Model Based Fault Detection and Isolation

Fault Diagnosis Technique Integration: Neural Networks and Fuzzy Systems for FDI

Silvio Simani

silvio.simani@unife.it

Machine Learning

- Improve automatically with experience
- Imitating human learning
  - Human learning

    Fast recognition and classification of complex classes of objects and concepts and fast adaptation
  - Example: neural networks
- Some techniques assume statistical source

  Select a statistical model to model the source
- Other techniques are based on reasoning or inductive inference (e.G. Decision tree)

Machine Learning Definition

A computer program is said to **learn** from *experience* **E** with respect to some class of *tasks* **T** and *performance measure* **P**, if its performance at tasks in **T**, as measured by **P**, improves with experience.

Example of Learning Problems

Handwriting recognition:

- T: recognizing and classifying handwritten words within images.
- P: percentage of words correctly classified.
- E: a database of handwritten words with given classification.

Issues in *Machine Learning*

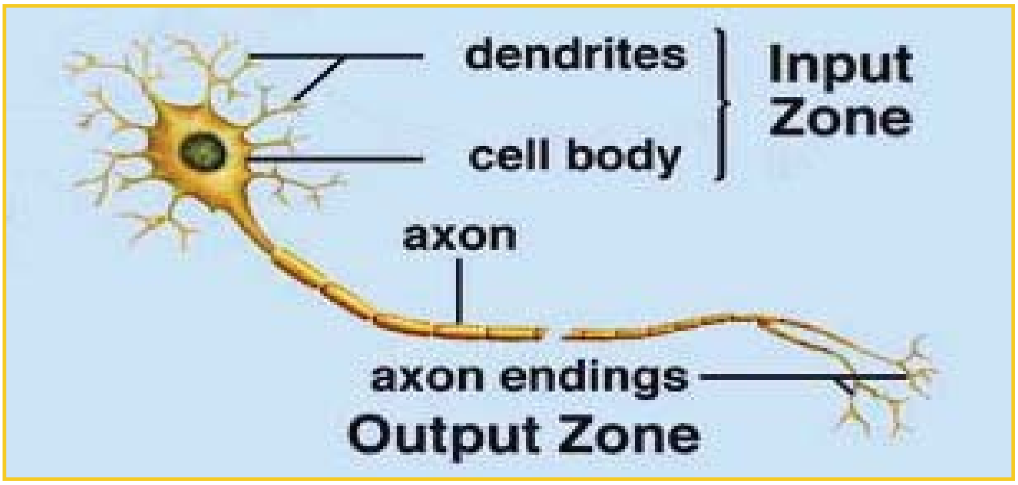
- What algorithms can approximate functions well and when?
- How does the number of training examples influence accuracy?
- How does the complexity of hypothesis representation impact it?
- How does noisy data influence accuracy?
- *How do you reduce a learning problem to a set of function approximation* ?

Summary

- *Machine learning* is useful for data mining, poorly understood domain (face recognition) and programs that must dynamically adapt.
- Draws from many diverse disciplines.
- Learning problem needs well-specified task, performance metric and training experience.
- Involve searching space of possible hypotheses. Different learning methods search different hypothesis space, such as numerical functions, *neural networks*, decision trees, symbolic rules.

Brain

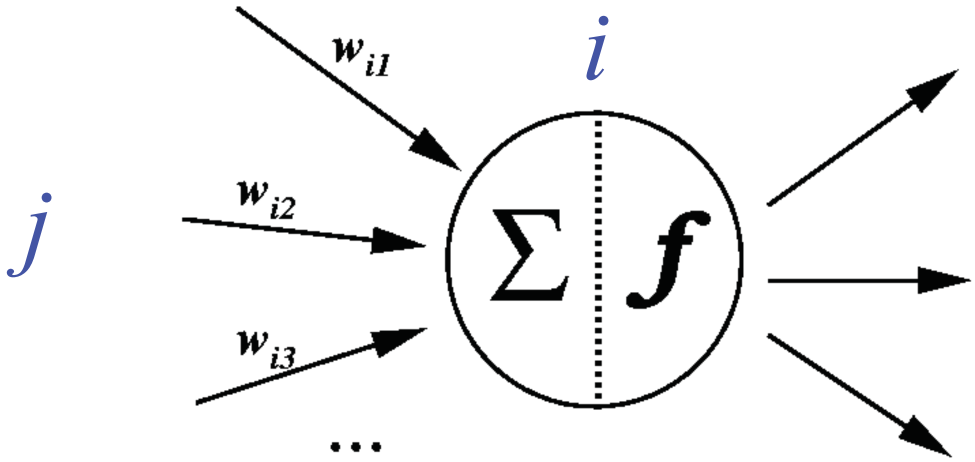
- 10^{11} neurons (processors)
- On average 1000-10000 connections



Artificial Neuron

bias

$$net_i = \sum_j w_{ij} y_j + b$$



$$y_i = f(net_i)$$

Artificial Neuron

- Input/Output Signal may be.
  - Real value.
  - Unipolar {0, 1}.
  - Bipolar {-1, +1}.
- Weight : w_{ij} – strength of connection.

Note that w_{ij} refers to the weight from **unit *j* to unit *i*** (not the other way round).

Artificial Neuron

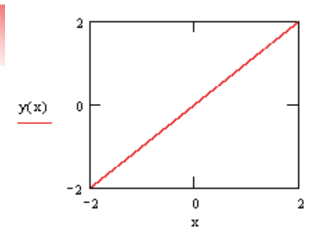
- The bias b is a constant that can be written as $w_{i0}y_0$ with $y_0 = b$ and $w_{i0} = 1$ such that

$$net_i = \sum_{j=0}^{n} w_{ij} y_j$$

- The function f is the unit's activation function. In the simplest case, f is the identity function, and the unit's output is just its net input. This is called a *linear unit*.
- Other activation functions are : step function, sigmoid function and Gaussian function.

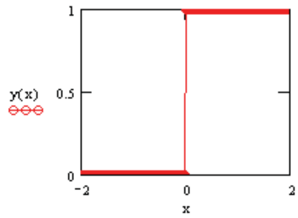
Activation Functions

$y(x) := x$



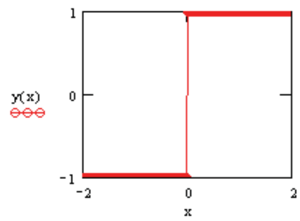
Identity function

$y(x) := \begin{vmatrix} temp \leftarrow 1 \ \text{if } x>0 \\ temp \leftarrow 0 \ \text{otherwise} \end{vmatrix}$



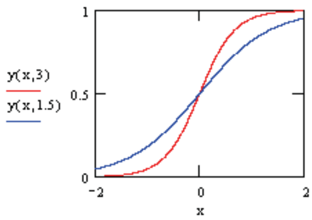
Binary Step function

$y(x) := \begin{vmatrix} temp \leftarrow 1 \ \text{if } x>0 \\ temp \leftarrow -1 \ \text{otherwise} \end{vmatrix}$



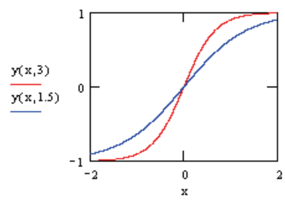
Bipolar Step function

$y(x, \sigma) := \frac{1}{1 + \exp(-\sigma \cdot x)}$



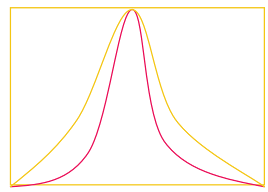
Sigmoid function

$y(x, \sigma) := \frac{2}{1 + \exp(-\sigma \cdot x)} - 1$



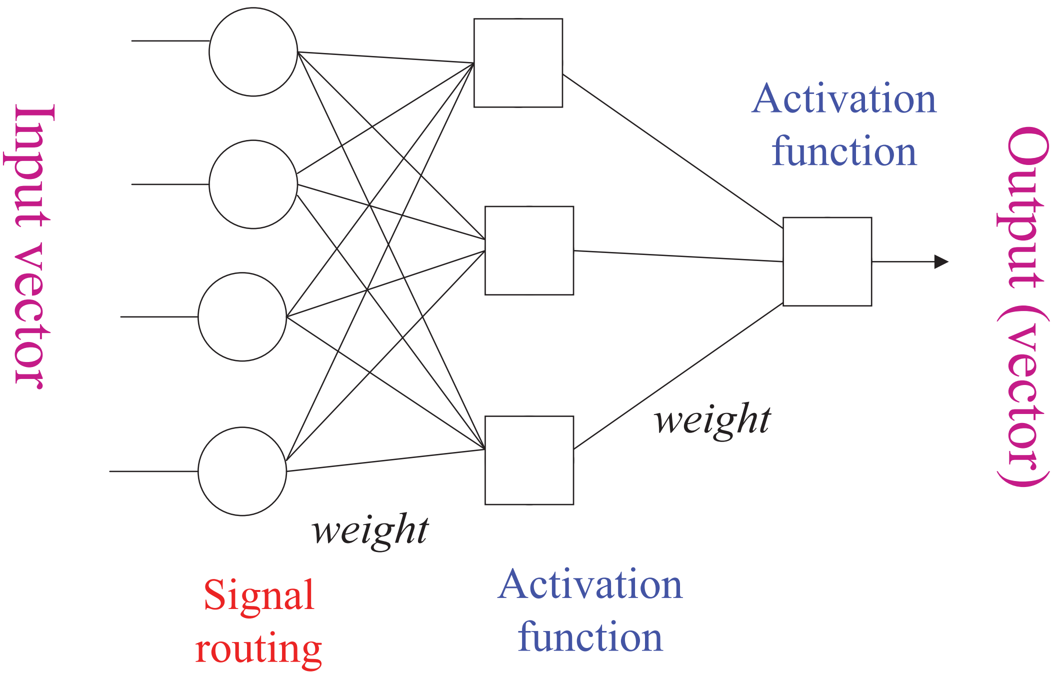
Bipolar Sigmoid function

$$y(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



Gaussian function

Artificial Neural Networks (ANN)



When Should ANN Solution Be Considered ?

- The solution to the problem cannot be explicitly described by an algorithm, a set of equations, or a set of rules.
- There is some evidence that an input-output mapping exists between a set of input and output variables.
- There should be a large amount of data available to train the network.

Problems That Can Lead to Poor Performance ?

- The network has to distinguish between very similar cases with a very high degree of accuracy.
- The train data does not represent the ranges of cases that the network will encounter in practice.
- The network has a several hundred inputs.
- The main discriminating factors are not present in the available data. *E.g.* Trying to assess the loan application without having knowledge of the applicant's salaries.
- The network is required to implement a very complex function.

Applications of Artificial Neural Networks

- Manufacturing : fault diagnosis, fraud detection.
- Retailing : fraud detection, forecasting, data mining.
- Finance : fraud detection, forecasting, data mining.
- Engineering : fault diagnosis, signal/image processing.
- Production : fault diagnosis, forecasting.
- Sales & marketing : forecasting, data mining.

THE PERCEPTRON

- The perceptron was a simple model of ANN introduced by Rosenblatt of MIT in the 1960' with the idea of learning.

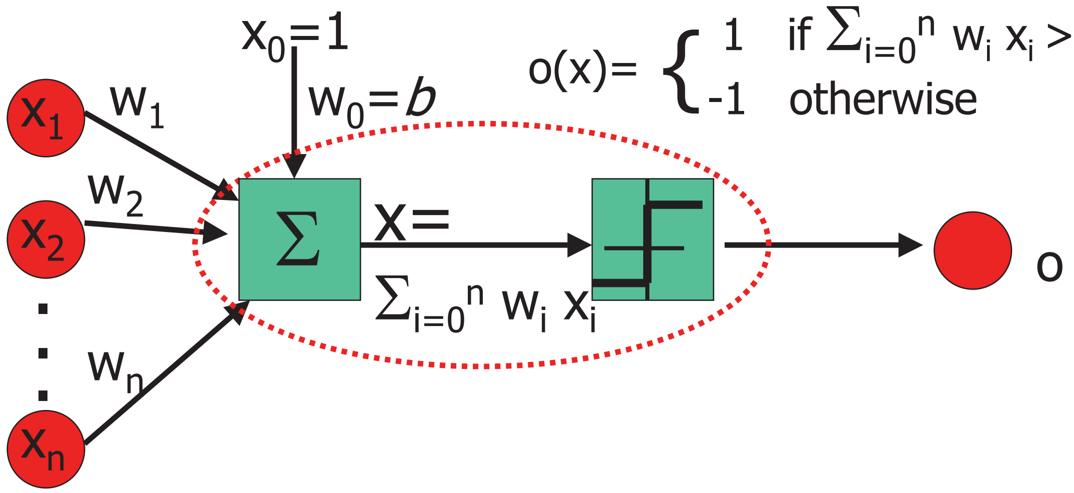
- Perceptron is designed to accomplish a simple pattern recognition task: after learning with real value training data
  $\{ \underline{x(i)}, d(i), \ i = 1,2, \ldots, p\}$ where $d(i) = 1$ or -1

- For a new signal (pattern) $\underline{x(i+1)}$, the perceptron is capable of telling you to which class the new signal belongs

$\underline{x(i+1)} \longrightarrow \boxed{\text{perceptron}} = 1 \text{ or } -1$

Perceptron

- Linear Threshold Unit (LTU)



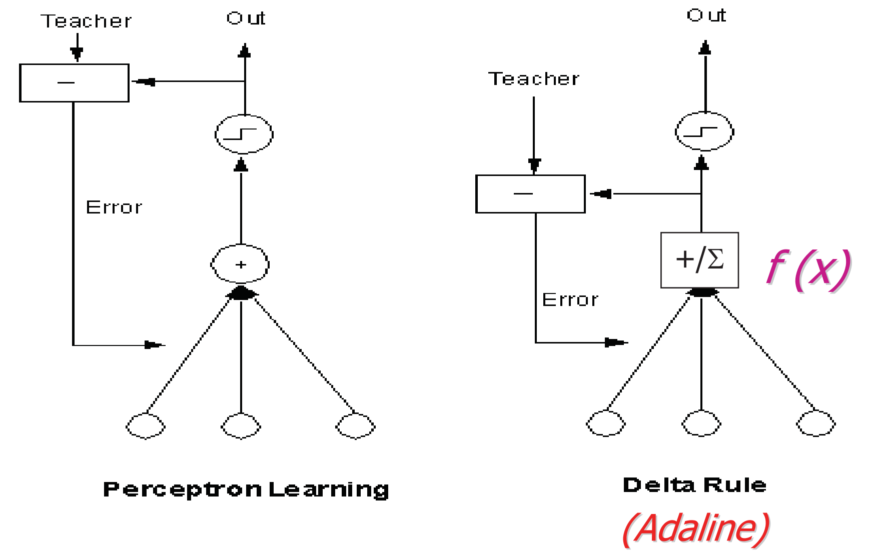
$$o(x)=\begin{cases} 1 & \text{if } \sum_{i=0}^{n} w_i x_i > 0 \\ -1 & \text{otherwise} \end{cases}$$

Unhappy Over Perceptron Training

- When a perceptron gives the right answer, no learning takes place
- Anything below the threshold is interpreted as 'no', even it is just below the threshold.
- It might be better to train the neuron based on how far below the threshold it is.

- Replace the step function in the perceptron with a continuous (differentiable) function f, e.g the simplest is linear function

- With or without the threshold, the Adaline is trained based on the output of the function f rather than the final output.



General Approach gradient descent method

To find g

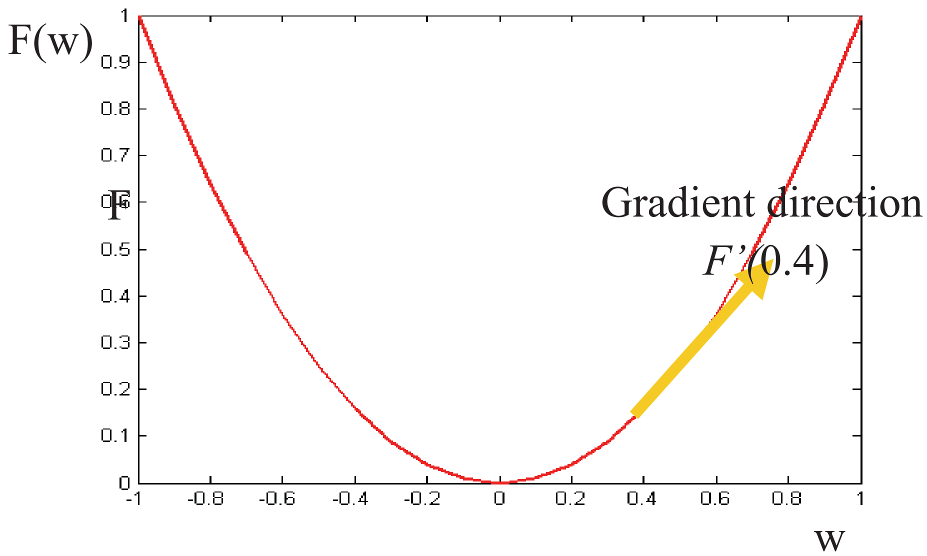
$$\underline{w}(t+1) = \underline{w}(t)+g(E(\underline{w}(t)))$$

so that $\underline{w}$ automatically tends to the global minimum of E(w).

$$\underline{w}(t+1) = \underline{w}(t)- E'(\underline{w}(t))\eta(t)$$

(see figure below)

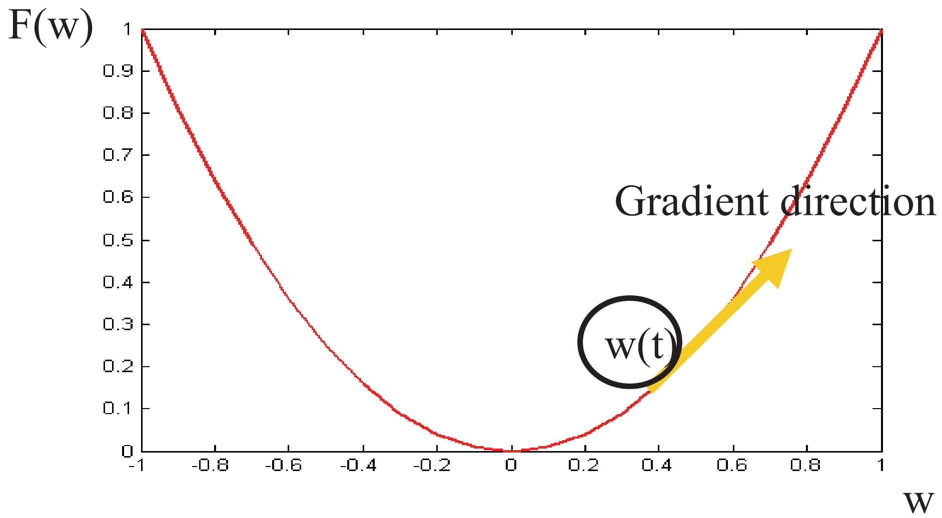
- Gradient direction is the direction of uphill for example, in the Figure, at position 0.4, the gradient is uphill (F is E, consider one dim case)



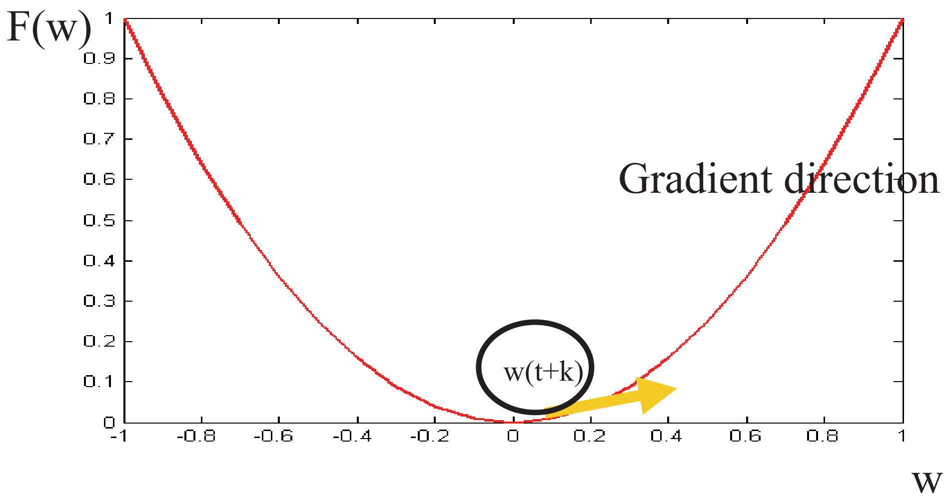
- In gradient descent algorithm, we have

$$\underline{w}(t+1) = \underline{w}(t) - F'(w(t))\,\eta(\tau)$$

therefore the ball goes downhill since – F′(w(t)) is downhill direction



- Gradually the ball will stop at a local minima where the gradient is zero



- **In words**

Gradient method could be thought of as a ball rolling down from a hill: the ball will roll down and finally stop at the valley

Thus, the weights are adjusted by

$$w_j(t+1) = w_j(t) + \eta(t) \Sigma \; [d(i) - f(\underline{w}(t) \cdot \underline{x}(i))] \, x_j(i) \, f'$$

This corresponds to gradient descent on the quadratic error surface E

When f′ =1, we have the perceptron learning rule (we have in general f′>0 in neural networks). The ball moves in the right direction.

Two types of network training:

Sequential mode (on-line, stochastic, or per-pattern) :
Weights updated after each pattern is presented (Perceptron is in this class)

Batch mode (off-line or per-epoch) :
Weights updated after all patterns are presented

Comparison Perceptron and Gradient Descent Rules

- Perceptron learning rule guaranteed to succeed if
  - Training examples are linearly separable
  - Sufficiently small learning rate η
- Linear unit training rule uses gradient descent guaranteed to converge to hypothesis with minimum squared error given sufficiently small learning rate η
  - Even when training data contains noise
  - Even when training data not separable by hyperplanes

Summary

Perceptron

$$\underline{W}(t+1)= \underline{W}(t)+\eta(t)\ [\ d(t) - \text{sign}\ (\underline{w}(t)\ .\ \underline{x})]\ \underline{x}$$

Adaline (Gradient descent method)

$$\underline{W}(t+1)= \underline{W}(t)+\eta(t)\ [\ d(t) - f(\underline{w}(t)\ .\ \underline{x})]\ \underline{x}\ f'$$

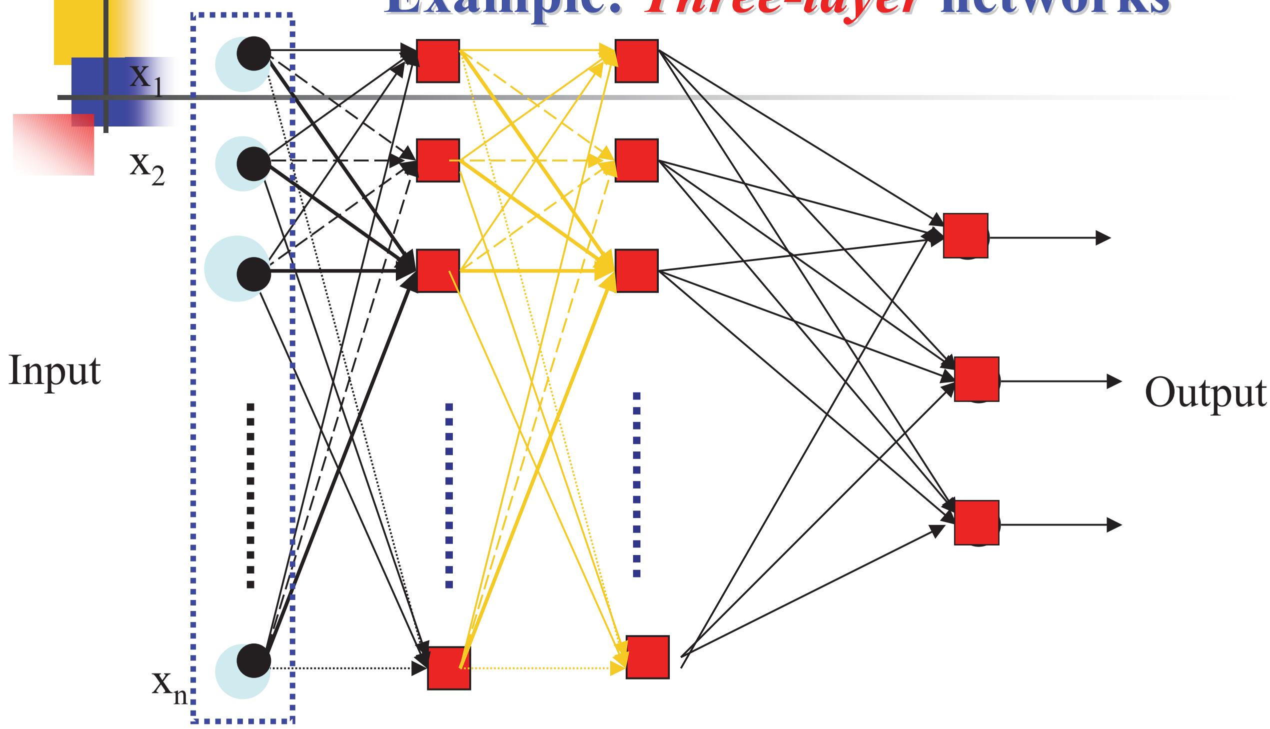
Multi-Layer Perceptron (MLP)

Idea: Credit assignment problem

- Problem of assigning 'credit' or 'blame' to individual elements involving in forming overall response of a learning system (hidden units)

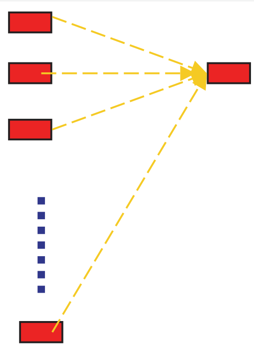
- In neural networks, problem relates to dividing which weights should be altered, by how much and in which direction.

Example: *Three-layer* networks



Properties of architecture

- No connections within a layer
- No direct connections between input and output layers
- Fully connected between layers
- Often more than 2 layers
- Number of output units need not equal number of input units
- Number of hidden units per layer can be more or less than input or output units



Each unit '▬' is a perceptron

$$y_i = f\left(\sum_{j=1}^{m} w_{ij} x_j + b_i\right)$$

BP (Back Propagation)

gradient descent method

+

multilayer networks

BP learning algorithm

Solution to "credit assignment problem" in MLP

Rumelhart, Hinton and Williams (1986)

BP has two phases:

Forward pass phase: computes 'functional signal', feedforward propagation of input pattern signals through network

Backward pass phase: computes 'error signal', propagation of error (difference between actual and desired output values) backwards through network starting at output units

Summary of BP learning algorithm

Set learning rate η

Set initial weight values (incl.. biases): w, W

Loop until stopping criteria satisfied:

present input pattern to NN inputs
compute functional signal for input units
compute functional signal for output units

present Target response to output units
compute error signal for output units
compute error signal for input units
update all weights at same time
increment n to n+1 and select next I and d

end loop

Advantages and disadvantages of different modes

Sequential mode:

- Less storage for each weighted connection
- Random order of presentation and updating per pattern means search of weight space is stochastic-reducing risk of local minima able to take advantage of any redundancy in training set (*i.e.* same pattern occurs more than once in training set, esp. for large training sets)
- Simpler to implement

Batch mode:

- Faster learning than sequential mode

Goals of Neural Network Training

To give the correct output for input training vector (Learning)

To give good responses to new unseen input patterns (Generalization)

Training and Testing Problems

- **Stuck neurons**: Degree of weight change is proportional to derivative of activation function, weight changes will be greatest when units receives mid-range functional signal than at extremes neuron. To avoid stuck neurons weights initialization should give outputs of all neurons approximate 0.5
- **Insufficient number of training patterns**: In this case, the training patterns will be learnt instead of the underlying relationship between inputs and output, i.e. network just memorizing the patterns.
- **Too few hidden neurons**: network will not produce a good model of the problem.
- **Over-fitting**: the training patterns will be learnt instead of the underlying function between inputs and output because of too many of hidden neurons. This means that the network will have a poor generalization capability.

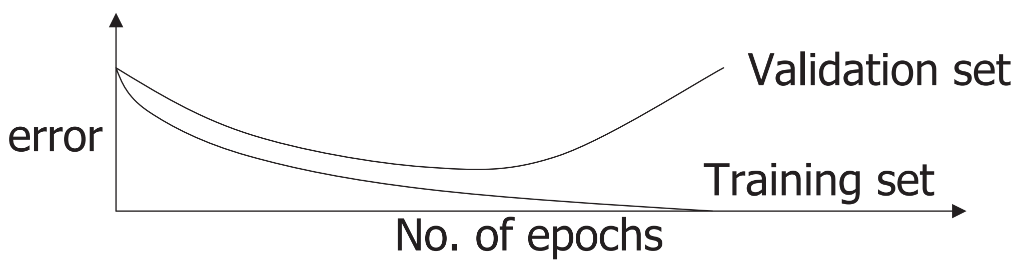
Use of Available Data Set for Training

The available data set is normally split into three sets as follows:

- Training set – use to update the weights. Patterns in this set are repeatedly in random order. The weight update equation are applied after a certain number of patterns.
- Validation set – use to decide when to stop training only by monitoring the error.
- Test set – Use to test the performance of the neural network. It should not be used as part of the neural network development cycle.

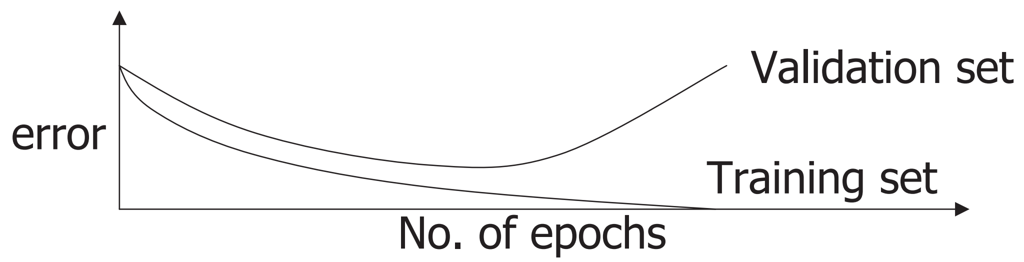
Earlier Stopping - Good Generalization

- Running too many epochs may overtrain the network and result in overfitting and perform poorly in generalization.
- Keep a hold-out validation set and test accuracy after every epoch. Maintain weights for best performing network on the validation set and stop training when error increases increases beyond this.



Model Selection by Cross-validation

- Too few hidden units prevent the network from learning adequately fitting the data and learning the concept (more than two layer networks).
- Too many hidden units leads to overfitting.
- Similar cross-validation methods can be used to determine an appropriate number of hidden units by using the optimal test error to select the model with optimal number of hidden layers and nodes.



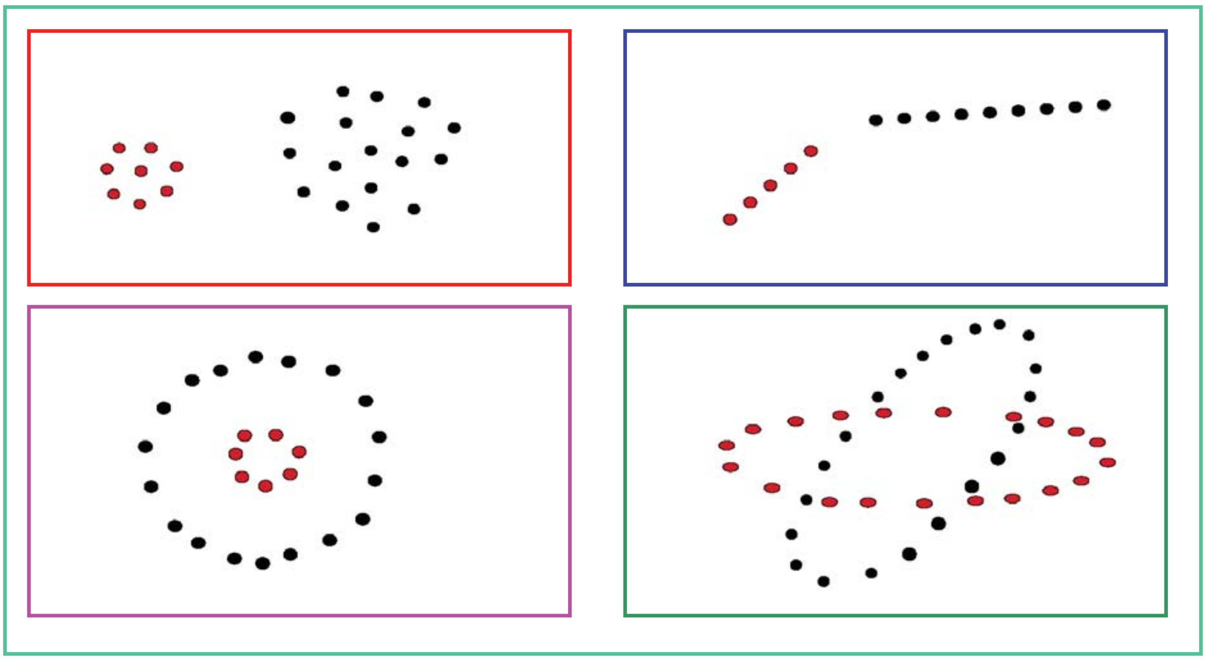
Fuzzy Modelling

Fuzzy Clustering with Application to Data-Driven Modelling

Introduction

- The ability to cluster data (concepts, perceptions, etc.)
  - essential feature of human intelligence.
- A cluster is a set of objects that are more similar to each other than to objects from other clusters.
- Applications of clustering techniques in pattern recognition and image processing.
- Some machine-learning techniques are based on the notion of similarity (decision trees, case-based reasoning)
- Non-linear regression and black-box modelling can be based on the partitioning data into clusters.

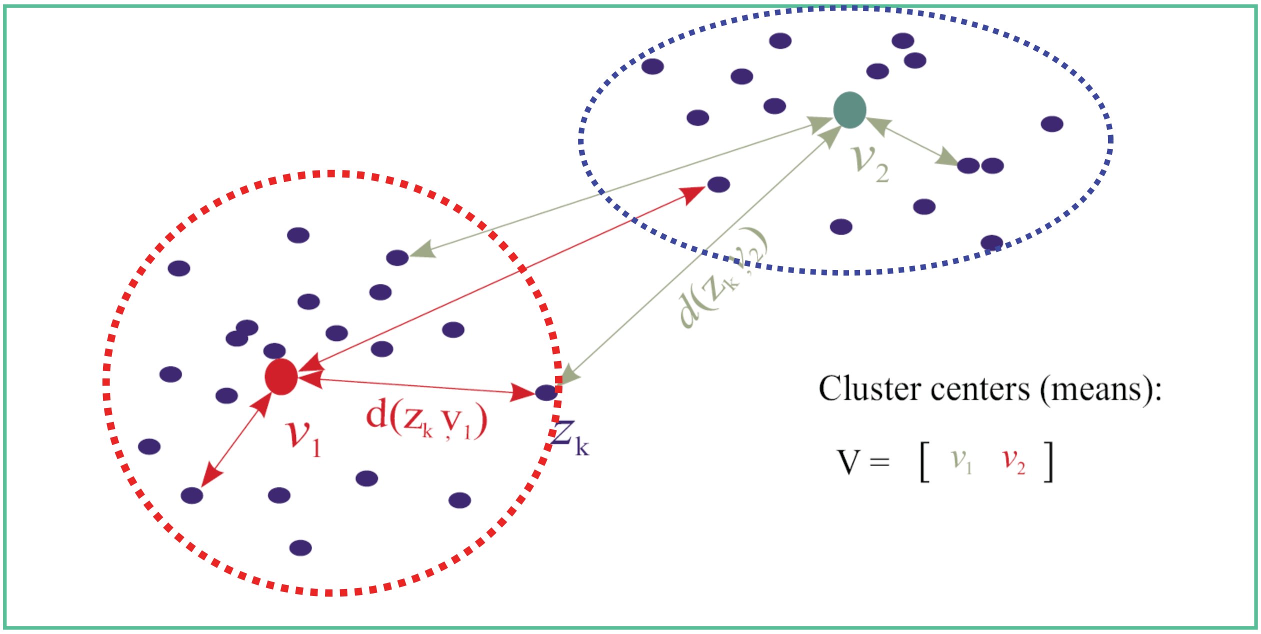
Examples of Clusters



Problem Formulation

- Given is a set of data in R^n and the (estimated) number of clusters to look for (a difficult problem, more on this later).
- Find the partitioning of the data into subsets (clusters), such that samples within a subset are more similar to each other than to samples from other subsets.
- Similarity is mathematically formulated by using a distance measure (i.e., a dissimilarity function).
- Usually, each cluster will have a prototype and the distance is measured from this prototype.

Distance Measure



Mathematical Formulation of Clustering

- **Given the data:**

$$\mathbf{z}_k = [z_{1k},\ z_{2k},\ \ldots,\ z_{nk}]^T \in \Re^n,\ k = 1, \cdots, N$$

Find:

- **the partition matrix:**

$$\mathbf{U} = \begin{bmatrix} \mu_{11} & \cdots & \mu_{1k} & \cdots & \mu_{1N} \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ \mu_{c1} & \cdots & \mu_{ck} & \cdots & \mu_{cN} \end{bmatrix}$$

- **and the cluster prototype (centres):**

$$\mathbf{V} = \{\mathbf{v}_1,\ \mathbf{v}_2,\ \ldots,\ \mathbf{v}_c\},\quad \mathbf{v}_i \in \Re^n$$

Fuzzy Clustering: an Optimisation Approach

➢ **Objective function (least-squares criterion):**

$$J(\mathbf{Z}; \mathbf{V}, \mathbf{U}, \mathbf{A}) = \sum_{i=1}^{c} \sum_{j=1}^{N} \mu_{i,j}^m d^2_{\mathbf{A}_i}(\mathbf{z}_j, \mathbf{v}_i)$$

➢ **subject to constraints:**

$0 \le \mu_{i,j} \le 1,$	$i = 1, \ldots, c,\ j = 1, \ldots, N$	membership degree
$0 < \sum_{j=1}^{N} \mu_{i,j} < 1,$	$i = 1, \ldots, c$	no cluster empty
$\sum_{i=1}^{c} \mu_{i,j} = 1,$	$j = 1, \ldots, N$	total membership

Gustafson–Kessel Algorithm

Repeat:

1. Compute cluster prototypes (means): $$v_i = \frac{\sum_{k=1}^{N} \mu_{i,k}^m \mathbf{z}_k}{\sum_{k=1}^{N} \mu_{i,k}^m}$$
2. Compute covariance matrices: $$\mathbf{F}_i = \frac{\sum_{k=1}^{N} \mu_{ik}^m (\mathbf{z}_k - \mathbf{v}_i)(\mathbf{z}_k - \mathbf{v}_i)^T}{\sum_{k=1}^{N} \mu_{ik}^m}$$
3. Compute distances: $$d_{ik} = (\mathbf{z}_k - \mathbf{v}_i)^T \rho_i \det(\mathbf{F}_i)^{1/n} \mathbf{F}_i^{-1} (\mathbf{z}_k - \mathbf{v}_i)$$
4. Compute partition matrix: $$\mu_{ik} = \frac{1}{\sum_{j=1}^{c} (d_{ik}/d_{jk})^{1/(m-1)}}$$

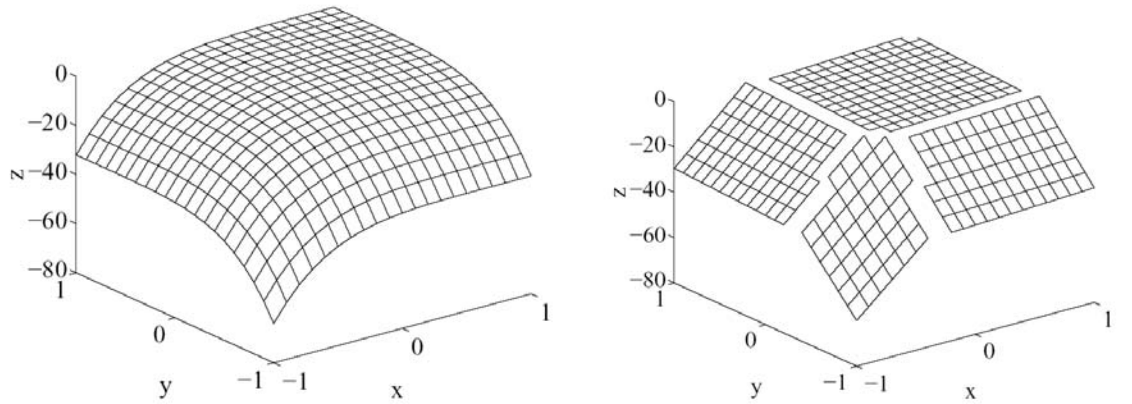
until $\|\Delta \mathbf{U}\| < \epsilon$

Number of Clusters

Validity measures

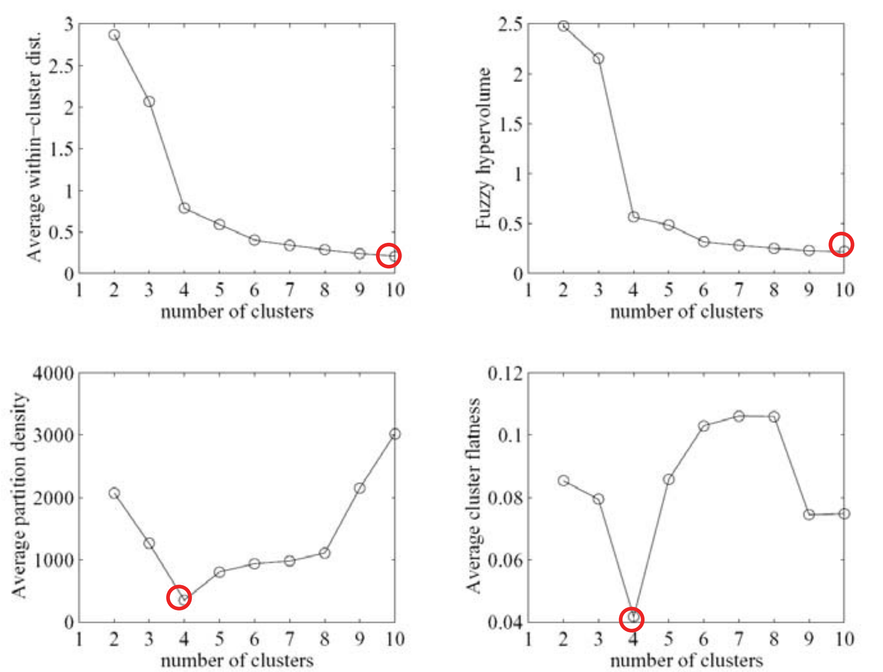
- Fuzzy hypervolume: $$V_h = \sum_{i=1}^{c} [\det(\mathbf{F}_i)]^{1/2}$$
- Average within-cluster distance: $$D_w = \frac{1}{c} \sum_{i=1}^{c} \frac{\sum_{k=1}^{N} \mu_{ik}^m D_{ik}^2}{\sum_{k=1}^{N} \mu_{ik}^m}$$
- Xie Beni index . . .

Validity Measures: Example



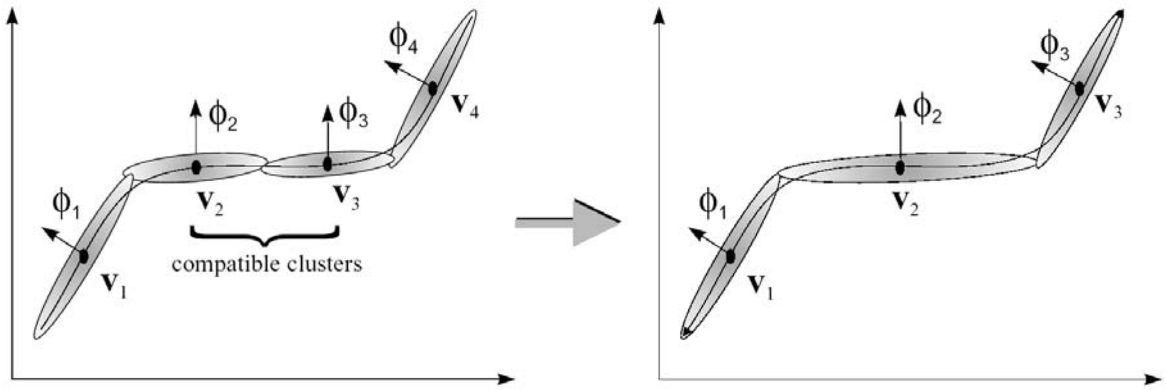
Data over 4 clusters

Validity Measures



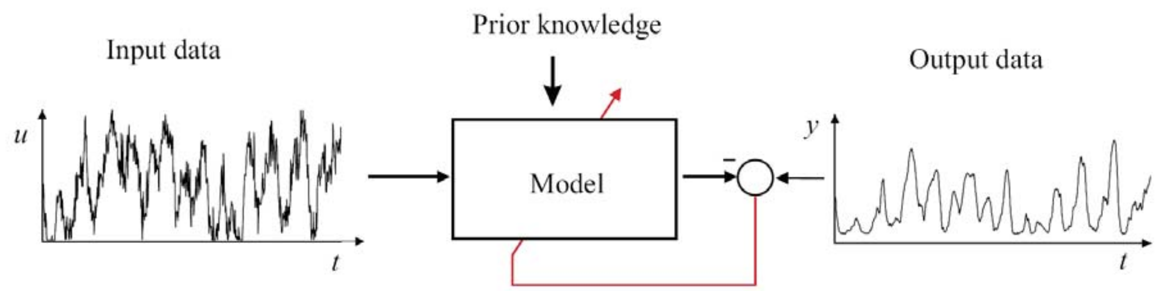
Number of Clusters

Compatible cluster merging



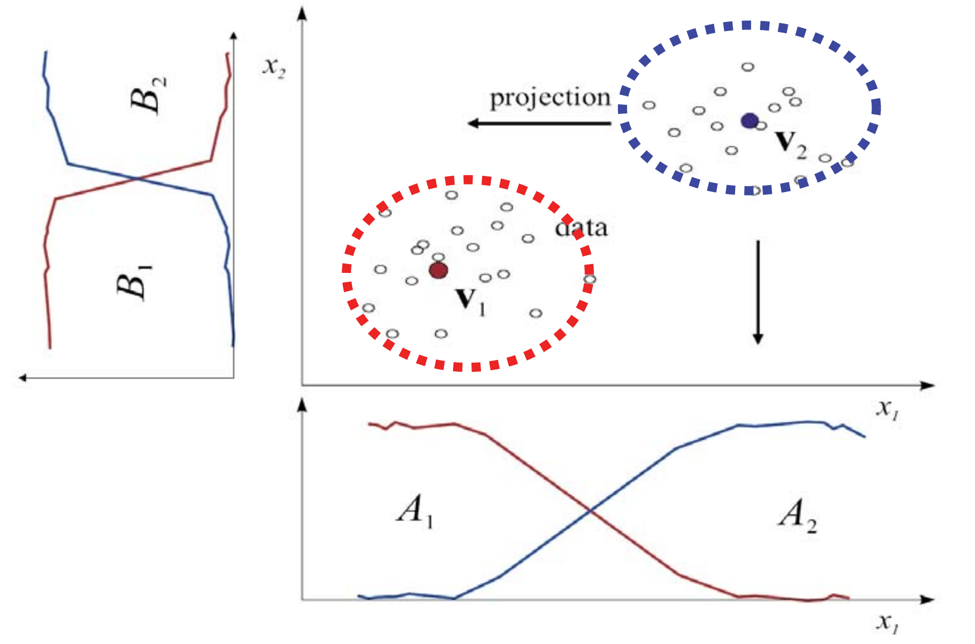
$$| \Phi_i \cdot \Phi_j | \geq k_1, \quad k_1 \to 1 \quad \text{and} \quad \|v_i - v_j\| \leq k_2, \quad k_2 \to 0$$

Data-Driven (Black-Box) Modelling

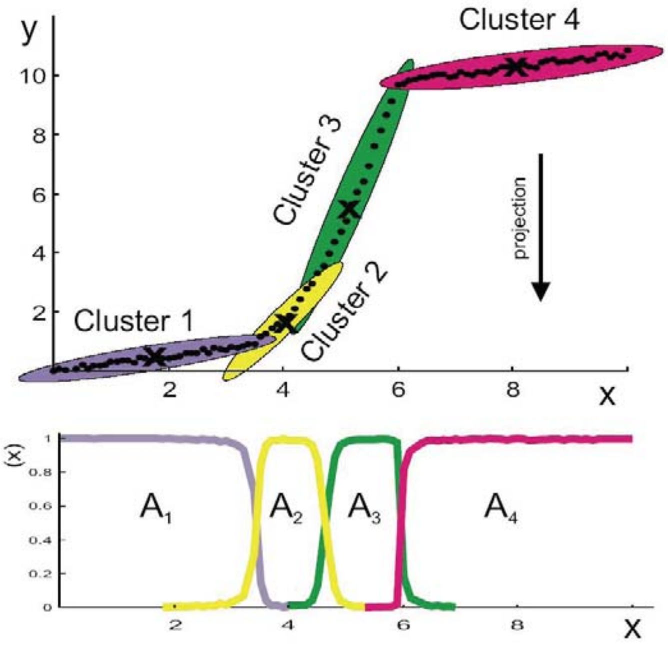


- Linear model (for linear systems only, limited in use)
- Neural network (black box, unreliable extrapolation)
- Rule-based model (more transparent, 'grey-box')

Extraction of Rules by Fuzzy Clustering



Extraction of Rules by Fuzzy Clustering



Takagi-Sugeno model

Rule-based description:

If x is A_1 then $y = a_1x + b_1$
If x is A_2 then $y = a_2x + b_2$

etc...

Example: Non-linear Autoregressive System (NARX)

$$\boxed{x(k+1) = f(x(k)) + \epsilon(k)}$$

$$f(x) = \begin{cases} 2x - 2, & 0.5 < x \\ -2x, & -0.5 \leq x < 0.5 \\ 2x + 2, & x \leq -0.5 \end{cases}$$

Structure Selection and Data Preparation

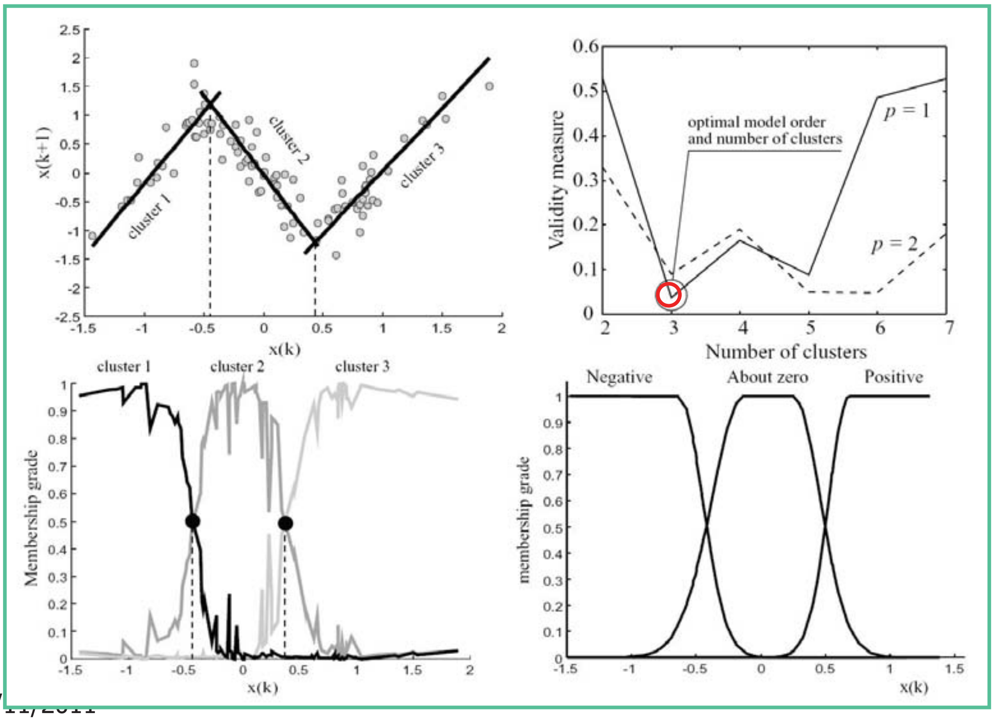
1. Choose model order p

$$x(k+1) = f(\underbrace{x(k), x(k-1), \ldots, x(k-p+1)}_{\mathbf{x}(k)})$$

2. Form pattern matrix **Z** to be clustered

$$\mathbf{Z}^T = \begin{bmatrix} x(1) & x(2) & \ldots & x(p) & x(p+1) \\ x(2) & x(3) & \ldots & x(p+1) & x(p+2) \\ \vdots & \vdots & & \vdots & \vdots \\ x(N-p) & x(N-p+1) & \ldots & x(N-1) & x(N) \end{bmatrix}$$

Clustering Results



References

• *Neural Networks for Identification, Prediction, and Control*, by Duc Truong Pham and Xing Liu. Springer Verlag; (December 1995). ISBN: 3540199594

• *Nonlinear Identification and Control: A Neural Network Approach*, by G. P. Liu. Springer Verlag; (October 2001). ISBN: 1852333421.

• *Fuzzy Modeling for Control*, by Robert Babuska. Springer; 1st edition (May 1, 1998) ISBN-10: 0792381548, ISBN-13: 978-0792381549.