0. \end{cases}$$

Simplified partial derivatives:

$$\frac{\partial o_1}{\partial a_{ij}} = \begin{cases} \frac{2b_{ij}}{a_{ij}} y_{ij}\left(1 - y_{ij}\right), & \text{if } x \neq c_{ij}. \\ 0, & \text{if } x = c_{ij}. \end{cases}$$

$$\frac{\partial o_1}{\partial b_{ij}} = \begin{cases} -2\ln\left|\frac{x - c_{ij}}{a_{ij}}\right| y_{ij}\left(1 - y_{ij}\right), & \text{if } x \neq c_{ij}. \\ 0, & \text{if } x = c_{ij}. \end{cases}$$

$$\frac{\partial o_1}{\partial c_{ij}} = \begin{cases} \frac{2b_{ij}}{x - c_{ij}} y_{ij}\left(1 - y_{ij}\right), & \text{if } x \neq c_{ij}. \\ 0, & \text{if } x = c_{ij}. \end{cases}$$

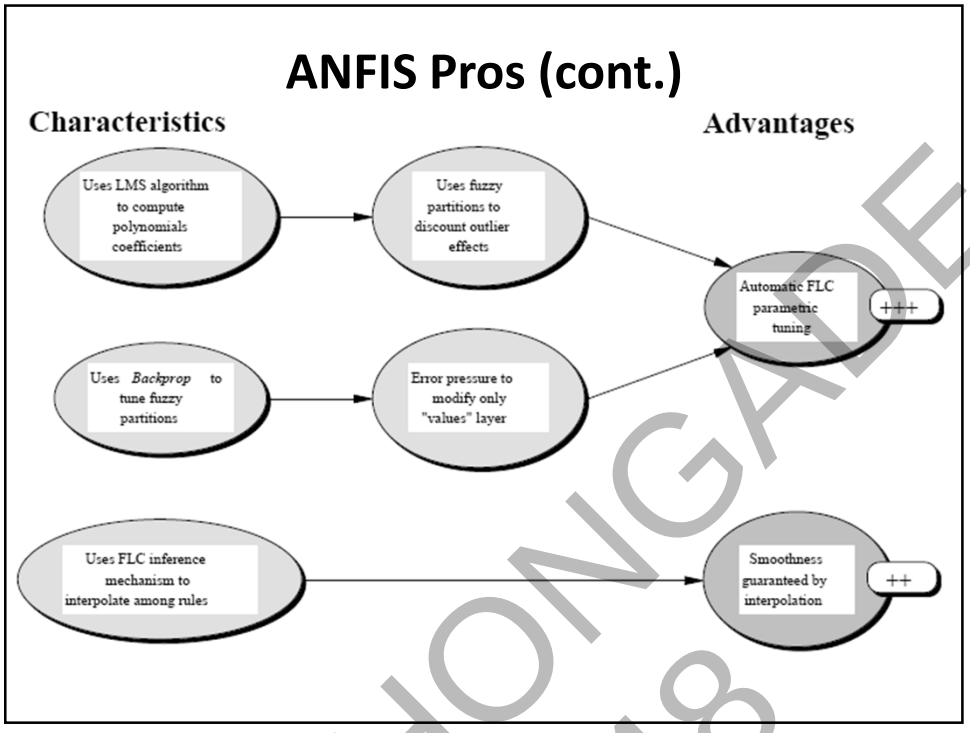
Where:

$$y_{ij} = \frac{1}{1 + \left|\frac{x - c_{ij}}{a_{ij}}\right|^{2b_{ij}}}.$$

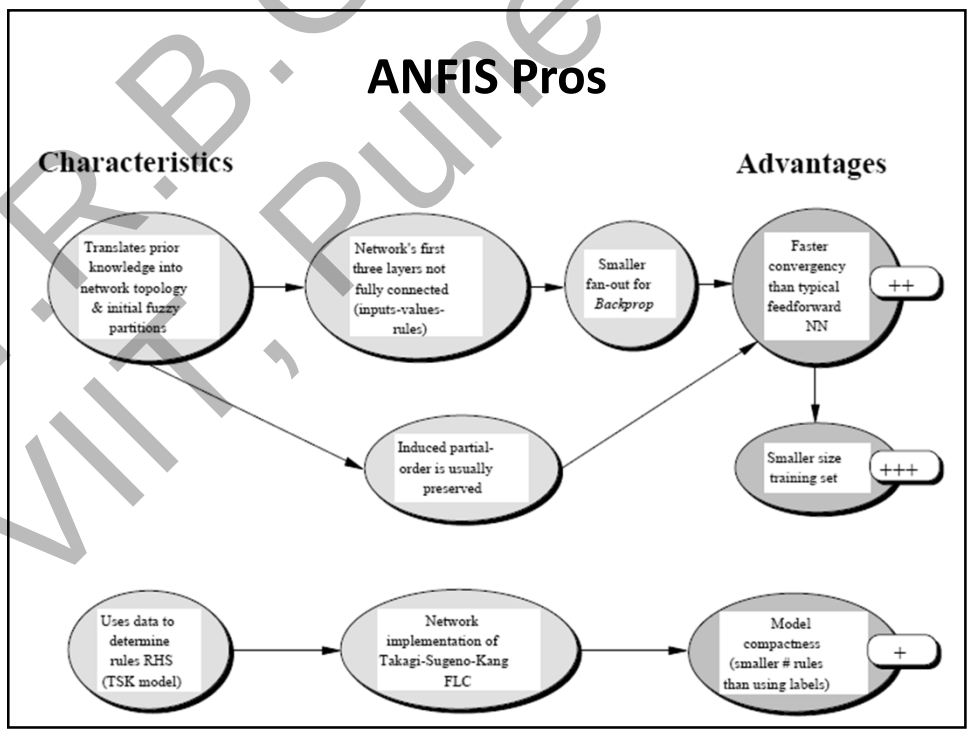
ANFIS Pros and Cons

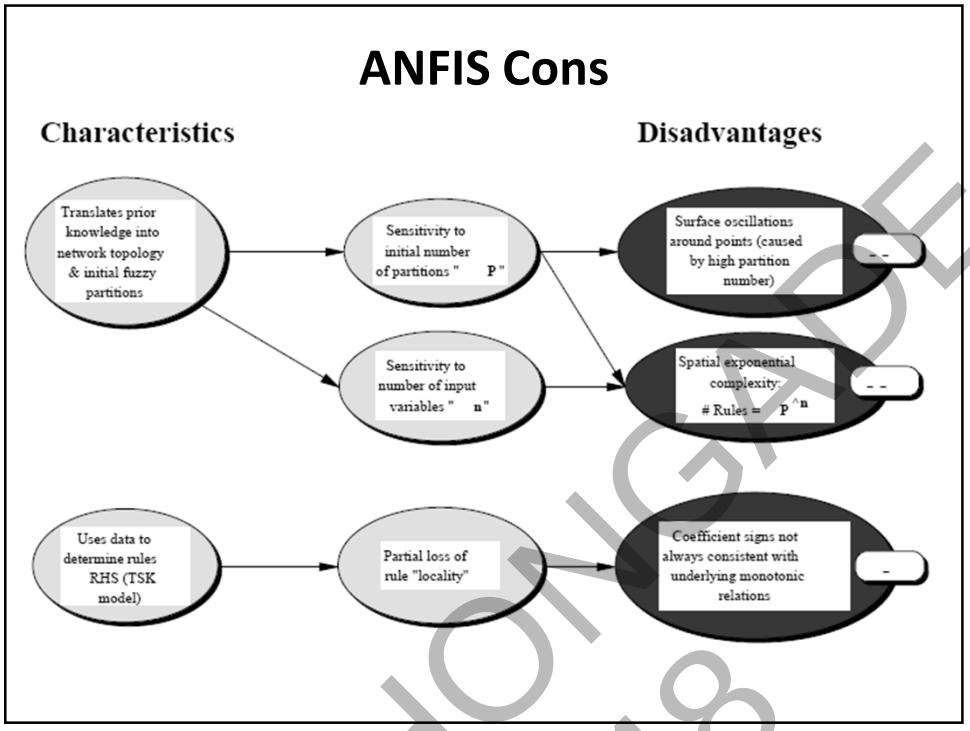
- ANFIS is one of the best tradeoff between neural and fuzzy systems, providing:
  - *smoothness*, due to the FC interpolation
  - *adaptability*, due to the NN Backpropagation
- ANFIS however has strong computational complexity restrictions

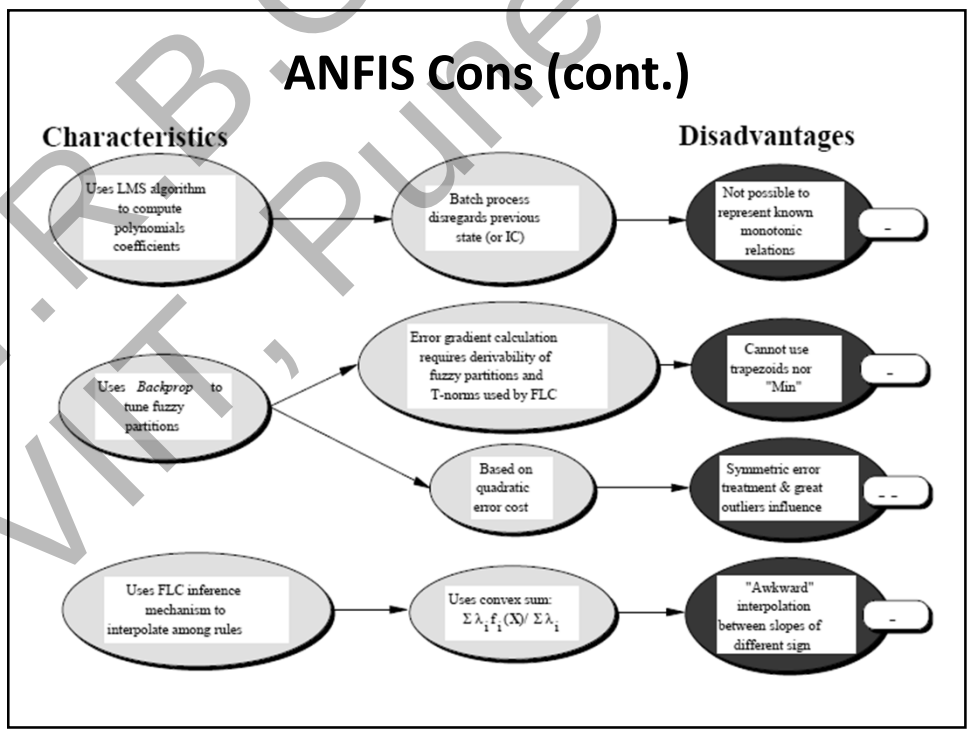
ANFIS Pros (cont.)



ANFIS Pros







ANFIS Pros & Cons Summary

- Pros:
  - Automatic FLC parametric tuning
  - Smoothness guaranteed by interpolation
  - Faster convergence than typical feedforward neural network
  - Smaller size training set
  - Model compactness(smaller number of rules than using labels)
- Cons:
  - Surface oscillations around points (caused by larger number of partitions
  - Spatial exponential complexity
  - Coefficient signs not always consistent with underlying monotonic relationships
  - Not possible to represent known monotonic relations
  - Cannot use trapezoids or MIN
  - Symmetric error treatment and great outlier influence
  - "Awkward" interpolation between slopes of different signs

ANFIS Possible Modifications

- Changes to decrease ANFIS *complexity*
  - Use "don't care" values in rules (no connection between any node of value layer and rule layer)
  - Use reduced subset of state vector in partition tree while evaluating linear functions on complete state

$$\bar{X} = \left(\overline{X_r} \cup \overline{X_{(n-r)}}\right)$$

  - Use heterogeneous partition granularity (different partitions p_i for each state variable, instead of "p")

$$\#RULES = \prod_{i=1}^{n} p_i$$

ANFIS Possible Modifications (contd.)

- Changes to extend ANFIS *applicability*
  - Use other cost function (rather than SSE) to represent the user's utility values of the error (error asymmetry, saturation effects of outliers, etc.)
  - Use other type of aggregation function (rather than convex sum) to better handle slopes of different signs

Summary

- ANFIS presents an alternative soft computing tool for regression, system identification and classification(multiple outputs: there ANFIS is called CANFIS ,*co-active neuro-fuzzy inference system*)
- It can be shown that ANFIS can be a generic RBFN network
- ANFIS can also be proved to be an universal approximator (like MLP with backpropagation)
- Further ANFIS takes less iterations to converge compared with an equivalent MLP with backpropagation

Thank you!!!