

Bogdan M. Wilamowski

Motivation for the topic of the seminar

Constrains:

- Not to talk about AMNSTC (nano-micro)
- Bring a new perspective to students
- Keep it to the state-of-the-art

Following topics were considered:

- ✓ Solving Engineering Problems with Computer
- ✓ Advanced Network Programming
- ✓ Analog Signal Processing
- ✓ Computational Intelligence

Keynotes:

AINA'10 – 24-th Conf. on Advanced Information Networking and Applications, Perth, Australia (April)
ICAISC'10 - 10th International Conference on Artificial Intelligence and Soft Computing, Zakopane, Poland (June)
ICIT'10 – 2-nd International Conf. on Information Technology , Gdansk, Poland (June)
ISIE'10 – 19-th International Symposium on Industrial Electronics, Bari, Italy (July)
ISRCS'10 – 3-rd International Symposium on Resilient Control Systems, Idaho Falls, USA (August)

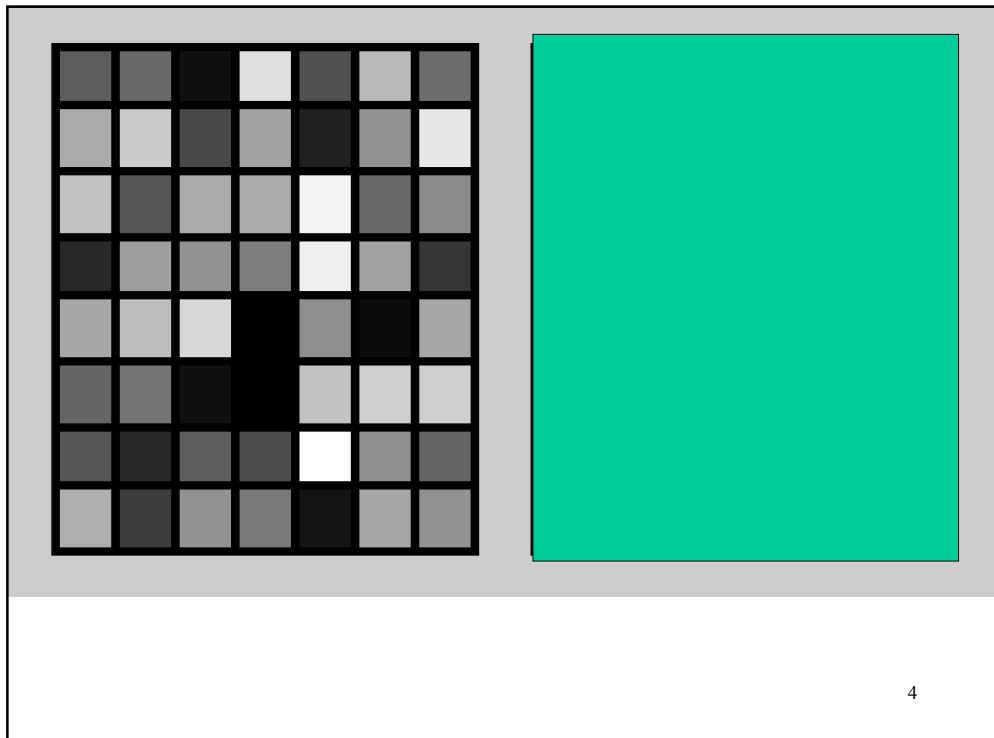
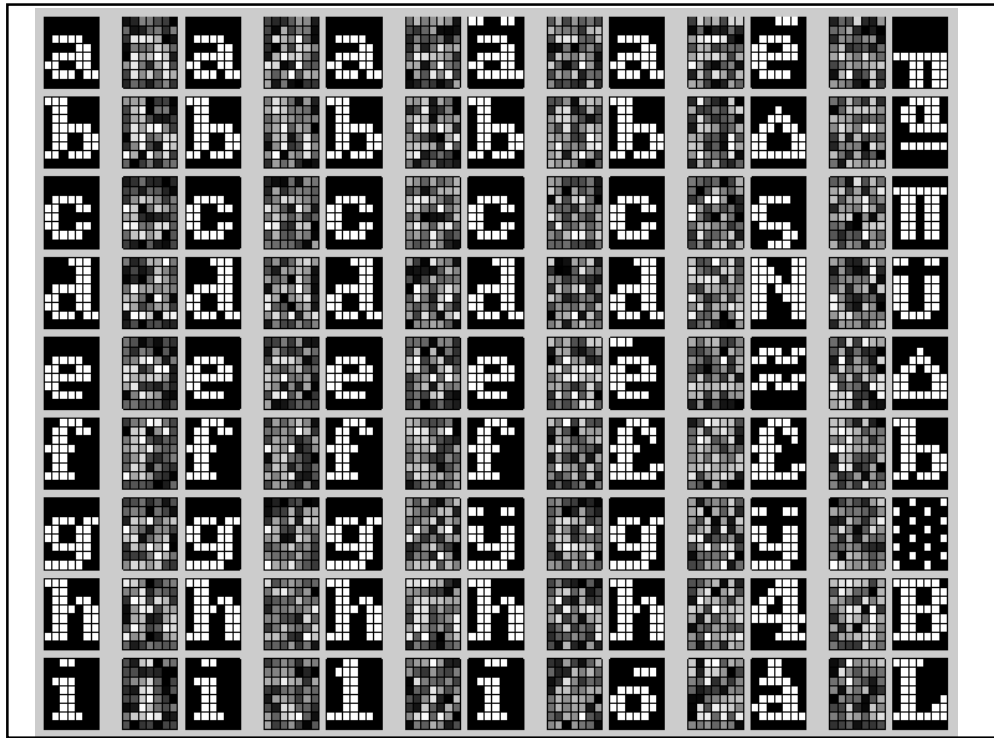
1

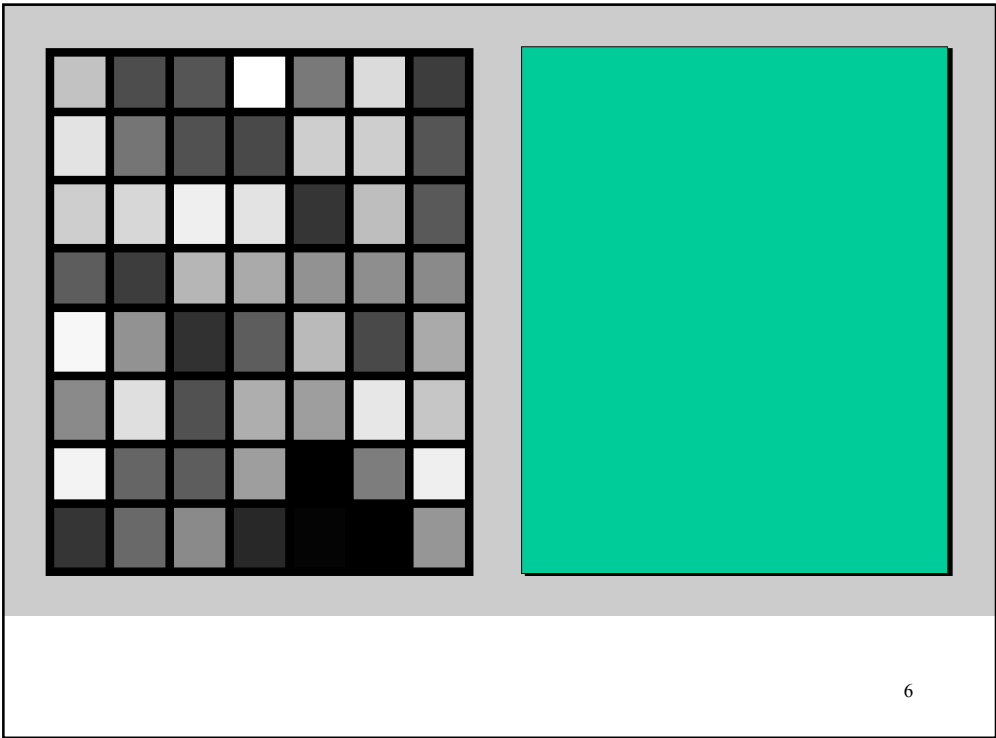
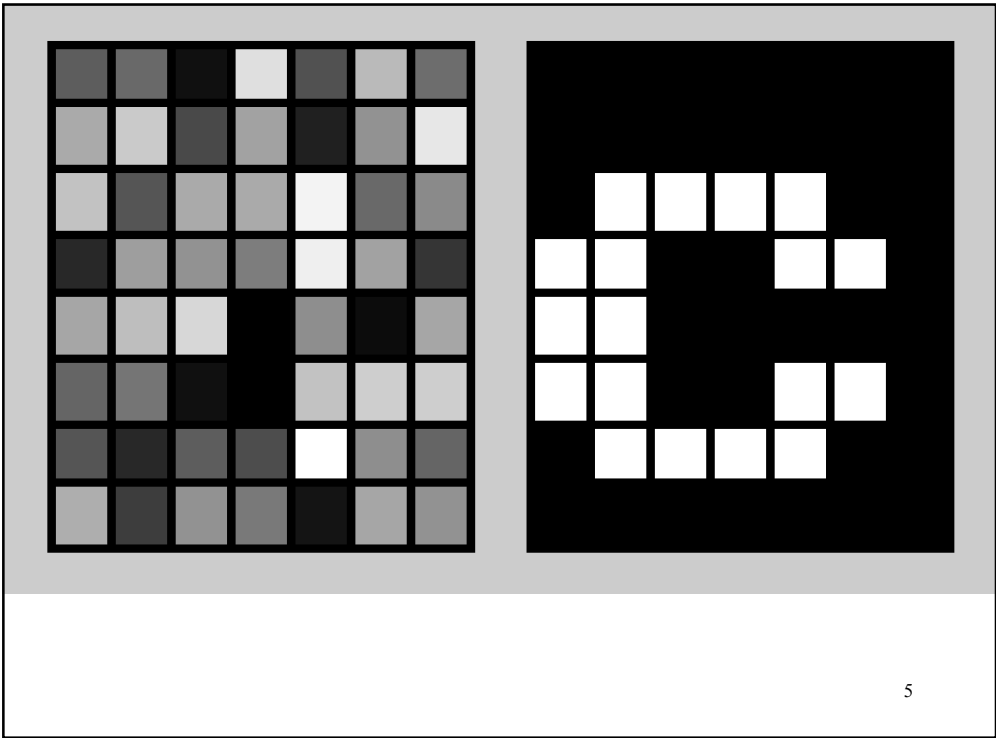
Bogdan M. Wilamowski

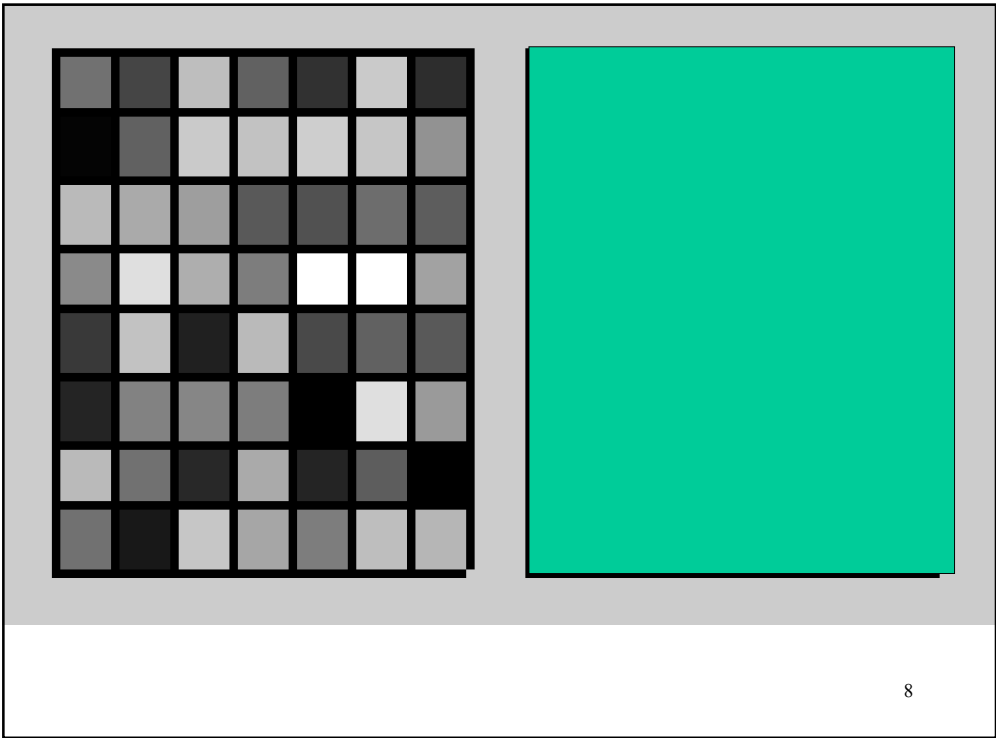
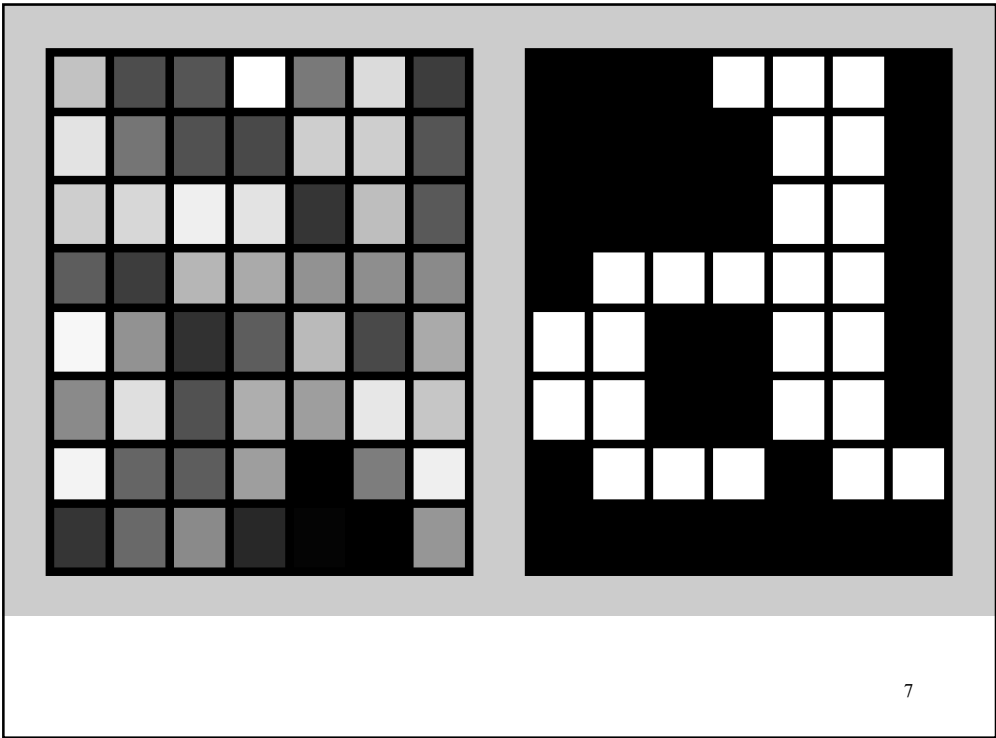
Problems with computational intelligence

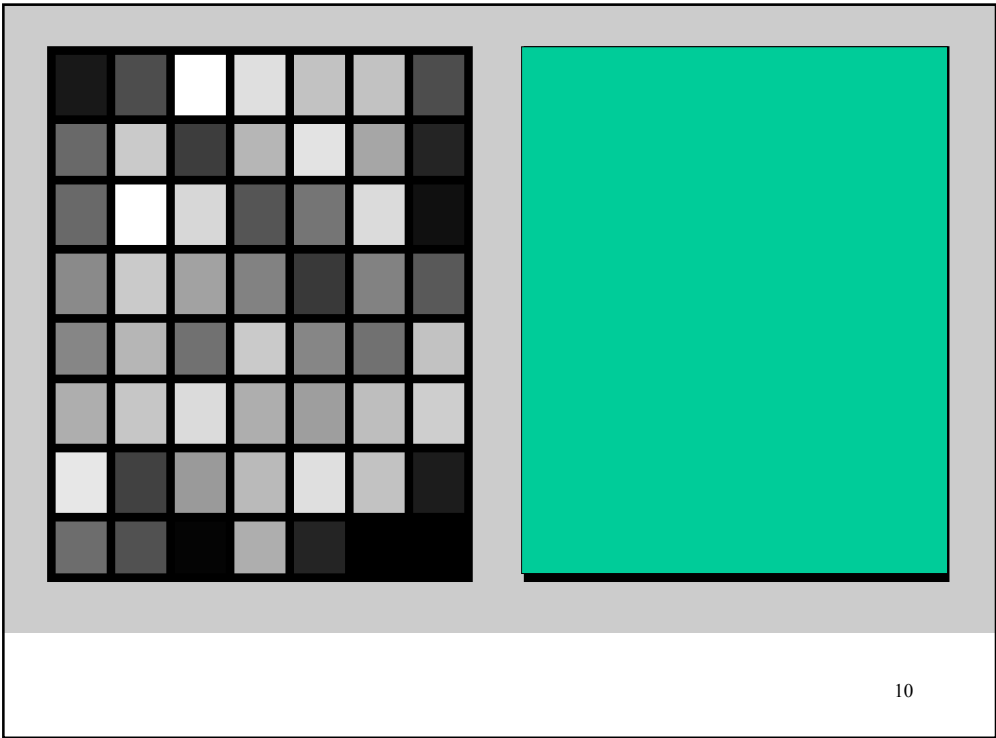
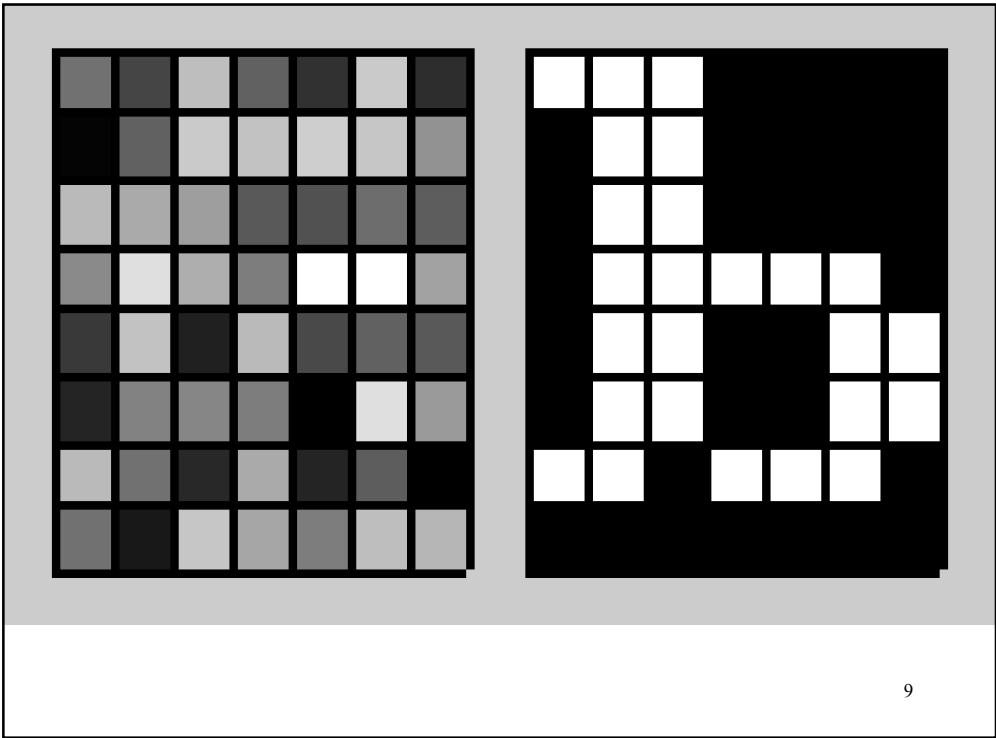
- Introduction
- Neural Network Learning
- Neural Networks Architectures
- Challenges in Neural Networks
- Fuzzy Systems
- Comparison of Neural and Fuzzy Systems
- Evolutionary Computation

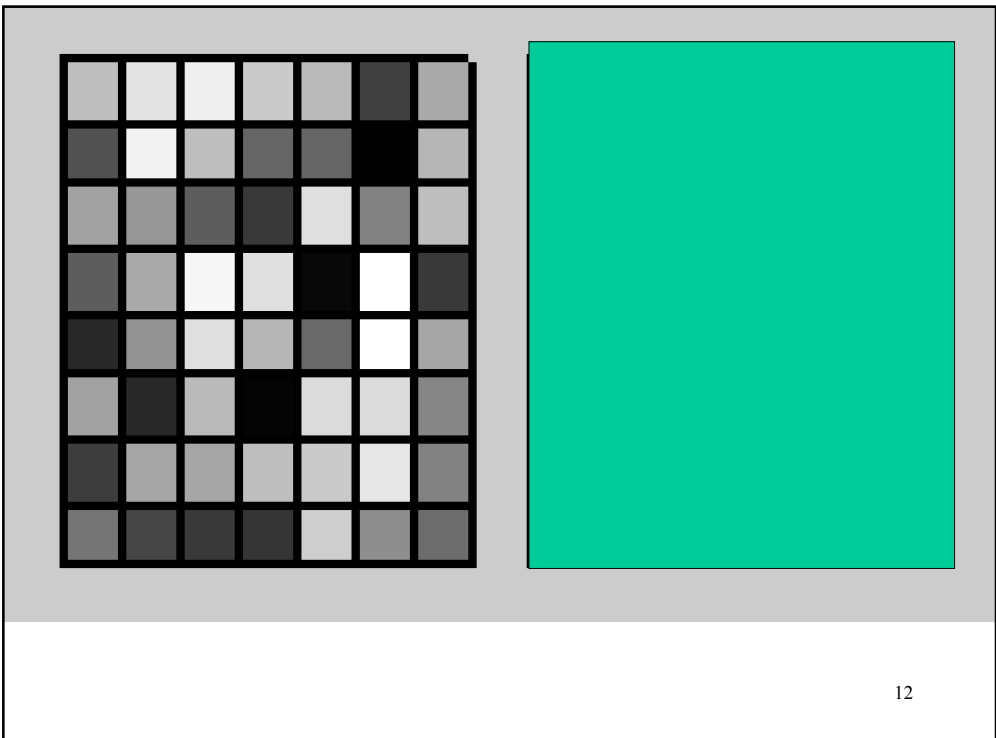
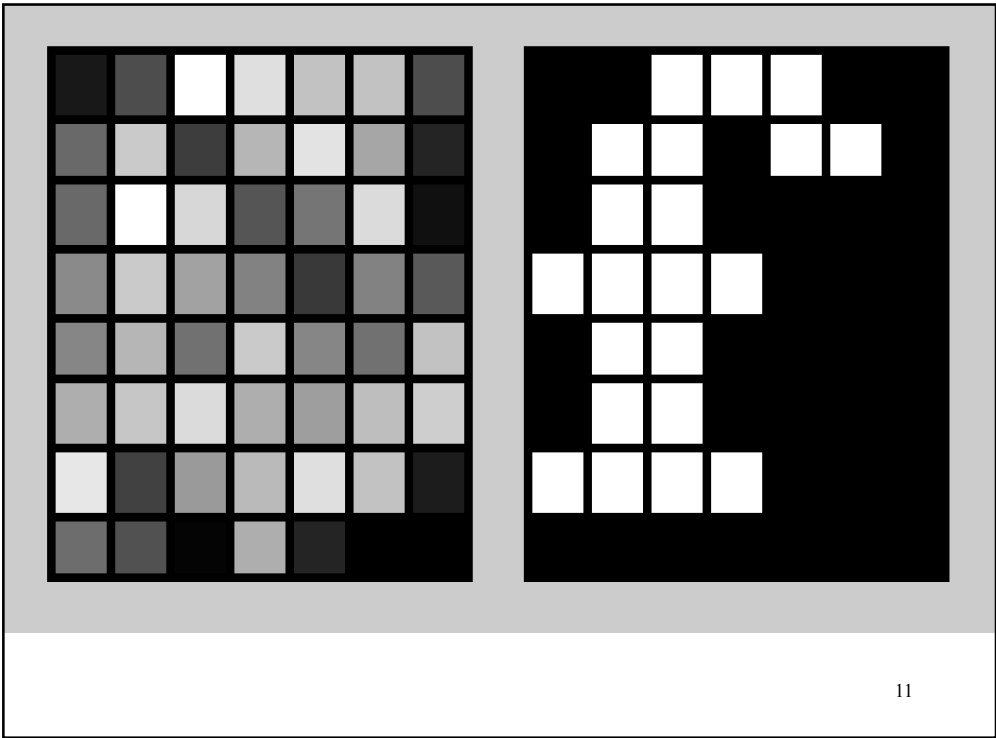
2

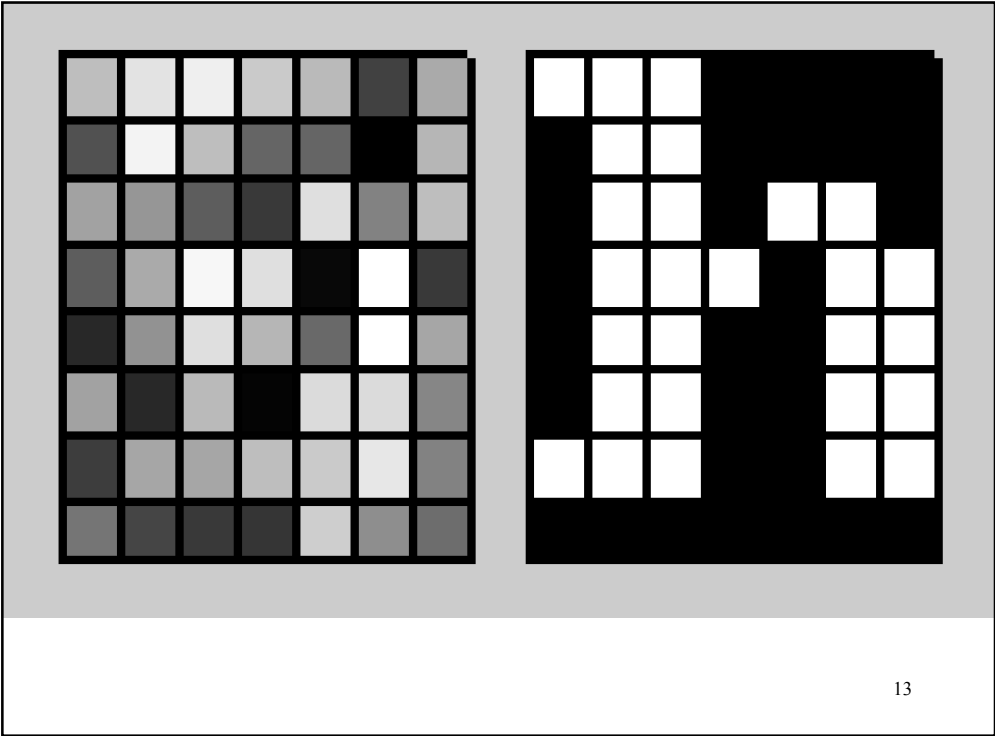




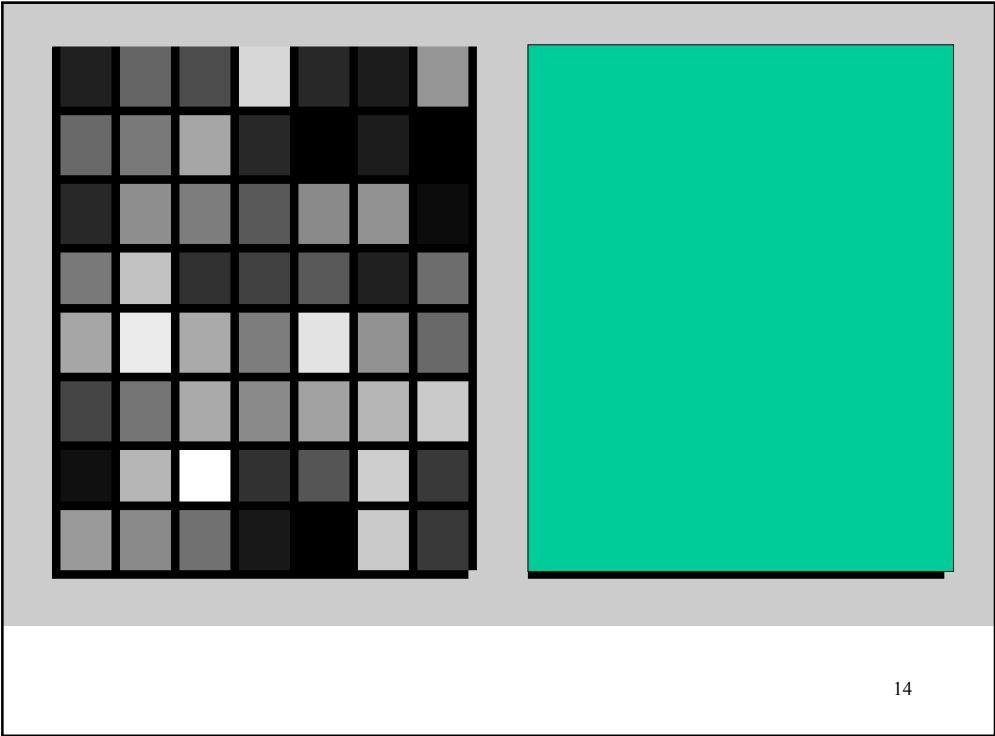




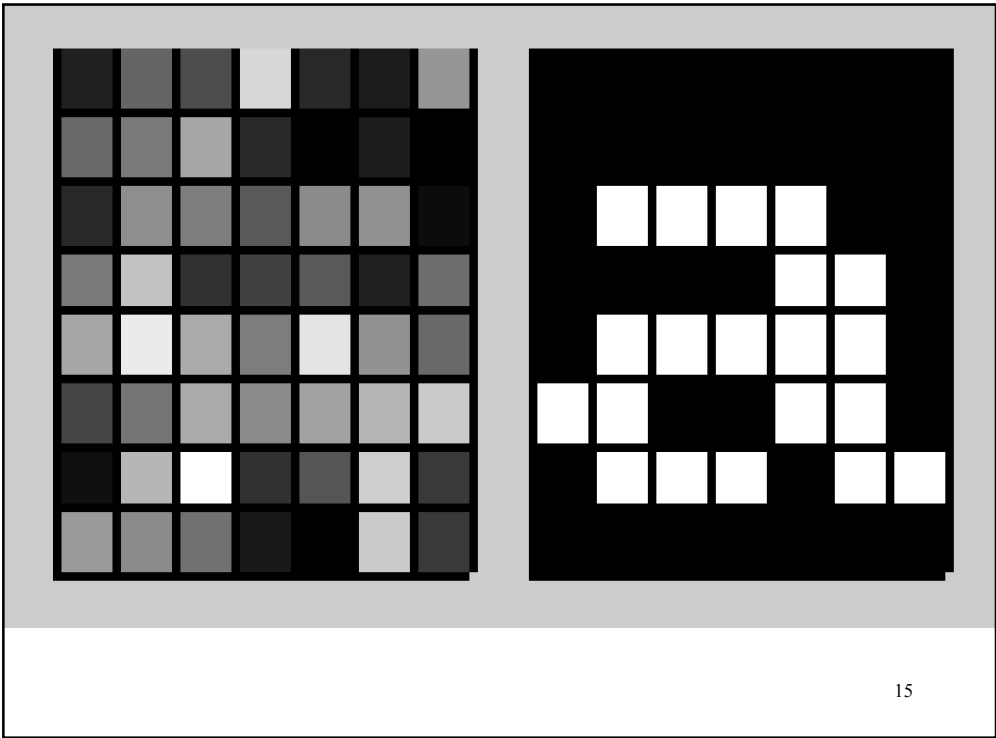




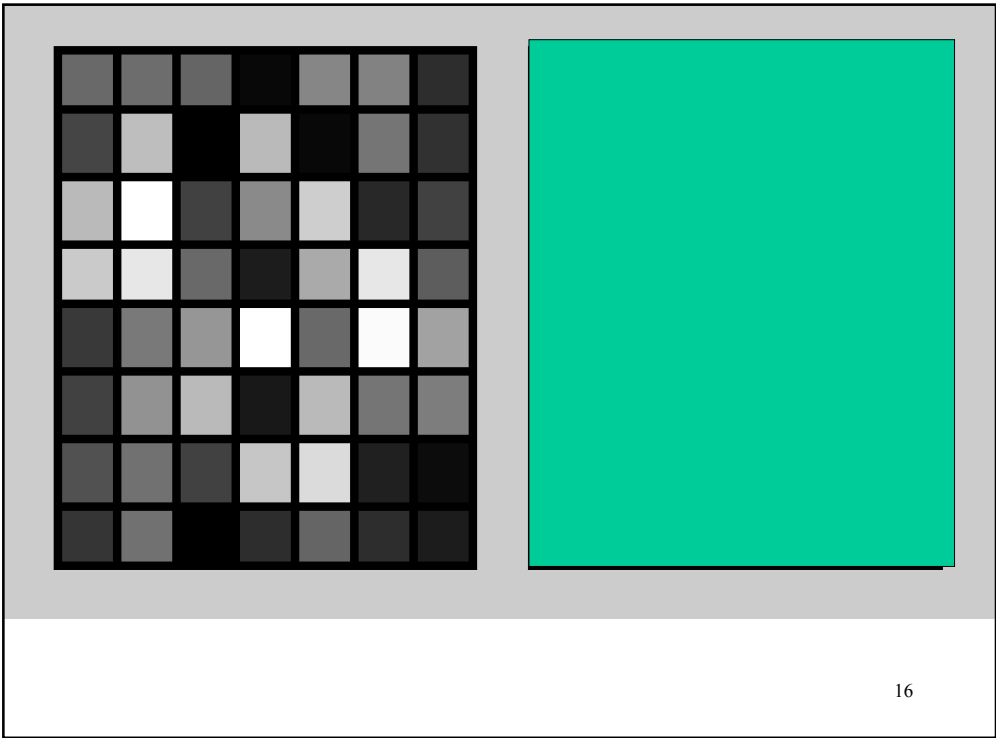
13



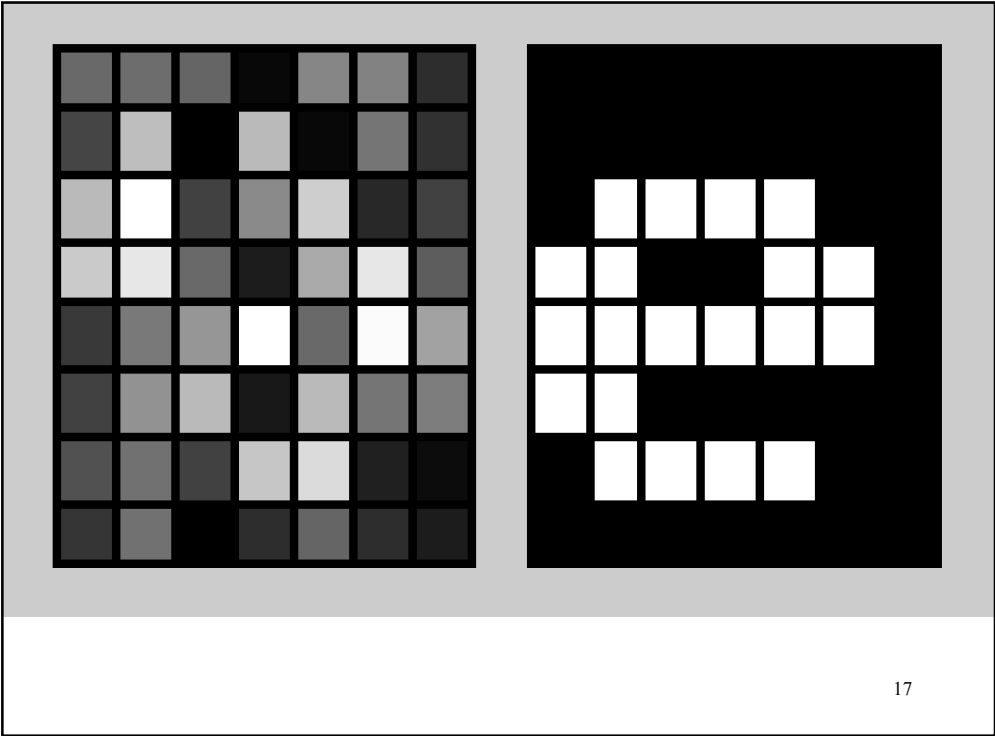
14



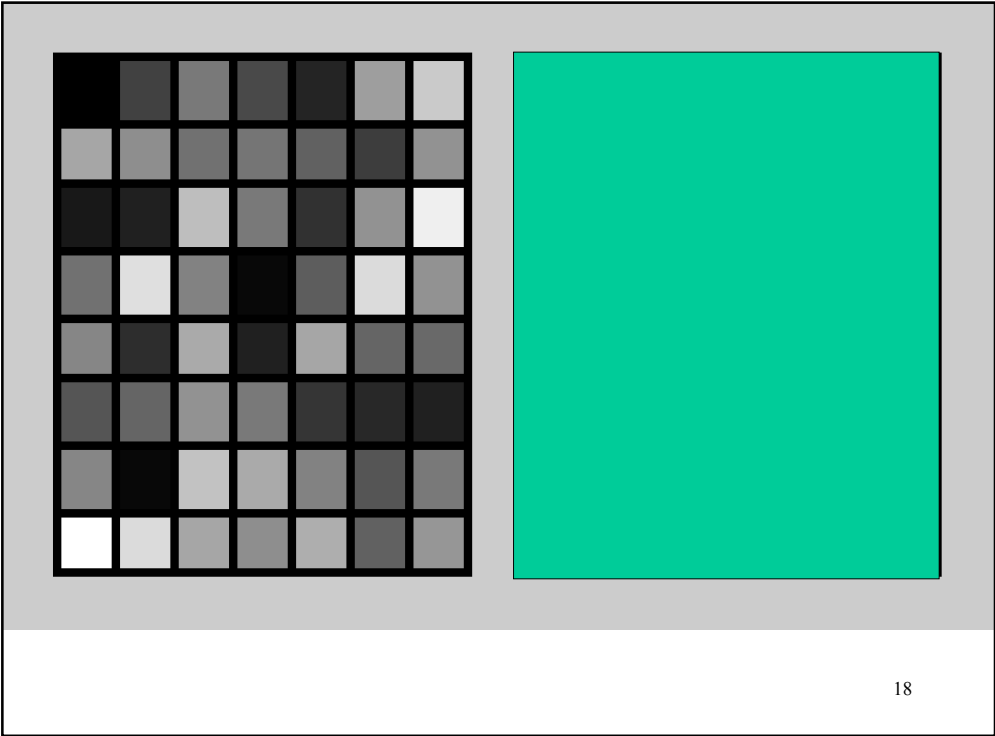
15



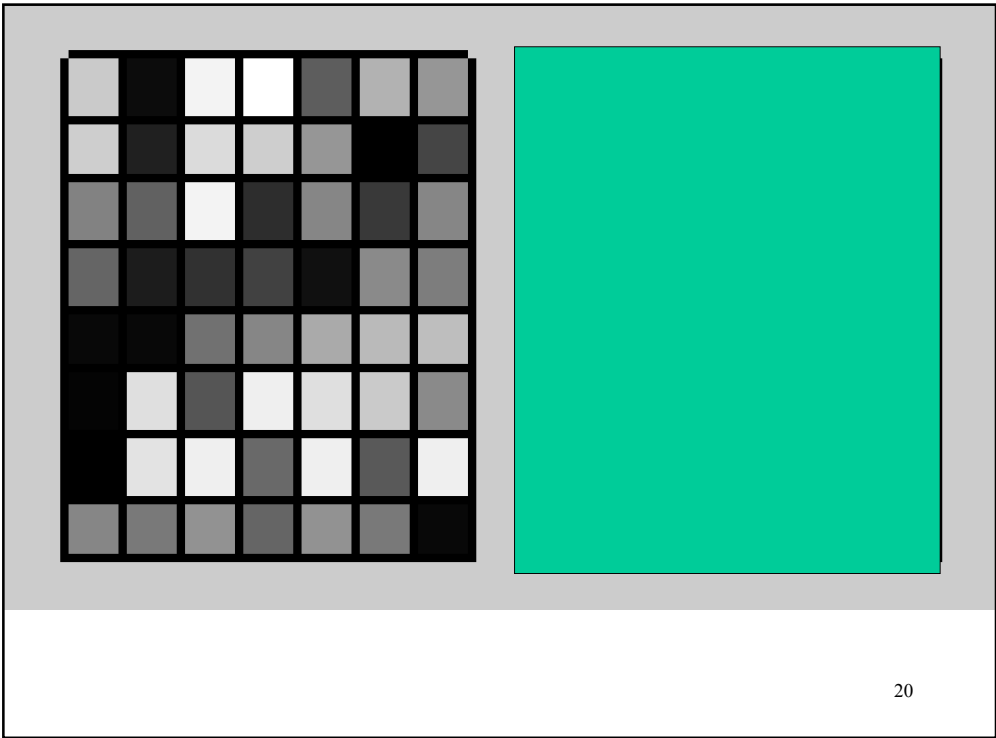
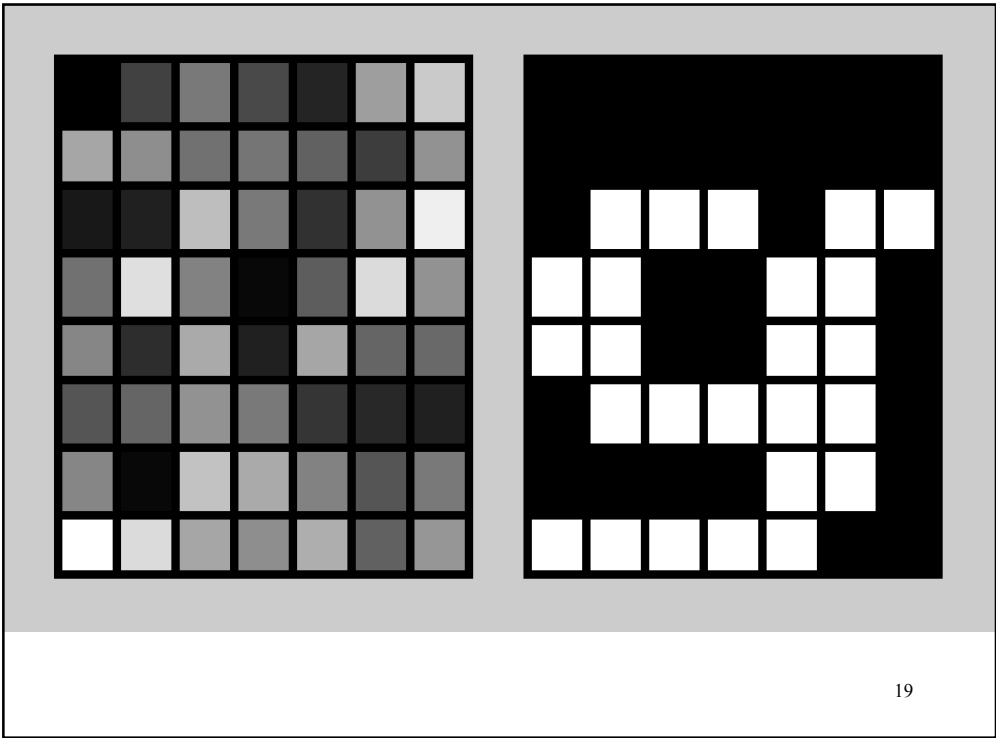
16

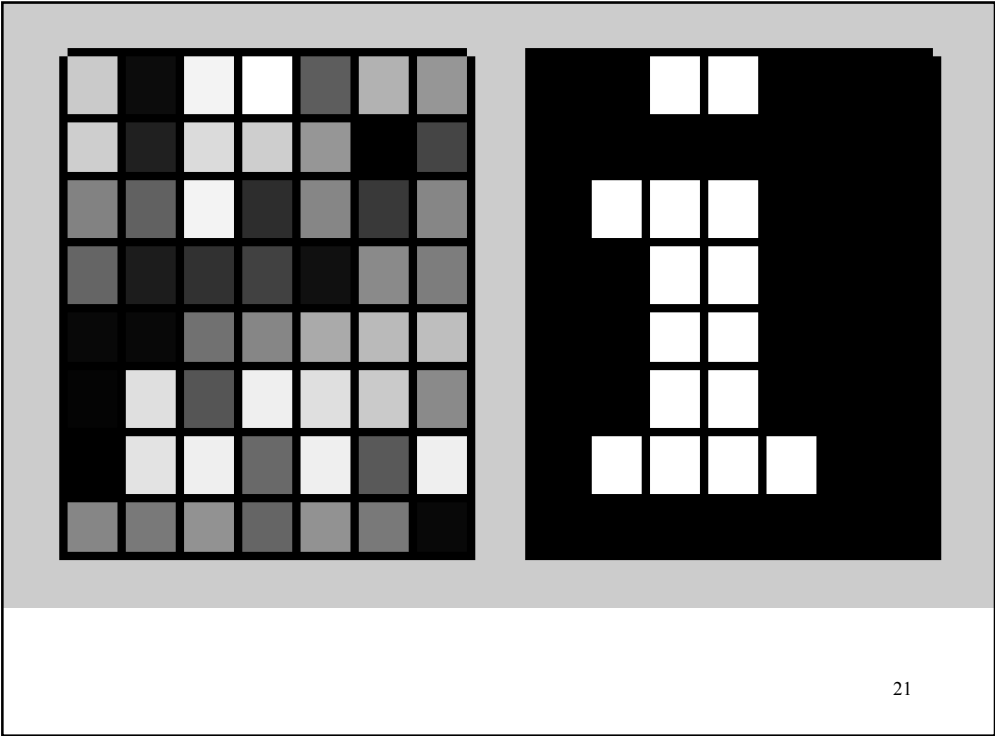


17

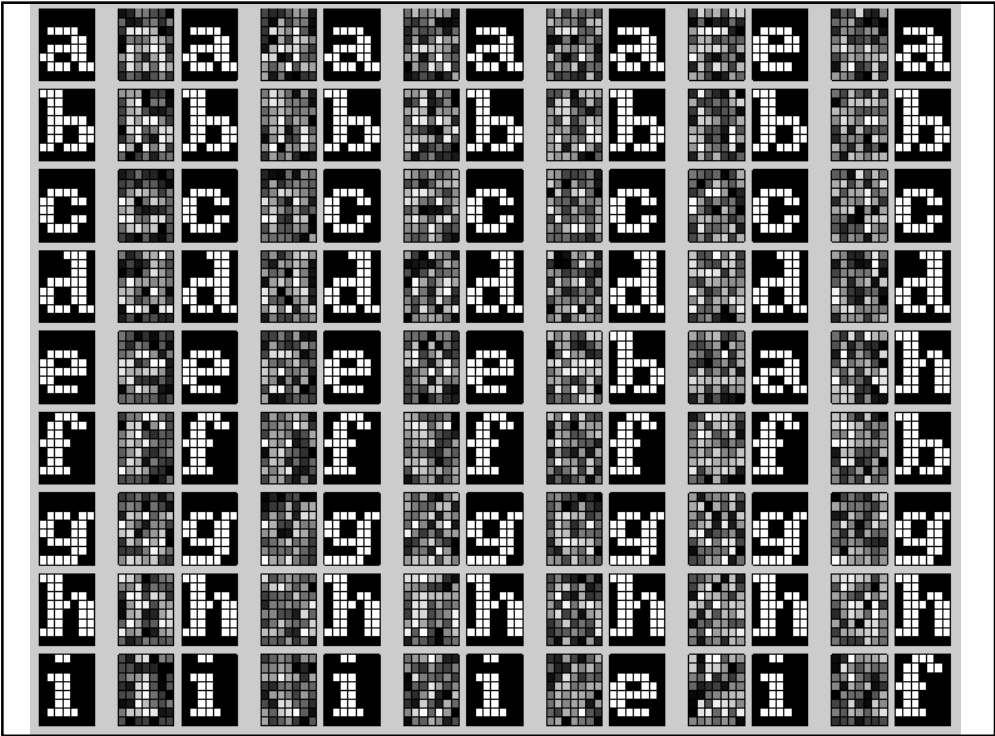


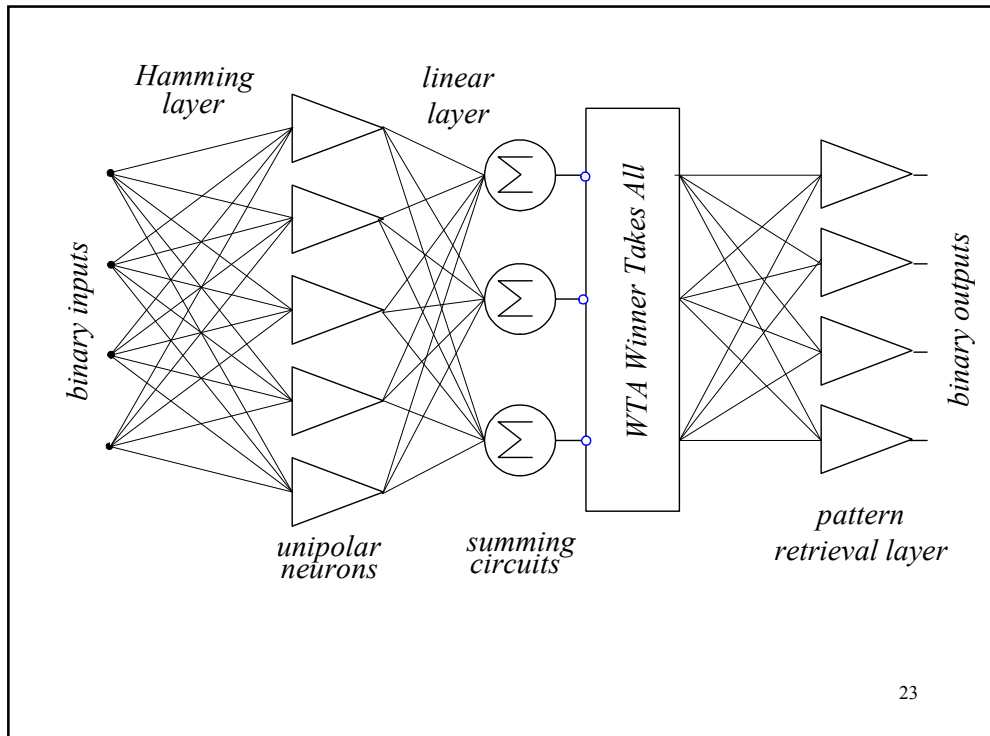
18





21





The conclusion:

The system of computational intelligence can be smarter than humans

Is this new technological revolution?

- 150 years ago man power was replaced by machines (steam and electric)
- 20 years ago significant portion of man brain functions were replaced by computer (*calculations, administrative functions, voice and image recognitions etc*)
- We are still claiming that we are the most intelligent creatures in the universe, but for how much longer?

24

Find clusters

Find number of clusters and its location in 4-dim. space

4	-3	4	7	-5	6	6	-3	-4	4	5	-2	8	4	-4	3
4	-3	4	8	7	4	-3	4	5	-3	5	7	6	6	-5	2
2	-5	3	6	-3	6	5	-3	4	-4	3	8	-5	6	7	-2
4	-4	6	7	-5	6	8	-1	3	-6	5	8	7	6	-5	4
-4	5	6	-4	3	-5	6	7	4	-3	5	6	3	-4	5	7
9	6	-3	4	-6	4	7	-1	9	7	-3	2	5	-4	6	9
2	-5	3	9	3	-5	6	8	3	-5	4	6	4	-4	4	6
3	-3	5	6	-4	4	5	-4	-4	6	6	-2	-5	6	7	-1
5	-4	6	8	4	-5	5	7	-5	4	7	-3	9	5	-4	2
7	7	-2	3	-5	5	6	-2	7	4	-2	4	-5	7	5	-1
-5	5	6	-3	9	7	-2	3	9	6	-4	4	-5	5	7	-2
9	6	-5	3	7	6	-4	2	-6	7	5	-1	8	5	-2	3
3	-5	6	8	8	5	-3	4	8	6	-3	3	7	6	-5	5
3	-5	3	8	-5	4	7	-4	-4	7	7	-2	-4	6	7	-3
4	-6	5	7	9	5	-4	2	8	5	-4	4	6	5	-3	5
-5	6	5	-4	8	6	-2	3	-4	5	8	-1	-5	6	5	-3
3	-3	6	7	-3	5	7	-2	-3	4	7	-2	8	4	-2	4
6	5	-5	4	3	-6	3	7	-4	5	6	-2	-4	5	7	-4
-5	7	7	-3	4	-5	4	9								

25

Adding neurons as needed using minimum distance concept

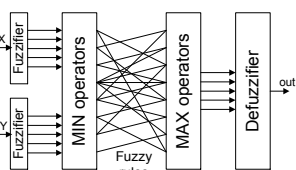
much simpler and more efficient than ART

1. First pattern is applied and the first neuron is introduced
2. Next pattern is applied and then:
 - a) If distance from all existing clusters is larger than threshold then a new neuron is added
 - b) Else weights of the closest neuron are updated

$$\mathbf{W}_k = \frac{m\mathbf{W}_k + \alpha\mathbf{X}}{m+1}$$

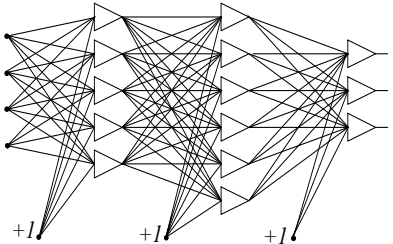
where m is the number of previous patterns of a given set which were used to update this particular neuron and α is the learning constant

26



Fuzzy systems

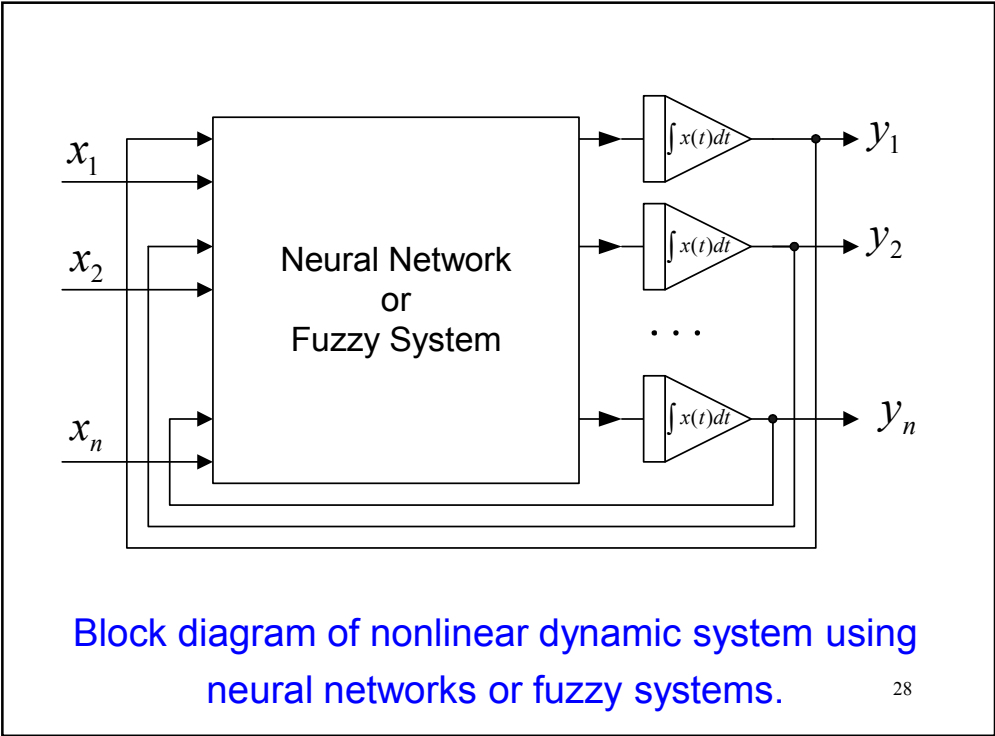
Practical Examples



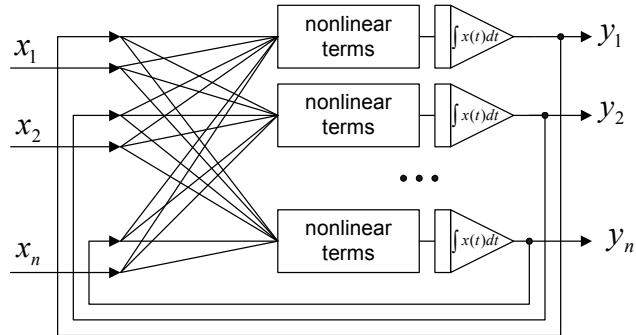
Neurl Networks

- **Fuzzy controllers**
 - used from cameras to elevators
- **Neural networks**
 - Diagnostic (medical, mechanical etc)
 - Modeling (natural phenomena, or complex systems)
 - Business and Military (various predictions)
- **Evolutionary computation**
 - Are they replacing design processes?

27



Introduction



$$y_1 = \int f_1(x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_n) dt$$

$$y_2 = \int f_2(x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_n) dt$$

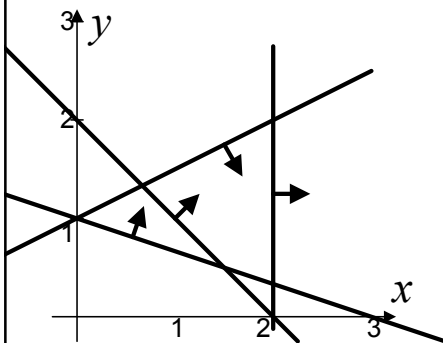
...

$$y_n = \int f_n(x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_n) dt$$

Block diagram of an arbitrary nonlinear dynamic system. 29

Another area with 4 partitions

neuron equations:



$$\frac{x}{2} + \frac{y}{2} > 1 \quad x + y - 2 > 0$$

$$\frac{x}{3} + \frac{y}{1} > 1 \quad x + 3y - 3 > 0$$

$$\frac{x}{2} + \frac{y}{\infty} > 1 \quad x + 0y - 2 > 0$$

$$\frac{x}{-2} + \frac{y}{1} < 1 \quad x - 2y + 2 > 0$$

Weights in the first layer:

$$1 \ 1 \ -2; \quad 1 \ 3 \ -3;$$

Weights in the second layer:

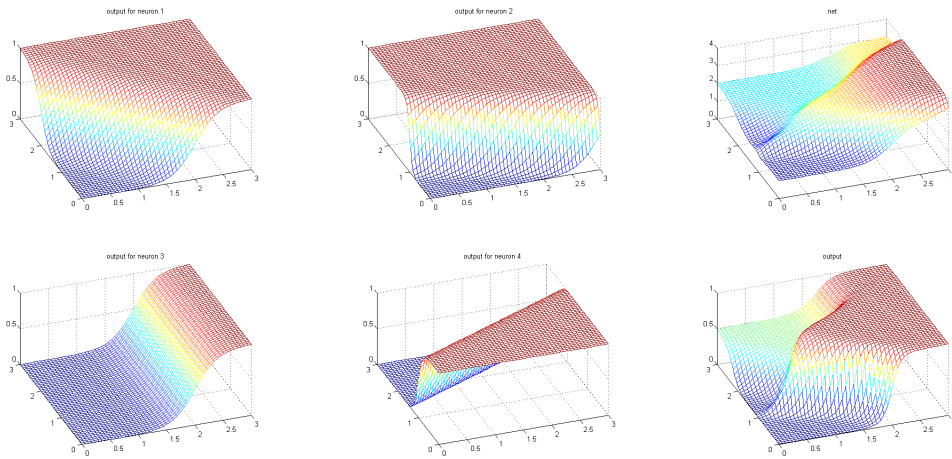
$$1 \ 0 \ -2; \quad 1 \ -2 \ 2;$$

$$1 \ 1 \ 1 \ 1 \ -2;$$

Another area with 4 partitions

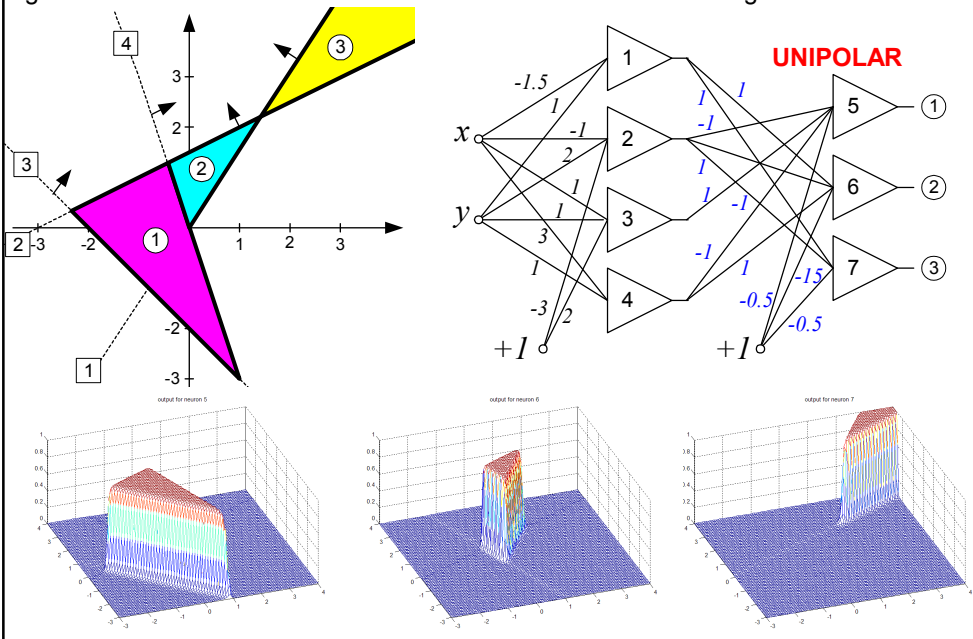
first layer

second layer



31

Design neural network with unipolar McCulloch -Pitts neurons, which has two input and three outputs. Each output respond to the patterns located in three areas as shown on figure below. Draw neural network and indicate value of each weight.

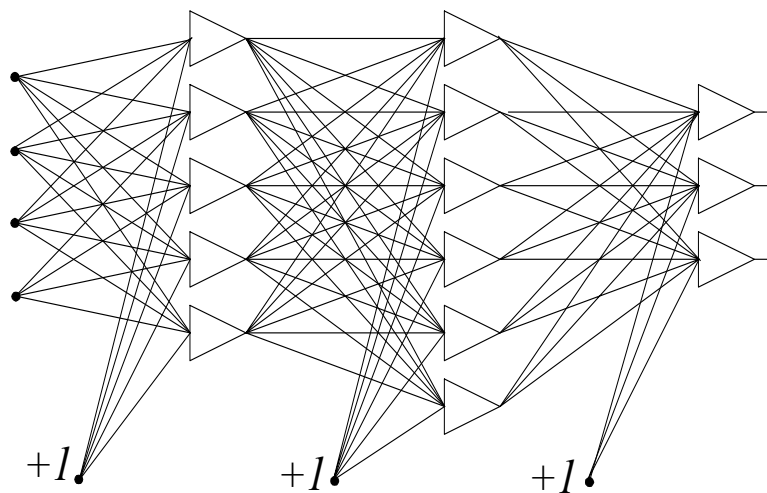


Problems with computational intelligence

- Introduction
- **Neural Network Learning**
- Neural Networks Architectures
- Fuzzy Systems
- Comparison of Neural and Fuzzy Systems
- Evolutionary Computation

33

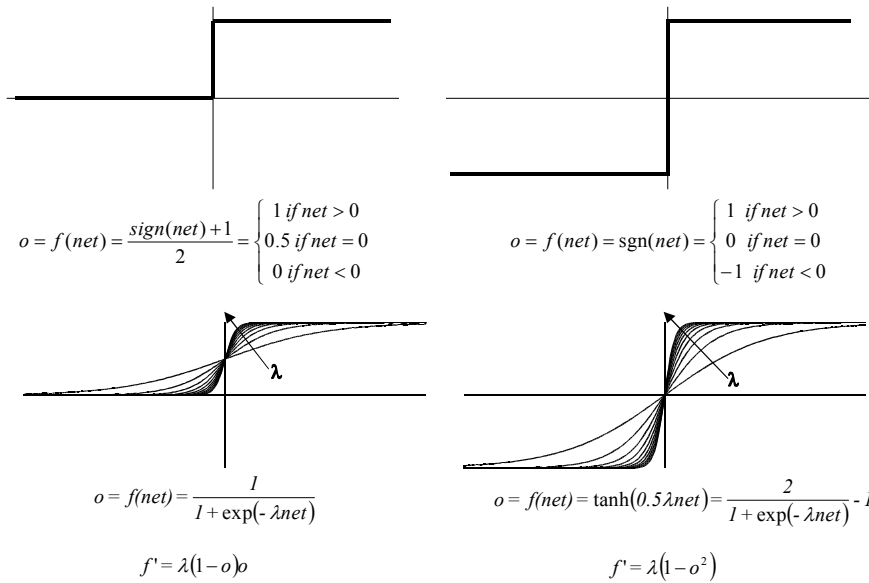
Neural networks as nonlinear elements



Feedforward neural networks

34

Soft activation functions



35

Neural Network Learning

Let us consider binary signals and weights such as

$$\mathbf{x} = +1 \quad -1 \quad -1 \quad +1 \quad -1 \quad +1 \quad +1 \quad -1$$

if weights $\mathbf{w} = \mathbf{x}$

$$\mathbf{w} = +1 \quad -1 \quad -1 \quad +1 \quad -1 \quad +1 \quad +1 \quad -1$$

then

$$net = \sum_{i=1}^n w_i x_i = 8$$

this is the maximum value net can have for any other combinations net would be smaller

36

Neural Network Learning

For the same pattern

$$\mathbf{x} = +1 \quad -1 \quad -1 \quad +1 \quad -1 \quad +1 \quad +1 \quad -1$$

and slightly different weights

$$\mathbf{w} = +1 \quad +1 \quad -1 \quad +1 \quad -1 \quad -1 \quad +1 \quad -1$$

$$net = \sum_{i=1}^n w_i x_i = 4$$

$$net = \sum_{i=1}^n w_i x_i = n - 2HD$$

HD is the
Hamming Distance

37

Supervised learning rules for single neuron

$$\Delta \mathbf{w}_i = c \delta \mathbf{x}$$

correlation rule (supervised): $\delta = d$

perceptron fixed rule: $\delta = d - o$

perceptron adjustable rule - as above but the learning constant is modified to:

$$\alpha^* = \alpha \lambda \frac{\mathbf{x} \mathbf{w}^T}{\mathbf{x} \mathbf{x}^T} = \alpha \lambda \frac{net}{\|\mathbf{x}\|^2}$$

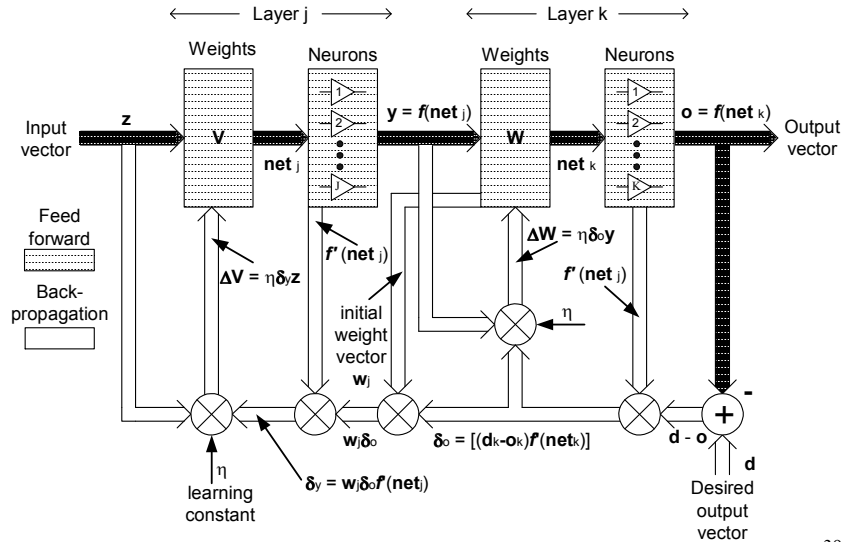
LMS (Widrow-Hoff) rule: $\delta = d - net$

delta rule: $\delta = (d - o) f'$

pseudoinverse rule (the same as LMS): $\mathbf{w} = (\mathbf{x}^T \mathbf{x})^{-1} \mathbf{x}^T d$

38

EBP Error Back Propagation algorithm for single pattern



EBP Error Back Propagation algorithm for single pattern

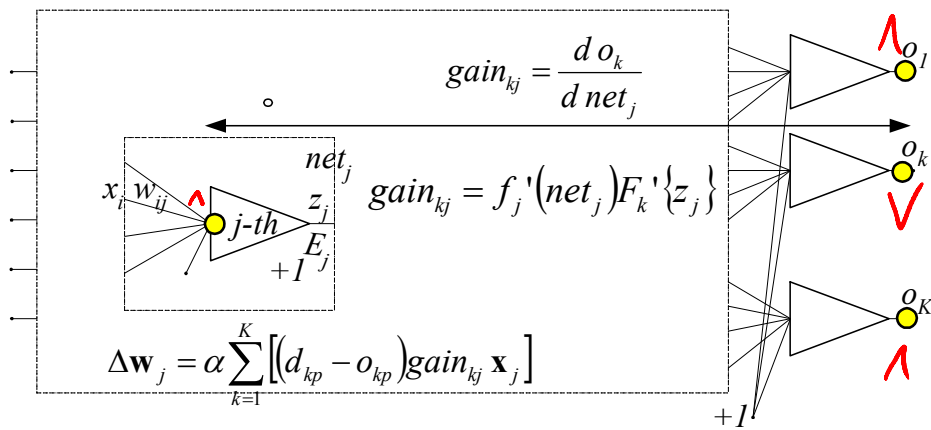


Illustration of the concept of the gain computation in neural networks

LM or NBN algorithm

Steepest descent method: $\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha \mathbf{g}$

Newton method: $\mathbf{w}_{k+1} = \mathbf{w}_k - \mathbf{A}_k^{-1} \mathbf{g}$

where \mathbf{A}_k is Hessian and \mathbf{g} is gradient vector

if error is defined as:
$$E = \sum_{p=1}^P \sum_{m=1}^M (d_{pm} - o_{pm})^2$$

then:

$$\mathbf{A} = 2\mathbf{J}^T \mathbf{J} \quad \text{and} \quad \mathbf{g} = 2\mathbf{J}^T \mathbf{e}$$

where \mathbf{J} is Jacobian and \mathbf{e} is error vector

$$\mathbf{J} = \begin{bmatrix} \frac{\partial e_{11}}{\partial w_1} & \frac{\partial e_{11}}{\partial w_2} & \dots & \frac{\partial e_{11}}{\partial w_n} \\ \frac{\partial e_{21}}{\partial w_1} & \frac{\partial e_{21}}{\partial w_2} & \dots & \frac{\partial e_{21}}{\partial w_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_{M1}}{\partial w_1} & \frac{\partial e_{M1}}{\partial w_2} & \dots & \frac{\partial e_{M1}}{\partial w_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_{1P}}{\partial w_1} & \frac{\partial e_{1P}}{\partial w_2} & \dots & \frac{\partial e_{1P}}{\partial w_n} \\ \frac{\partial e_{2P}}{\partial w_1} & \frac{\partial e_{2P}}{\partial w_2} & \dots & \frac{\partial e_{2P}}{\partial w_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_{MP}}{\partial w_1} & \frac{\partial e_{MP}}{\partial w_2} & \dots & \frac{\partial e_{MP}}{\partial w_n} \end{bmatrix}$$

$$\begin{array}{cccc} \frac{\partial E}{\partial w_1} & \frac{\partial^2 E}{\partial w_1^2} & \frac{\partial^2 E}{\partial w_2 \partial w_1} & \dots & \frac{\partial^2 E}{\partial w_n \partial w_1} \\ \frac{\partial E}{\partial w_2} & \frac{\partial^2 E}{\partial w_1 \partial w_2} & \frac{\partial^2 E}{\partial w_2^2} & \dots & \frac{\partial^2 E}{\partial w_n \partial w_2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial E}{\partial w_n} & \frac{\partial^2 E}{\partial w_1 \partial w_n} & \frac{\partial^2 E}{\partial w_2 \partial w_n} & \dots & \frac{\partial^2 E}{\partial w_n^2} \end{array}$$

Jacobian

Gradient

Hessian

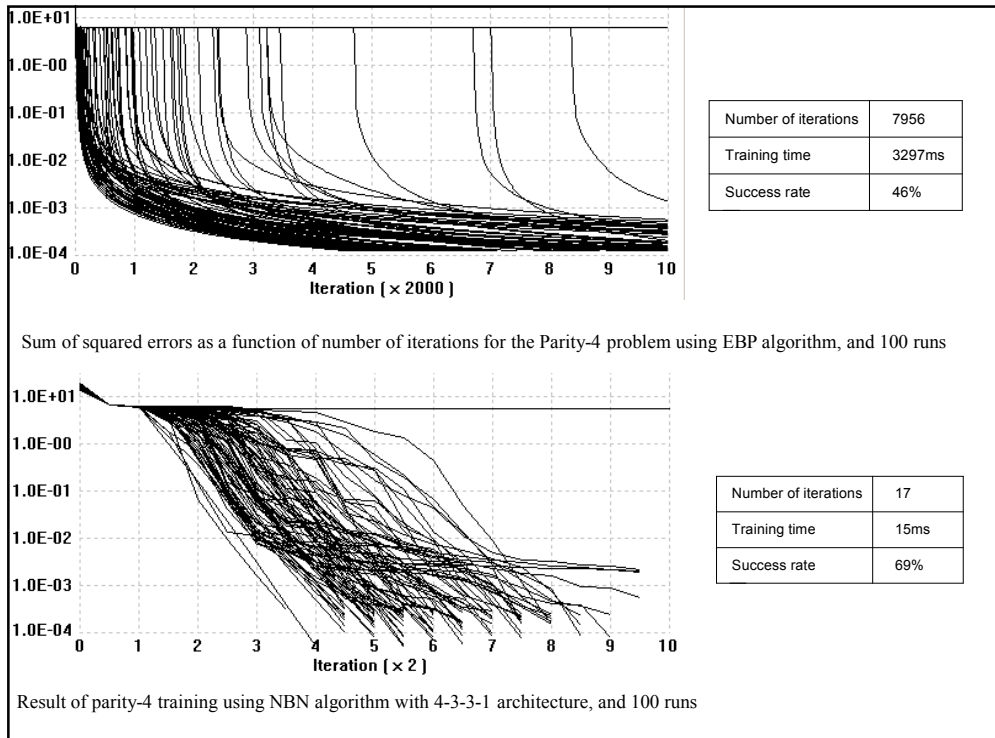
N is number of weights

M is number of outputs

41

Advantages of NBN algorithm over LM algorithm

- Both LM and NBN are very fast
- NBN do not calculate Jacobian matrix so it can handle problems with basically unlimited number of patterns, while LM can be used only for small problems
- LM needs the forward and back propagation processes to calculate Jacobian, while NBN uses only forward calculation so it is faster (especially for networks with multiple outputs)
- NBN can handle arbitrarily connected neural networks, while LM was developed only for MLP

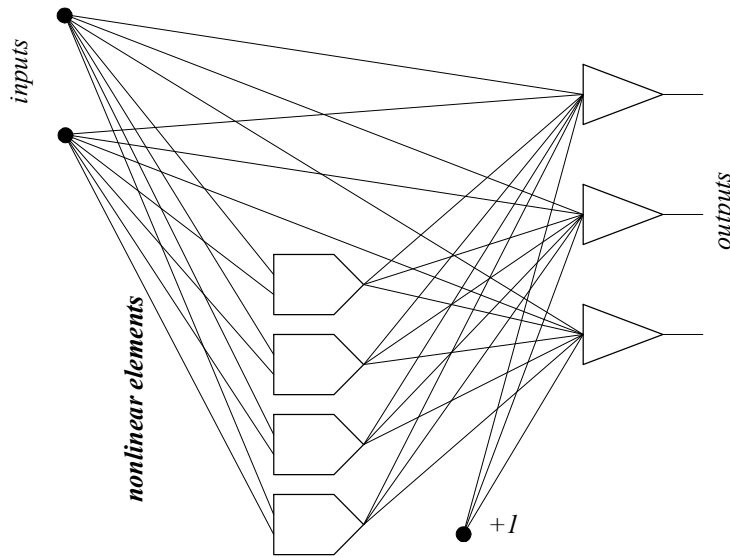


Bogdan M. Wilamowski

Problems with computational intelligence

- Introduction
- Neural Network Learning
- **Neural Networks Architectures**
- Fuzzy Systems
- Comparison of Neural and Fuzzy Systems
- Evolutionary Computation

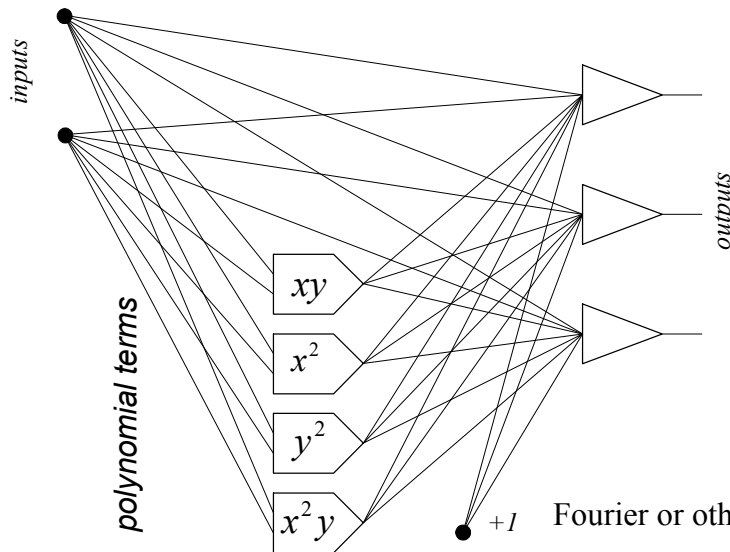
Functional Link Networks



Genetic algorithms?

45

Polynomial Networks

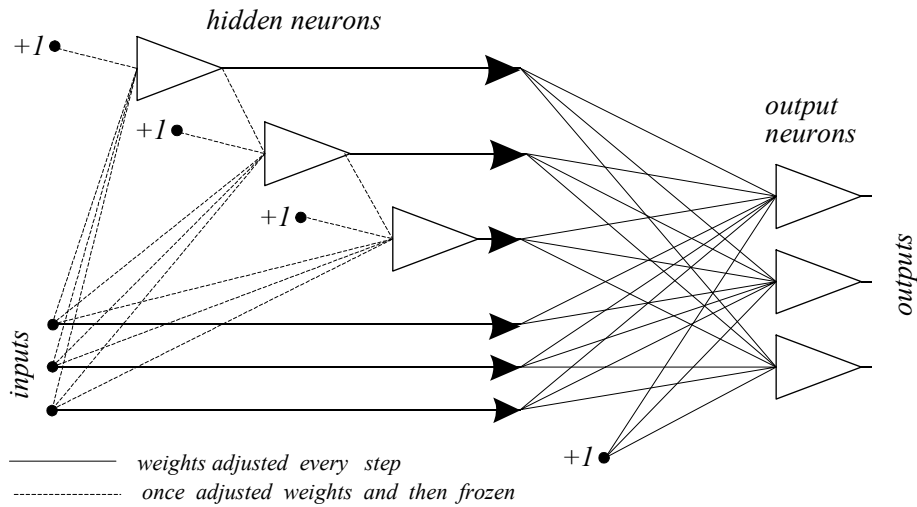


Fourier or other series?

Nonlinear regression?

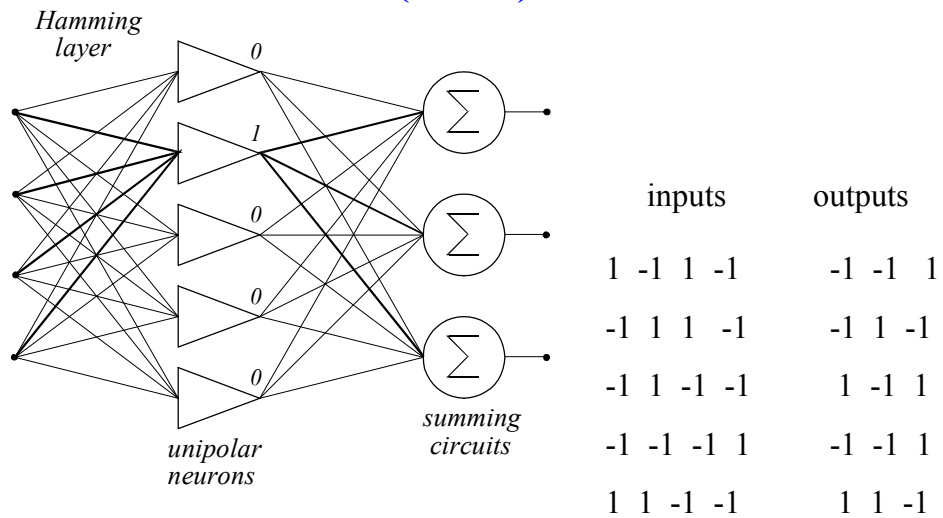
46

The cascade correlation architecture



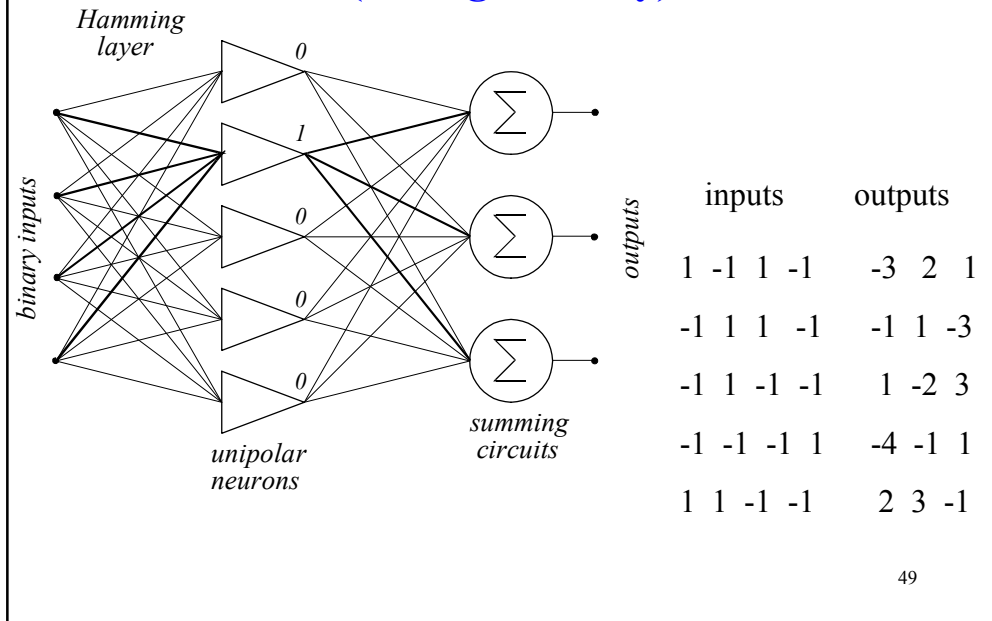
47

The counterpropagation networks (ROM)

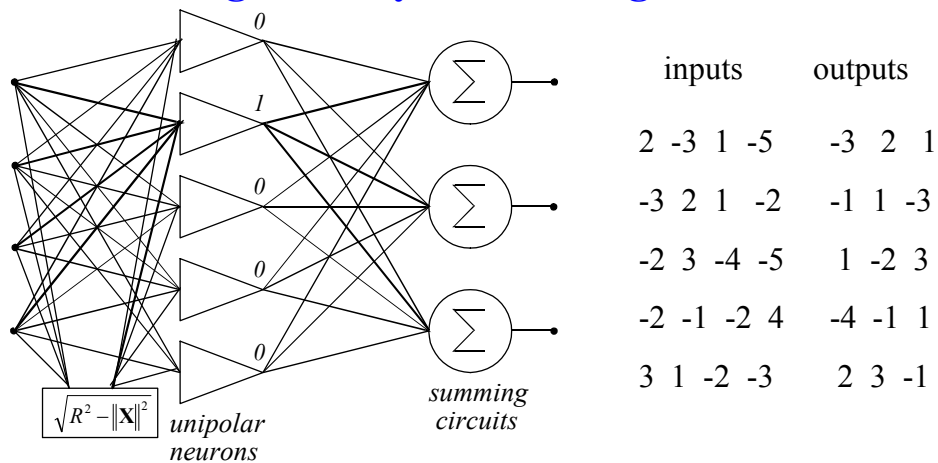


48

The counterpropagation networks (analog memory)



analog memory with analog address

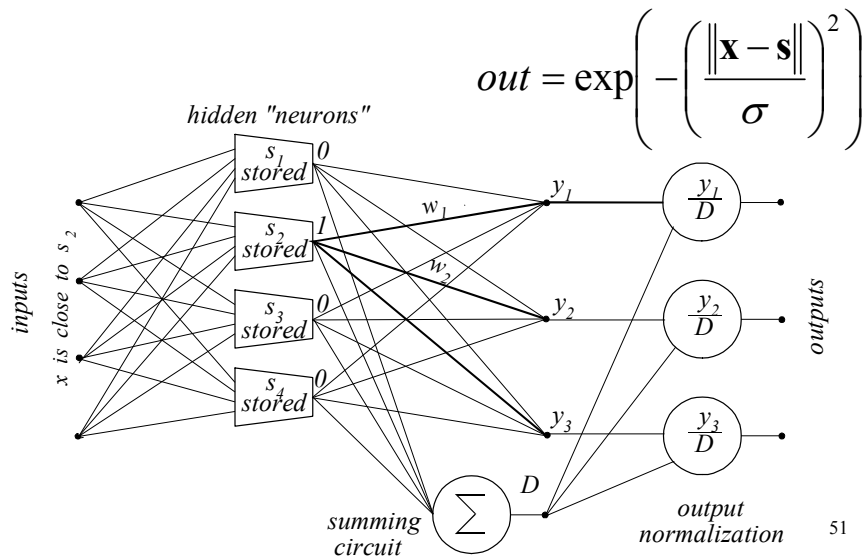


Consider using it as alternative to fuzzy systems

Number of neurons = number of predefined values

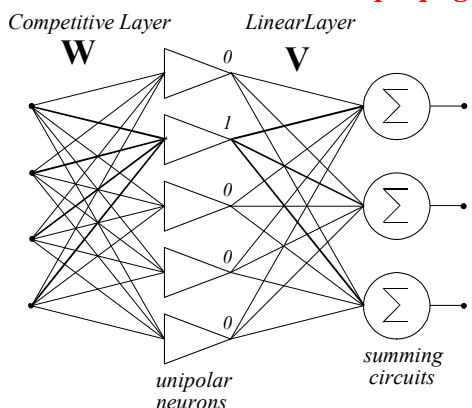
Easy implementation of systems with multiple inputs

RBF - Radial Basis Function networks *minimum distance classifier*



LVQ Learning Vector Quantization

Counterpropagation network

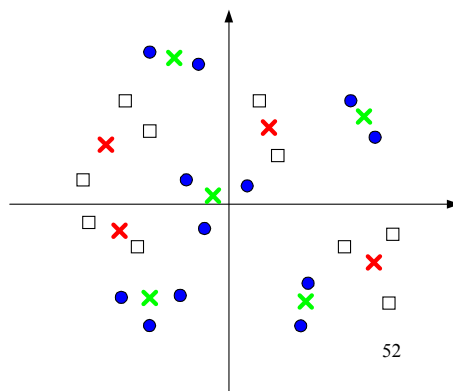


First layer computes Euclidean distances between input pattern and stored patterns.

Winning "neuron" is with the minimum distance

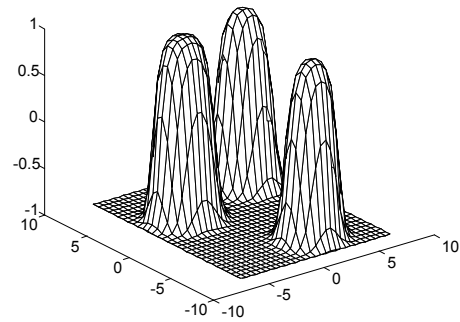
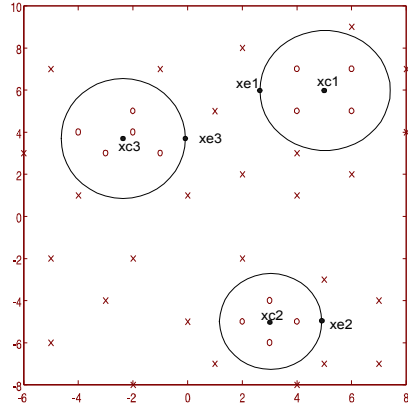
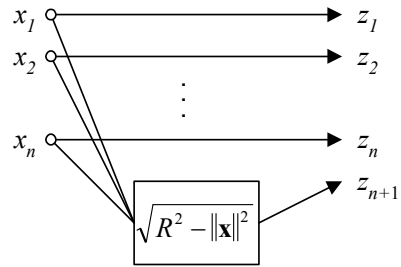
First layer detect subclasses

Second layer combines subclasses into a single class



Input pattern transformation on a sphere

*Fix to Kohonen network
deficiency*



Bogdan M. Wilamowski

Problems with computational intelligence

- Introduction
- Neural Network Learning
- Neural Networks Architectures
- Challenges in Neural Networks
- **Fuzzy Systems**
- Comparison of Neural and Fuzzy Systems
- Evolutionary Computation

Fundamentals of Fuzzy Systems **Fuzzy logic**

Comparison Boolean algebra with fuzzy logic

Boolean $A \cap B$			Boolean $A \cup B$			Fuzzy $A \cap B$			Fuzzy $A \cup B$		
0	0	0	0	0	0	0.2	0.3	0.2	0.2	0.3	0.3
0	1	0	0	1	1	0.2	0.8	0.2	0.2	0.8	0.8
1	0	0	1	0	1	0.7	0.3	0.3	0.7	0.3	0.7
1	1	1	1	1	1	0.7	0.8	0.7	0.7	0.8	0.8

Fuzzy systems

- Inputs can be any value from 0 to 1.
- The basic fuzzy principle is similar to Boolean logic.
- Max and min operators are used instead of AND and OR. The NOT operator also becomes $1 - \#$.

$A \cap B \cap C \Rightarrow \min\{A, B, C\}$ – smallest value of A, B or C

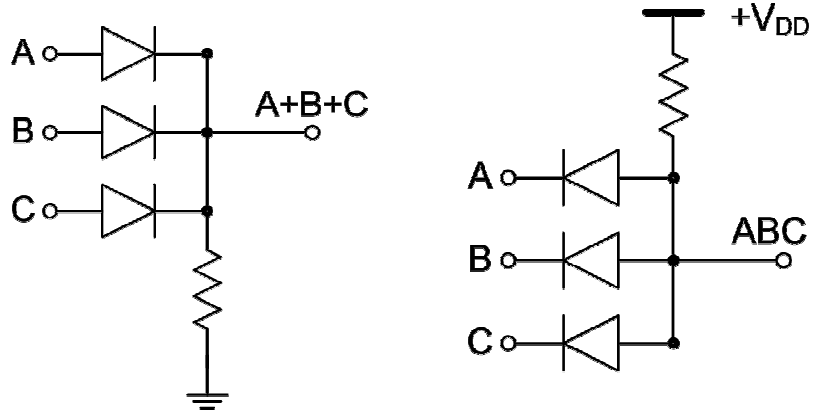
$A \cup B \cup C \Rightarrow \max\{A, B, C\}$ – largest value of A, B or C

$\overline{A} \Rightarrow 1 - A$	– one minus A	complement
Boolean		Fuzzy

Boolean $A \cap B$			Boolean $A \cup B$			Fuzzy $A \cap B$			Fuzzy $A \cup B$		
0	0	0	0	0	0	0.2	0.3	0.2	0.2	0.3	0.3
0	1	0	0	1	1	0.2	0.8	0.2	0.2	0.8	0.8
1	0	0	1	0	1	0.7	0.3	0.3	0.7	0.3	0.7
1	1	1	1	1	1	0.7	0.8	0.7	0.7	0.8	0.8

union *intersection*

Boolean or Fuzzy systems



Fuzzy systems

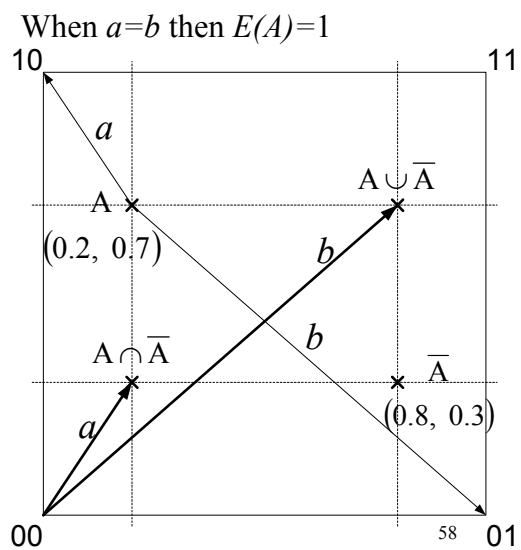
Entropy –measure of uncertainty

Entropy

$$E(A) = \frac{a}{b} = \frac{l^1(A, A_{near})}{l^1(A, A_{far})}$$

Fuzzy entropy theorem

$$E(A) = \frac{a}{b} = \frac{M(A \cap \bar{A})}{M(A \cup \bar{A})}$$



Fuzzy systems

$m_A(x_i)$ degree of association of variable x_i with fuzzy set A

Size of fuzzy set $M(A) = \sum_{i=1}^n m_A(x_i)$

Distance between fuzzy sets A and B

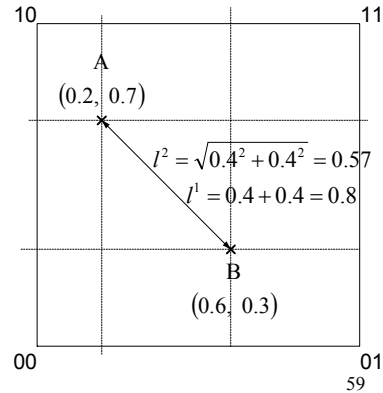
$$l^p(A, B) = \sqrt[p]{\sum_{i=1}^n |m_A(x_i) - m_B(x_i)|^p}$$

where p is distance order

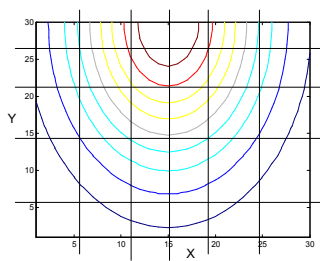
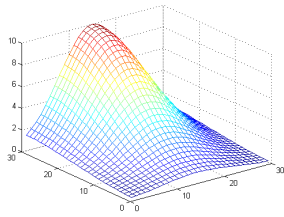
If $p=1$ this is fuzzy Hamming distance

$$l^1(A, 0) = M(A)$$

If $p=2$ this is Euclidean distance

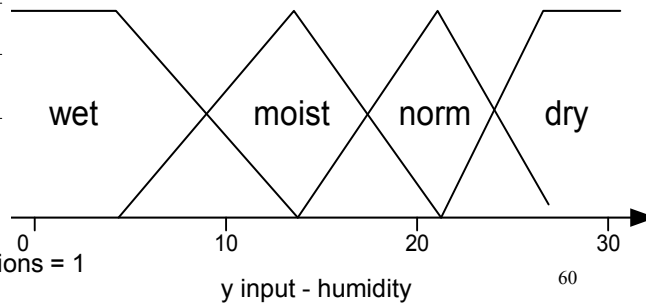
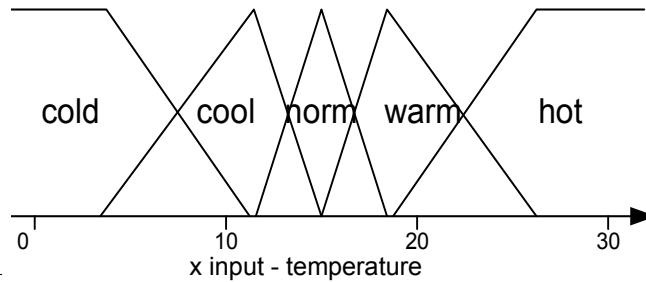


Design Example of Mamdani Fuzzy Controller

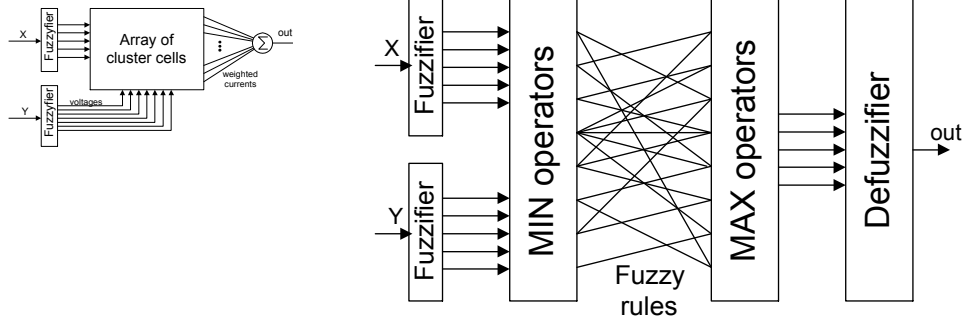


Sum of membership functions = 1

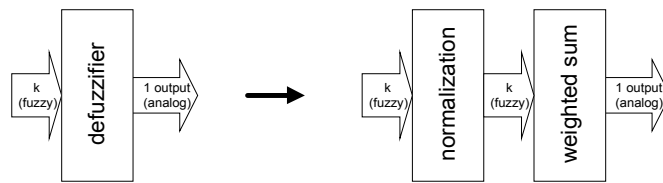
centroid



Fuzzy systems



Block diagram for Zadeh fuzzy controller

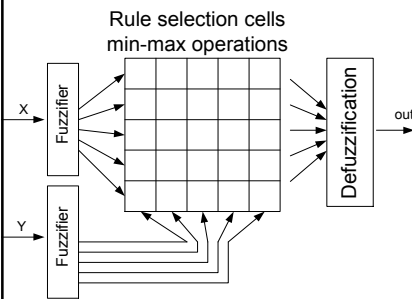


Takagi-Sugeno type defuzzifier

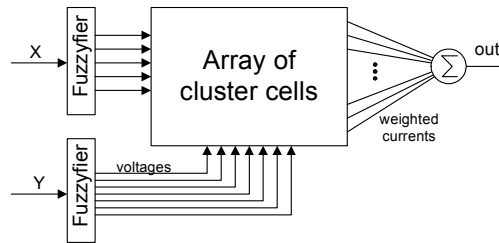
61

Fundamentals of Fuzzy Systems

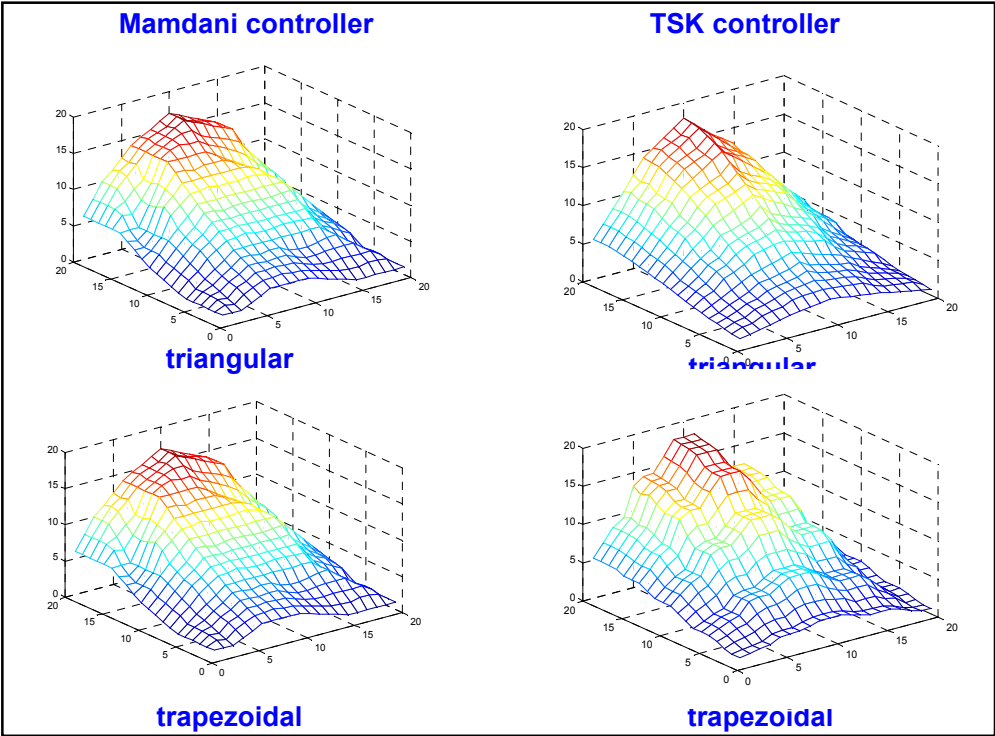
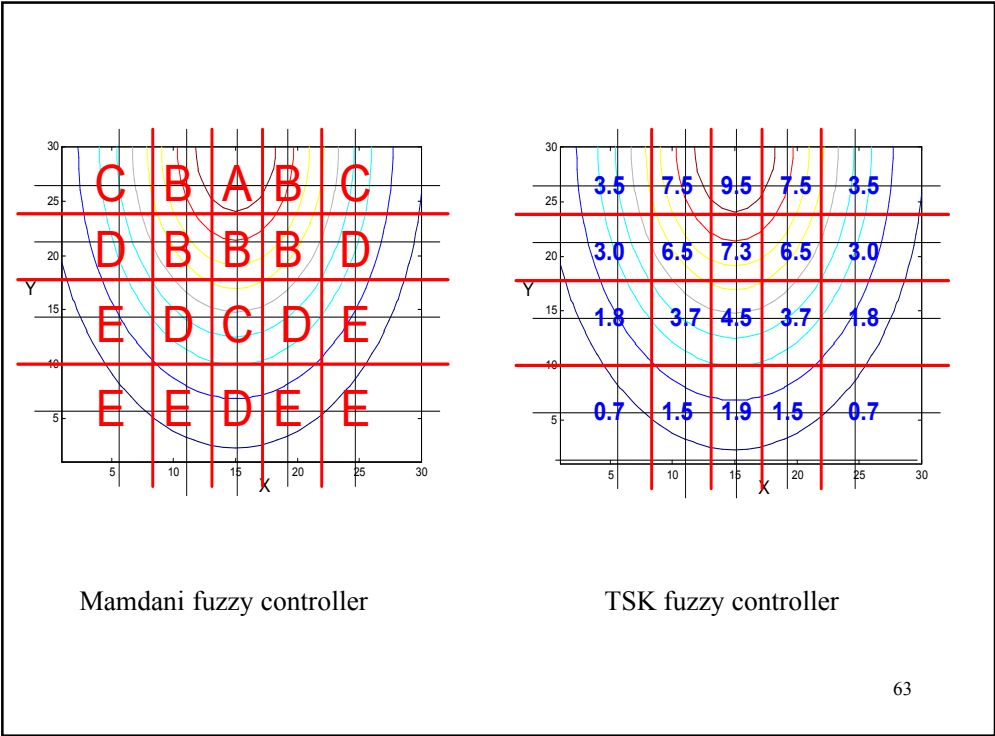
Fuzzy controllers

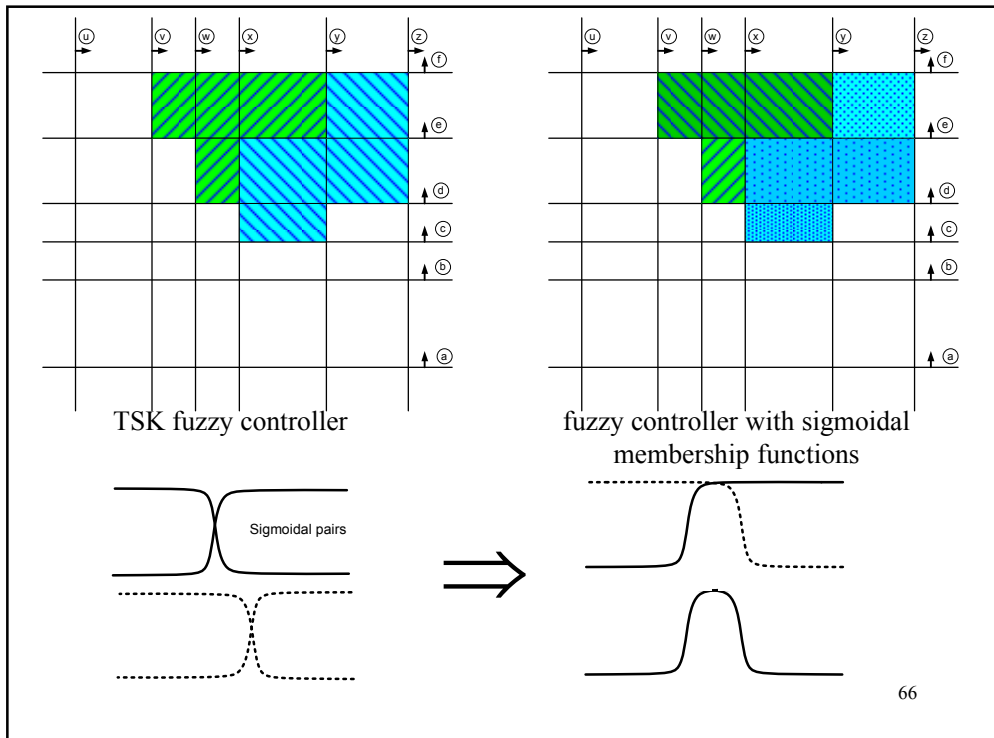
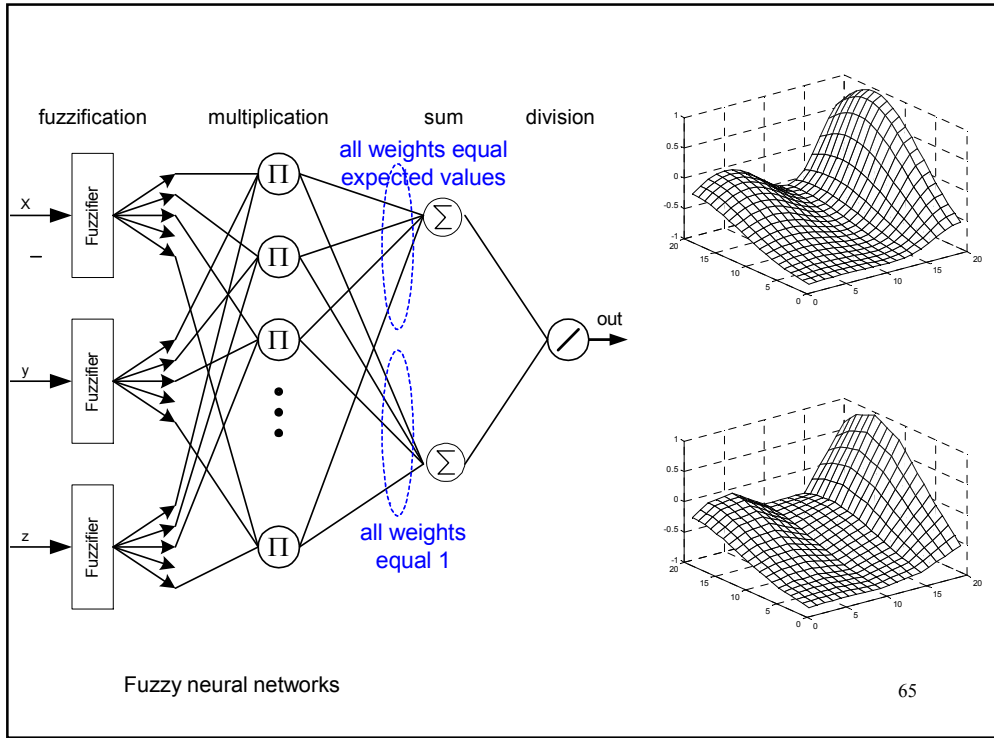


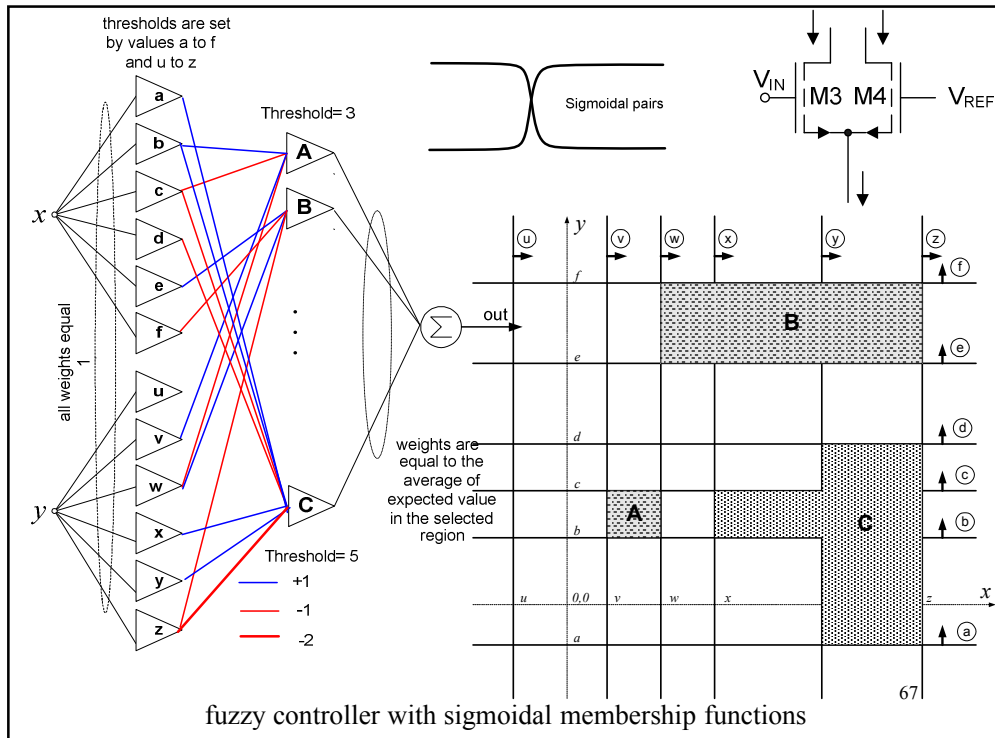
Block diagram of a Mamdani type fuzzy controller



Block diagram of a TSK (Takagi-Sugeno-Kang) fuzzy controller







Neural Networks or Fuzzy Systems

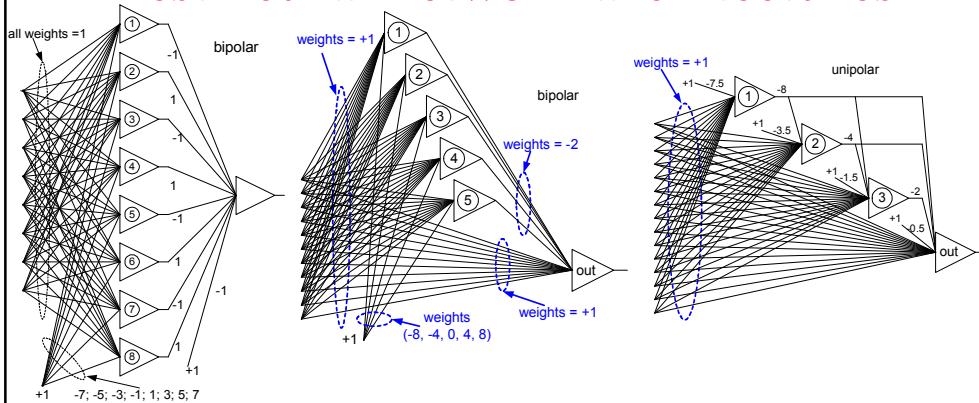
	Fuzzy	Neural
Number of inputs	-	+
Analog implementation	-	+
Digital implementation	-	+
Speed	-	+
Smoothens of the surface	-	+
Design complexity	+	+

So, most researches use FUZZY
Why researches are frustrated with neural ?

Problems with computational intelligence

- Introduction
- Neural Network Learning
- Neural Networks Architectures
- **Challenges in Neural Networks**
- Fuzzy Systems
- Comparison of Neural and Fuzzy Systems
- Evolutionary Computation

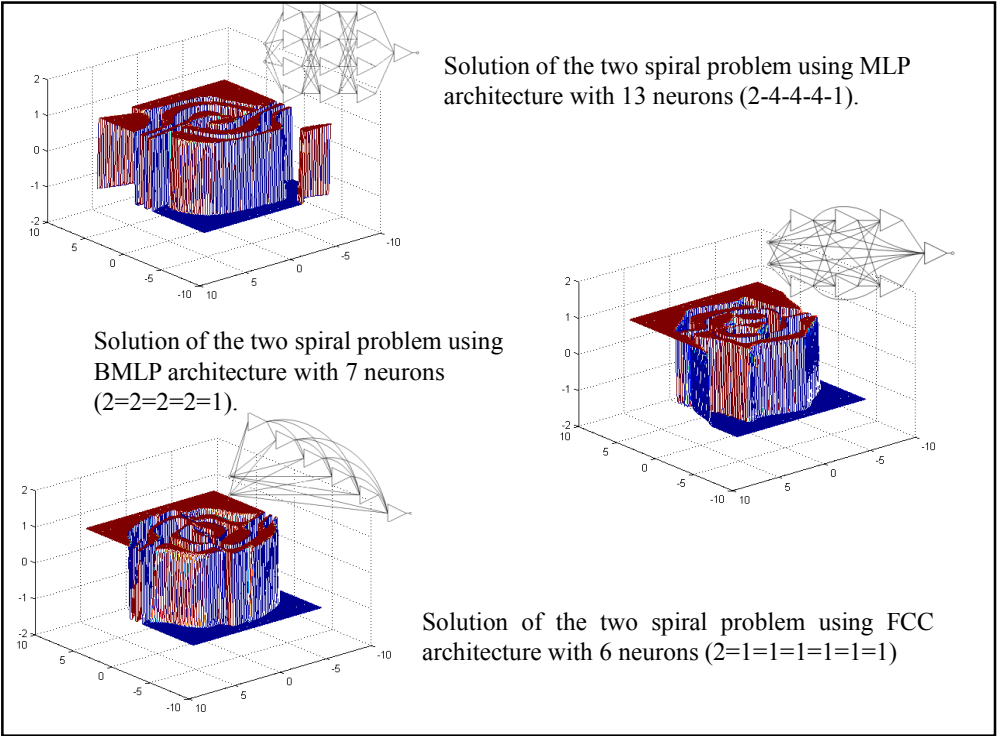
Best neural network architectures



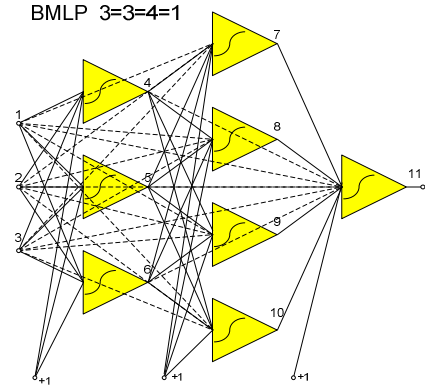
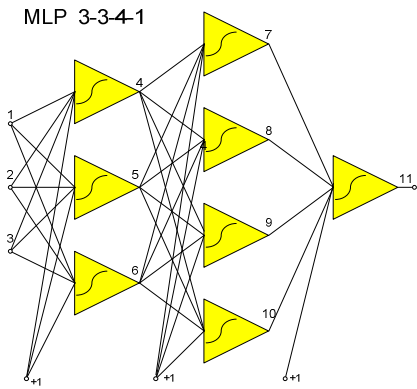
Layered bipolar neural network with one hidden layer for the **parity-8** problem.

Parity-11 implemented in fully connected bipolar neural networks with five neurons in the hidden layer.

Parity-15 implemented with 4 neurons in one cascade

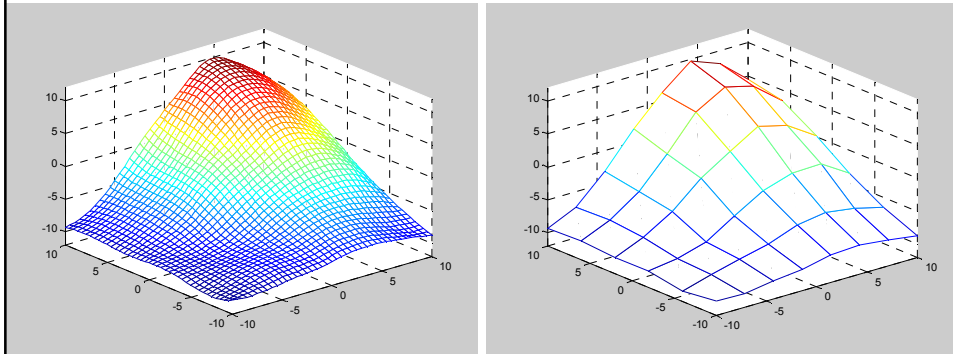


Best neural network architectures



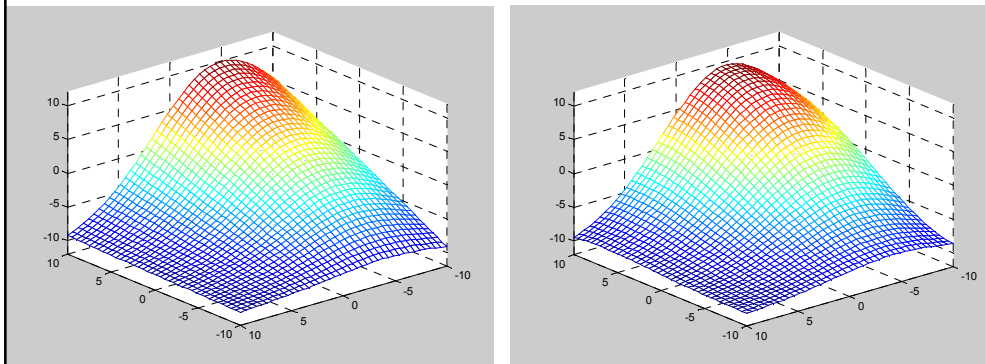
Most software can train only MLP
exceptions are SNNs and NBN

Performance of Neural Networks



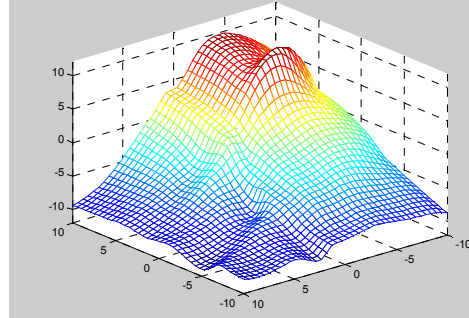
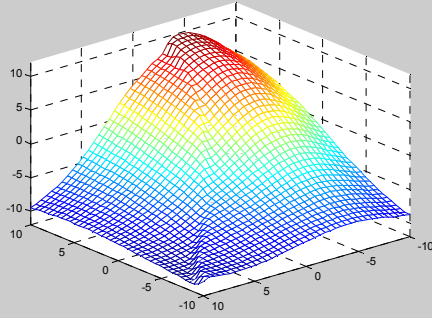
Control surface of TSK fuzzy controller (a) required control surface
(b) $8*6=48$ defuzzification rules

Performance of Neural Networks



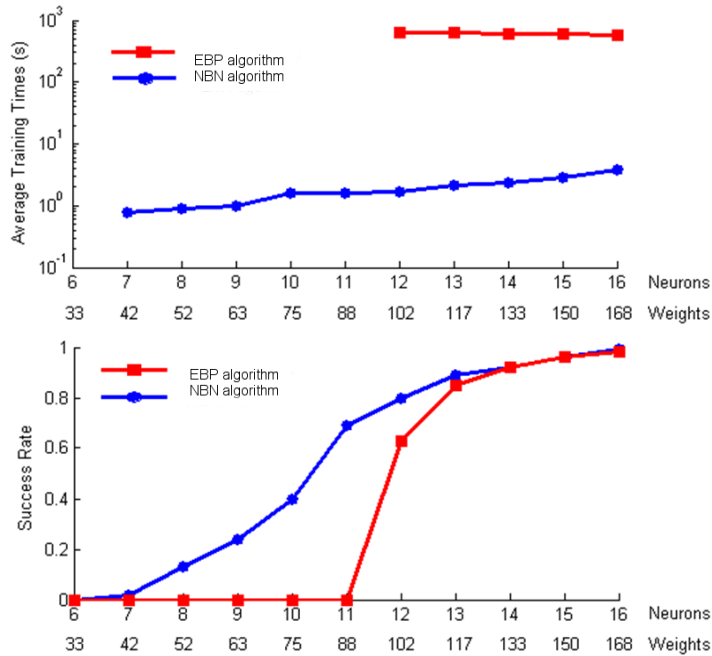
Control surface obtained with neural networks (a) 3 neurons in cascade
(12 weights) Training Error=0.21049 (b) 4 neurons in cascade (18
weights) Training Error=0.049061

Performance of Neural Networks



Control surface obtained with neural networks (a) 5 neurons in cascade (25 weights) Training Error=0.023973 (b) 8 neurons in cascade (52 weights) Training Error=1.118e-005

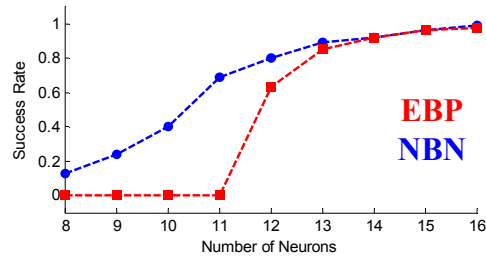
EBP is not able to train optimal architectures



Comparison between EBP algorithm and NBN algorithm, for different number of neurons in fully connected cascade networks: (a) average training time; (b) success rate

Common Mistakes

- ✓ Researchers are using wrong architectures
- ✓ Researchers are using excessive number of neurons
- ✓ First order algorithm such as EBP is not able to train optimal networks
- ✓ Second order algorithms such as LM can train only MLP networks



Newly developed NBN algorithm is not only very fast but it can train all neural network architectures and it can find solutions for optimal neural network architectures

Bogdan M. Wilamowski

Problems with computational intelligence

- Introduction
- Neural Network Learning
- Neural Networks Architectures
- Challenges in Neural Networks
- Fuzzy Systems
- **Comparison of Neural and Fuzzy Systems**
- Evolutionary Computation

Digital implementation

Neural network implementations usually require computation of the sigmoidal function

$$f(net) = \frac{1}{1 + \exp(-net)}$$

for unipolar neurons, or

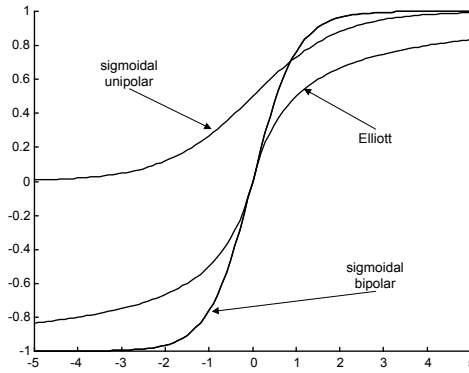
$$f(net) = \tanh(net) = \frac{2}{1 - \exp(-2net)} - 1$$

for bipolar neurons. These functions are relatively difficult to compute, making implementation on a microprocessor difficult. If the Elliott function is used:

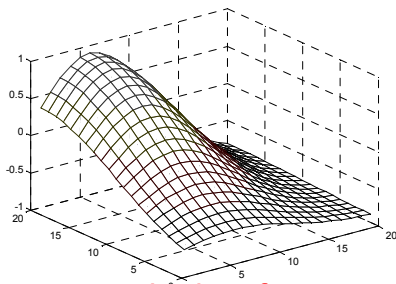
$$f(net) = \frac{net}{1 + |net|}$$

instead of the sigmoidal, then the computations are relatively simple and the results are almost as good as in the case of sigmoidal function.

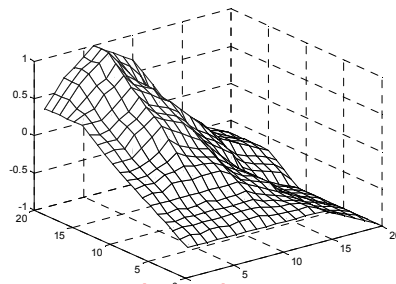
Elliott function



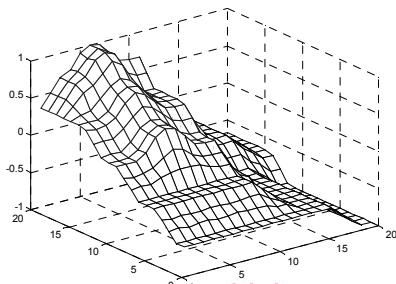
79



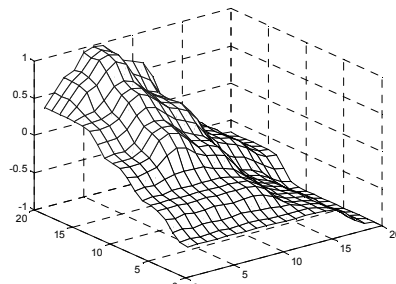
required surface



triangular



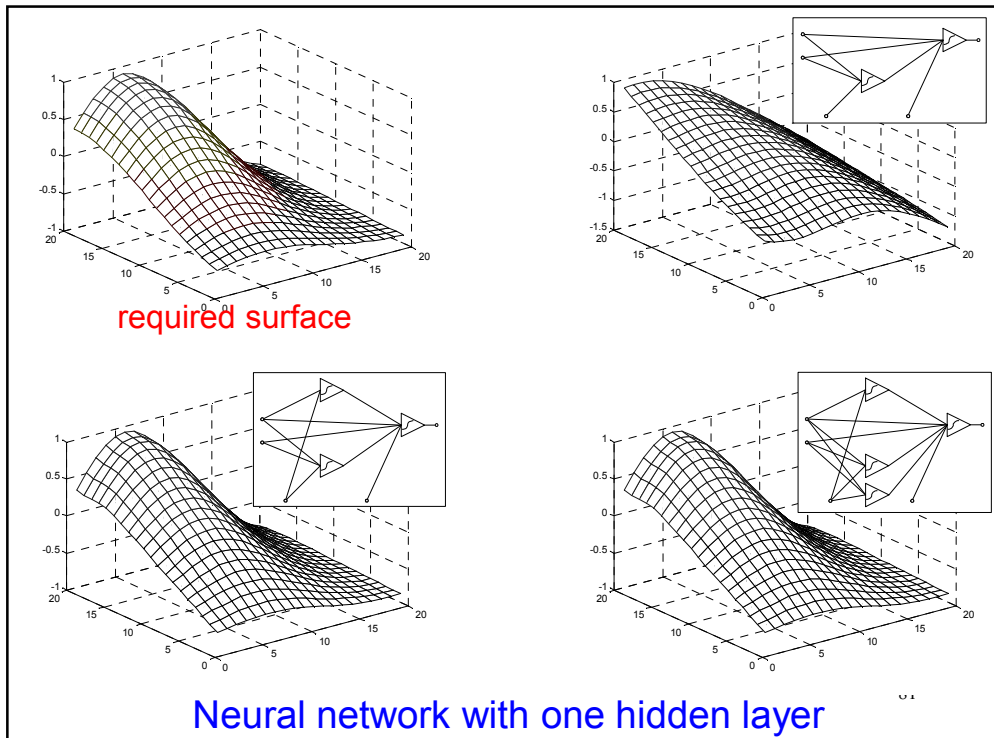
trapezoidal



Gaussian

Mamdani fuzzy with MIN

80



two inputs cases

<p>Fuzzy with 6 membership functions</p> <p><i>Mamdani fuzzy controllers require:</i> $2*6+6 = 18$ analog values + rule table $36*3 = 108$ bits</p> <p><i>TSK fuzzy controllers require:</i> $2*6+6*6 = 48$ analog values No rule table has to be stored</p>	<p>Neural networks</p> <p>2 hidden neurons $2*3+5 = 11$ analog values</p> <p>3 hidden neurons $3*3+5 = 14$ analog values</p> <p>4 hidden neurons $4*3+6 = 18$ analog values</p>
--	---

Comparison of various fuzzy and neural controllers

Type of controller	length of code	processing time (ms)	Error MSE
Mamdani with trapezoidal	2324	1.95	0.945
Mamdani with triangular	2324	1.95	0.671
Mamdani with Gaussian	3245	39.8	0.585
Tagagi-Sugeno with trapezoidal	1502	28.5	0.309
Tagagi-Sugeno with triangular	1502	28.5	0.219
Tagagi-Sugeno with Gaussian	2845	52.3	0.306
Neural network with 3 neurons in cascade	680	1.72	0.00057
Neural network with 5 neurons in cascade	1070	3.3	0.00009
Neural network with 6 neurons in one hidden layer	660	3.8	0.00030 83

Bogdan M. Wilamowski

Problems with computational intelligence

- Introduction
- Neural Network Learning
- Neural Networks Architectures
- Challenges in Neural Networks
- Fuzzy Systems
- Comparison of Neural and Fuzzy Systems
- **Evolutionary Computation**

84

Genetic Algorithms

The genetic algorithms follow the evolution process in the nature to find the better solutions of some complicated problems.

Foundations of genetic algorithms are given in Holland (1975) and Goldberg (1989) books.

Genetic algorithms consist the following steps:

- Initialization
- Selection
- Reproduction with crossover and mutation

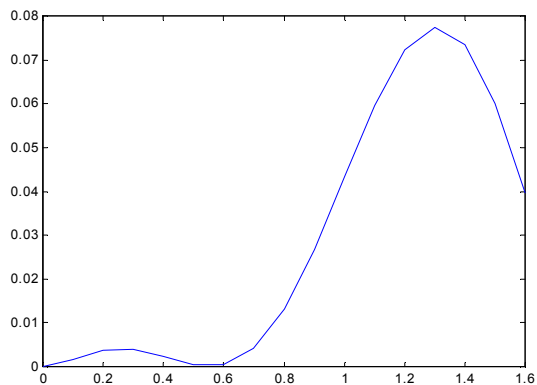
Selection and reproduction are repeated for each generation until a solution is reached

During this procedure a certain strings of symbols, known as chromosomes, evaluate toward better solution.

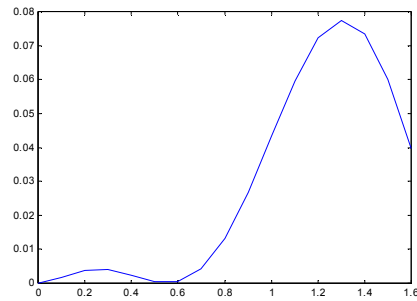
85

Genetic Algorithms 2

All significant steps of the genetic algorithm will be explained using a simple example of finding a maximum of the function $(\sin^2(x) - 0.5*x)^2$ with the range of x from 0 to 1.6. Note, that in this range the function has global maximum at $x=1.309$, and local maximum at $x=0.262$.



Genetic Algorithms 2



Coding and initialization

At first, the variable x has to be represented as a string of symbols. With longer strings process converges usually faster, so less symbols for one string field are used it is the better. While this string may be the sequence of any symbols, the binary symbols "0" and "1" are usually used. In our example, let us use for coding six bit binary numbers having a decimal value of $40x$. Process starts with a random generation of the initial population given in Table

87

Genetic Algorithms 3

Initial Population					
	string	decimal value	variable value	function value	fraction of total
1	101101	45	1.125	0.0633	0.2465
2	101000	40	1.000	0.0433	0.1686
3	010100	20	0.500	0.0004	0.0016
4	100101	37	0.925	0.0307	0.1197
5	001010	10	0.250	0.0041	0.0158
6	110001	49	1.225	0.0743	0.2895
7	100111	39	0.975	0.0390	0.1521
8	000100	4	0.100	0.0016	0.0062
Total				0.2568	1.0000

88

Genetic Algorithms 4

Selection and reproduction

Selection of the best members of the population is an important step in the genetic algorithm. Many different approaches can be used to rank individuals. In our example the ranking function is given. Highest rank has member number 6 and lowest rank has member number 3. Members with higher rank should have higher chances to reproduce. The probability of reproduction for each member can be obtained as fraction of the sum of all objective function values. This fraction is shown in the last column of the Table.

Using a random reproduction process the following population arranged in pairs could be generated:

<i>101101 -> 45</i>	<i>110001 -> 49</i>	<i>100101 -> 37</i>	<i>110001 -> 49</i>
<i>100111 -> 39</i>	<i>101101 -> 45</i>	<i>110001 -> 49</i>	<i>101000 -> 40</i>

89

Genetic Algorithms 5

Reproduction

<i>101101 -> 45</i>	<i>110001 -> 49</i>	<i>100101</i>	<i>-> 37</i>
<i>110001 -> 49</i>			
<i>100111 -> 39</i>	<i>101101 -> 45</i>	<i>110001 -> 49</i>	<i>101000 -> 40</i>

If size of the population from one generation to another is the same, therefore two parents should generate two children. By combining two strings two another strings should be generated. The simplest way to do it is to split in half each of the parent string and exchange substrings between parents. For example from parent strings *010100* and *100111* the following child strings will be generated *010111* and *100100*. This process is known as the crossover and resultant children are shown below

<i>101111 -> 47</i>	<i>110101 -> 53</i>	<i>100001 -> 33</i>	<i>110000 -> 48</i>
------------------------	------------------------	------------------------	------------------------

Genetic Algorithms 6

Mutation

On the top of properties inherited from parents they are acquiring some new random properties. This process is known as mutation. In most cases mutation generates low ranked children, which are eliminated in reproduction process. Sometimes however, the mutation may introduce a better individual with a new property into. This prevents process of reproduction from degeneration. In genetic algorithms mutation plays usually secondary role. Mutation rate is usually assumed on the level much below 1%. In our example mutation is equivalent to the random bit change of a given pattern. In this simple example with short strings and small population with a typical mutation rate of 0.1%, our patterns remain practically unchanged by the mutation process. The second generation for our example is shown in Table

91

Genetic Algorithms 7

Population of Second Generation					
string number	string	decimal value	variable value	function value	fraction of total
1	010111	47	1.175	0.0696	0.1587
2	100100	37	0.925	0.0307	0.0701
3	110101	53	1.325	0.0774	0.1766
4	010001	41	1.025	0.0475	0.1084
5	100001	33	0.825	0.0161	0.0368
6	110101	53	1.325	0.0774	0.1766
7	110000	48	1.200	0.0722	0.1646
8	101001	41	1.025	0.0475	0.1084
<i>Total</i>				0.4387	1.0000

92

Genetic Algorithms 8

Note, that two identical highest ranking members of the second generation are very close to the solution $x=1.309$. The randomly chosen parents for third generation are

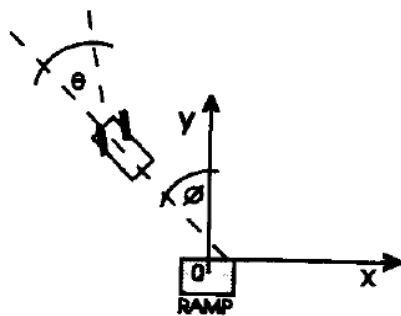
$010111 \rightarrow 47$ $110101 \rightarrow 53$ $110000 \rightarrow 48$
 $101001 \rightarrow 41$
 $110101 \rightarrow 53$ $110000 \rightarrow 48$ $101001 \rightarrow 41$ $110101 \rightarrow 53$

which produces following children:

$010101 \rightarrow 21$ $110000 \rightarrow 48$ $110001 \rightarrow 49$
 $101101 \rightarrow 45$
 $110111 \rightarrow 55$ $110101 \rightarrow 53$ $101000 \rightarrow 40$ $110001 \rightarrow 49$

The best result in the third population is the same as in the second one. By careful inspection of all strings from second or third generation one may conclude that using crossover, where strings are always split into half, the best solution $110100 \rightarrow 52$ will never be reached no matter how many generations are created.

Genetic Algorithms 9

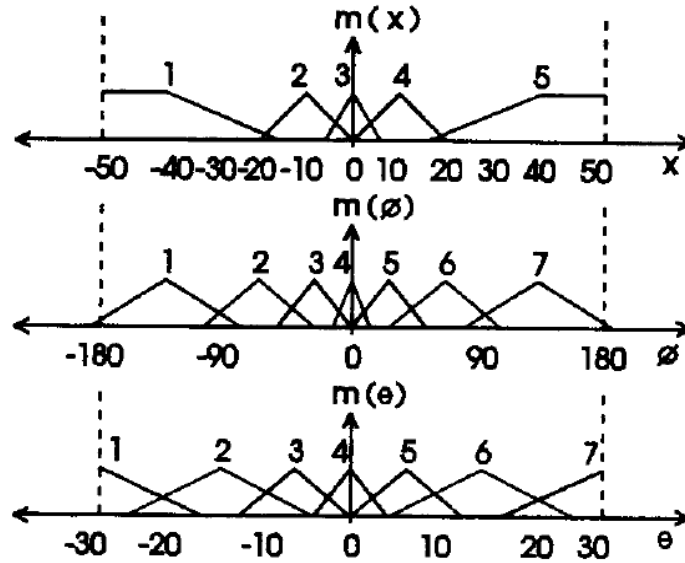


$$\begin{aligned}
 x_{i+1} &= x_i - r \sin(\phi_i) \\
 y_{i+1} &= y_i + r \cos(\phi_i) \\
 \phi_{i+1} &= \phi_i + \Theta_i
 \end{aligned}$$

$$E = w_x x^2 + w_y y^2 + w_\phi \phi^2$$

$$E = w_x x^2 + w_\phi \phi^2$$

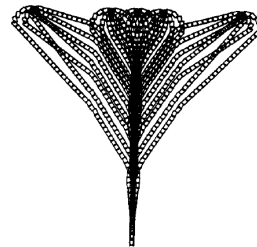
Genetic Algorithms 10



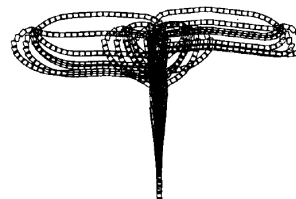
95

Genetic Algorithms 11

	x				
	1	2	3	4	5
1	7	7	7	7	7
2	4	6	7	7	7
3	1	3	5	6	7
4	1	2	4	6	7
5	1	2	3	5	7
6	1	1	1	2	4
7	1	1	1	1	1



	x				
	1	2	3	4	5
1	5	6	6	7	7
2	3	3	7	7	7
3	2	3	5	7	6
4	3	3	4	5	7
5	1	1	3	5	7
6	1	1	1	5	6
7	1	1	1	2	3

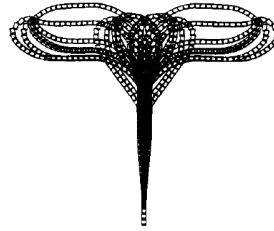


96

Genetic Algorithms 12

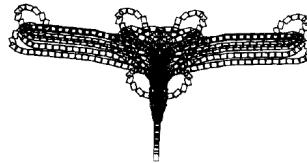
	x				
ϕ	1	2	3	4	5
1	5	6	6	7	7
2	3	5	6	7	7
3	2	3	5	6	7
4	2	3	4	5	6
5	1	2	3	5	6
6	1	1	2	3	5
7	1	1	2	2	3

(a)



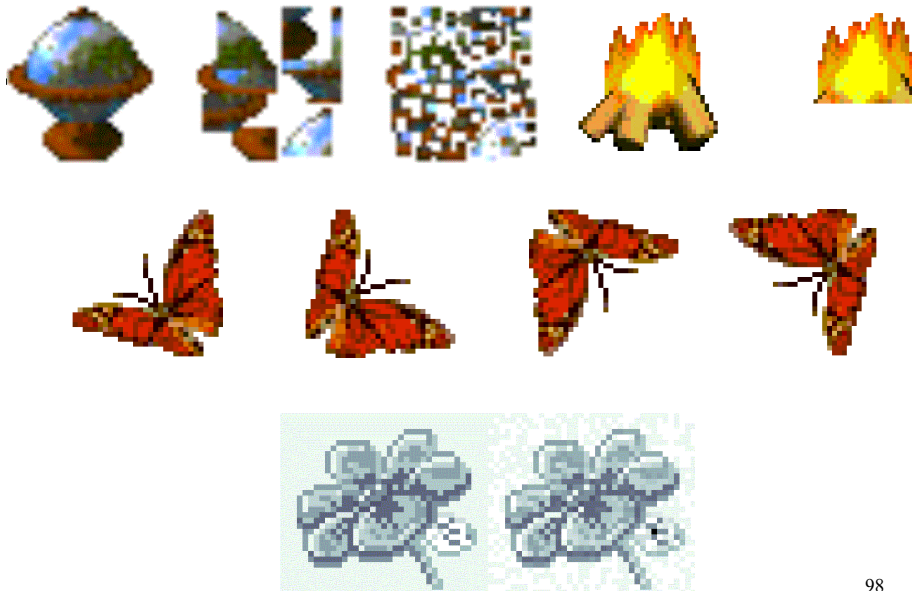
(b)

	x				
ϕ	1	2	3	4	5
1	7	7	7	1	1
2	2	4	7	7	7
3	1	1	6	7	7
4	1	1	6	7	7
5	1	1	2	7	7
6	1	1	1	4	6
7	7	7	1	1	1



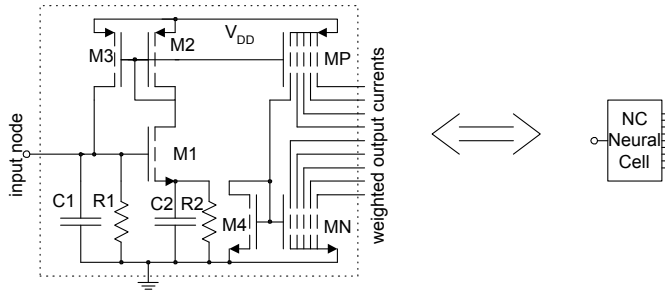
97

Pulse Coded Neural Networks 5

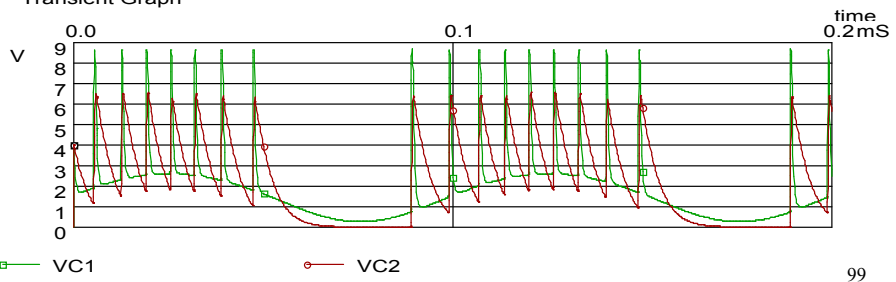


98

Pulse Coded Neural Networks

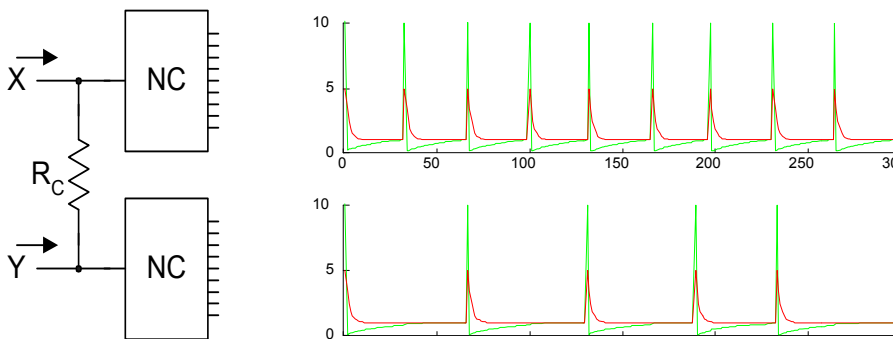


Transient Graph



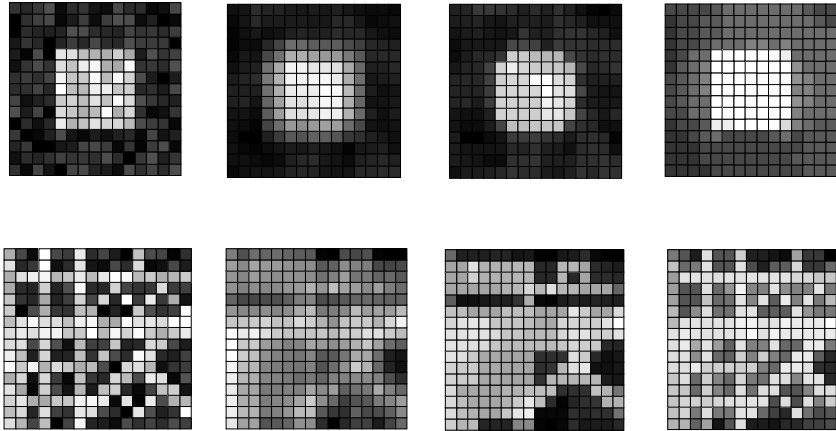
99

Pulse Coded Neural Networks 2



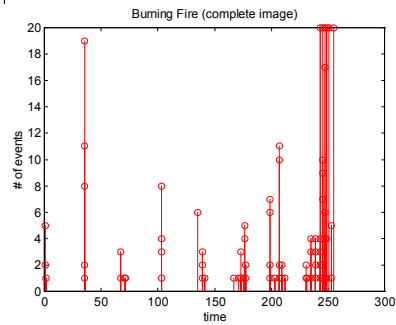
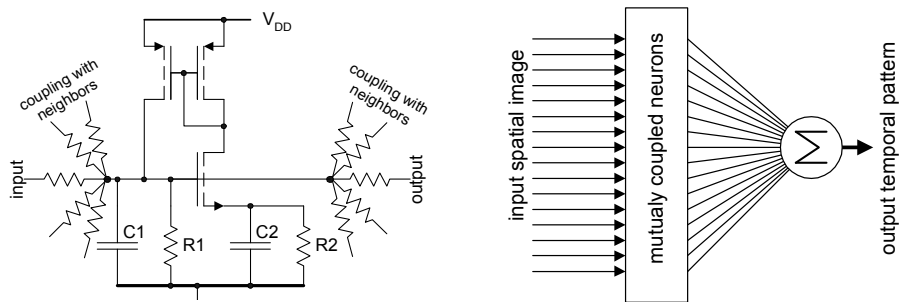
100

Pulse Coded Neural Networks 3



101

Pulse Coded Neural Networks 4



102

Pulse Coded Neural Networks 5



103