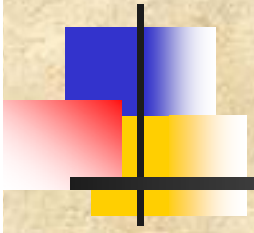
2013 International Joint Conference on Neural Networks – IJCNN-2013

Tutorial

Complex-Valued Neural Networks with Multi-Valued Neurons



Igor Aizenberg

Texas A&M University-Texarkana

Computational Intelligence Laboratory

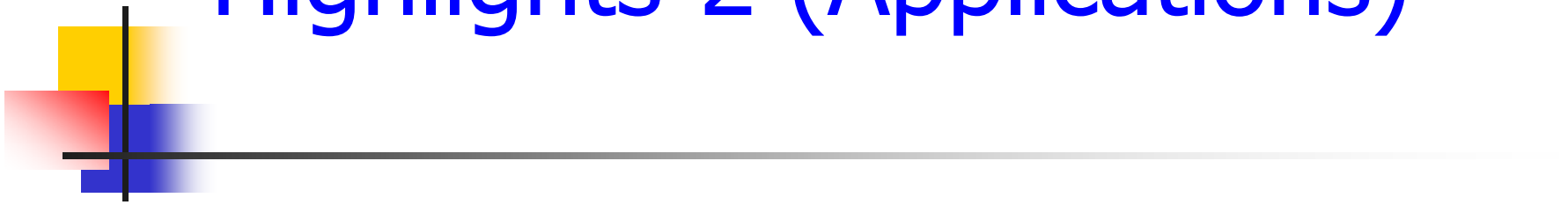
igor.aizenberg@tamut.edu

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

Highlights-1 (Theoretical Aspects)

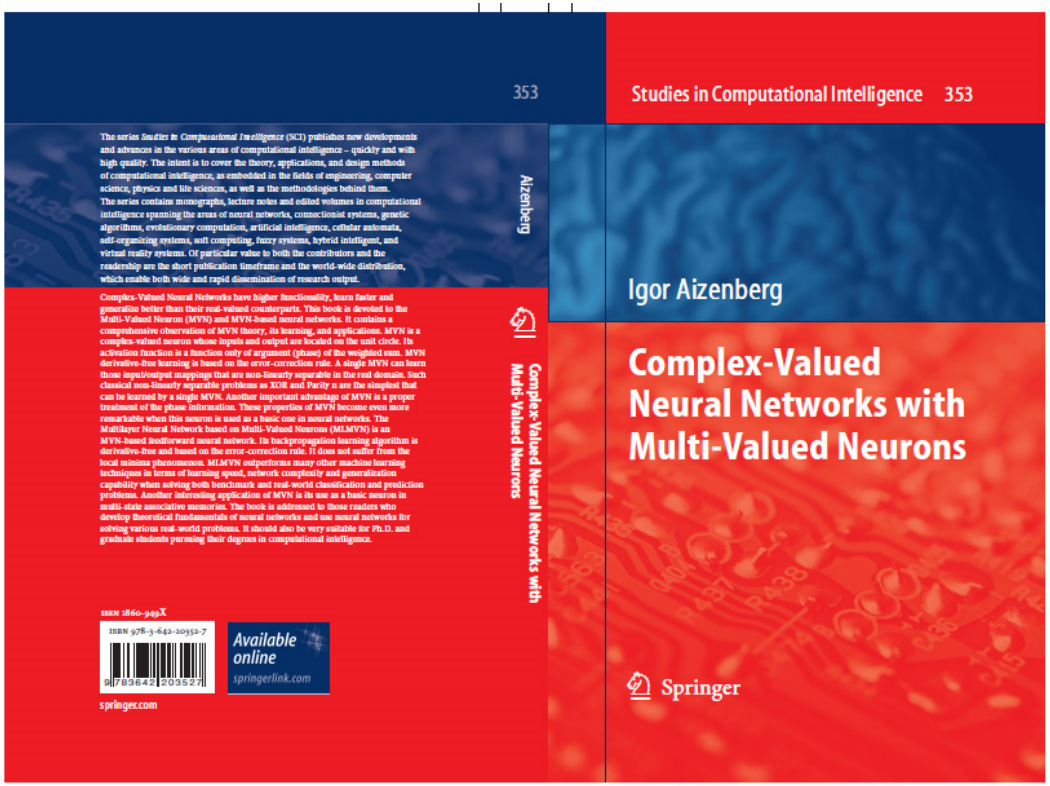
- Why we need complex-valued neural networks?
- The role of phase information
- Multiple-Valued Logic (k-valued Logic) over the Field of the Complex Numbers
- Multi-Valued Neuron (MVN)
- Error-Correction Learning Rule for MVN
- MVN with a periodic activation function and solving non-linearly separable problems
- MVN-based multilayer feedforward neural network (MLMVN) and its derivative-free backpropagation learning algorithm based on the error-correction rule
- MVN as a model of a biological neuron

Highlights-2 (Applications)

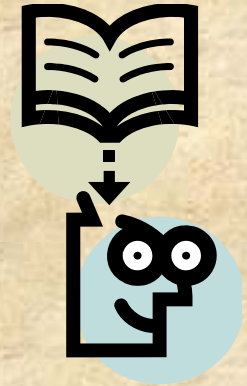


- Long-term time series prediction using MLMVN
- Identification of blur and its parameters in image deblurring using MLMVN
- Recognition of blurred images using MLMVN
- Decoding of signals in an EEG-based brain-computer interface using MLMVN
- MVN-based associative memories

Complex-Valued Neural Networks with Multi-Valued Neurons



- Was published in June 2011



Why we need Complex-Valued Neurons?

Why we need Complex-Valued Neurons?

- "Why is my verse so barren of new pride,

  So far from variation or quick change?

  Why with the time do I not glance aside

  To new-found methods and to compounds strange?"

  William Shakespeare, Sonnet 76

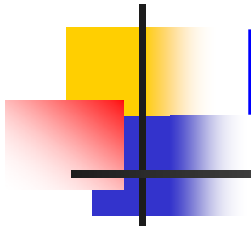
Why we need Complex-Valued Neurons?

- "All truth passes through three stages. First, it is ridiculed. Second, it is violently opposed. Third, it is accepted as being self-evident."

  Arthur Schopenhauer

Why we need Complex-Valued Neurons?

- Complex-valued neurons are as natural as complex numbers
- Complex-valued neurons are more functional than their real-valued counterparts
- Complex-valued neurons learn faster and generalize better
- Complex-valued neurons can be used for simulation of biological neurons
- Complex-valued neurons treat the phase information properly

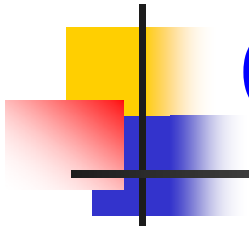


Motivation

- 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

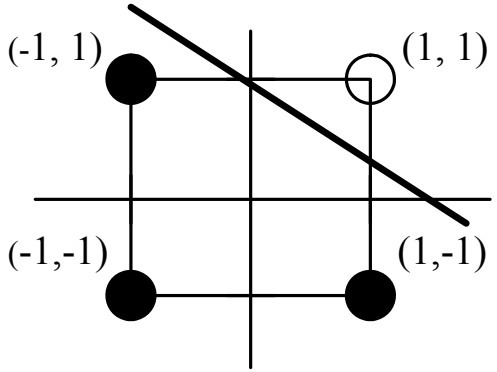
Limited Functionality of Classical Neurons



- A classical real-valued 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?

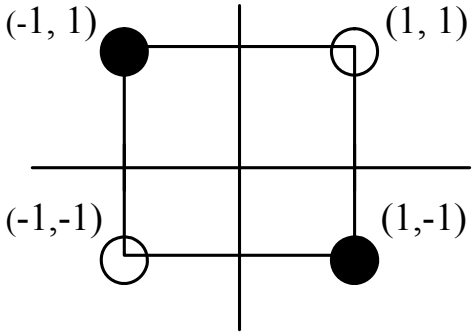
Linear Separability/Non-separability

"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

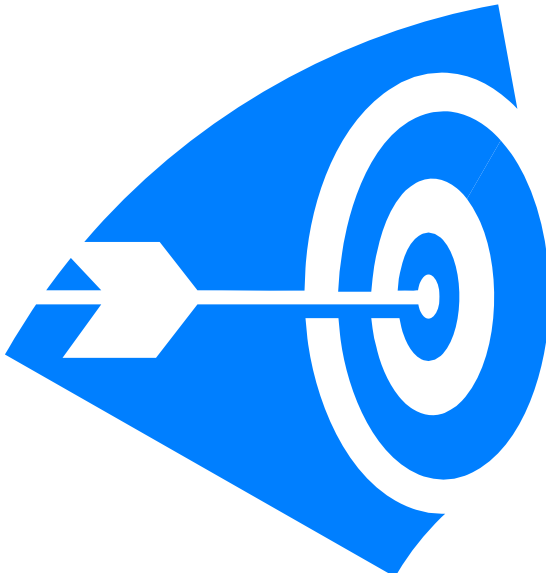
Is it possible to overcome

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.

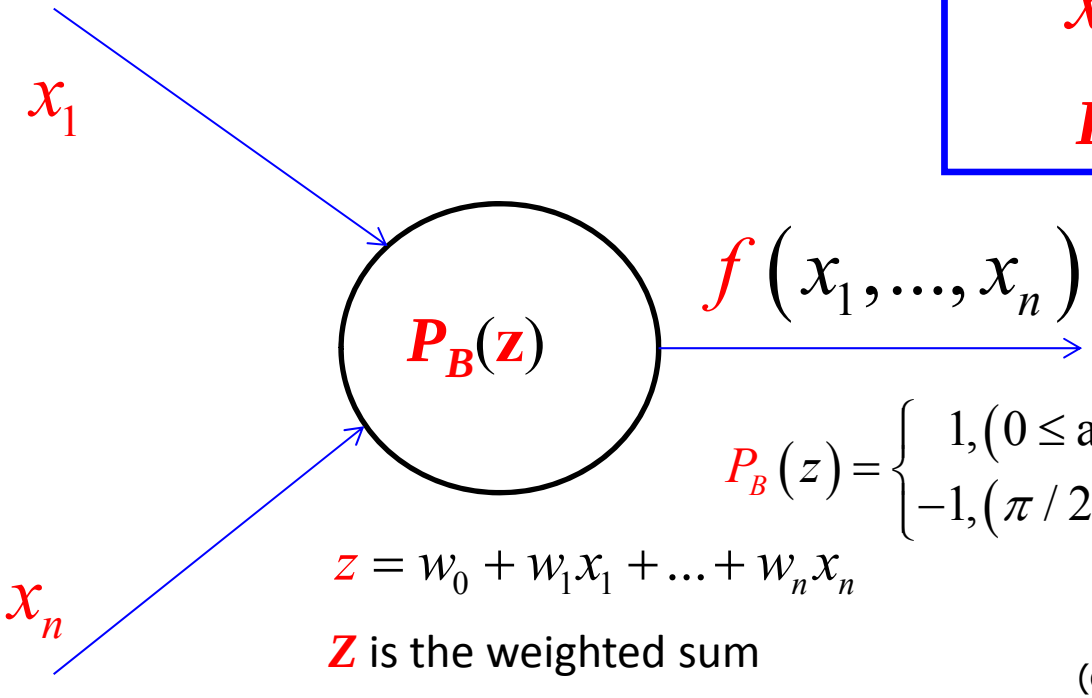
A complex-valued neuron and XOR problem

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

f is a function to be learned

$x_1,\ldots,x_n$ are the inputs

P_B is the activation function

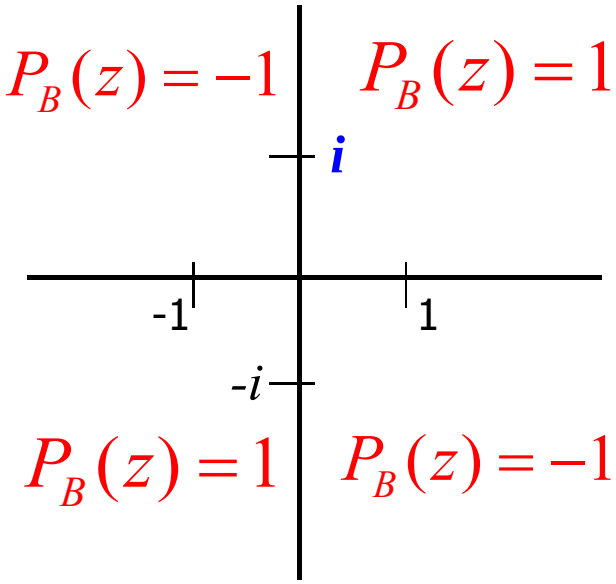


$$P_B(z) = \begin{cases} 1, (0 \le \arg z < \pi/2) \, or \, (\pi \le \arg z < 3\pi/2) \\ -1, (\pi/2 \le \arg z < \pi) \, or \, (3\pi/2 \le \arg z < 2\pi) \end{cases}$$

$z = w_0 + w_1x_1 + \ldots + w_nx_n$

z is the weighted sum

Learning of the XOR problem using a single neuron!



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

Let weights be complex and P_B be the activation function

$W=(0, 1, i)$ – the weighting vector

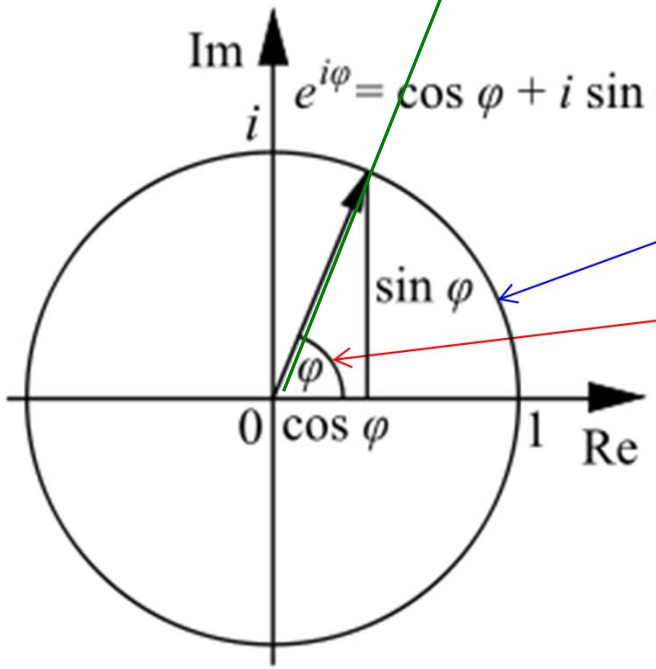
x_1	x_2	$z = = w_0 + w_1 x_1 + w_2 x_2$	$P_B(z)$	x_1 xor x_2
1	1	$1+i$	1	1
1	-1	$1-i$	-1	-1
-1	1	$-1+i$	-1	-1
-1	-1	$-1-i$	1	1

Phase-Dependent Activation Function

- 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

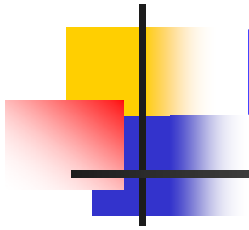
$z = w_0 + w_1 x_1 + \ldots + w_n x_n$ - **weighted sum of inputs**



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

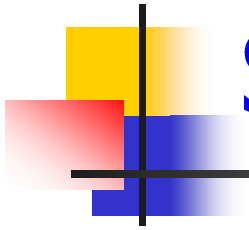
A unit circle

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



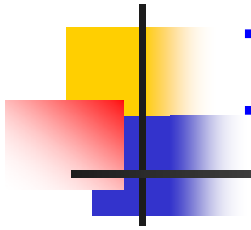
Higher Functionality

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



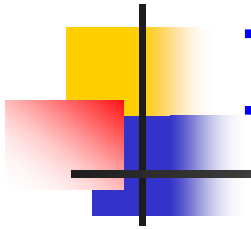
Simplicity of Learning

- 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.



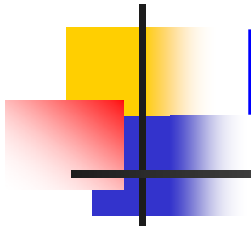
Importance of Phase

- From the today's point of view one of the main advantages of complex-valued neurons is their ability to work with the **phase**, which is very important for the analysis of signals and for solving different pattern recognition and classification problems.



Importance of Phase

- Even in the analysis of real-valued signals one of the most efficient approaches is the frequency domain analysis, which immediately involves complex numbers.
- In fact, analyzing signal properties in the frequency domain, we see that each signal is characterized by magnitude and **phase** that carry different information about the signal



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.

Image Recognition: Features Selection

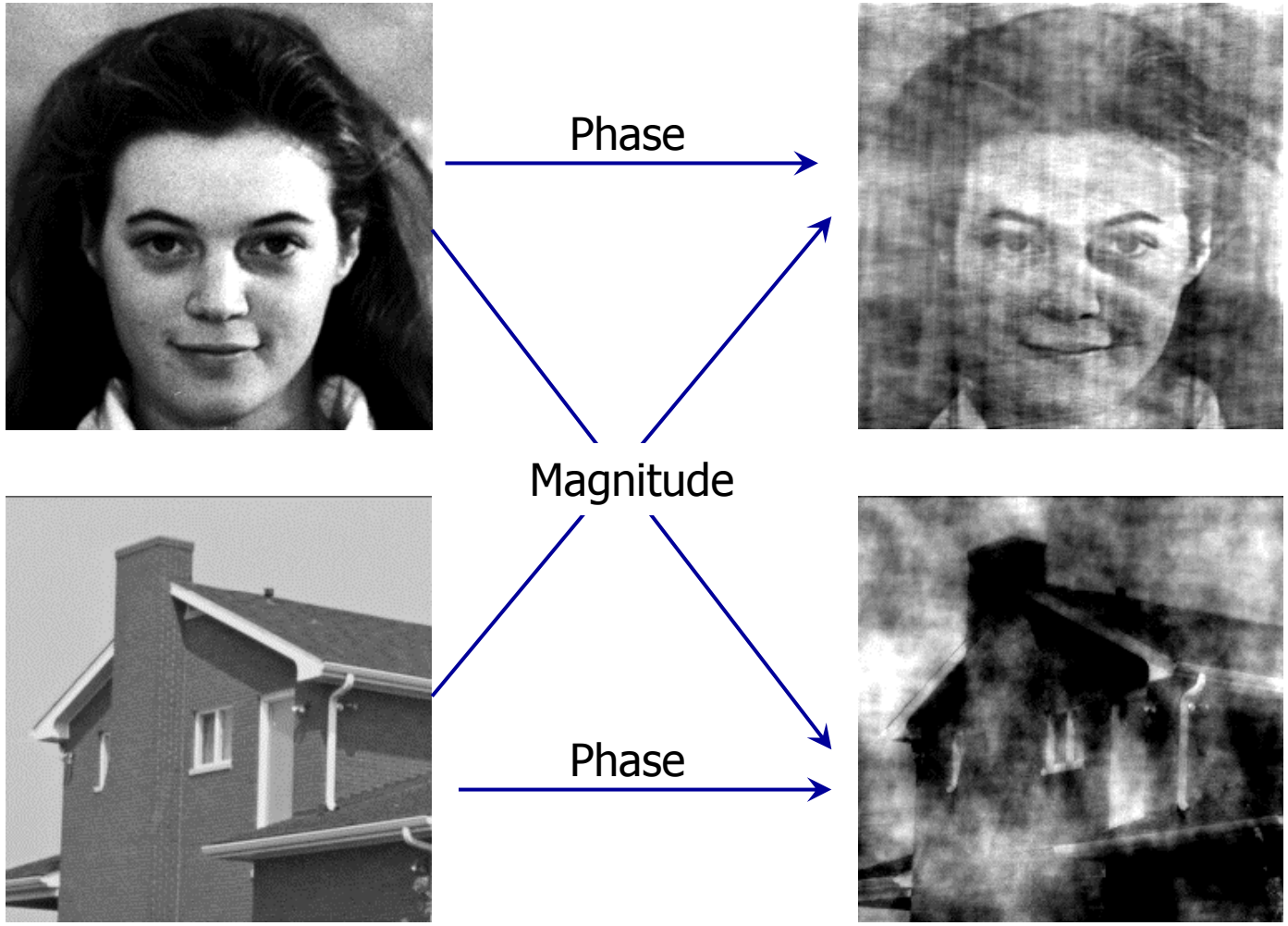
- Thus the Fourier Phase Spectrum can be a very good source of the objective features that describe all objects represented by the corresponding signals.
- The Power Spectrum (magnitude) describes some global signal properties (for, example blur, noise, cleanness, contrast, brightness, etc. for images).

Importance of Phase

- Phase contains the information about all the edges, shapes, and their orientation

This image results from the inverse Fourier transform applied to the spectrum synthesized from the original phases, but with magnitudes replaced by the constant "1"

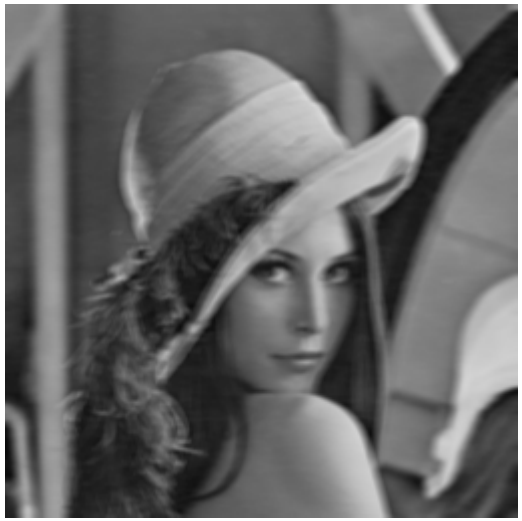
Importance of Phase



Phase and Magnitude

Magnitude contains the information about the signal's properties

(a)



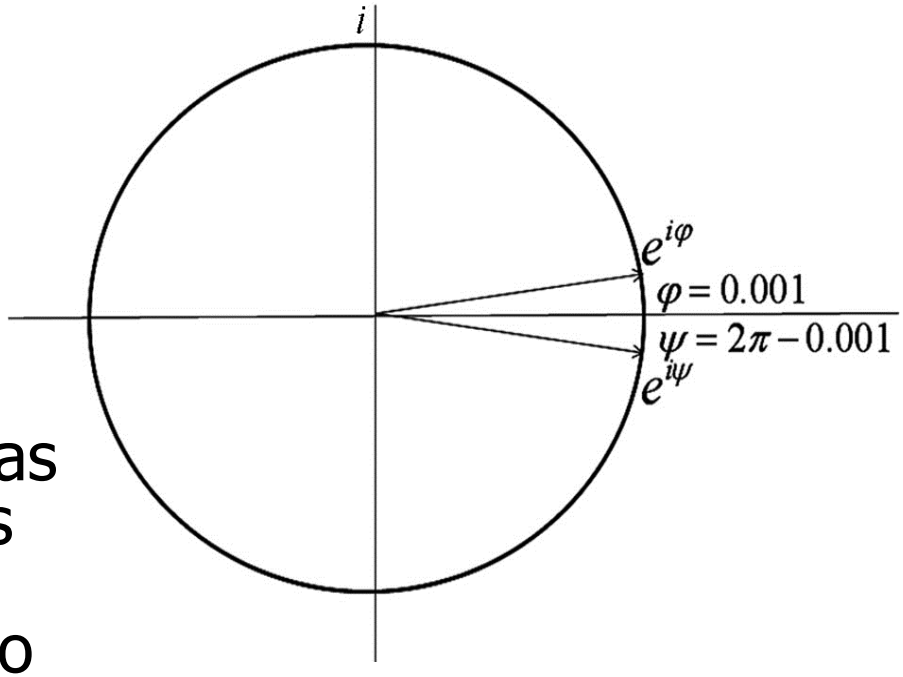
(b)

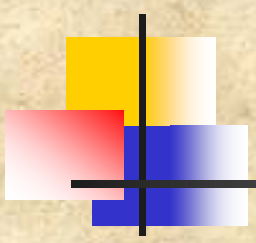
Phase (a) & **Magnitude** (b)

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

Proper treatment of phase

- Complex-valued neurons treat the phase information properly
- If phases are treated as some abstract real numbers, we may treat phases such as $\varphi = 0.001$ and $\psi = 2\pi - 0.001 = 6.282$ as much different values. This would be incorrect because in fact they are very close to each other



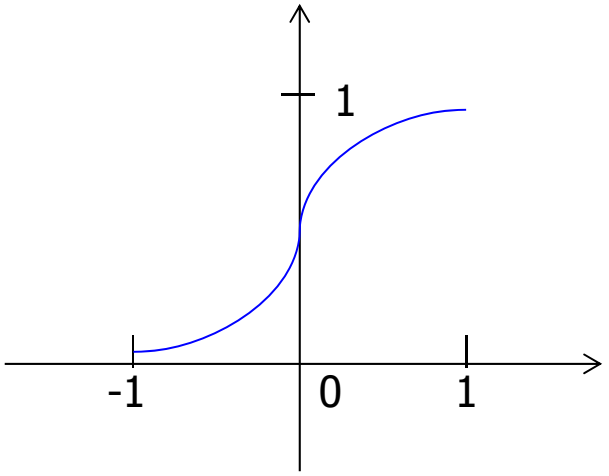


Multiple-Valued Threshold Logic over the field of Complex numbers and a Multi-Valued Neuron

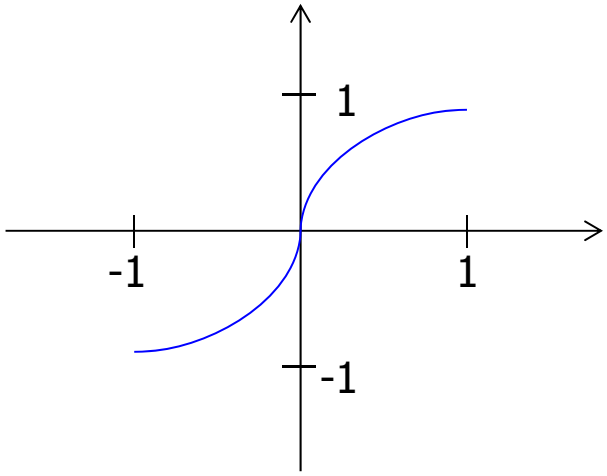
Traditional approach to learn multiple-valued and continuous mappings using a neuron:

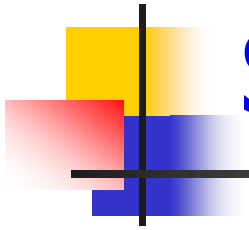
- Sigmoid activation function (the most popular):

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



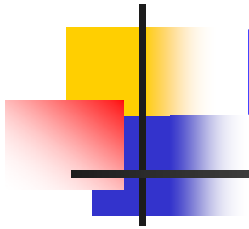
$$F(z) = \tanh z$$





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 a neuron with another activation function?



Multi-Valued Neuron (MVN)

- A Multi-Valued Neuron is a neural element with n inputs and one output located on the unit circle, and with the complex-valued weights.
- The theoretical background behind the MVN is the Multiple-Valued (k-valued) Threshold Logic over the field of complex numbers

Motivation

Classical Boolean Threshold Logic

A neuron with the threshold activation function is able to learn input/output mappings described by Boolean threshold (linearly separable) functions

Multiple-Valued Logic

A neuron with a multi-valued activation function?

Multi-Valued Mappings and Multiple-Valued Logic

- We traditionally use Boolean functions and Boolean (two-valued) logic, to represent two-valued i/o mappings:

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

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

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

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

Importance of a normalized structural alphabet

- A classical Boolean alphabet {0, 1} is not normalized
- The alphabet {1, -1} is normalized
- This is especially important for learning algorithms!

Multiple-Valued Logic: Classical View

- The values of multiple-valued (***k***-valued) logic are usually encoded by the integers from the set $K = \{0, 1, \ldots, k\text{-}1\}$
- On the one hand, this approach looks natural
- On the other hand, values of multiple-valued logic are not normalized in K

Multiple-Valued Logic: A problem with non-normalized values

- 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 Logic Over the Field of Complex Numbers: Basic Idea

- To obtain a normalized alphabet of multiple-valued (***k***-valued) logic (like the normalized alphabet $E_2 = \{1, -1\}$ of Boolean logic), we can move to the complex domain

Multiple-Valued Logic Over the Field of Complex Numbers: Basic Idea

- $\varepsilon = e^{i2\pi/k}$, where i is an imaginary unity, is a primitive k^{th} root of unity

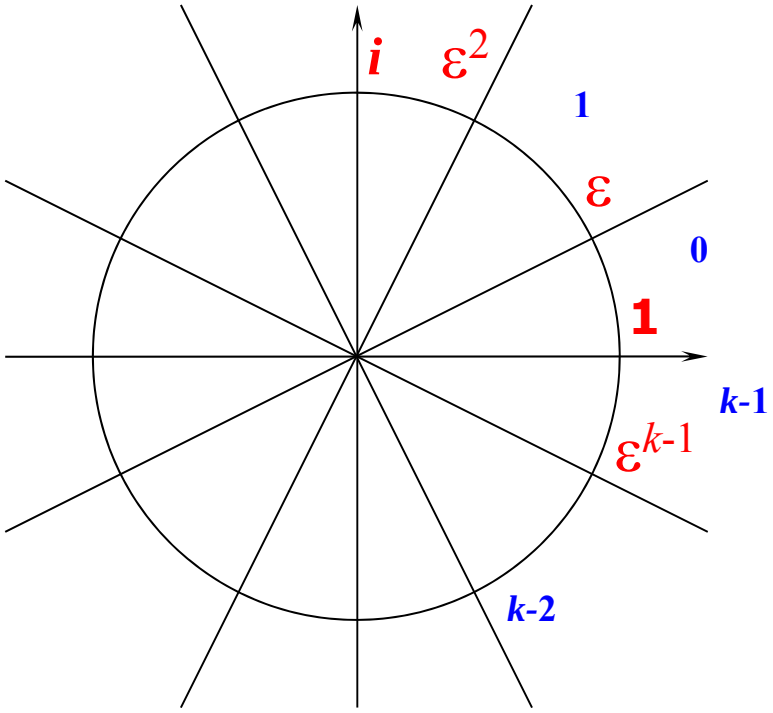
- Let $A_k = E_k = \left\{\varepsilon^0, \varepsilon, \varepsilon^2, \ldots, \varepsilon^{k-1}\right\}$ (a set of all the k^{th} root of unity) be our structural alphabet

Multiple-Valued (k-valued) Logic Over the Field of Complex Numbers

- Important conclusion: Any function of n variables $f(x_1,...,x_n): E_k^n \to E_k$ is a function of k-valued logic over the field of complex numbers according to the basic definition

Multiple-Valued (k-valued) Logic Over the Field of Complex Numbers

(introduced by Naum Aizenberg in **1971**)



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

regular values of k-valued logic

$$j \rightarrow \varepsilon^{j} = e^{i2\pi j/k}$$

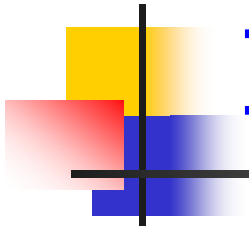
one-to-one correspondence

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

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

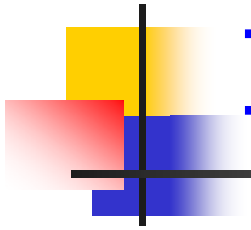
$$\varepsilon = e^{i2\pi/k}$$

primitive k^{th} root of unity



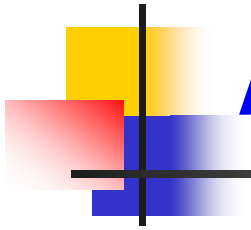
Important Advantage

- 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 are encoded according to this approach by the elements of the set $E_2 = \{1, -1\}$



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 a physical nature of the colors.



A neuron

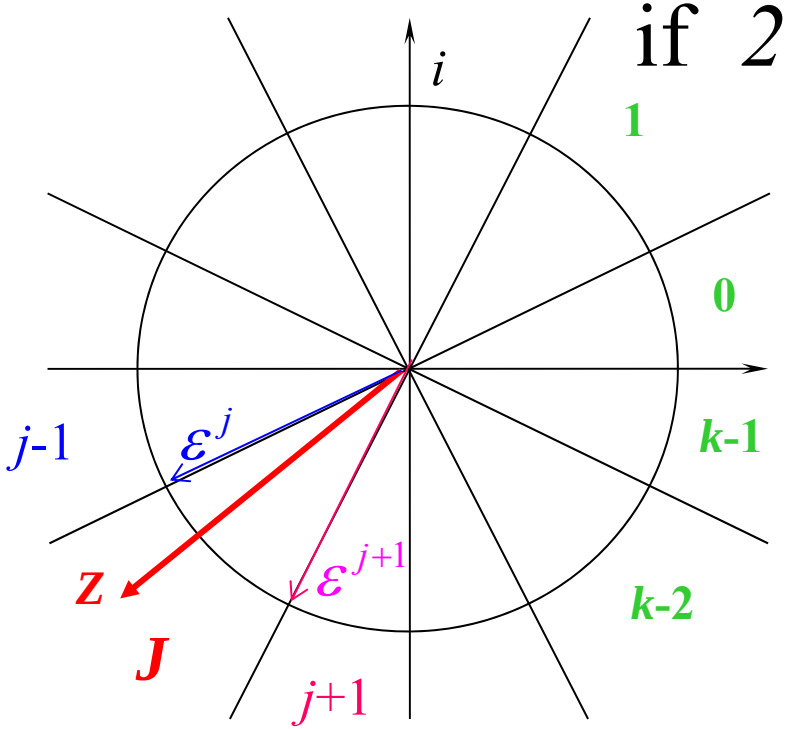
- Can we consider a neuron whose input/output mapping is described by a multiple-valued function over the field of complex numbers?
- To consider such a neuron, we have to introduce its activation function

Discrete (k-valued) Activation Function

(introduced in **1971** by **Naum Aizenberg**; the first historically known complex-valued activation function)

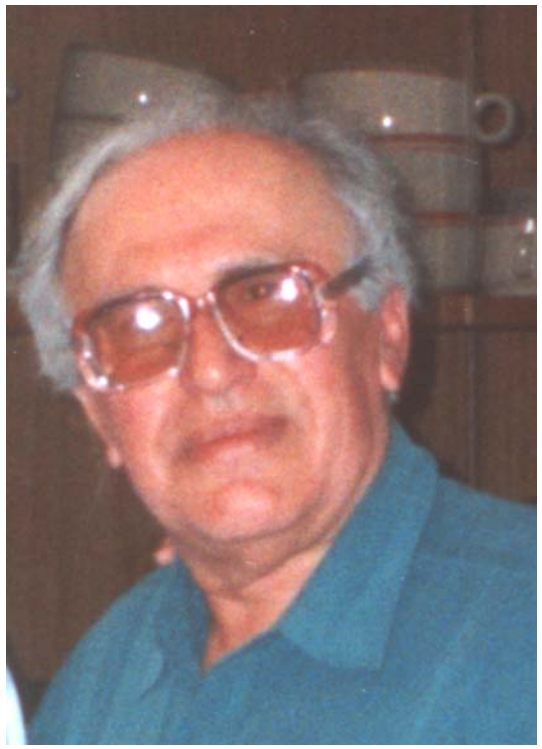
$$P(z) = \exp\left(i 2\pi j / k\right) = \varepsilon^j,$$

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



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

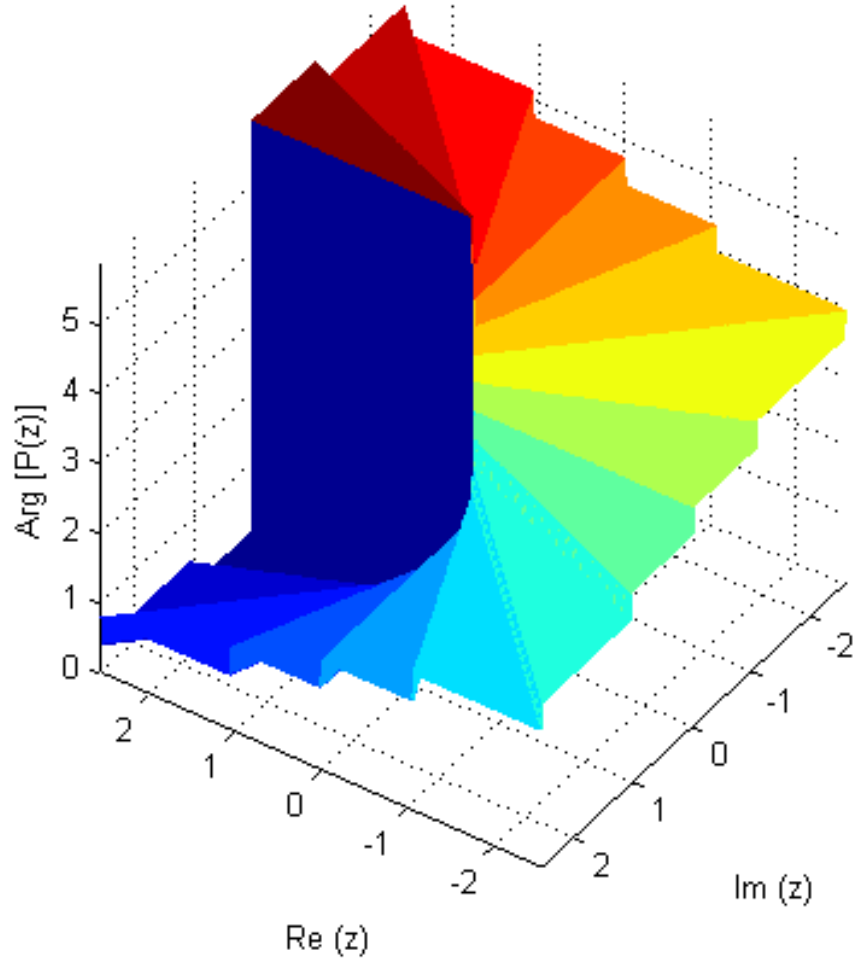
Naum Aizenberg



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

Discrete (k-valued) Activation Function

k=16



Multiple-Valued (k-Valued) Threshold Function

The k-valued function $f(x_1,...,x_n): T \to E_k; T \subseteq E_k^n \vee T \subseteq O^n$ 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:

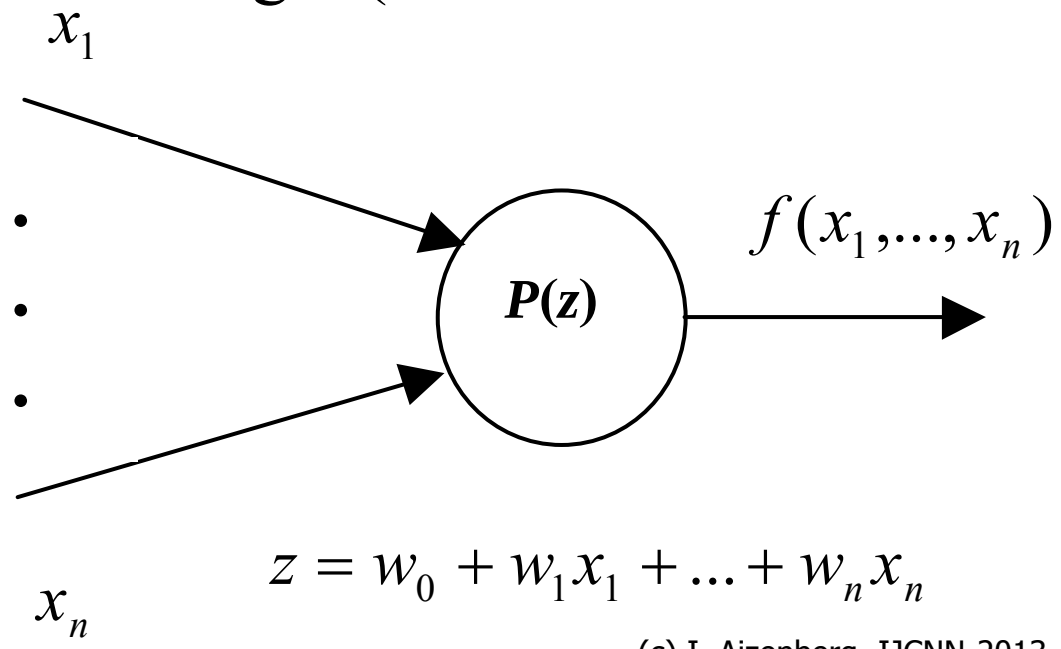
$$f(x_1,..., x_n) = P(w_0 + w_1x_1 + ... + w_nx_n)$$

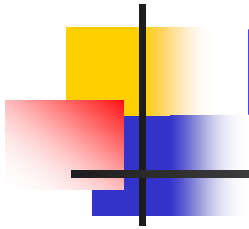
Discrete Multi-Valued Neuron (MVN)

(introduced first by Naum Aizenberg in **1973** as an element of multiple-valued threshold logic; re-introduced by Naum Aizenberg and Igor Aizenberg as a multi-valued neuron in **1992**)

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

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



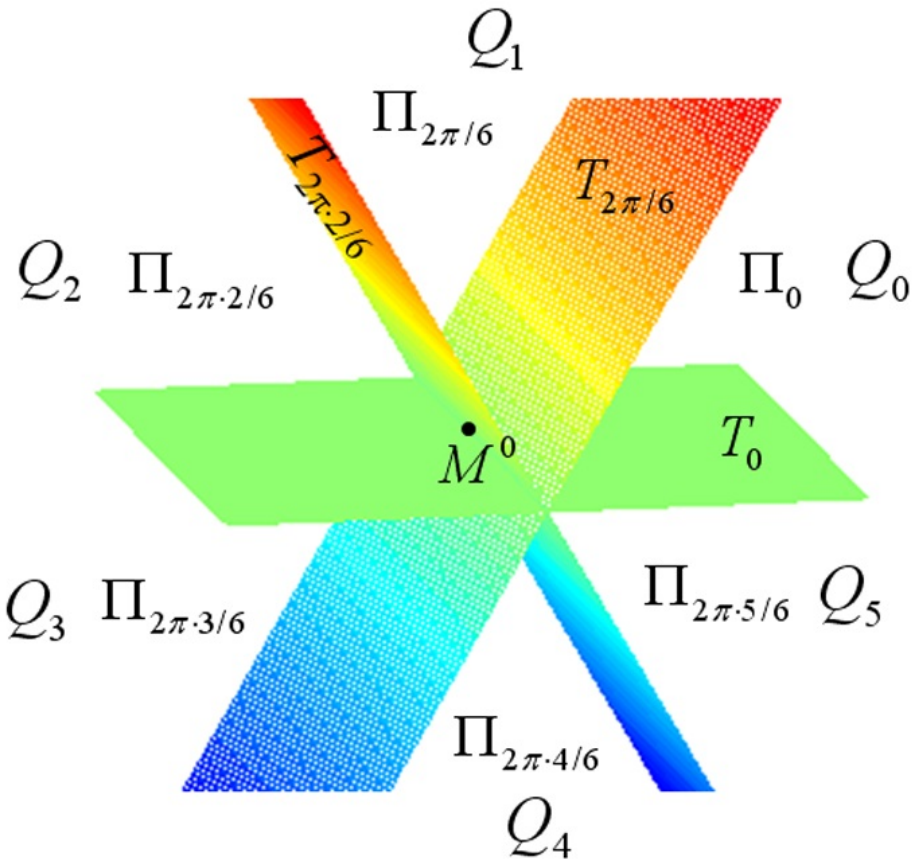


MVN: Key Properties

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

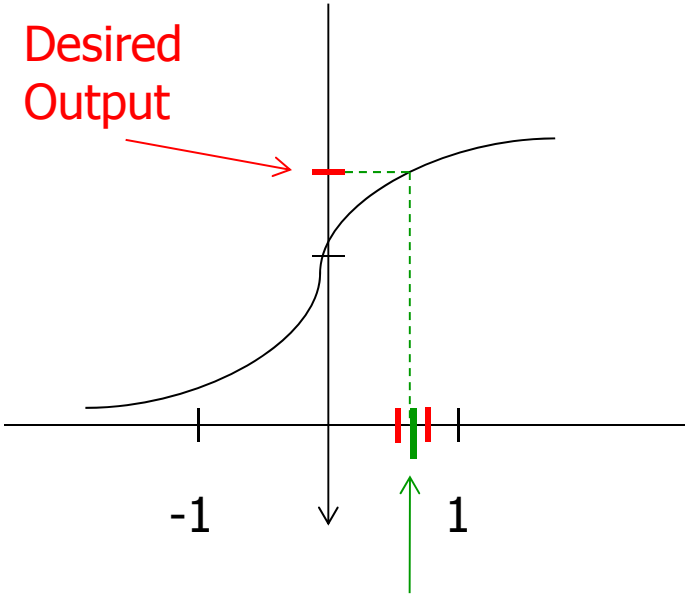
MVN divides the n-dimensional space into k subspaces by k-edge

- Division of 3-dimensional space by the 6-edge implemented by the 6-valued MVN activation function P



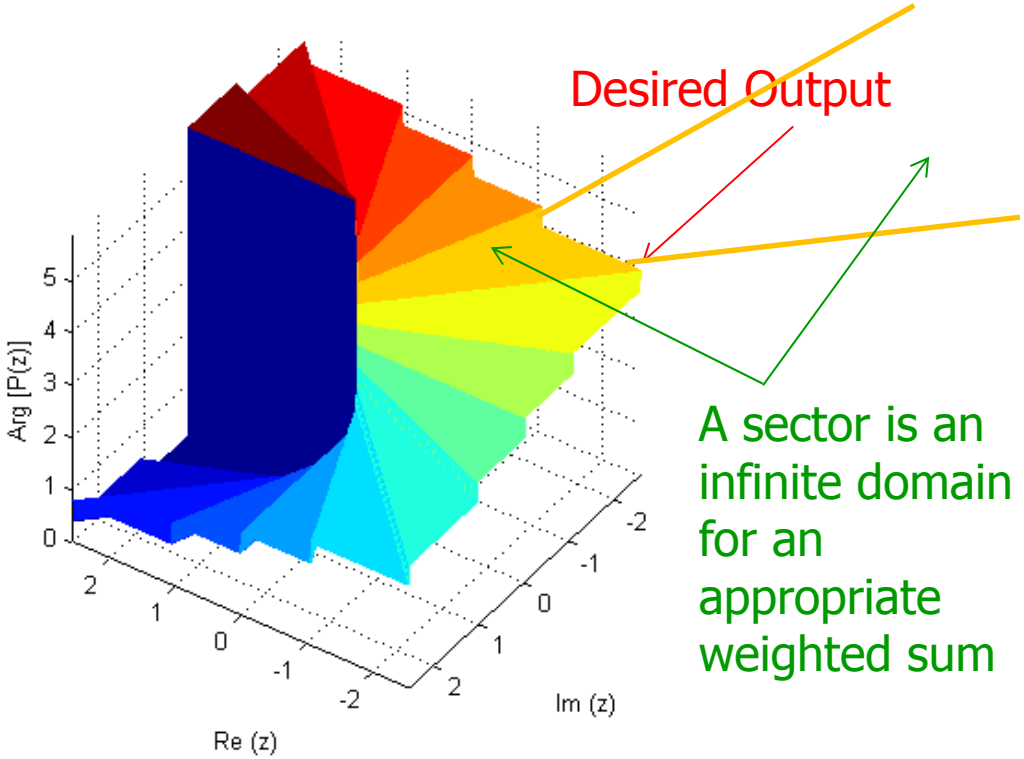
Higher functionality of MVN

SIGMOID ACTIVATION FUNCTION:



A very limited acceptable interval for a weighted sum

MVN's MULTI-VALUED ACTIVATION FUNCTION:

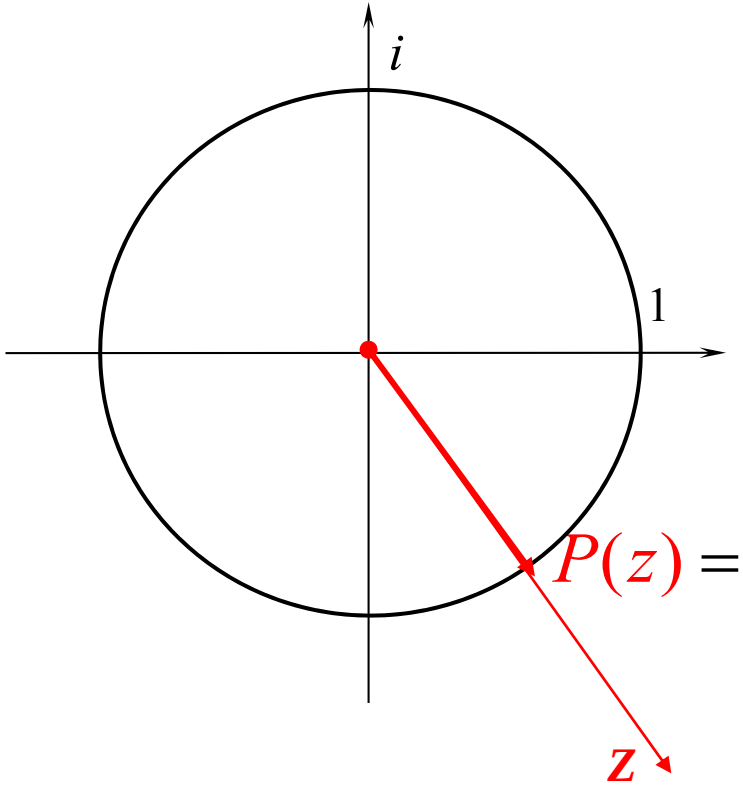


A sector is an infinite domain for an appropriate weighted sum

- 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 Activation Function

(introduced by I. Aizenberg and C. Moraga in **2007**)



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

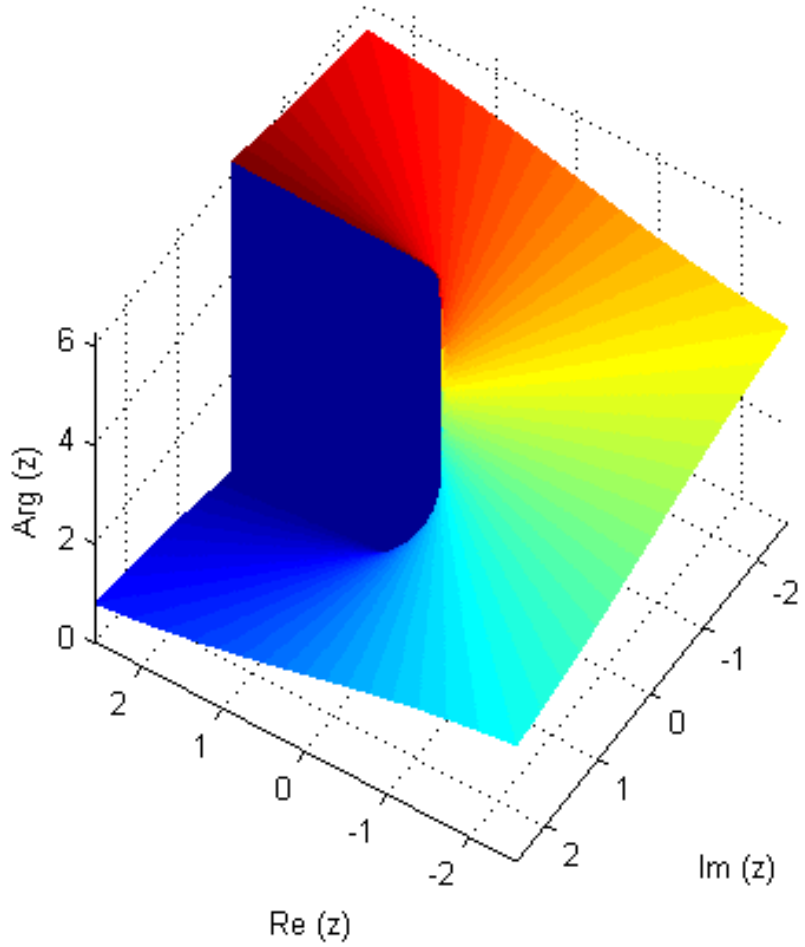
$$P(z) = e^{i\,\mathrm{Arg}\,z} = z/\,|\,z\,|$$

Function P maps the complex plane into the unit circle

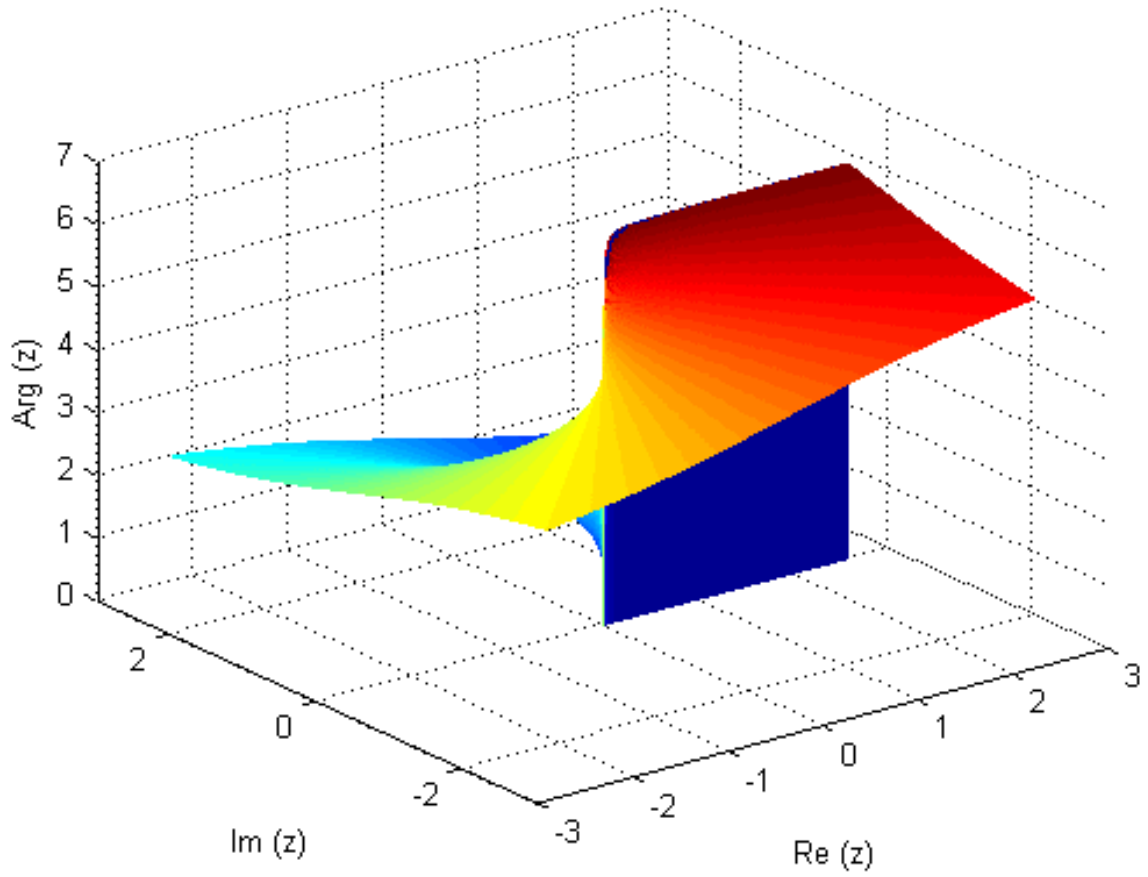
Continuous MVN

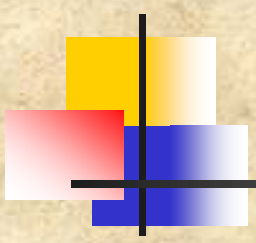
- MVN with the continuous activation function $P(z) = e^{i\mathrm{Arg}\, z} = z/|z|$ implements an input/output mapping $T^n \to O; T \subseteq E_k \vee T \subseteq O$
- where O is the set of points located on the unit circle

Continuous Activation Function



Continuous Activation Function





MVN: Error-Correction Learning Rule

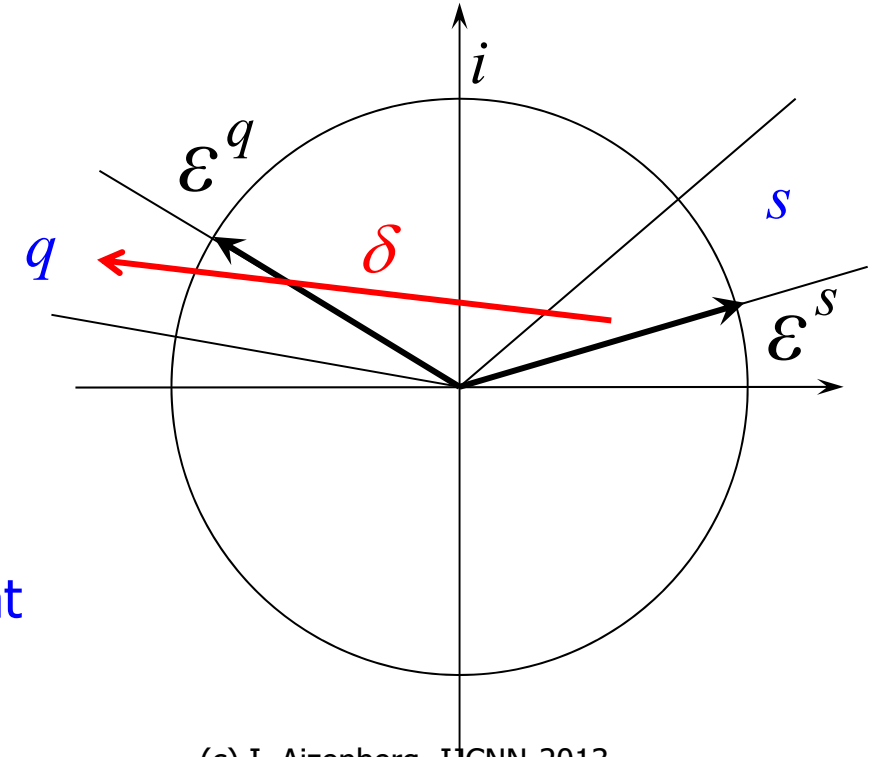
MVN Error-Correction Learning

- MVN 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 MVN

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

$$W_{r+1} = W_r + \frac{\alpha_r}{(n+1)\,|\,z_r\,|}\,\delta\,\overline{X}$$

W – weighting vector; X - input vector

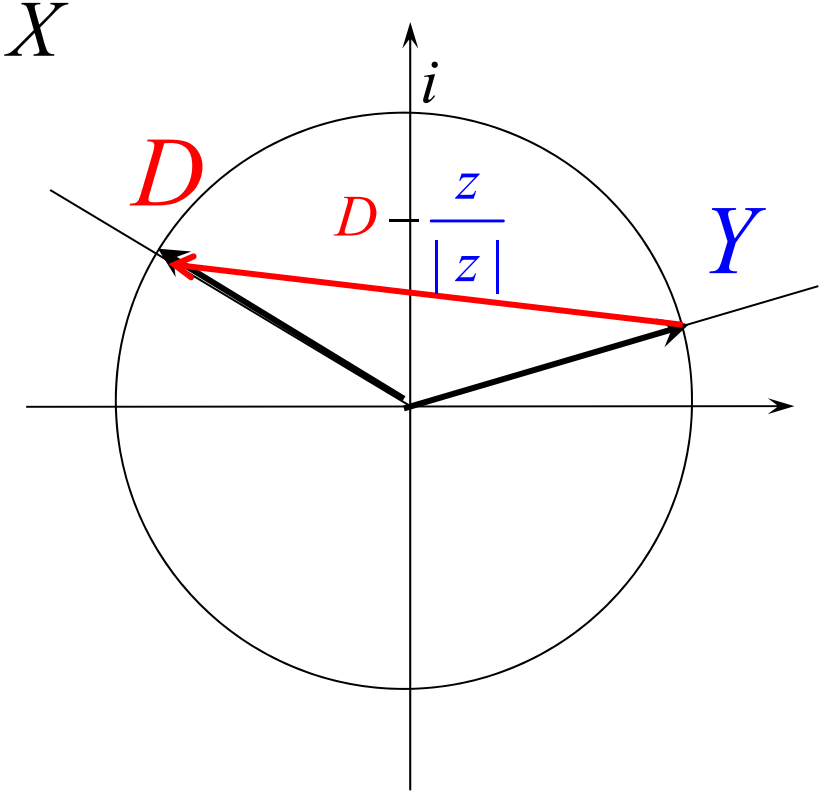
$\overline{X}$ is a complex conjugated to X

α_r – a learning rate (in general, it is complex)

r - current learning step;

r+1 – the next learning step

Z – the weighted sum



D is a desired output

Y is an actual output

Error-Correction Learning Rule for MVN: Error Calculation

- The error in the MVN error-correction rules can be calculated in several ways:

$$\delta = D - Y$$ - most natural and the only way for the continuous MVN

Modifications for the discrete MVN:

$$\delta = D - \frac{z}{|z|}$$ - the difference between the desired output and projection of the weighted sum on the unit circle

$$\delta = D' - \frac{z}{|z|}$$ - the difference between the desired output shifted to the bisector of the desired sector and the projection of the weighted sum on the unit circle

The Error Sharing Principle

- The MVN error must be shared among all the weights of the neuron because we do not know in advance, which of them contribute more to the error
- This sharing is utilized through the division of the error by the number of the weights $(n+1)$

Error sharing using the factor 1/(n+1) in the Learning Rule

$$\delta = \varepsilon^q - \frac{z}{|z|}$$ - 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:

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

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

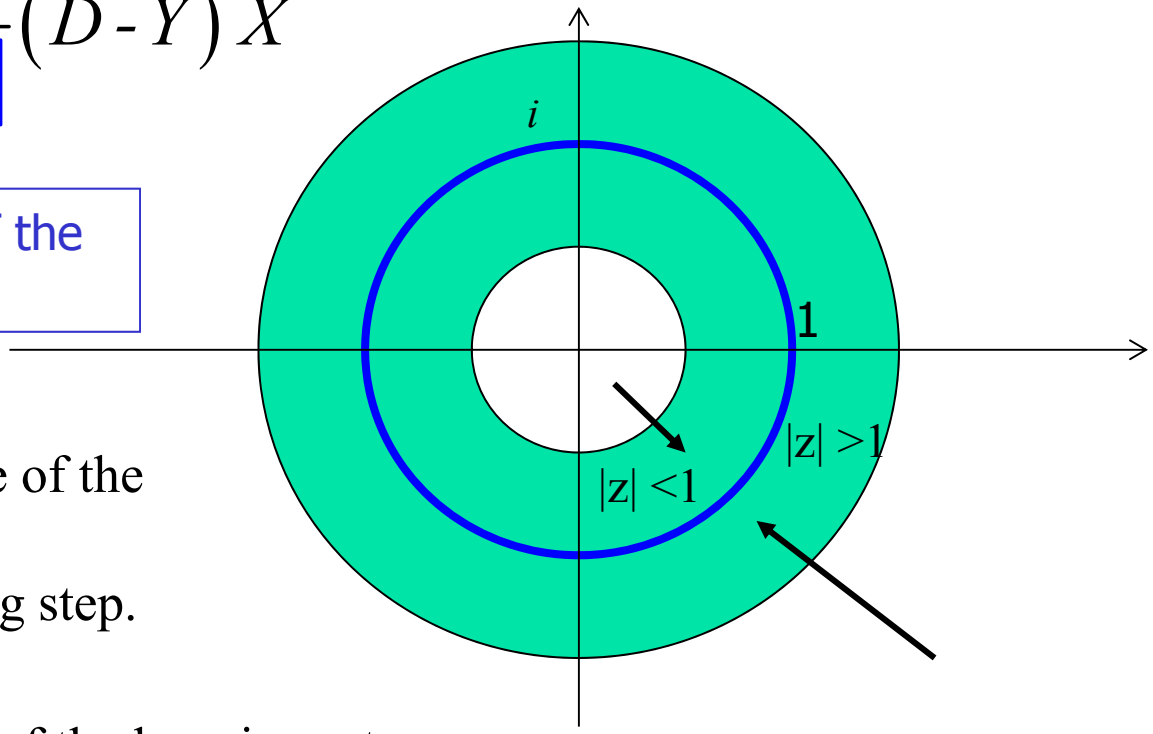
$$= w_0 + \frac{\delta}{(n+1)} + w_1 x_1 + \frac{\delta}{(n+1)} + \ldots + w_n x_n + \frac{\delta}{(n+1)} =$$

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

Self-Adaptation of the Learning Rate

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

$1/|z_r|$ is a self-adaptive part of the learning rate



$|z_r|$ -is the absolute value of the weighted sum on the previous (r^{th}) learning step.

$\frac{1}{|z_r|}$ is a self-adaptive part of the learning rate

Modified Learning Rules with the Self-Adaptive Learning Rate

$$W_{r+1} = W_r + \frac{\alpha_r}{(n+1)\,|\,z_r\,|}(\varepsilon^q - \varepsilon^s)\,\bar{X}$$

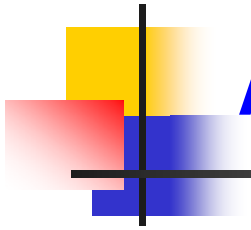
Discrete MVN

$1/|z_r|$ is a self-adaptive part of the learning rate

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

Continuous MVN

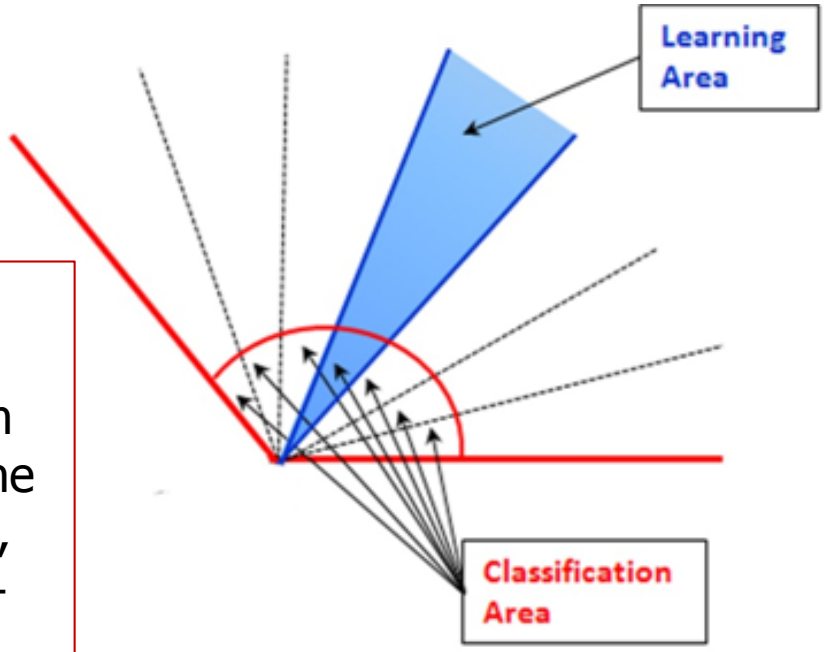
Convergence of the Learning Algorithm



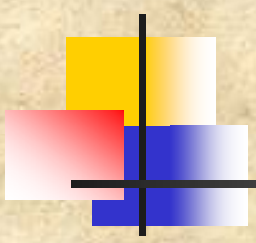
- It is proven that the MVN learning algorithm requires a finite number of learning iterations to converge.
- Since the MVN learning is based on the error-correction rule, it does not suffer from the local minima phenomenon.
- Learning algorithm for the continuous MVN with the accuracy λ is reduced to the one of the discrete MVN in π/λ –valued logic.

Hard and Soft Margins for Solving Classification Problems

- Classification accuracy can be significantly improved if margins of the pre-determined size separate learning samples from the sectors borders



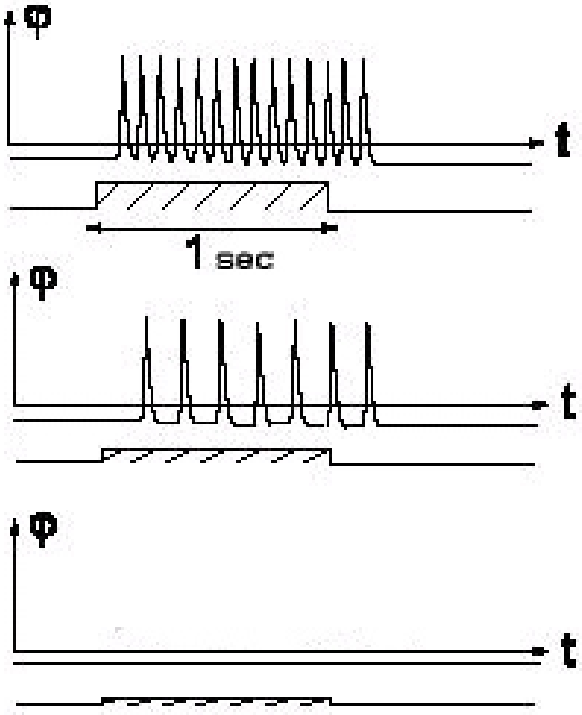
- Learning with **Soft margins** will be considered in detail in the presentation at the CVNN special session,
- Tuesday, August 6, 5-20pm



MVN as a Model of a Biological Neuron

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

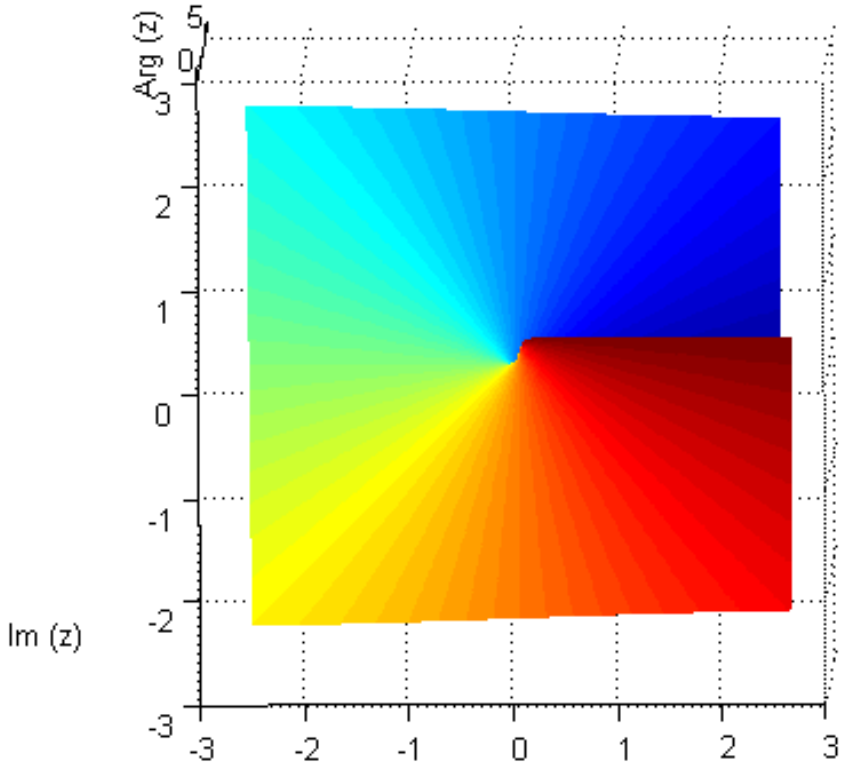
Intermediate State → Medium frequency

No impulses → Inhibition → Zero frequency

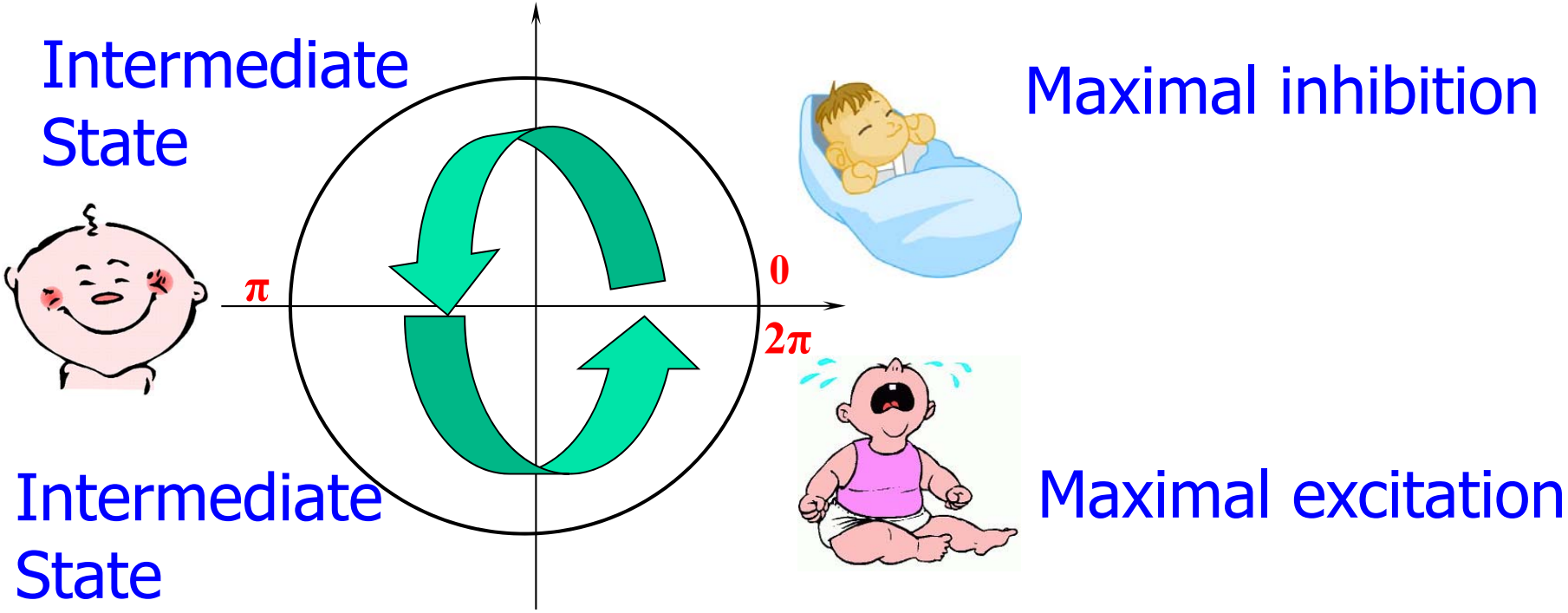
MVN as a model of a biological neuron

- 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

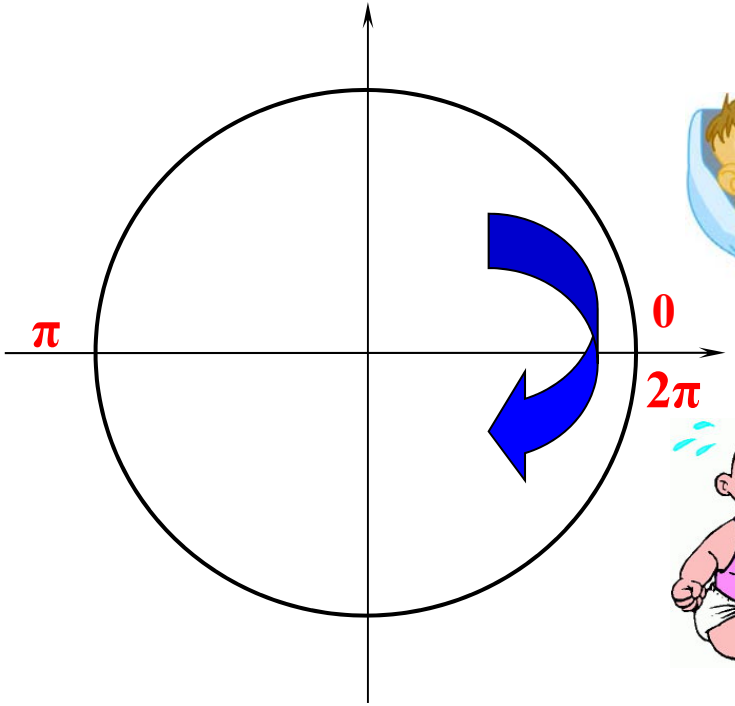
MVN as a Model of a Biological Neuron



MVN as a model of a biological neuron

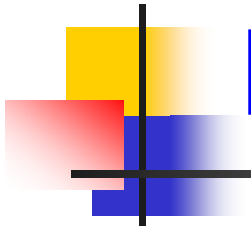


MVN as a model of a biological neuron



Maximal inhibition

Maximal excitation



MVN:

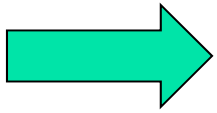
- Learns faster
- Adapts better
- Learns even highly nonlinear functions
- Opens new very promising opportunities for the network design
- Is much closer to the biological neuron
- Allows for the use of the Fourier Phase Spectrum as a feature space for solving different recognition/classification problems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

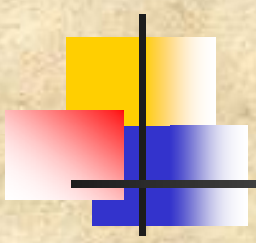
Fundamental Questions

- **MVN can learn *k*-valued threshold functions.**
- **What about non-threshold multiple-valued functions? Is it absolutely necessary to use a neural network to learn them?**

Fundamental Questions

- **Is it possible to increase the functionality of a single multi-valued neuron?**
- **Is it possible to project non-threshold *k*-valued functions into *m*-valued threshold functions (*m*>*k*)?**

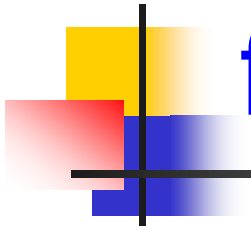
Yes !!! 



MVN with a Periodic Activation Function and Learning of Non-Linearly Separable Input/Output Mappings

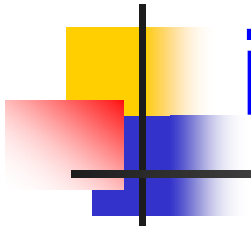
Introduced by I. Aizenberg in **2009-2010**

MVN with a periodic activation function



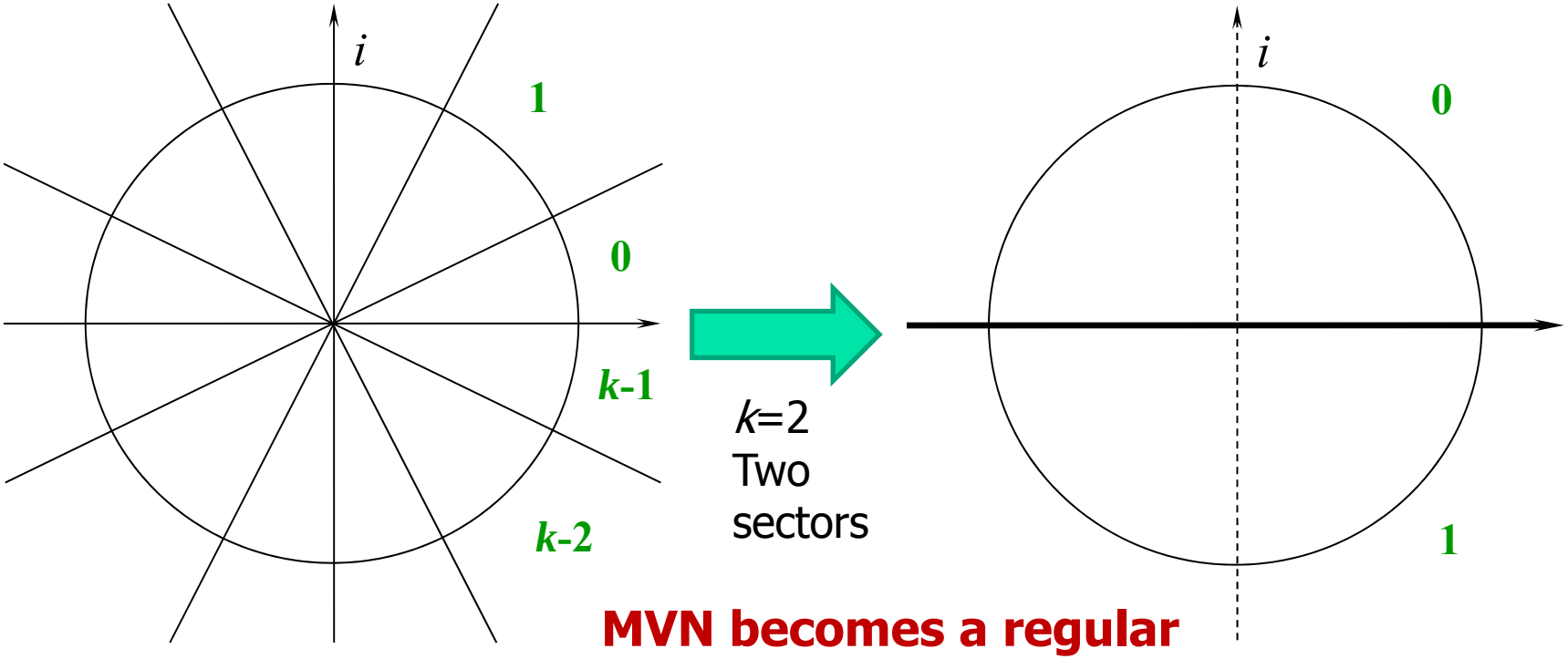
- I. Aizenberg, "A Multi-Valued Neuron with a Periodic Activation Function", *Proceedings of the International Joint Conference on Computational Intelligence*, Funchal-Madeira, Portugal, October 5-7, **2009**, pp. 347-354.
- I. Aizenberg, "A Periodic Activation Function and a Modified Learning Algorithm for a Multi-Valued Neuron", ***IEEE Transactions on Neural Networks***, vol. 21, No 12, December **2010**, pp. 1939-1949.
- I. Aizenberg, M. Caudill, J. Jackson, and S. Alexander, "Learning Nonlinearly Separable mod k Addition Problem Using a Single Multi-Valued Neuron With a Periodic Activation Function", *Proceedings of the 2010 IEEE International Joint Conference on Neural Networks*, Barcelona, Spain, July 18-23, **2010**, pp. 2577-2584.
- I. Aizenberg, "*Complex-Valued Neural Networks with Multi-Valued Neurons*", Springer, Heidelberg, **2011**.

Learning of non-threshold input/output mappings



- MVN may learn input/output mappings described by multiple-valued threshold functions over the field of complex numbers
- What about non-threshold (non-linearly separable) functions?

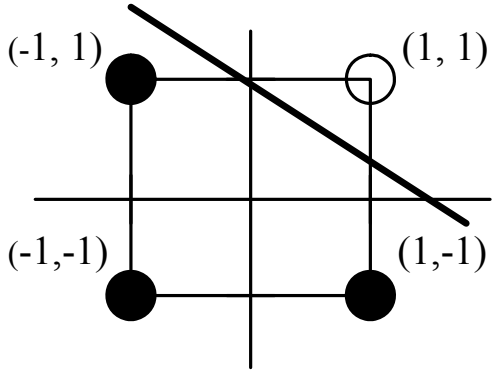
What happens to the MVN and its activation function when k=2 ?



MVN becomes a regular threshold neuron with its low functionality

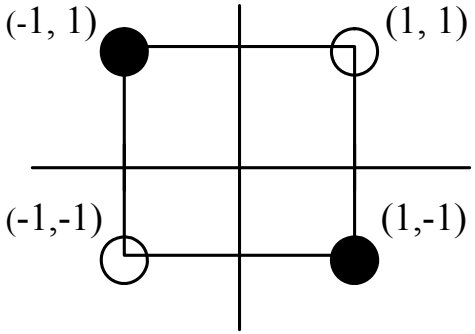
Linear Separability/Non-separability

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



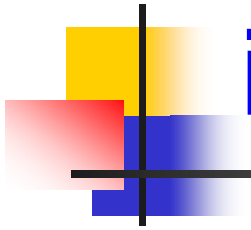
- 1 1→ 1
- 1 -1→ -1
- -1 1→ -1
- -1 -1→ -1

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



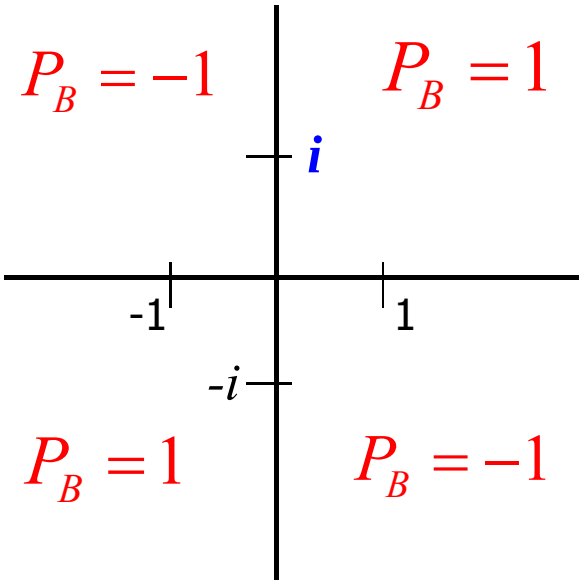
- 1 1 → 1
- 1 -1→ -1
- -1 1→ -1
- -1 -1→ 1

Learning of non-threshold input/output mappings



- To learn non-threshold functions of k-valued logic$\left(k \geq 2\right)$ using a single MVN, it is necessary to consider a periodic activation function, which projects k-valued logic into m-valued logic $(k<m)$

Reminder: the XOR problem



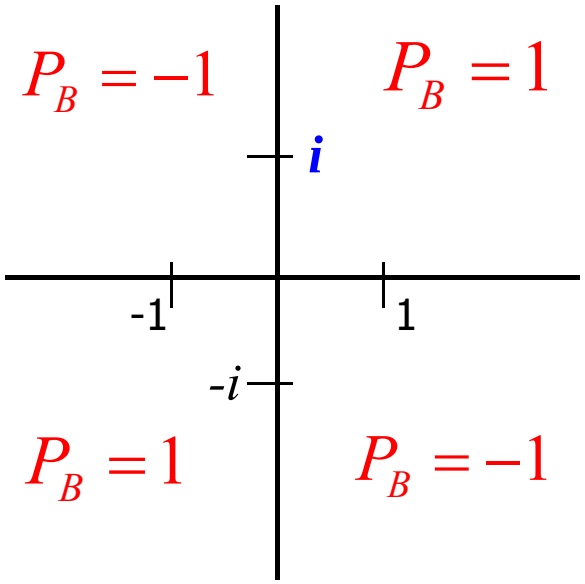
The activation function P_B is a **periodic function**! It separates the complex plane into 4 equal sectors

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 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

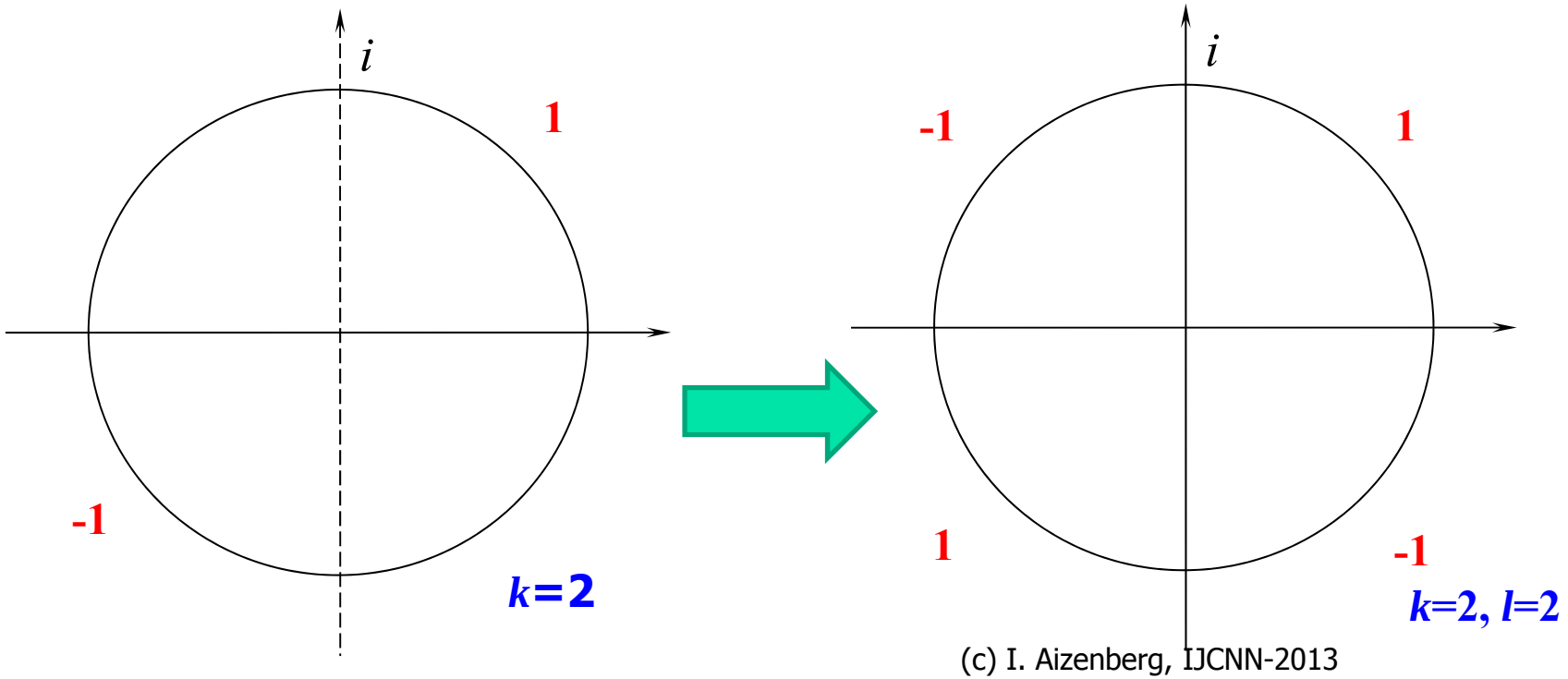
Periodicity of the Activation Function



- The activation function P_B is periodic. It divides the complex plane into 4=2x2 sectors and takes its values as alternating sequence 1, -1, 1, -1

MVN-P: MVN with a Periodic Activation Function

- Let us introduce a periodic activation function where the values of ***k***-valued logic are repeated periodically with a periodicity coefficient ***l***:

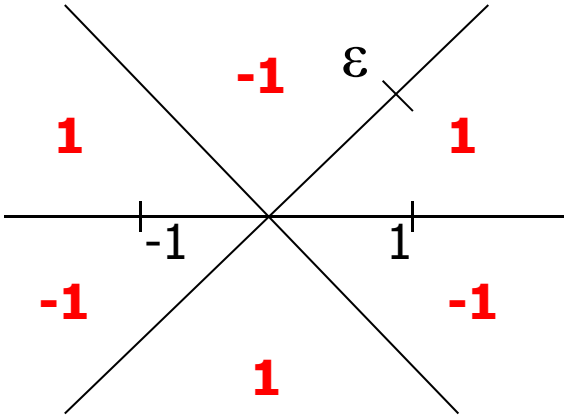


Parity 3 Problem (mod 2 addition of 3 Boolean inputs)

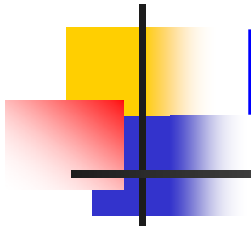
$(x_1 + x_2 + x_3) \bmod 2$

n=3, k=2, l=3, m=6 : 6 sectors

W=(0, ε, 1, 1) – the weighting vector



X_1	X_2	X_3	$Z = w_0 + w_1 x_1 + + w_2 x_2 + w_3 x_3$	$P_B(Z)$	$f(x_1, x_2, x_3)$
1	1	1	$\varepsilon + 2$	1	1
1	1	-1	ε	-1	-1
1	-1	1	ε	-1	-1
1	-1	-1	$\varepsilon - 2$	1	1
-1	1	1	$-\varepsilon + 2$	-1	-1
-1	1	-1	$-\varepsilon$	1	1
-1	-1	1	$-\varepsilon$	1	1
-1	-1	-1	$-\varepsilon - 2$	-1	-1



Parity N Problem

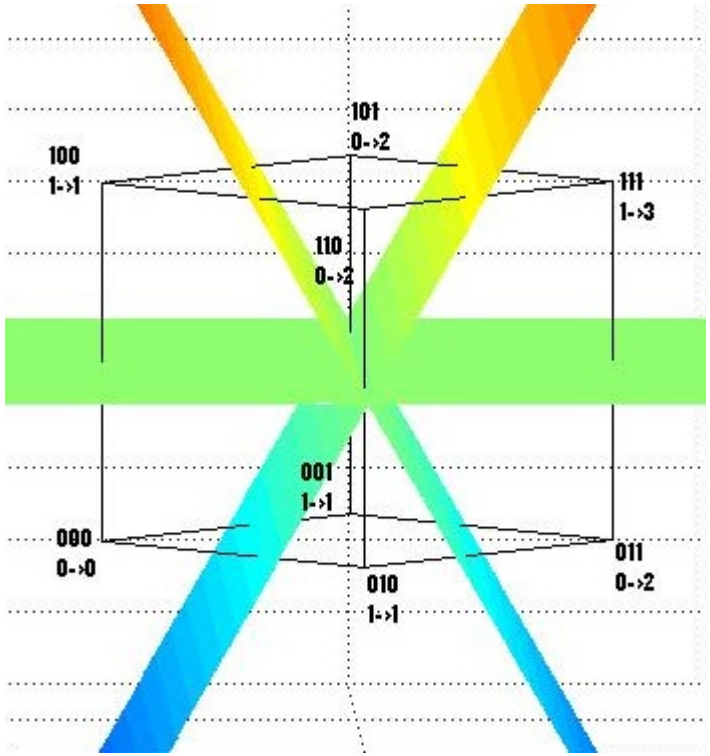
- Parity N problem can be easily learned by a single MVN-P for any N
- It is confirmed by the experimental results up to N=18

Separation of n-Dimensional Space by the m-edge

$$\left(x_1 + x_2 + x_3\right) \bmod 2$$

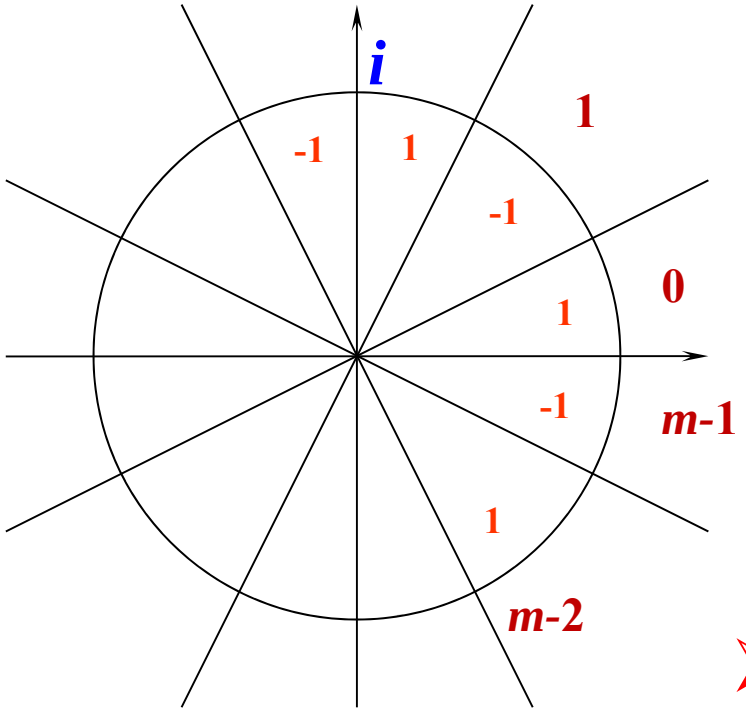
For example, Parity 3: k=2, l=3, n=3

x_1	x_2	x_3	$\oplus$	A partially defined threshold function of 6-valued logic, to which the initial function is projected
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	2
1	0	0	1	1
1	0	1	0	2
1	1	0	0	2
1	1	1	1	3



3-dimensional space is linearly separated by the 6-edge

What a Periodic Activation Function Gives?

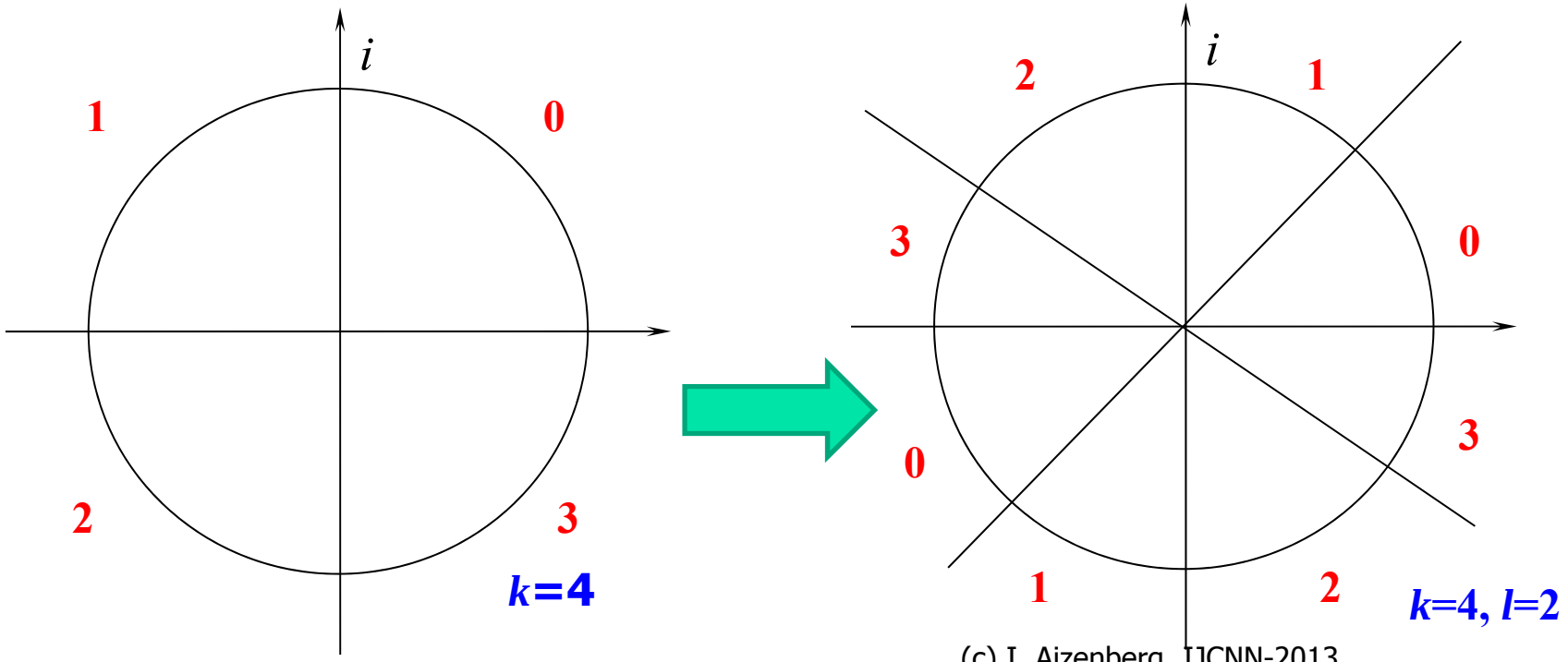


- In two-valued (Boolean) case, a periodic activation function projects 2-valued logic into $2l=m$-valued logic, and this leads to the dramatic raise of the functionality of a single neuron, which can learn non-threshold Boolean functions

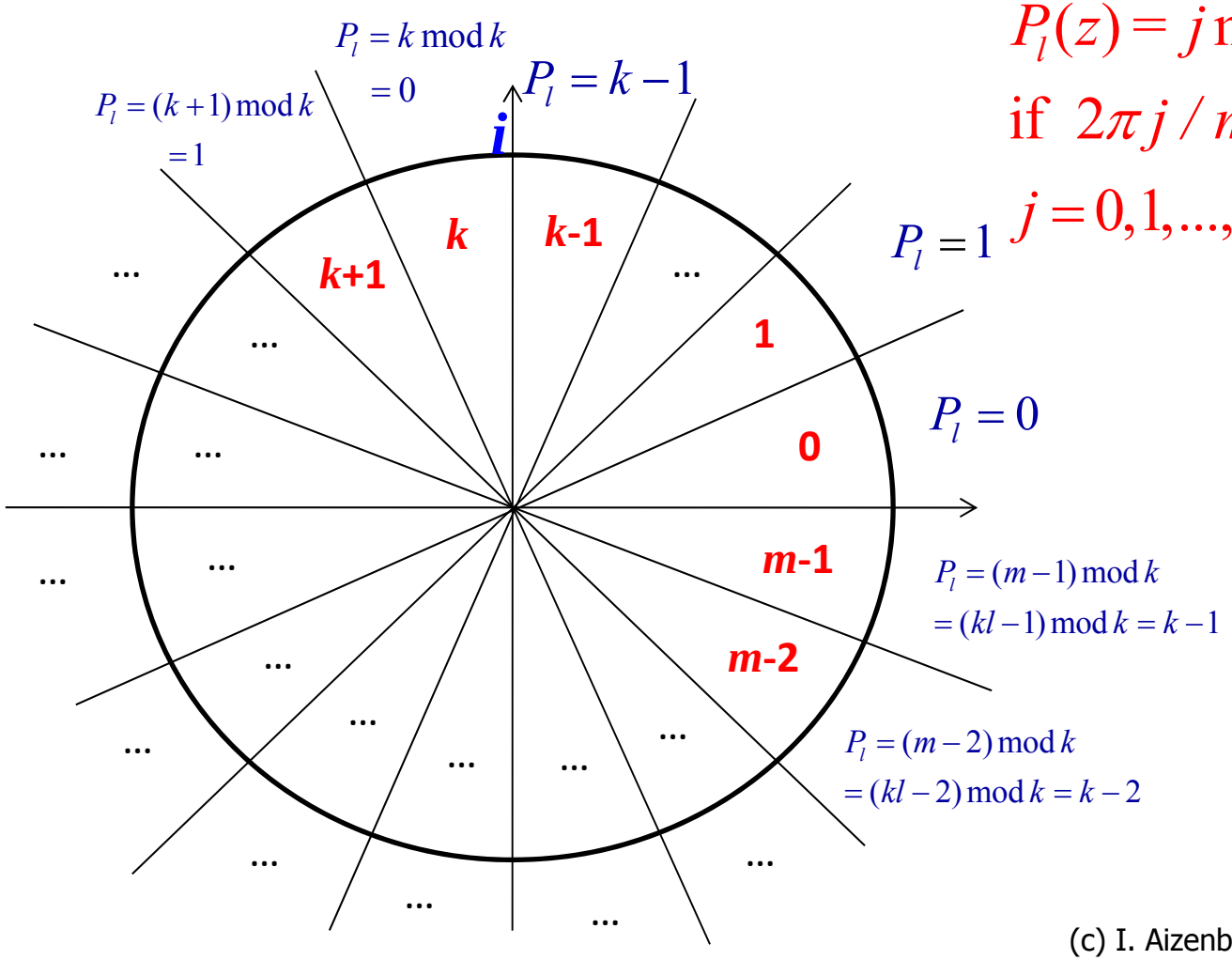
- The same is true for the k-valued case!

MVN-P: MVN with a Periodic Activation Function

- Let us introduce a periodic activation function where the values of ***k***-valued logic will be repeated periodically with a periodicity coefficient ***l***:



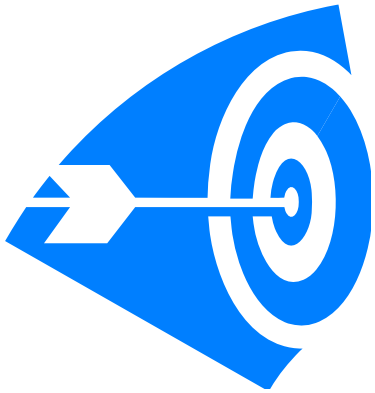
A Periodic Activation Function for MVN (introduced by I. Aizenberg in **2009-2010**)



$$P_l(z) = j \bmod k,$$

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

$$j = 0,1,\ldots,m-1; m = kl, l \ge 2$$



A Periodic Activation Function for MVN

$$P_l(z) = j \bmod k,$$

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

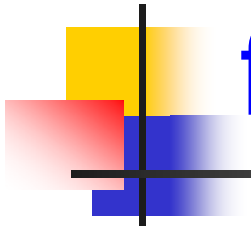
$$j = 0,1,\ldots,m-1; m = kl, l \ge 2$$

- The activation function P_l is a k-periodic l-repetitive activation function
- A periodic activation function projects k-valued logic into m-valued logic, where $m=kl$

A Periodic Activation Function for MVN

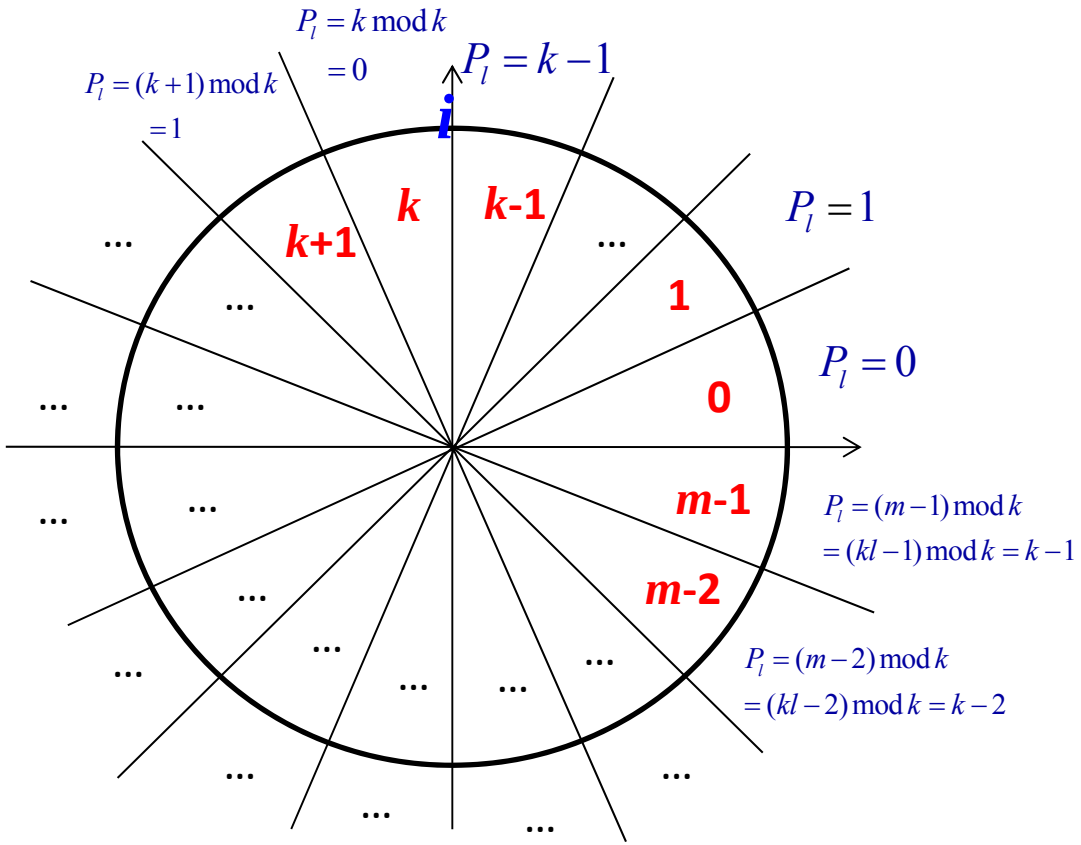
- A periodic activation function makes it possible to learn discrete k-valued input/output mappings $f : E_k^n \to M$ where $E_k = \{1, \varepsilon_k, \varepsilon_k^2, ..., \varepsilon_k^{k-1}\}$ is the set of kth roots of unity, $M = \{0,1,\ldots,k,\ldots \text{m-1}, \text{m}\}$ and
- continuous/discrete input/output mappings $f : O^n \to M$ where O is the set of points in the unit circle

A Periodic Activation Function for MVN



- Many of those k-valued functions, which cannot be learned by a single MVN with a regular activation function, can be learned by a single MVN with a periodic activation function
- Many of non- threshold functions of k-valued logic become partially defined threshold functions in kl=m-valued logic

Learning algorithm for MVN with a periodic activation function



- Learning is reduced to the regular MVN learning algorithm with the error-correction rule
- To choose the desired sector among the l ones, we have to find that sector, which the current value of the weighted sum is closer to in terms of angular distance
- Hence, the learning algorithm is semi-supervised/semi-self-adaptive

Solving Some Non-Linearly Separable Problems using a Single MVN with a Periodic Activation Function

mod k addition of n variables

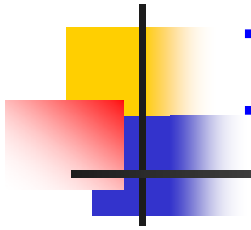
- This problem is a multiple-valued analogue of the Parity N problem for the binary case

$$f\left(x_1,\ldots,x_n\right)=\left(x_1+\ldots+x_n\right)\bmod k$$

- This problem is non-linearly separable in k-valued logic for any k and any n

mod k addition of n variables

***k* (value of *k*–valued logic)**	***n* (number of variables/)**	Average number of learning iterations (Iter.) for 7 independent runs, its standard deviation (SD), and minimal value of ***l*** for which the learning process converged.					
		Learning rule without normalization			Learning rule with normalization		
		Iter.	SD	***l***	Iter.	SD	***l***
3	**2**	14	5	**2**	18048	44957	**2**
3	**3**	2466	2268	**10**	3773	1721	**10**
3	**4**	4296	2921	**11**	391404	158688	**7**
3	**5**	78596	87158	**18**	344440	308044	**22**
3	**6**	237202	172100	**36**	50292	91260	**39**
3	**7**	518313	395671	**41**	1556379	798841	**41**
4	**2**	2693	3385	**3**	135	163	**3**
4	**3**	2571	1772	**7**	12175	5407	**7**
4	**4**	50151	35314	**10**	140850	88118	**10**
4	**5**	734691	231353	**13**	355910[1]	98208	**13**
4	**6**	131139[1]	185316	**42**	171269[1]	104685	**15**
4	**7**	108050	30309	**39**	110809	37286	**38**
5	**2**	96	22	**4**	81	16	**4**
5	**3**	1202	193	**9**	1419[1]	264	**9**
5	**4**	4604	393	**13**	4893[1]	211	**13**
5	**5**	22812	3977	**22**	17274	1682	**18**
5	**6**	105672	20071	**26**	196441[1]	5635	**22**
5	**7**	630490	192494	**52**	557635	305579	**49**
6	**2**	272	110	**4**	120[1]	33	**4**
6	**3**	109733	2400	**14**	4131	4039	**10**
6	**4**	118128	15596	**14**	48002	11652	**14**
6	**5**	71241	74463	**21**	15986	14835	**21**
6	**6**	92257	4773	**33**	194544[1]	203936	**26**
6	**7**	247232	64747	**31**	262564[1]	13614	**31**



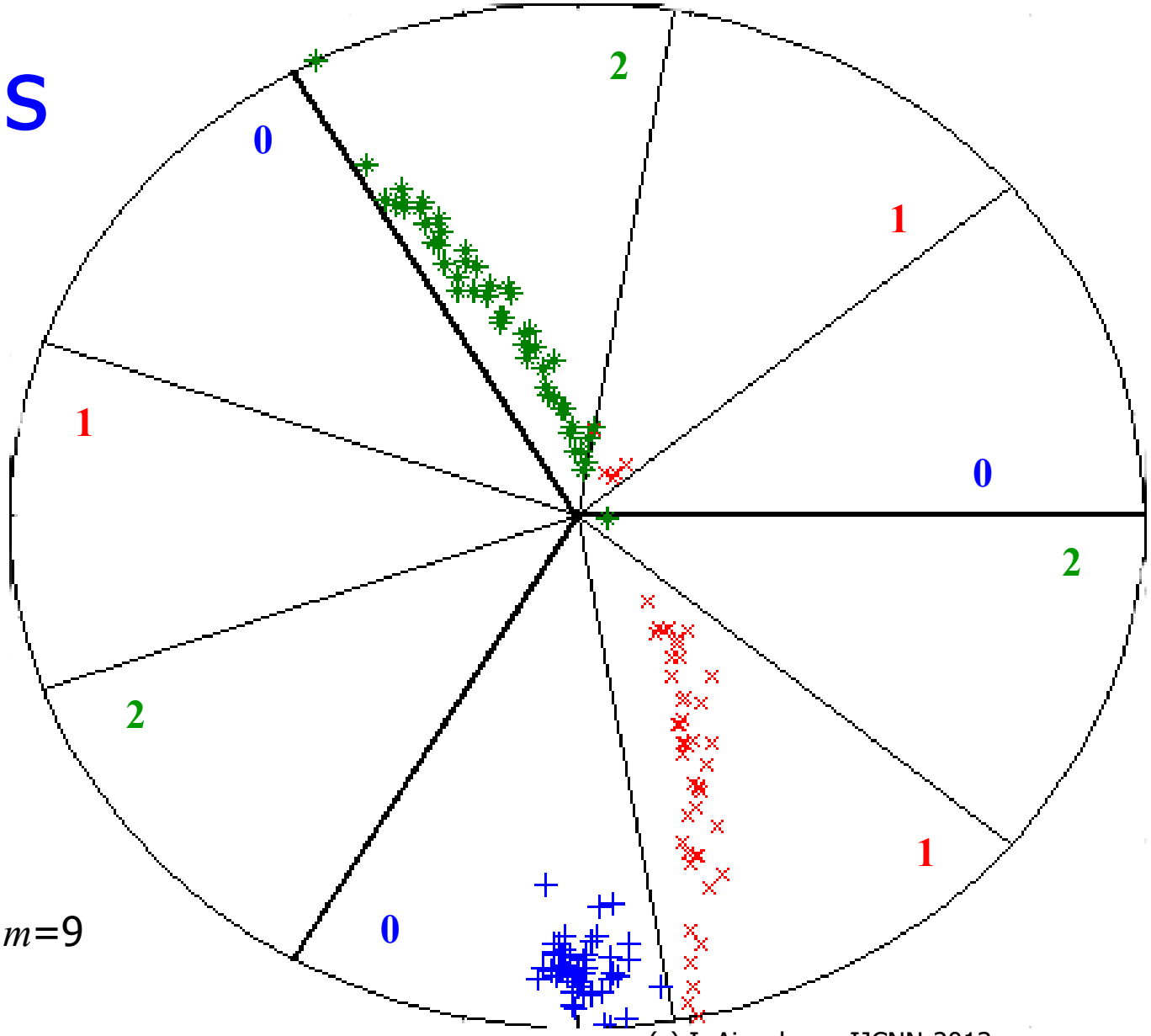
Iris

- 3 classes, 50 samples in each, 5-fold cross-validation learning set-75, testing set-75
- k=3, l=3, m=9 in the periodic activation function
- Classification rate is 97.33%, which is comparative to the best known result (97.62% dendogram-based SVM, which is much more complicated than a single neuron)
- The data set can also be learned completely with no errors

Iris

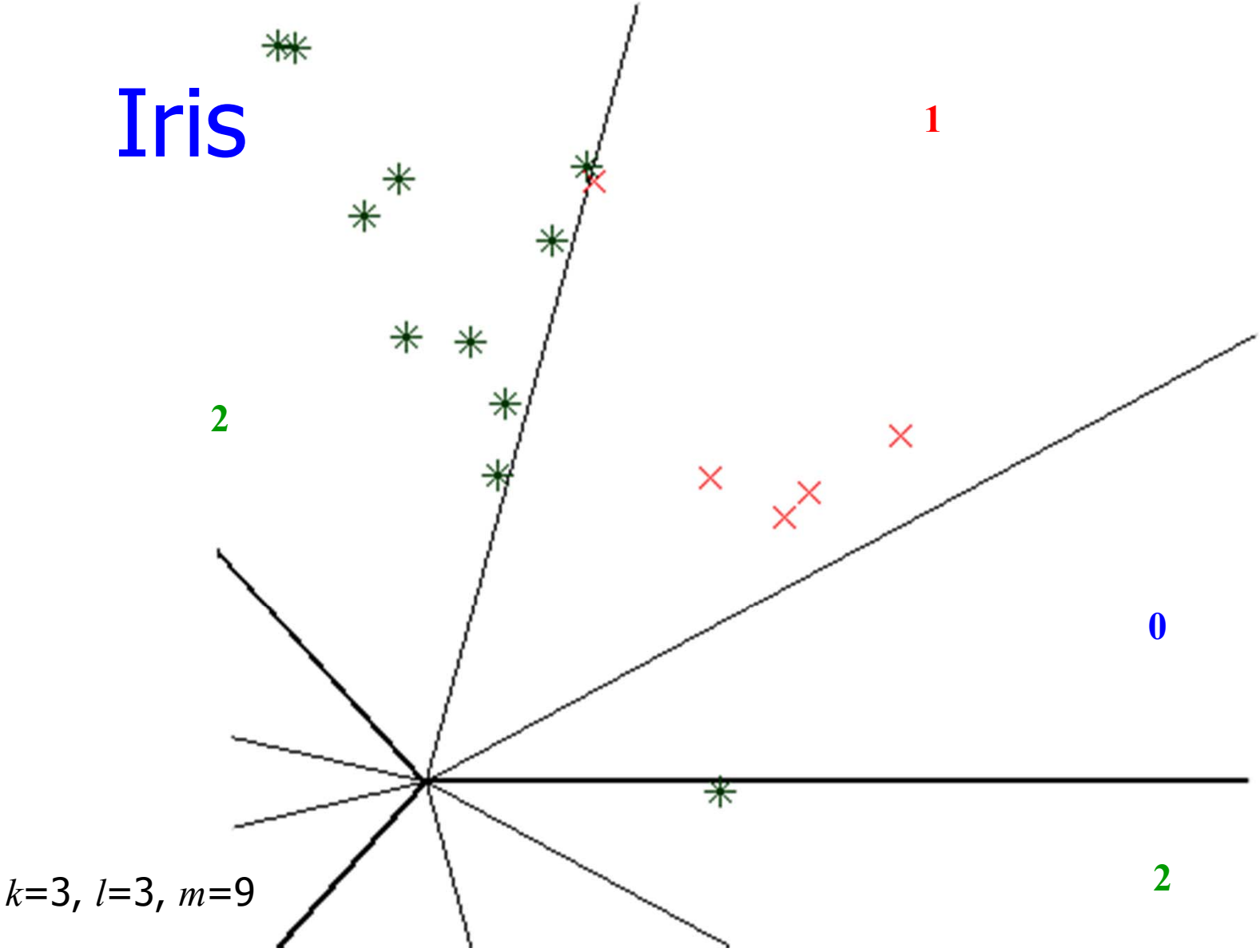
- Iris is a typical **multi-cluster** classification problem.
- While one of the classes has a single cluster, two other classes contain two disjoint clusters each.
- This means that Iris is non-linearly separable problem.
- However, it can be learned by a single MVN with a periodic activation function.

Iris

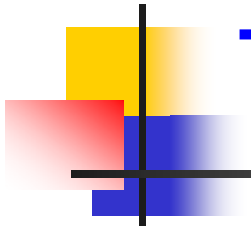


k=3, l=3, m=9

Iris



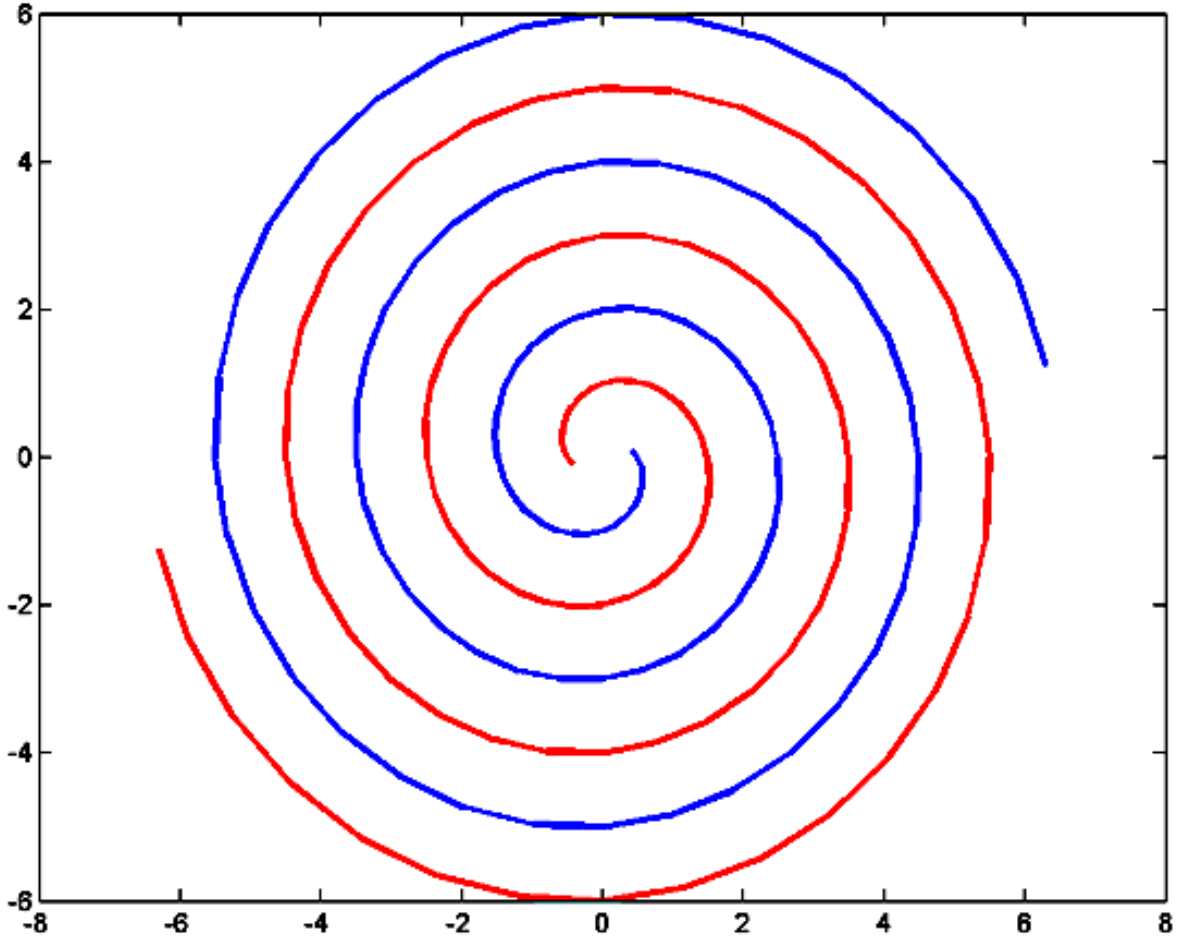
k=3, l=3, m=9



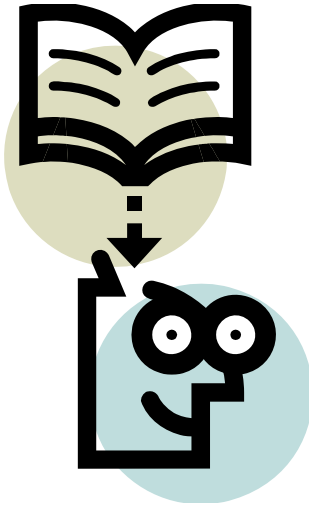
Two spirals

- 2 classes, 87 samples in each, cross-validation learning set-98, testing set-96
- k=2, l=2, m=2 in the periodic activation function
- Classification rate is 100%, which is the best known result, there is no other technique (except RBF), which gives this absolute rate
- The data set can also be learned completely with no errors

The Two Spirals Problem

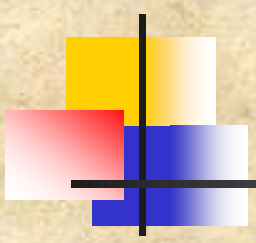


MVN: Main Applications



MVN: Applications

- Multilayer feedforward MVN-based neural network (2004-2007), which outperforms other neural networks and kernel-based techniques SVM in terms of learning speed and generalization capability when solving many benchmark and real-world problems
- Different MVN-based associative memories (1992-2004) with a high retrieval rate
- MVN-based cellular neural networks for nonlinear image filtering



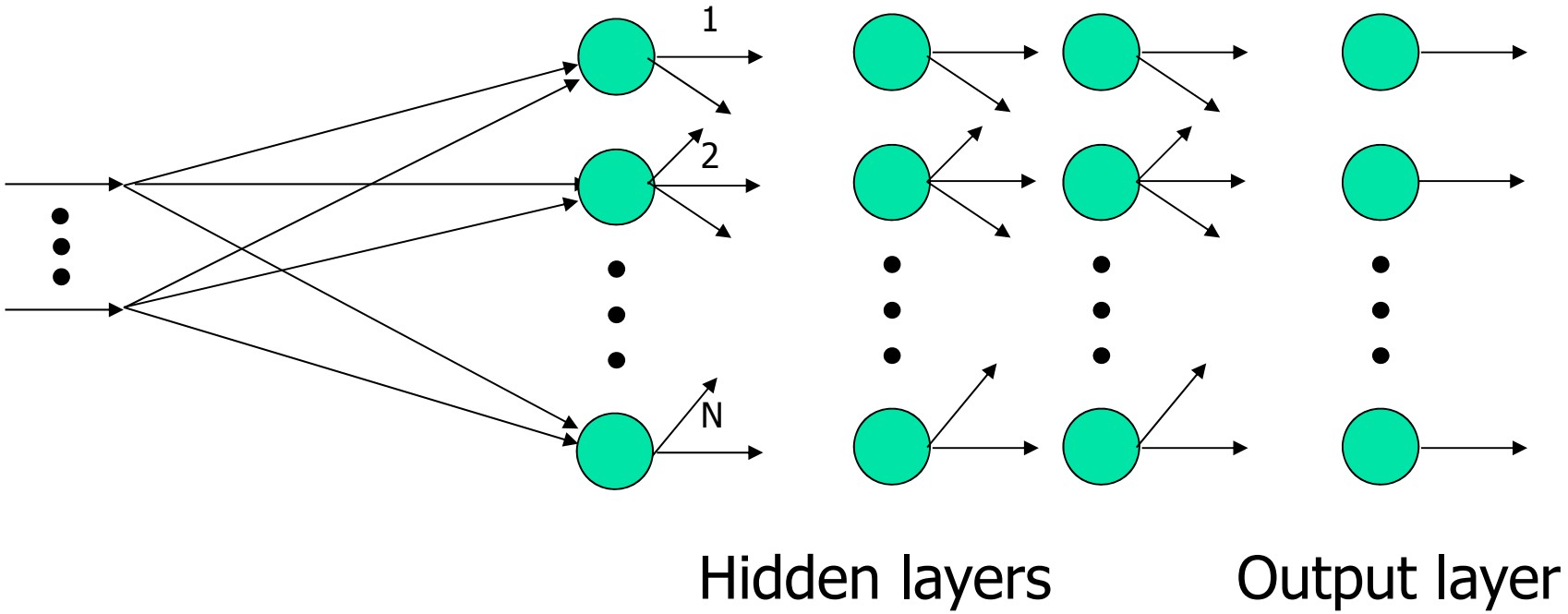
MVN- based Multilayer Feedforward Neural Network (MLMVN) and its Derivative-Free Learning

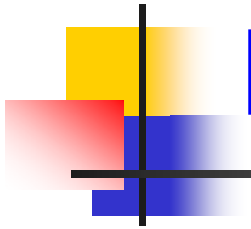
Introduced in **2004-2007** by I. Aizenberg, C. Moraga, and D. Paliy

MLMVN

- 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.
- 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.
- I. Aizenberg, "*Complex-Valued Neural Networks with Multi-Valued Neurons*", Springer, Heidelberg, **2011**.
- I. Aizenberg, A. Luchetta, and S. Manetti, "A modified Learning Algorithm for the Multilayer Neural Network with Multi-Valued Neurons based on the Complex QR Decomposition", ***Soft Computing***, vol. 16, No 4, April **2012**, pp. 563-575.

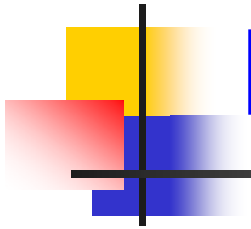
MVN-based Multilayer Feedforward Neural Network





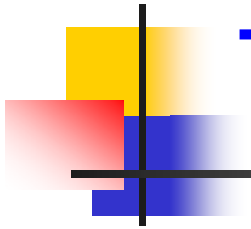
MLMVN: Key Properties

- 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 feedforward neural networks
- Better recognition/prediction/classification rate in comparison with a number of other feedforward neural networks, neuro-fuzzy networks and kernel based techniques



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



The Error Sharing Principle

1) The error of a single MVN must be shared among all the weights of the neuron
2) The network error and the errors of each particular neuron in MLMVN must be shared among those neurons from the network that contribute to this error
3) After errors of all the neurons are known, the MVN error-correction learning rule is used to correct the weights of all the neurons, one by one, layer by layer

MLMVN: Derivative- Free Learning Algorithm based on the Error-Correction rule

m - the number of layers in the network

w_i^{kj} - ith weight of the k^{th} neuron from the j^{th} layer

D_{kj} - a desired output of the k^{th} neuron from the j^{th} layer

Y_{kj} - an actual output of the k^{th} neuron from the j^{th} layer

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

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

δ_{kj} - the error for the k^{th} neuron from the j^{th} layer

N_j -the number of all neurons in the layer j

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

MLMVN: Error Backpropagation

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

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

Using this factors the error is shared with those neurons, which are connected to the neuron kj

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

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

MLMVN:
Error-Correction Learning Rule

Correction rule for the neurons from the m^{th} (output) layer (k^{th} neuron of m^{th} layer) (α is a learning rate, which should always be equal to 1):

$$\tilde{w}_i^{km} = w_i^{km} + \alpha_{km}\delta_{km}\bar{\tilde{Y}}_{im-1}, \quad i = 1,\ldots,N_{m-1}$$

$$\tilde{w}_0^{km} = w_0^{km} + \alpha_{km}\delta_{km}$$

MLMVN: Error-Correction Learning Rule

Correction rule for the neurons from the 2nd through m-1st layer
(k^{th} neuron of the j^{th} layer (j=2, …, m-1)
(α is a learning rate, which should always be equal to 1)
z_{kj} is the weighted sum before the correction of the weights:

$$\tilde{w}_i^{kj} = w_i^{kj} + \frac{\alpha_{kj}}{|z_{kj}|}\delta_{kj}\bar{\tilde{Y}}_i, \quad i = 1,\ldots,N_{j-1}$$

$$\tilde{w}_0^{kj} = w_0^{kj} + \frac{\alpha_{kj}}{|z_{kj}|}\delta_{kj}$$

MLMVN:
Error-Correction Learning Rule

Correction rule for the neurons from the 1st hidden layer
(α is a learning rate, which should always be equal to 1)
z_{kj} is the weighted sum before the correction of the weights
$x_1, \ldots, x_n$ are the network inputs:

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

$$\tilde{w}_0^{k1} = w_0^{k1} + \frac{\alpha_{k1}}{|z_{k1}|}\delta_{k1}$$

MLMVN Learning: Example

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

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

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

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

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

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.

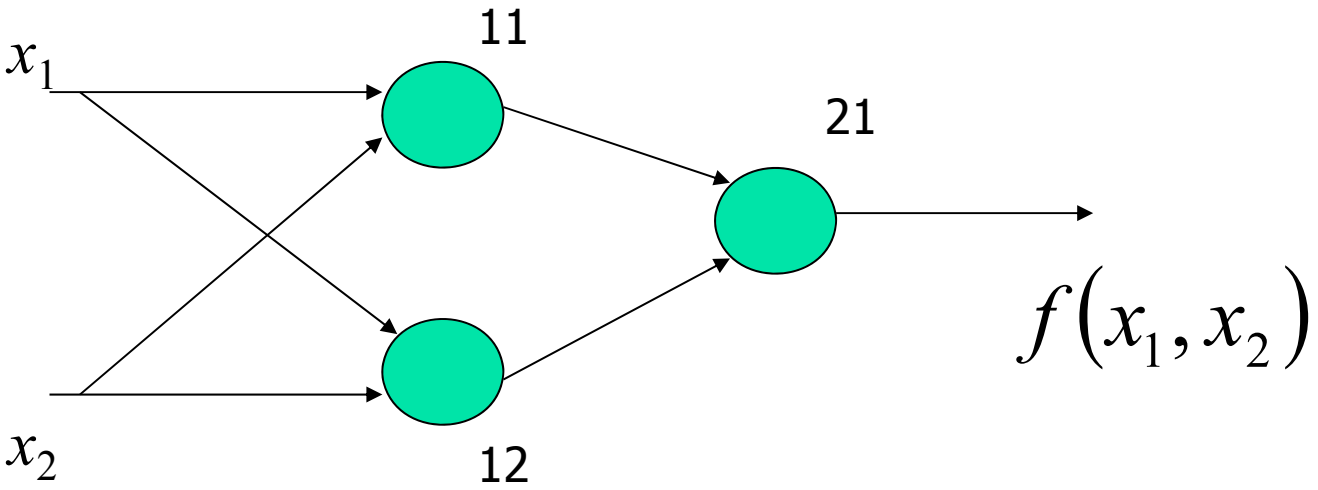
and for the mean square error

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

MLMVN Learning: Example

Let us train the 2→1 MLMVN

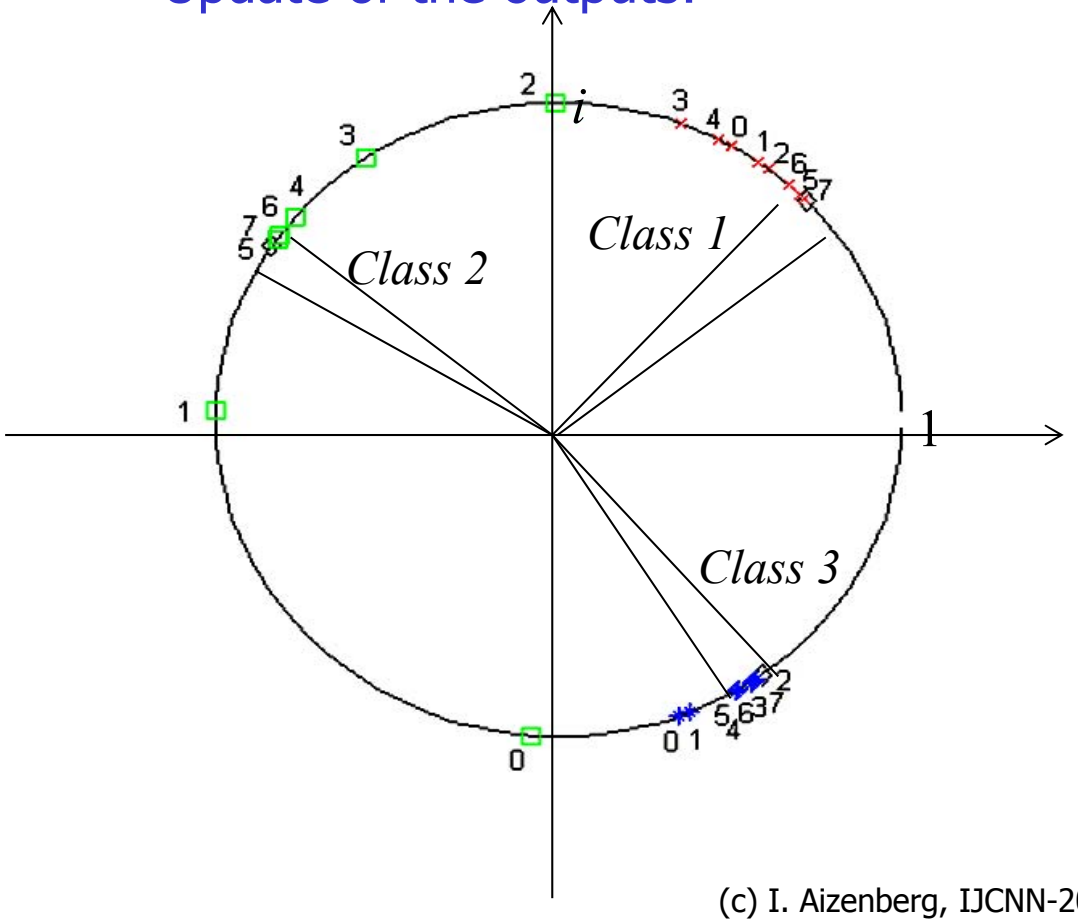
(two hidden neurons and the single neuron in the output layer



MLMVN Learning: Example

The training process converges after 7 iterations

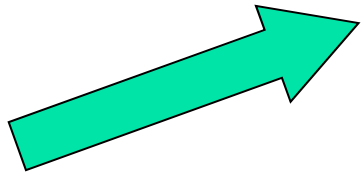
Update of the outputs:



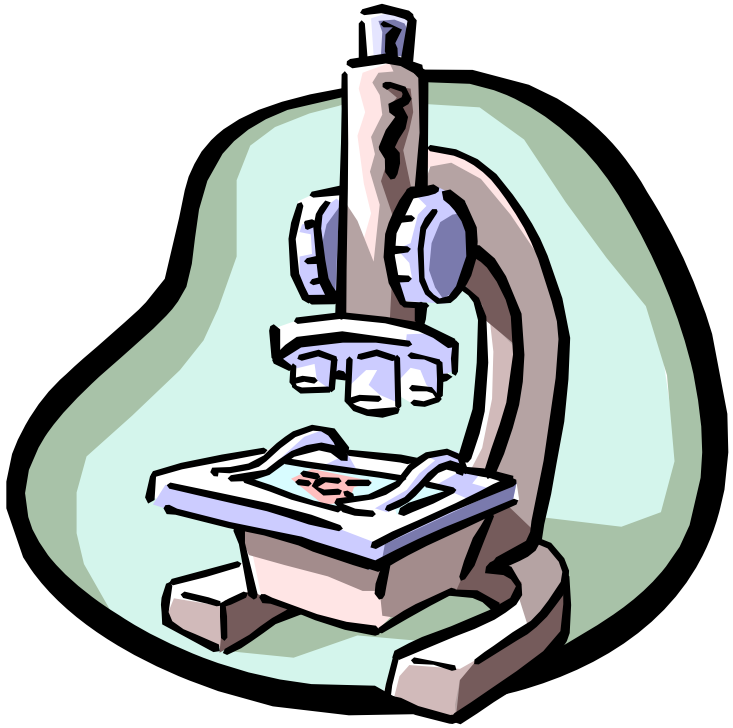
MLMVN Learning: Example

Iter	1	2	3	4	5	6	7
MSE	0.1346	0.3280	0.1178	0.0417	0.0035	0.0032	0.0009

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

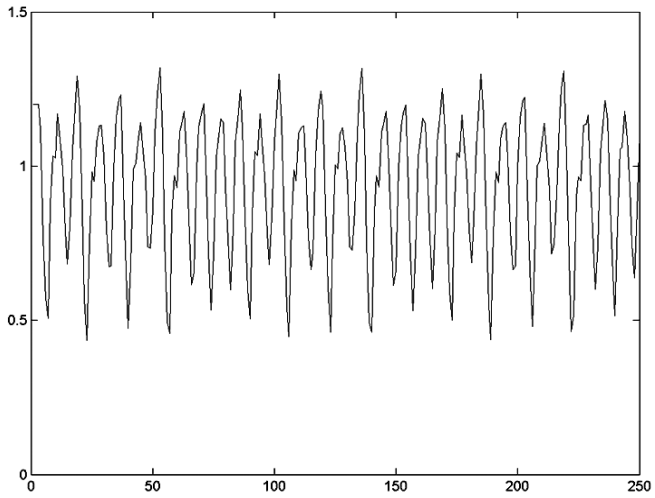


MLMVN: Simulation Results

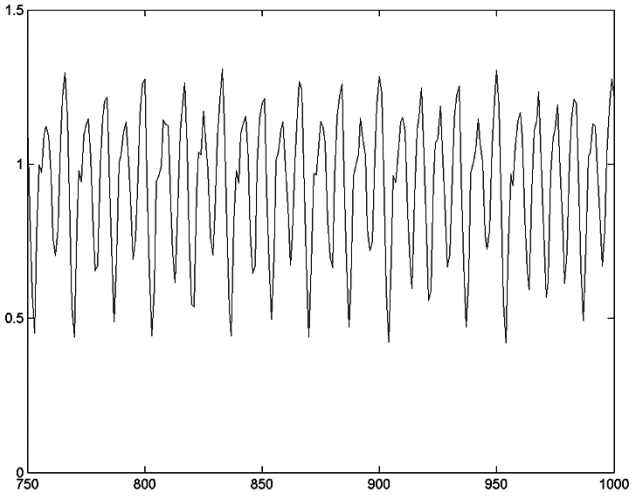


Mackey-Glass time series prediction

Training Data:

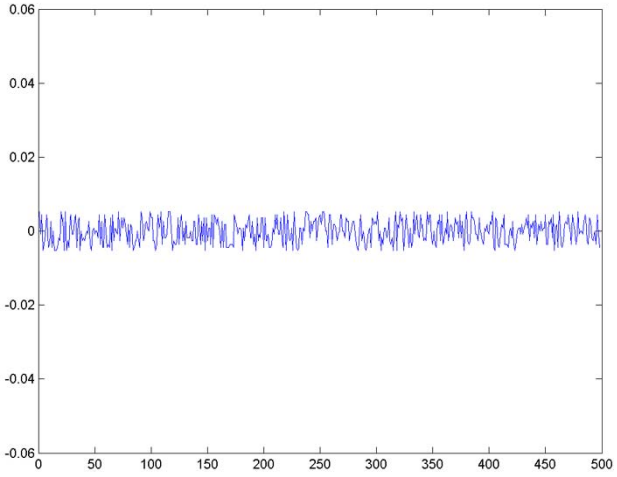


Testing Data:

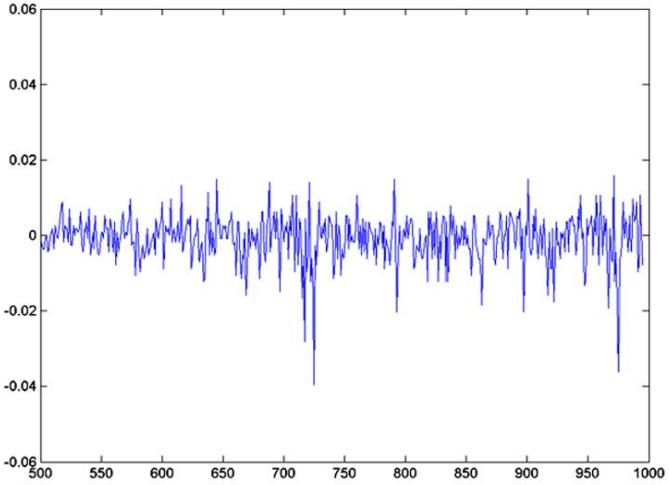


Mackey-Glass time series prediction

RMSE Training:



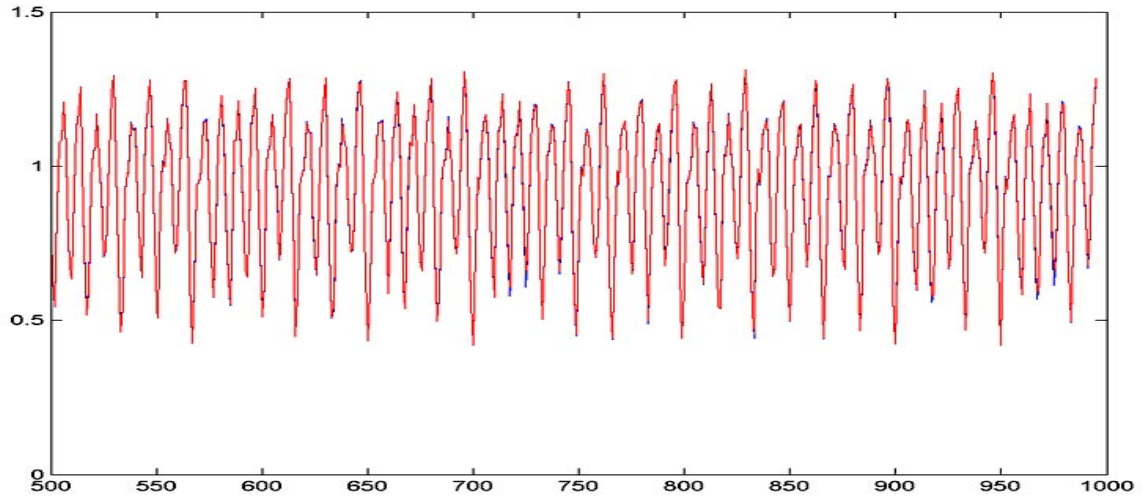
RMSE Testing:



Mackey-Glass time series prediction

Testing Results:

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



Mackey-Glass time series prediction

Comparison of MLMVN to other tools:

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

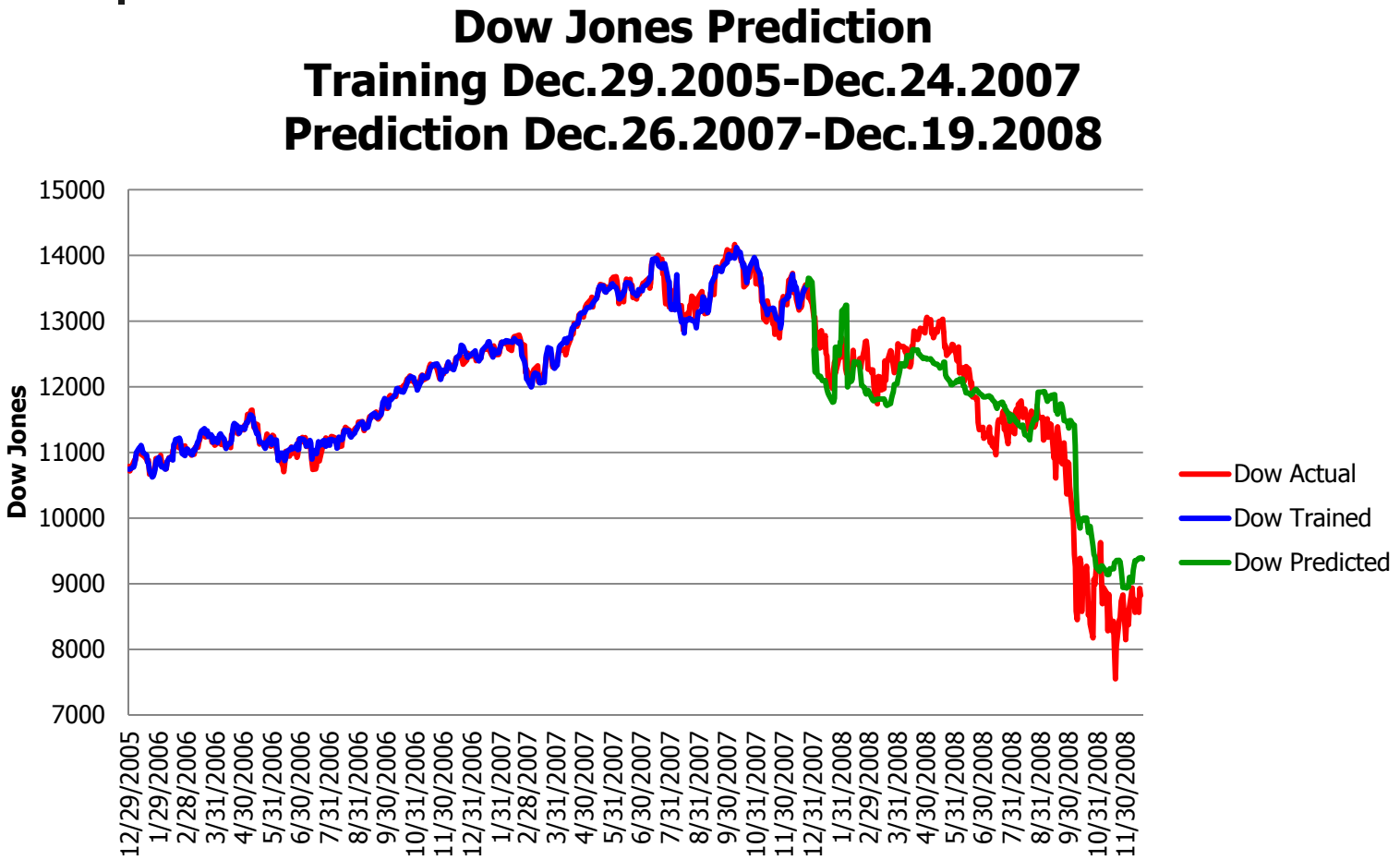
MLMVN outperforms all other networks in:

- The number of parameters (complexity)
- Learning speed
- Generalization Capability

Real World Problems:

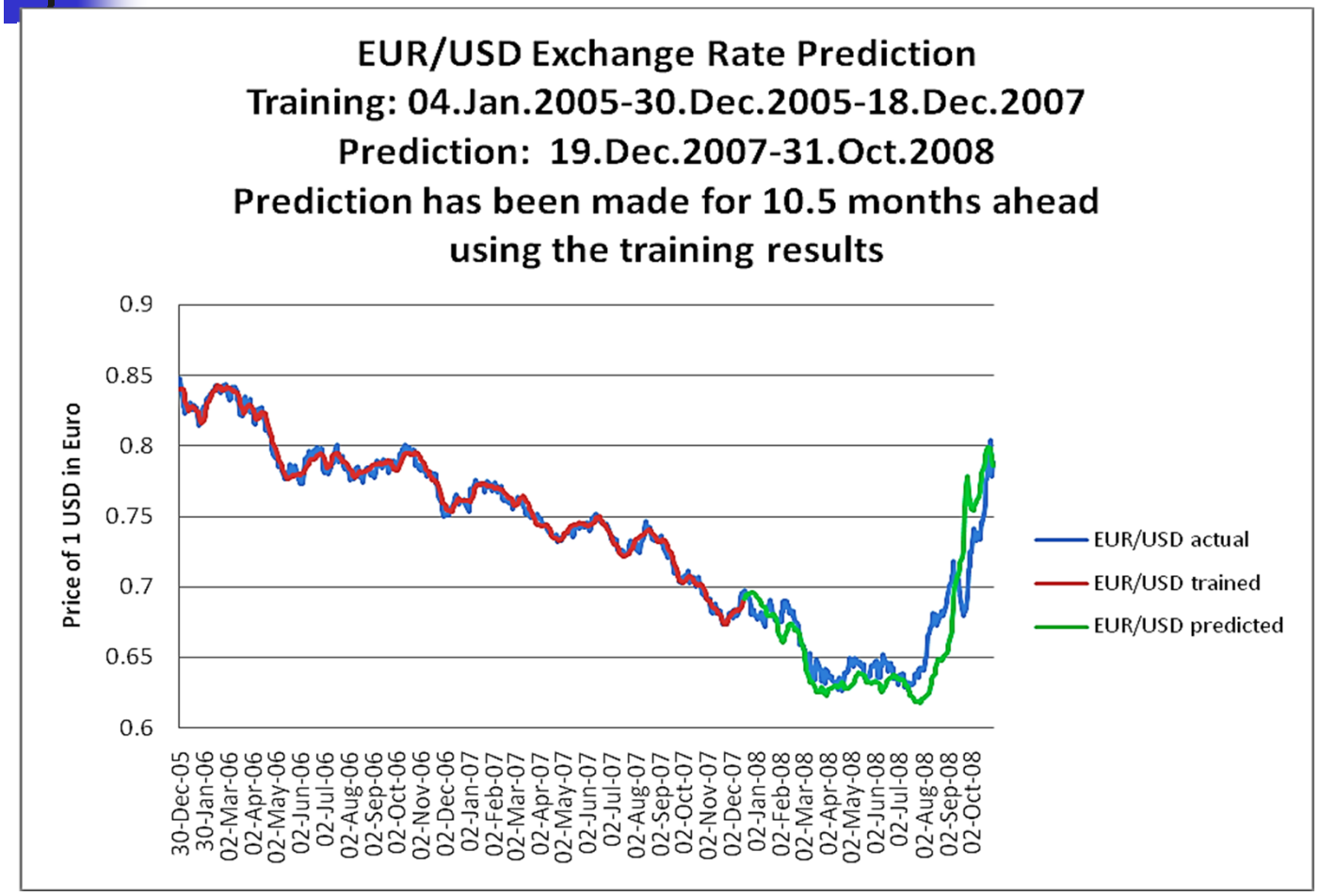
Solving Using MLMVN

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 business days within a year)

Prediction of the EUR/USD Exchange Rate



- 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)

Blurred Image Restoration (Deblurring) and Blur Identification by MLMVN

Blurred Image Restoration (Deblurring) and Blur Identification by MLMVN

- 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.
- I. Aizenberg, "*Complex-Valued Neural Networks with Multi-Valued Neurons*", Springer, Heidelberg, **2011**.

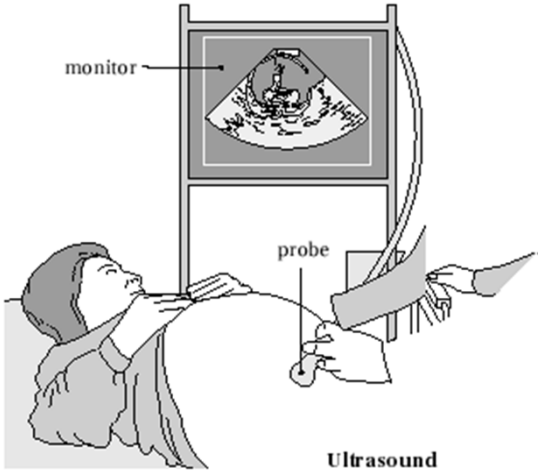
Problem statement: capturing

- Photo

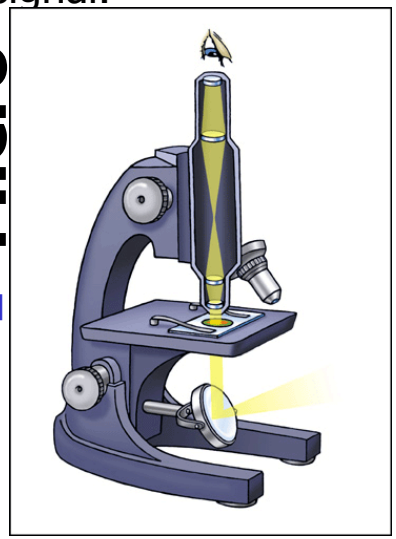
- Mathematically a variety of capturing principles can be described by the Fredholm integral of the first kind

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

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



- Micro



- Tomography

Problem statement: deblurring

- Mathematically blur is caused by the convolution of an image with the distorting kernel.
- Thus, removal of the blur is reduced to the deconvolution.
- Deconvolution is an ill-posed problem, which results in the instability of a solution. The best way to solve it is to use some regularization technique.
- To use any kind of regularization technique, it is absolutely necessary to know the distorting kernel corresponding to a particular blur: so **it is necessary to identify the blur**.

Problem statement: deblurring

- The observed image given in the following form:

$$z(x,y) = (v * q)(x,y) + \eta(x,y)$$

where "*" denotes a convolution operation, q is an image, v is a point spread function (PSF), and η is noise

- In the frequency domain, this convolution results in

$$Z(u,v) = V(u,v)Q(u,v) + N(u,v)$$

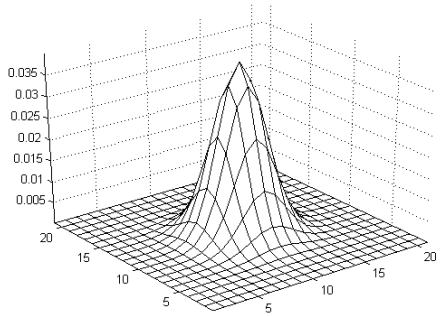
where Q is an image Fourier spectrum, V is a PSF Fourier Spectrum, N is a noise Fourier spectrum, u and v are the frequencies

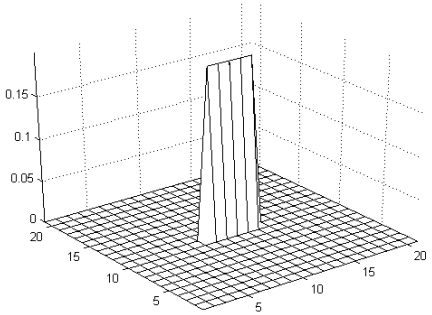
Blur Identification

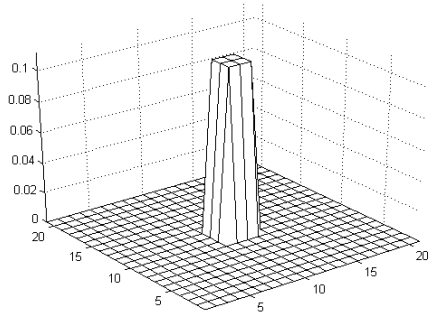
- We use MLMVN to recognize **Gaussian**, **motion** and **rectangular** (**boxcar**) blurs.
- We aim to identify simultaneously both blur (PSF), and its parameters using a single neural network.

Considered PSF

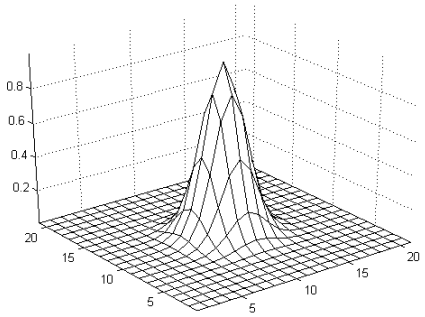
PSF in time domain



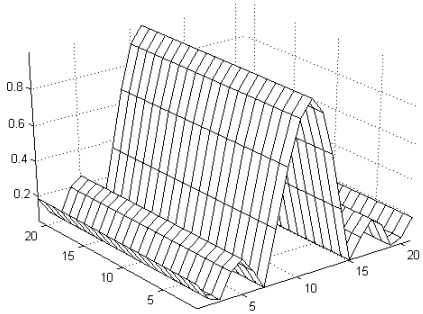




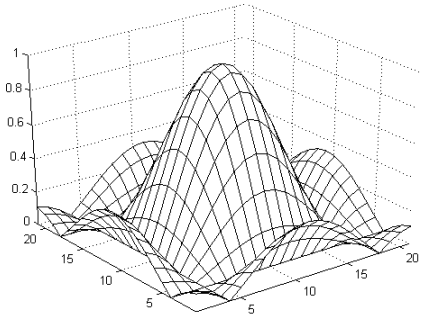
PSF in frequency domain



Gaussian



Motion



Rectangular

Considered PSF

The **Gaussian** PSF: $$v(t) = \frac{1}{2\pi\tau^2}\exp\left(-\frac{t_1^2 + t_2^2}{\tau^2}\right)$$

τ^2 is a parameter (variance)

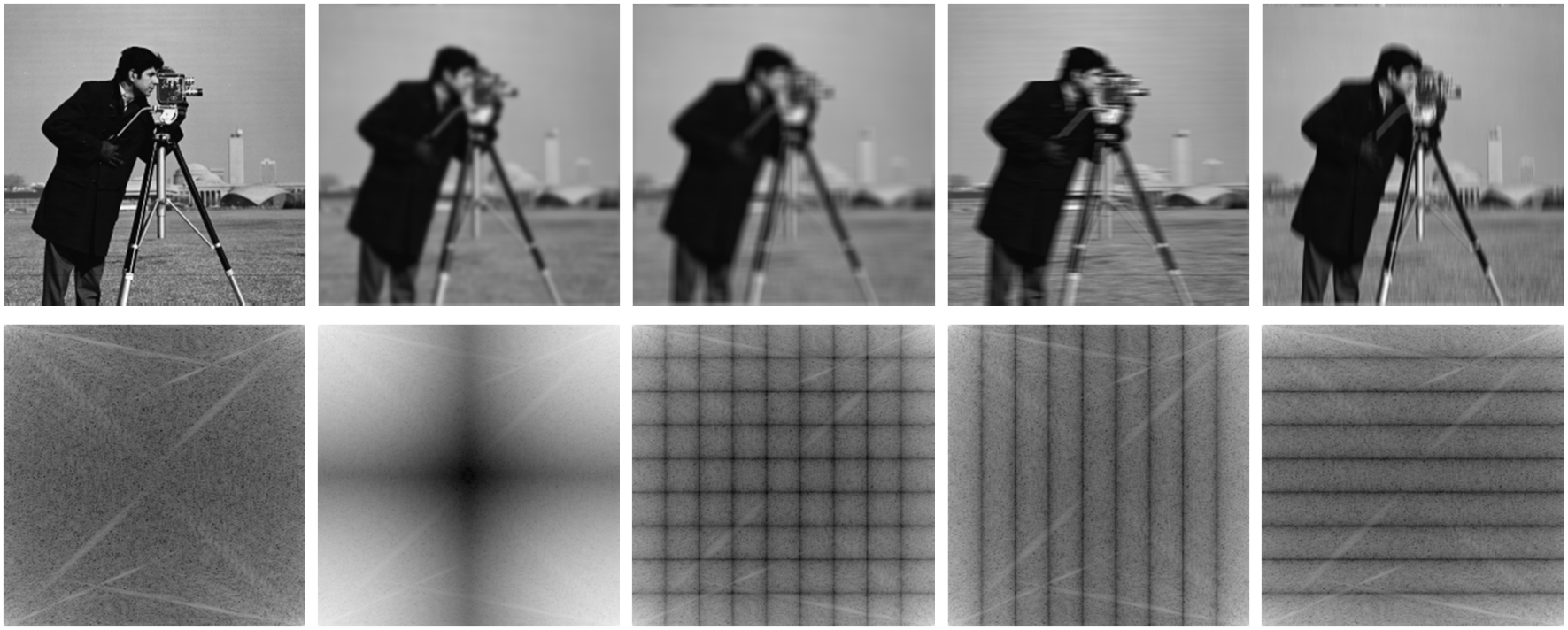
The **uniform linear motion**: $$v(t) = \begin{cases} \frac{1}{h}, & \sqrt{t_1^2 + t_2^2} < h/2,\ t_1\cos\phi = t_2\sin\phi, \\ 0, & \text{otherwise}, \end{cases}$$

h is a parameter (the length of motion)

The **uniform rectangular**: $$v(t) = \begin{cases} \frac{1}{h^2}, & |t_1| < \frac{h}{2},\ |t_2| < \frac{h}{2}, \\ 0, & \text{otherwise}, \end{cases}$$

h is a parameter
(the size of smoothing area)

Degradation in the frequency domain:



True Image | Gaussian | Rectangular | Horizontal Motion | Vertical Motion

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

Training Vectors

- We state the problem as a recognition of the shape of $\boldsymbol{V}$, which is a Fourier spectrum of PSF $\boldsymbol{v}$ and its parameters from the Power-Spectral Density, whose distortions are typical for each type of blur.

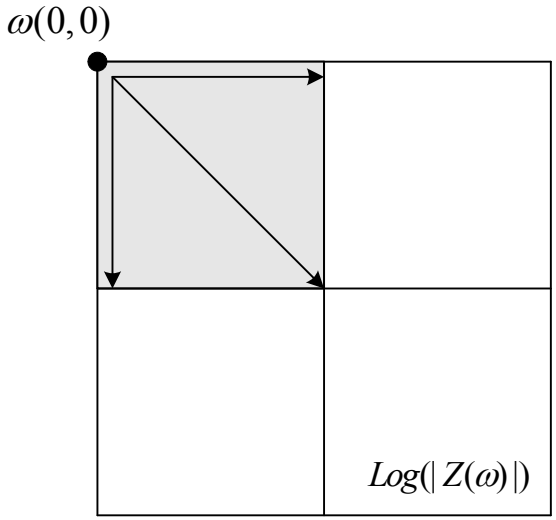
Training Vectors

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

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

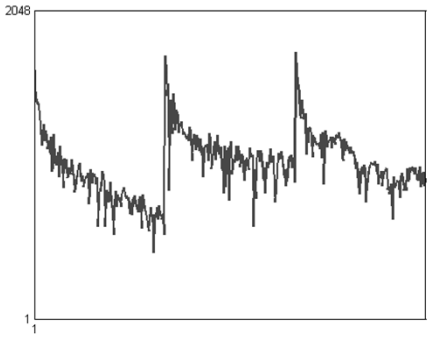
for

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

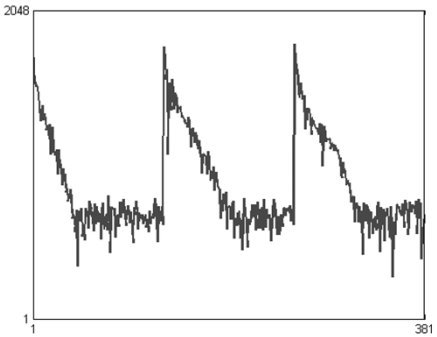


- LxL is a size of an image
- the length of the **pattern vector** is $n = 3L/2-3$

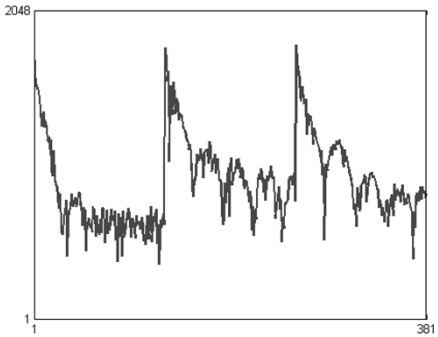
Examples of training vectors



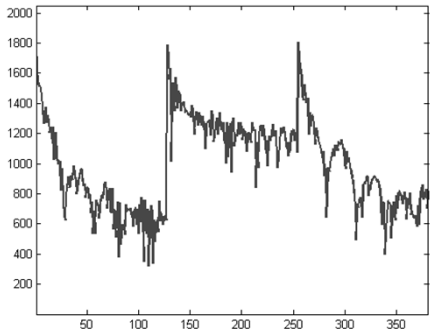
True Image



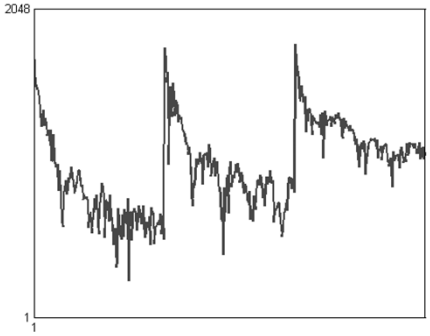
Gaussian



Rectangular

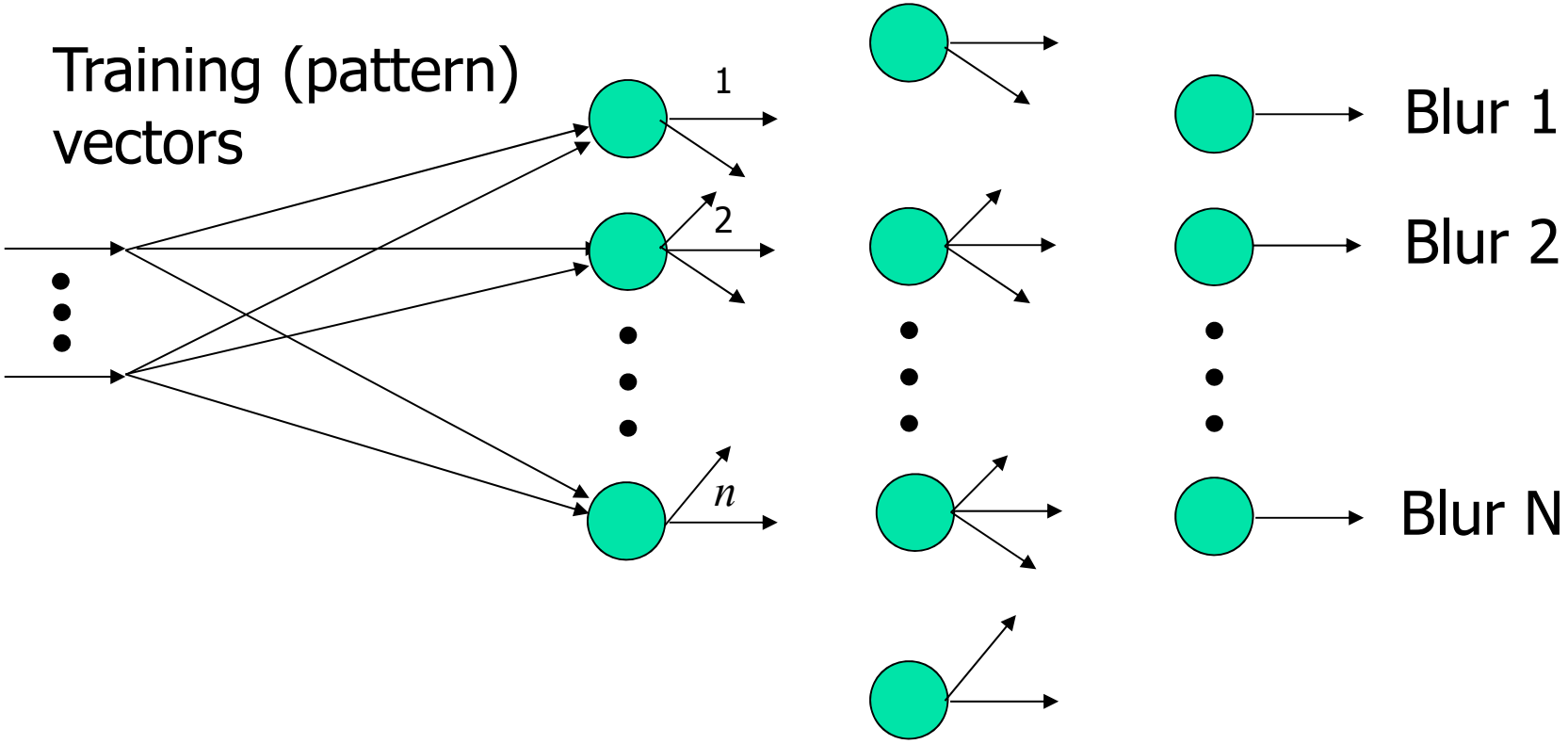


Horizontal Motion

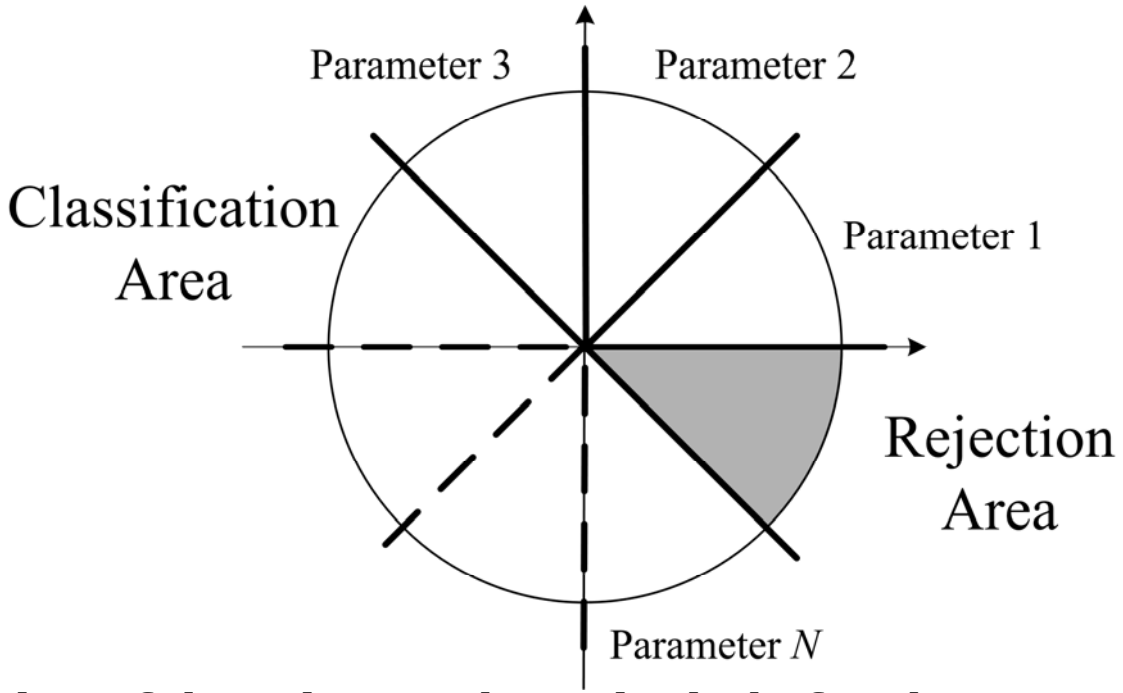


Vertical Motion

MLMVN 5→35→6


Training (pattern) vectors
1
2
n
Blur 1
Blur 2
Blur N
Hidden layers
Output layer

Output Layer Neuron



- **Reservation of domains on the unit circle for the output neuron**
- **Each output neuron recognizes a certain blur and its parameter value**
- **The number of domains matches the number of parameter values**

Simulations

Experiment 1 (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;

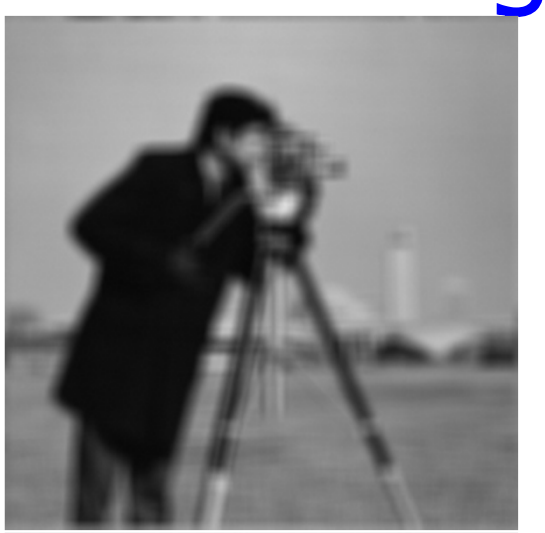
6) rectangular has sizes 3x3, 5x5, 7x7, 9x9.

Recognition 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

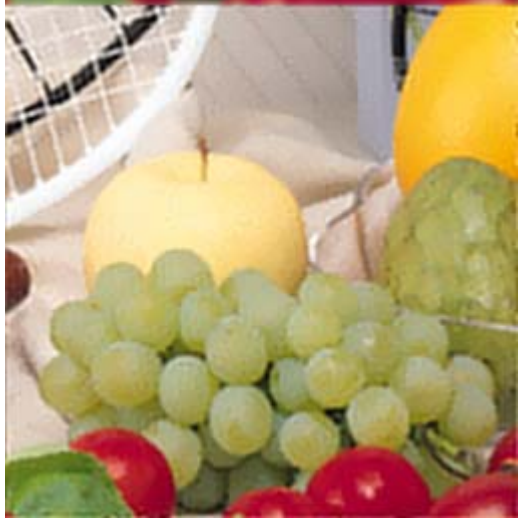
Restored images

Blurred noisy image: rectangular 9x9



Blurred noisy image: Gaussian, σ=2

Restored



Restored

Recognition of Blurred Images

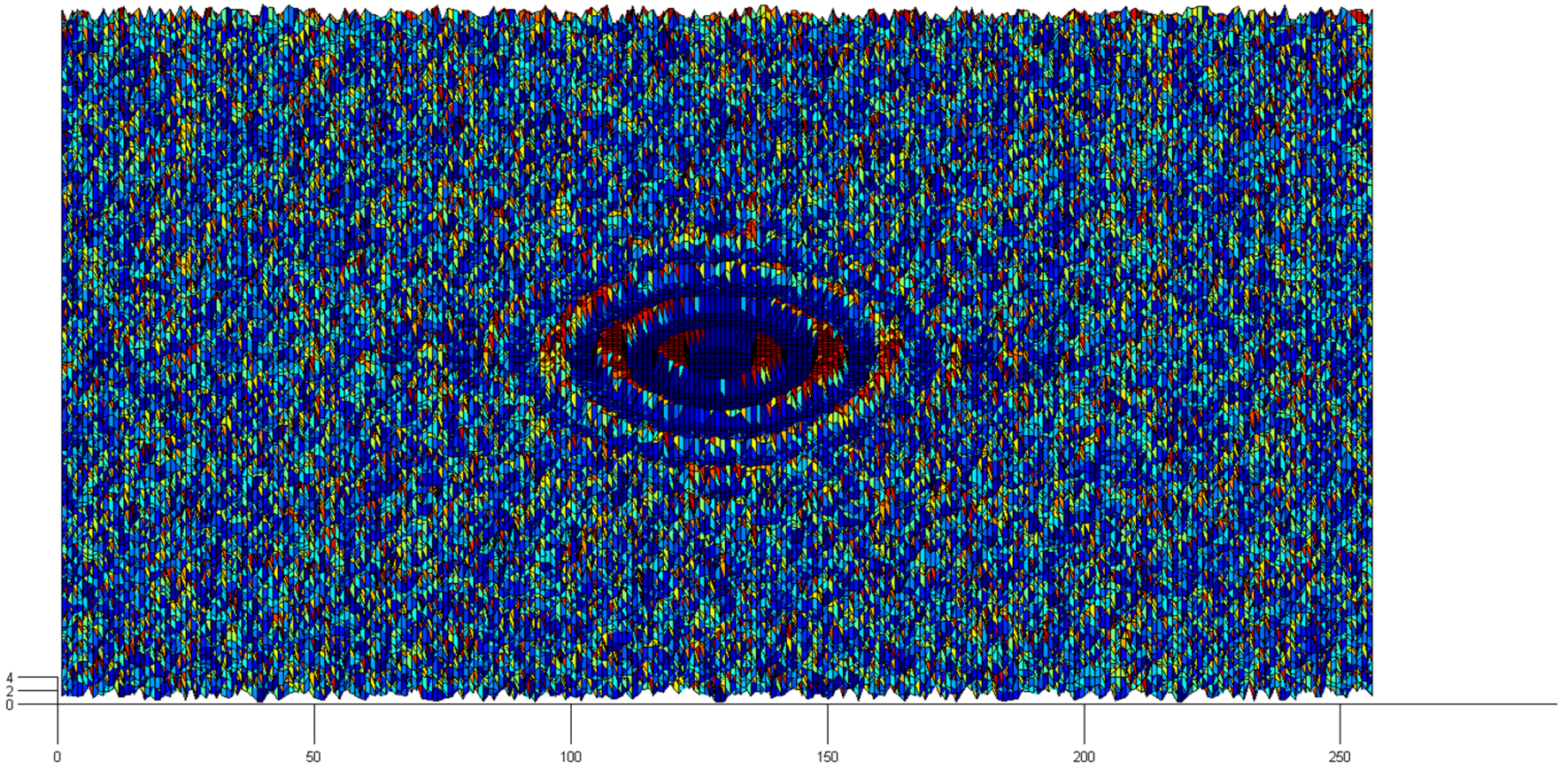
Blurred Image Restoration (Deblurring) and Blur Identification by MLMVN

- I. Aizenberg, S. Alexander, and J. Jackson, "Recognition of Blurred Images Using Multilayer Neural Network Based on Multi-Valued Neurons", *Proceedings of the 41st IEEE International Symposium on Multiple-Valued Logic* (ISMVL-2011), May, **2011**, IEEE Computer Society Press, pp. 282-287.
- I. Aizenberg, S. Alexander, and J. Jackson, "Classification of Blurred Textures using Multilayer Neural Network based on Multi-Valued Neurons", *Proceedings of the 2011 IEEE International Joint Conference on Neural Networks* (*IJCNN* 2011), August, **2011**, pp. 1328-1335.

Recognition of Blurred Images

- Blur seriously affects magnitude of the Fourier spectrum
- However, symmetric PSFs theoretically should not affect phase of the Fourier spectrum
- In practice, symmetric PSFs just slightly affect phases corresponding to lower frequencies

3D Plot of Difference of Phases of the original and blurred images



Blue – the difference is close to 0

Feature Space

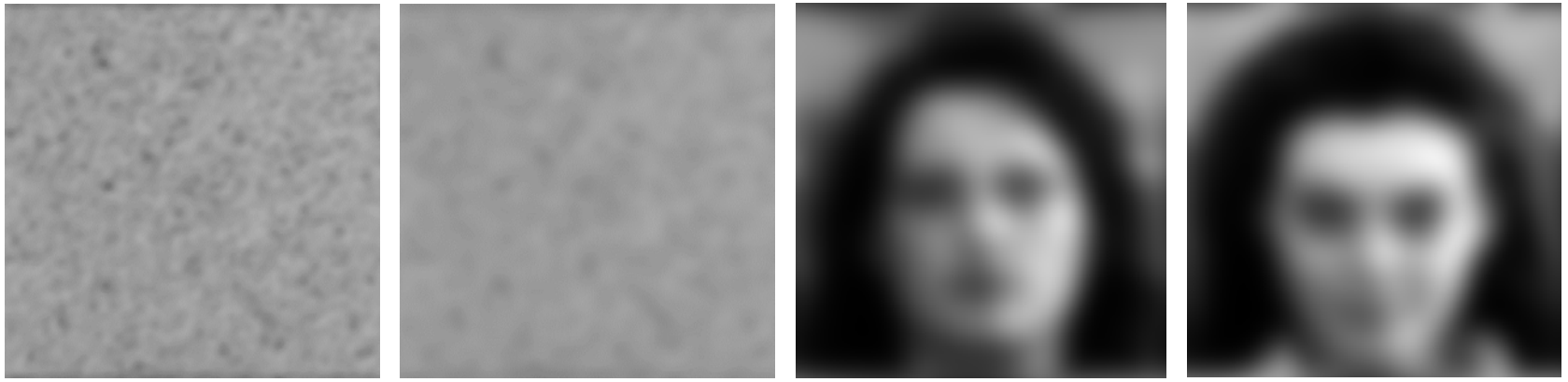
- Hence phase of the Fourier transform can be used as the feature space for blurred image recognition and classification
- Moreover, it is enough to use just phases corresponding to the lowest frequencies (1-3)

Recognition of Blurred Images

- 42 classes (21 facial and 21 textural images)
- Only clean images (1 per class) were used for learning

Recognition of Blurred Images

- Then 6468 blurred images (154 per each of 42 classes), which did not participate in the learning process, were used for testing



Recognition of Blurred Images

- The best results are obtained using MLMVN 24→7→1 (24 inputs, 7 neurons in the hidden layer, and 1 output neuron with 42-valued activation function)
- Learning takes less than 1 minute
- 100% classification accuracy (all 6468 blurred images are correctly recognized) for 5 independent learning sessions

EEG Brain-Computer Interface with the MLMVN-based Decoder

EEG Brain-Computer Interface with the MLMVN-based Decoder

- N.V. Manyakov, I. Aizenberg, N. Chumerin, and M. Van Hulle, "Phase-Coded Brain-Computer Interface Based on MLMVN", book chapter in *Complex-Valued Neural Networks: Advances and Applications* (A. Hirose – Ed.), Wiley, **2013**, pp. 185-208

- N. Manyakov, N. Chumerin, A. Combaz, A. Robben, M. van Vliet, and M. Van Hulle, "Decoding Phase-based Information from Steady-State Visual Evoked Potentials with Use of Complex-Valued Neural Network", to appear in the Proc. of. IDEAL-2011, University of East Anglia, Norwich, UK, September, **2011**

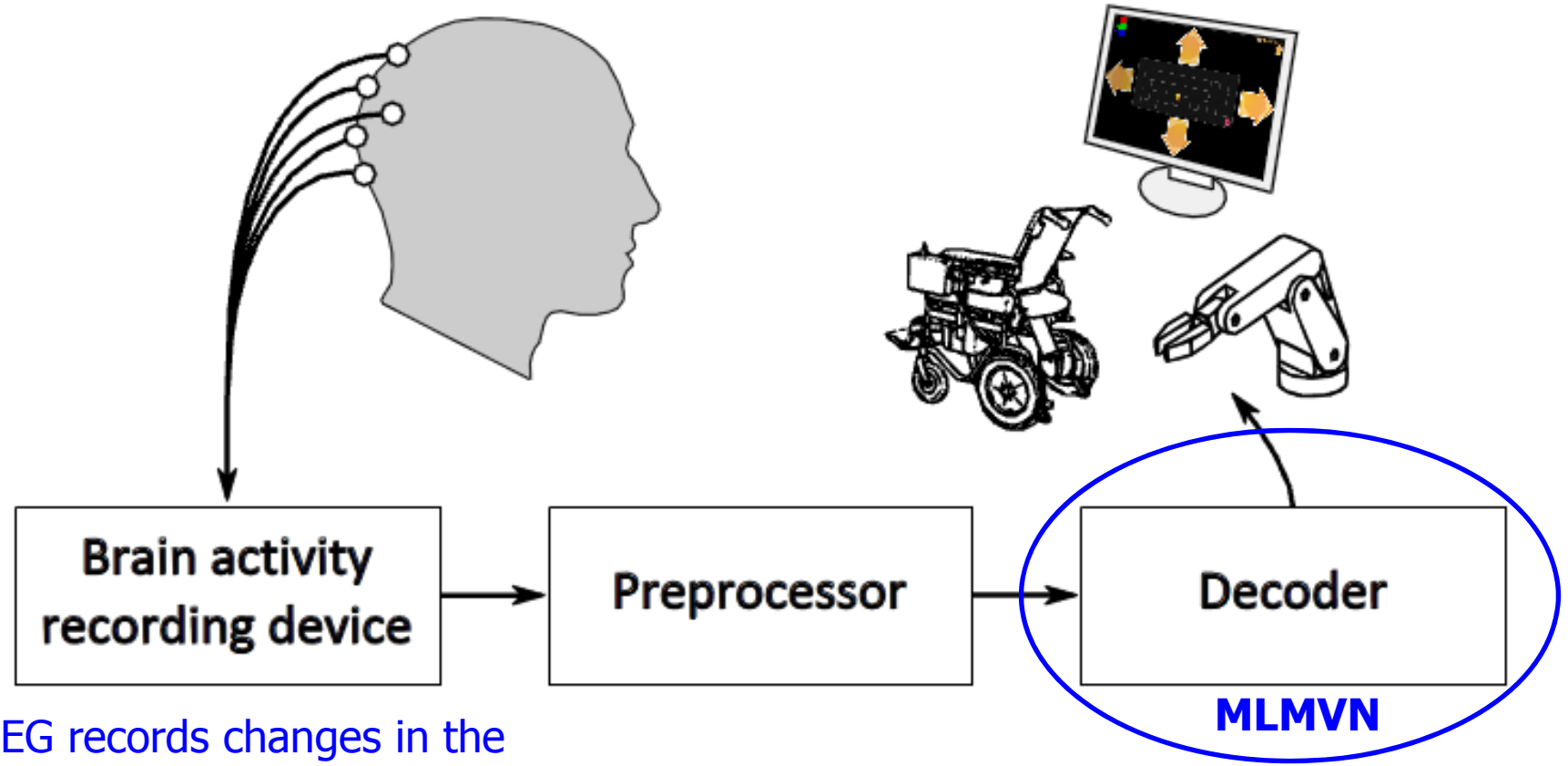
Brain-Computer Interface

- The Brain-Computer Interface (BCI) records brain activity, decodes it, and issues commands, all with the aim to enable the subject, from whom the recording are made, to interact with the external world, by controlling a computer program, a robot actuator, and so on, bypassing the need for muscular activity.

Decoding Phase-based Information in Brain-Computer Interface

- Steady-State Visual Evoked Potentials (SSVEP) brain-computer interfaces are based on the decoding of the phase information
- Since MVN inputs are completely determined by phase, an MVN-based neural network can be used for solving this problem

EEG Brain-Computer Interface



EEG records changes in the brain electrical activity

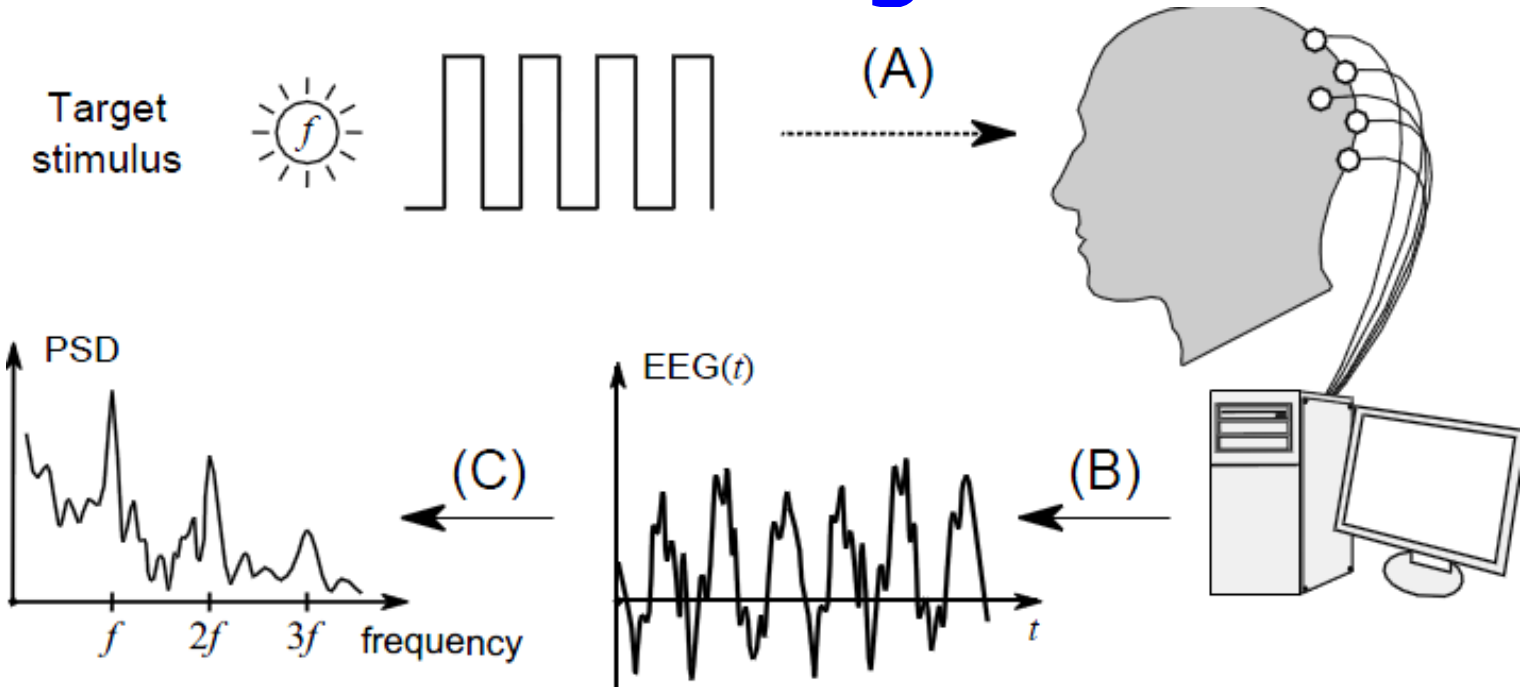
EEG Decoding

- EEG Decoding is a crucial part of BCI design
- A decoder transforms a preprocessed signal into the control signal of the external device
- The brain generates multiple signals at any moment, so their proper decoding is of crucial importance for proper functioning of the external device

Steady-State Visual Evoked Potential (SSVEP)

- SSVEP relies on the psychophysiological properties of EEG responses recorded from the occipital pole during the periodic presentation of identical visual stimuli (i.e., flickering stimuli).
- When the periodic presentation is at a sufficiently high rate (above 6 Hz), the individual transient visual responses overlap, leading to a steady state signal: the signal resonates at the stimulus rate and its multipliers
- This means that, when the subject is looking at stimuli flickering at frequency f, the frequency f and its harmonics 2f, 3f, … are salient in the Fourier transform of the EEG signal

SSVEP Decoding Scheme



Schematic overview of the SSVEP decoding approach:
(A) the subject looks at the Target stimulus, flickering at frequency f
(B) noisy EEG-signals are recorded
(C) the power spectral density plot of the EEG signal (estimated over a sufficiently large time window) shows dominant peaks at f, 2f and 3f.

Phase-coded SSVEP BCI

- In order to increase the number of encodable targets in SSVEP-based BCI, the phase can be used in addition (or as an alternative) to the frequency
- Even a single frequency could be used with different phase lags for encoding different targets.
- When the subject is presented with such phase-coded stimulations, by extracting phase information from the Fourier transform of the EEG signal and by comparing it to the phase of some reference signal (for example, the phase of the EEG response for a stimulus with zero phase lag), one can detect the target the subject is looking at.

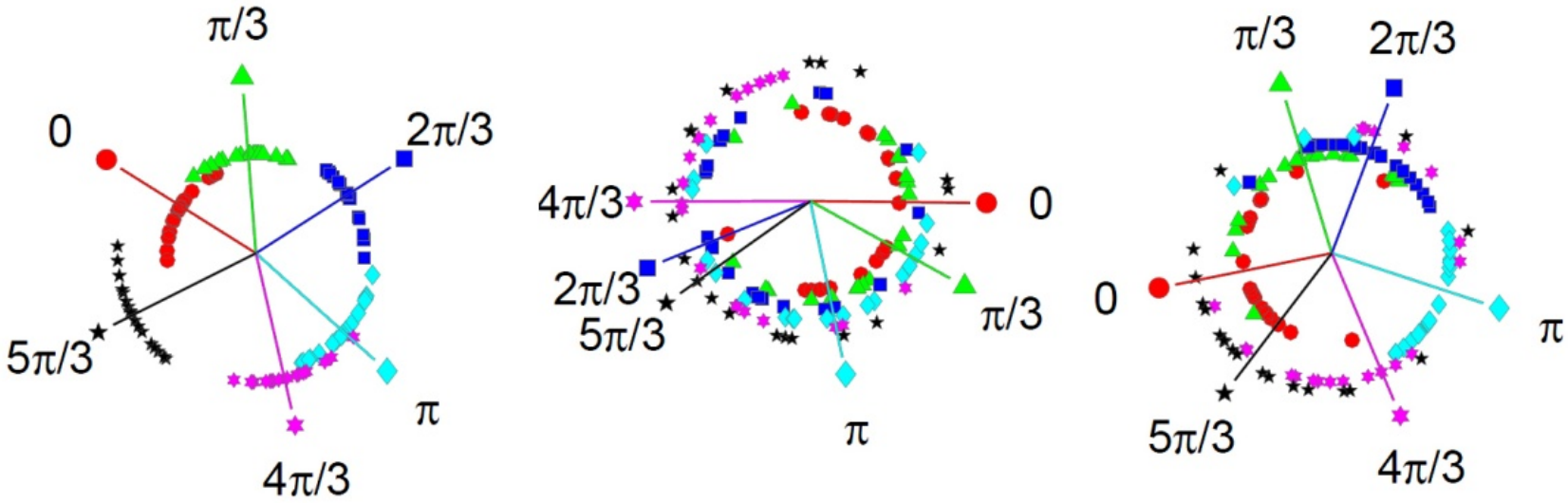
EEG BCI Data Acquisition

- The phases, which are used as the MLMVN inputs, are extracted from the Fourier transform of the EEG:

$$\varphi_d = \arg\left(\sum_t s_d(t)\cos(2\pi nft) + i\sum_t s_d(t)\sin(2\pi nft) \right)$$

- d is the number of a channel, n should be taken just equal to 1, so only the principal harmonic is considered
- Then $e^{i\varphi_d}, d = 1,\ldots,N$ are MLMVN inputs

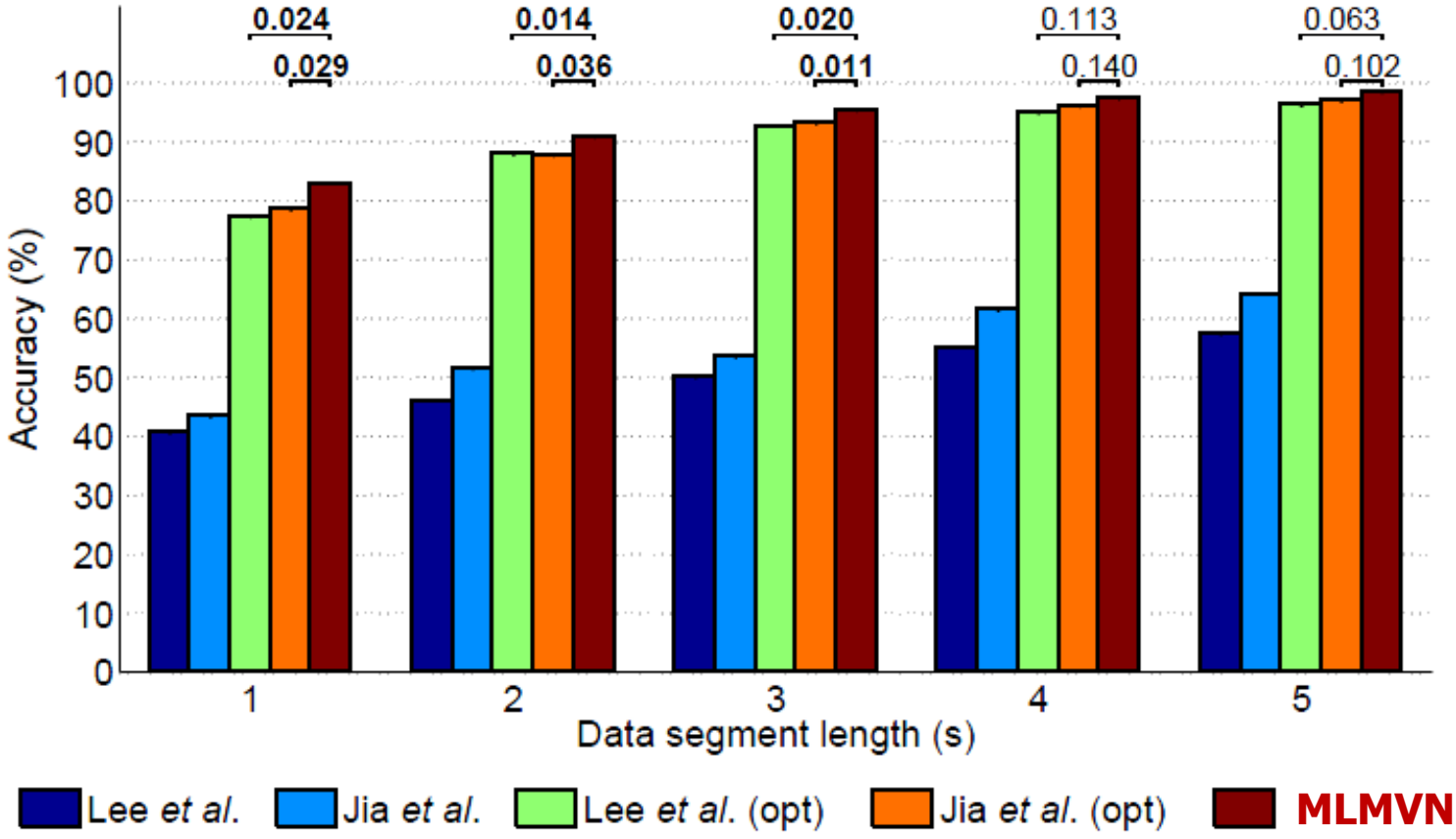
EEG BCI Data Acquisition

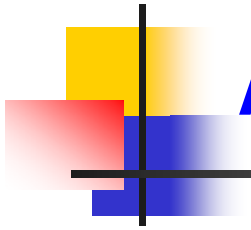


Distribution of estimated phases (expressed as angular values) for subject 1 in the experiment described in Sec. 1.3.2 for 3 different channels: Oz (left), POz–Oz (central) and POz–O2 (right).
Each dot corresponds to the phase estimated from a one second interval recorded when the subject was observing a particular phase-shifted stimulus.
Colors represent target-classes

Results and Comparison to Commonly used Methods

5-fold cross-validation

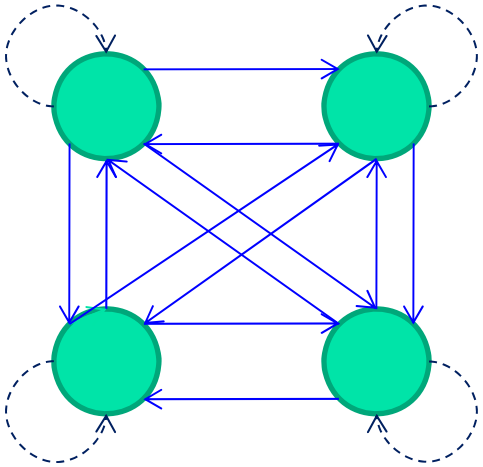




Associative Memory

- The most popular model of the neural associative memory is the Hopfield'd model suggested in 1982. In this model, each neuron learns some small fragment of the data. The output of each neuron is connected to the corresponding input of all the other neurons. Thus, all the neurons in the network are fully-interconnected.

Hopfield and Hopfield-like memories



- 2x2 2D memory
- Each neuron is connected to all the others
- If feedback connections are allowed, then the memory (the neural network) is recurrent

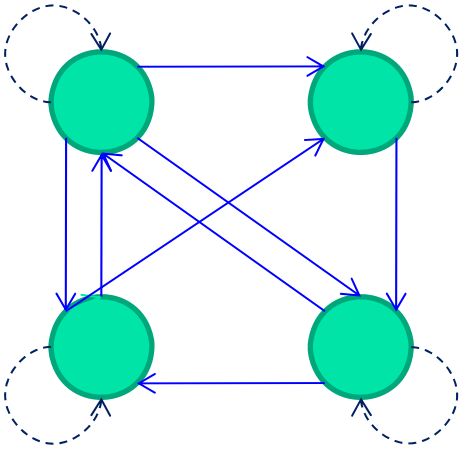
MVN-based Hopfield memory

- 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.
- 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.
- 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 2085, (J. Mira, A. Prieto - Eds.), Springer, **2001**, pp. 369-374
- MVN-based Hopfield memory storage capacity and its ability to retrieve the stored information is much higher than the ones for the standard Hopfield's associative memory

MVN-based associative memory with random connections

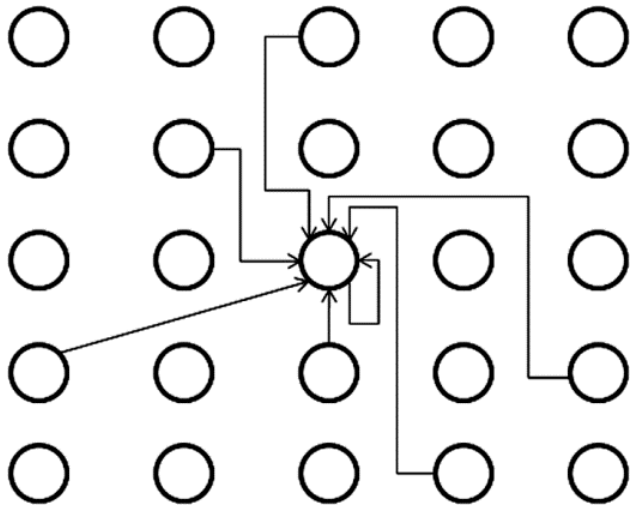
- I. Aizenberg, "*Complex-Valued Neural Networks with Multi-Valued Neurons*", Springer, Heidelberg, **2011**.
- 2D memory for storage of the gray-scale images contains the same number of neurons as the number of pixels in the images to be stored.
- Inputs of each neuron are connected with the outputs of **just 1% of other neurons**, a feedback connection is also presented

MVN-based associative memory with random connections



- Example:
  - The 2x2 2D memory
  - Inputs of each neuron are connected to the outputs of the two randomly chosen other neurons.
  - Feedback connections are allowed

MVN-based associative memory with random connections



➢ Example:

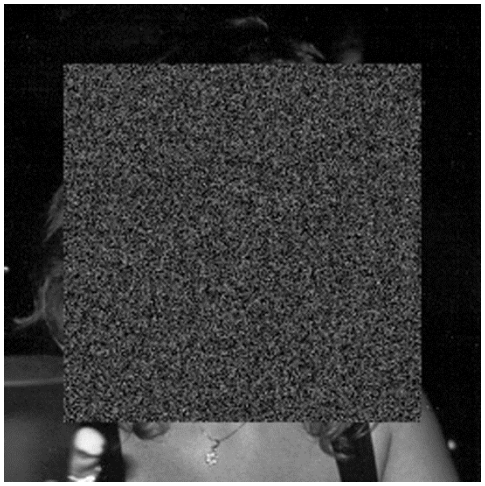
- The 5x5 2D memory
- Inputs of each neuron are connected to the outputs of the 5 randomly chosen other neurons.
- Feedback connections are allowed

MVN-based associative memory with random connections

- This network can store up to NM^n of NxM gray-scale images, where n is the number of the neuron's inputs
- All NxM neurons of this network should be trained using the error-correction learning algorithm
- This network can easily retrieve a stored pattern even when 75% of that pattern is lost and completely replaced by noise

MVN-based associative memory with random connections

Original 256x256 image "Alenka"

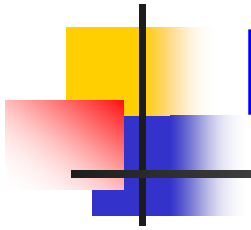


Original image with only 25% of information preserved while 75% of pixels contain just Gaussian noise

Image restored after 40 iterations using the MVN-based 256x256 network where each neuron is connected to 59 other neurons and to itself

Image restored after 80 iterations using the MVN-based 256x256 network where each neuron is connected to 59 other neurons and to itself

MLMVN with Soft Margins Learning



- Will be presented in detail at the IJCNN-2013 special session (S09) on Complex-Valued Neural Networks, Tuesday, August 6th, Continental room, 5-20pm

Prospective Directions

- Modification of the MVN discrete activation function for better solving unbalanced classification problems
- Further development of batch learning techniques
- More intensive use of MVN with a periodic activation function in networks
- Intelligent image processing
- Modeling of complex hybrid dynamical systems depending on many parameters using MLMVN
- Simulation of a biological neuron
- Etc. …

CONCLUSION

The best results are still ahead ☺