ANFIS: Adaptive Neuro-Fuzzy Inference Systems

Adriano Oliveira Cruz

PPGI, IM-NCE, UFRJ

Summary

- Introduction
- ANFIS Architecture
- Hybrid Learning Algorithm
- ANFIS as a Universal Approximatior
- Simulation Examples

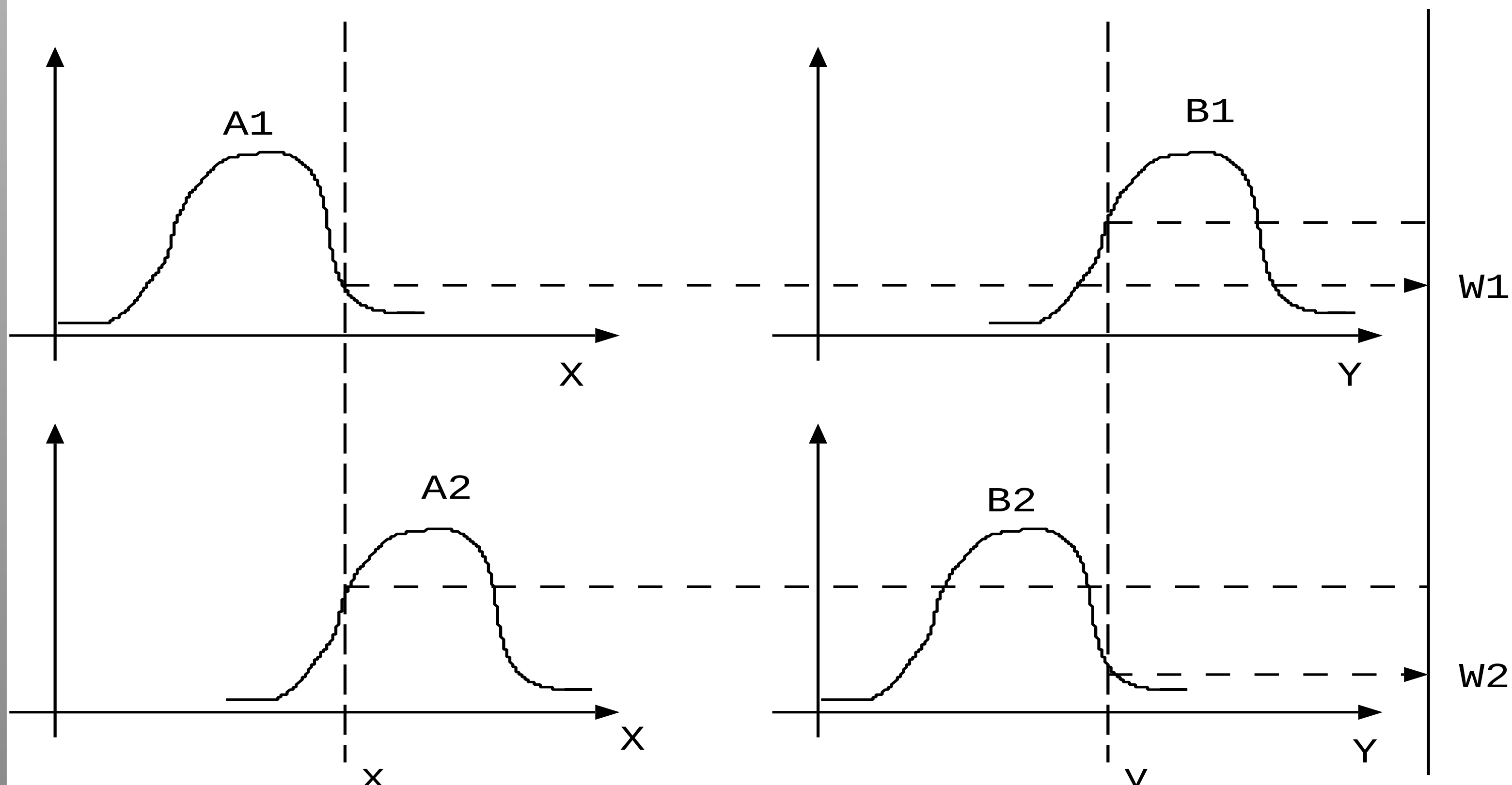
Introduction

- ANFIS: Artificial Neuro-Fuzzy Inference Systems
- ANFIS are a class of adaptive networks that are funcionally equivalent to fuzzy inference systems.
- ANFIS represent Sugeno e Tsukamoto fuzzy models.
- ANFIS uses a hybrid learning algorithm

Sugeno Model

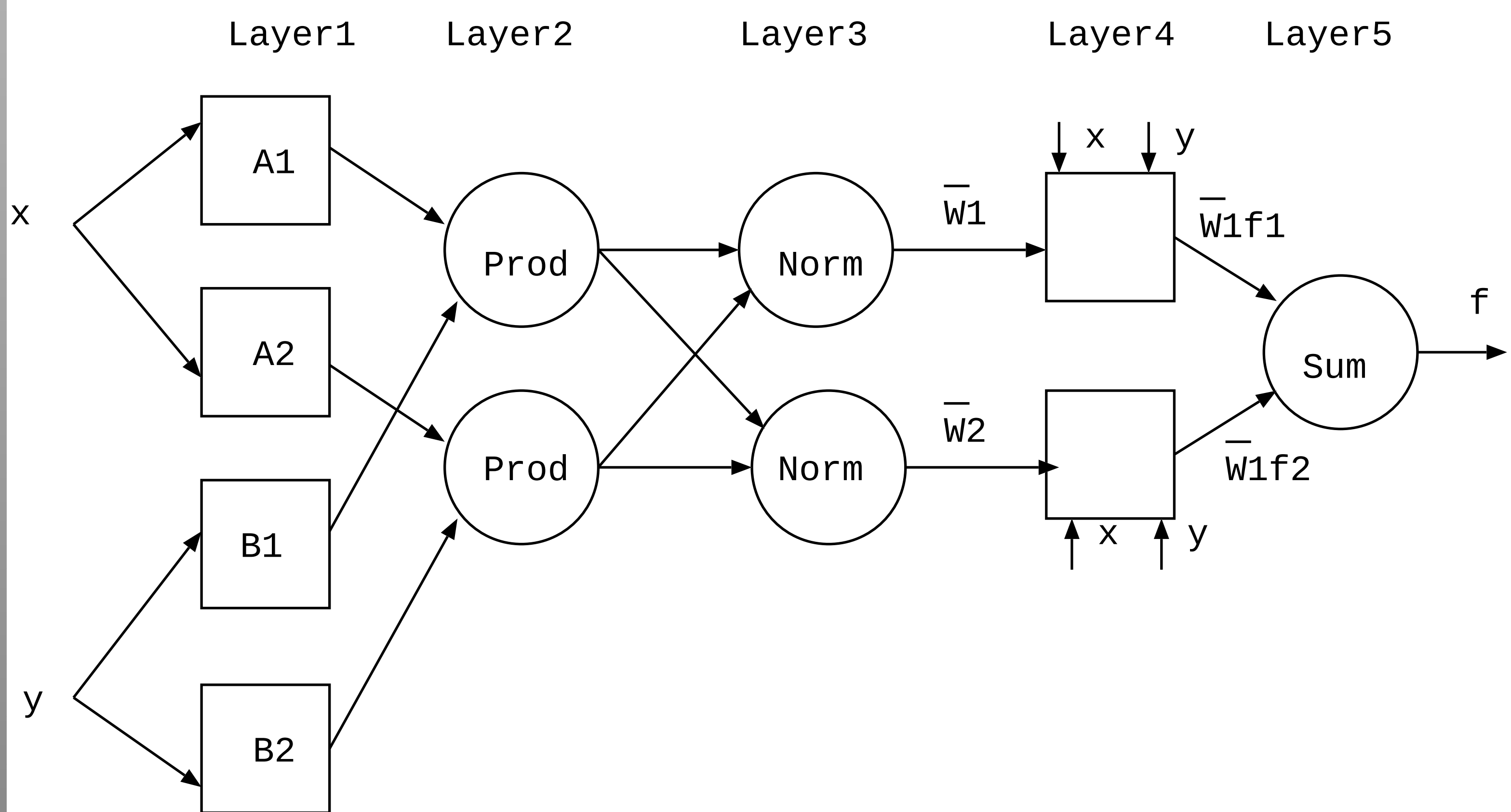
- Assume that the fuzzy inference system has two inputs x and y and one output z.
- A first-order Sugeno fuzzy model has rules as the following:
- Rule1:
  If x is A_1 and y is B_1, then $f_1 = p_1x + q_1y + r_1$
- Rule2:
  If x is A_2 and y is B_2, then $f_2 = p_2x + q_2y + r_2$

Sugeno Model - I



$f1=p1x+q1y+r1$ $\quad$ $f2=p2x+q2y+r2$ $\quad$ $f=\dfrac{w1.f1+w2.f2}{w1+w2}$

ANFIS Architecture



Layer 1 - I

- $O_{l,i}$ is the output of the ith node of the layer l.
- Every node i in this layer is an adaptive node with a node function
  $O_{1,i} = \mu_{A_i}(x)$ for $i = 1, 2,$ or
  $O_{1,i} = \mu_{B_{i-2}}(x)$ for $i = 3, 4$
- x (or y) is the input node i and A_i (or B_{i-2}) is a linguistic label associated with this node
- Therefore $O_{1,i}$ is the membership grade of a fuzzy set (A_1, A_2, B_1, B_2).

Layer 1 - II

- Typical membership function:

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

- a_i, b_i, c_i is the parameter set.
- Parameters are referred to as **premise parameters**.

Layer 2

- Every node in this layer is a fixed node labeled `Prod`.
- The output is the product of all the incoming signals.
- $O_{2,i} = w_i = \mu_{A_i}(x) \cdot \mu_{B_i}(y), \quad i = 1, 2$
- Each node represents the fire strength of the rule
- Any other T-norm operator that perform the AND operator can be used

Layer 3

- Every node in this layer is a fixed node labeled `Norm`.
- The ith node calculates the ratio of the ith rulet's firing strenght to the sum of all rulet's firing strengths.
- $O_{3,i} = \overline{w}_i = \frac{w_i}{w_1+w_2}, \quad i = 1, 2$
- Outputs are called **normalized firing strengths.**

Layer 4

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

$$O_{4,1} = \overline{w}_i f_i = \overline{w}_i (p_x + q_i y + r_i)$$

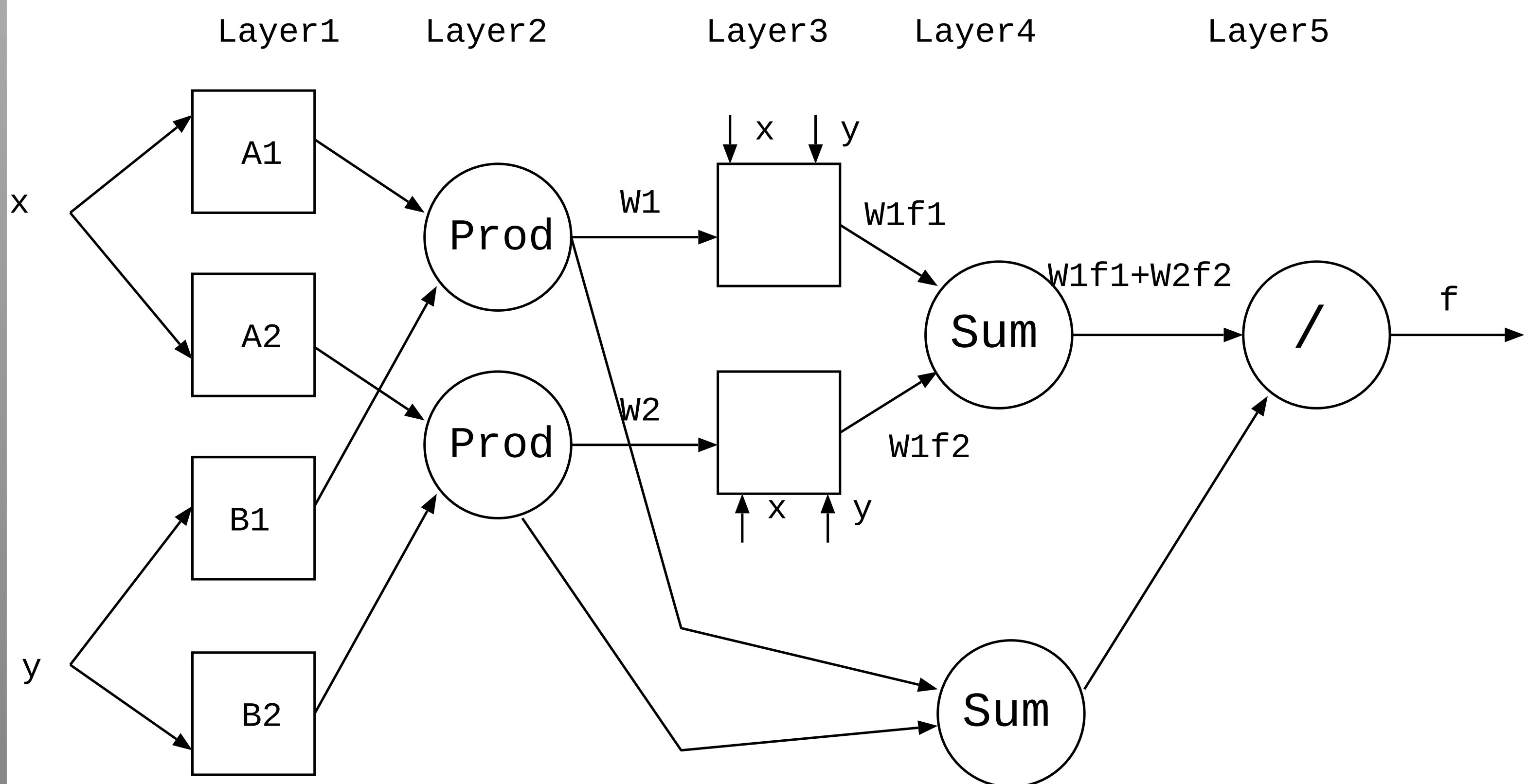
- $\overline{w}_i$ is the normalized firing strenght from layer 3.
- $\{p_i, q_i, r_i\}$ is the parameter set of this node.
- These are referred to as **consequent parameters.**

Layer 5

- The single node in this layer is a fixed node labeled `sum`, which computes the overall output as the summation of all incoming signals:
- $overall\ output = O_{5,1} = \sum_i \overline{w}_i f_i = \frac{\sum_i w_i f_i}{\sum_i w_i}$

Alternative Structures

- There are other structures



Learning Algorithm

Hybrid Learning Algorithm - I

- The ANFIS can be trained by a hybrid learning algorithm presented by Jang in the chapter 8 of the book.
- In the forward pass the algorithm uses least-squares method to identify the consequent parameters on the layer 4.
- In the backward pass the errors are propagated backward and the premise parameters are updated by gradient descent.

Hybrid Learning Algorithm - II

	Forward Pass	Backward Pass
Premise Parameters	Fixed	Gradient Descent
Consequent Parameters	Least-squares estimator	Fixed
Signals	Node outputs	Error signals

Two passes in the hybrid learning algorithm for ANFIS.

Basic Learning Rule Definitions

- Suppose that an adptive network has L layers and the kth layer has #(k) nodes.
- We can denote the node in the ith position of the kth layer by (k, i).
- The node function is denoted by O_i^k.
- Since the node output depends on its incoming signals and its parameter set (a, b, c), we have

$$O_i^k = O_i^k(O_i^{k-1}, \ldots, O_{\#(k-1)}^{k-1}, a, b, c)$$

- Notice that O_i^k is used as both node output and node function.

Error Measure

- Assume that a training data set has P entries.
- The error measure for the pth entry can be defined as the sum of the squared error

$$E_p = \sum_{m=1}^{\#(L)} (T_{m,p} - O_{m,p}^L)^2$$

- $T_{m,p}$ is the mth component of the pth target.
- $O_{m,p}^L$ is the mth component the actual output vector.
- The overall error is

$$E = \sum_{p=1}^{P} E_p$$

Error Rate for each output

- In order to implement the gradient descent in E we calculate the error rate $\frac{\partial E}{\partial O}$ for the pth training data for each node output O.
- The error rate for the output note at (L, i) is

$$\frac{\partial Ep}{\partial O_{i,p}^{L}} = -2(T_{i,p} - O_{i,p}^{L}) \tag{1}$$

- For the internal node at (k, i), the error rate can be derived by the chain rule:

$$\frac{\partial E_p}{\partial O_{i,p}^{k}} = \sum_{m=1}^{\#(k+1)} \frac{\partial E_p}{\partial O_{m,p}^{k+1}} \frac{\partial O_{m,p}^{k+1}}{\partial O_{i,p}^{k}}, \tag{2}$$

  where $1 \leq k \leq L-1$

- The error rate of an internal node is a linear combination of the error rates of the nodes in the next layer.

Error Rate for each parameter

- Consider α one of the parameters.
- Therefore
$$\frac{\partial E_p}{\partial \alpha} = \sum_{O^* \in S} \frac{\partial E_p}{\partial O^*} \frac{\partial O^*}{\partial \alpha}, \tag{3}$$
where S is the set of nodes whose outputs depend on α
- The derivative of the overall error with respect to α is
$$\frac{\partial E}{\partial \alpha} = \sum_{p=1}^{P} \frac{\partial E_p}{\partial \alpha}, \tag{4}$$
- The update formula for α is
$$\Delta \alpha = \eta \frac{\partial E}{\partial \alpha}$$

Learning Paradigms

- If the parameters are to be updated after each input-output pair (on-line training) then the update formula is:

$$\frac{\partial E_p}{\partial \alpha} = \sum_{O^* \in S} \frac{\partial E_p}{\partial O^*} \frac{\partial O^*}{\partial \alpha} \tag{5}$$

- With the batch learning (off-line learning) the update formula is based on the derivative of the overall error with respect to α:

$$\frac{\partial E}{\partial \alpha} = \sum_{p=1}^{P} \frac{\partial E_p}{\partial \alpha}, \tag{6}$$

Gradient Problems

- The method is slow.
- It is likely to be trapped in local minima.

Hybrid Learning Rule

Hybrid Learning Rule

- Combines:
  - the gradient rule;
  - the least squares estimate.

Definitions

- Considere that the adptive network has only one output.
- $output = F(\mathbf{I}, S)$
- $\mathbf{I}$ is the vector of input variables.
- S is the set of parameters.
- F is the function implemented by the ANFIS.
- If there exists a function H such that the composite function $H \circ F$ is linear in some elements of S then these elements can be identified by LSM.

Continuing Definitions

- More formally, if the parameter set S can be decomposed into two sets $S = S_1 \oplus S_2$ ($\oplus$ direct sum), such that $H \circ F$ is linear in the elements of S_2
- then applying H to $output = F(\mathbf{I}, S)$ we have

$$H(output) = H \circ F(\mathbf{I}, S) \tag{7}$$

which is linear in the elements of S_2.

- Given values of elements of S_1, it is possible to plug P training data in equation 7.
- As a result we obtain a matrix equation $\mathbf{A}\theta = \mathbf{y}$ where θ is the unknown vector whose elements are parameters in S_2.
- This is the standard linear least-square problem.

Combining LSE and gradient descent - forward pass

- In batch mode, each epoch is composed of a forward pass and a backward pass.
- In the forward pass an input vector is presented and the output is calculated creating a row in the matrices $\mathbf{A}$ and $\mathbf{y}$.
- The process is repeated for all training data and the parameters S_2 are identified by BLS or RLS.
- After S_2 is identified the error for each pair is computed.

Combining LSE and gradient descent - backward pass

- The derivative of the error measure with respect to each node output propagate from the output toward the input.
- The derivatives are:

$$\frac{\partial Ep}{\partial O_{i,p}^{L}} = -2(T_{i,p} - O_{i,p}^{L})$$

$$\frac{\partial E_p}{\partial O_{i,p}^{k}} = \sum_{m=1}^{\#(k+1)} \frac{\partial E_p}{\partial O_{m,p}^{k+1}} \frac{\partial O_{m,p}^{k+1}}{\partial O_{i,p}^{k}}$$

- The parameters in S_2 are updated by the gradient method

$$\Delta\alpha = -\eta \frac{\partial E}{\partial \alpha}$$

Universal Aproximator

ANFIS is a Universal Aproximator

- When the number of rules is not restricted, a zero-order Sugeno model has unlimited approximation power for matching well any nonlinear function arbitrarily on a compact set.
- This can be proved using the Stone-Weierstrass theorem.
- Let domain D be a compact space of N dimensions, and let $\mathcal{F}$ be a set of continuous real-valued functions on D satisfying the following criteria:

Stone-Weierstrauss theorem - I

Indentity function: The constant $f(x) = 1$ is in $\mathcal{F}$.

Separability: For any two points $x_1 \neq x_2$ in D, there is an f in $\mathcal{F}$ such that $f(x_1) \neq f(x_2)$.

Algebraic closure: If f and g are any two functions in $\mathcal{F}$, then fg and $af + bg$ are in $\mathcal{F}$ for any two real numbers a and b.

Stone-Weierstrauss theorem - II

- Then $\mathcal{F}$ is dense on $C(D)$, the set of continuous real-valued functions on D.
- For any $\epsilon > 0$ and any function g in $C(D)$, there is a function f in $\mathcal{F}$ such that $|g(x) - f(x)| < \epsilon$ for all $x \in D$.
- The ANFIS satisfies all these requirements.

Stone-Weierstrauss theorem - III

- In applications of fuzzy inference systems, the domain is almost always compact.
- It is possible, applying this theorem to prove the universal approximation power of the zero-order Sugeno model.

Indentity Function

- **Indentity function:** The constant $f(x) = 1$ is in $\mathcal{F}$.
- The first hypothesis requires that our fuzzy inference system be able to compute the identity function $f(x) = 1$.
- An obvious solution is to set the consequence part of each rule equal to one.
- A fuzzy inference system with only one rule is able to compute the identity function.

Separability

- **Separability:** For any two points $x_1 \neq x_2$ in D, there is an f in $\mathcal{F}$ such that $f(x_1) \neq f(x_2)$.
- The second hypothesis requires that our fuzzy inference system be able to compute functions that have different values for different points.
- This is achievable by any fuzzy inference system with appropriate parameters.

Algebraic Closure - Addition I

- **Algebraic closure addition:** If f and g are any two functions in $\mathcal{F}$, then $af + bg$ are in $\mathcal{F}$ for any two real numbers a and b.
- Suppose that we have two fuzzy inference systems S and $\hat{S}$; each of them has two rules.
- The final output of each system is specified as

$$S : z = \frac{w_1 f_1 + w_2 f_2}{w_1 + w_2}$$

$$\hat{S} : \hat{z} = \frac{\hat{w}_1 f_1 + \hat{w}_2 f_2}{\hat{w}_1 + \hat{w}_2}$$

Algebraic Closure - Addition II

- sum of z and $\hat{z}$ is:

$$
\begin{aligned}
az + b\hat{z} &= a\frac{w_1 f_1 + w_2 f_2}{w_1 + w_2} + b\frac{\hat{w}_1 f_1 + \hat{w}_2 f_2}{\hat{w}_1 + \hat{w}_2} \\
&= \frac{w_1\hat{w}_1(af_1 + b\hat{f}_1) + w_1\hat{w}_2(af_1 + b\hat{f}_2) + w_2\hat{w}_1(af_2 + b\hat{f}_1) + w_2\hat{w}_2(af_2 + b\hat{f}_2)}{w_1\hat{w}_1 + w_1\hat{w}_2 + w_2\hat{w}_1 + w_2\hat{w}_2}
\end{aligned}
$$

Algebraic Closure - Addition III

- Therefore, it is possible to construct a four-rule inference system that computes $az + b\hat{z}$.
- The firing strength of each rule is given by $w_i \hat{w}_j \quad (i, j = 1 \text{ or } 2)$
- The output of each rule is given by $af_i + b\hat{f}_j \quad (i, j = 1 \text{ or } 2)$

Algebraic Closure - Multiplication I

- **Algebraic closure multiplication:** If f and g are any two functions in $\mathcal{F}$, then fg are in $\mathcal{F}$.
- product of z and $\hat{z}$ is:

$$z\hat{z} = \frac{w_1\hat{w}_1 f_1\hat{f}_1 + w_1\hat{w}_2 f_1\hat{f}_2 + w_2\hat{w}_1 f_2\hat{f}_1 + w_2\hat{w}_2 f_2\hat{f}_2}{w_1\hat{w}_1 + w_1\hat{w}_2 + w_2\hat{w}_1 + w_2\hat{w}_2}$$

- Therefore, it is possible to construct a four-rule inference system that computes $z\hat{z}$.
- The firing strength and output of each rule is defined by $w_i\hat{w}_j$ and $f_i\hat{f}_j \quad (i, j = 1 \text{ or } 2)$ respectively.

Conclusion

- ANFIS architectures that compute $z\hat{z}$ and $az + b\hat{z}$ are of the same class as those of S and $\hat{S}$ if and only if the membership functions used are invariant under multiplication.
- The Gaussian membership functions satisfy this property.
- $\mu_{A_i} = k_i e^{[-(\frac{x-c_i}{a_i})^2]}$

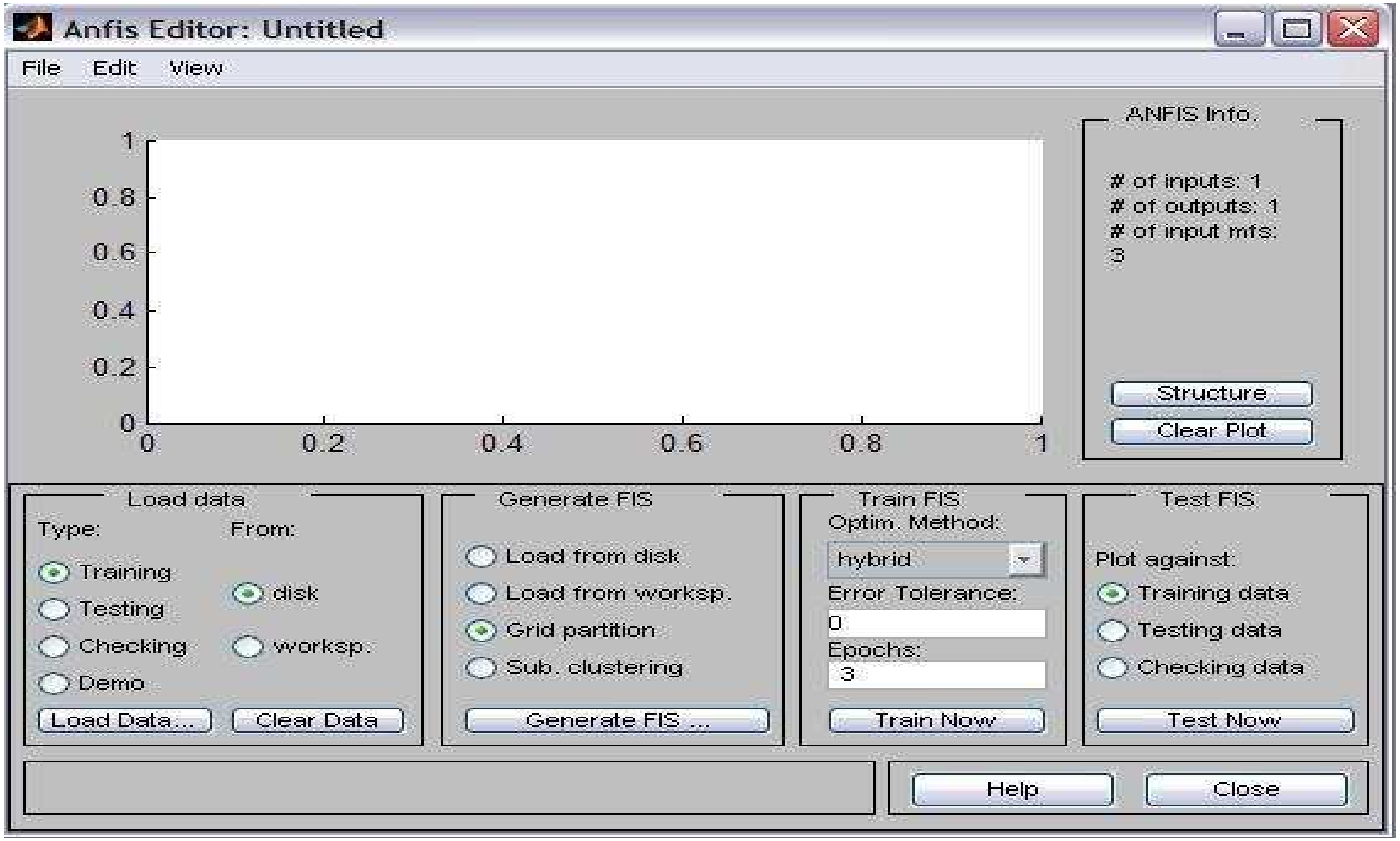
Anfis and Matlab

Matlab

- It is possible to use a graphics user interface
- Command `anfisedit`.

- It is possible to use the command line interface or m-file programs.
- There are functions to generate, train, test and use these systems.

ANFIS gui



Anfis Editor: Untitled
File
Edit
View
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1
ANFIS Info.
# of inputs: 1
# of outputs: 1
# of input mfs:
3
Structure
Clear Plot
Load data
Type:
From:
Training
Testing
Checking
Demo
disk
worksp.
Load Data...
Clear Data
Generate FIS
Load from disk
Load from worksp.
Grid partition
Sub. clustering
Generate FIS ...
Train FIS
Optim. Method:
hybrid
Error Tolerance:
0
Epochs:
3
Train Now
Test FIS
Plot against:
Training data
Testing data
Checking data
Test Now
Help
Close

Applying

- Initializing
- Training
- Testing
- Using

Initializing - GENFIS1 - 1

- `FIS = GENFIS1(DATA)` generates a single-output Sugeno-type fuzzy inference system (`FIS`) using a grid partition on the data (no clustering).
- `FIS` is used to provide initial conditions for posterior ANFIS training.
- `DATA` is a matrix with N+1 columns where the first N columns contain data for each `FIS` input, and the last column contains the output data.

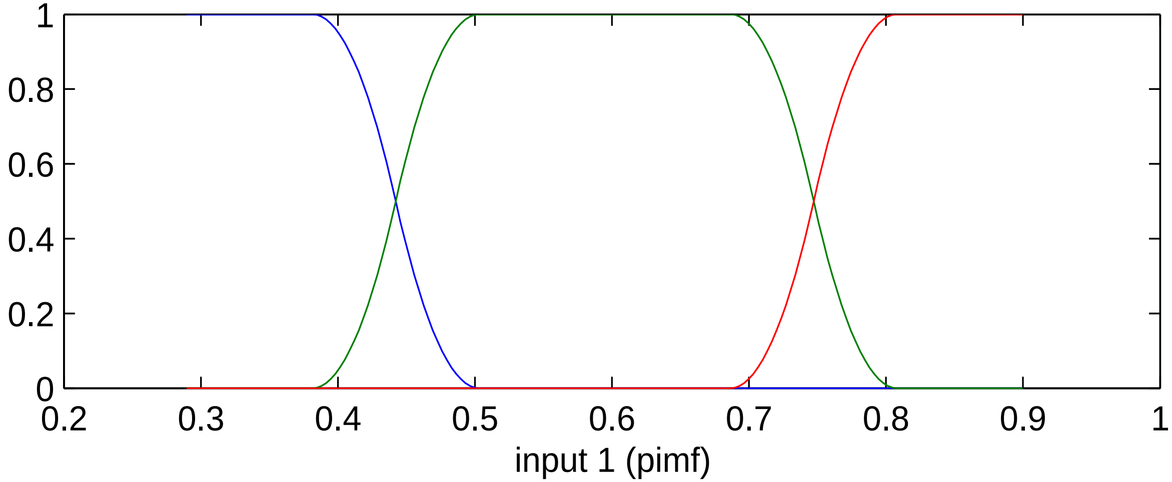
Initializing - GENFIS1 - 2

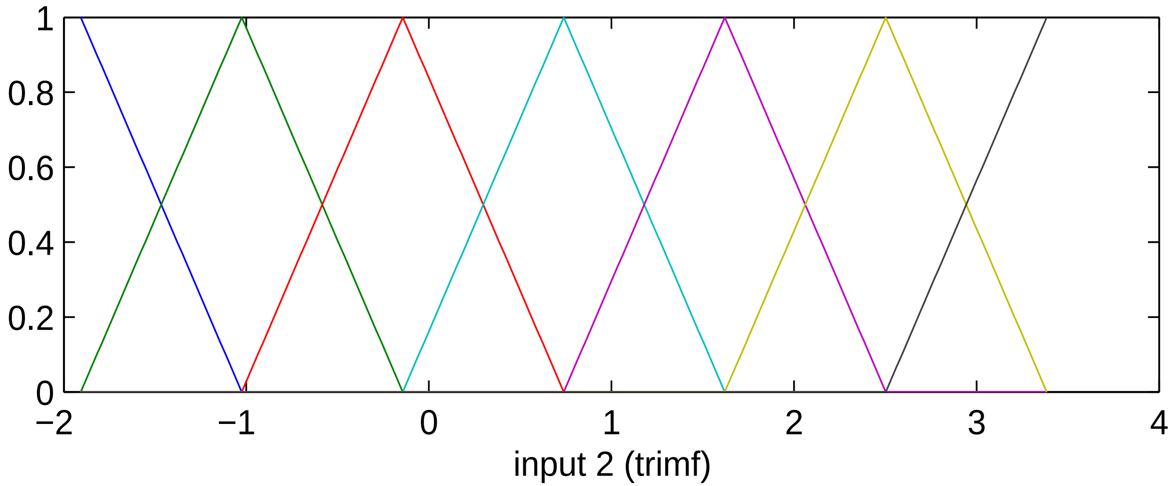
- By default `GENFIS1` uses two `'gbellmf'` type membership functions for each input.
- Each rule generated has one output membership function, which is of type 'linear' by default.
- It is possible to define these parameters using `FIS = GENFIS1(DATA, NUMMFS, INPUTMF, OUTPUTMF)`
- `fis = genfis1(data, [3 7], char('pimf', 'trimf'));`

Initializing - GENFIS1 - 3

```
data = [rand(10,1) 10*rand(10,1)-5 rand(10,1)];
fis = genfis1(data, [3 7], char('pimf','trimf'));
[x,mf] = plotmf(fis,'input',1);
subplot(2,1,1), plot(x,mf);
xlabel('input 1 (pimf)');
[x,mf] = plotmf(fis,'input',2);
subplot(2,1,2), plot(x,mf);
xlabel('input 2 (trimf)');
```

Initializing - GENFIS1 - 4





Initializing - GENFIS2

- `GENFIS2` generates a Sugeno-type FIS using subtractive clustering.
- `GENFIS2` extracts a set of rules that models the data behavior.
- The rule extraction method first determines the number of rules and antecedent membership functions and then uses linear least squares estimation to determine each rule's consequent equations.

Training

- `ANFIS` uses a hybrid learning algorithm to identify the membership function parameters of single-output, Sugeno type fuzzy inference systems (FIS).
- There are many ways of using this function.
- Some examples:
  - `[FIS,ERROR] = ANFIS(TRNDATA)`
  - `[FIS,ERROR] = ANFIS(TRNDATA,INITFIS)`

Using

- `EVALFIS` evaluates a FIS.
- `Y = EVALFIS(U,FIS)` simulates the Fuzzy Inference System FIS for the input data U and returns the output data Y.

Example

- run exemplo06_03.m

The End