

Adaptive Neural Fuzzy Inference Systems (ANFIS): Analysis and Applications

Piero P. Bonissone
GE CRD,
Schenectady, NY USA
Email: Bonissone@crd.ge.com

Outline

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

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

FC of Type II, III, and ANFIS

- Type II Fuzzy Control must be tuned manually
- Type III Fuzzy Control (TSK variety) have an automatic RHS tuning
- ANFIS will provide both RHS and LHS tuning

ANFIS Neurons: Clarification note

- Note that neurons in ANFIS have different structures:
 - Values [membership function defined by parameterized soft trapezoids]
 - Rules [differentiable T-norm - usually product]
 - Normalization [arithmetic division]
 - Functions [linear regressions and multiplication with normalized λ]
 - Output [Algebraic Sum]

ANFIS as a Type III FC

- (L_0): FC state variables are nodes in ANFIS *inputs layer*
- (L_1): termsets of each state variable are nodes in ANFIS *values layer*, computing the membership value
- (L_2): each rule in FC is a node in ANFIS *rules layer* using soft-min or product to compute the rule matching factor λ_i
- (L_3): each λ_i is scaled into λ'_i in the *normalization layer*
- (L_4): each λ'_i weighs the result of its linear regression f_i in the *function layer*, generating the rule output
- (L_5): each rule output is added in the *output layer*

ANFIS as a generalization of CART

- Classification and Regression Tree (CART)
 - Algorithm defined by Breiman et al in 1984
 - Creates a binary decision tree to classify the data into one of 2^n linear regression models to minimize the Gini index for the current node c :

$$\text{Gini}(c) = 1 - \sum_j p_j^2$$

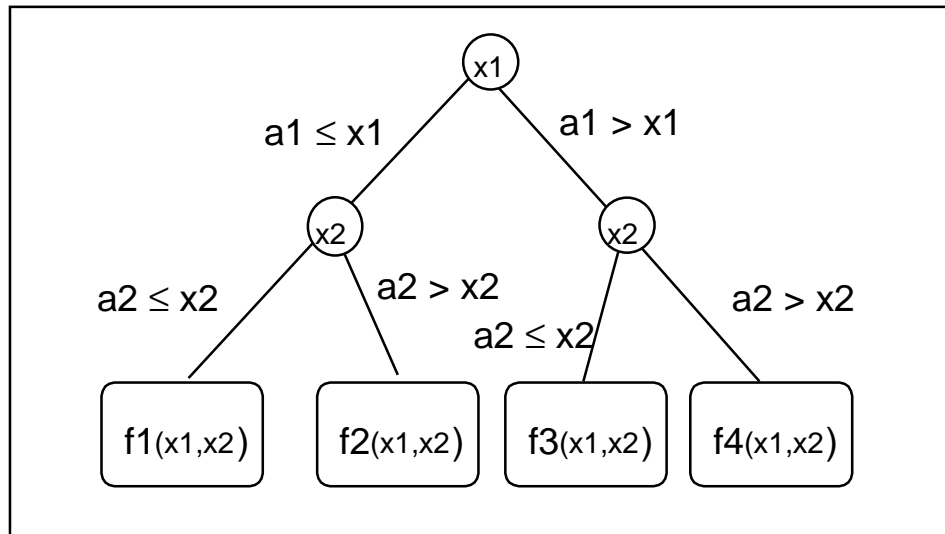
where:

- p_j is the probability of class j in node c
- $\text{Gini}(c)$ measure the amount of “impurity” (incorrect classification) in node c

CART Problems

- Discontinuity
- Lack of locality (sign of coefficients)

Cart Binary Partion Tree and Rule table Representation

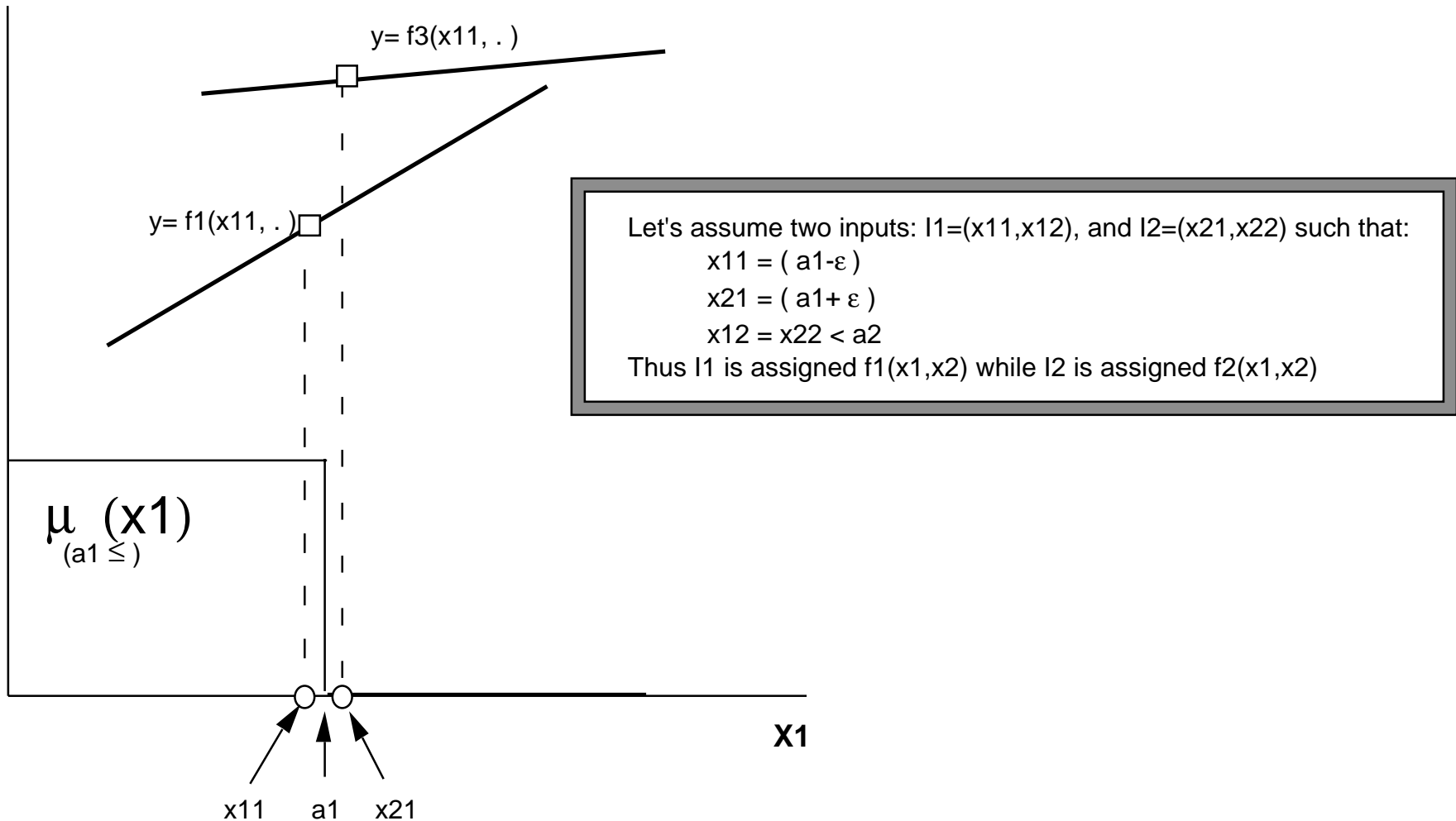


Partition Tree

x1	x2	y
$a1 \leq$	$a2 \leq$	$f1(x1,x2)$
$a1 \leq$	$> a2$	$f2(x1,x2)$
$a1 >$	$a2 \leq$	$f3(x1,x2)$
$a1 >$	$> a2$	$f4(x1,x2)$

Rule Table

Discontinuities due to small input perturbations



ANFIS Characteristics

- Adaptive Neural Fuzzy Inference System (ANFIS)
 - Algorithm defined by J.-S. Roger Jang in 1992
 - Creates a fuzzy decision tree to classify the data into one of 2^n (or p^n) linear regression models to minimize the sum of squared errors (SSE):

$$SSE = \sum_j e_j^2$$

where:

- e_j is the error between the desired and the actual output
- p is the number of fuzzy partitions of each variable
- n is the number of input variables

ANFIS Computational Complexity

<u>Layer #</u>	<u>L-Type</u>	<u># Nodes</u>	<u># Param</u>
L ₀	Inputs	n	0
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

ANFIS Parametric Representation

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

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

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

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

ANFIS Learning Algorithms

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

ANFIS LSE Batch Algorithm

- LSE used in Forward pass:
 - $S = \{S1 \cup S2\}$, and $\{S1 \cap S2 = \emptyset\}$
Output = $F(\bar{I}, S)$, where \bar{I} is the input vector
 $H(\text{Output}) = H \circ F(\bar{I}, S)$, where $H \circ F$ is linear in S_2
 - For given values of S_1 , using K training data, we can transform the above equation into **$B=AX$** , where X contains the elements of S_2
 - This is solved by: **$(A^T A)^{-1} A^T B = X^*$** where $(A^T A)^{-1} A^T$ is the pseudo-inverse of A (if $A^T A$ is nonsingular)
 - The LSE minimizes the error $\|AX-B\|^2$ by approximating X with X^*

ANFIS LSE Batch Algorithm (cont.)

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

$$S_{i+1} = S_i - \frac{S_i a_{(i+1)} a_{(i+1)}^T S_i}{1 + a_{(i+1)}^T S_i a_{(i+1)}},$$

$$X_{i+1} = X_i + S_{(i+1)} a_{(i+1)} (b_{(i+1)}^T - a_{(i+1)}^T X_i)$$

$$X_0 = \bar{0},$$

} for $i = 0, 1, \dots, K - 1$

$S_0 = \gamma I$, (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 Backprop

- Error measure E_k
(for the k^{th} ($1 \leq k \leq K$) entry of the training data)

$$E_k = \sum_{i=1}^{N(L)} (d_i - x_{L,i})^2$$

where:

$N(L)$ = number nodes in layer L

d_i = i^{th} component of *desired* output vector

$x_{L,i}$ = i^{th} component of *actual* output vector

- Overall error measure E :

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

ANFIS Backprop (cont.)

- For each parameter α_i the update formula is:

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

where:

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

κ is the step size

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

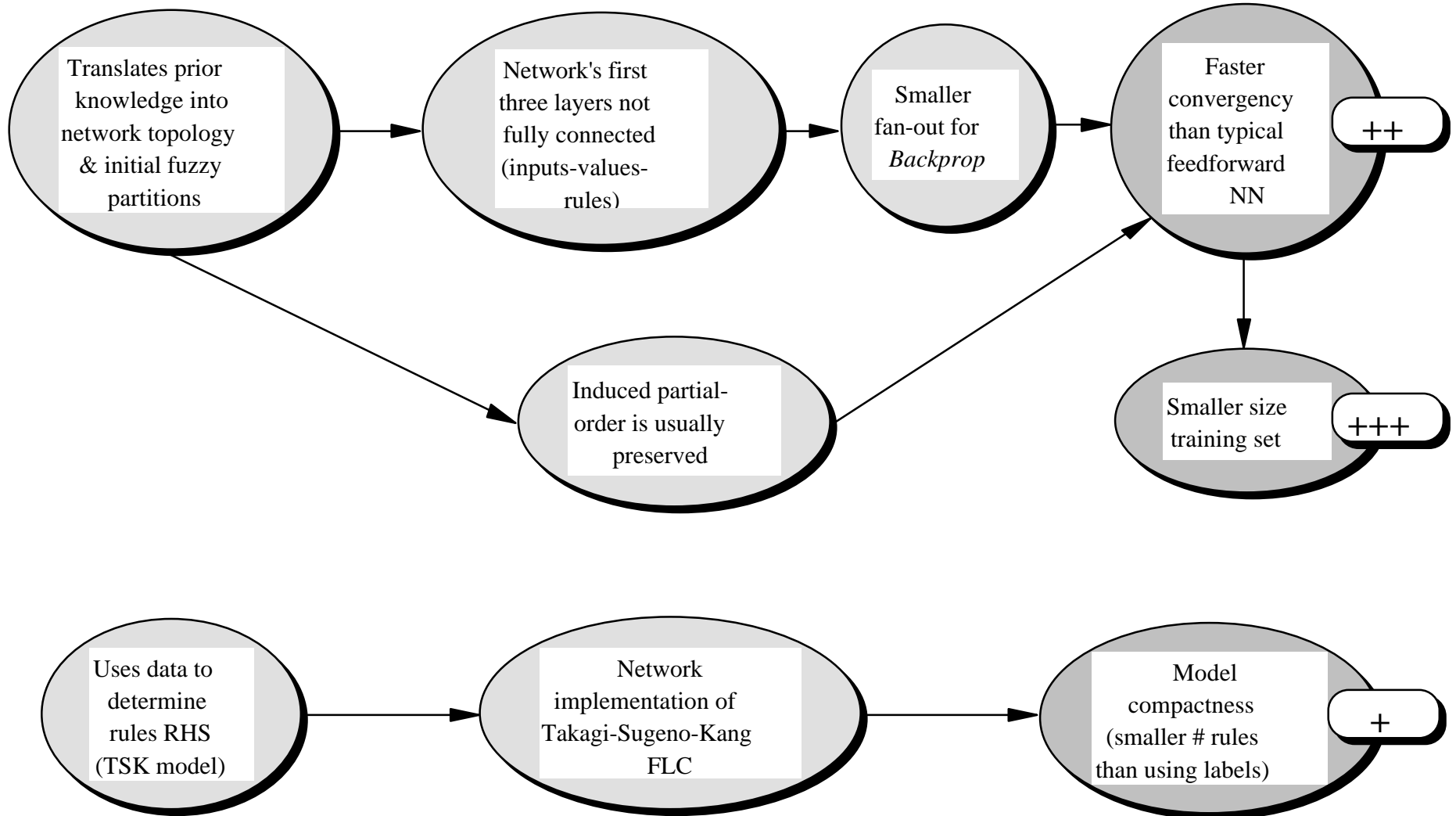
ANFIS Pros and Cons

- 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

Characteristics

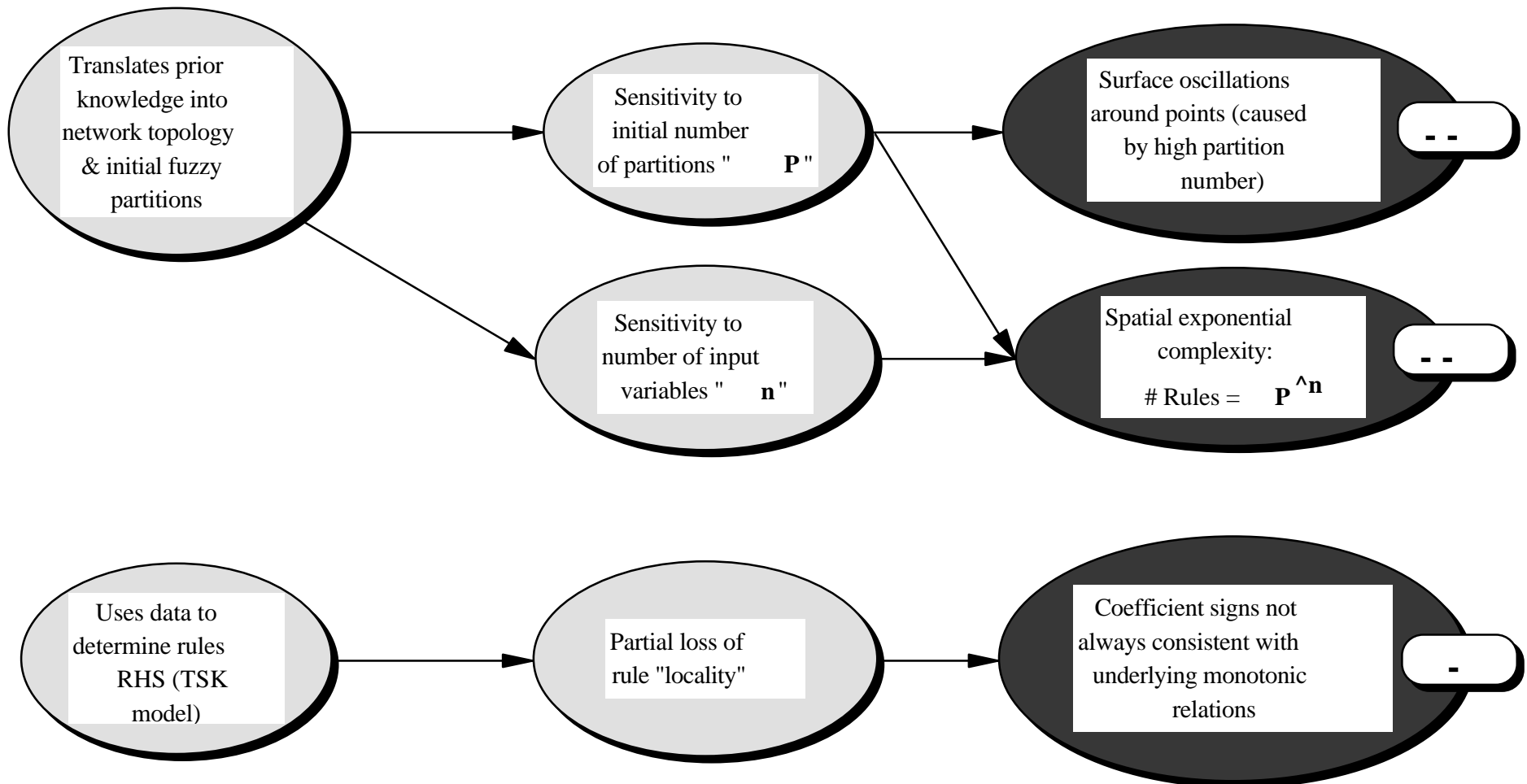
Advantages



ANFIS Cons

Characteristics

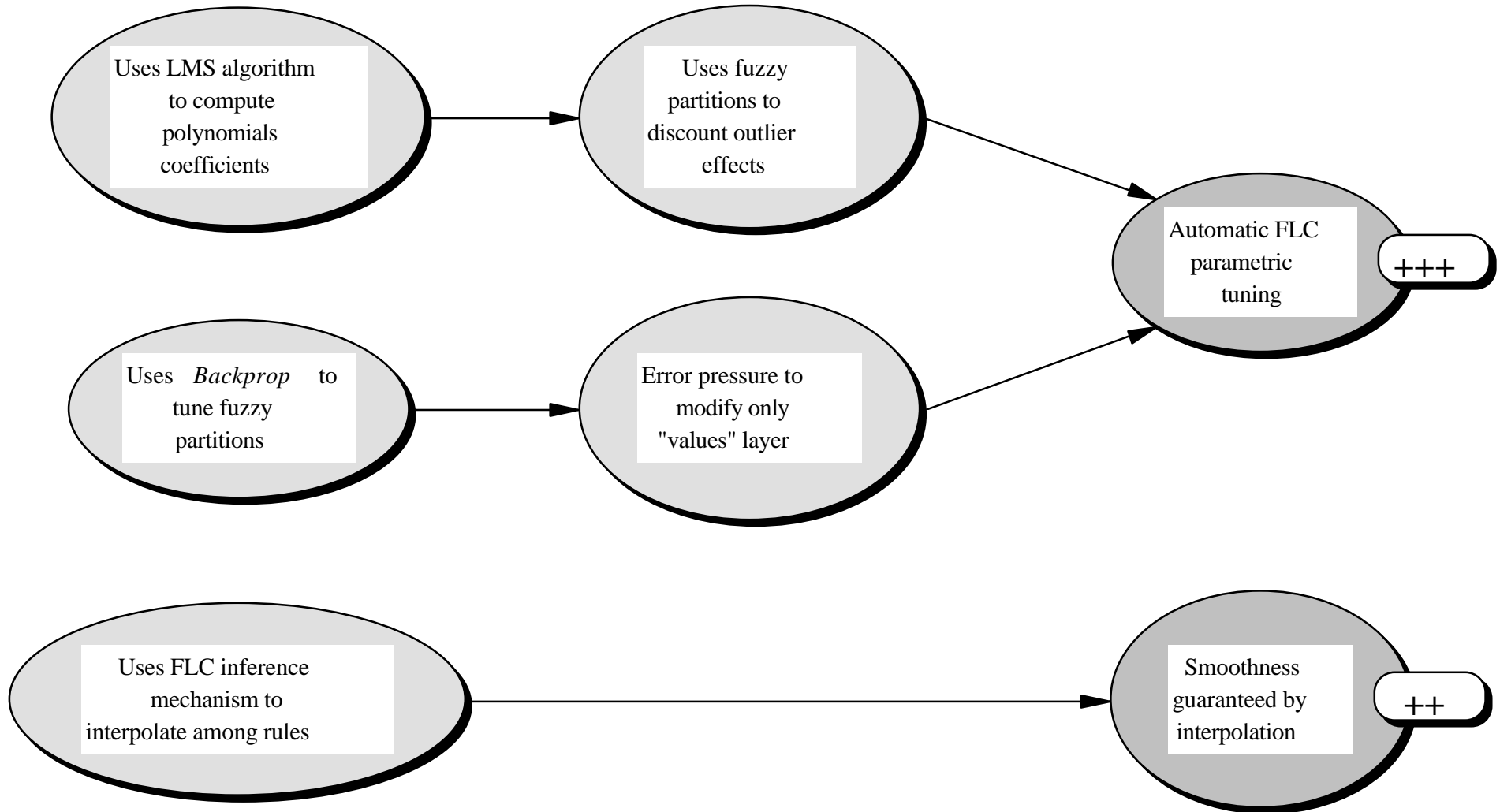
Disadvantages



ANFIS Pros (cont.)

Characteristics

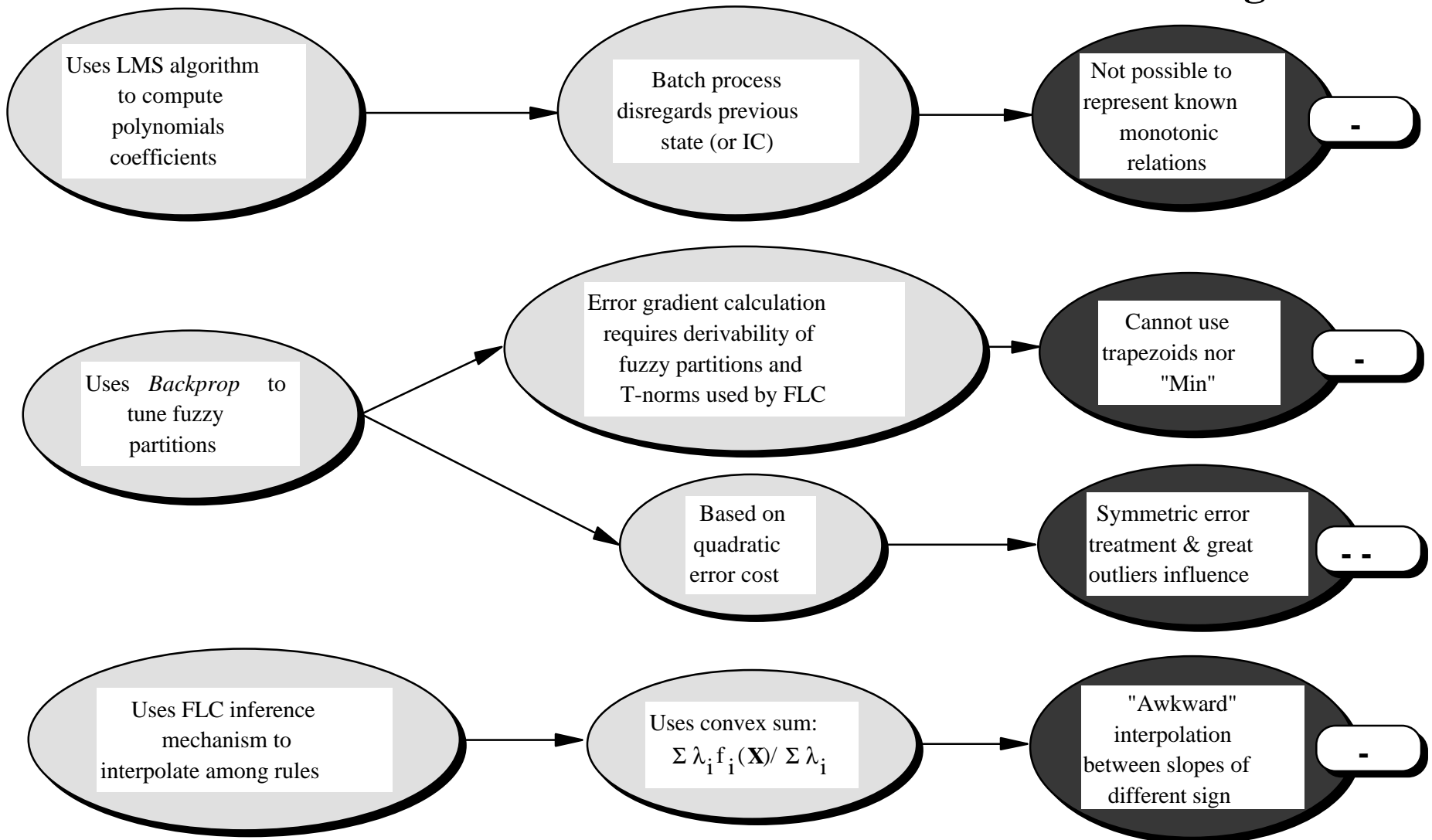
Advantages



ANFIS Cons (cont.)

Characteristics

Disadvantages



ANFIS Opportunities

- 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

$$\bar{X} = (\bar{X}_r \cup \bar{X}_{(n-r)})$$

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

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

ANFIS Opportunities (cont.)

- 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.

ANFIS Applications at GE

- Margoil Oil Thickness Estimator
- Voltage Instability Predictor (Smart Relay)
- Collateral Evaluation for Mortgage Approval

ANFIS References

- “ANFIS: Adaptive-Network-Based Fuzzy Inference System”, J.S.R. Jang, *IEEE Trans. Systems, Man, Cybernetics*, 23(5/6):665-685, 1993.
- “Neuro-Fuzzy Modeling and Control”, J.S.R. Jang and C.-T. Sun, *Proceedings of the IEEE*, 83(3):378-406
- “Industrial Applications of Fuzzy Logic at General Electric”, Bonissone, Badami, Chiang, Khedkar, Marcelle, Schutten, *Proceedings of the IEEE*, 83(3):450-465
- *The Fuzzy Logic Toolbox for use with MATLAB*, J.S.R. Jang and N. Gulley, Natick, MA: The MathWorks Inc., 1995
- *Machine Learning, Neural and Statistical Classification* Michie, Spiegelhart & Taylor (Eds.), NY: Ellis Horwood 1994
- *Classification and Regression Trees*, Breiman, Friedman, Olshen & Stone, Monterey, CA: Wadsworth and Brooks, 1985