G53FUZ
Fuzzy Sets and Systems

ANFIS
Adaptive Neuro-Fuzzy Inference System

Bob John
ASAP
Research Group

Overview

- The ANFIS model
  - model structure
  - description of layers
  - model tuning
- Worked example
  - fully automatic model determination
  - ANFIS tuning of manual model

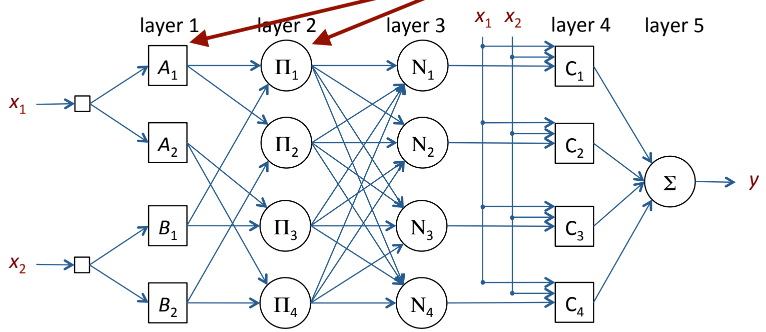
ANFIS Model

Background

- Introduced by J-S Roger Jang in 1993
  - "ANFIS: Adaptive-Network-Based Fuzzy Inference System", *IEEE Transactions on Systems, Man and Cybernetics*, 23(3), 665-685, 1993
- Often (usually?) referred to as
  - Adaptive **Neuro**-Fuzzy Inference System
  - ANFIS
- TSK system viewed as a network of nodes
  - similar to a neural-network

ANFIS

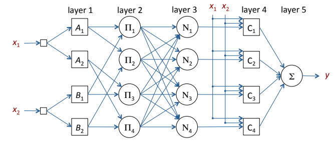
square nodes have parameters (that may be altered) round nodes do not



First-order Sugeno fuzzy model: 2 inputs (2 mfs), 4 rules

Rule 1: IF x_1 is A_1 AND x_2 is B_1 THEN $y_1 = p_1x_1 + q_1x_2 + r_1$
Rule 2: IF x_1 is A_1 AND x_2 is B_2 THEN $y_2 = p_2x_1 + q_2x_2 + r_2$
Rule 3: IF x_1 is A_2 AND x_2 is B_1 THEN $y_3 = p_3x_1 + q_3x_2 + r_3$
Rule 4: IF x_1 is A_2 AND x_2 is B_2 THEN $y_4 = p_4x_1 + q_4x_2 + r_4$

Layer 1 – Inputs



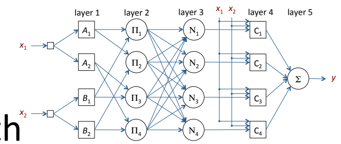
- Premise parameters

$$O_{1,i} = \mu_{A_i}(x_1) \quad \textit{for } i = 1,2$$
$$O_{1,i} = \mu_{B_{i-2}}(x_2) \quad \textit{for } i = 3,4$$

- Often, m.f.s are Gaussians

$$\mu_{A_i}(x) = e^{-\frac{(x-c_i)^2}{a_i^2}}$$

Layer 2 – Rule Firing



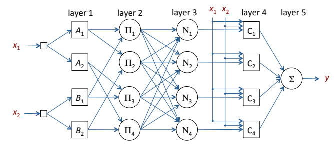
- T-norm operator combining the separate antecedents into single rule firing strength
  - using the product t-norm

$$O_{2,i} = w_i = \mu_{A_i}(x_1)\mu_{B_i}(x_2) \quad i = 1,2$$

  - in general, for n inputs (each with m m.f.s), we have $R = m^n$ rules

$$O_{2,i} = \prod_{j=1}^{n} \mu(x_j) = \mu(x_1).\mu(x_2)\cdots\mu(x_n) \quad \text{for } i = 1\ldots R$$

Layer 3 – Normalised Firing

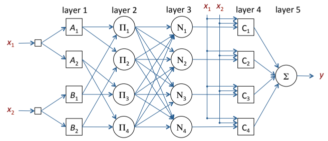


- Outputs of layer 3 are the normalised rule firing strengths

$$O_{3,i} = \frac{w_i}{\sum_{j=1}^{r} w_j} \qquad \text{for } i = 1\ldots R$$

Layer 4 – Rule Consequents

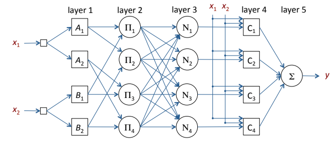
- Consequent parameters (towards defuzzification)



$$O_{4,i} = \overline{w}_i y_i = \overline{w}_i (p_i x_1 + q_i x_2 + r_i)$$

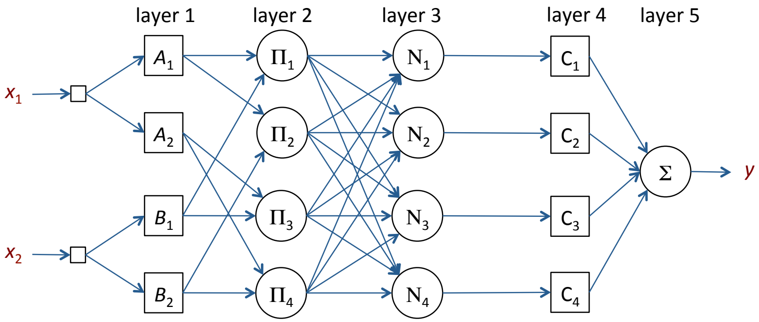
Layer 5 – Defuzzification

- Final crisp output
  - sums each normalised rule output



$$y = O_{5,i} = \sum_i \overline{w}_i y_i = \frac{\sum_i w_i y_i}{\sum_i w_i}$$

ANFIS – Zeroth Order



Zeroth-order Sugeno fuzzy model: 2 inputs (2 mfs), 4 rules

Rule 1: IF x_1 is A_1 AND x_2 is B_1 THEN $y_1 = r_1$
Rule 2: IF x_1 is A_1 AND x_2 is B_2 THEN $y_2 = r_2$
Rule 3: IF x_1 is A_2 AND x_2 is B_1 THEN $y_3 = r_3$
Rule 4: IF x_1 is A_2 AND x_2 is B_2 THEN $y_4 = r_4$

Parameters

- ANFIS features two sets of parameters
  - S_1
    - the parameters that define the input m.f.s
    - non-linear
    - modified by using a gradient descent tuning algorithm in a backward pass (back-propagation of errors)
  - S_2
    - the parameters that define the consequent functions
    - linear
    - modified by using an iterative least squares estimation algorithm in a forward pass

ANFIS Learning

Two passes in the hybrid learning procedure

	Forward Pass	Backward Pass
Premise Parameters (nonlinear)	Fixed	Gradient descent
Consequent parameters (linear)	Least-square estimator	Fixed
Signals	Node outputs	Error signals

LSE Forward Pass

- LSE used to minimise the squared error

$$||AX - B||^2$$

  - where
    - A is a matrix of outputs produced by layer 3
    - B is a matrix of target outputs
    - X is a matrix of consequent parameters
- A sequential (iterative) estimation algorithm estimates *X*, essentially as per a Kalman filter
  - run over each of the *P* training instances

Back Propagation

- The overall error measure is

$$E = \sqrt{\frac{\sum_{p=1}^{P}(T_p - Y_p)^2}{P}}$$

  - where Y_p are actual outputs and T_p are targets
- For each parameter α_i, the update formula is

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

  - where η is the learning rate, $\eta = \frac{k}{\sqrt{\sum_i \left(\frac{\partial E}{\partial \alpha_i}\right)^2}}$

Example

- Jang *et al.*, ***Neuro-Fuzzy and Soft Computing***, Prentice Hall, 1997
- ANFIS is used to model a two-dimensional *sinc* equation defined by

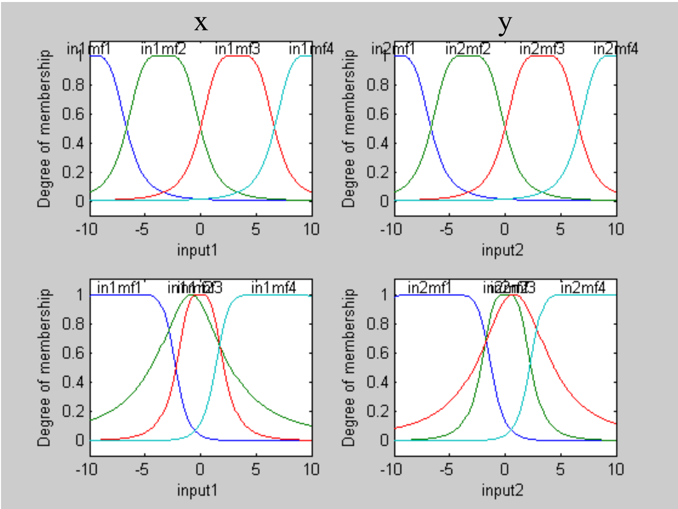
$$z = \sin c(x, y) = \frac{\sin(x)\sin(y)}{xy}$$

  - *x* and *y* are in the range [-10,10]
  - number of m.f.s for each input: 4
  - number of rules: 16

Result

Initial membership functions

Final (trained) membership functions after 100 epochs



ANFIS Example

Overview

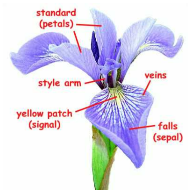
Classification

iris setosa

iris versicolor

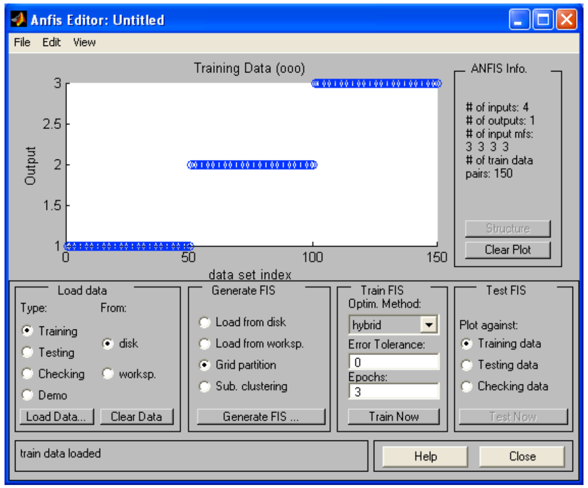
iris virginica

- Can we identify the type of iris from measurements of its petals and sepals?

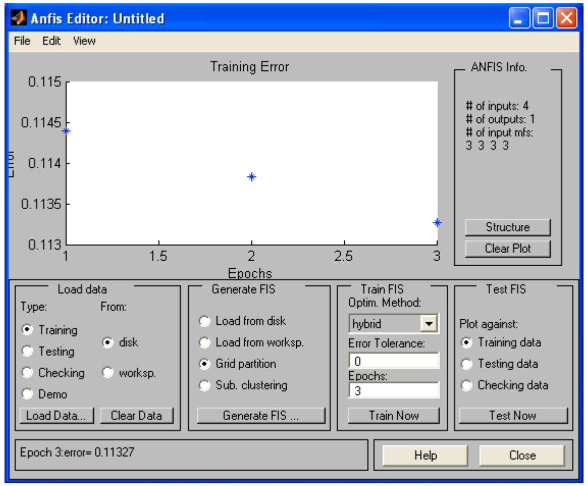


	Sepal length	Sepal width	Petal length	Petal width	Species
1	5.8	4.0	1.2	0.2	I. setosa
2	5.7	4.4	1.5	0.4	I. setosa
3	5.7	3.8	1.7	0.3	I. setosa
			…		
51	7.0	3.2	4.7	1.4	I. versicolor
52	6.9	3.1	4.9	1.5	I. versicolor
53	6.8	2.8	4.8	1.4	I. versicolor
			…		
101	7.9	3.8	6.4	2.0	I. virginica
102	7.7	3.8	6.7	2.2	I. virginica
103	7.7	2.6	6.9	2.3	I. virginica
			…		
150	4.9	2.5	4.5	1.7	I. virginica

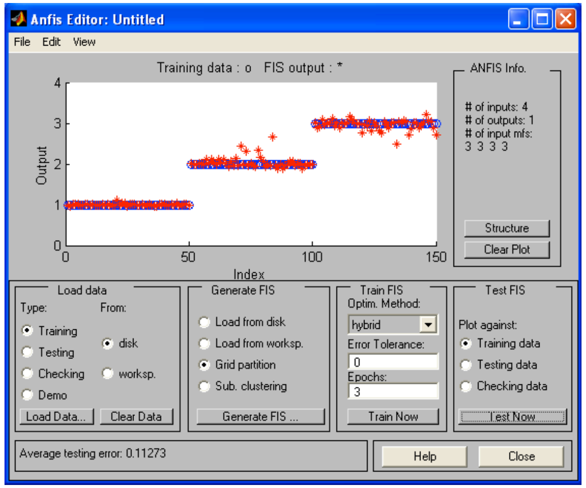
Iris Data (3 Species)



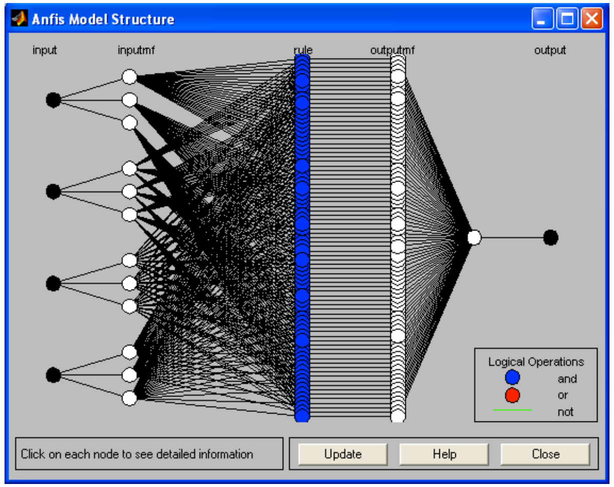
Grid Partition Train



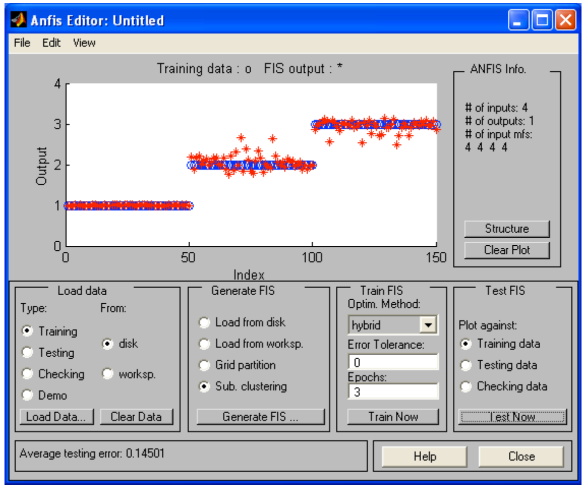
GP Test



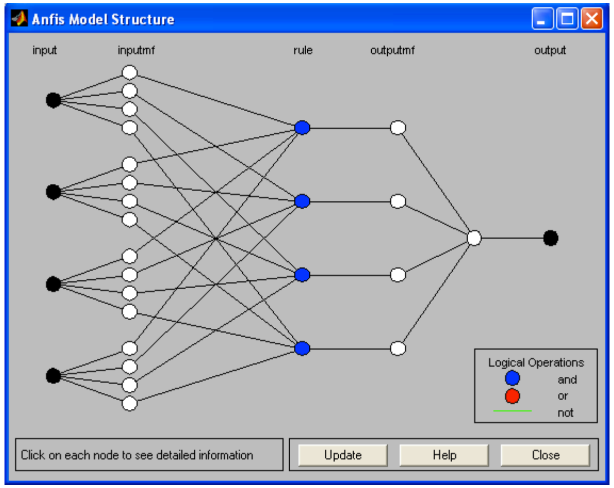
GP Structure!



Subtractive Clustering



SC Structure

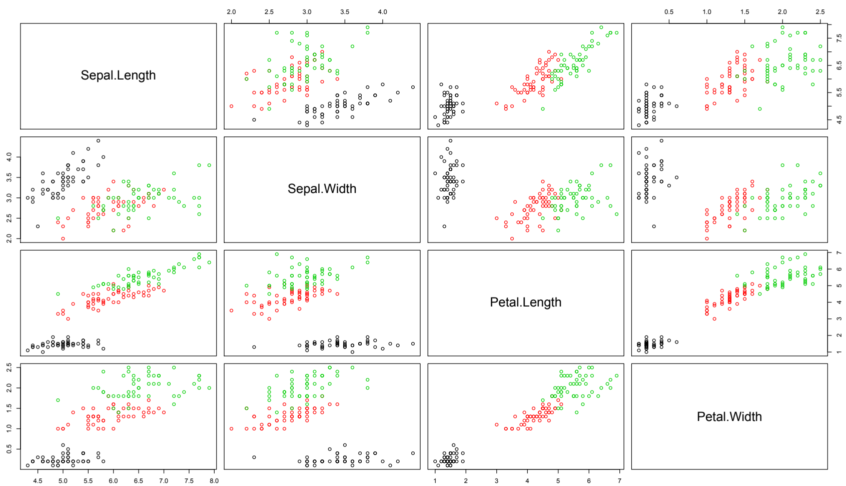


SC Structure Better?

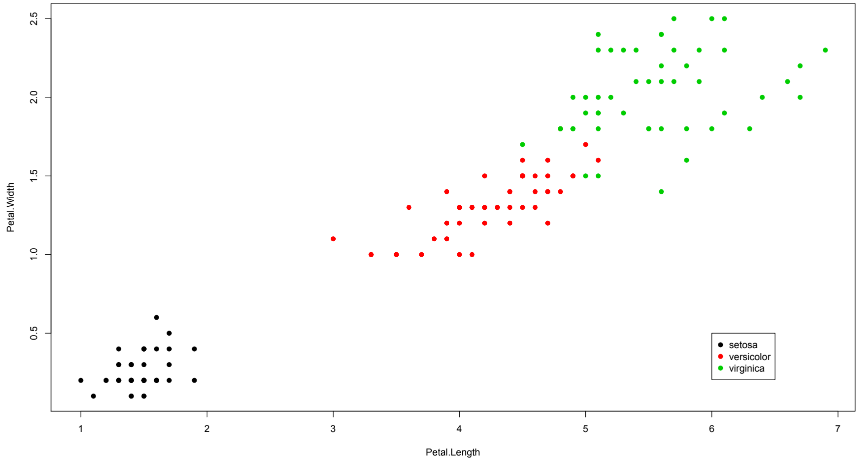
- This is now 'only' a 4-rule 1st-order TSK FIS
  1. If *in1* is *i1mf1* and *in2* is *i2mf1* and *in3* is *i3mf1* and *in4* is *i4mf1* then *out1* is *o1mf1*
  2. If *in1* is *i1mf2* and *in2* is *i2mf2* and *in3* is *i3mf2* and *in4* is *i4mf2* then *out1* is *o1mf2*
  3. If *in1* is *i1mf3* and *in2* is *i2mf3* and *in3* is *i3mf3* and *in4* is *i4mf3* then *out1* is *o1mf3*
  4. If *in1* is *i1mf4* and *in2* is *i2mf4* and *in3* is *i3mf4* and in4 is *i4mf4* then *out1* is *o1mf4*
- But each outmf is a linear combination, e.g.
  - *outmf4* =
    0.851873*in1* + 1.659586*in2* +
    1.352931*in3* - 5.703523*in4* - 8.063416
- Still, far from comprehensible?

Manual Equivalent

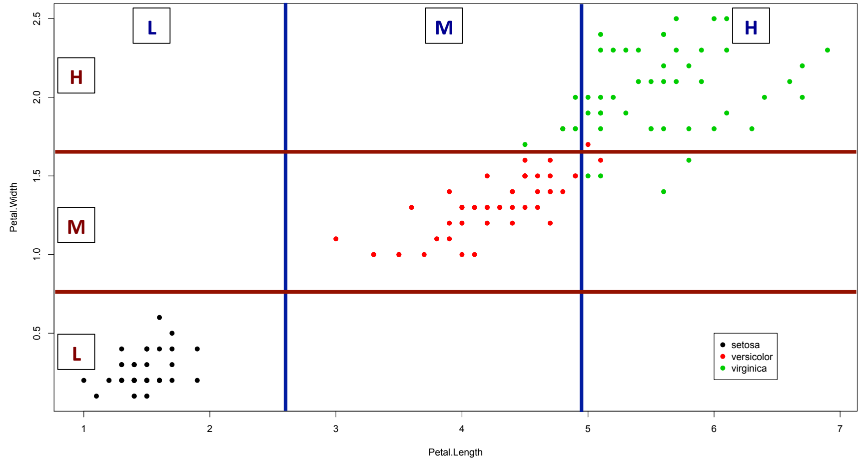
IRIS Pairs



Petal Length and Width



Petal Length and Width

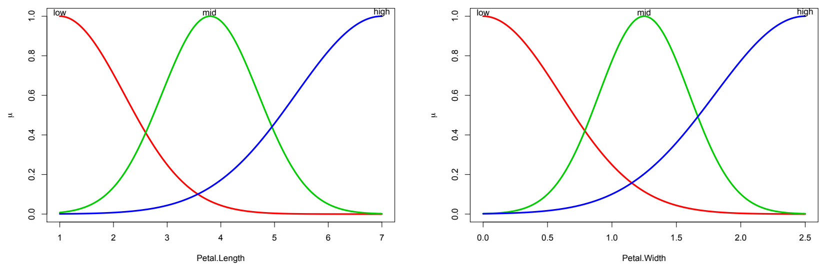


Classification Rules

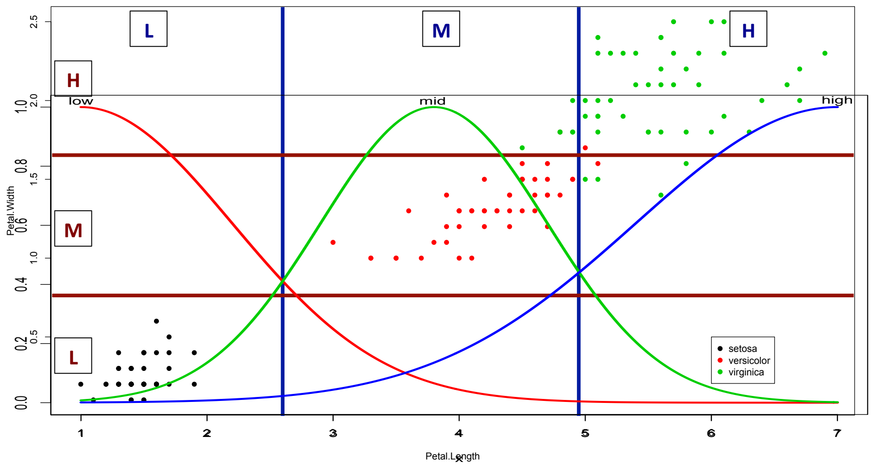
- IF *Petal.Length* is *Low* and *Petal.Width* is *Low* THEN *Species* is *setosa*
- IF *Petal.Length* is *Mid* and *Petal.Width* is *Mid* THEN *Species* is *versicolor*
- IF *Petal.Length* is *High* and *Petal.Width* is *High* THEN *Species* is *virginica*

Hand-Crafted FIS

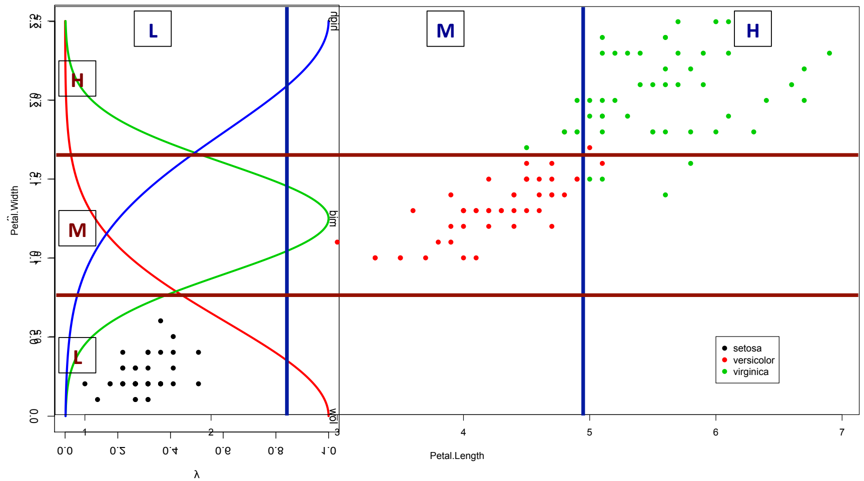
- Four input variables, one output
  – only two inputs needed
  – petal.length and petal.width m.f.s manually tuned to match desired partitions in input variables



Petal Length and Width



Petal Length and Width



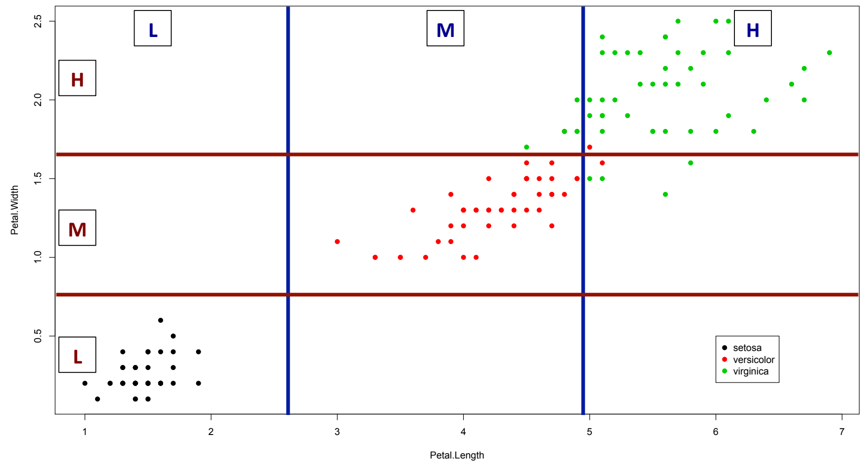
Hand-crafted FIS Performance

- Cross-tabulation

	1	2	3
setosa	50	0	0
versicolor	0	47	3
virginica	0	2	48

- RMSE
  - 0.183

Petal Length and Width



Classification Rules

- IF *Petal.Length* is *Low* and *Petal.Width* is *Low* THEN *Species* is *setosa*
- IF *Petal.Length* is *Mid* and *Petal.Width* is *Mid* THEN *Species* is *versicolor*
- IF *Petal.Length* is *High* and *Petal.Width* is *High* THEN *Species* is *virginica*
- IF *Petal.Length* is *Mid* and *Petal.Width* is *High* THEN *Species* is *virginica*
- IF *Petal.Length* is *High* and *Petal.Width* is *Mid* THEN *Species* is *virginica*

5-Rule FIS Performance

- Cross-tabulation

	1	2	3
setosa	50	0	0
versicolor	0	47	3
virginica	0	0	50

- RMSE
  - 0.141

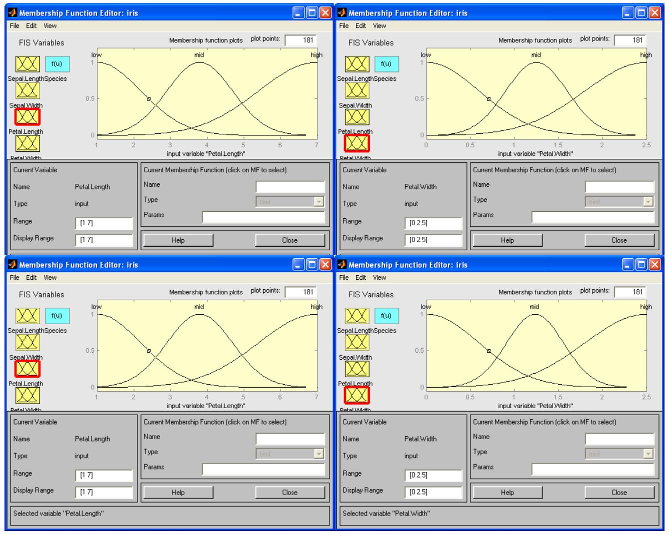
ANFIS Tuning of Manual System

Process

- We can start with a manually created TSK structure and use ANFIS to tune mfs (only)
  - take the previous system and convert to 0^{th}-order TSK with constant outputs (1, 2, 3)
- Process
  - load TSK FIS into ANFIS
  - load training/testing data
  - train and test!

Result

initial
m.f.s



final
m.f.s

Summary

- Lecture summary
  - ANFIS provides a completely automated way of constructing and tuning fuzzy inference systems
    - often producing very good performance (RMSE)
  - automatic ANFIS models are not necessarily either parsimonious or easily interpretable
    - ANFIS can be used to tune a manual FIS
  - TRY: experiment with Iris data in MATLAB-ANFIS
- Next lecture
  - real-world applications