Complex-Valued Neural Networks II. Multi-Valued Neurons: Theory and Applications

Article · January 2011

DOI: 10.1007/978-3-642-20353-4_2

1 author:

Igor Aizenberg
Manhattan College

Complex-Valued Neural Networks
II. Multi-Valued Neurons: Theory and Applications

Igor Aizenberg

Texas A&M University-Texarkana,
Department of Computer Science

igor.aizenberg@tamut.edu

http://www.eagle.tamut.edu/faculty/igor
http://www.freewebs.com/igora

Content

- Multiple-Valued Logic (*k*-valued Logic) over the Field of the Complex Numbers
- Multi-Valued Neuron (MVN)
- Derivative-free Learning Agorithm for MVN
- **MVN-based multilayer feedforward neural network** and its derivative free backpropagation learning algorithm
- Applications
- Further development

Brief Introduction

- What is a multi-valued neuron?

Motivation

- The functionality of real-valued neurons is quite limited

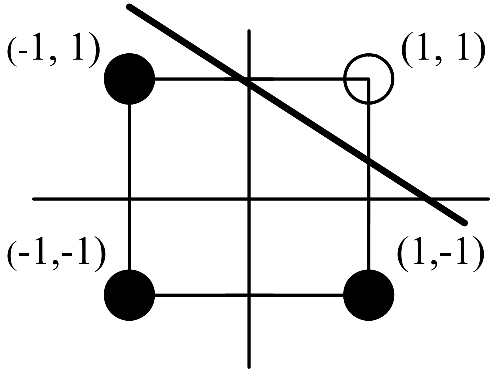
- One of the main ideas behind any complex-valued neuron is the aspiration for significant increasing of the neuron’s functionality and simplification of its learning

Limited Functionality of Classical Neurons

- A classical neuron can learn only linearly-separable input/output mappings and cannot learn nonlinearly-separable input/output mappings (Minsky-Papert, 1969 – a classical limitation)
- What does it mean?

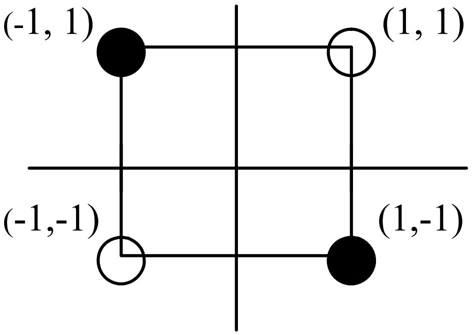
Linear Separability/Non-separability

"OR" is an example of the **threshold** (**linearly separable**) Boolean function:
"-1s" are separated from "1" by a line



- 1 1 → 1
- 1 -1 → -1
- -1 1 → -1
- -1 -1 → -1

XOR is an example of the **non-threshold** (**non-linearly separable**) Boolean function: there is no way to separate "1s" from "-1s" by any single line

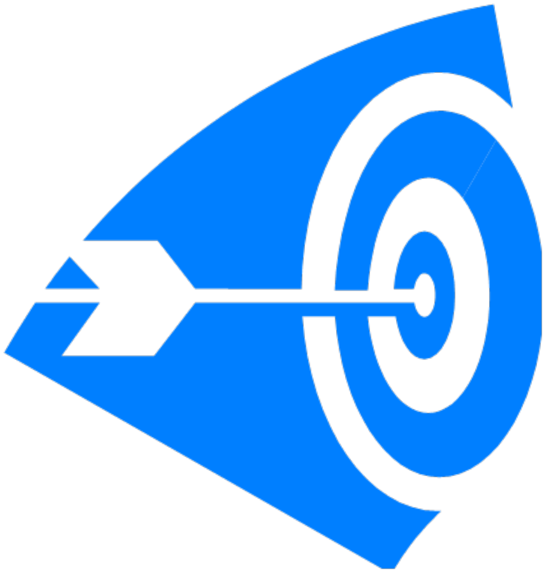


- 1 1 → 1
- 1 -1 → -1
- -1 1 → -1
- -1 -1 → 1

Is it possible to overcome the Minsky's-Papert's limitation for the classical perceptron?

Yes !!!

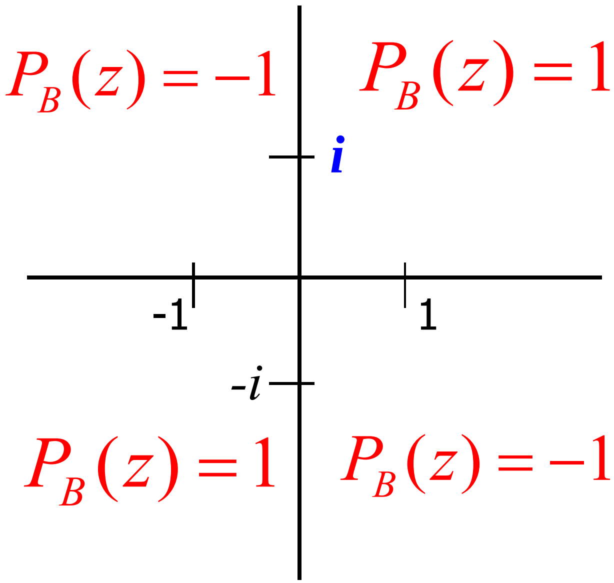
We can overcome the Minsky's-Papert's limitation using the complex-valued weights and the complex activation function



Is it possible to learn XOR and Parity n functions using a single neuron?

- Any classical monograph/text book on neural networks claims that to learn the XOR function a network from at least three neurons is needed.
- This is true for the real-valued neurons and real-valued neural networks.
- However, this is not true for the complex-valued neurons !!!
- A jump to the complex domain is a right way to overcome the Misky-Papert’s limitation and to learn multiple-valued and Boolean nonlinearly separable functions using a single neuron.

Solution of the XOR problem



Let weights be complex and P_B be the activation function

$W=(0, 1, i)$ – the weighting vector

x_1	x_2	$z = w_0 + w_1 x_1 + w_2 x_2$	$P_B(z)$	$x_1 \text{ xor } x_2$
1	1	$1+i$	1	1
1	-1	$1-i$	-1	-1
-1	1	$-1+i$	-1	-1
-1	-1	$-1-i$	1	1

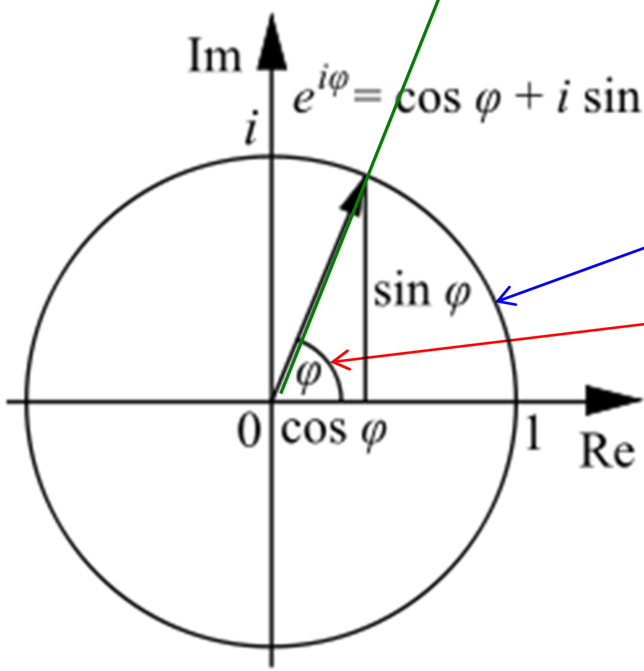
The activation function P_B separates the complex plane into 4 equal sectors

Solution of the XOR problem

- In fact, the activation function P_B is a function of the argument (phase) of the weighted sum: it depends only on the argument of the weighted sum and does not depend on its magnitude.

Phase-Dependent Activation Function

$z = w_0 + w_1 x_1 + \ldots + w_n x_n$ - **weighted sum of inputs**



- **the output $P(z)$ is located on the unit circle**

A unit circle

φ is the argument (phase) of the weighted sum z

Higher Functionality

- Higher functionality of a complex-valued neuron with the phase-dependent activation function means not only the ability to learn non-linearly separable Boolean $\{0,1\}^n \rightarrow \{0,1\}$ or Real-to-Boolean $\mathbb{R}^n \rightarrow \{0,1\}$ input/output mappings, but also multiple-valued input/output mappings up to the continuous ones

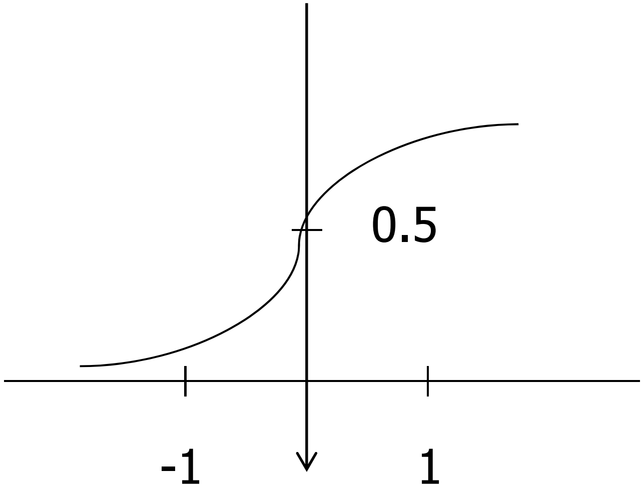
Simplicity of Learning

- Learning of a complex-valued neuron with the phase-dependent activation function and of a feedforward neural network based on this neuron is derivative-free.
- A "local minima" problem does not exist for the corresponding learning process.

Traditional approaches to learn multiple-valued and continuous mappings using a neuron:

- Radial-basis activation function
- Sigmoid activation function (the most popular):

$$F(z) = \frac{1}{1+e^{-z}}$$



Sigmoidal neurons: limitations

- Sigmoid activation function has a limited plasticity and a limited flexibility.
- Thus, to learn those input/output mappings whose behavior is quite different in comparison with the one of the sigmoid function, it is necessary to create a network, because a single sigmoidal neuron is not able to learn such functions.
- Can we take another direction and consider another activation function?

Multiple-Valued Threshold Logic over the field of Complex numbers and a Multi-Valued Neuron

Multi-Valued Mappings and Multiple-Valued Logic

- We traditionally use Boolean functions and Boolean (two-valued) logic, to present two-valued mappings:

$$x_1,\dots,x_n \in K_2 = \{0,1\};\ f(x_1,\dots,x_n): K_2^n \to K_2 = \{0,1\}$$

$$x_1,\dots,x_n \in E_2 = \{1,-1\};\ f(x_1,\dots,x_n): E_2^n \to E_2 = \{1,-1\}$$

- To present multi-valued mappings, we should use multiple-valued logic

Multiple-Valued Logic: classical view

- The values of multiple-valued (*k*-valued) logic are traditionally encoded by the integers $\{0, 1, \ldots, k-1\}$
- On the one hand, this approach looks natural.
- On the other hand, it presents only the quantitative properties, while it can not present the qualitative properties.

Multiple-Valued Logic: classical view

- For example, we need to present different colors in terms of multiple-valued logic. Let Red=0, Orange=1, Yellow=2, Green=3, etc.
- What does it mean?
- Is it true that Red<Orange<Yellow<Green ??!

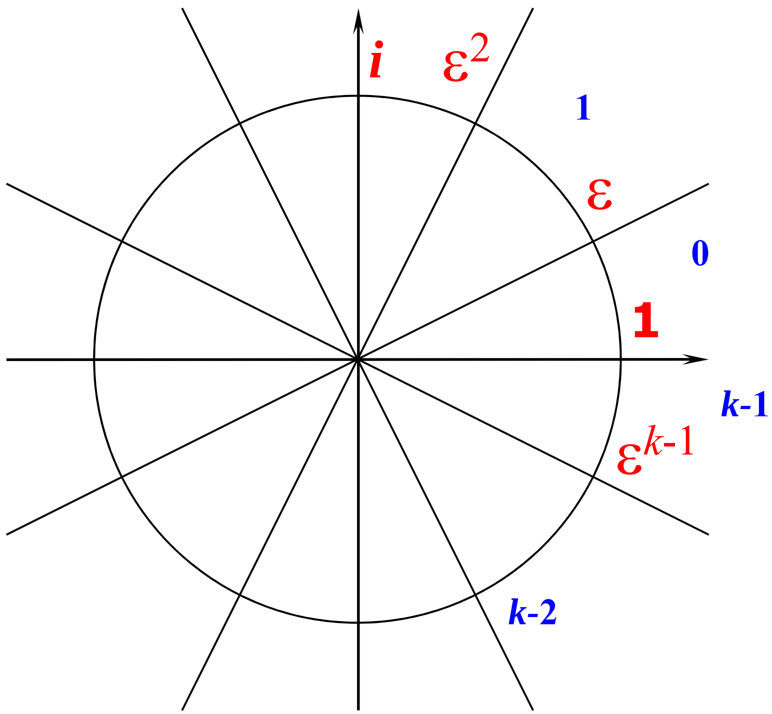
Multiple-Valued (k-valued) logic over the field of complex numbers

(Proposed and formulated by Naum Aizenberg in **1971-1973**)

- To represent and handle both the quantitative properties and the qualitative properties, it is possible to move to the field of complex numbers.
- In this case, the argument (phase) may be used to represent the quality and the amplitude may be used to represent the quantity if it is needed.

Multiple-Valued (k-valued) logic over the field of complex numbers

(Proposed and formulated by Naum Aizenberg in **1971-1973**)



$$j \in \{0,\ 1,\ \ldots,\ k-1\}$$

regular values of k-valued logic

$$j \to \varepsilon^j = \exp(i2\pi j / k)$$

one-to-one correspondence

$$\varepsilon^j \in \left\{\varepsilon^0, \varepsilon, \varepsilon^2, \ldots, \varepsilon^{k-1}\right\} = E_k$$

The ***k*^th^ roots of unity** are **values** of ***k*-valued logic over the field of complex numbers**

$$\varepsilon = \exp(i2\pi / k)$$

primitive k^{th} root of unity

Important Advantage

- In multiple-valued logic over the field of complex numbers all values of this logic are normalized: their absolute values are equal to 1
- Particularly, in the Boolean case (k=2) the values of logic will be encoded according to this approach by {1,-1}

Importance of Phase

- In the example with the colors, in terms of multiple-valued logic over the field of complex numbers they can be encoded by the corresponding phases (if f is the frequency, then $2\pi f$ is the phase). Hence, their quality is presented by the **phase**.
- Since the phase determines the corresponding frequency, this representation meats the physical nature of the colors.
- If now we want to "mix" colors as phases

Importance of Phase vs. Magnitude

- Oppenheim, A.V.; Lim, J.S., **The importance of phase in signals**, IEEE Proceedings, v. 69, No 5, 1981, pp.: 529- 541
- In this paper, it was shown that the **phase** in the Fourier spectrum of a signal is much more informative than the **magnitude**: particularly in the Fourier spectrum of images phase contains the information about all shapes, edges, orientation of all objects, etc.

Phase and Magnitude

Phase contains the information about an object presented by a signal

(a)

(b)

Phase (a) & **Magnitude** (b)

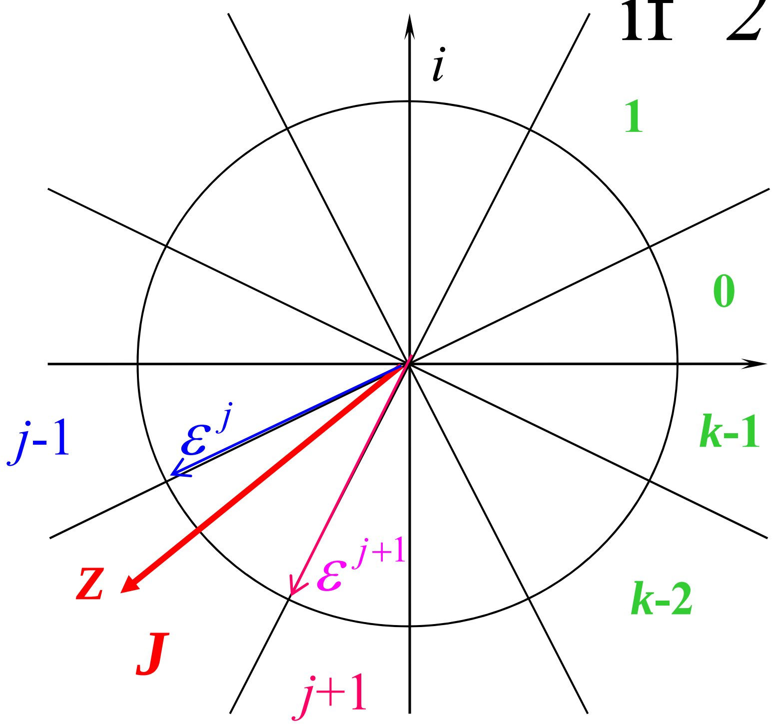
Phase (b) & **Magnitude** (a)

Discrete-Valued (k-valued) Activation Function

(introduced in **1971** by N. Aizenberg, the first historically known complex-valued activation function)

$$P(z) = \exp\left(i2\pi j/k\right) = \varepsilon^{j},$$

$$\text{if } 2\pi j/k \le arg(z) < 2\pi(j+1)/k$$

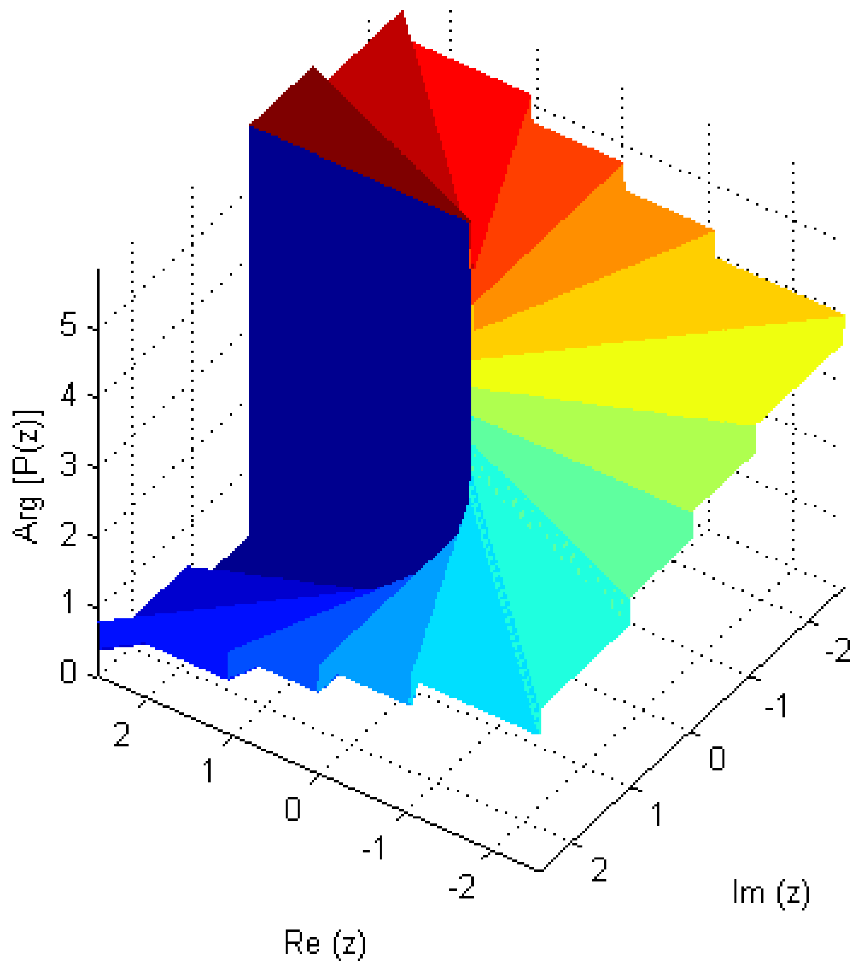


Function P maps the complex plane into the set of the k^{th} roots of unity

Naum Aizenberg

- Naum Aizenberg (1928-2002)
- Professor of Uzhgorod National University (Ukraine)
- Founder of multiple-valued threshold logic over the field of complex numbers and author of the first complex-valued activation function for artificial neurons (**1971**)

Discrete-Valued (k-valued) Activation Function



k=16

Multiple-Valued (k-Valued) Threshold Functions

The k-valued function $f(x_1,...,x_n): E_k^n \to E_k$ is called a k-valued threshold function, if such a complex-valued weighting vector $(w_0, w_1,..., w_n)$ exists that for all $X=(x_1,...,x_n)$ from the domain of the function f:

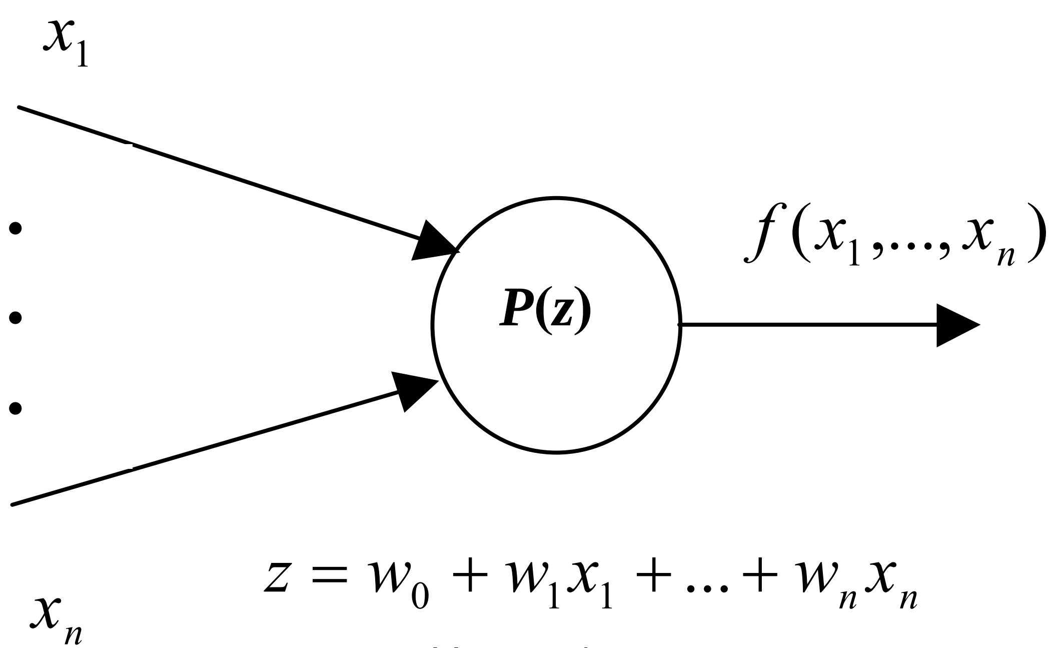
$$\boxed{f(x_1,...,x_n) = P(w_0 + w_1x_1 + ... + w_nx_n)}$$

Multi-Valued Neuron (MVN)

(introduced in 1992 by I. Aizenberg and N. Aizenberg)

$$f(x_1,\ldots,x_n) = P(w_0 + w_1x_1 + \ldots + w_nx_n)$$

f is a function of ***k***-valued logic
(***k***-valued threshold function)

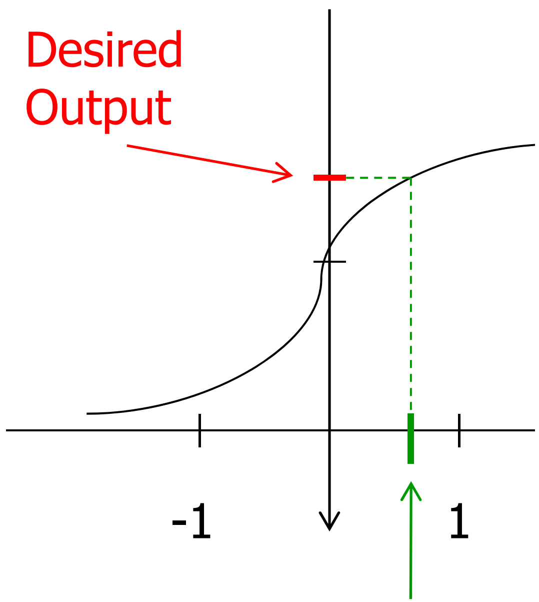


MVN: main properties

- The key properties of MVN:
  - Complex-valued weights
  - The activation function is a function of the argument of the weighted sum
  - Complex-valued inputs and output that are lying on the unit circle (k^{th} roots of unity)
  - Higher functionality than the one for the traditional neurons (e.g., sigmoidal)
  - Simplicity of derivative-free learning

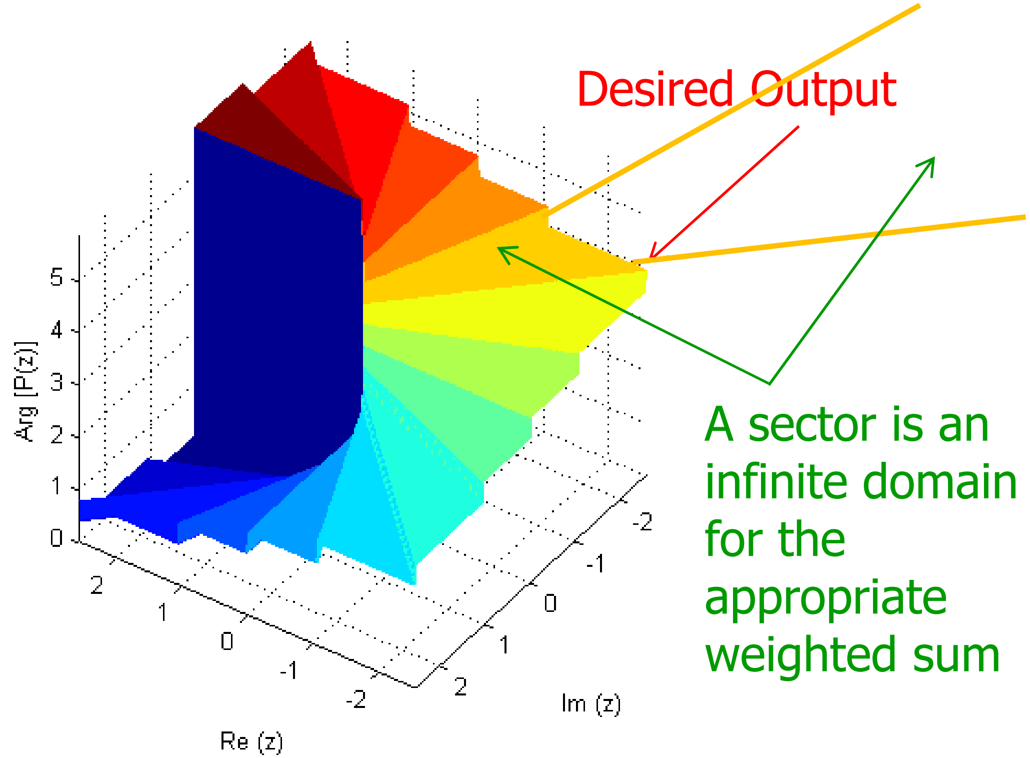
Higher functionality of MVN

SIGMOID ACTIVATION FUNCTION:



A single acceptable value of the weighted sum

MVN's MULTI-VALUED ACTIVATION FUNCTION:



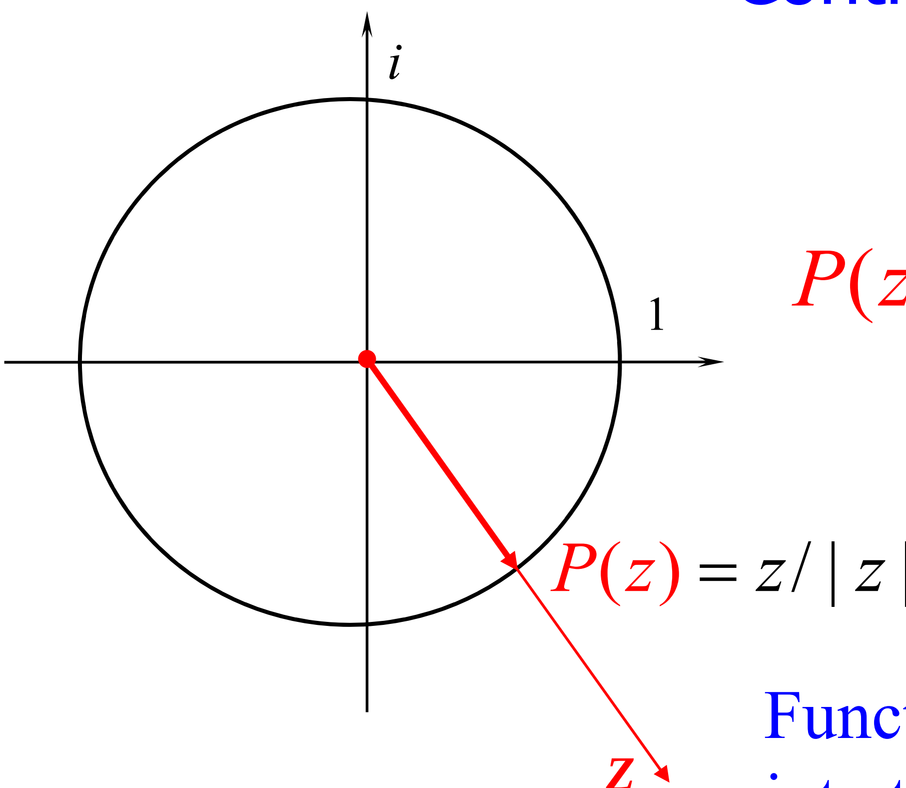
A sector is an infinite domain for the appropriate weighted sum

- MVN's multi-valued activation function makes this neuron more flexible and adaptive. As a result its functionality is much higher than the one of the sigmoidal neuron.
- It also learns faster and its learning algorithm is simpler

Continuous-Valued Activation Function

(introduced by I. Aizenberg and C. Moraga in 2004)

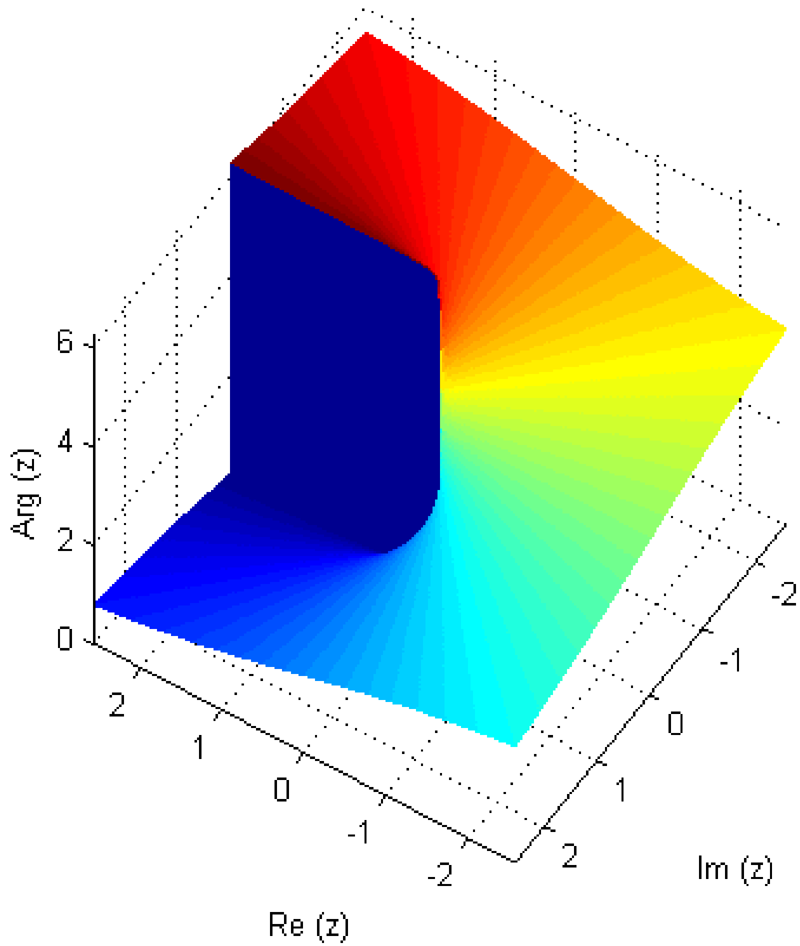
Continuous-valued case ($k \to \infty$):



$$P(z) = \exp(i\,(\arg z)) = e^{i \operatorname{Arg} z} =$$
$$= z / |z|$$

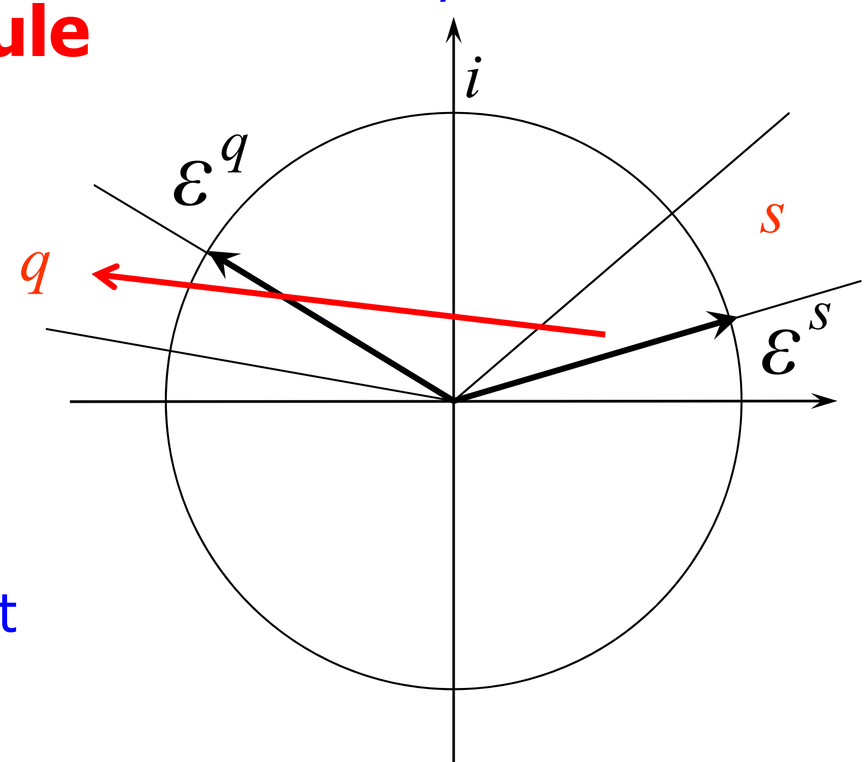
Function P maps the complex plane into the unit circle

Continuous-Valued Activation Function



MVN Learning

- Learning is reduced to movement along the unit circle
- Learning algorithm is **derivative-free**, it is based on the **error-correction rule**



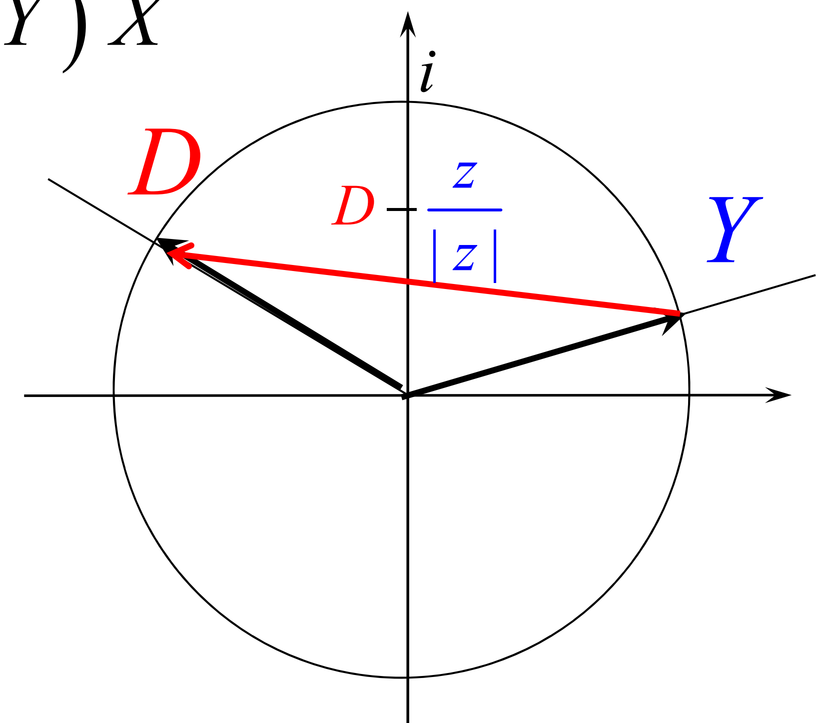
ε^q - Desired output

ε^s - Actual output

$\delta = \varepsilon^q - \varepsilon^s$ -error, which completely determines the weights adjustment

Error-Correction Learning Rule for the MVN

$$W_{r+1} = W_r + \frac{\alpha_r}{(n+1)\,|z_r|}(D - Y)\,\bar{X}$$



W – weighting vector; X - input vector

$\bar{X}$ is a complex conjugated to X

α_r – a learning rate (should always be =1)

r - current iteration;

r+1 – the next iteration

Z – the weighted sum

D is a desired output

Y is an actual output

$\delta = D - Y$ - neuron's error

A role of the factor 1/(n+1) in the Learning Rule

$$\boxed{\delta = \varepsilon^q - \frac{z}{|z|}}$$ - neuron's error

The weights after the correction:

$$\tilde{w}_0 = w_0 + \frac{\delta}{(n+1)}; \quad \tilde{w}_1 = w_1 + \frac{\delta}{(n+1)}\bar{x}_1; \quad \ldots; \quad \tilde{w}_n = w_n + \frac{\delta}{(n+1)}\bar{x}_n$$

The weighted sum after the correction:

$$\tilde{z} = \tilde{w}_0 + \tilde{w}_1 x_1 + \ldots + \tilde{w}_n x_n =$$

$$= (w_0 + \frac{\delta}{(n+1)}) + (w_1 + \frac{\delta}{(n+1)}\bar{x}_1)x_1 + \ldots + (w_1 + \frac{\delta}{(n+1)}\bar{x}_n)x_n =$$

$$= w_0 + \frac{\delta}{(n+1)} + w_1 x_1 + \frac{\delta}{(n+1)} + \ldots + w_n x_n + \frac{\delta}{(n+1)} =$$

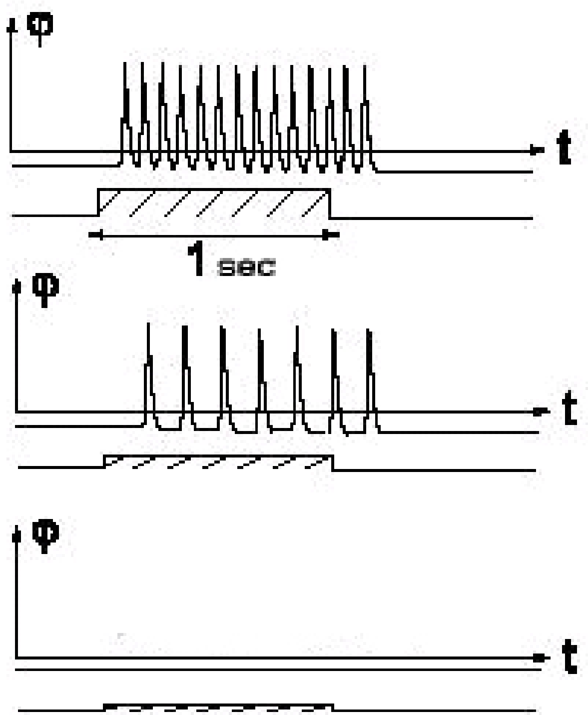
$$= w_0 + w_1 x_1 + \ldots + w_n x_n + \delta = z + \delta$$ - exactly what we are looking for

Convergence of the Learning Algorithm

- It is proven that the MVN learning algorithm requires a finite number of learning iterations to converge.
- Since the MVN learning is not reduced to solving some optimization problem, it does not suffer from the local minima problems.
- Learning for the continuous MVN with the precision λ is reduced to learning for the discrete MVN in π/λ –valued logic.

MVN as a model of a biological neuron

- The State of a biological neuron is determined by the **frequency of the generated impulses**
- The magnitude of impulses is always a constant



Excitation → High frequency

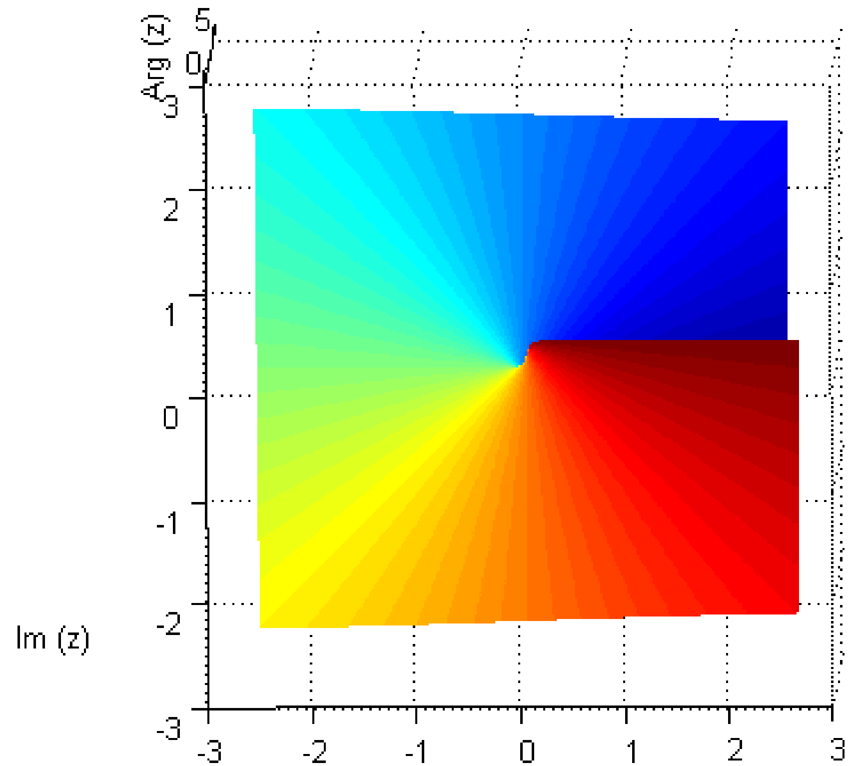
Intermediate State → Medium frequency

No impulses → Inhibition → Zero frequency

MVN as a model of a biological neuron

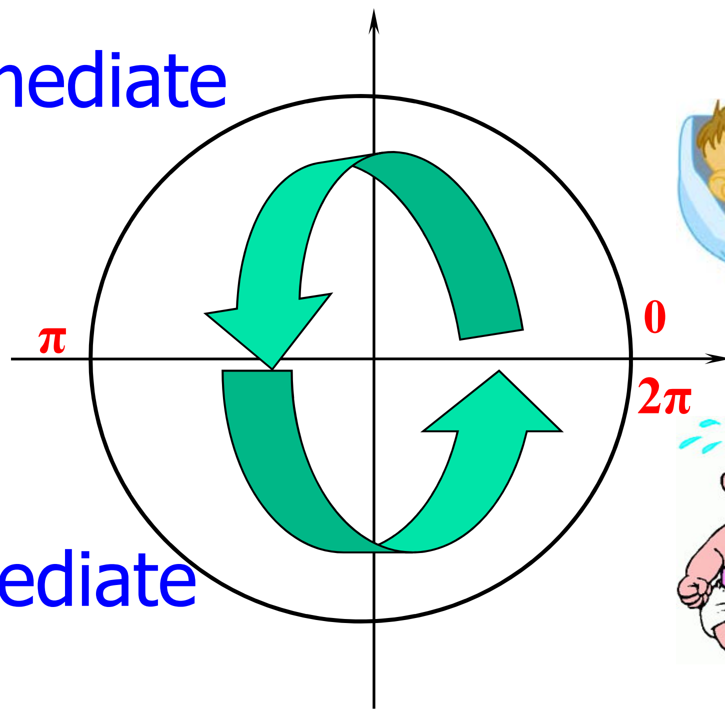
- The state of a biological neuron is determined by the frequency of the generated impulses
- The amplitude of impulses is always a constant
- The state of the multi-valued neuron is determined by the argument (phase) of the weighted sum
- The amplitude of the state of the multi-valued neuron is always a constant
- If f is the frequency then $2\pi f$ is the corresponding phase

MVN as a model of a biological neuron



MVN as a model of a biological neuron

Intermediate State

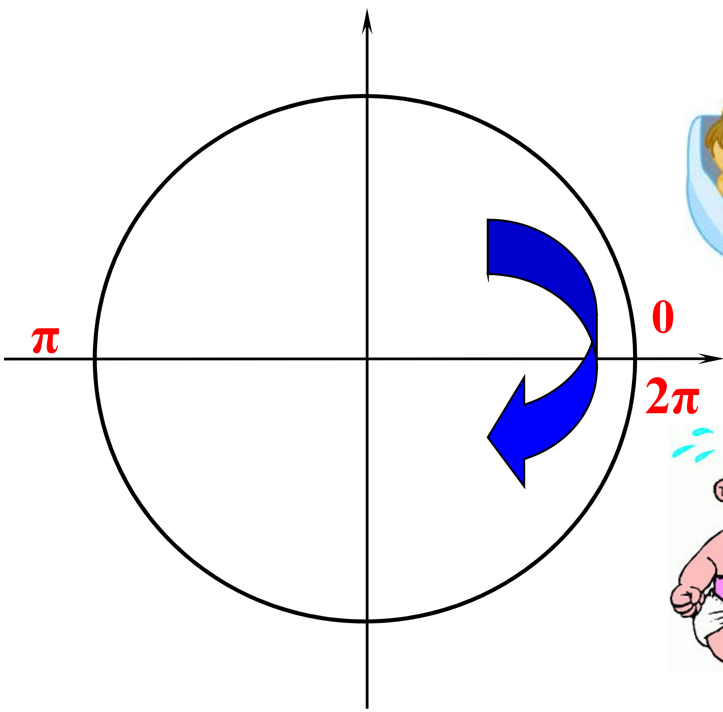


Maximal inhibition

Intermediate State

Maximal excitation

MVN as a model of a biological neuron



Maximal inhibition

Maximal excitation

MVN:

- Learns faster
- Adapts better
- Learns even highly nonlinear functions
- Opens new very promising opportunities for the network design
- Is much closer to the biological neuron
- Allows to use the Fourier Phase Spectrum as a feature space for solving different recognition/classification problems
- Allows to use hybrid (discrete/continuous) inputs/output

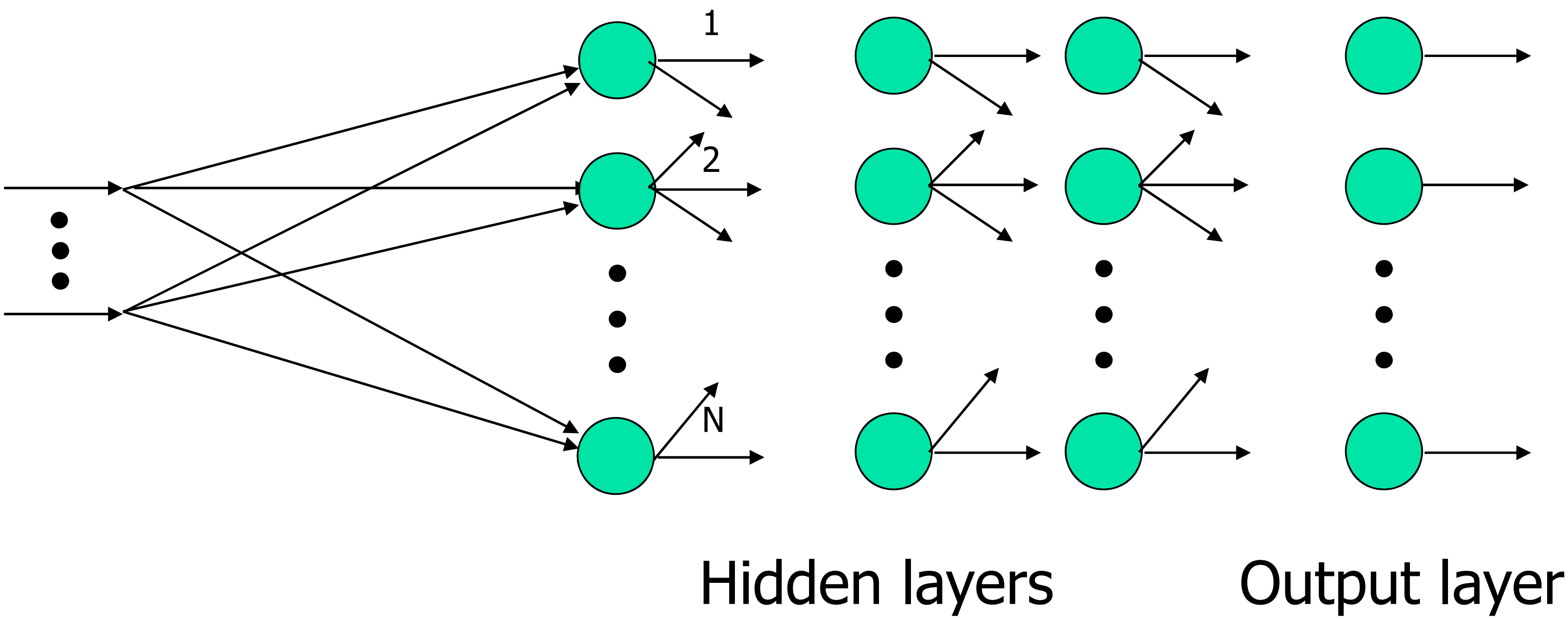
MVN: Applications

- Multilayer feedforward MVN-based neural network (2005-2008), which outperforms other neural networks, kernel-based networks including SVM in terms of learning speed and quality, and in terms of classification/prediction rate when solving many benchmark and real-world problems
- Different MVN-based associative memories (1992-2004) with a higher retrieval rate
- MVN-based cellular neural networks for nonlinear image filtering

MVN- based Multilayer Feedforward Neural Network (MLMVN) and a Backpropagation Learning Algorithm

Introduced in 2004-2007 by I. Aizenberg, C. Moraga, and D. Paliy
Developed in 2008 by I. Aizenberg, D. Paliy, J. Zurada, and J. Astola

MVN- based Multilayer Feedforward Neural Network



MLMVN: Key Properties

- Derivative-free learning algorithm based on the error-correction rule
- Self-adaptation of the learning rate for all the neurons
- Much faster learning than the one for other neural networks
- A single step is always enough to adjust the weights for the given set of inputs independently on the number of hidden and output neurons
- Better recognition/prediction/classification rate in comparison with other neural networks, neuro-fuzzy networks and kernel based techniques including SVM

MLMVN: Key Properties

- MLMVN can operate with both continuous and discrete inputs/outputs, as well as with the hybrid inputs/outputs:
- continuous inputs → discrete outputs
- discrete inputs → continuous outputs
- hybrid inputs → hybrid outputs

A Backpropagation Derivative-Free Learning Algorithm

D_{km} - a desired output of the k^{th} neuron from the m^{th} (output) layer

Y_{km} - an actual output of the k^{th} neuron from the m^{th} (output) layer

$\delta^*_{km} = D_{km} - Y_{km}$ - the network error for the k^{th} neuron from output layer

$\delta_{km} = \frac{1}{s_m}\delta^*_{km}$ - the error for the k^{th} neuron from output layer

$s_m = N_{m-1} + 1$ -the number of all neurons on the previous layer (m-1, to which the error is backpropagated) incremented by 1

A Backpropagation Derivative-Free Learning Algorithm

The error backpropagation:

The error for the k^{th} neuron from the hidden (j^{th}) layer, j=1, …, m-1

$$\delta_{kj} = \frac{1}{s_j} \sum_{i=1}^{N_{j+1}} \frac{1}{\left| w_k^{ij+1} \right|^2} \delta_{ij+1} (\overline{w}_k^{ij+1}) = \frac{1}{s_j} \sum_{i=1}^{N_{j+1}} \delta_{ij+1} (w_k^{ij+1})^{-1}$$

$$s_j = N_{j-1} + 1, \quad j = 2, \ldots, m; \;\; s_1 = 1$$

-the number of all neurons on the previous layer
(previous to j, to which the error is backpropagated) incremented by 1

A Backpropagation Derivative-Free Learning Algorithm

Correction rule for the neurons from the m^{th} (output) layer (k^{th} neuron of m^{th} layer):

$$\tilde{w}_i^{km} = w_i^{km} + \frac{1}{(n+1)} \delta_{km} \bar{\tilde{Y}}_{im-1}, \quad i = 1, \ldots, n$$

$$\tilde{w}_0^{km} = w_0^{km} + \frac{1}{(n+1)} \delta_{km}$$

A Backpropagation Derivative-Free Learning Algorithm

Correction rule for the neurons from the 2nd through m-1st layer (k^{th} neuron of the j^{th} layer (j=2, …, m-1):

$$\tilde{w}_i^{kj} = w_i^{kj} + \frac{1}{(n+1)|z_{kj}|}\delta_{kj}\bar{\tilde{Y}}_i, \quad i = 1,\ldots,n$$

$$\tilde{w}_0^{kj} = w_0^{kj} + \frac{1}{(n+1)|z_{kj}|}\delta_{kj}$$

A Backpropagation Derivative-Free Learning Algorithm

Correction rule for the neurons from the 1st hidden layer:

$$\tilde{w}_i^{k1} = w_i^{k1} + \frac{1}{(n+1)|z_{k1}|}\delta_{k1}\bar{x}_i, \quad i = 1, \ldots, n$$

$$\tilde{w}_0^{k1} = w_0^{k1} + \frac{1}{(n+1)|z_{k1}|}\delta_{k1}$$

Criteria for the convergence of the learning process

- Learning should continue until either minimum MSE/RMSE criterion will be satisfied or zero-error is reached.
- It is proven in the same way as for a single MVN that the learning process needs a finite number of iterations to converge.
- Since MLMVN learning is not considered as the optimization problem, there is no local minima problem for the learning process.

MLMVN Learning: Example

Suppose, we need to classify three vectors belonging to three different classes:

$$X_1 = (\exp(4.23i), \exp(2.10i)) \to \tilde{T}_1,$$

$$X_2 = (\exp(5.34i), \exp(1.24i)) \to \tilde{T}_2,$$

$$X_3 = (\exp(2.10i), \exp(0i)) \quad \to \tilde{T}_3.$$

$$T_1 = \exp\left(0.76i\right),\ T_2 = \exp\left(2.56i\right),\ T_3 = \exp\left(5.35i\right).$$

Classes $\tilde{T}_1, \tilde{T}_2, \tilde{T}_3$

are determined in such a way that the argument of the desired output of the network must belong to the interval

$$[\arg(T_j) - 0.05, \arg(T_j) + 0.05],\ j = 1, 2, 3,$$

MLMVN Learning: Example

Thus, we have to satisfy the following conditions:

$$\left|\arg\left(T_j\right) - \arg\left(e^{iArg\, z}\right)\right| \le 0.05$$, where

$e^{iArg\, z}$ is the actual output.

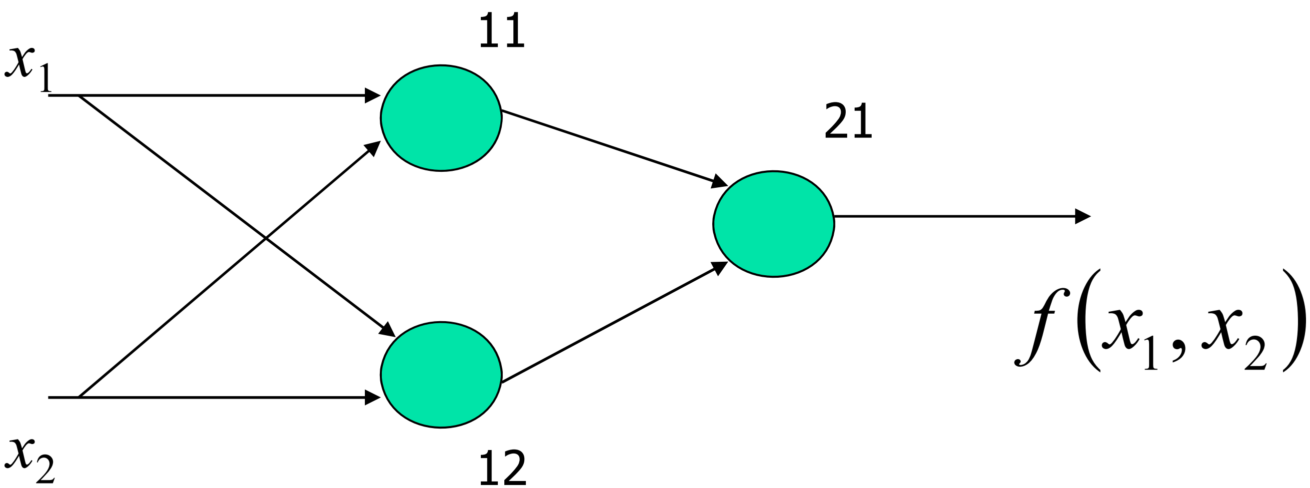
and for the mean square error

$$E \le 0.05^2 = 0.0025$$

MLMVN Learning: Example

Let us train the 2→1 MLMVN

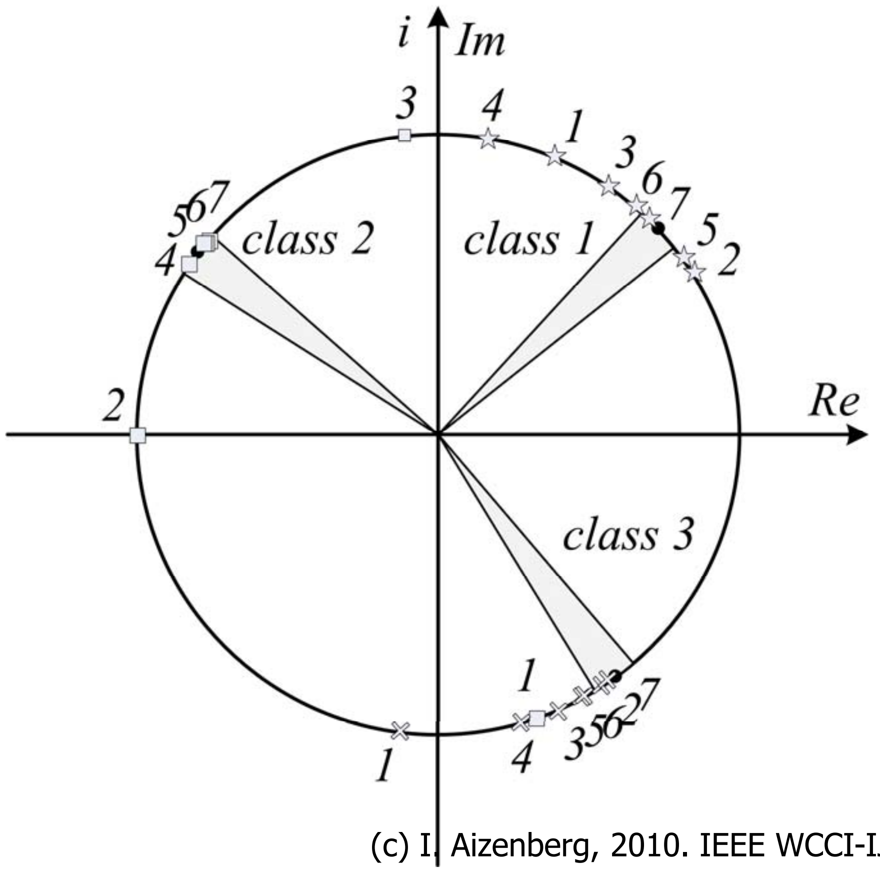
(two hidden neurons and the single neuron in the output layer



MLMVN Learning: Example

The training process converges after 7 training epochs.

Update of the outputs:

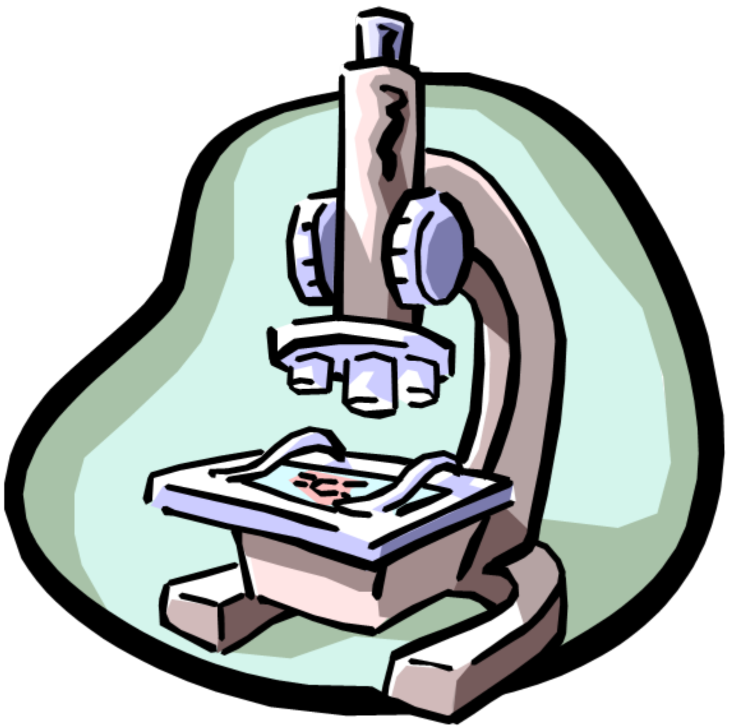


MLMVN Learning: Example

Epoch	1	2	3	4	5	6	7
MSE	2.4213	0.1208	0.2872	0.1486	0.0049	0.0026	0.0009

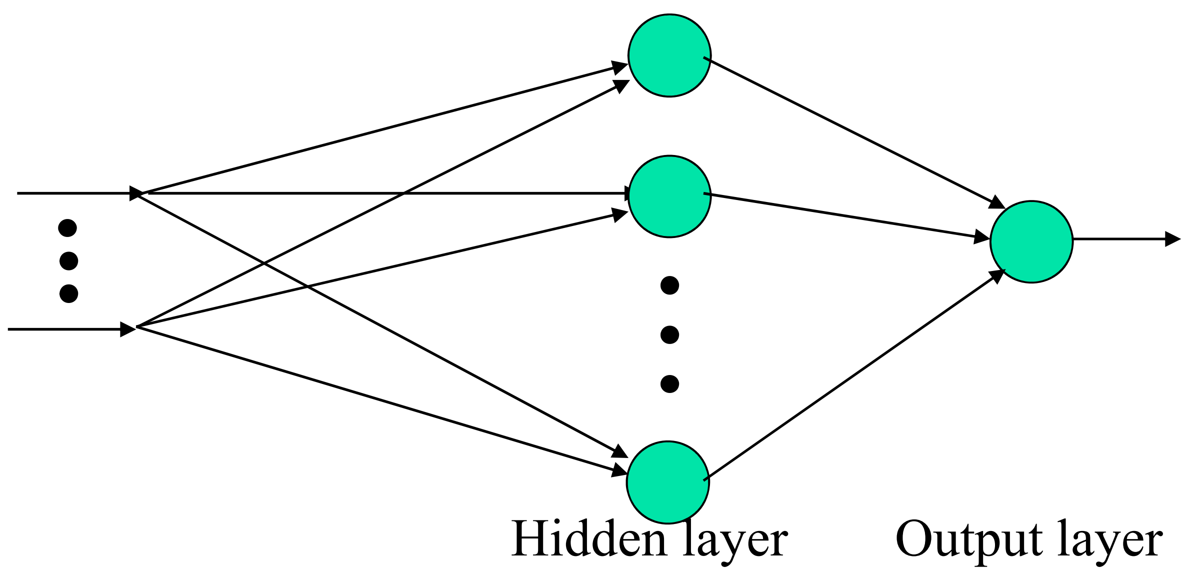
$$E \leq 0.05^2 = 0.0025$$

MLMVN: Simulation Results



Simulation Results: Mackey-Glass time series prediction

All simulation results for the Mackey-Glass time series prediction
are obtained using the network with *n* inputs
n→*S*→1 containing
a single hidden layer with *S* neurons
and a single neuron in the output layer:



Mackey-Glass time series prediction

Mackey-Glass differential delay equation:

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) + n(t),$$

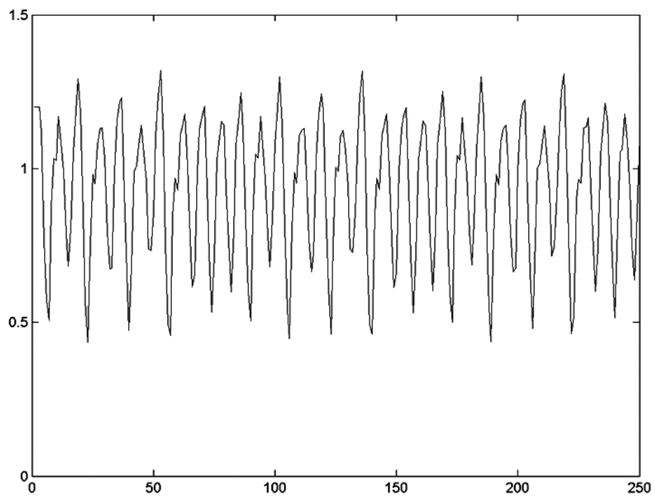
.

The task of prediction is to predict $x(t+6)$

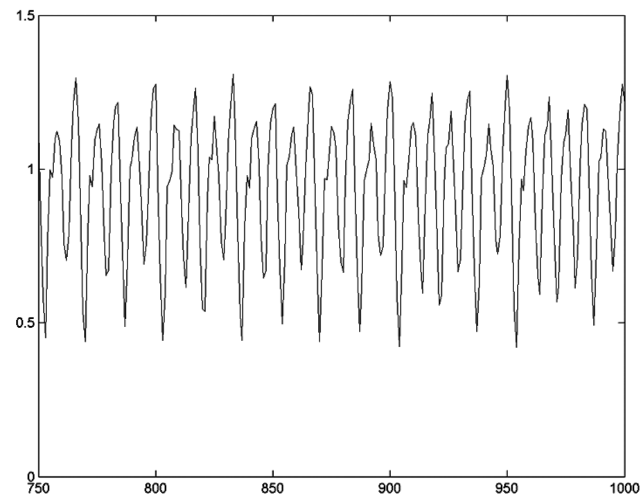
from $x(t), x(t-6), x(t-12), x(t-18)$

Mackey-Glass time series prediction

Training Data:

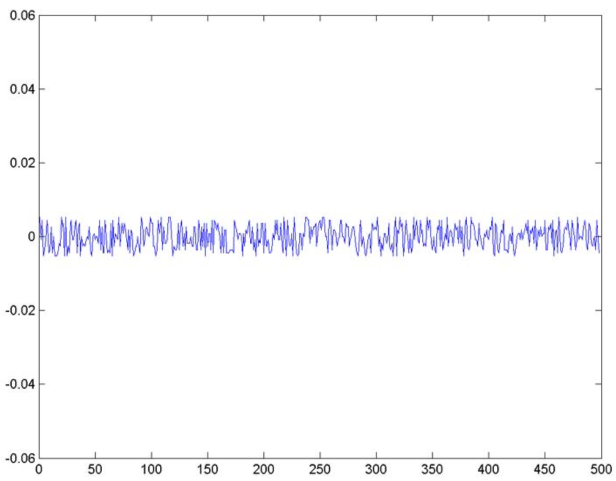


Testing Data:

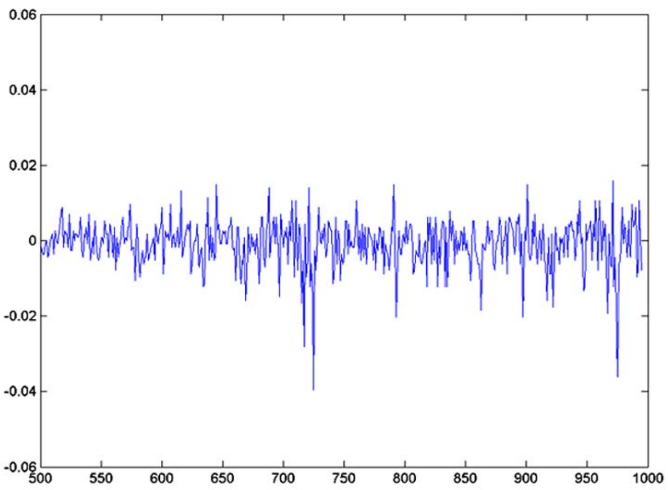


Mackey-Glass time series prediction

RMSE Training:



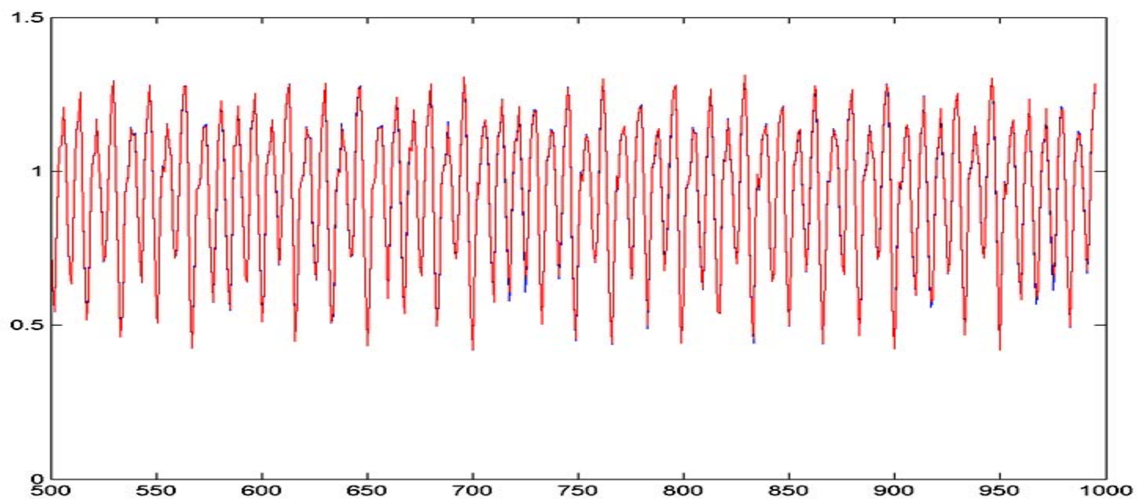
RMSE Testing:



Mackey-Glass time series prediction

Testing Results:

Blue curve – the actual series;
Red curve – the predicted series



Mackey-Glass time series prediction

The results of 30 independent runs:

# of neurons on the hidden layer		**50**	**50**	**40**
ε- a maximum possible **RMSE**		0.0035	0.0056	0.0056
Actual RMSE for the training set (min - max)		0.0032 - 0.0035	0.0053 – 0.0056	0.0053 – 0.0056
RMSE for the **testing set**	Min	**0.0056**	**0.0083**	**0.0086**
	Max	**0.0083**	0.0101	0.0125
	Median	**0.0063**	**0.0089**	**0.0097**
	Average	**0.0066**	**0.0089**	**0.0098**
	SD	0.0009	0.0005	0.0011
Number of training epochs	Min	95381	24754	34406
	Max	272660	116690	137860
	Median	145137	56295	62056
	Average	162180	58903	70051

Mackey-Glass time series prediction

Comparison of MVN to other models:

MLMVN min	MLMVN average	GEFREX M. Rosso, Generic Fuzzy Learning, 2000 min	EPNet You-Liu, Evolution. system, 1997	ANFIS J.S. R. Jang, Neuro-Fuzzy, 1993	CNNE M.M.Islam et all., Neural Networks Ensembles, 2003	SuPFuNIS S.Paul et all, Fuzzy-Neuro, 2002	Classical Backprop. NN 1994
0.0056	**0.0066**	**0.0061**	0.02	0.0074	**0.009**	0.014	0.02

MLMVN outperforms all other networks in:

- The number of either hidden neurons or supporting vectors
- Speed of learning
- Prediction quality

Prediction of Dow Jones Index

- MLMVN 250→2→32768→1 has been used
- Prediction of the next value from the previous 250 values (the approximate amount of the business days within a year)

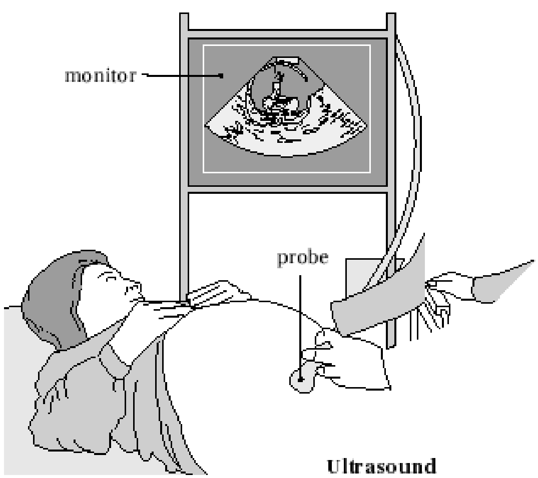
Blur Classification for Image Restration

- Photo

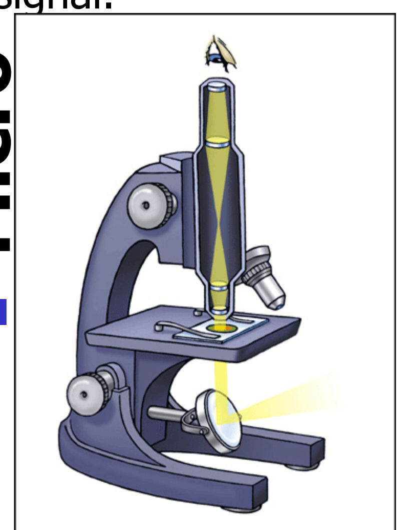
- Mathematically a variety of capturing principles can be described by the Fredholm integral of the first kind

$$z(x)=\int_{\mathbb{R}^2} v(x,t)\,y(t)\,dt, \quad x,t\in\mathbb{R}^2$$

- where x,t $\mathbb{R}^2$, $v(t)$ is a point-spread function (PSF) of a system, $y(t)$ is a function of a real object and $z(x)$ is an observed signal.

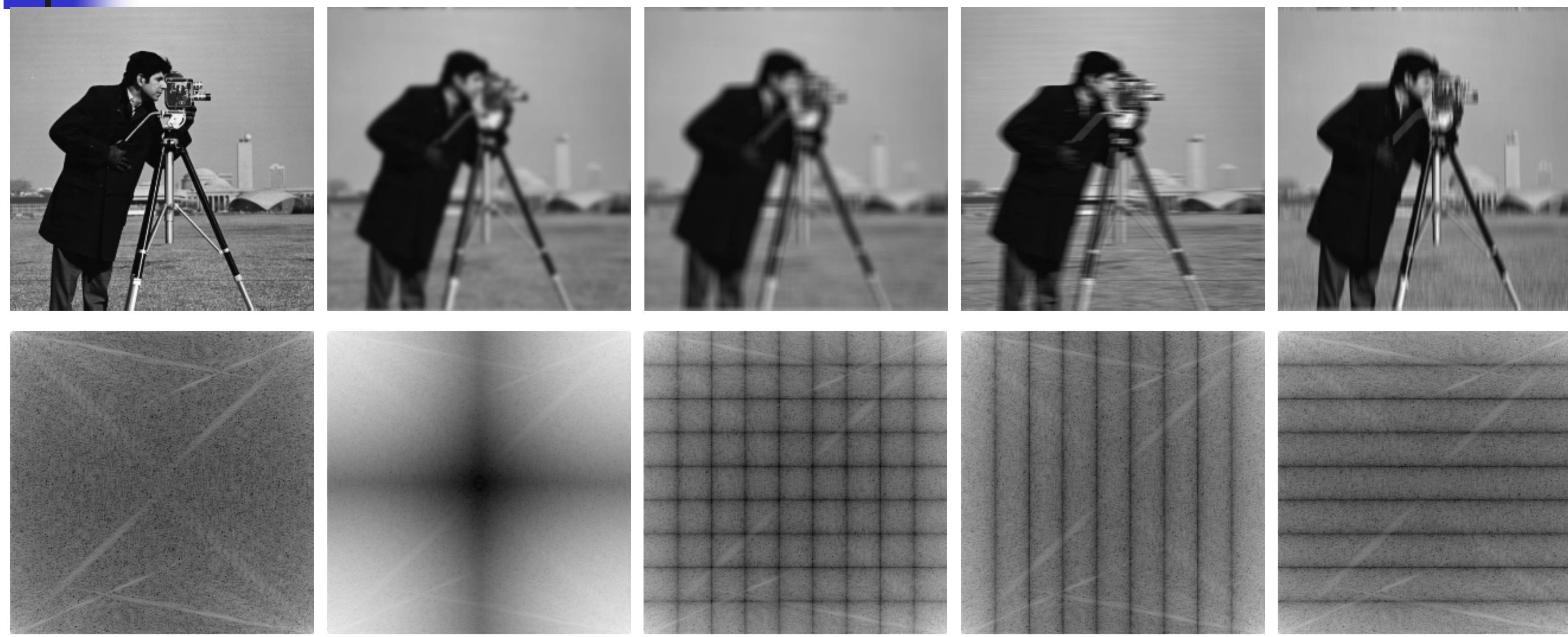


- Micro



- Tomography

Degradation in the frequency domain:



True Image — Gaussian — Rectangular — Horizontal Motion — Vertical Motion

Images and log of their Power Spectra $\log|Z|$

Simulation

2700 training pattern vectors corresponding to 72 images: six types of blur with the following parameters:

MLMVN structure: 5→35→6

1) The Gaussian blur is considered with $\tau \in \{1,\ 1.33,\ 1.66,\ 2,\ 2.33,\ 2.66,\ 3\};$

2) The linear uniform horizontal motion blur of the lengths 3, 5, 7, 9;

3) The linear uniform vertical motion blur of the length 3, 5, 7, 9;

4) The linear uniform diagonal motion from South-West to North-East blur of the lengths 3, 5, 7, 9;

5) The linear uniform diagonal motion from South-East to North-West blur of the lengths 3, 5, 7, 9;

6) rectangular has sizes 3x3, 5x5, 7x7, 9x9.

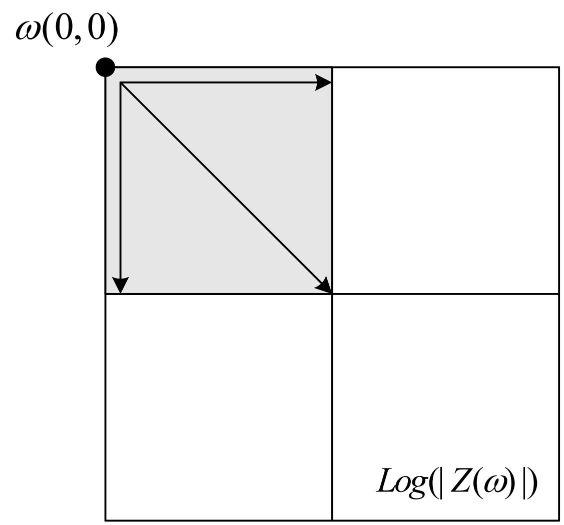
Training Vectors

The **training vectors** $X=(x_1,...,x_n)$ are formed as follows:

$$x_j = \exp\left(2\pi i \cdot (K-1) \frac{\log\left(\left|Z\left(\omega_{k_1,k_2}\right)\right|\right) - \log\left(\left|Z_{\min}\right|\right)}{\log\left(\left|Z_{\max}\right|\right) - \log\left(\left|Z_{\min}\right|\right)} \right),$$

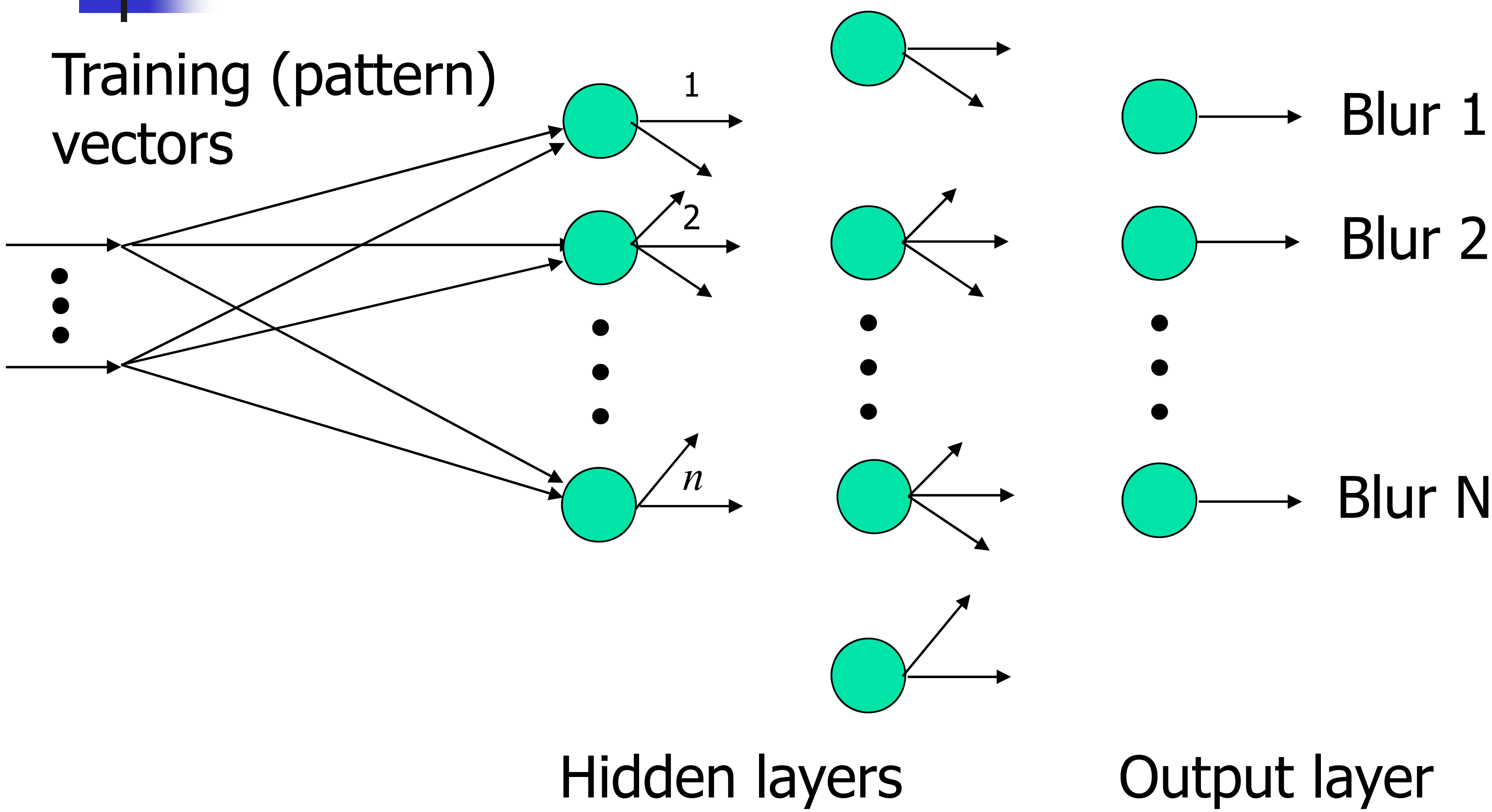
for

$$\begin{cases} j=1,...,L/2-1, & \text{for } k_1=k_2,\ k_2=1,...,L/2-1, \\ j=L/2,...,L-2, & \text{for } k_1=1,\ k_2=1,...,L/2-1, \\ j=L-1,...,3L/2-3, & \text{for } k_2=1,\ k_1=1,...,L/2-1, \end{cases}$$

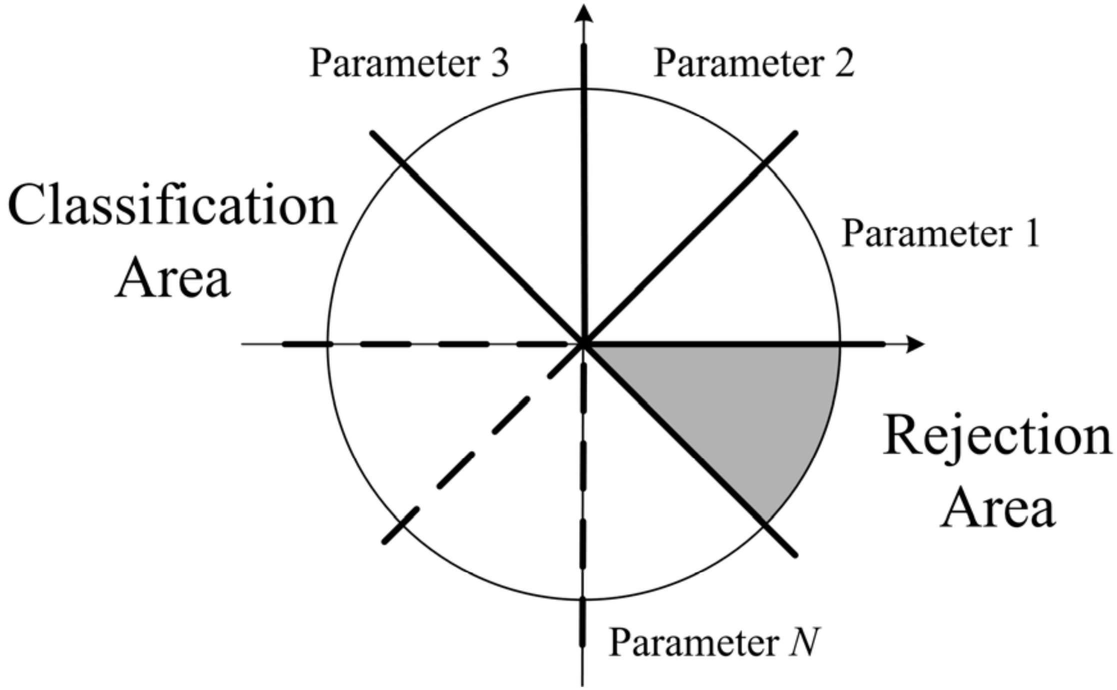


- ***LxL*** is a size of an image
- the length of the **pattern vector** is ***n= 3L/2-3***

Neural Network 5→35→6



Output Layer Neuron



Reservation of domains on the unit circle for the output neuron

Results

Classification Results

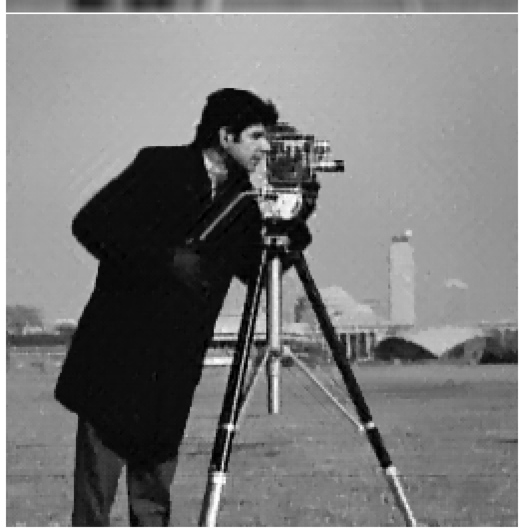
Blur	**MLMVN,** 381 inputs, 5→35→6, **2336 weights in total**	**SVM** Ensemble from 27 binary decision SVMs, **25.717.500 support vectors in total**
No blur	**96.0%**	**100.0%**
Gaussian	**99.0%**	**99.4%**
Rectangular	**99.0%**	96.4
Motion horizontal	**98.5%**	96.4
Motion vertical	**98.3%**	96.4
Motion North-East Diagonal	**97.9%**	96.5
Motion North-West Diagonal	**97.2%**	96.5

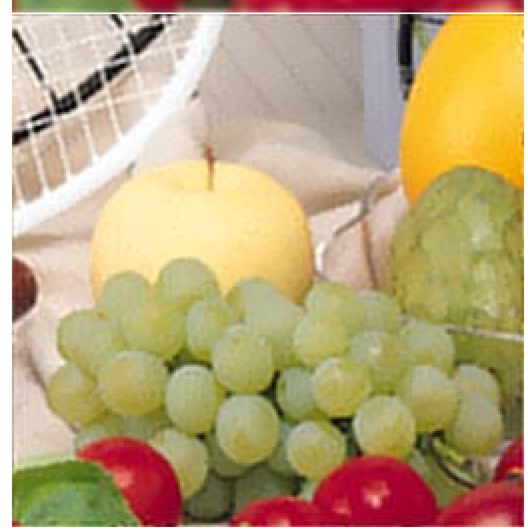
Restored images

Blurred noisy image: rectangular 9x9

Blurred noisy image: Gaussian, $\sigma=2$

Restored





Restored

Multi-Valued Neuron with a Periodic Activation Function

(to be considered in detail in the regular presentation on Thursday, July 22, Special Session on Complex-Valued Neural Networks, 8-40)

Some Prospective Applications

- Solving different recognition and classification problems, especially those, where the formalization of the decision rule is a complicated problem
  - Simulation of a biological neuron. Classification and prediction in biology including protein secondary structure prediction
- Time series prediction
- Modeling of complex systems including hybrid systems that depend on many parameters
- Nonlinear filtering (MLMVN can be used as a nonlinear filter)
- Etc. …

Main Publications (early)

1. N.N. Aizenberg, Yu. L.Ivaskiv and D.A. Pospelo v, "About one Generalization of the Threshold Function", ***Doklady Akademii Nauk SSSR*** **(*The Reports of the Academy of Sciences of USSR*)**, vol. 196, No. 6, **1971**, pp. 1287–1290. (in Russian).
2. N.N. Aizenberg, Yu.L. Ivaskiv, D.A. Pospelov, and G.F. Hudiakov, "Multi-Valued Threshold Functions. 1. The Boolean Complex-Threshold Functions and their Generalization", ***Kibernetika*** **(*Cybernetics*)**, No. 4, **1971**, pp. 44-51 (in Russian).
3. N.N. Aizenberg, Z. Toshich. A Generalization of the Threshold Functions, // ***Proceedings of the Electronic Department University of Belgrade***, No.399, **1972**, pp.97-99.
4. N.N. Aizenberg, Yu.L. Ivaskiv, D.A. Pospelov, and G.F. Hudiakov, "Multi-Valued Threshold Functions. 2. Synthesis of the Multi-Valued Threshold Element" ***Kibernetika*** **(*Cybernetics*)**, No. 1, **1973**, pp. 86-94 (in Russian).
5. N.N. Aizenberg and Yu.L. Ivaskiv *Multiple-Valued Threshold Logic*, Naukova Dumka Publisher House, Kiev, **1977** (in Russian).
6. I.N. Aizenberg, "Model of the Element with a Complete Functionality", *Izvestia AN SSSR. Technicheskaya Kibernetika ("The News of the USSR Academy of Sciences. Technical Cybernetics"),* **1985**, No. 2, pp. 188-191 (in Russian).
7. I.N. Aizenberg, "The Universal Logical Element over the Field of the Complex Numbers", ***Kibernetika i Sistemnyi Analiz*** **(*Cybernetics and Systems Analysis*)**, No.3, **1991**, pp.116-121 (in Russian).

* Official Ehglish Translations of the Russian journals are available from Springer

Main Publications (later)

8. N.N.Aizenberg and I.N.Aizenberg, "CNN Based on Multi-Valued Neuron as a Model of Associative Memory for Gray-Scale Images", *Proceedings of the Second IEEE International Workshop on Cellular Neural Networks and their Applications,* Technical University Munich, Germany October 14-16, **1992**, pp.36-41.
9. N.N.Aizenberg, I.N. Aizenberg and G.A.Krivosheev, "Method of Information Processing" **Patent of Russia** No 2103737, G06 K 9/00, G 06 F 17/10, priority date **23.12.1994**.
10. N.N. Aizenberg, I.N. Aizenberg and G.A. Krivosheev, "Multi-Valued Neurons: Learning, Networks, Application to Image Recognition and Extrapolation of Temporal Series", *Lecture Notes in Computer Science, vol. 930 (J.Mira, F.Sandoval - Eds.), Springer-Verlag,* **1995**,pp.389-395.
11. S. Jankowski, A. Lozowski and J. M. Zurada,, "Complex-Valued Multistate Neural Associative Memory", ***IEEE Transactions on Neural Networks***, vol. 7, **1996**, pp. 1491-1496.
12. H. Aoki and Y. Kosugi, "An Image Storage System Using Complex-Valued Associative Memory", *Proc. of the 15th International Conference on Pattern Recognition*, Barcelona, 2000, IEEE Computer Society Press, vol. 2, **2000**, pp. 626-629.
13. I. Aizenberg., N. Aizenberg, and J. Vandewalle, "*Multi-valued and universal binary neurons: theory, learning, applications*" (**monograph book**), Kluwer Academic Publishers, Boston/Dordrecht/London, **2000**.

Main Publications (recent)

14. H. Aoki, E. Watanabe, A. Nagata and Y. Kosugi, "Rotation-Invariant Image Association for Endoscopic Positional Identification Using Complex-Valued Associative Memories", *Bio-inspired Applications of Connectionism, Lecture Notes in Computer Science*, (J. Mira, A. Prieto - Eds.), vol. 2085 Springer **2001**, pp. 369-374.
15. M. K. Muezzinoglu, C. Guzelis, and J.M. Zurada "A New Design Method for the Complex-Valued Multistate Hopfield Associative Memory", ***IEEE Transactions on Neural Networks*** vol. 14, **2003**, pp. 891-899.
16. I. Aizenberg and C. Moraga, "Multilayer Feedforward Neural Network based on Multi-Valued Neurons and a Backpropagation Learning Algorithm", ***Soft Computing***, vol. 11, No 2, January, **2007**, pp. 169-183.
17. I. Aizenberg and J. Zurada., "Solving Selected Classification Problems in Bioinformatics Using Multilayer Neural Network based on Multi-Valued Neurons (MLMVN)", *Proceedings of the International Conference on Artificial Neural Networks (ICANN-2007)*, *Lecture Notes in Computer Science* (J. Marques de Sá et al. -Eds.), vol. 4668, Part I, Springer, Berlin, Heidelberg, New York, **2007**, pp. 874-883.
18. I. Aizenberg and C. Moraga, "The Genetic Code as a Function of Multiple-Valued Logic Over the Field of Complex Numbers and its Learning using Multilayer Neural Network Based on Multi-Valued Neurons", ***Journal of Multiple-Valued Logic and Soft Computing***, No 4-6, November **2007**, pp. 605-618
19. I. Aizenberg, "Solving the XOR and Parity *n* Problems Using a Single Universal Binary Neuron", ***Soft Computing***, vol. 12, No 3, February **2008**, pp. 215-222.
20. I. Aizenberg, D. Paliy, J. Zurada, and J. Astola, "Blur Identification by Multilayer Neural Network based on Multi-Valued Neurons", ***IEEE Transactions on Neural Networks***, vol. 19, No 5, May **2008**, pp. 883-898.

Main Publications (recent)

21. I. Aizenberg, "A Multi-Valued Neuron with a Periodic Activation Function", *Proceedings of the International Joint Conference on Computational Intelligence*, Funchal-Madeira, Portugal, October 5-7, 2009, pp. 347-354.
22. I. Aizenberg, “Learning of the Non-Threshold Functions of Multiple-Valued Logic by a Single Multi-Valued Neuron With a Periodic Activation Function”, accepted for publication in *Proceedings of the 40th IEEE International Symposium on Multiple-Valued Logic* (ISMVL-2010), Barcelona, Spain, May 26-28, 2010, IEEE Computer Society Press, pp. 33-38.
23. I. Aizenberg,"Complex-Valued Neurons with Phase-Dependent Activation Functions", in book "Artificial Intelligence and Soft Computing" (Rutkowski L. et al. – Eds.), Springer Series "Lecture Notes in Computer Science", vol. 6114, Part II, 2010, pp. 3-10.