Neural Network Solutions for Object Modeling and Pattern Recognition

Emil M. Petriu, Ana-Maria Cretu

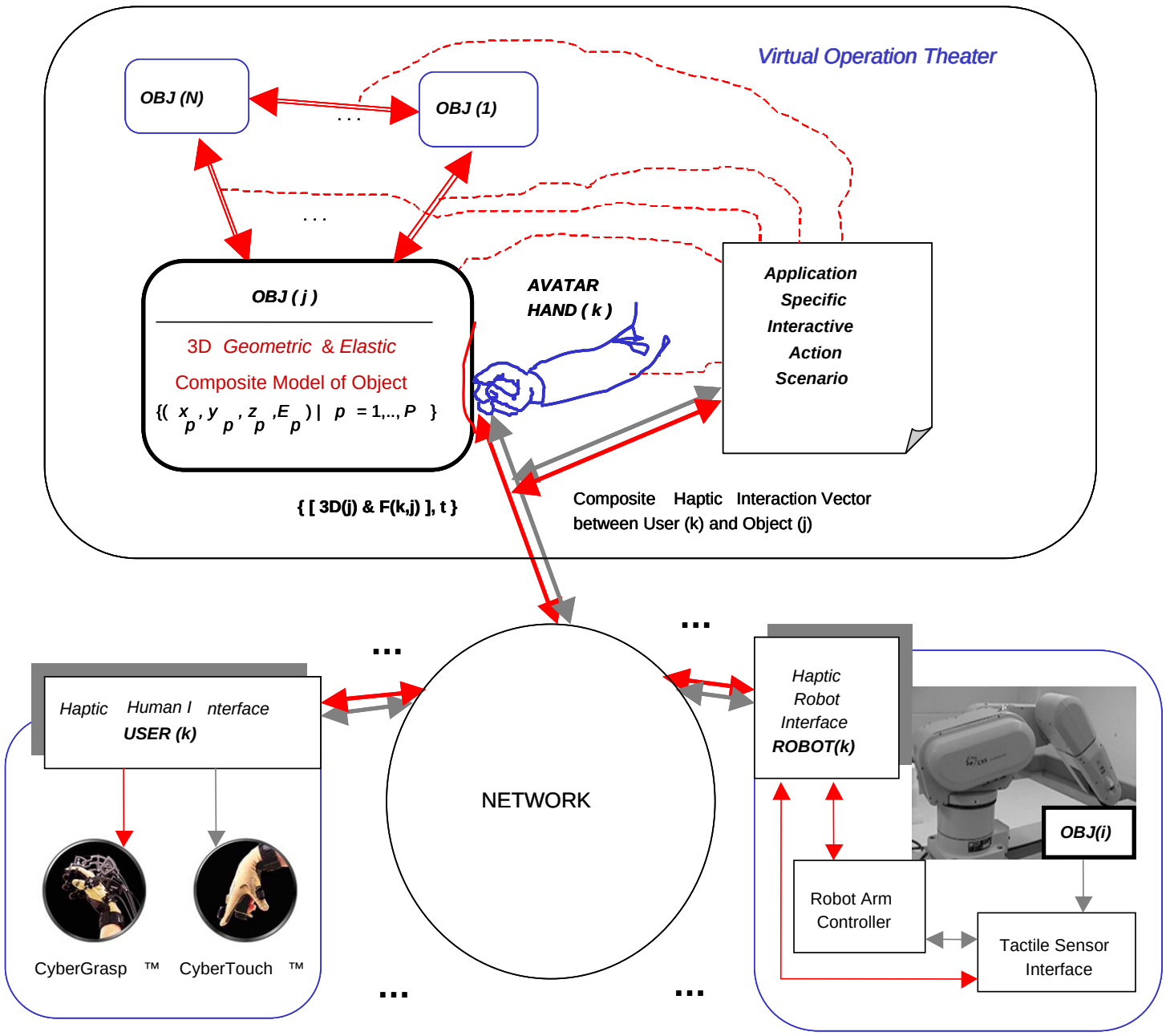
School of Information Technology and Engineering
University of Ottawa, Canada

Abstract

Neural Networks (NNs), which are able to learn nonlinear behaviors from a limited set of measurement data, can provide efficient modeling and pattern recognition solutions for many applications.

Due to their continuous memory behavior, NNs are able to provide instantaneously an estimation of the output values corresponding to input data that were not part of the initial training set.

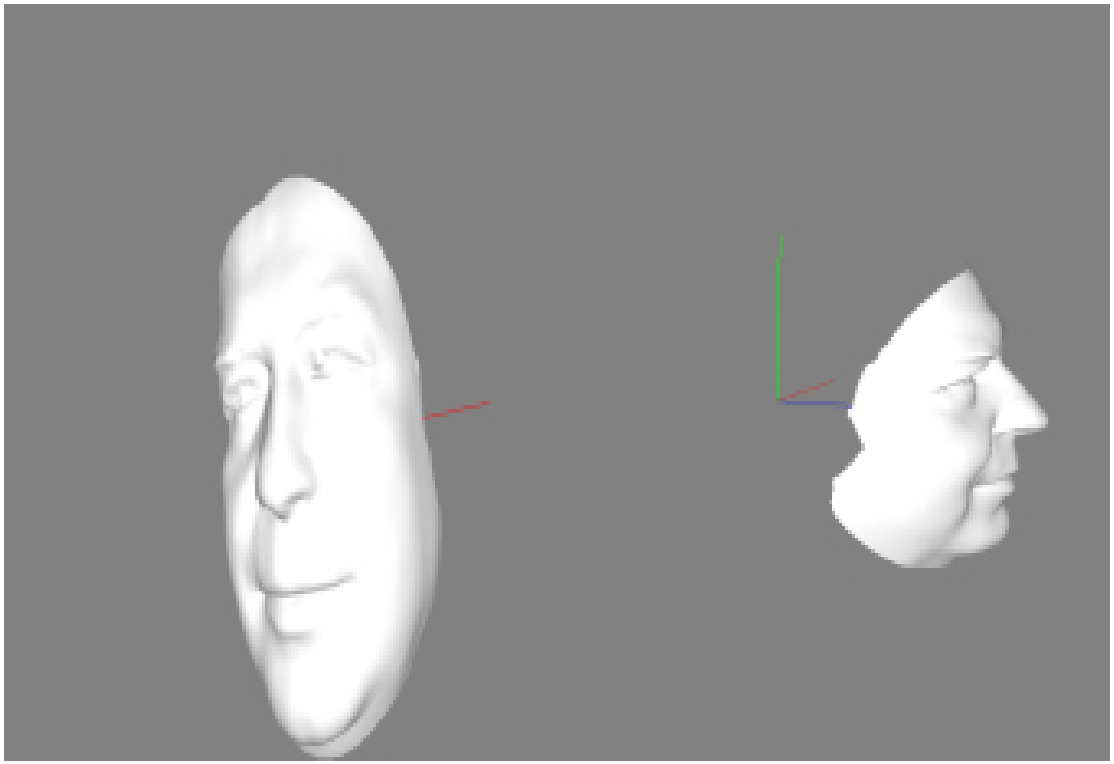
Hardware NNs consisting of a collection of simple neuron circuits provide the massive computational parallelism allowing for high speed real-time applications.



Distributed interactive haptic-visual virtual environment

3D Geometric Shape Acquisition

Laser Scanned 3D Images



XYZ RGB IMAGE IS EVERYTHING http://www.xyzrgb.com/ : 3D scanning services offered by XYZ RGB are used throughout the visual effects, video game and reverse engineering when superior visualization, quality, and process of manufacturing are required.

Pseudo-Random Multi Valued Sequence (PRMVS)
structured-light grid projected on a 3D object

NN Modelling of the Geometric and Elastic Properties of 3D Objects from Experimental Measurement Data

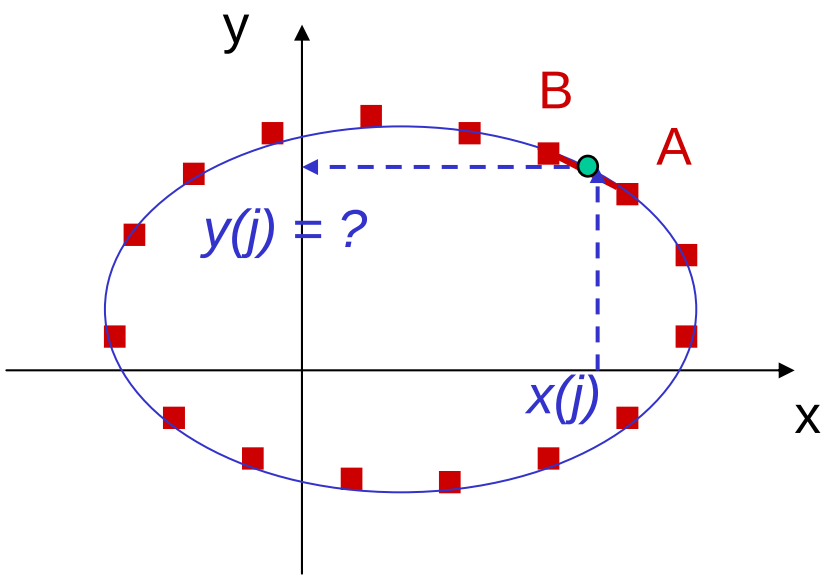
Modelling allows to *simulate the behavior of a system* for a variety of initial conditions,excitations and systems configurations

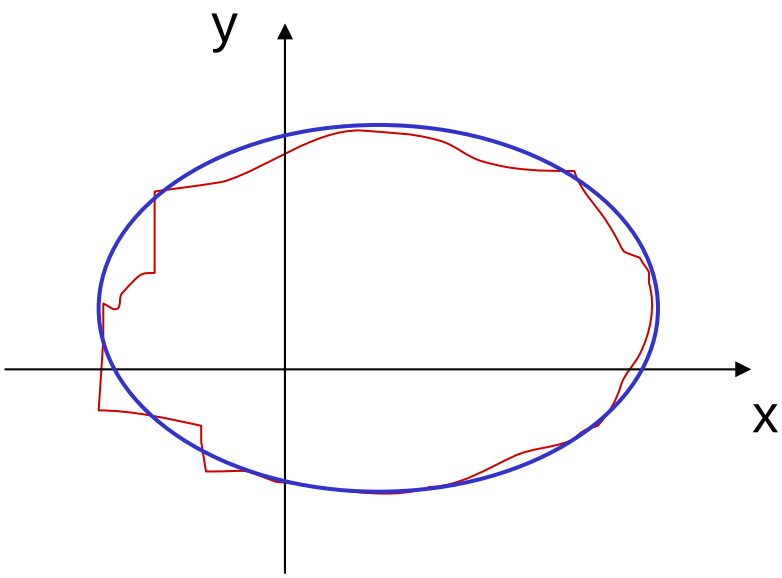
- The ***quality and the degree of the approximation*** of the model can be determined only by a validation against experimental measurements.

- The ***convenience*** of the model means that it is capable of performing extensive parametric studies, in which independent parameters describing the model can be varied over a specified range in order to gain a global understanding of the response.

E.M. Petriu, "Neural Networks for Measurement and Instrumentation in Virtual Environments," in *Neural Networks for Instrumentation, Measurement and Related Industrial Applications* , S. Ablameyko, L. Goras, M. Gori, V. Piuri - Eds.), NATO Science Series, Series III: Computer and System Sciences Vol. 185, pp.273-290, IOS Press, 2003

Discreet vs. Continuous Modelling of Physical Objects and Processes





DISCREET MODEL

- ***sampling*** *=> INTERPOLATION COST*

  y(j) = y(A) +

  [x(j)-x(B)] · [y(B)-x(A)] / [x(A)-x(B)]

CONTINUOUS MODEL

- *NO sampling =>*

  NO INTEPPOLATION COST

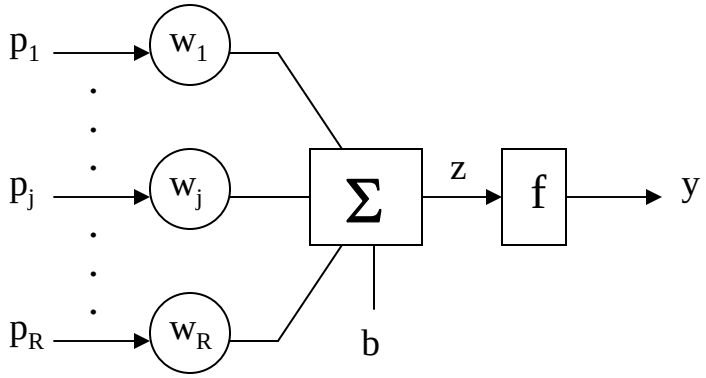
Neural Network vs.
Analog Computer Modelling

- Both the Analog Computers and Neural Networks are *continuous modelling devices.*

- Neural Networks don't require a prior mathematical models. A *learning algorithm* is used to adjust by trial and error during the learning phase the synaptic weights of the neurons.

Artificial Neural Networks (ANN)

McCulloch-Pitts model of an artificial neuron



$y = f (w_1 \cdot p_1 + \ldots + w_j \cdot p_j + \ldots w_R \cdot p_R + b)$

$y = f (\boldsymbol{W} \cdot \boldsymbol{p} + b)$

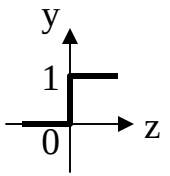
$\boldsymbol{p} = (p_1, \ldots , p_R)^T$ is the input column-vector

$\boldsymbol{W} = (w_1, \ldots , w_R)$ is the weight row-vector

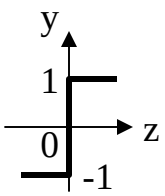
*) The bias *b* can be treated as a weight whose input is always 1.

Some transfer functions "**f**"

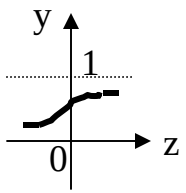
Hard Limit: y = 0 if z<0
y = 1 if z>=0



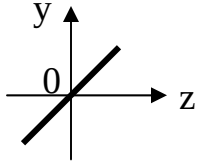
Symmetrical Hard Limit: y = -1 if z<0
y = +1 if z>=0



Log-Sigmoid:
$y = 1/(1+e^{-z})$



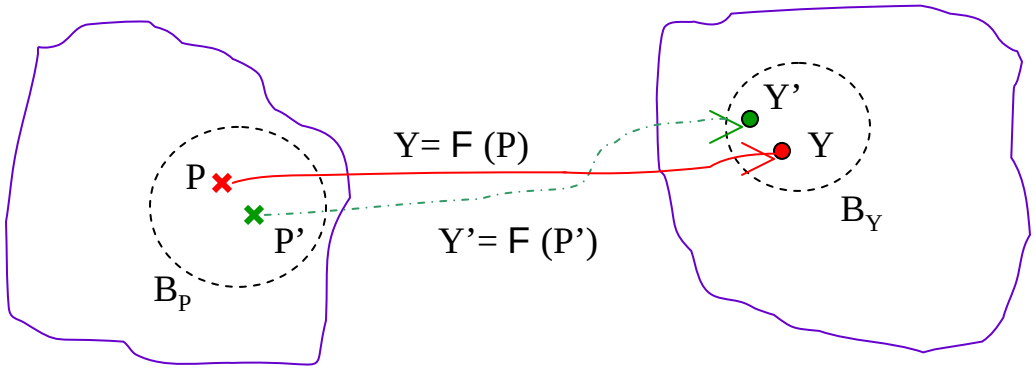
Linear:
y = z



The Architecture of an ANN

- Number of inputs and outputs of the network;
- Number of layers;
- How the layers are connected to each other;
- The transfer function of each layer;
- Number of neurons in each layer;

ANNs map input/stimulus values to output/response values: $Y = F(P)$.

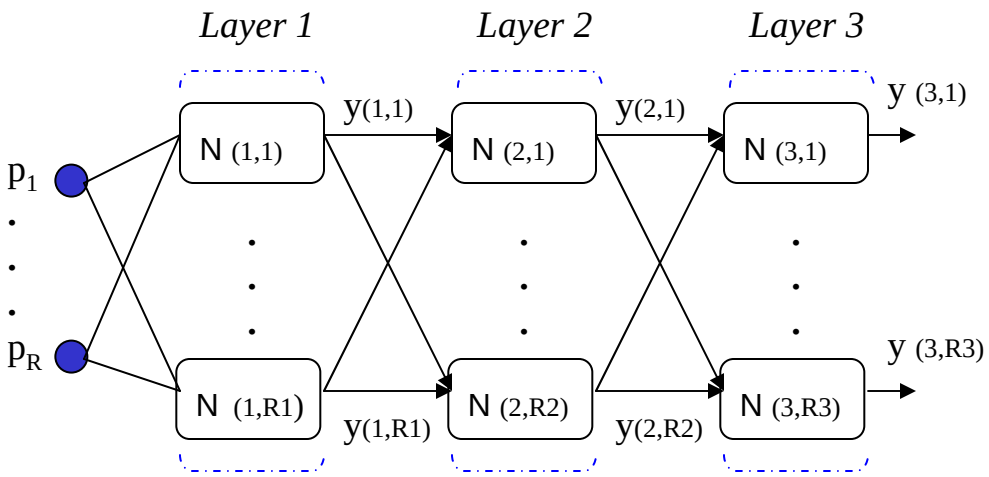


Intelligent systems generalize: their behavioral repertoires exceed their experience. An intelligent system is said to have a creative behaviour if it provides appropriate responses when faced with new stimuli. Usually the new stimuli *P'* resemble known stimuli *P* and their corresponding responses *Y'* resemble known/learned responses *Y*.

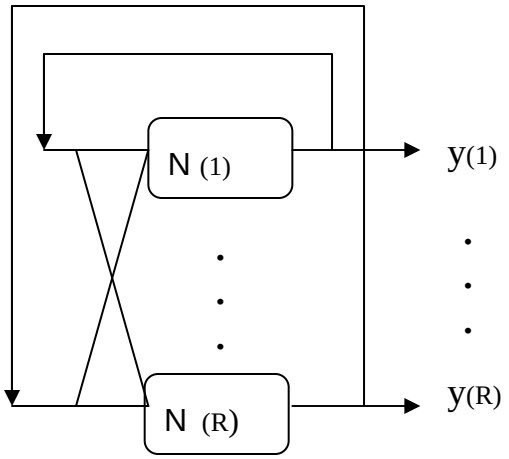
Measure of system's F creativity:

$$\frac{\text{Volume of "stimuli ball } B_P \text{ "}}{\text{Volume of "response ball } B_Y \text{"}}$$

- Most of the mapping functions can be implemented by a two-layer ANN: a sigmoid layer feeding a linear output layer.
- ANNs with biases can represent relationships between inputs and outputs than networks without biases.
- *Feed-forward* ANNs cannot implement temporal relationships. *Recurrent* ANNs have internal feedback paths that allow them to exhibit temporal behaviour.



Feed-forward architecture with three layers



Recurrent architecture (*Hopfield* NN)

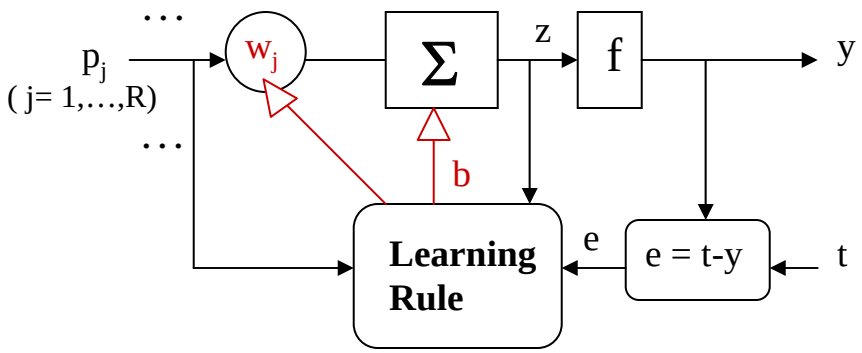
The ANN is usually supplied with an initial input vector and then the outputs are used as inputs for each succeeding cycle.

Learning Rules (Training Algorithms)

Procedure/algorithm to adjust the weights and biases in order for the ANN to perform the desired task.

Supervised Learning

For a given training set of pairs {***p***(1),t(1)},...,{***p***(n),t(n)}, where ***p***(i) is an instance of the input vector and **t**(i) is the corresponding *target* value for the output ***y***, the learning rule calculates the updated value of the neuron weights and bias.



Reinforcement Learning

Similar to supervised learning - instead of being provided with the correct output value for each given input, the algorithm is only provided with a given grade/score as a measure of ANN's performance.

Unsupervised Learning

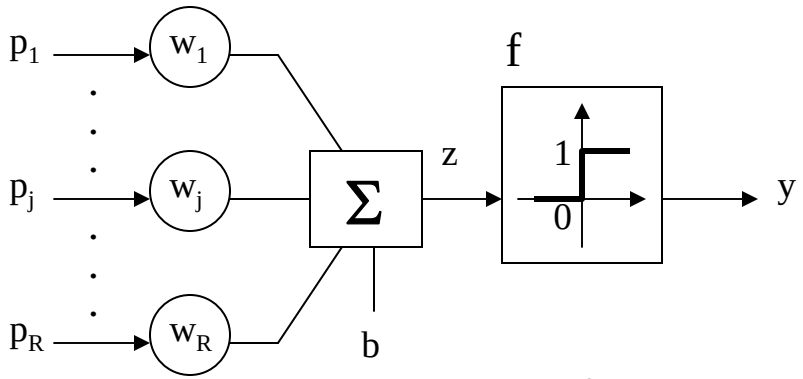
The weight and unbiased are adjusted based on inputs only. Most algorithms of this type learn to cluster input patterns into a finite number of classes. ==> e.g. vector quantization applications

THE PERCEPTRON

- Frank Rosenblatt (1958), Marvin Minski & Seymour Papert (1969)

- [Minski] “Perceptrons make decisions/determine whether or not event fits a certain pattern by adding up evidence obtained from many small experiments”

- The **perceptron** is a neuron with a hard limit transfer function and a weight adjustment mechanism (“learning”) by comparing the actual and the expected output responses for any given input /stimulus.



$y = f(\boldsymbol{W} \cdot \boldsymbol{p} + b)$

- Perceptrons are well suited for pattern classification/recognition.

- The weight adjustment/training mechanism is called the *perceptron learning rule*.

NB: $\boldsymbol{W}$ is a row-vector and $\boldsymbol{p}$ is a column-vector.

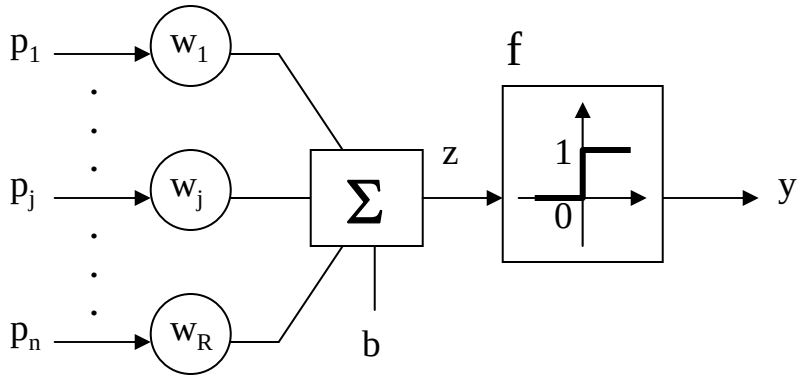
Perceptron Learning Rule

- **Supervised learning**

t <== the target value

e = t-y <== the error

Because of the perceptron's hard limit transfer function *y, t, e* can take only binary values



$\boldsymbol{p} = (p_1, \ldots, p_R)^T$ is the input column-vector

$\boldsymbol{W} = (x_1, \ldots, x_R)$ is the weight row-vector

Perceptron learning rule:

if e = 1, then $\boldsymbol{W}^{new} = \boldsymbol{W}^{old} + \boldsymbol{p}$, $b^{new} = b^{old} + 1$;

if e = -1, then $\boldsymbol{W}^{new} = \boldsymbol{W}^{old} - \boldsymbol{p}$, $b^{new} = b^{old} - 1$;

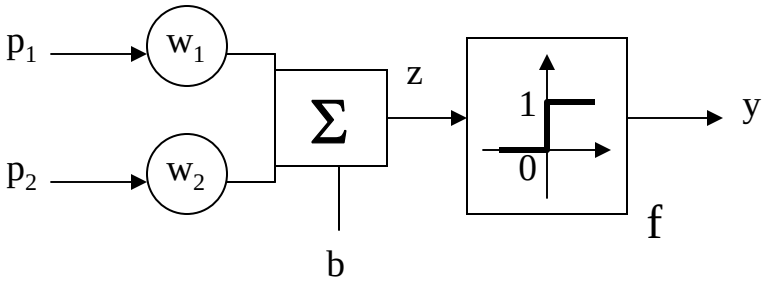
if e = 0, then $\boldsymbol{W}^{new} = \boldsymbol{W}^{old}$.

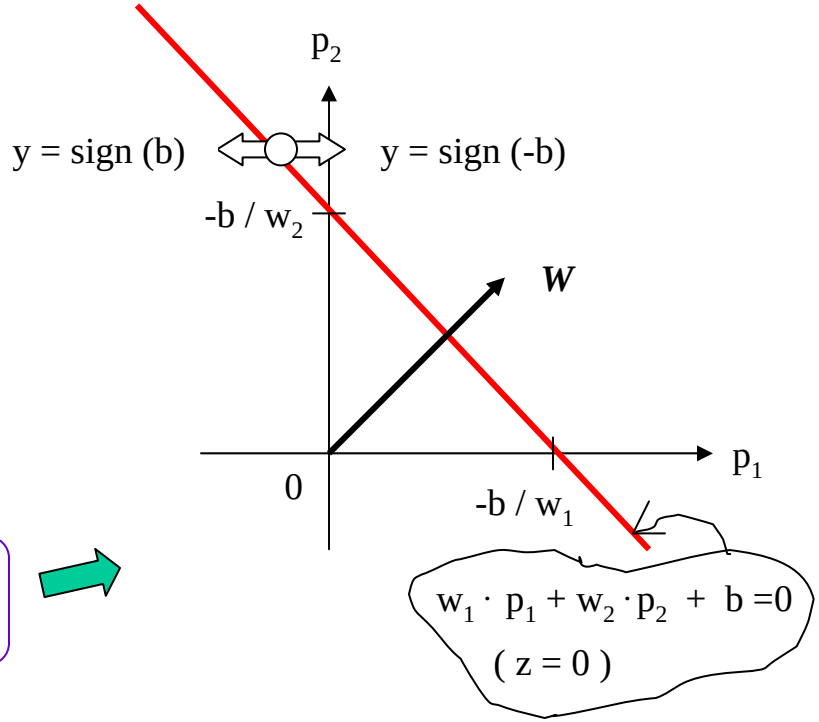
$$\boldsymbol{W}^{new} = \boldsymbol{W}^{old} + e \cdot \boldsymbol{p}^T$$

$$b^{new} = b^{old} + e$$

The hard limit transfer function (threshold function) provides the ability to classify input vectors by deciding whether an input vector belongs to one of two *linearly separable classes.*

Two-Input Perceptron





$y = hardlim\,(z) = hardlim\{ [w_1 , w_2] \cdot [p_1 , p_2]^T + b \}$

- The two classes (linearly separable regions) in the two-dimensional input space (p_1, p_2) are separated by the line of equation $z = 0$.
- The boundary is always orthogonal to the weight vector ***W***.

- The larger an input vector ***p*** is, the larger is its effect on the weight vector ***W*** during the learning process

  Long training times can be caused by the presence of an "outlier," i.e. an input vector whose magnitude is much larger, or smaller, than other input vectors.

  Normalized perceptron learning rule, the effect of each input vector on the weights is of the same magnitude:

$$\boldsymbol{W}^{new} = \boldsymbol{W}^{old} + e\cdot\boldsymbol{p}^{T}/\|\boldsymbol{p}\|$$

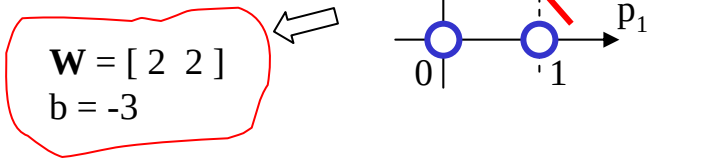
$$b^{new} = b^{old} + e$$

Perceptron Networks for Linearly Separable Vectors

- The hard limit transfer function of the perceptron provides the ability to classify input vectors by deciding whether an input vector belongs to one of two *linearly separable classes.*

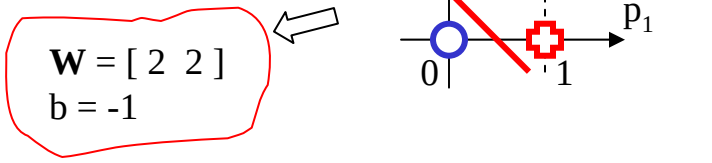
$$\mathbf{p} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

$$\mathbf{t}_{AND} = [\,0\ 0\ 0\ 1\,]$$



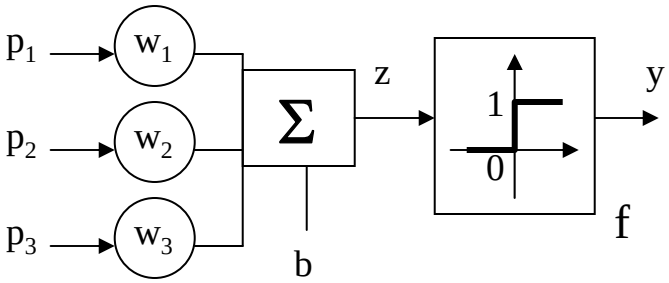
$$\mathbf{p} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

$$\mathbf{t}_{OR} = [\,0\ 1\ 1\ 1\,]$$



Three-Input Perceptron

✔ The two classes in the 3-dimensional input space (p_1, p_2, p_3) are separated by the plane of equation $z = 0$.

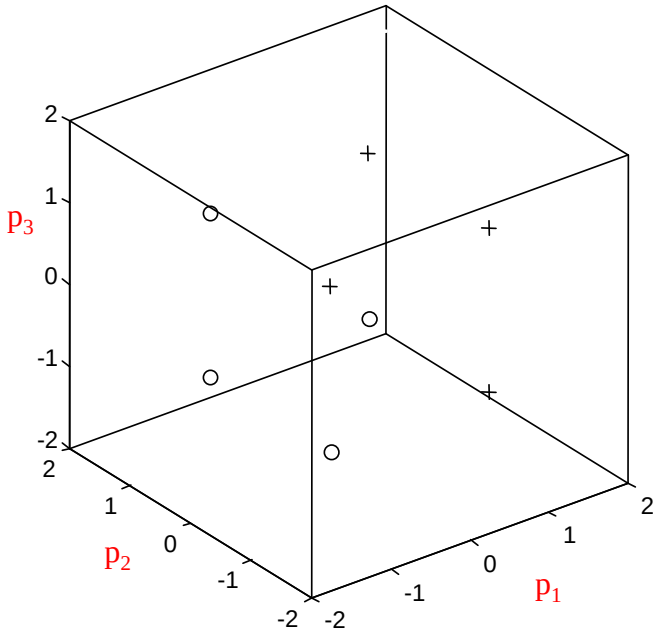


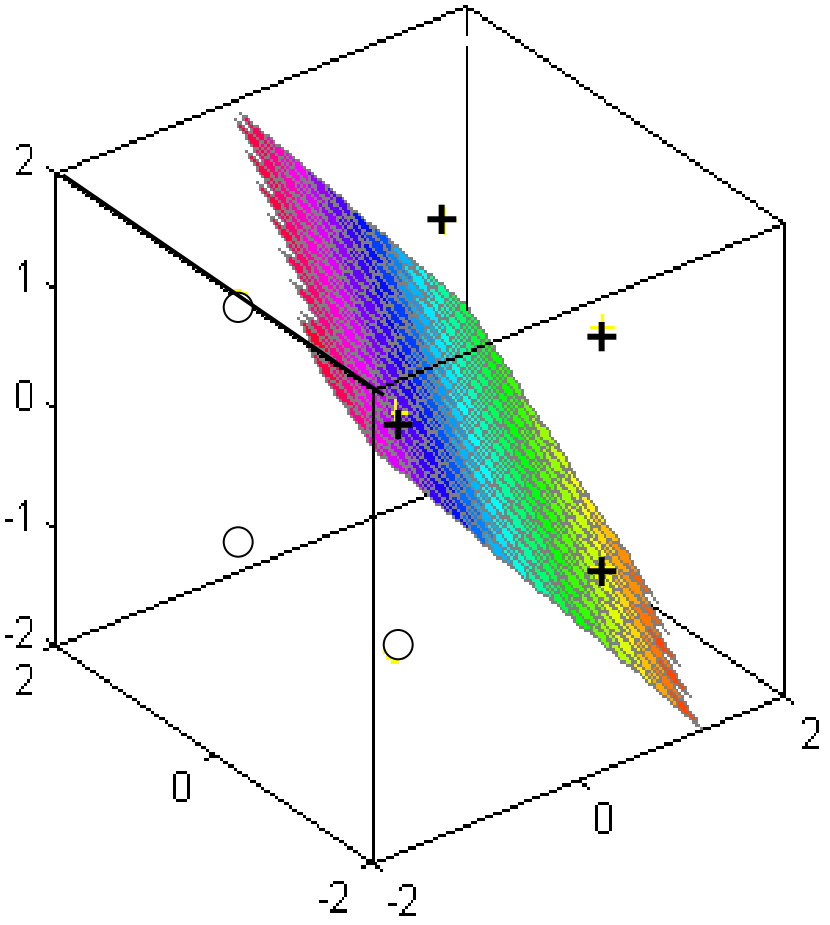
$$y = hardlim(z) = hardlim\{[w_1, w_2, w_3] \cdot [p_1, p_2 p_3]^T + b\}$$

EXAMPLE

```
P = [ -1  1  1  -1 -1  1  1 -1;
      -1  -1  1  1  -1 -1  1  1;
      -1  -1 -1 -1   1  1  1   1]
```

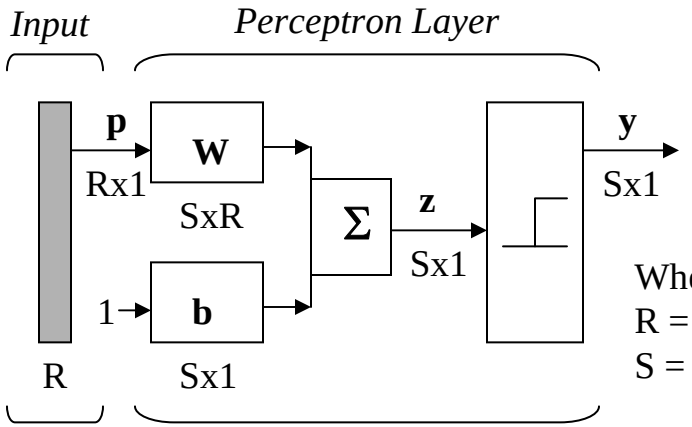
```
T = [ 0 1 0 0 1 1 1 0 ]
```





One-layer multi-perceptron classification of linearly separable patterns

MATLAB representation:

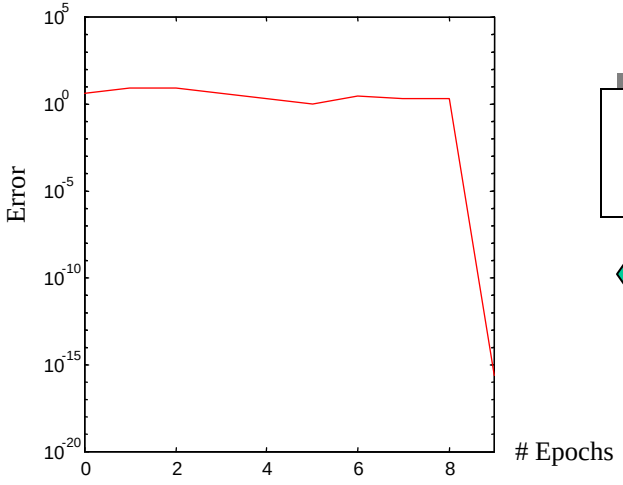


Demo P3 in the "MATLAB Neural Network Toolbox - User's Guide"

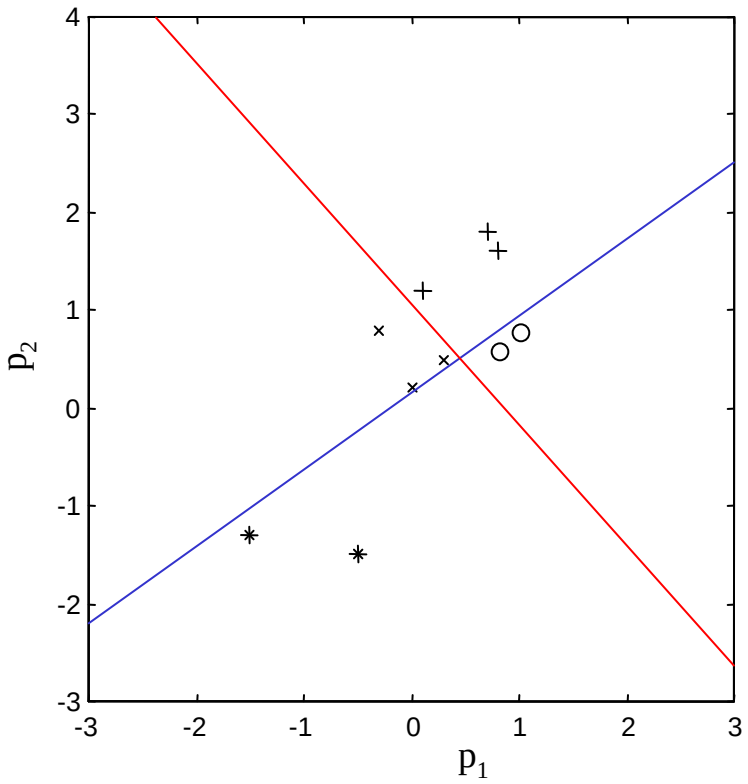
P = [0.1 0.7 0.8 0.8 1.0 0.3 0.0 -0.3 -0.5 -1.5;
1.2 1.8 1.6 0.6 0.8 0.5 0.2 0.8 -1.5 -1.3]

T = [1 1 1 0 0 1 1 1 0 0;
0 0 0 0 0 1 1 1 1 1]

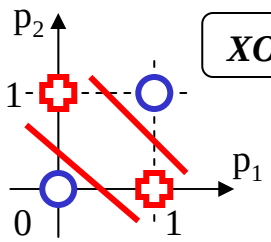
{ 00 = O ; 10 = +
01 = * ; 11 = x }



R = 2 inputs
S = 2 neurons



Perceptron Networks for Linearly Non-Separable Vectors



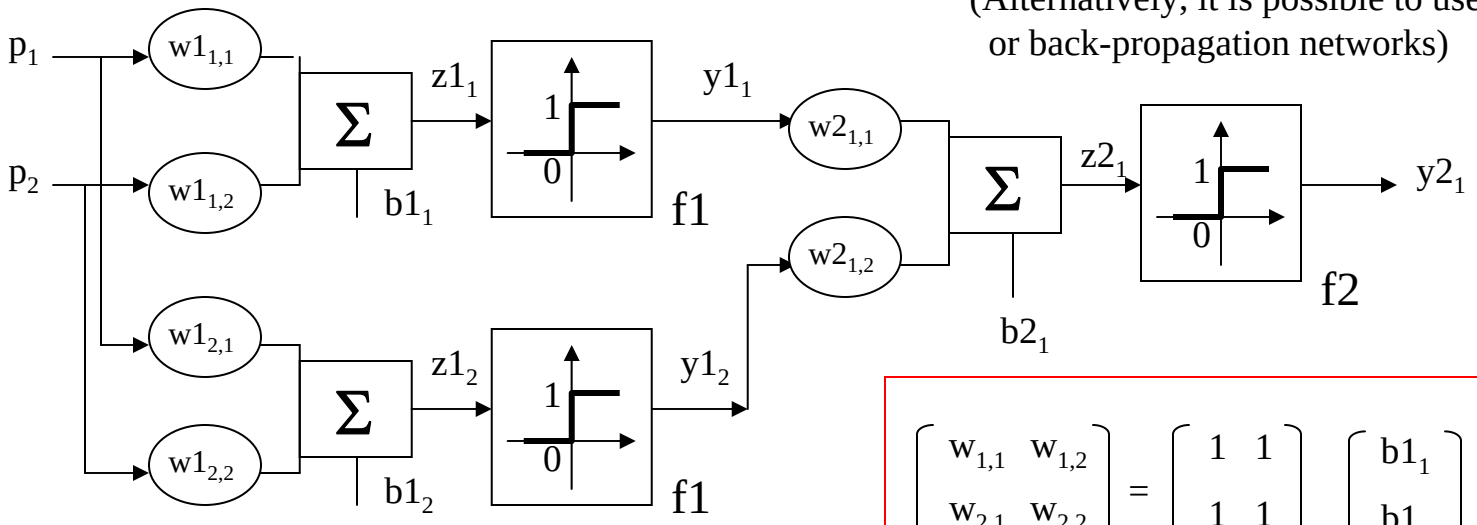
XOR

$$\mathbf{p} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

$$\mathbf{t}_{XOR} = [\,0 \;\; 1 \;\; 1 \;\; 0\,]$$

If a straight line cannot be drawn between the set of input vectors associated with targets of 0 value and the input vectors associated with targets of 1, than a perceptron cannot classify these input vectors.

One solution is to use a two layer architecture, the perceptrons in the first layer are used as preprocessors producing linearly separable vectors for the second layer.
(Alternatively, it is possible to use linear ANN or back-propagation networks)

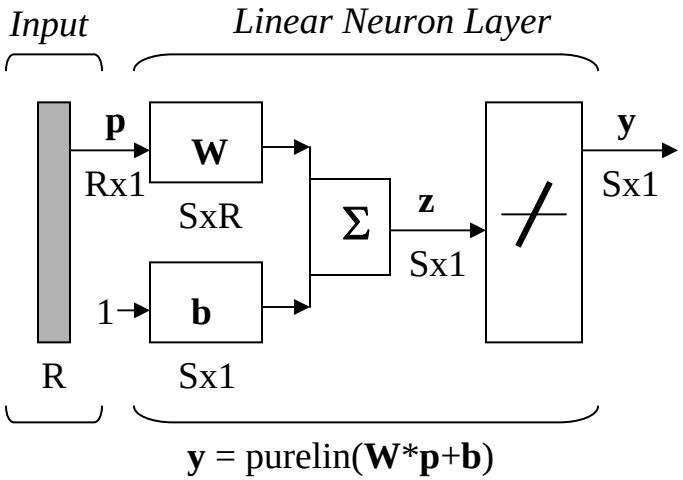


$$\begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \qquad \begin{bmatrix} b1_1 \\ b1_2 \end{bmatrix} = \begin{bmatrix} -1.5 \\ -0.5 \end{bmatrix}$$

$$[\,w2_{1,1} \;\; w2_{1,2}\,] = [-1 \;\; 1] \qquad [\,b2_1\,] = [-0.5]$$

The row index of a weight indicates the destination neuron of the weight and the column index indicates which source is the input for that weight.

LINEAR NEURAL NETWORKS (*ADALINE* NETWORKS)

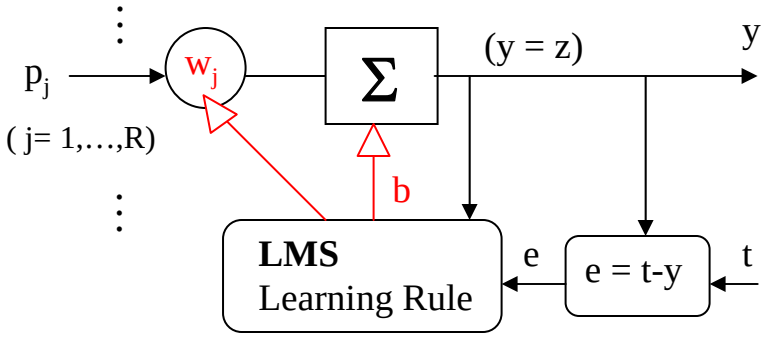


- Linear neurons have a *linear transfer function*that allows to use a Least Mean-Square (LMS) procedure - *Widrow-Hoff learning rule*- to adjust weights and biases according to the magnitude of errors.
- Linear neurons suffer from the same limitation as the perceptron networks: they can only solve *linearly separable problems*.

Where: R = # Inputs, S = # Neurons

(*ADALINE* <== *ADA*ptive *LI*near *NE*uron)

Widrow-Hoff Learning Rule (The ☺ Rule)



The LMS algorithm will adjust ADALINE's weights and biases in such away to *minimize the <u>mean-square-error</u> E [e^2] between all sets of the desired response and network's actual response*:

$$E[(t-y)^2] = E[(t - (w_1 \dots w_R \ b) \cdot (p_1 \dots p_R \ 1)^T)^2] = E[(t - \mathbf{W} \cdot \mathbf{p})^2]$$

(<u>NB</u>: E[…] denotes the "expected value"; **p** is column vector)

$E[\,e^2\,] = E[\,(t - \mathbf{W}\cdot\mathbf{p})^2\,] = \{$as for deterministic signals the expectation becomes a time-average$\}$

$= E[t^2] - 2\cdot\mathbf{W}\cdot E[t\cdot\mathbf{p}] + \mathbf{W}\cdot E[\mathbf{p}\cdot\mathbf{p}^T]\cdot\mathbf{W}^T$

The cross-correlation between the input vector and its associated target.

The input cross-correlation matrix

If the **input correlation matrix is positive** the LMS algorithm will converge as there will a *unique minimum of the mean square error*.

- The W-H rule is an iterative algorithm uses the "steepest-descent" method to reduce the mean-square-error. The key point of the W-H algorithm is that it replaces $E[e^2]$ estimation by the squared error of the iteration *k*: $e^2(k)$. At each iteration step *k* it estimates the gradient of this error ∇_k with respect to **W** as a vector consisting of the partial derivatives of $e^2(k)$ with respect to each weight:

$$\nabla_k^* = \frac{\partial e^2(k)}{\partial W(k)} = \left[\frac{\partial e^2(k)}{\partial w_1(k)} \cdots \frac{\partial e^2(k)}{\partial w_R(k)}, \frac{\partial e^2(k)}{\partial b(k)}\right]$$

The weight vector is then modified in the direction that decreases the error:

$$W(k+1) = W(K) - \mu \bullet \nabla_k^* = W(k) - \mu \bullet \frac{\partial e^2(k)}{\partial W(k)} = W(k) - 2\mu \bullet e(k) \bullet \frac{\partial e(k)}{\partial W(k)}$$

- As t(k) and p(k) - both affecting e(k) - are independent of **W**(k), we obtain the final expression of the **Widrow-Hoff learning rule**:

$$\mathbf{W}(k+1) = \mathbf{W}(k) + 2\cdot\mu\cdot e(k)\cdot\mathbf{p}(k) \quad \Rightarrow \quad b(k+1) = b(k) + 2\cdot\mu\cdot e(k)$$

where μ the "learning rate" and $e(k) = t(k) - y(k) = t(k) - \mathbf{W}(k)\cdot\mathbf{p}(k)$

Back-Propagation Learning - The Generalized ❸ Rule

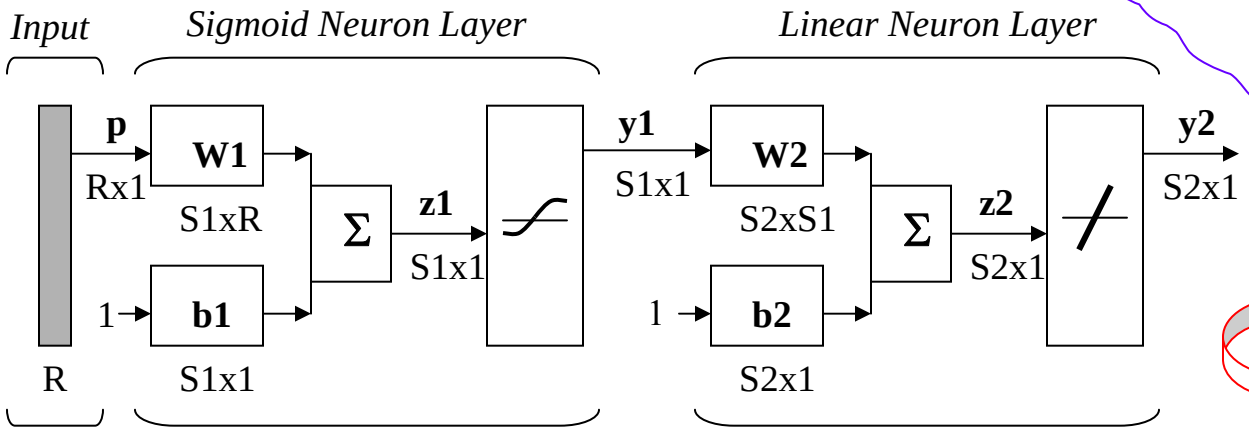
P. Werbos (Ph.D. thesis 1974);
D. Parker (1985), Yann Le Cun(1985),
D. Rumelhart, G. Hinton, R. Williams (1986)

❑ Single layer ANNs are suitable to only solving linearly separable classification problems. Multiple feed-forward layers can give an ANN greater freedom. Any reasonable function can be modeled by a two layer architecture: a sigmoid layer feeding a linear output layer.

❑ Single layer ANNs are only able to solve linearly Widrow-Hoff learning applies to single layer networks. ==> generalized W-H algorithm (Δ -rule) ==> **back-propagation learning**.

❑ Back-propagation ANNs often have one or more hidden layers of sigmoid neurons followed by an output layer of linear neurons.

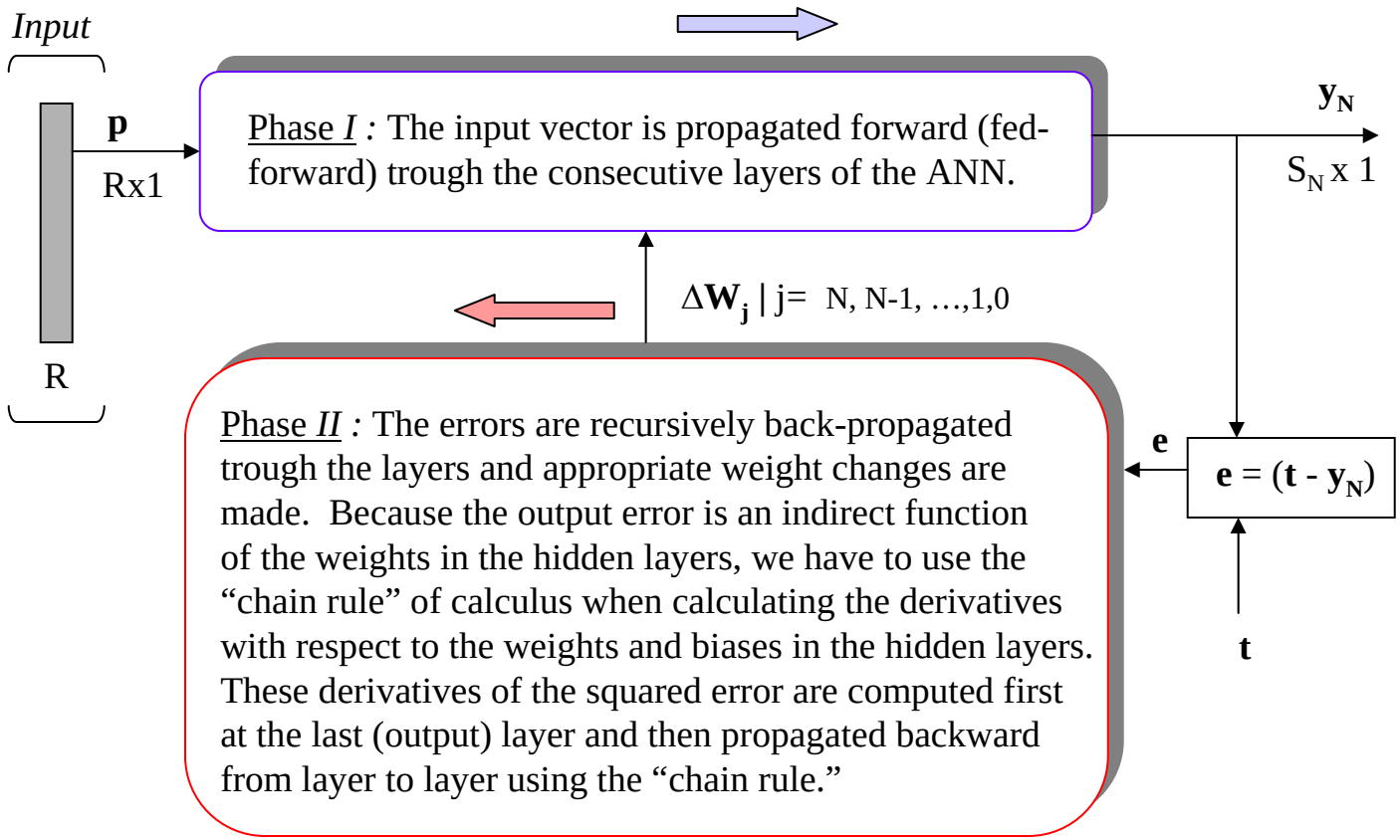
Two-layer ANN that can approximate any function with a finite number of of discontinuities, arbitrarily well, given sufficient neurons in the hidden layer.



e2= (t-y2) = (t- purelin (**W2***tansig(**W1*p** +**b1**) +**b2**))

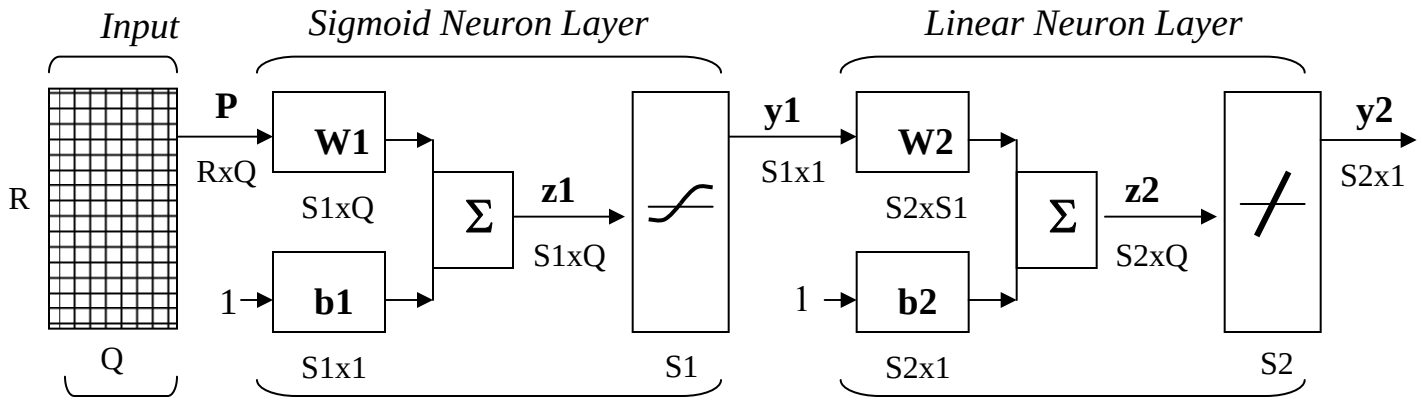
The error is an indirect function of the weights in the hidden layers.

- Back-propagation is an iterative steepest descent algorithm, in which the performance index is the mean square error E [e^2] between the desired response and network's actual response:

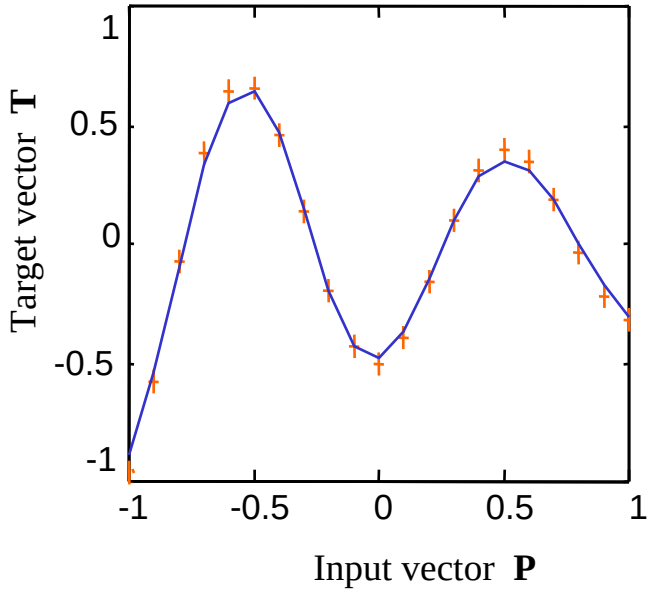


EXAMPLE: **Function Approximation by Back-Propagation**

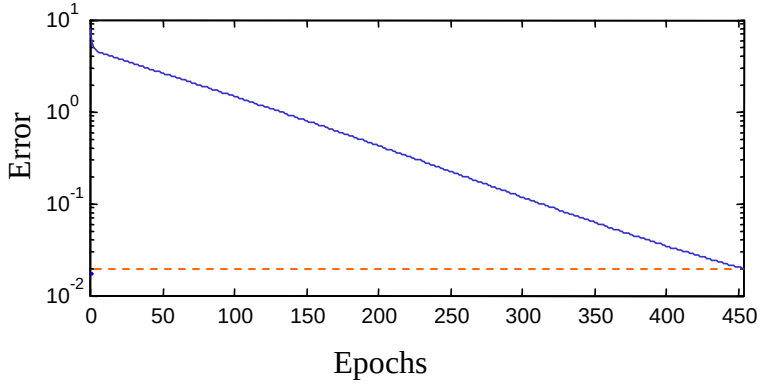
Demo BP4 in the" MATLAB Neural Network Toolbox User's Guide"



R = 1 input
S1 = 5 neurons in layer #1
S2 = 1 neuron in layer #2
Q = 21 input vectors



The back-propagation algorithm took 454 epochs to approximate the 21 target vectors with an error < 0.02

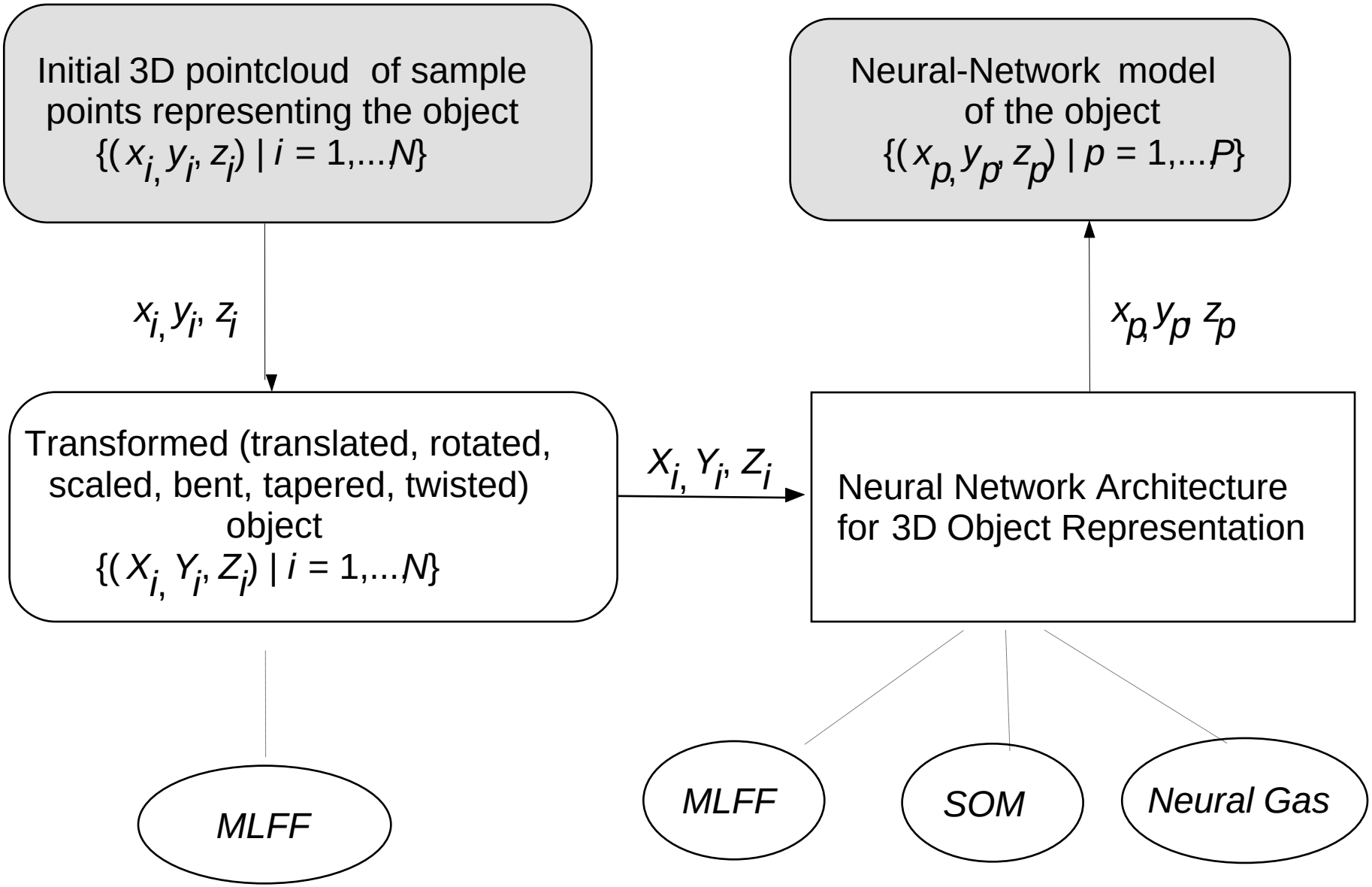


NN Modelling of 3D Object Shape

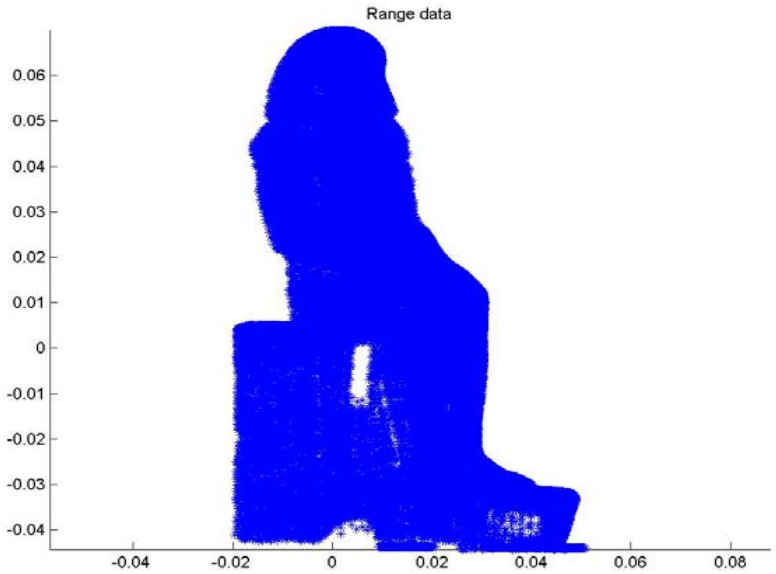
Compare the performance of three NN architectures used for 3D object shape modelling:

- Multilayer Feedforward (*MLFF*)
- Self-Organizing Map (*SOM*)
- Neural Gas Network

A.-M. Cretu, E.M. Petriu, G.G. Patry, “Neural-Network-Based Models of 3-D Objects for Virtualized Reality: A Comparative Study**,”** *IEEE Trans. Instrum. Meas.,"* Vol. 55, No. 1, pp.99-111, 2006.

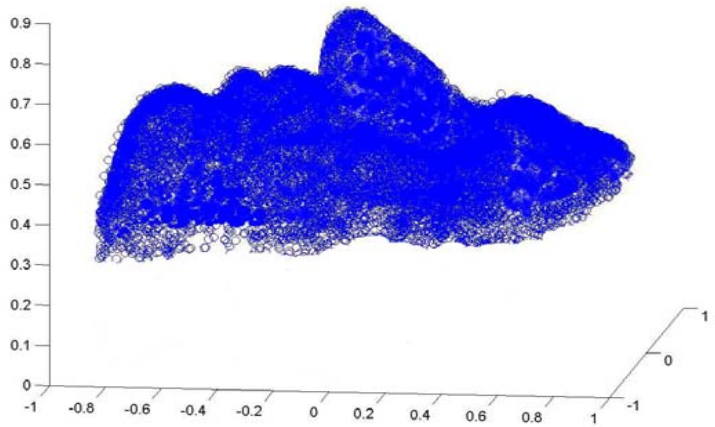

Initial 3D pointcloud of sample points representing the object
$\{(x_i, y_i, z_i) \mid i = 1,...,N\}$
Neural-Network model of the object
$\{(x_p, y_p, z_p) \mid p = 1,...,P\}$
x_i, y_i, z_i
x_p, y_p, z_p
Transformed (translated, rotated, scaled, bent, tapered, twisted) object
$\{(X_i, Y_i, Z_i) \mid i = 1,...,N\}$
X_i, Y_i, Z_i
Neural Network Architecture for 3D Object Representation
MLFF
MLFF
SOM
Neural Gas

MLFF Representation - Results

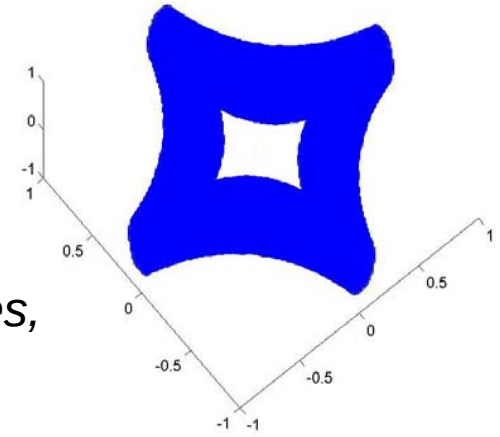


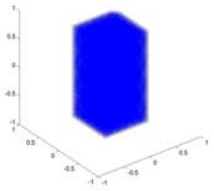
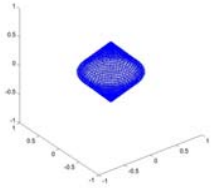
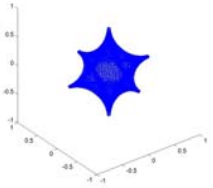
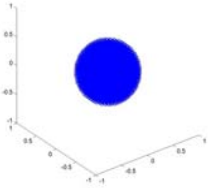
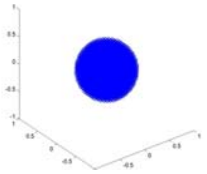
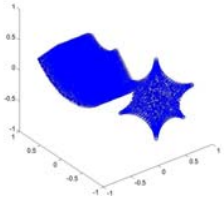
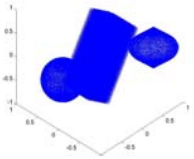
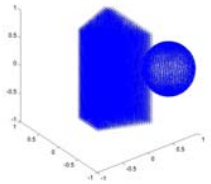
51096 points, 20-10-1,
5 extra surfaces,
d=0.055, 2000 epochs,
5.2 hrs.

19000 points, 14-7-1,
4 extra surfaces,
d=0.055, 1100 epochs,
3.3 hrs



2500 points, 12-6-1,
2 extra surfaces,
d=0.06,
1020 epochs,
45 min.



				
	2.4%	4.9%	1.6%	99.1%
	93.3%	3.3%	91.7%	3.1%
	95%	99.1%	5.78%	98.3%
	91.7%	6.6%	1.6%	92.5%

MLFF Neural Network Modelling – Summary

Advantages

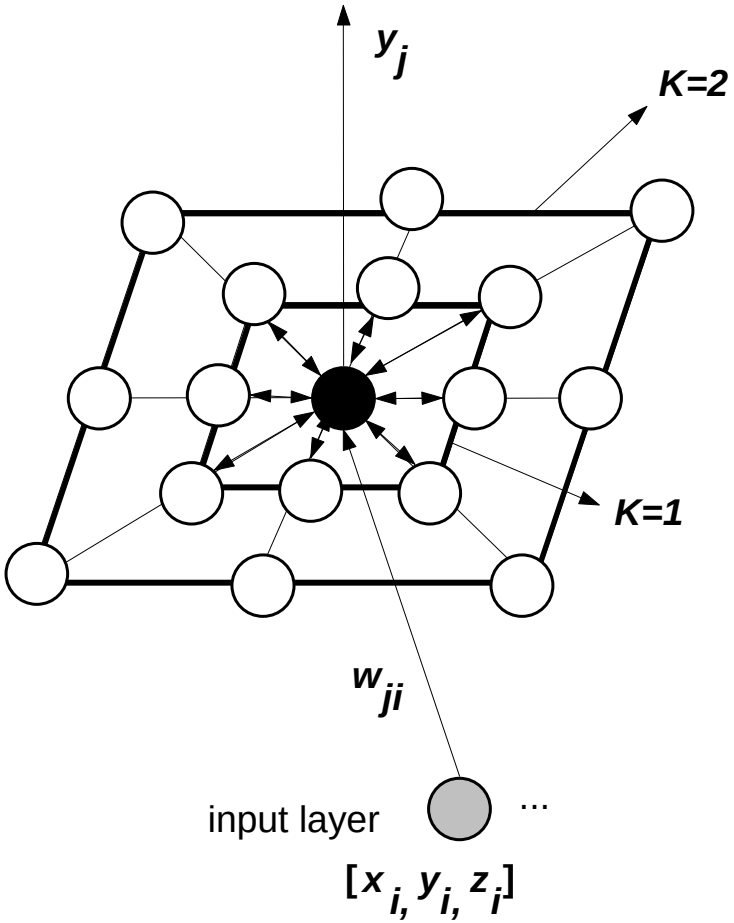
- simple and compact architecture
- continuous volumetric model (though trained with surface)
- information about the entire object space
- provides desired accuracy
- represents objects of varied complexity
- preserves details
- morphing, set operations, recognition, collision detection

Disadvantages

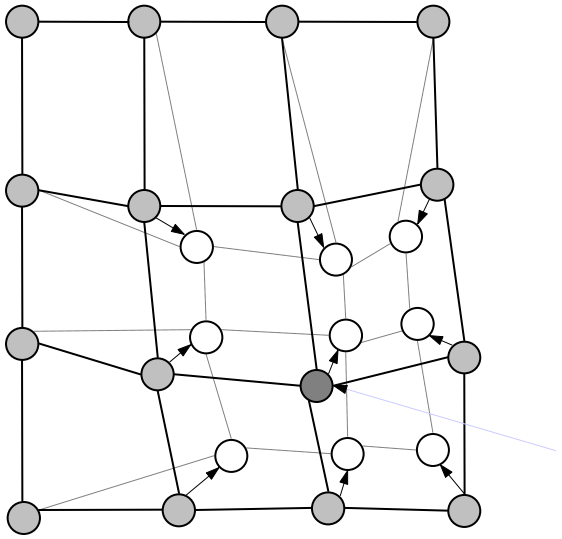
- computationally intensive (for both learning and rendering)
- lack of local control of the object

A **Self-Organizing Map (SOM)** has a input layer, where vector containing the *x,y,z* coordinates of points will be presented. Computations are feedforward in the first layer and bidirectional in the connection (output layer). The purpose of the lateral connection (represented by a Gaussian) is to reinforce the activation values of strong neurons and decrease the activation of weak ones. The output neurons are arranged on a grid, usually 2D. The network is based on competition - soft competition. The winning neurons is the one closest to the input vector. Its value and the value of a given neighbourhood are updated during learning. After learning, two vectors belonging to the same cluster will be projected on two close neurons in the output space.

Self-Organizing Map (*SOM*) NN Architecture

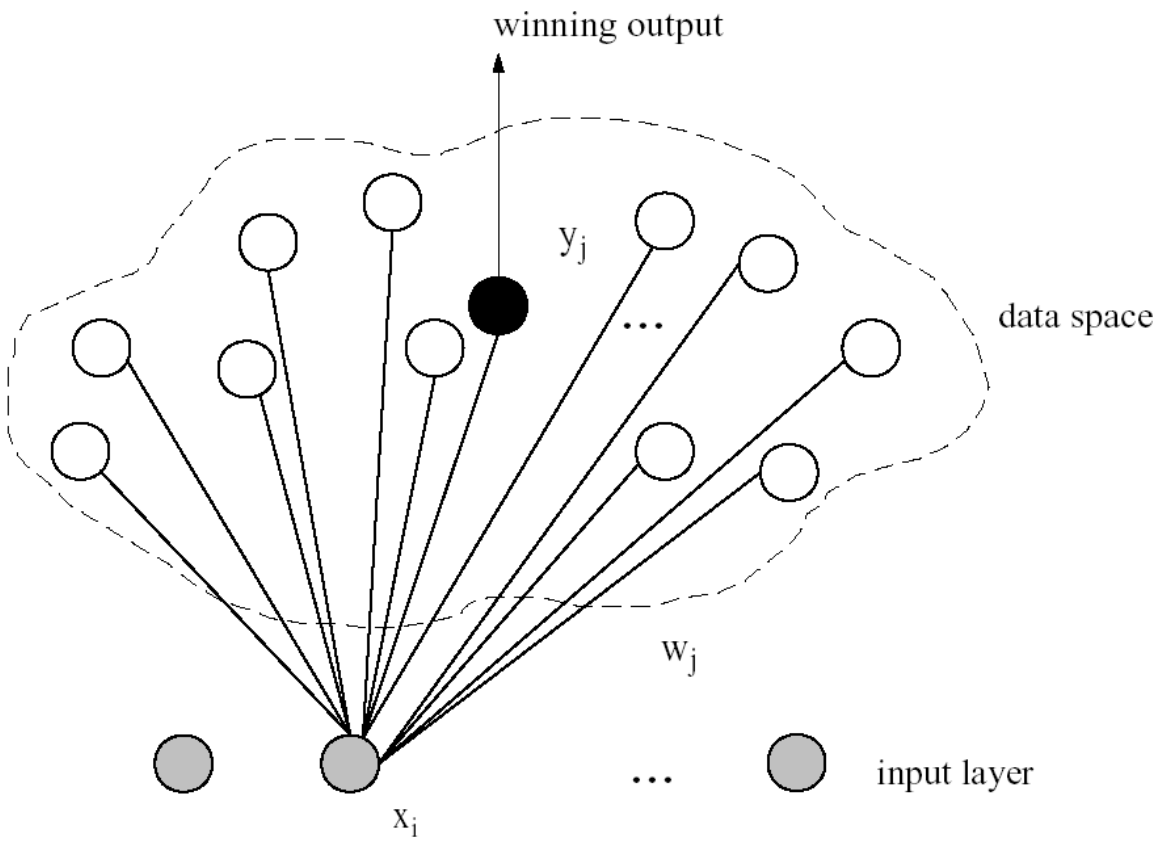


- **Activation Function**
  - soft competition
- **Learning**
  - unsupervised

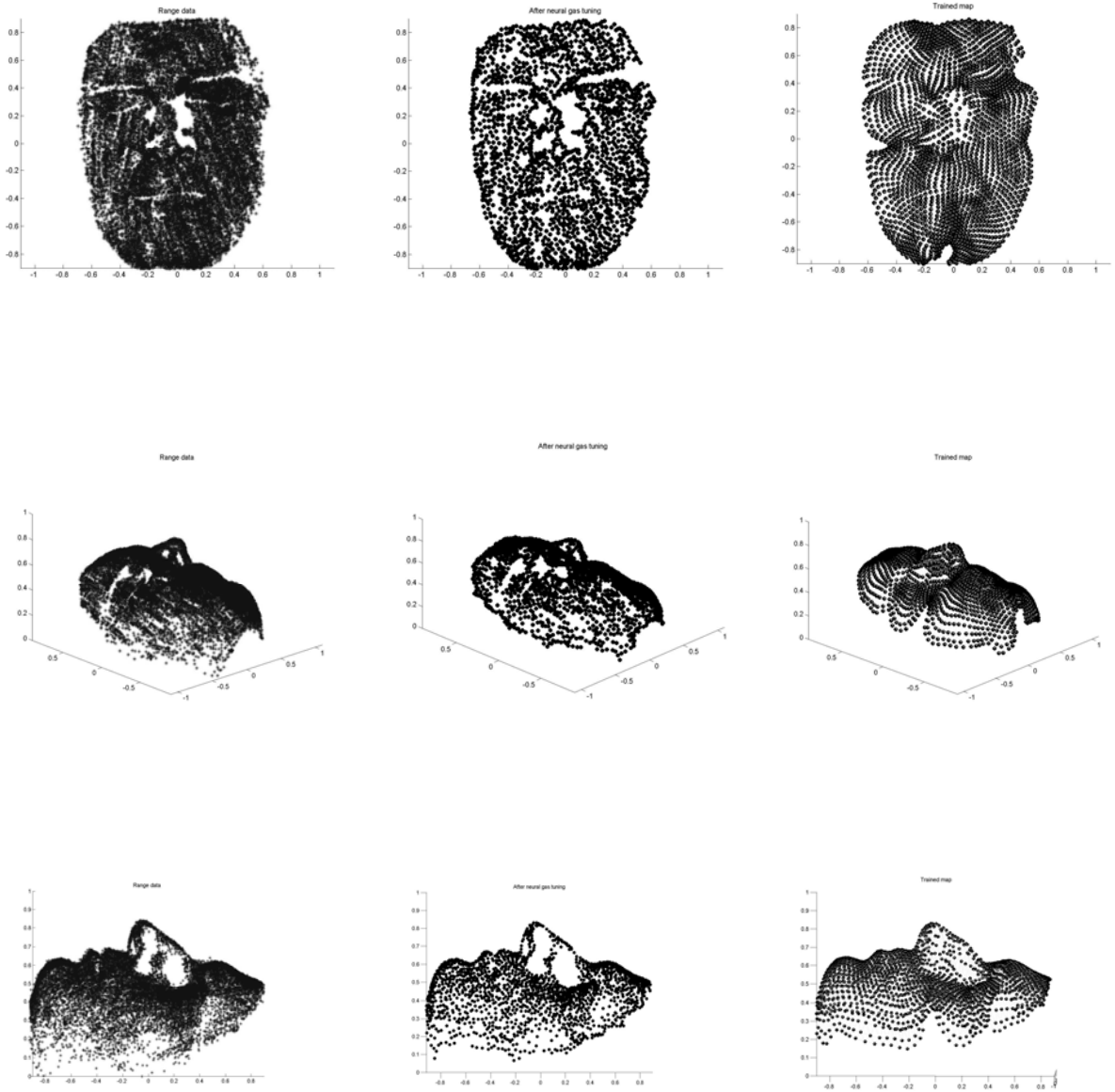


The **Neural Gas NN** => There are no connections between units in the connection layer. The nodes move independently over the data space. It also exploits the idea of soft competition, but the neurons to be updated are not selected according to topological relation, but rather according to rank. The rank is the rank the neurons have in an ordered list of distances between their weights and the input vector. NG converges quickly and to a lower distortion error.

Neural Gas Representation – NN Architecture



- **Activation Functions**:
  - soft competition
  - neighbourhood ranking
- **Learning**
  - unsupervised



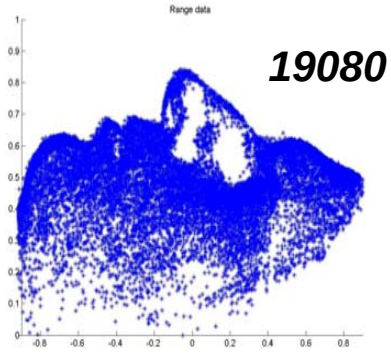
• *The first column* presents three views of the **original point-cloud** of **19,080 points** representing a human face.

• *The second column* presents the compressed model of **1,152 points** obtained using the **Neural Gas** network.

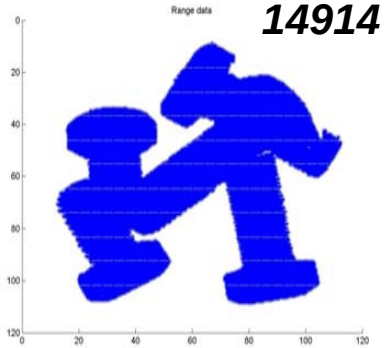
• *The third column* presents the compressed model of **1,152 points** obtained using **SOM**.

SOM and Neural Gas Modelling - Results

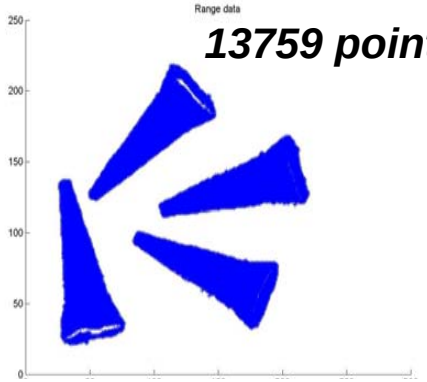
Initial point-cloud



19080 points

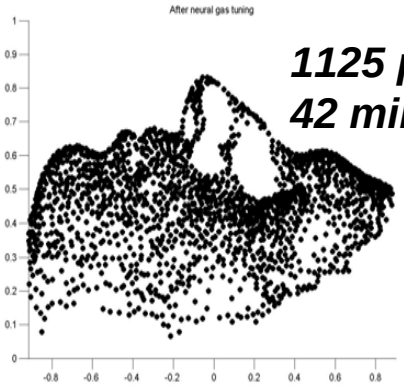


14914 points

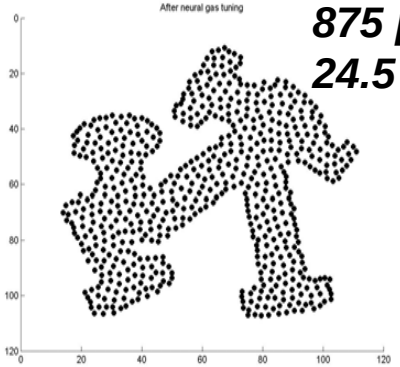


13759 points

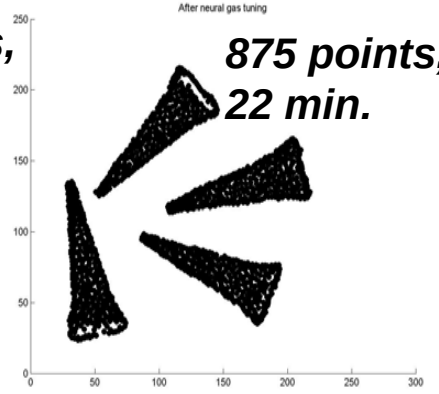
Neural Gas



1125 points, 42 min.



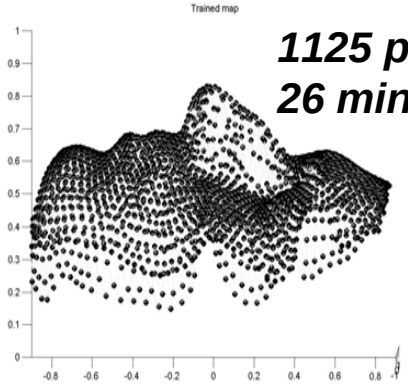
875 points, 24.5 min.



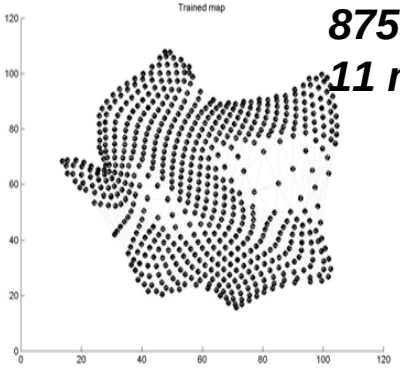
875 points, 22 min.

er= 0.0098

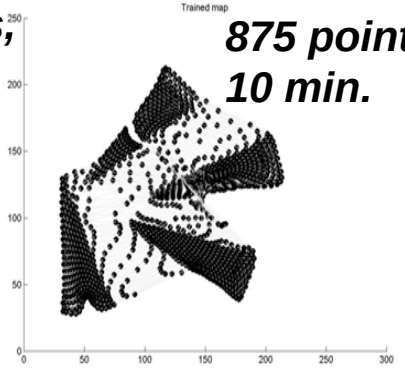
SOM



1125 points, 26 min.



875 points, 11 min.

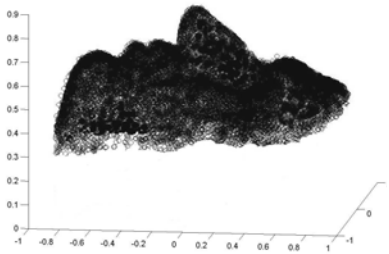


875 points, 10 min.

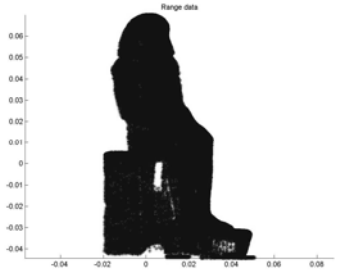
er= 0.0125

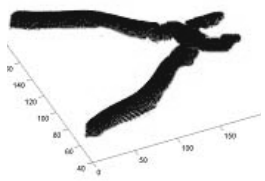
MLFF, Self-Organizing Map and *Neural Gas* Modelling
– Performance Comparison: Training Time -

MLFF

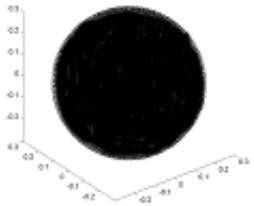


3.3 hrs

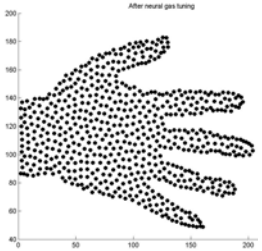


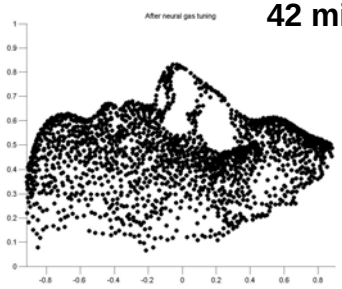


1 hrs

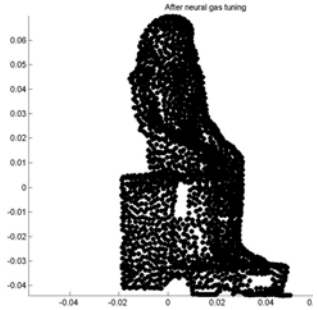


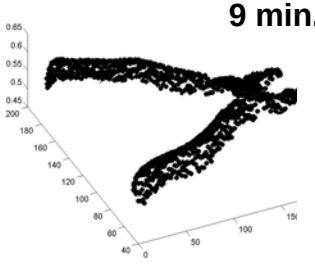
Neural Gas



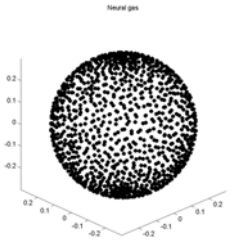


42 min.



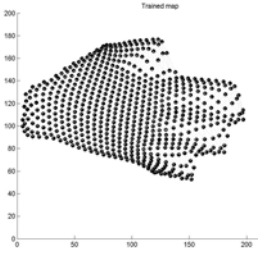


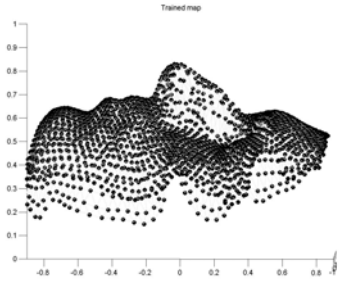
9 min.

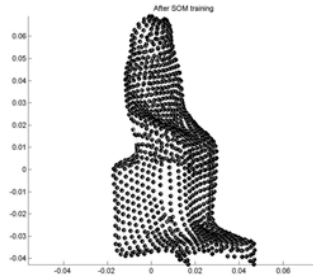


SOM

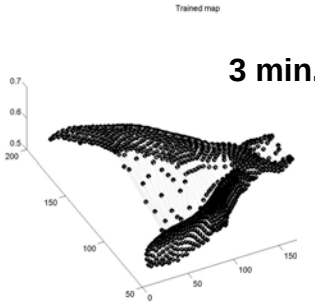
26 min.

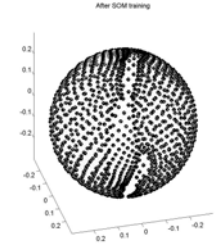




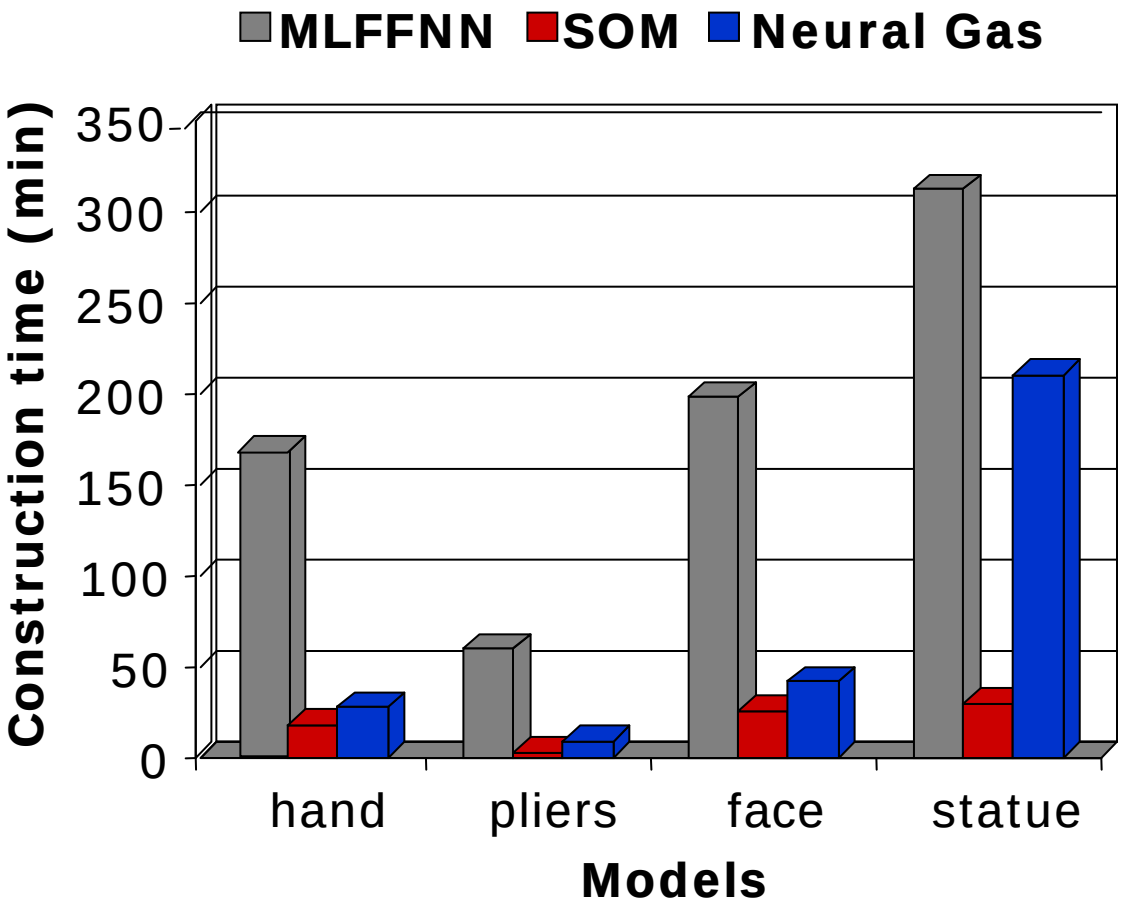


3 min.





MLFF, SOM, and *Natural Gas* Modelling
Performance Comparison: Construction Time



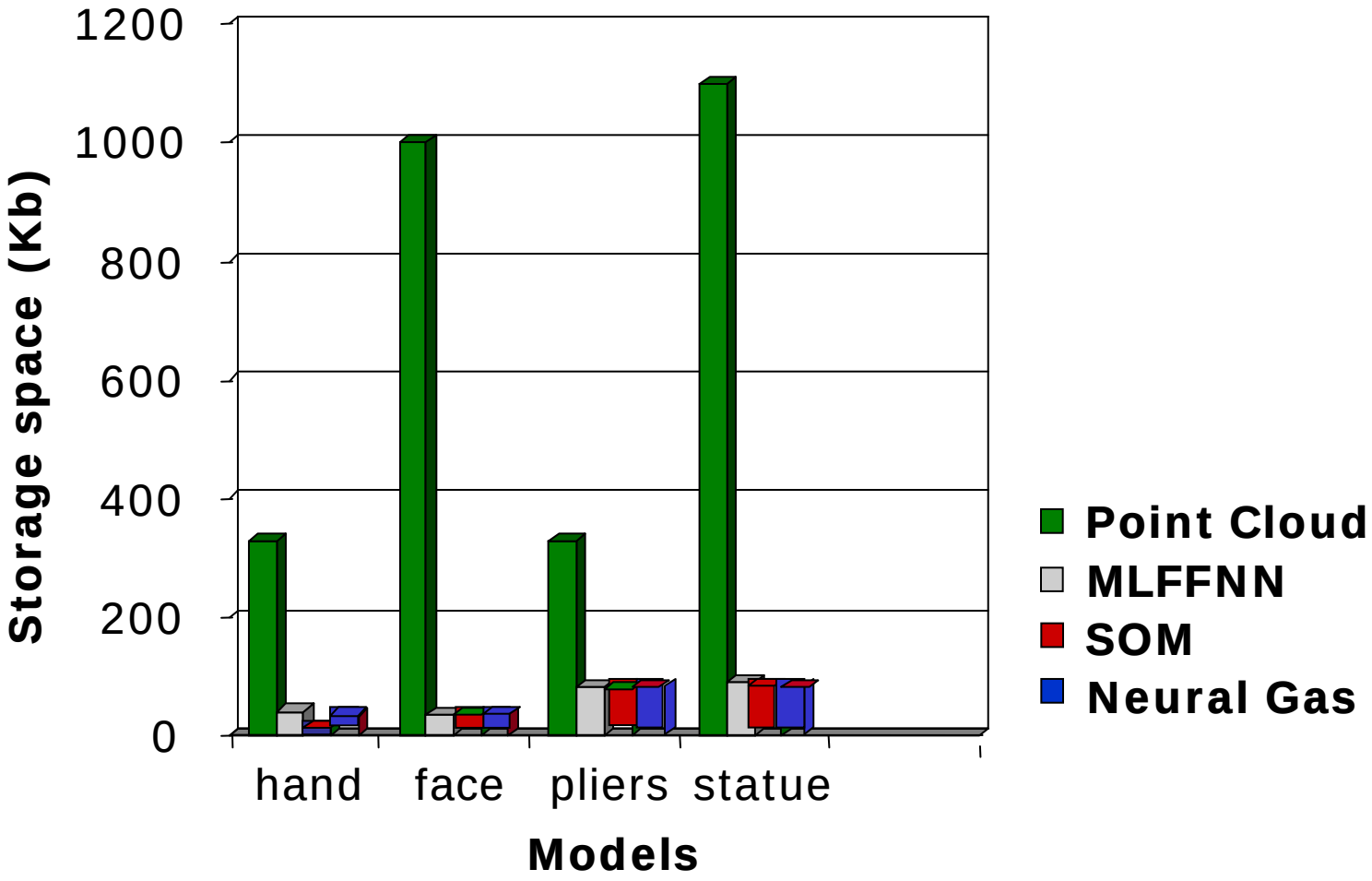
MLFNN

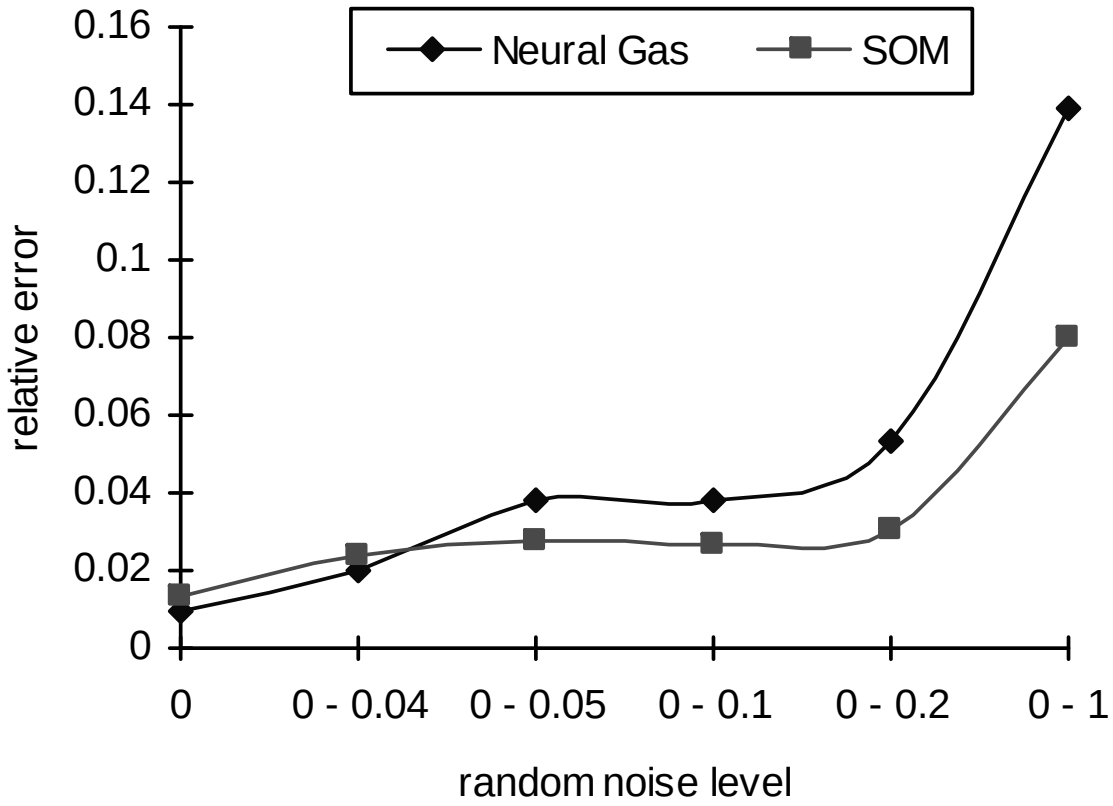
- computational time = construction time + generation time+rendering

SOM and Neural Gas

- computational time = construction time + rendering

MLFF, SOM, and *Natural Gas* Modelling Performance Comparison: Compactness





For low levels of noise the Neural Gas network performs better than **SOM**. For higher level of noise, **SOM** tends to smooth the effect of noise, while the Neural Gas network, which has high sensitivity, follows the noisy patterns.

MLFF, SOM, and *Natural Gas* Modelling of 3D Objects
- conclusions -

* The use of neural network modeling is advantageous from the point of view simplicity and compactness.

* *MLFNN* – provide continuous models, information on the entire object space, convenient for many applications, however they are time consuming.

* *SOM and Neural Gas* – provide compressed models while maintaining the properties of the objects, have very good accuracy, and they are less time consuming

* The use of any specific techniques depends on the application requirements => ***All things considered, the quality of the Neural Gas models is better than that of the SOM models***

Measuring the Elastic Properties

Recovery of the elastic material properties requires touching each point of interest on the explored object surface and then conducting a strain-stress relation measurement on each point.

Tactile probing is a time consuming Sequential operation

Find fast sampling procedures able to minimize the number of the sampling points by selecting only those points that are relevant to the elastic characteristics.

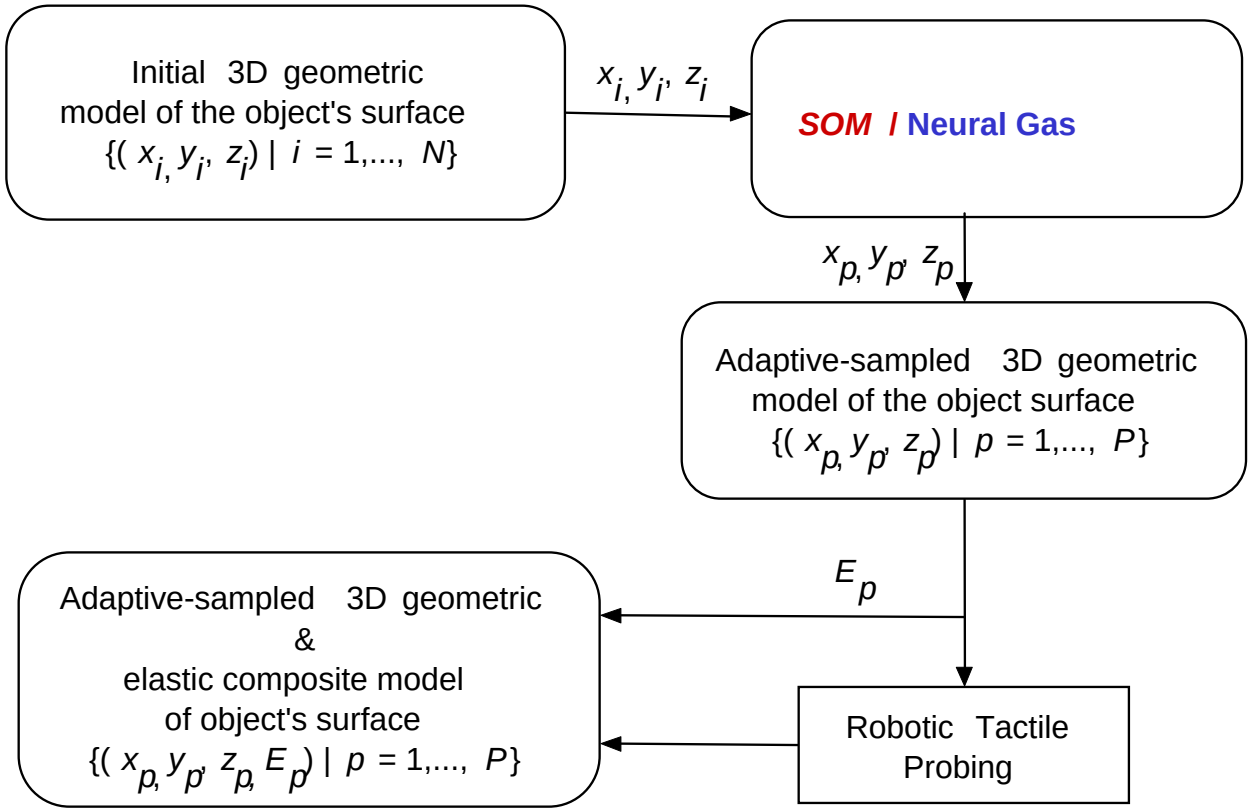
non-uniform adaptive sampling algorithm of the object's surface, which exploits the SOM (*self-organizing map) ability to find optimal finite quantization of the input space.*

The elastic behaviour at any given point (x_p, y_p, z_p) on the object surface is described by the Hooke's law:

$$\begin{cases} \sigma_p = E_p \cdot \varepsilon_p & if \ 0 \le \varepsilon_p \le \varepsilon_{p\,\max} \\ \sigma_p = \sigma_{p\,\max} & if \ \varepsilon_{p\,\max} < \varepsilon_p \end{cases}$$

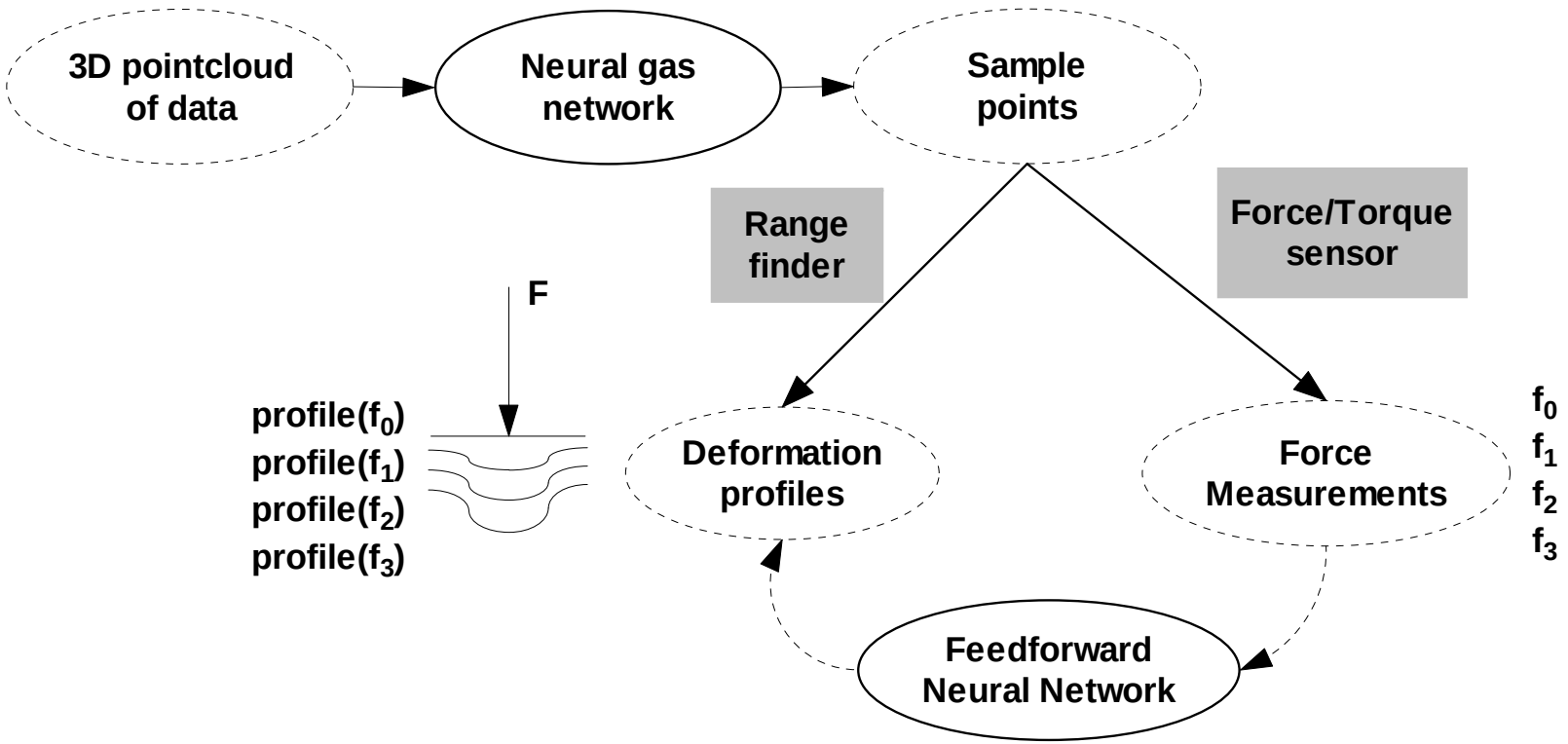
where E_p is the modulus of elasticity , s_p is the stress, and e_p is the strain on the normal direction.

Adaptive Sampling Control of of Elastic Properties of 3D Object Surfaces



A.-M. Cretu, E.M. Petriu, “Neural-Network –Based Adaptive Sampling of Three-Dimensional-Object Surface Elastic Properties,” *IEEE Trans. Instrum. Meas.,"* Vol. 55, No. 2, pp. 483-492, 2006.

Neural Network Mapping an Clustering of Elastic Behavior from Tactile and Range Imaging



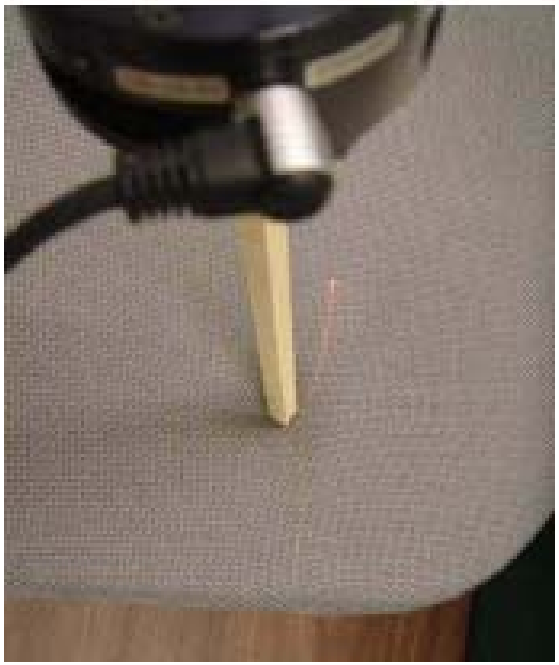
Starting from a 3D point-cloud, a **neural gas NN** *yields a reduced set of points on the 3D object's surface which are relevant for the tactile probing*. The density of these tactile probing points is higher in the regions with more pronounced variations in the geometric shape. A **feedforward NN** *is then employed to model the force/displacement behavior of selected sampled points* that are probed simultaneously by a force/torque sensor and the active range finder.

The elastic behaviour at any given point (x_p, y_p, z_p) on the object surface is described by the Hooke's law:

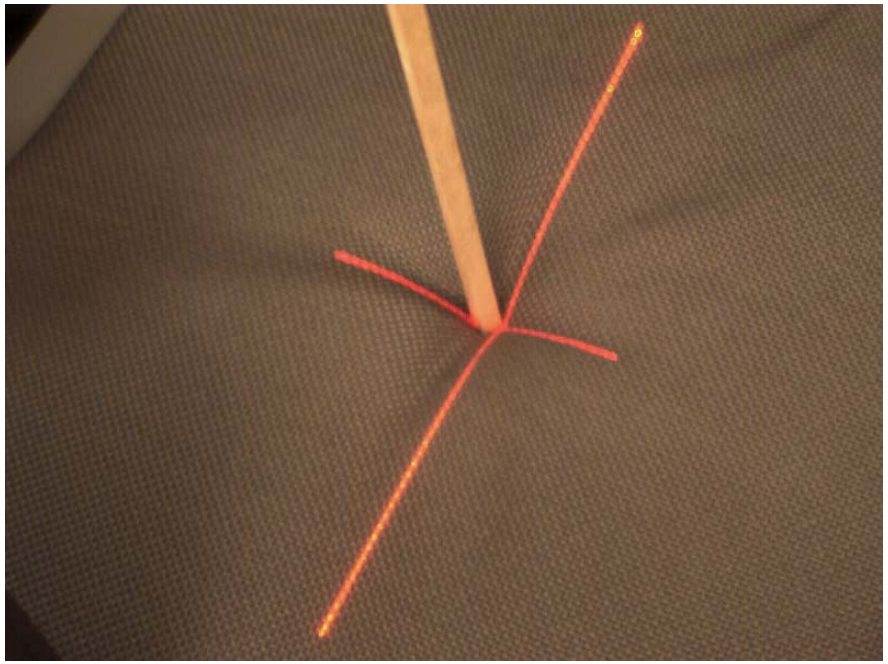
$$\begin{cases} \sigma_p = E_p \cdot \varepsilon_p \quad if \ 0 \le \varepsilon_p \le \varepsilon_{p\,\max} \\ \sigma_p = \sigma_{p\max} \quad if \ \varepsilon_{p\max} < \varepsilon_p \end{cases}$$

where E_p is the modulus of elasticity, s_p is the stress, and e_p is the strain on the normal direction.

Recovery of the elastic material properties requires touching each point of interest on the explored object surface and then conducting a strain-stress relation measurement on each point.

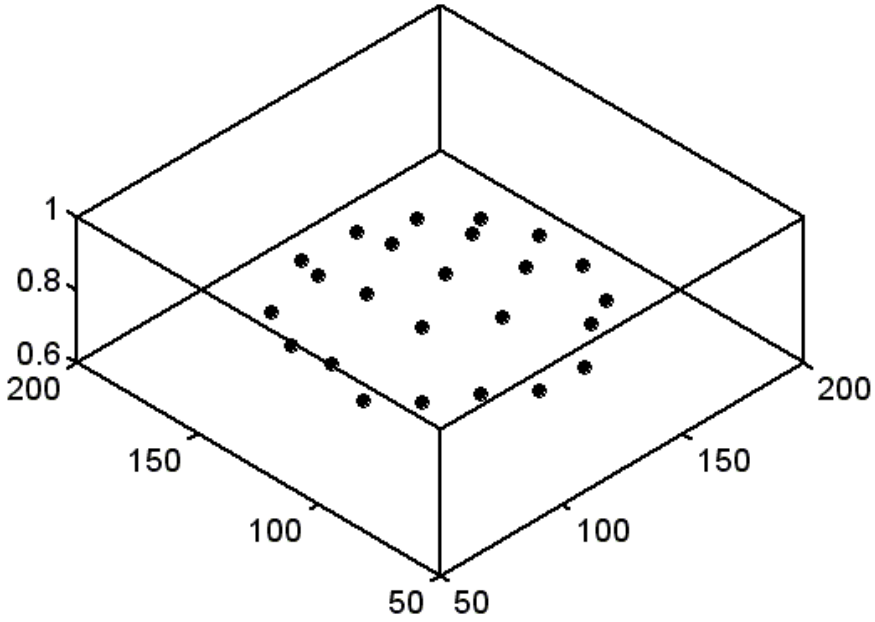


Force-torque sensor measuring the interaction force and torque at the point of contact between the robot manipulated probe and the object.



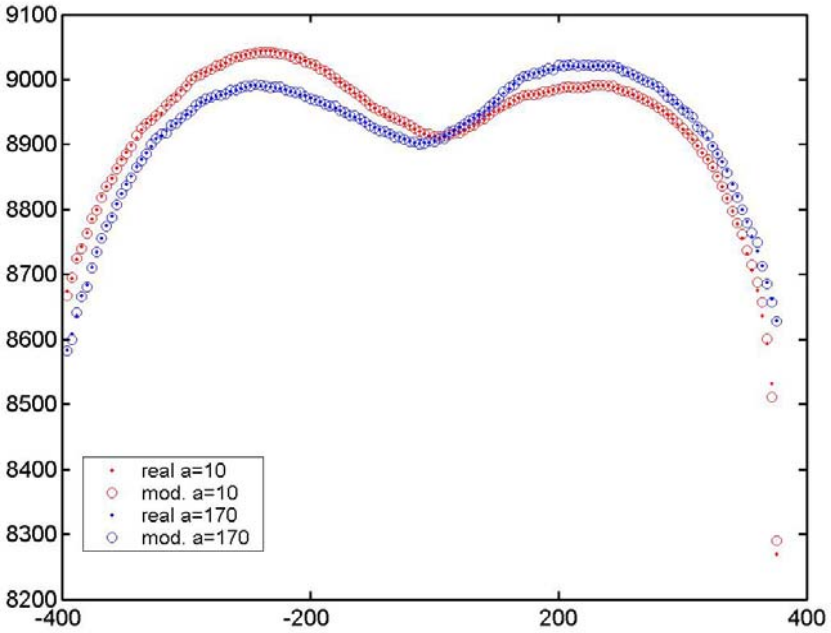
Laser range-finder based recovery of the geometric profiles in an area around the contact point between theprobe and the object.

Elastic ball used
for experimentation.

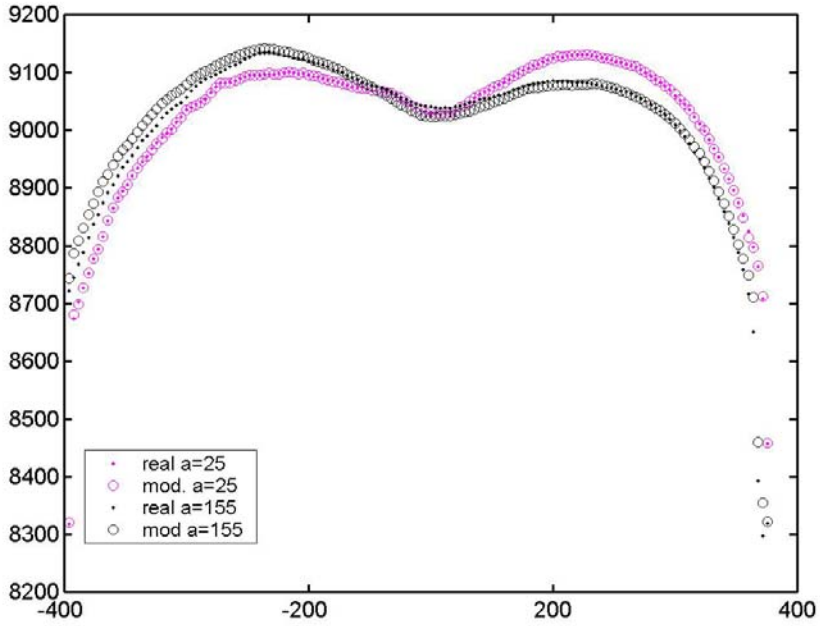


Sampling points selected with
the neural gas network for the ball.

(from A.M. Cretu, E.M. Petriu, P.Payeur “Neural Network Mapping and Clustering of Elastic Behavior from Tactile and Range Imaging for Virtualized Reality Applications,” submitted to *IEEE Tr. Instr. Meas.,* Nov. 2006).



(a)



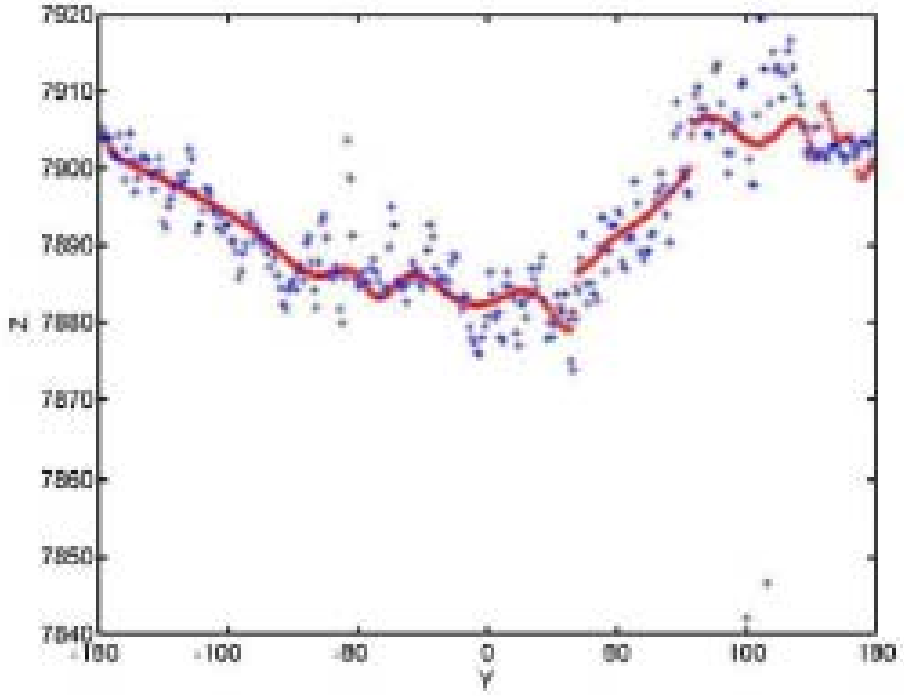
(b)

Real and modeled deformation curves using neural network for rubber under forces applied at different angles:

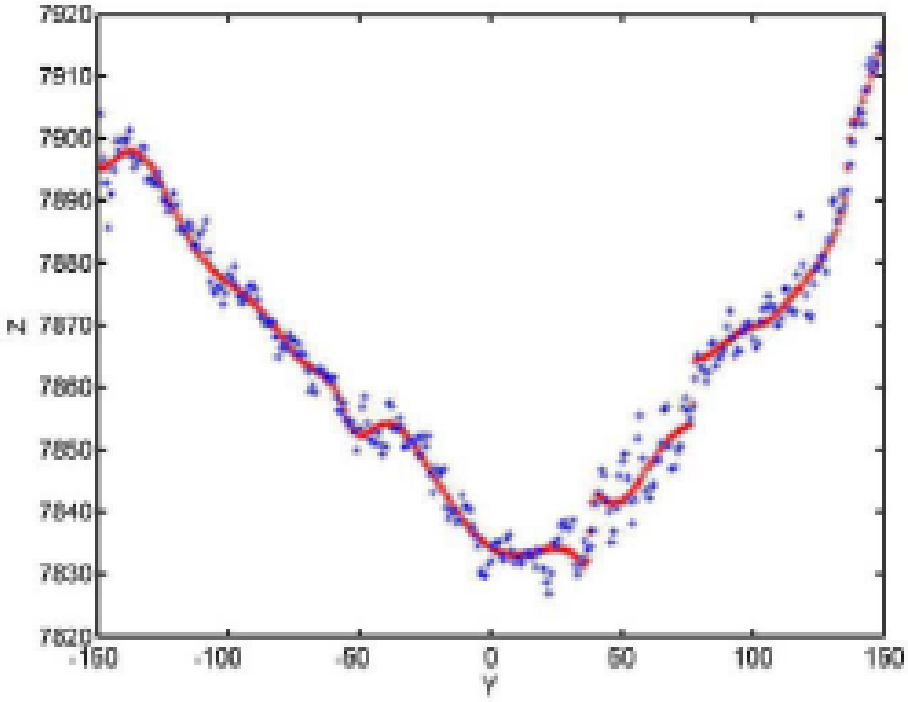
a) F=65N, α_1=10° and F=65N, α_2=170°,

b) F=36N, α_1=25°, and F=36N, α_2=155

(from .A.M. Cretu, E.M. Petriu, P.Payeur “Neural Network Mapping and Clustering of Elastic Behavior from Tactile and Range Imaging for Virtualized Reality Applications,” submitted to *IEEE Tr. Instr. Meas.,* Nov. 2006).



Real and NN modeled (Y, Z) geometric profile of a semi-stiff material (cardboard) pressed with a normal force F=0.37 N. The unit of measurement for both Y and Z axes is 0.1 mm.



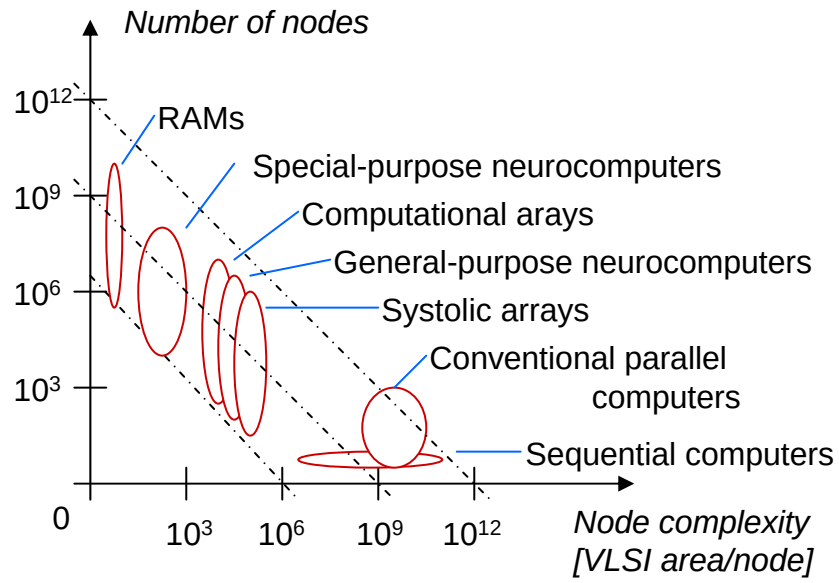
Real and NN modeled (Y, Z) geometric profile of a semi-stiff material (cardboard) pressed with a normal force F= 2.65 N. The unit of measurement for both Y and Z axes is 0.1 mm.

Hardware Neural Network Architectures

ANNs / **Neurocomputers** =>architectures optimized for neuron model implementation

- *general-purpose*, able to emulate a wide range of NN models;
- *special-purpose*, dedicated to a specific NN model.

Hardware NNs consisting of a collection of simple neuron circuits provide the massive computational parallelism allowing for a higher Model rendering speed.



[from P. Treleaven, M. Pacheco, M. Vellasco, "VLSI Architectures for Neural Networks," IEEE Micro, Dec. 1989, pp. 8-27]

ANN VLSI Architectures:

- *analog* ==> compact,high speed, asynchronous, no quantization errors, convenient weight "+"and "X";
- digital ==> more efifcient VLSI technology, robust, convenient weight storage;

Pulse Data Representation:

- *Pulse Amplitude Modulation* (PAM) - not satisfactory for NN processing;
- Pulse Width Modulation (PWM);
- Pulse Frequency Modulation (PFM).

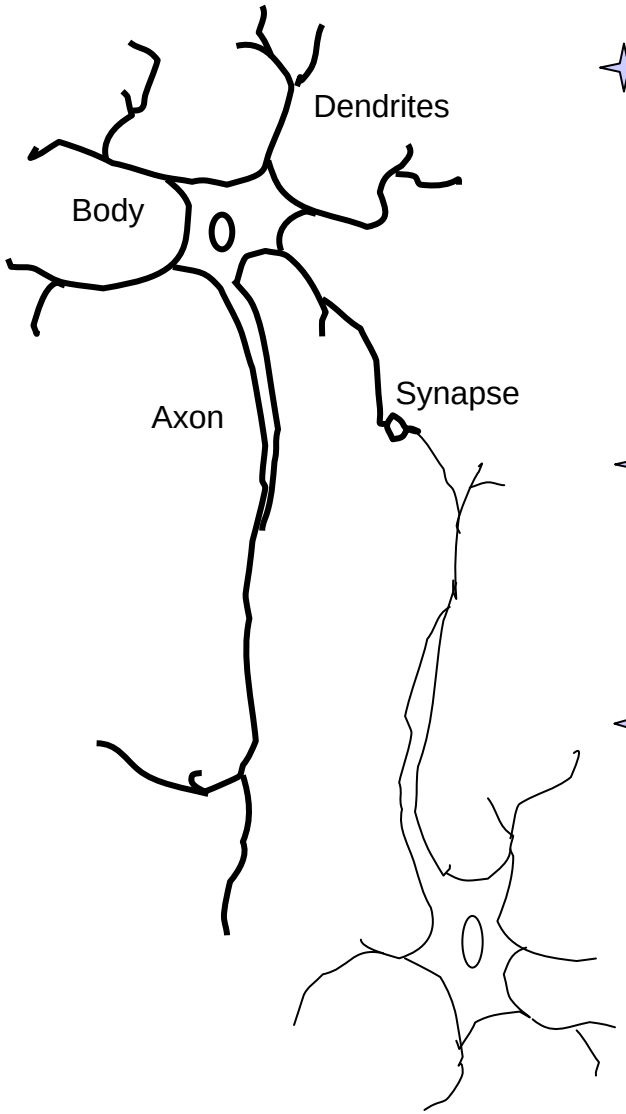
⇩

Pulse Stream ANNs: combination of different pulse data representation methods and opportunistic use of both analog and digital implementation techniques.

Stochastic Data Representation

Looking for a model to prove that algebraic operations with analog variables can be performed by logic gates, Professor **J. von Neuman** advanced in 1956 the *idea of representing analog variables by the mean rate of random-pulse streams [J. von Neuman, "**Probabilistic logics and the synthesis of reliable organisms from unreliable components**," in Automata Studies, (C.E. Shannon, Ed.), Princeton, NJ, Princeton University Press, 1956].*

Biological Neurons

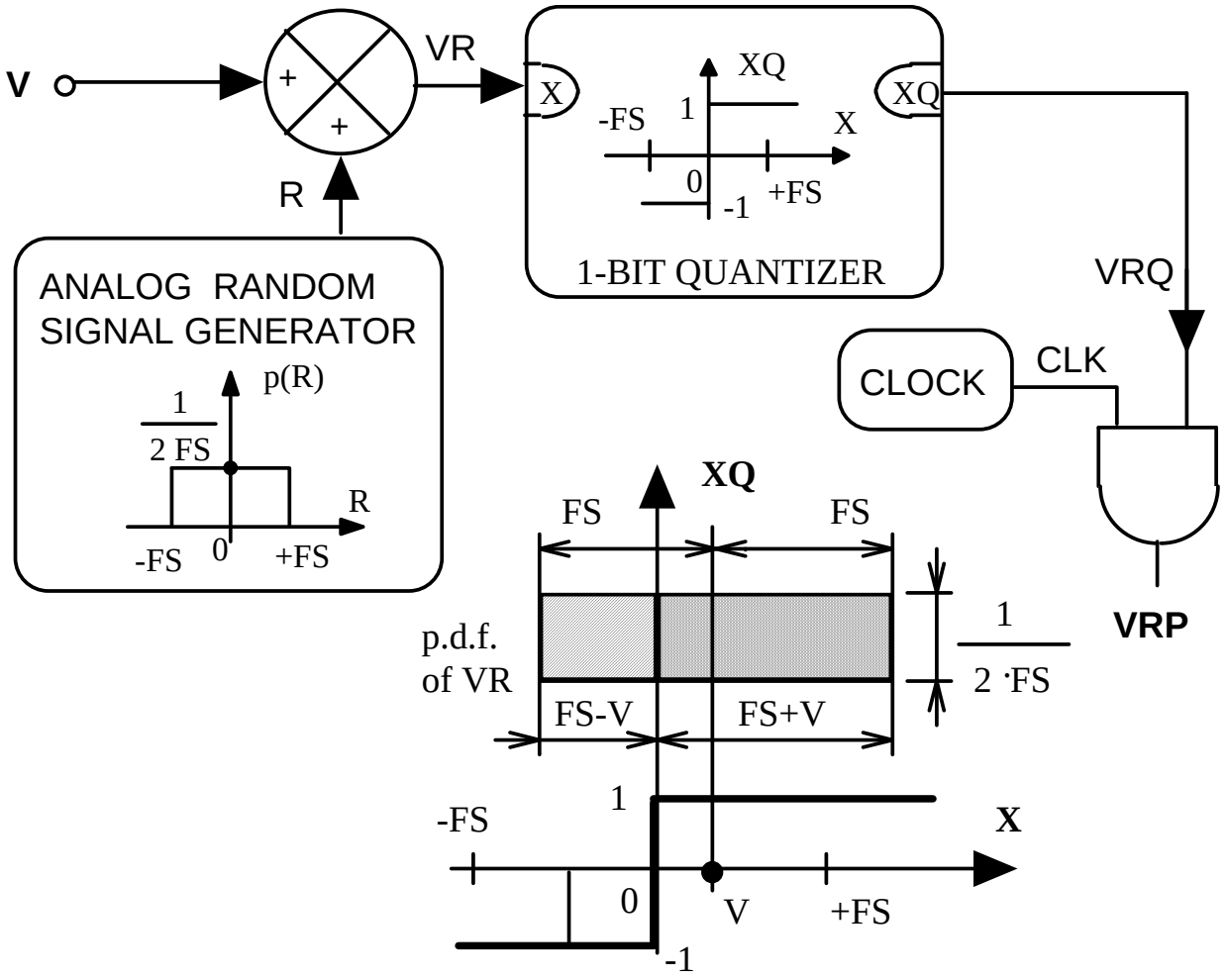


- **Dendrites** carry electrical signals in into the neuron body. The neuron **body** integrates and thresholds the incoming signals.The **axon** is a single long nerve fiber that carries the signal from the neuron body to other neurons. A **synapse** is the connection between dendrites of two neurons.

- *Memories* are formed by the modification of the **synaptic strengths** which can change during the entire life of the neural systems.

- **Neurons are rather slow (10^{-3} s)** when compared with the modern electronic circuits. ==> The brain is faster than an electronic computer because of its massively parallel structure. The **brain has approximately 10^{11} highly connected neurons** (approx. 10^4 connections per neuron).

Analog/Random-Pulse Conversion



Random-Pulse/Digital Conversion

The *deterministic component of the random-pulse sequence*, conveniently unbiased and rescaled for this purpose to take values +1 and -1 (instead of 1 and respectively 0) , can be calculated as a **statistical estimation** from the quantization diagram:

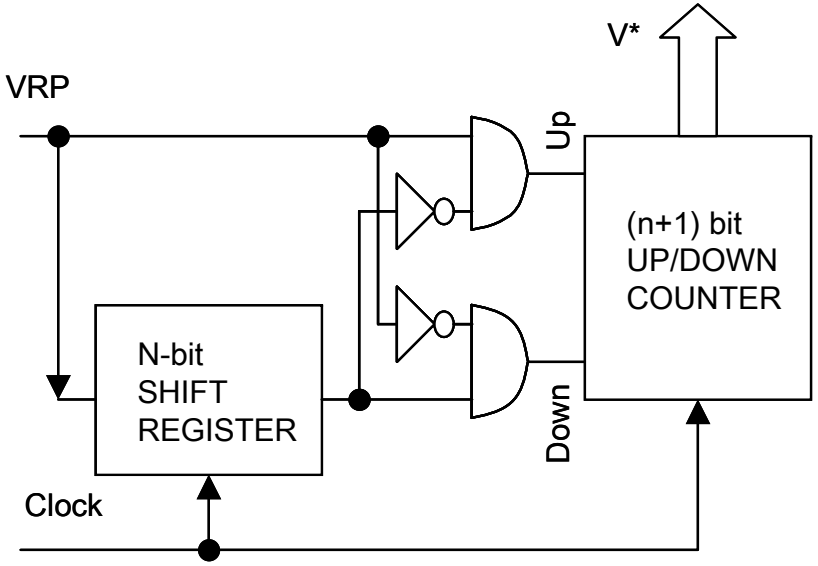
$$E[VRP] = (+1) .p[VR>=0] + (-1) .p[VR<0] = p(VRP) - p(VRP')$$
$$= (FS+V)/(2.FS) - (FS-V)/(2.FS) = V/FS;$$

This finally gives the deterministic analog value V associated with the binary VRP sequence:

$$\mathbf{V = [p(VRP) - p(VRP')] . FS ;}$$

where the *apostrophe* (') denotes a logical inversion.

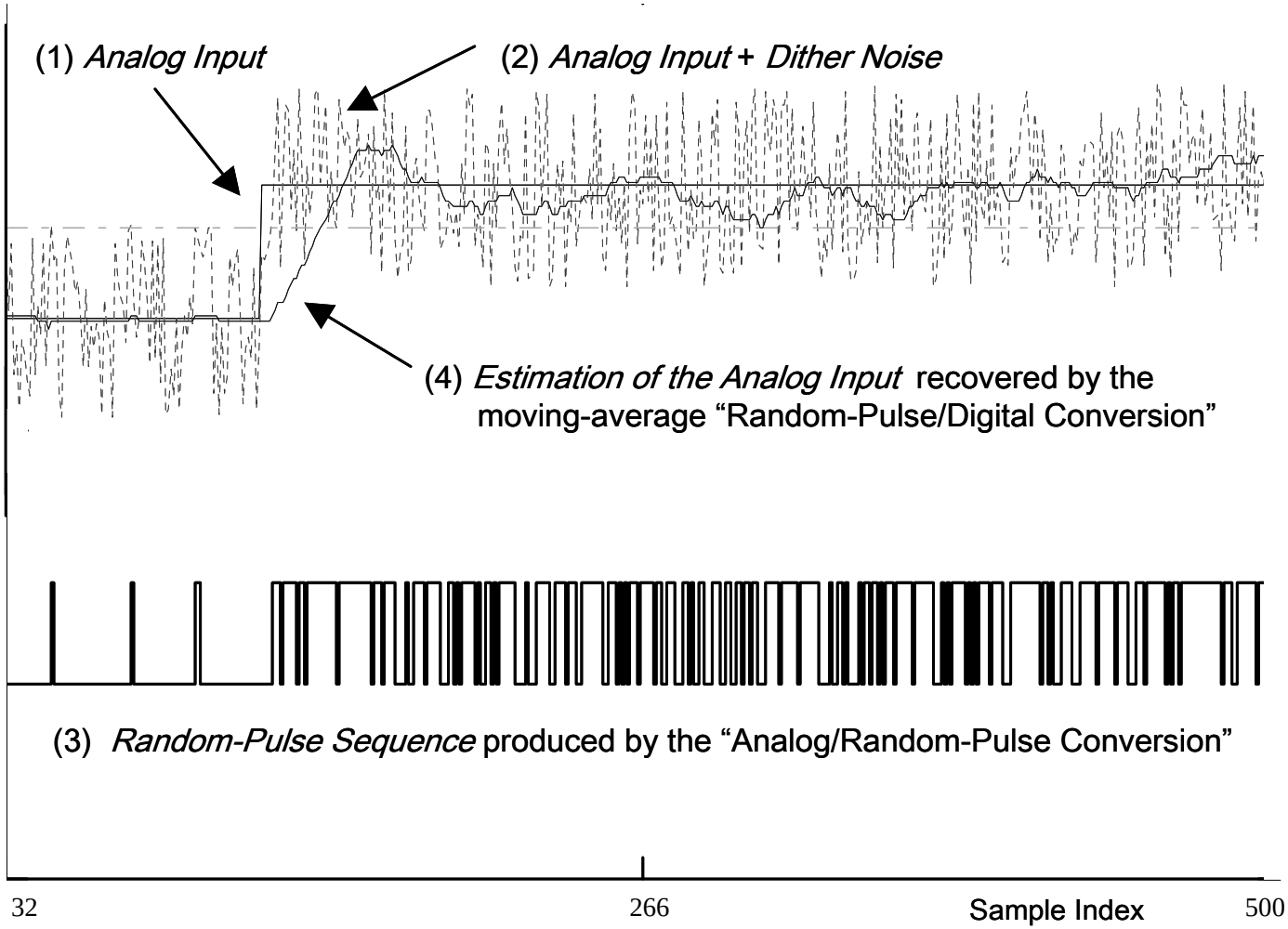
Random-Pulse/Digital Conversion using the Moving Average Algorithm .



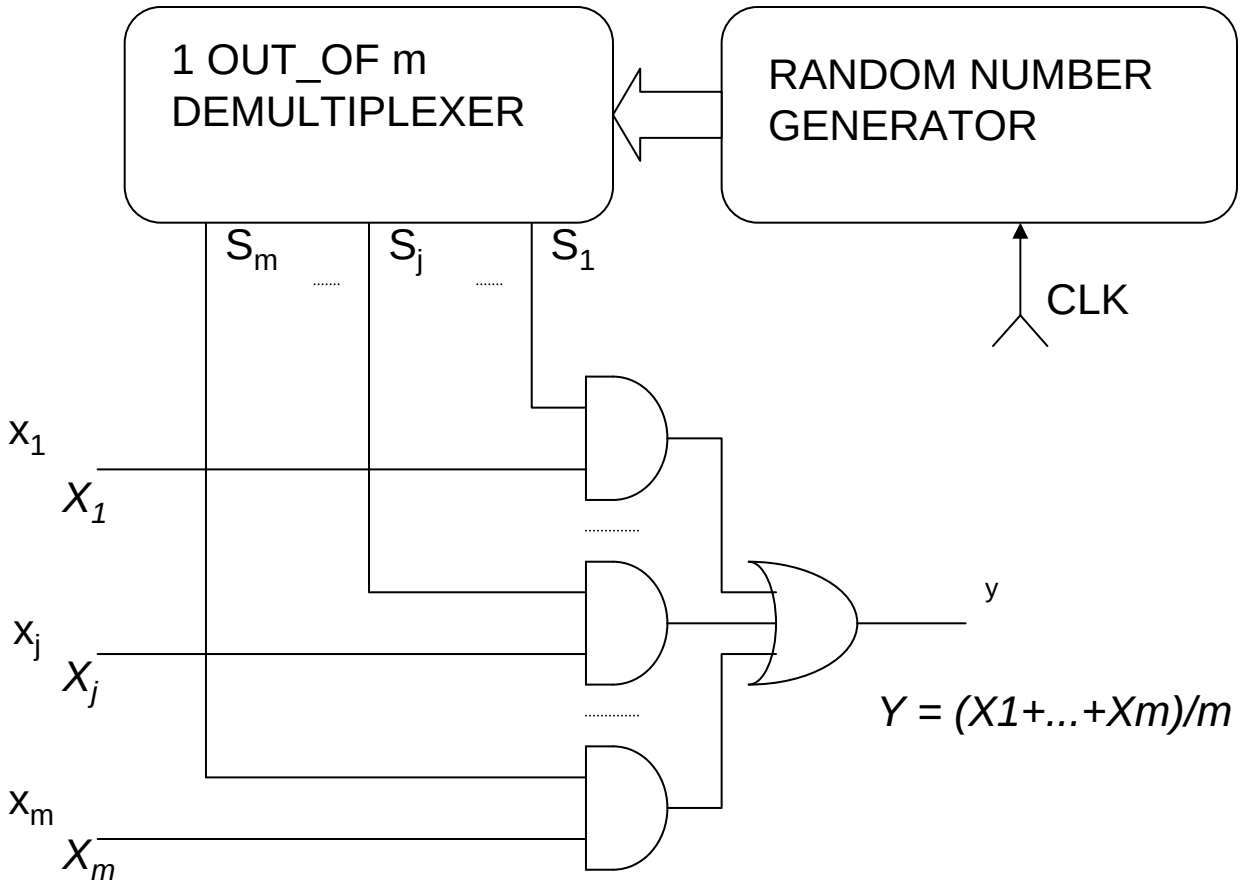
$$V^*_N = \frac{1}{N}\sum_{i=1}^{N} VRP_i = \frac{1}{N}\left(\sum_{i=1}^{N-1} VRP_i + VRP_N\right);$$

$$V^*_N = V^*_{N-1} + \frac{VRP_N - VRP_0}{N};$$

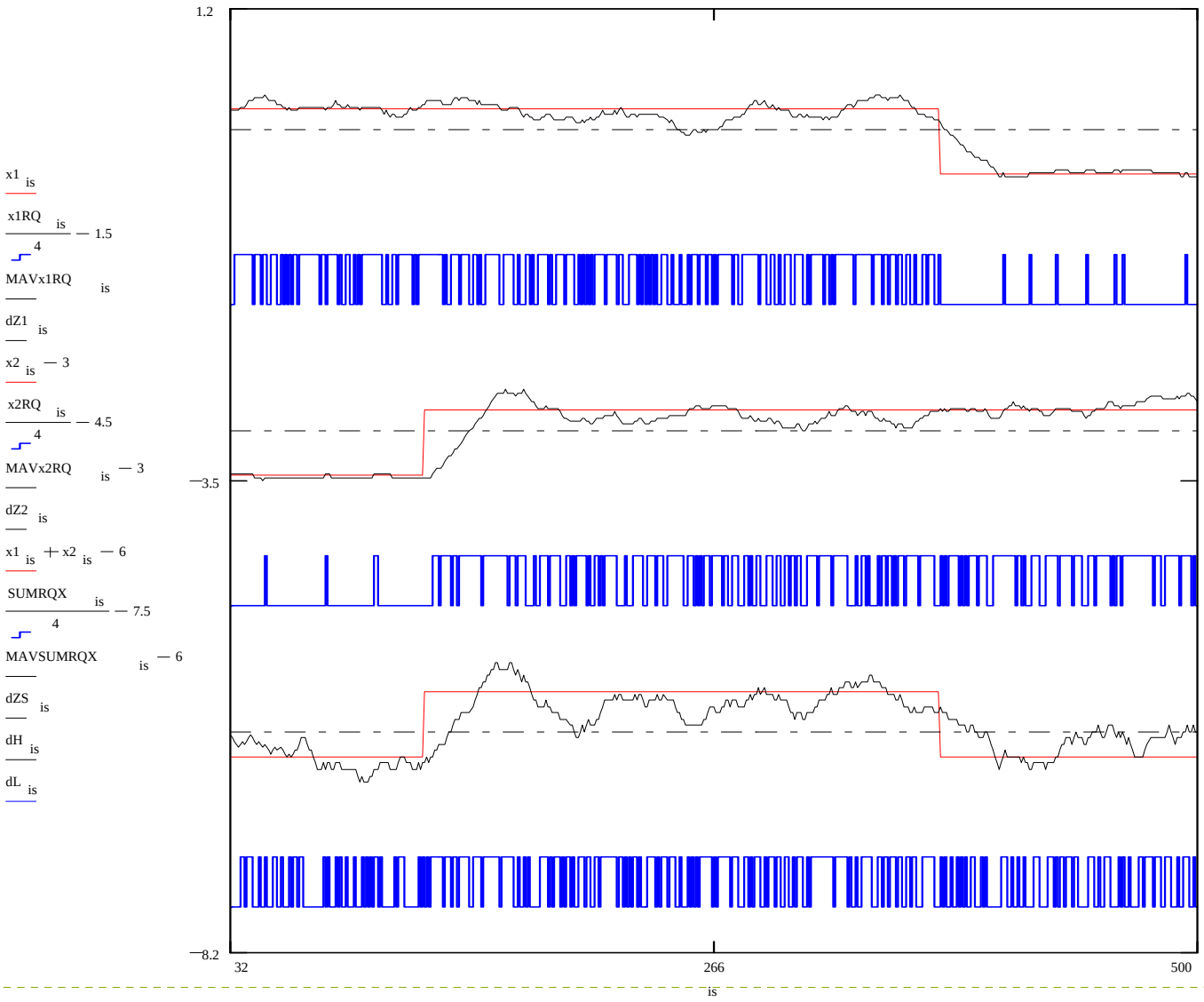
Analog/Random-Pulse and Random-Pulse/Digital Conversion



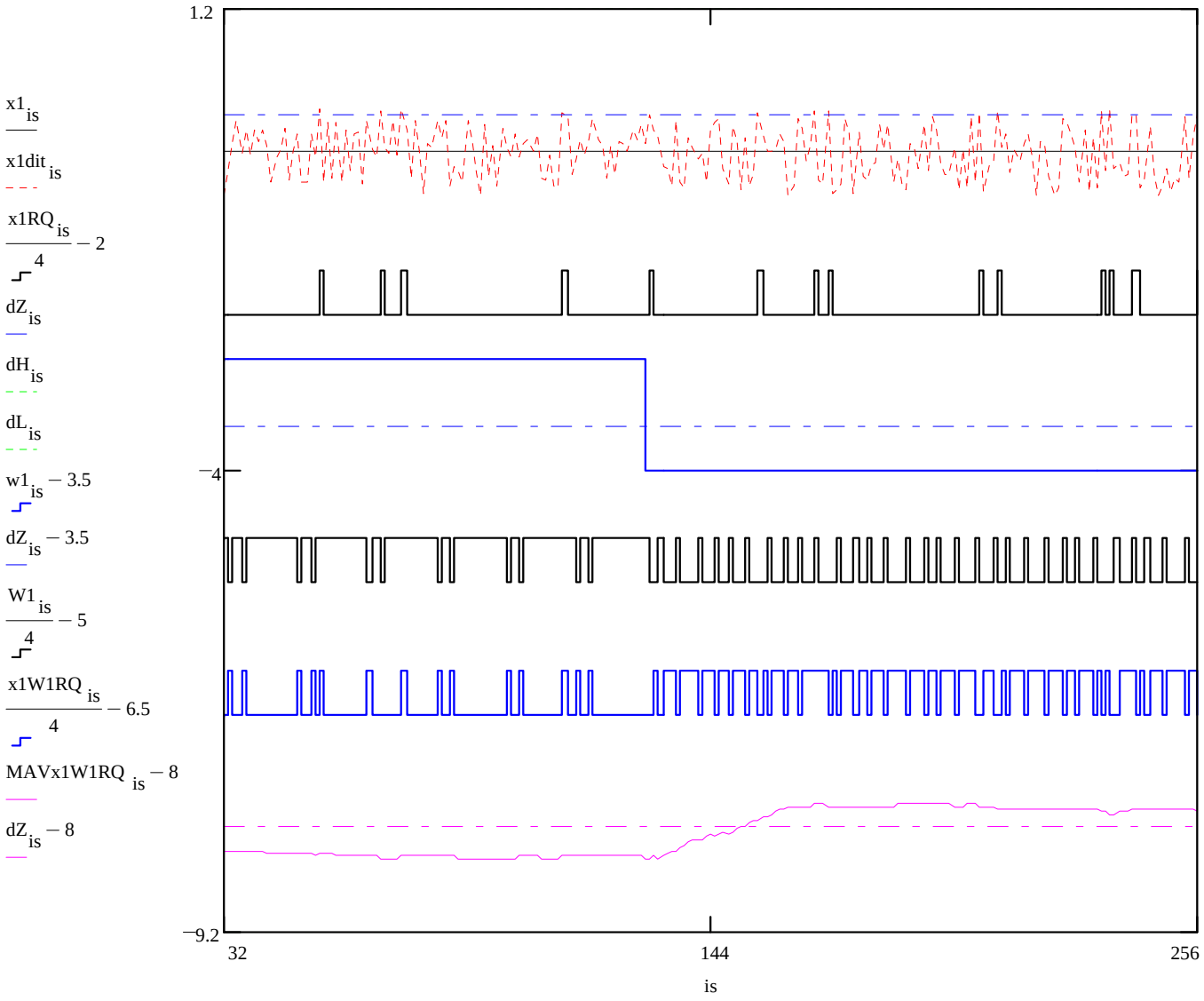
Random-Pulse Addition



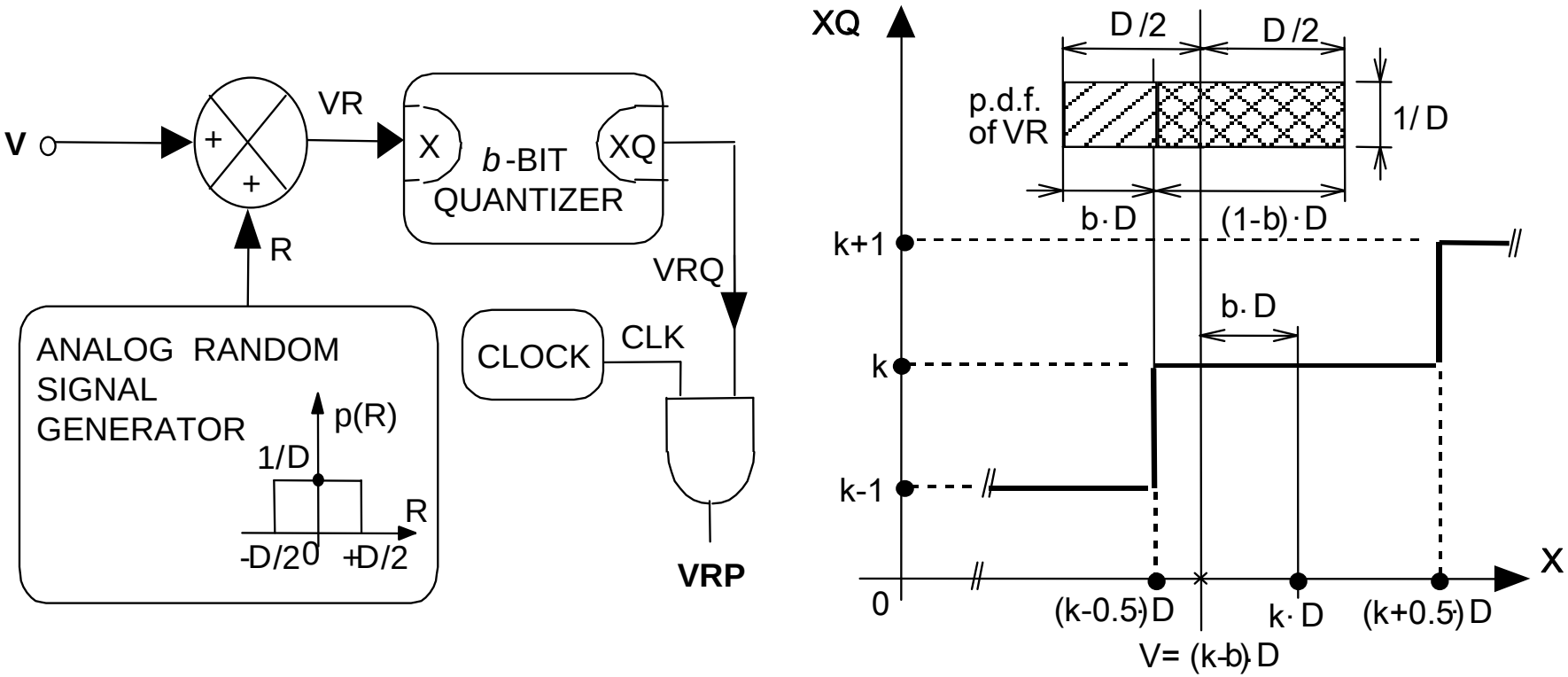
Random-Pulse Addition



Random-Pulse Multiplication



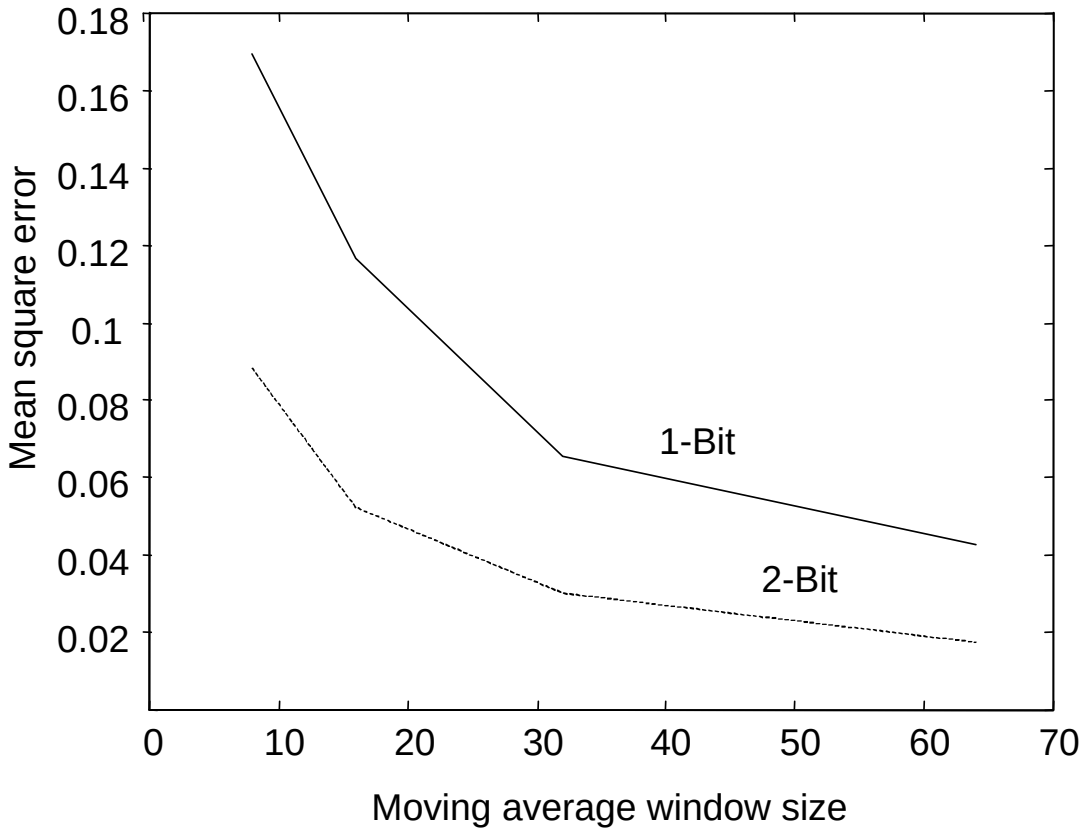
Stochastic Data Representation



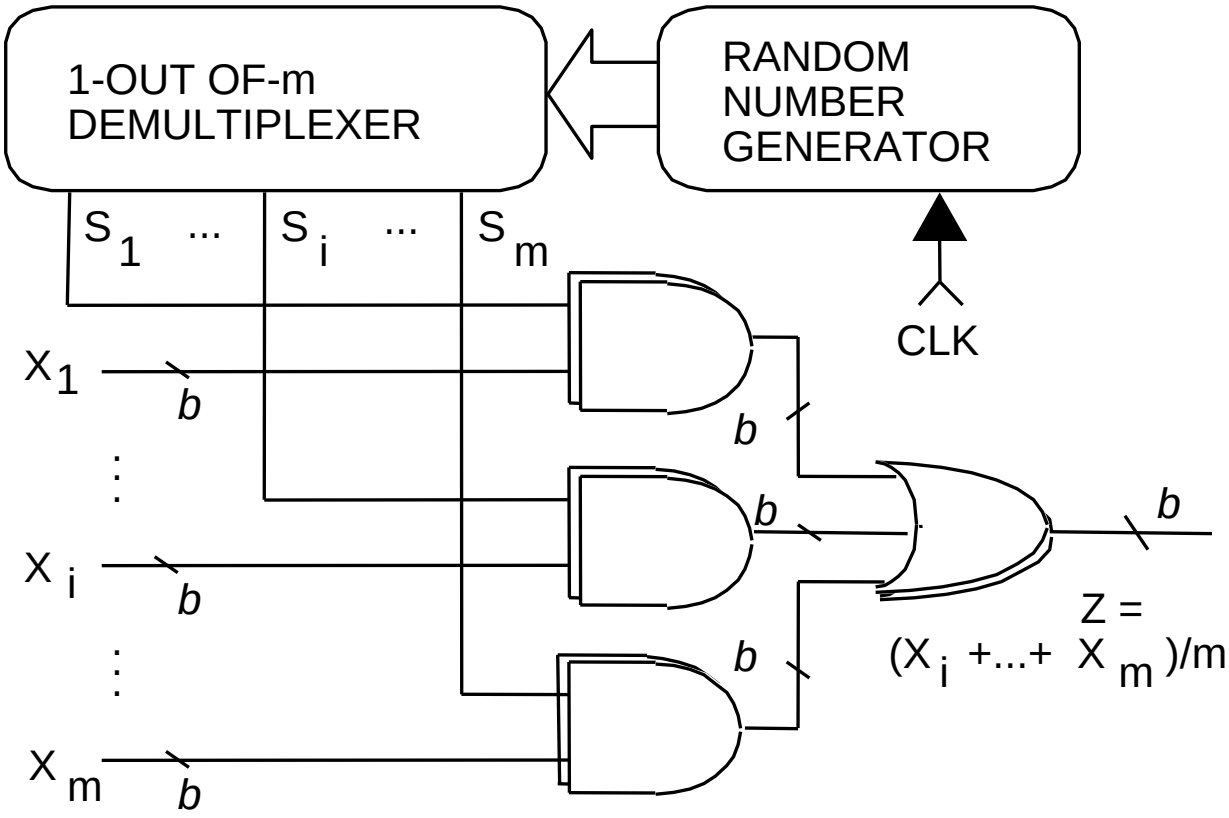
Generalized b-bit analog/random-data conversion

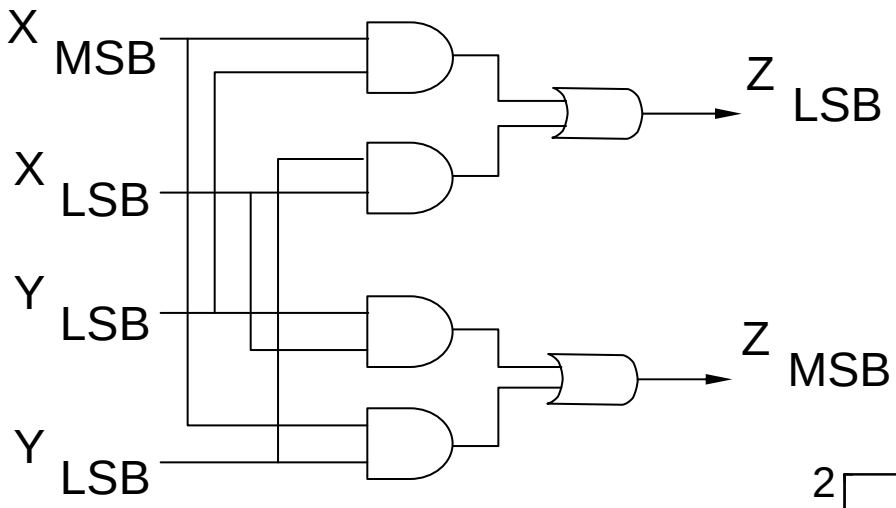
E.M. Petriu, L. Zhao, S.R. Das, V.Z. Groza, A. Cornell, "Instrumentation Applications of Multibit Random-Data Representation," *IEEE Trans. Instrum. Meas.,* Vol. 52, No. 1, pp. 175- 181, 2003.

Quantization levels	Relative mean square error
2	72.23
3	5.75
4	2.75
...	...
8	1.23
...	...
analog	1



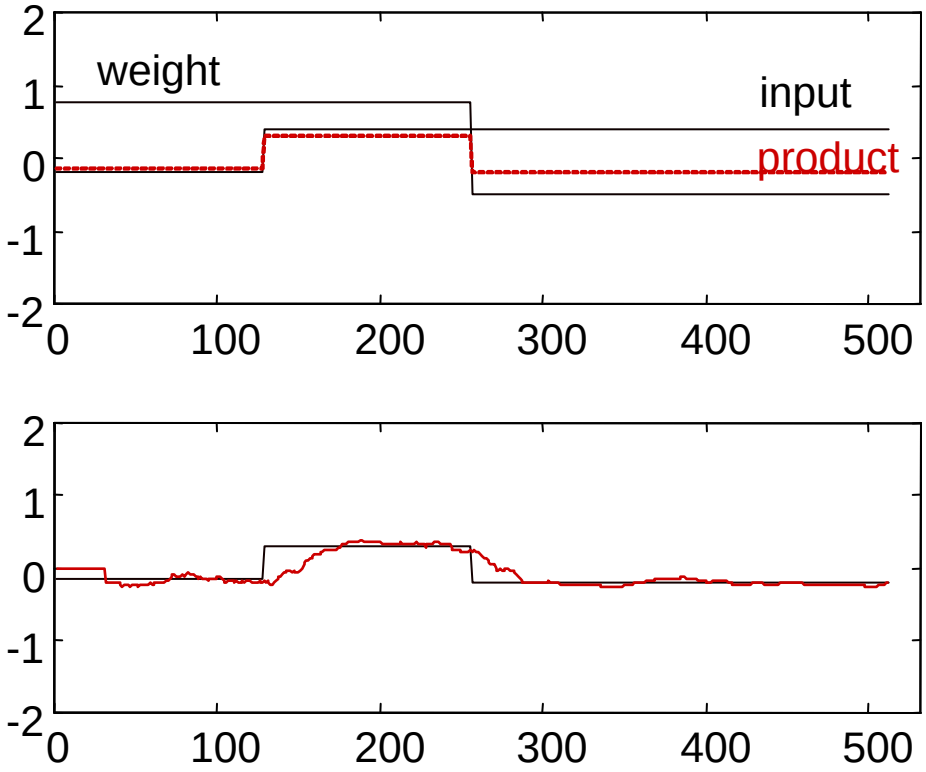
Stochastic-Data Addition.



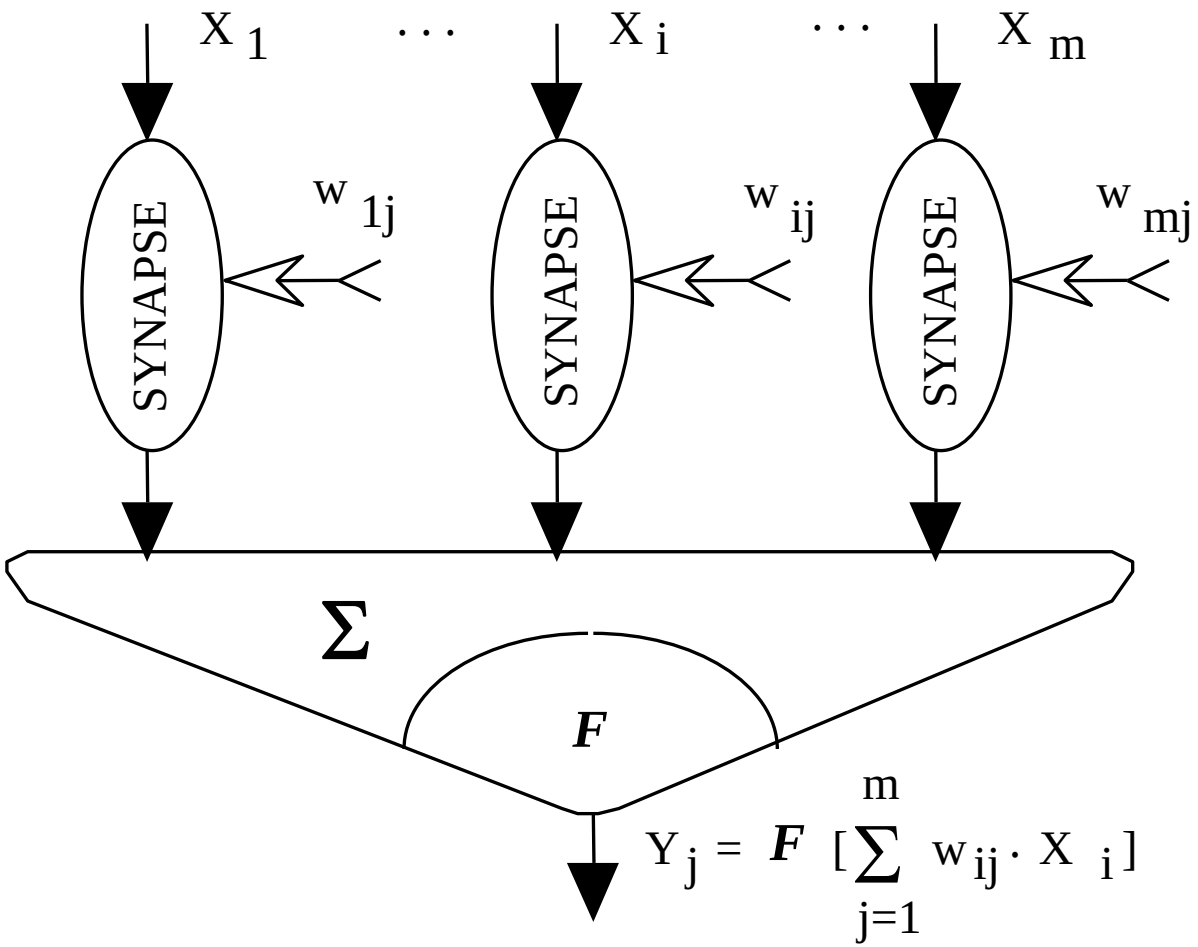


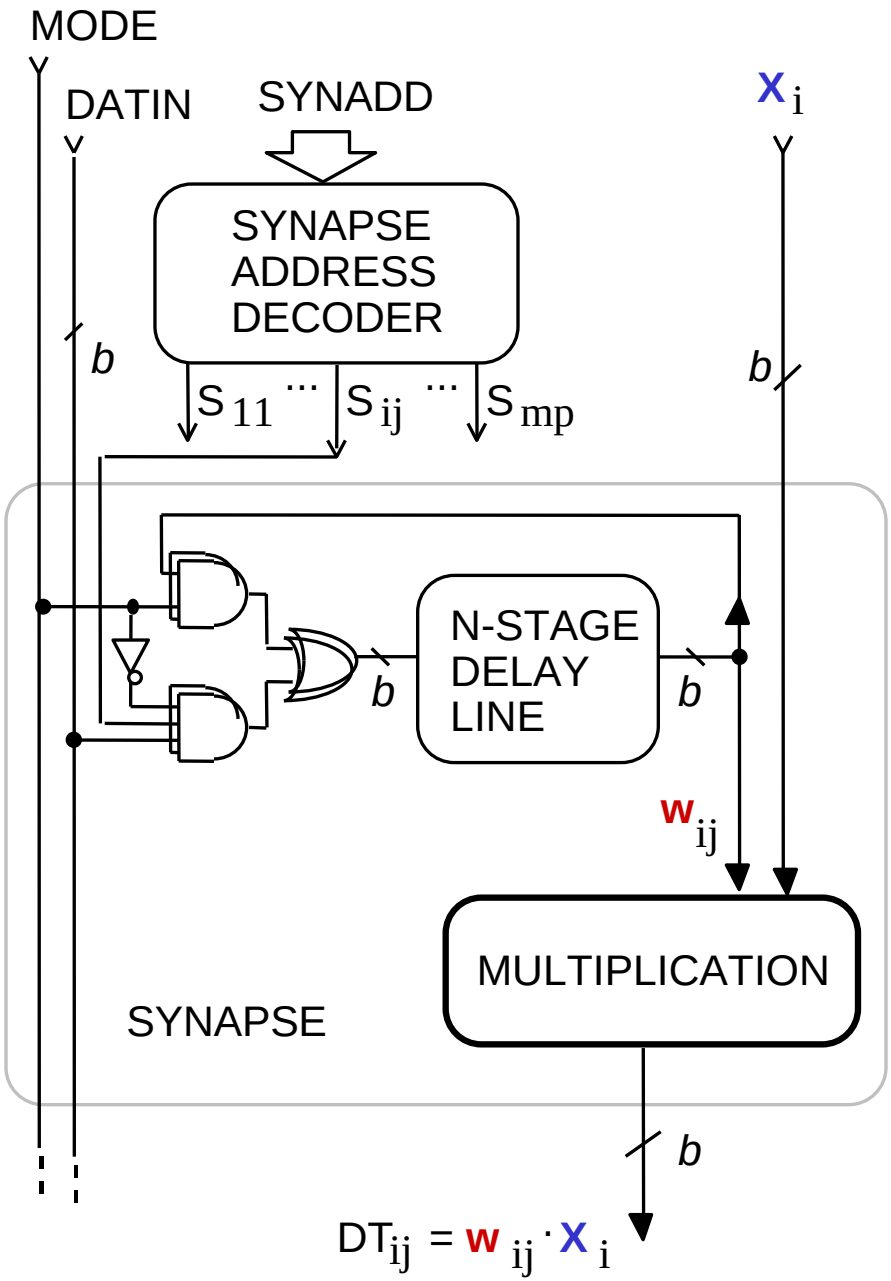
2-bit Stochastic-Data Multiplication

X \ Y		**0**	**1**	**-1**
		00	*01*	*10*
0	*00*	**0** *00*	**0** *00*	**0** *00*
1	*01*	**0** *00*	**1** *01*	**-1** *10*
-1	*10*	**0** *00*	**-1** *10*	**1** *01*



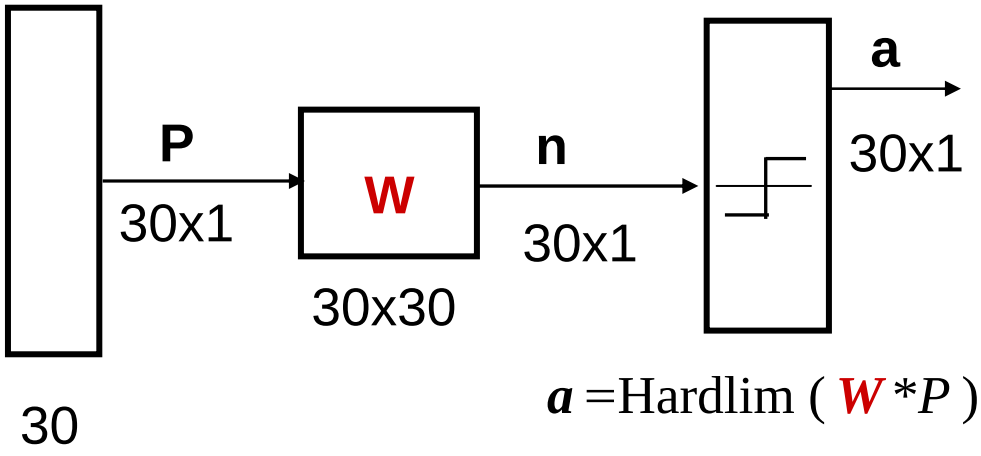
Neural Network Architectures Using Stochastic Data Representation



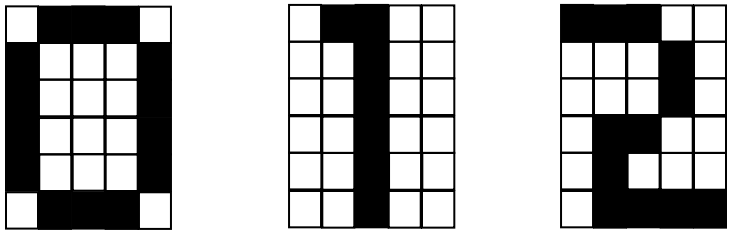


Stochastic-data implementation of a synapse

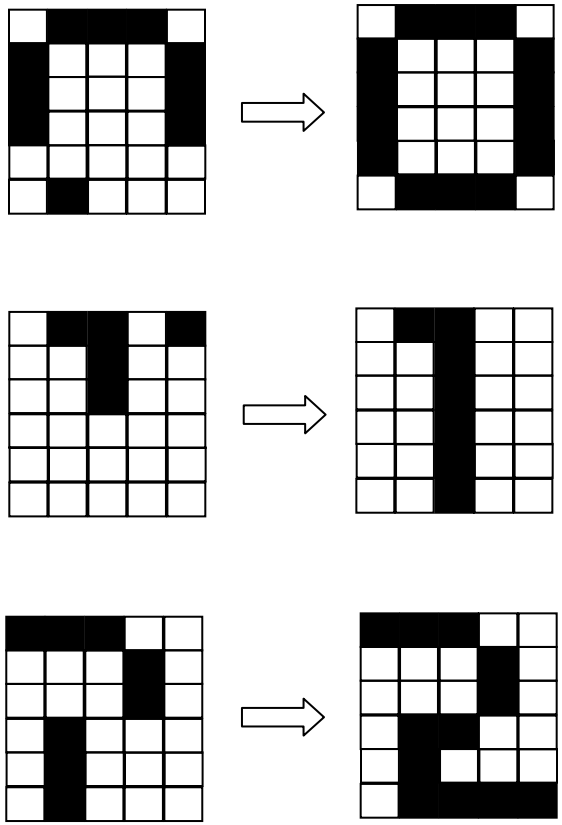
Auto-associative memory NN architecture



$$\boldsymbol{a} = \text{Hardlim}\,(\boldsymbol{W} * P)$$



Training set



Recovery of 30% occluded patterns

Conclusions

- "**Improved accuracy and richness in object modeling and haptic rendering** will require advances in our understanding of how to represent and render psychophysically and cognitively germane attributes of objects, as well as algorithms and perhaps specialty hardware (such as ***haptic or physics engines***) to perform real-time computations" [K. Salisbury, F. Conti, F. Barbagli, "Haptic Rendering: Introductory Concepts," *IEEE Computer Graphics and Applications*, Vol. 24, No. 2, pp. 24 – 32, 2004].

- **Neural Networks** which are able to learn nonlinear behaviors from a limited set of measurement data can **provide efficient and compact multi-media object modeling solutions**. Due to their continuous, analog-like, memory behavior, NNs are able to provide instantaneously an estimation of the output value for input values that were not part of the initial training set.

- **NNs** consisting of a collection of simple neuron circuits provide **the massive computational parallelism offering efficient storage, model transformation, and real-time rendering capabilities for large numbers of composite geometric & haptic object models** involved in the model-based interactive telemanipulation.

Thank you!