NEURAL NETWORKS – PART II:
Self-Organizing Maps / Introduction to Neuro Fuzzy Systems

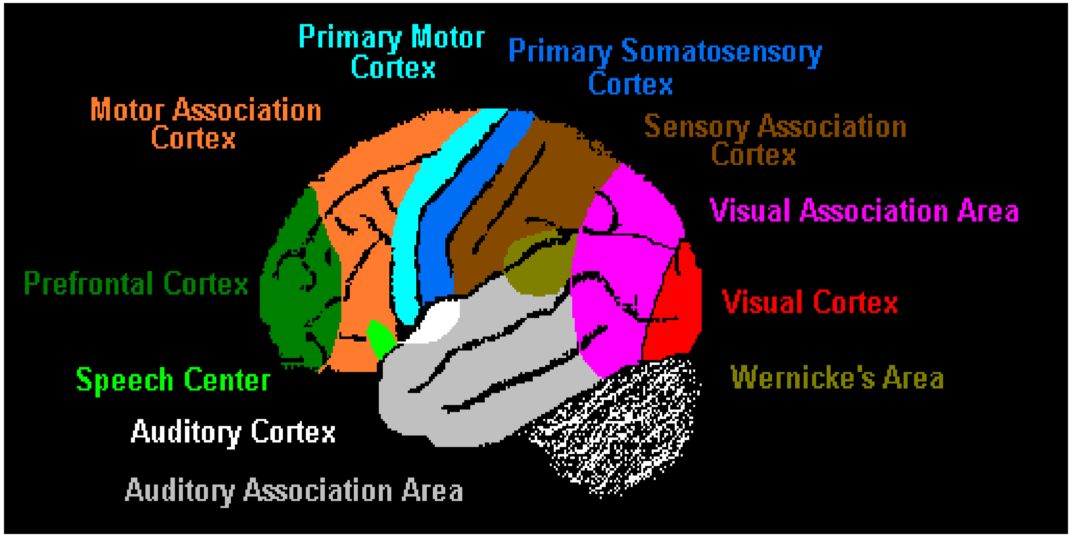
Lecture 5

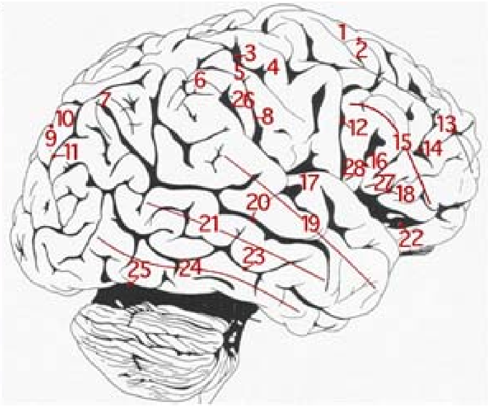
Dr. Vassilis S. Kodogiannis
Reader in Computational Intelligence
Email: V.Kodogiannis@westminster.ac.uk
https://scholar.google.co.uk/citations?user=meTTcLAAAAAJ&hl=en&oi=ao

Unsupervised Learning

- Neural networks for unsupervised learning attempt to discover interesting structure in the available data.
  - There is no information about the desired class (or output) of an example.
- Self Organizing Maps (SOM) combine a competitive learning principle with a topological structuring of neurons.

SOM motivated by human brain

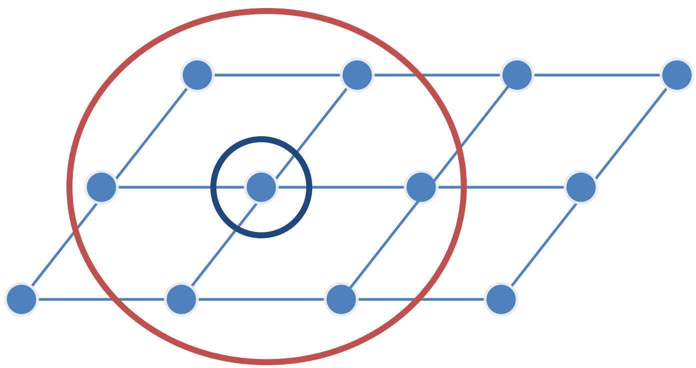




- Brain is organized such a way that different sensory data is represented by topologically ordered computational maps
  - tactile, visual, acoustic sensory input are mapped onto areas of cerebral cortex in topologically ordered manner
  - building block of information processing infrastructure of nervous system

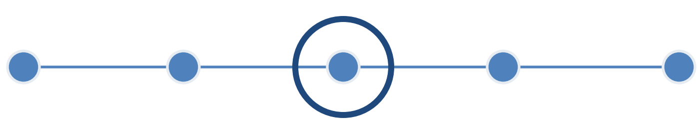
SOM ARCHITECTURE

- The input is connected with each neuron of a lattice.
- The topology of the lattice allows one to define a neighborhood structure on the neurons, like those illustrated below.



2-dimensional topology

and two possible neighborhoods



1-dimensional topology

with a small neighborhood

The idea

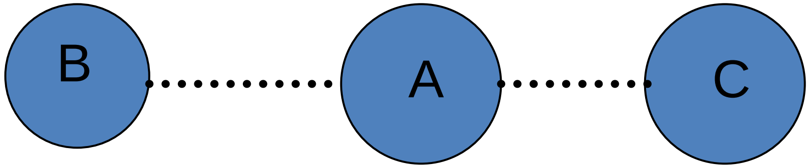
- Upon repeated presentations of the training examples, the weight vectors of the neurons tend to follow the distribution of the examples.
- This results in a topological ordering of the neurons, where neurons adjacent to each other tend to have similar weight vectors.
- The input space of patterns is mapped onto a discrete output space of neurons.

The approach

- One has to find values for the weight vectors of the links from the input layer to the nodes of the lattice, in such a way that adjacent neurons will have similar weight vectors.
- For an input, the **output** of a SOM neural network will be the neuron with weight vector most similar (with respect to Euclidean distance) to that input.
- **Each neuron** can be seen as representing the **cluster** containing all the input examples which are mapped to that neuron by the SOM NN.

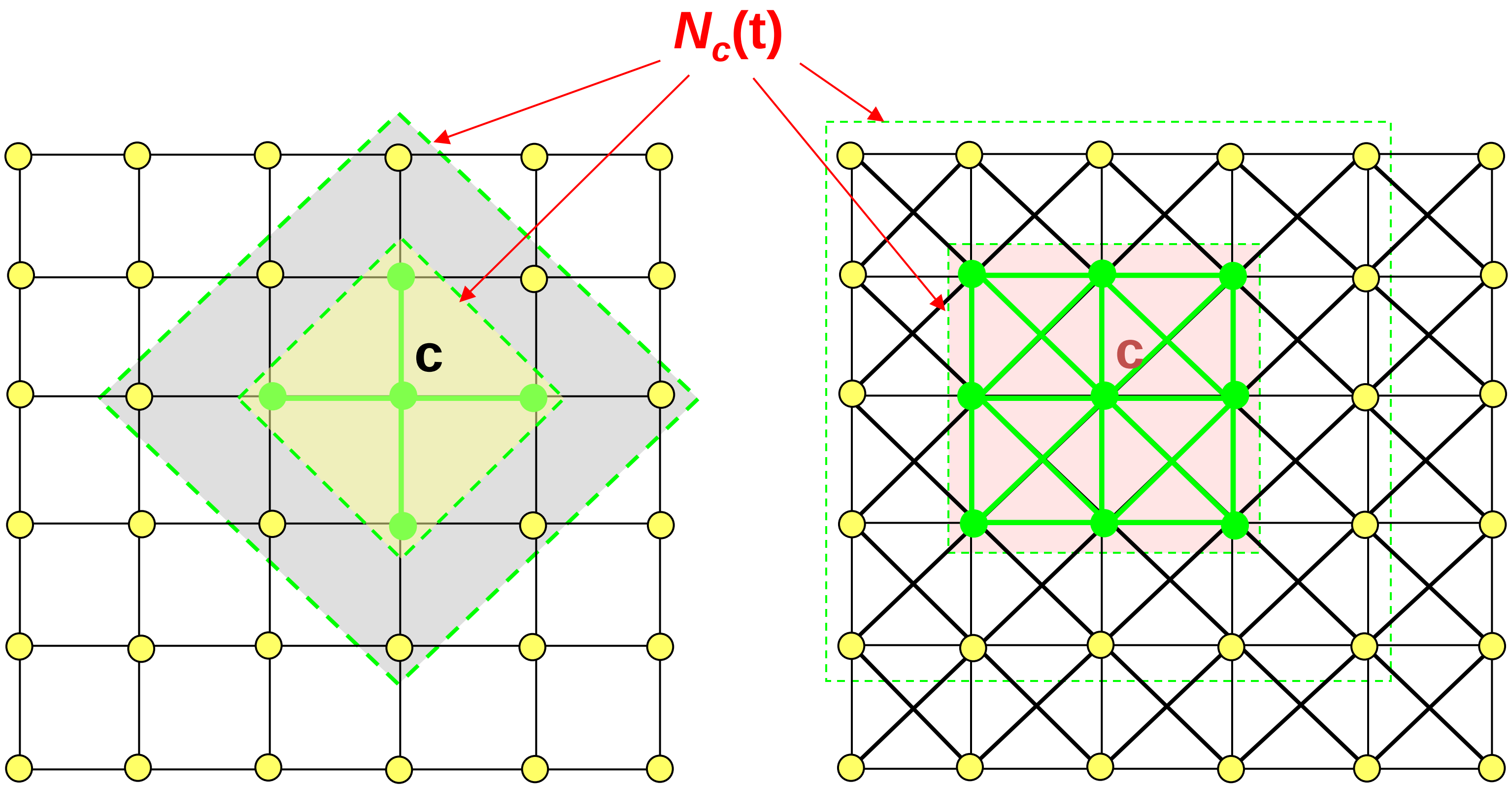
Output Layer Topology

- Often view output in spatial manner
  - E.g., a 1D or 2D arrangement
- 1D arrangement
  - Topology defines which output layer units are neighbors with which others
  - Have a function, D(*t*), which gives output unit neighborhood as a function of time (iterations) of the training algorithm
- E.g., 3 output units



D(*t*) = 1: update weight B & A if input maps onto B

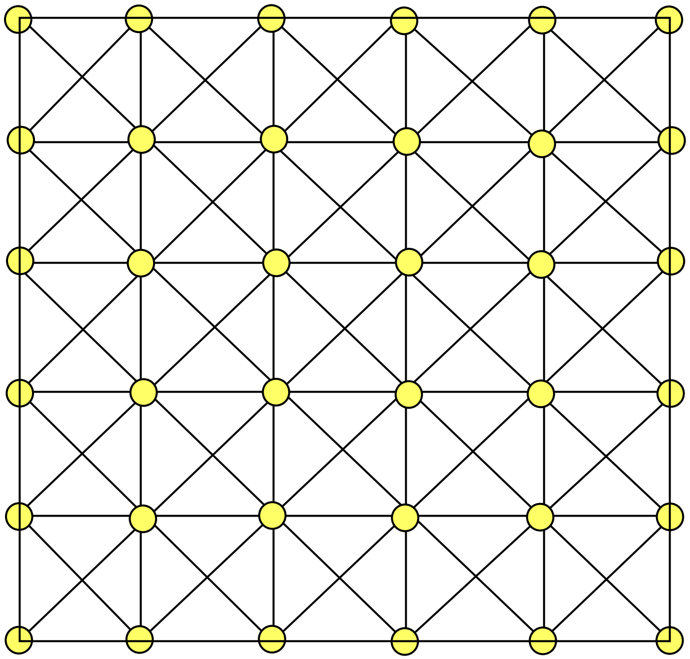
Topological Neighborhoods



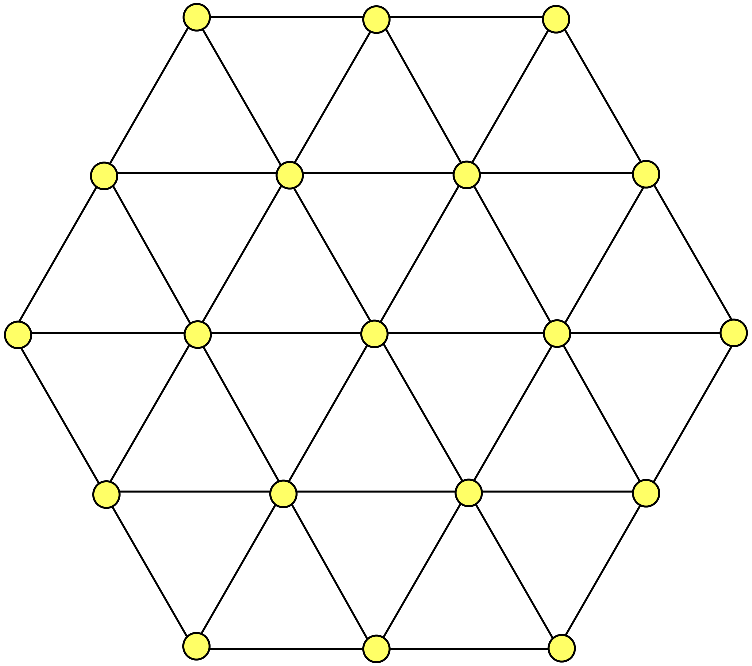
ROMBIC
NEIGHBORHOODS

SQUARE
NEIGHBORHOODS

MOST COMMON 2-D NEURON LATTICES (TOPOLOGICAL STRUCTURES)



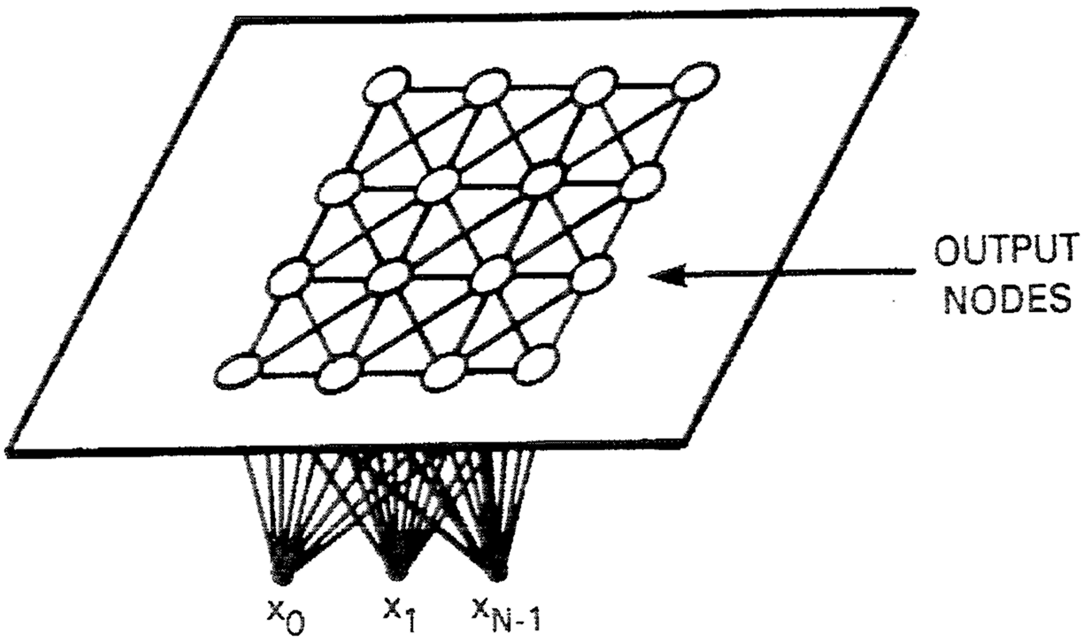
RECTANGULAR LATTICE



HEXAGONAL LATTICE

Kohonen Model

- Captures essential features of computational maps in Brain
  - capable of dimensionality reduction



Neighborhood Function

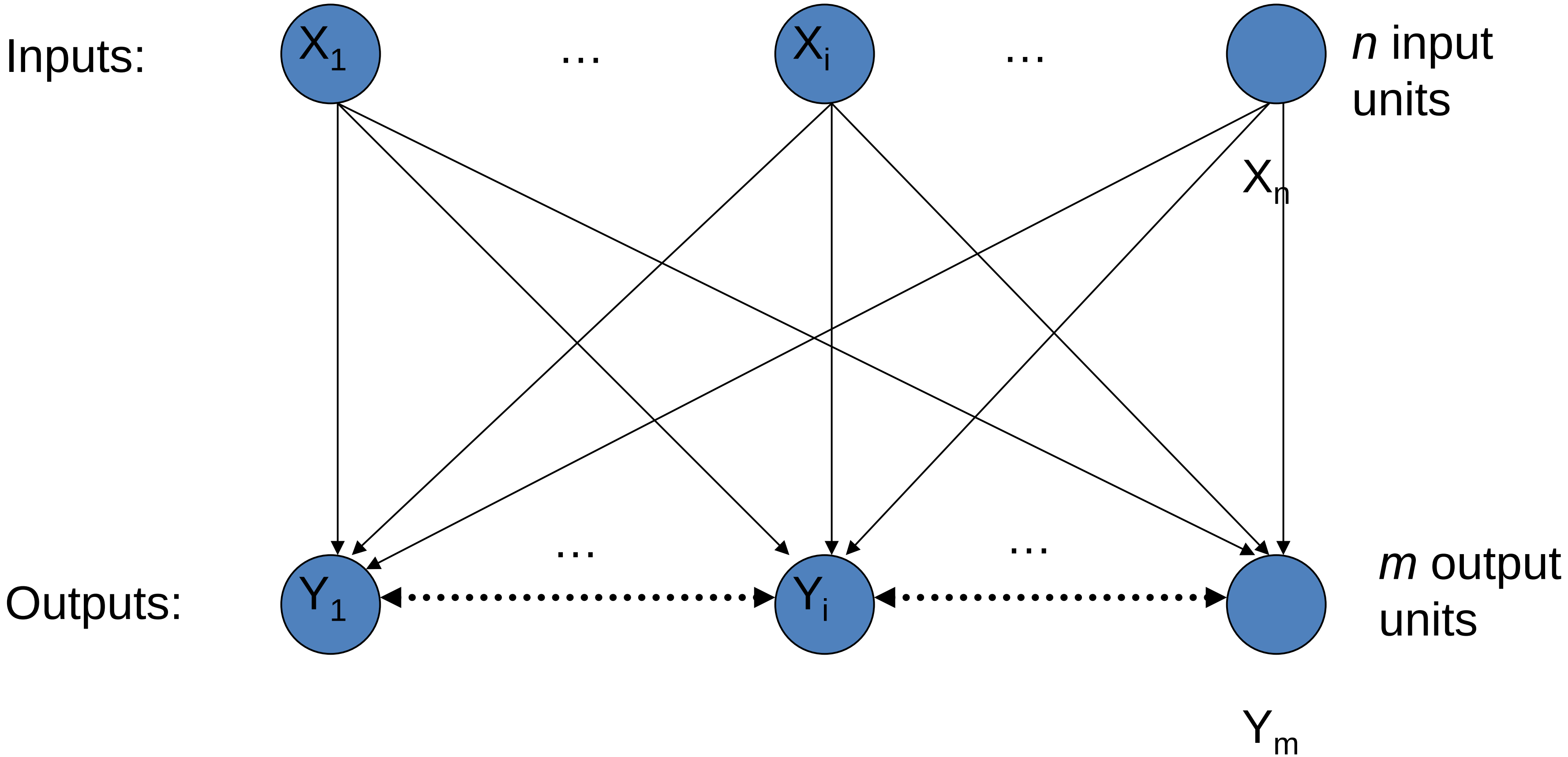
- σ measures the degree to which excited neurons in the vicinity of the winning neuron cooperate in the learning process.
- In the learning algorithm σ is updated at each iteration during the ordering phase using the following exponential decay update rule, with parameters σ_0 and T_1

$$\sigma(n) = \sigma_0 \exp\left(-\frac{n}{T_1}\right)$$

Network Architecture

- Two layers of units
  - ➢ Input: *n* units (length of training vectors)
  - ➢ Output: *m* units (number of categories)
- Input units fully connected with weights to output units
- Intra-layer ("lateral") connections
  - ➢ Within output layer
  - ➢ Defined according to some topology
  - ➢ No weights between these connections, but used in algorithm for updating weights

Network Architecture



Note: There is one weight vector of length *n* associated with each output unit

Overall SOM Algorithm

- Initially
  - Select number of inputs & outputs
  - Select output layer topology
  - Initialize weights
- Training
  - Train weights connecting inputs to outputs
  - Topology is used, in conjunction with current mapping of inputs to outputs, to define which weights updated
  - Distance measure using the topology is reduced over time; reduces the number of weights that get updated per iteration
  - Learning rate is reduced over time
- Testing
  - Use weights from training

SOM Algorithm

- Select output layer network topology
  - Initialize current neighborhood distance, D(1), to a positive value
- Initialize weights from inputs to outputs to small random values
- Let t = 1
- While computational bounds are not exceeded do

  1) Select an input sample i_l

  2) Compute the square of the Euclidean distance of i_l from weight vectors (w_j) associated with each output node

$$\sum_{k=1}^{n} (i_{l,k} - w_{j,k}(t))^2$$

  3) Select output node *j** that has weight vector with minimum value from step 2)

  4) Update weights to all nodes within a topological distance given by D(t) from *j**, using the weight update rule:

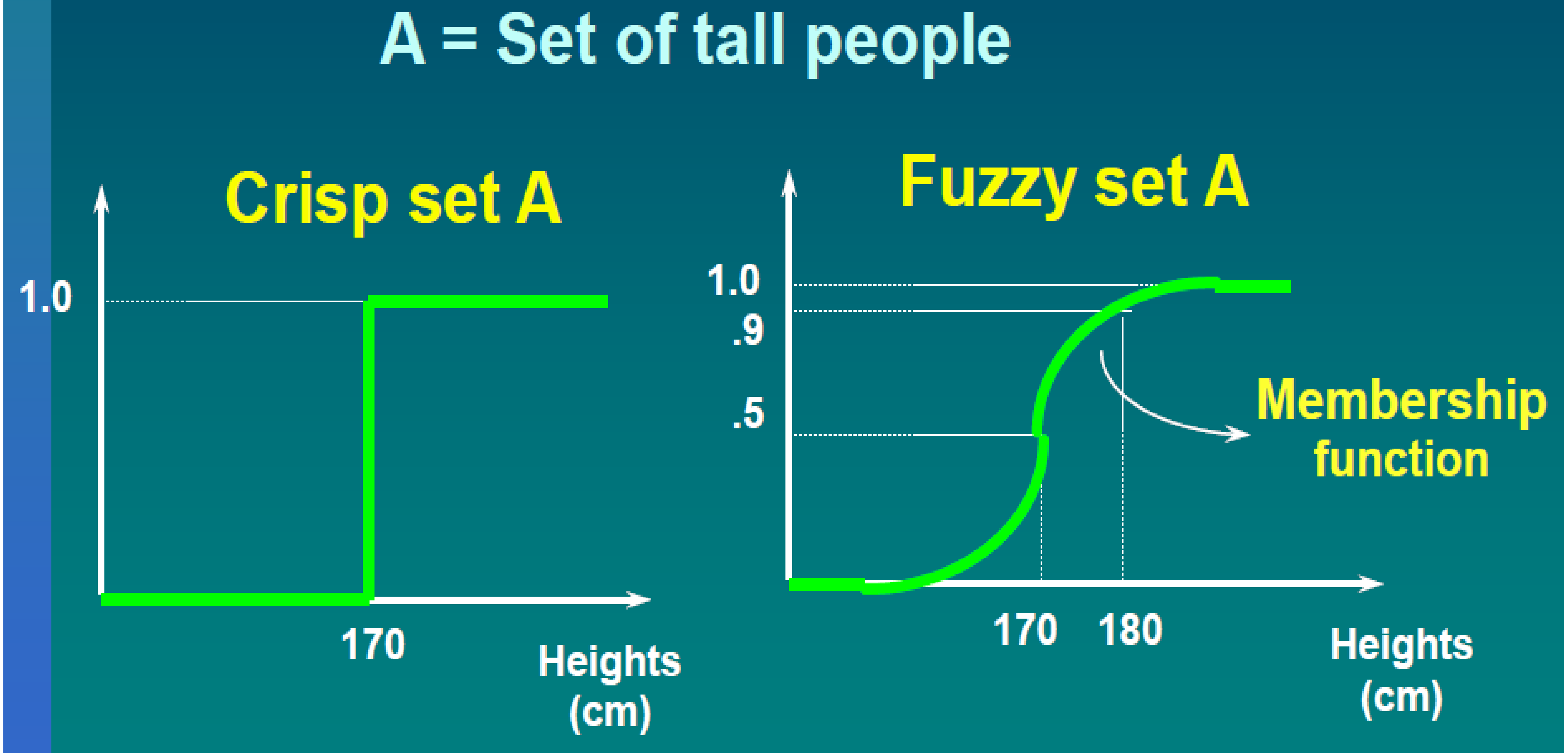
$$w_j(t+1) = w_j(t) + \eta(t)(i_l - w_j(t))$$

  5) Increment t
- End while

Learning rate generally decreases with time: $0 < \eta(t) \le \eta(t-1) \le 1$

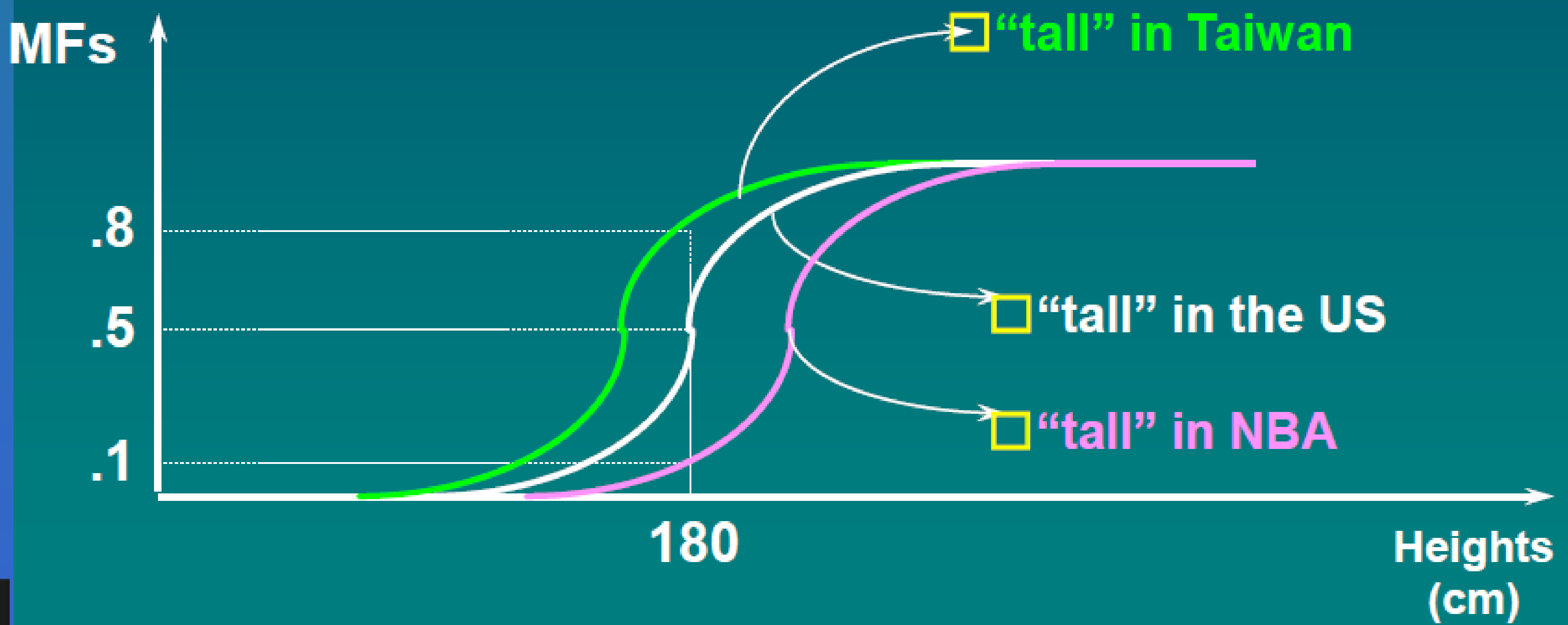
Fizzy Sets

- Sets with fuzzy boundaries



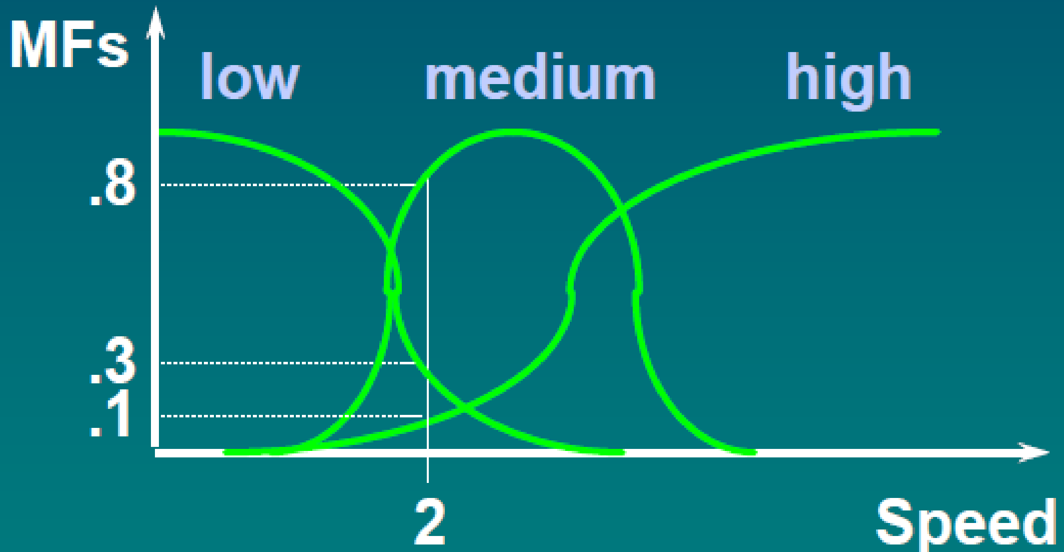
Membership Functions (MFs)

- About MFs
  - Subjective measures
  - Not probability functions



Fuzzy Inference System (FIS)

If speed is low then resistance = 2
If speed is medium then resistance = 4*speed
If speed is high then resistance = 8*speed



Rule 1: w1 = .3; r1 = 2
Rule 2: w2 = .8; r2 = 4*2
Rule 3: w3 = .1; r3 = 8*2

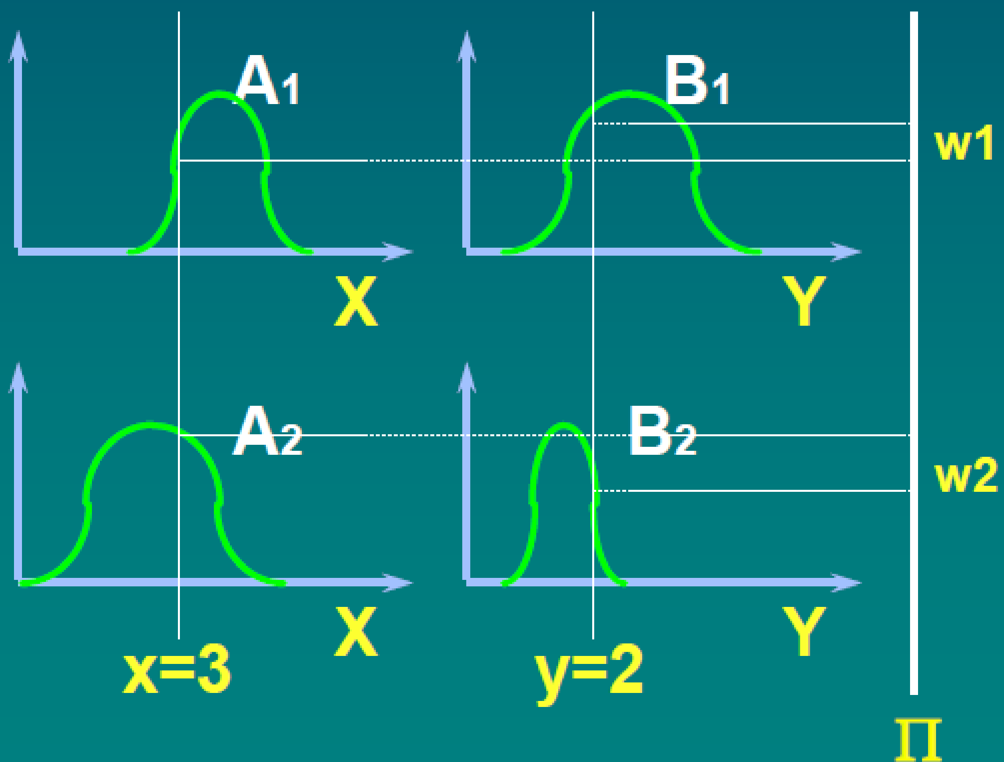
Resistance = Σ(wi*ri) / Σwi
= 7.12

- **Rule base**

  If X is A_1 and Y is B_1 then $Z = p_1*x + q_1*y + r_1$

  If X is A_2 and Y is B_2 then $Z = p_2*x + q_2*y + r_2$

- **Fuzzy reasoning**

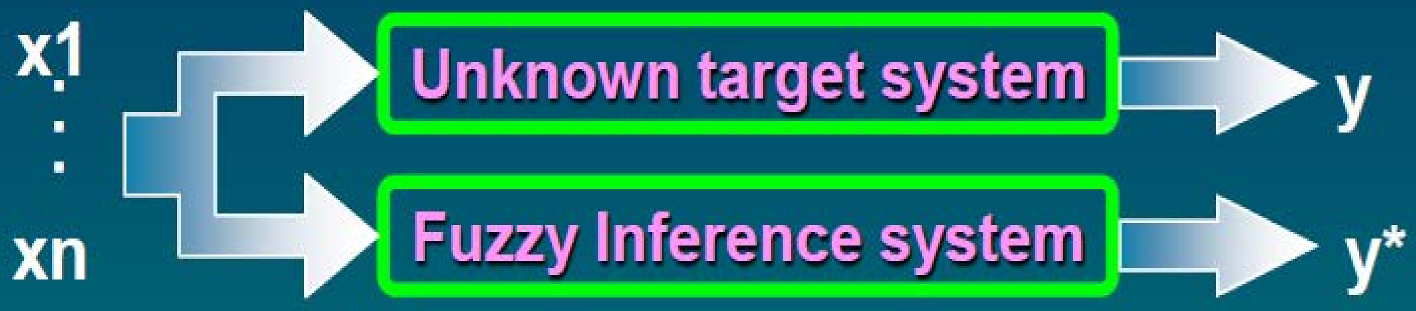


$z_1 = p_1*x+q_1*y+r_1$

$z_2 = p_2*x+q_2*y+r_2$

$$z = \frac{w_1*z_1+w_2*z_2}{w_1+w_2}$$

Fuzzy Modeling



- Given desired i/o pairs (training data set) of the form (x1, ..., xn; y), construct a FIS to match the i/o pairs

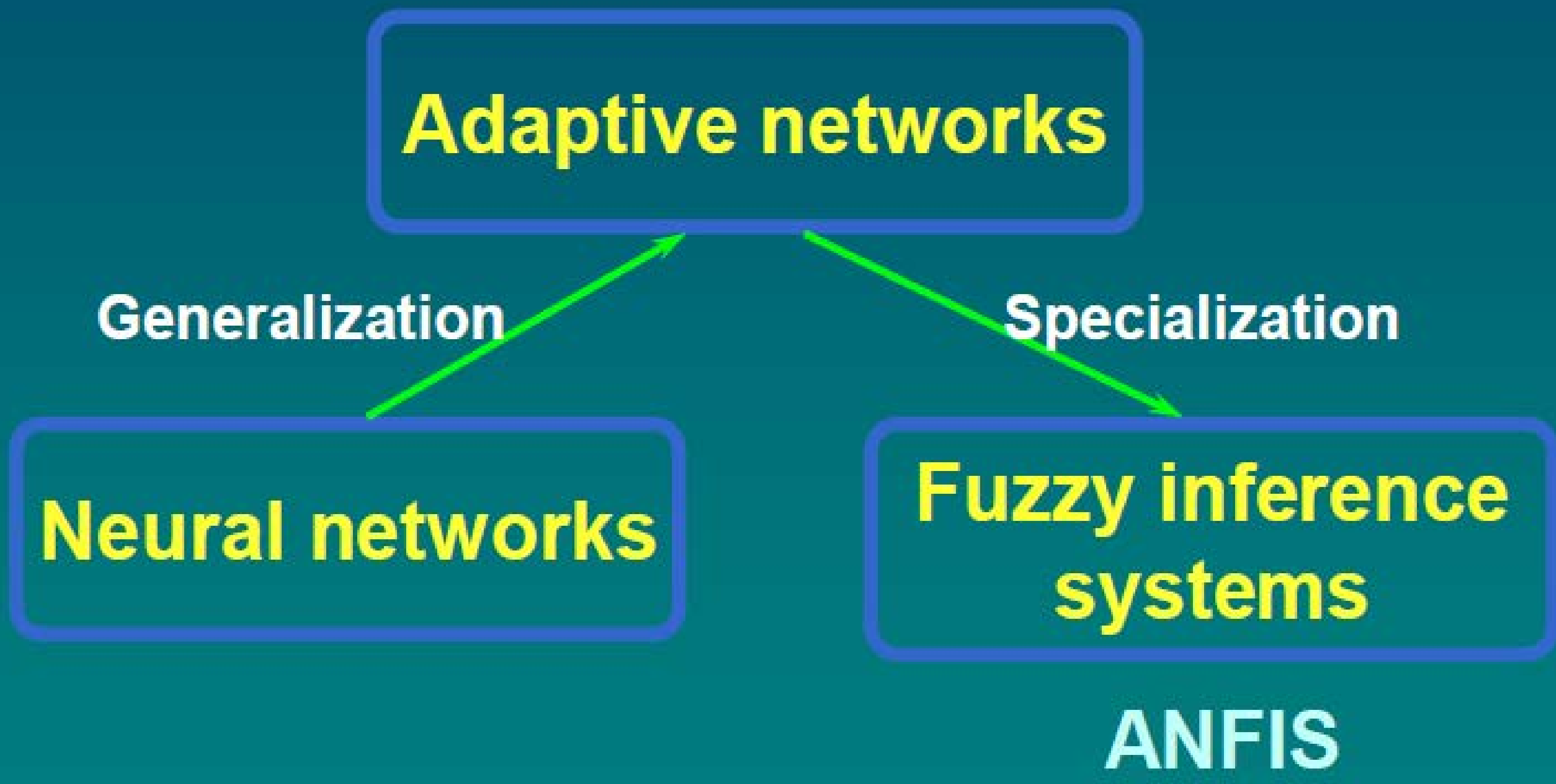
- Two steps in fuzzy modeling

  structure identification --- input selection, MF numbers

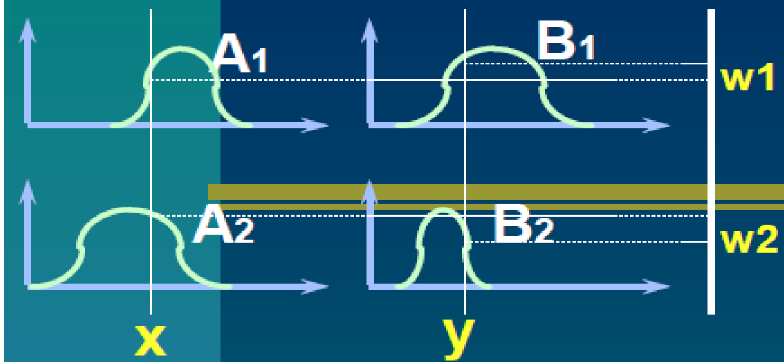
  parameter identification --- optimal parameters

Neuro-Fuzzy Modeling

- Basic approach of ANFIS



•Sugeno Fuzzy Inference

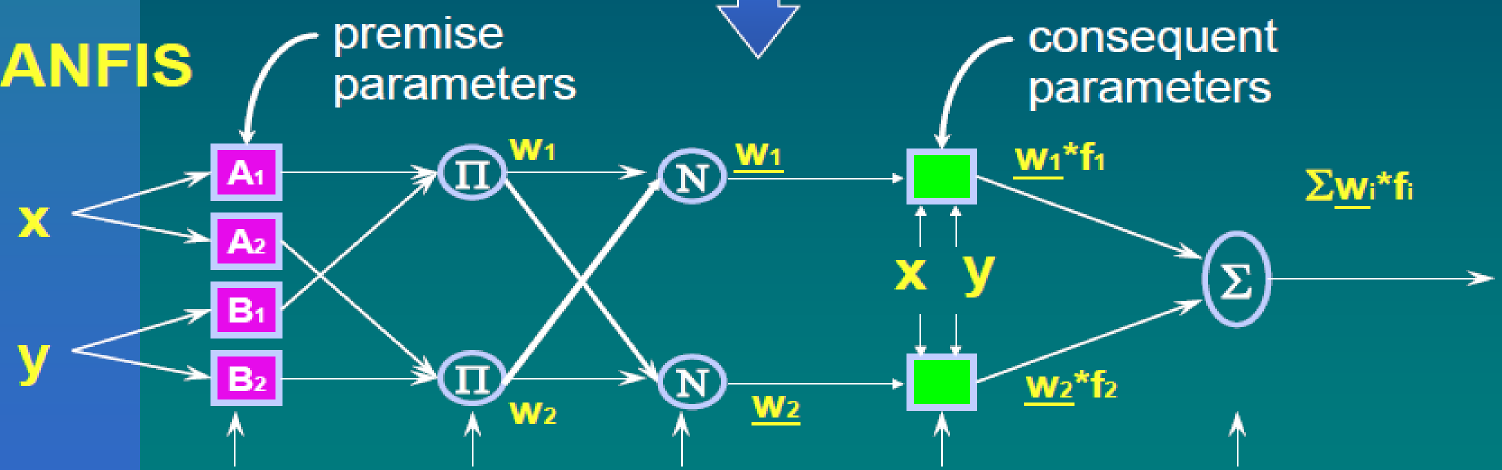


$f_1 = p_1*x+q_1*y+r_1$

$f_2 = p_2*x+q_2*y+r_2$

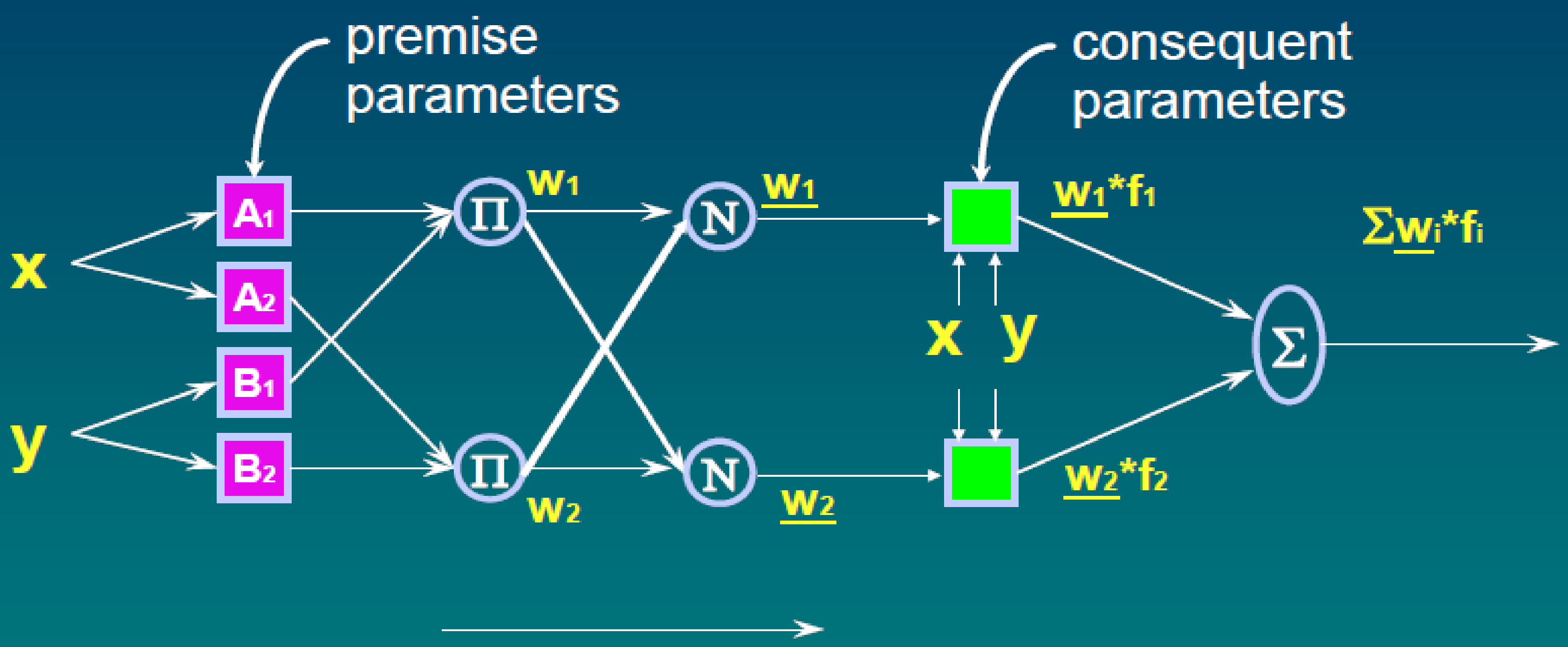
$$f = \frac{w_1*f_1+w_2*f_2}{w_1+w_2}$$

•ANFIS



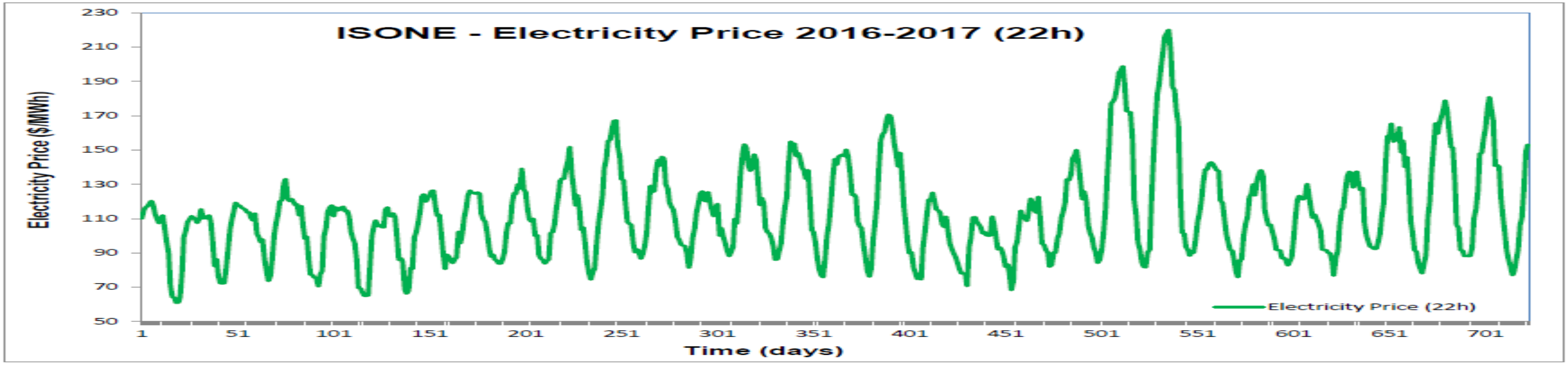
Functionally speaking, the ANFIS architecture is completely equivalent to a Sugeno fuzzy inference system. However, by implementing a fuzzy controller as ANFIS, we can easily employ the back-propagation-type learning procedure to find its parameters for achieving a minimal error measure.

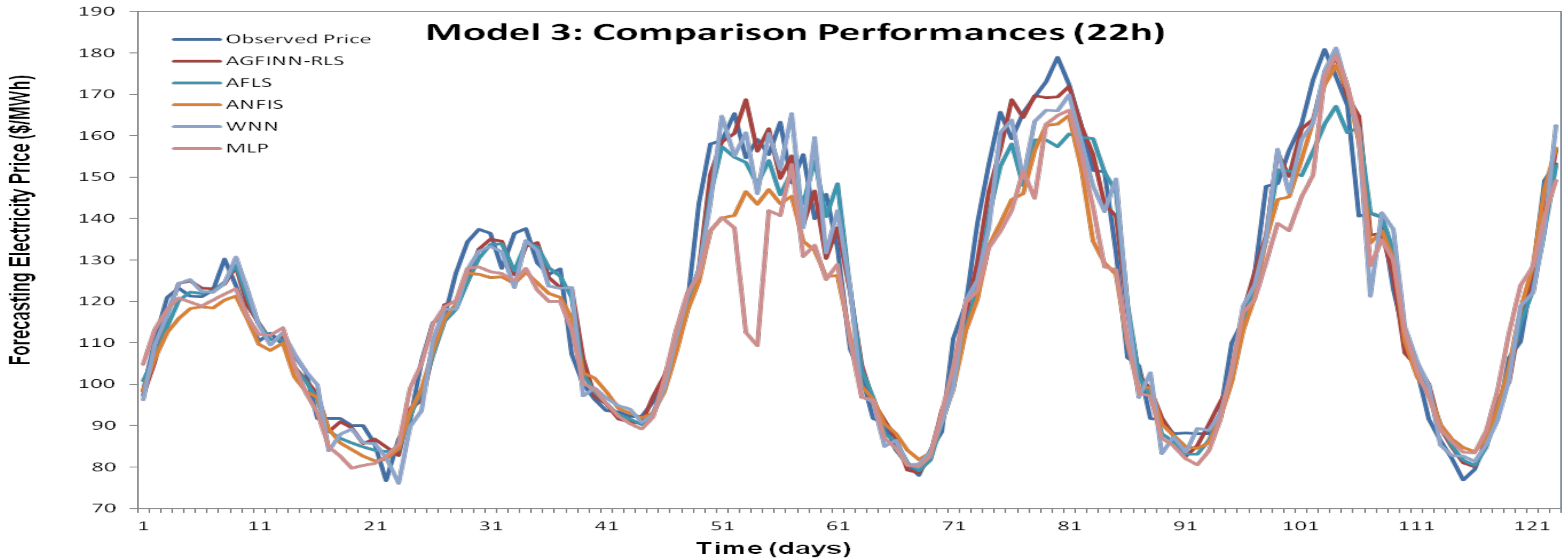
ANFIS Learning



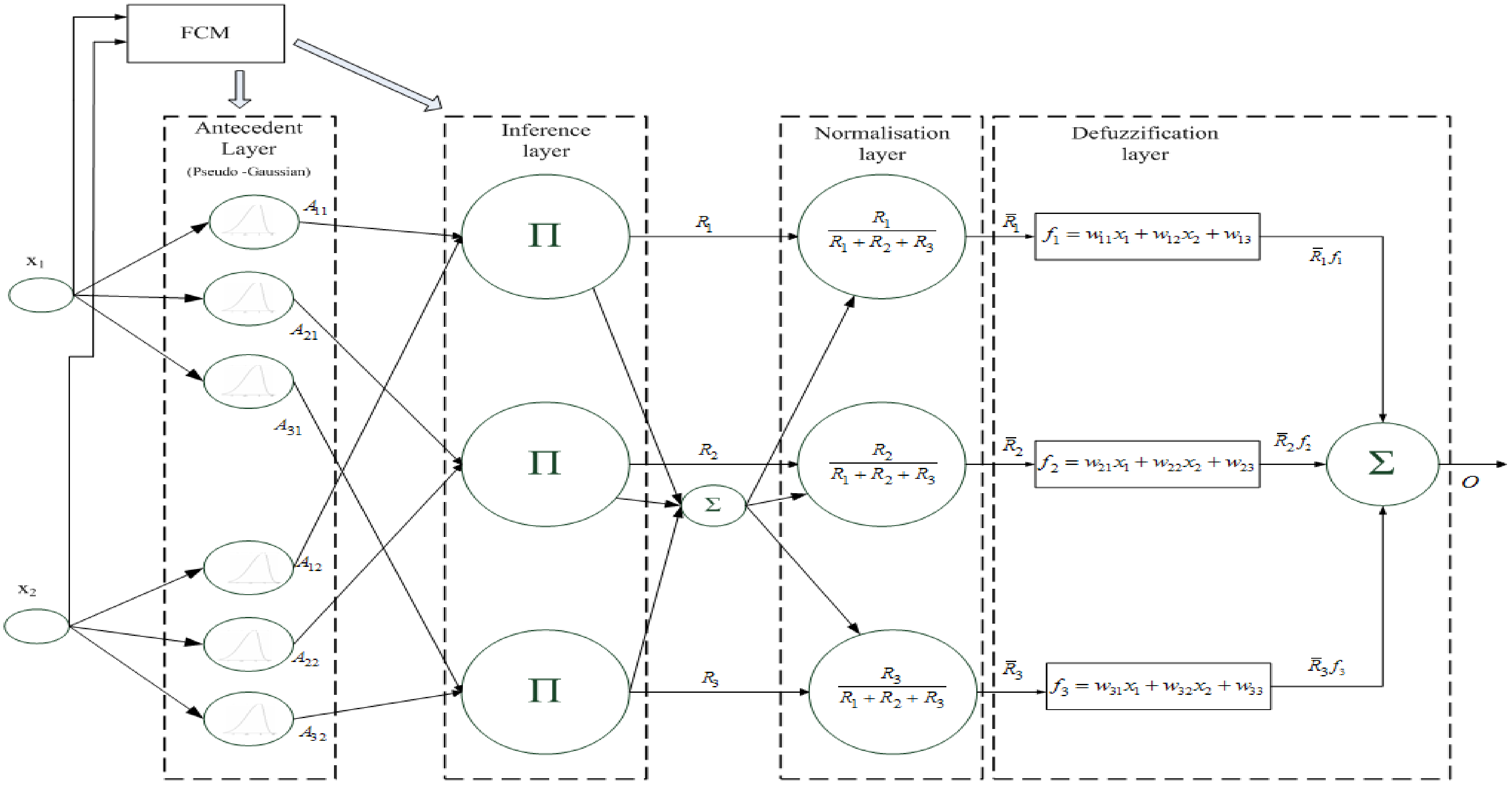
	forward pass	backward pass
MF param. (premise)	fixed	back propagation
Rule param. (consequence)	least-squares	fixed

Day Ahead Hourly Electricity Price Prediction in ISO New England Market using Neuro-Fuzzy Systems





AGFINN-TSK Models



MISO Configuration – TSK Defuzzification

Learning Algorithms: Gradient Descent Algorithm (GD)
Hybrid (GD + RLS)

Case Studies – Input Selection

Model 1
4 Past Price Variables + Load Demand

•**Target:**
•*Price(i,j): electricity price at the i^{th} hour on the $(j)^{th}$ day,*
•**Inputs:**
•*Price(i, j-1): price at the i^{th} hour on the $(j-1)^{th}$ day,*
•*Price(i, j-2): price at the i^{th} hour on the $(j-2)^{th}$ day,*
•*Price(i-1, j-1): price at the $(i-1)^{th}$ hour on the $(j-1)^{th}$ day,*
•*Price(i-2, j-1): price at the $(i-2)^{th}$ hour on the $(j-1)^{th}$ day,*
•*Load(i,j): electricity load at the i^{th} hour on the j^{th} day,*

Model 2
6 Past Price Variables

Target:
•*Price(i,j): electricity price at the i^{th} hour on the $(j)^{th}$ day,*
Inputs:
•*Price(i, j-1): price at the i^{th} hour on the $(j-1)^{th}$ day,*
•*Price(i, j-2): price at the i^{th} hour on the $(j-2)^{th}$ day,*
•*Price(i, j-3): price at the i^{th} hour on the $(j-3)^{th}$ day,*
•*Price(i, j-7): price at the i^{th} hour on the $(j-7)^{th}$ day,*
•*Price(i-1, j-1): price at the $(i-1)^{th}$ hour on the $(j-1)^{th}$ day,*
•*Price(i-2, j-1): price at the $(i-2)^{th}$ hour on the $(j-1)^{th}$ day,*

Model 3
6 Past Price Variables + Load Demand

Target:
•*Price(i,j): electricity price at the i^{th} hour on the $(j)^{th}$ day,*
Inputs:
•*Price(i, j-1): price at the i^{th} hour on the $(j-1)^{th}$ day,*
•*Price(i, j-2): price at the i^{th} hour on the $(j-2)^{th}$ day,*
•*Price(i, j-3): price at the i^{th} hour on the $(j-3)^{th}$ day,*
•*Price(i, j-7): price at the i^{th} hour on the $(j-7)^{th}$ day,*
•*Price(i-1, j-1): price at the $(i-1)^{th}$ hour on the $(j-1)^{th}$ day,*
•*Price(i-2, j-1): price at the $(i-2)^{th}$ hour on the $(j-1)^{th}$ day,*
•*Load(i,j): electricity load at the i^{th} hour on the j^{th} day,*

Results: Model 3

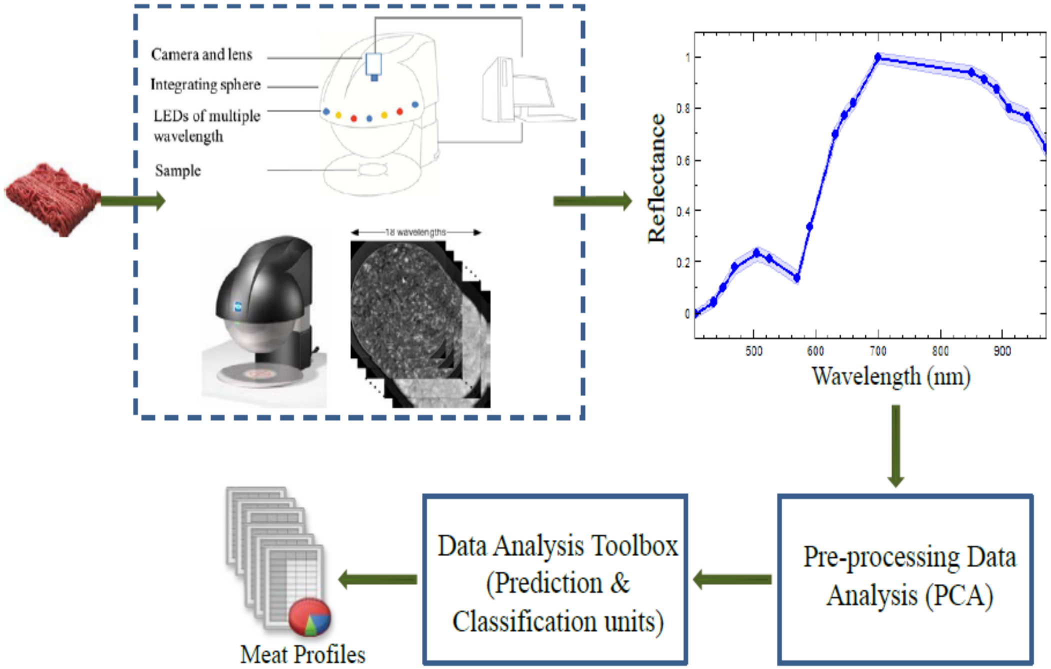
Stats	AGFINN_RLS	AGFINN_TSK	AGFINN_CA	AFLS	ANFIS	WNN	MLP
RMSE	2.6612	2.9988	3.5844	4.3667	5.4409	5.4184	8.0055
MAPE	4.9492	5.4832	6.9741	7.6772	8.3168	10.3565	16.5878
MAE	2.0810	2.2845	2.9241	3.3409	3.5089	4.3761	6.8644
SEP	5.9536	6.7089	8.0189	9.7692	12.1724	12.1219	17.9098
U1	0.0290	0.0326	0.0383	0.0474	0.0590	0.0579	0.0815
APE	608.757	674.439	857.814	944.293	1023	1273.9	2040.3
R^2	0.9635	0.9555	0.9588	0.9229	0.8630	0.8802	0.9079
Af	1.0499	1.0559	1.0691	1.0803	1.0909	1.1041	1.1606

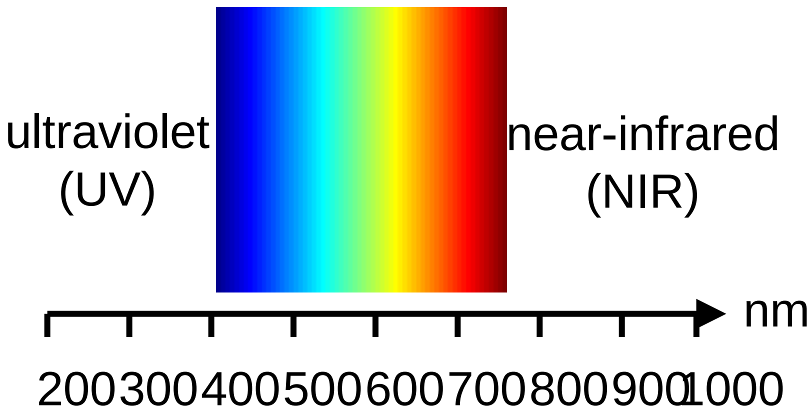
Electricity Price Forecasting Models at 4:00

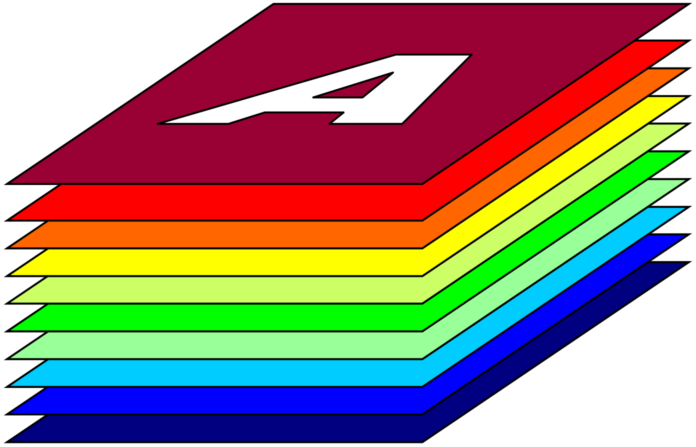
Stats	AGFINN_RLS	AGFINN_TSK	AGFINN_CA	AFLS	ANFIS	WNN	MLP
RMSE	6.4605	6.8514	7.5032	7.7340	9.1584	8.0368	11.4835
MAPE	3.9675	4.2418	5.0097	4.4775	5.3308	4.9288	6.0115
MAE	4.8768	5.2331	5.9290	5.7301	6.9329	6.1892	7.9360
SEP	5.3391	5.6621	6.2007	6.3915	7.5686	6.6417	9.4901
U1	0.0260	0.0277	0.0303	0.0314	0.0377	0.0325	0.0473
APE	487.999	521.740	616.196	550.734	655.688	606.248	739.416
R^2	0.9742	0.9718	0.9699	0.9670	0.9682	0.9604	0.9388
Af	1.0405	1.0434	1.0522	1.0462	1.0564	1.0508	1.0648

Electricity Price Forecasting Models at 22:00

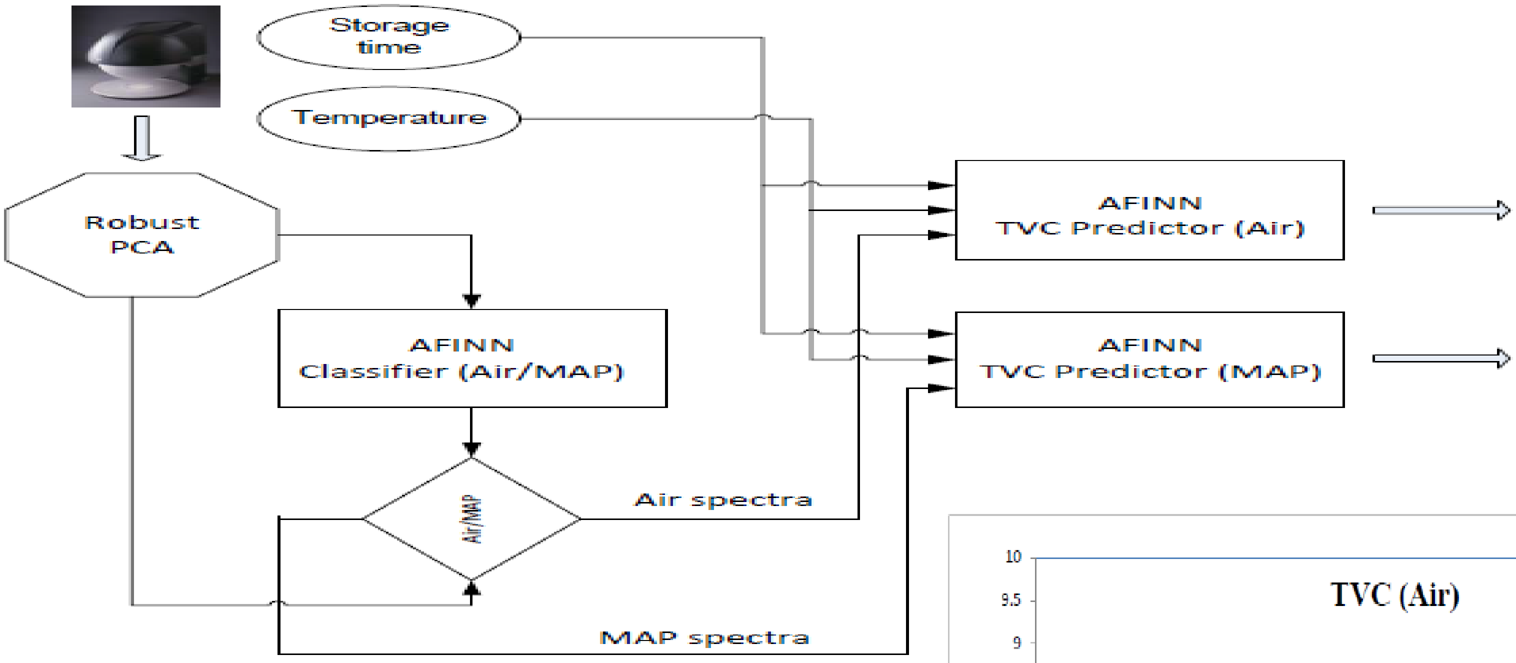
An Adaptive Neuro-Fuzzy Model for the Detection of Meat Spoilage using Multispectral Images

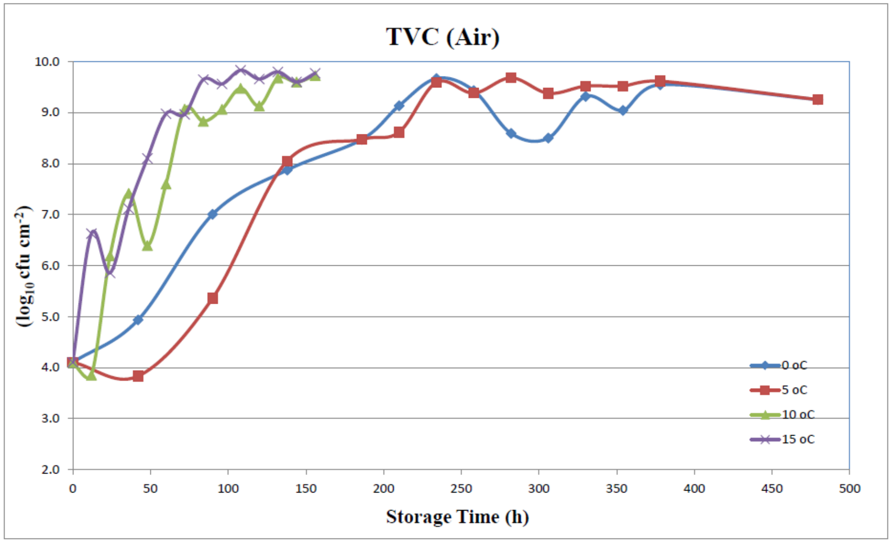


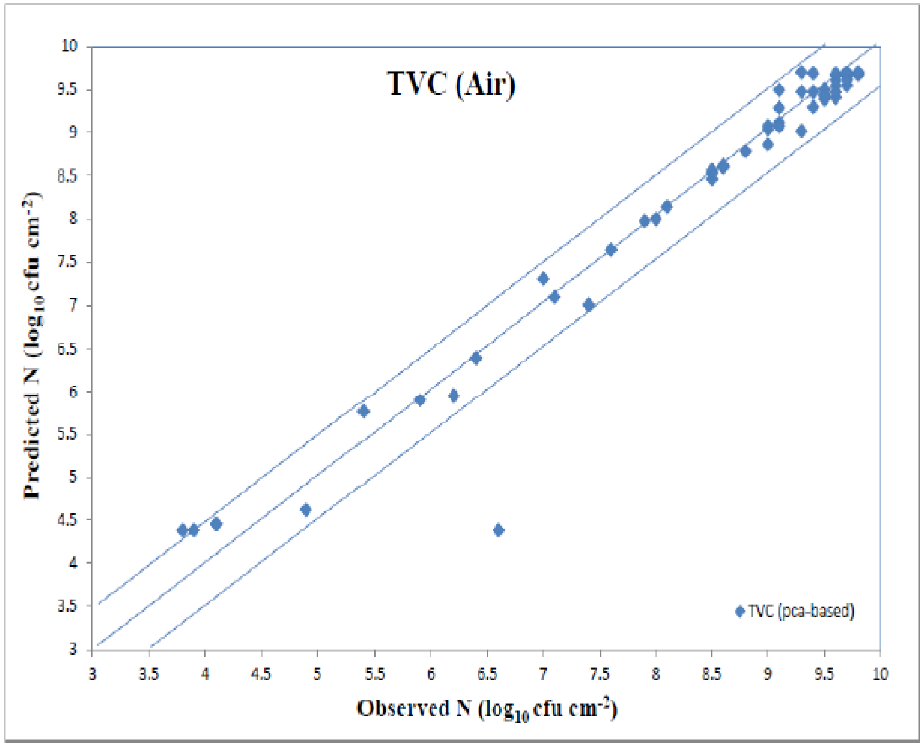




Vassilis S. Kodogiannis


Storage time
Temperature
Robust PCA
AFINN Classifier (Air/MAP)
Air/MAP
Air spectra
MAP spectra
AFINN TVC Predictor (Air)
AFINN TVC Predictor (MAP)


TVC (Air)
10.0
9.0
8.0
7.0
6.0
5.0
4.0
3.0
2.0
0
50
100
150
200
250
300
350
400
450
500
(log10 cfu cm-2)
Storage Time (h)
0 oC
5 oC
10 oC
15 oC


TVC (Air)
10
9.5
9
8.5
8
7.5
7
6.5
6
5.5
5
4.5
4
3.5
3
Predicted N (log10 cfu cm-2)
Observed N (log10 cfu cm-2)
TVC (pca-based)