WARSAW UNIVERSITY OF TECHNOLOGY
FACULTY OF MATHEMATICS
AND INFORMATION SCIENCE

Neural Networks

Lecture 5

Hamming Model

Hamming Model

The model with binary inputs and weights fixed during the preparatory phase.

The Hamming Model is a optimal classifier. The systems calculates the similarity (the Hamming distance) between the input signal and each pattern stored in the network. Next, the most similar stored pattern is selected.

Hamming Model

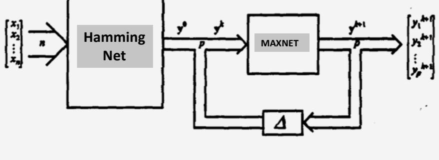
2 independent layers:

- the first layer – calculating the similarity
- the second layer – blocking all signals except the biggest one.

The weights and thresholds in the 1 layer are selected to assure that the s^{th} element input signal will be equal to $\boldsymbol{N - H^{input,s}}$ where $\boldsymbol{N}$ is the number of bits in the input signal (and of course in the stored patterns)), $H^{input,s}$ is the Hamming distance between the input signal and s^{th} stored pattern.

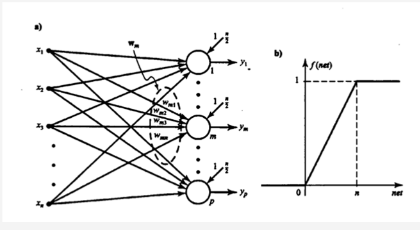
Hamming Model

Block diagram



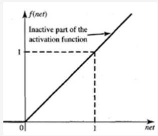
Hamming Model

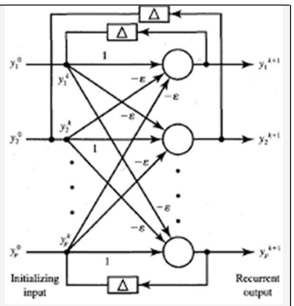
Hammings' network



Hamming Model

MAXNET

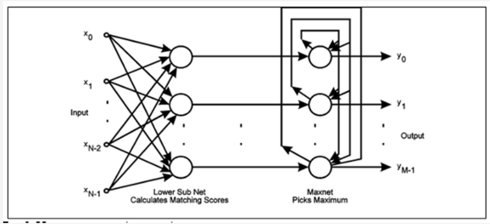




7

Hamming Model

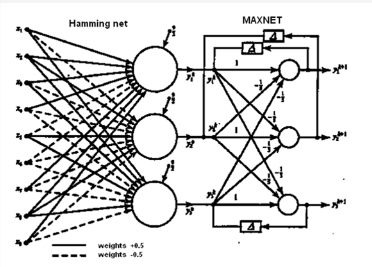
Two layer Hamming network



8

Hamming Model

Two layer Hamming network (example)



9

Hamming Model

Output signals from Hamming's net are equal to: 1,2, ..., *N*. The greater value of output signal means that input signal ***X*** is more similar to the stored pattern *s*.

MAXNET, with internal connections based on the ***lateral inhibition rule*** has to select the greatest signal suppressing to zero the other signals.

10

Hamming Model

The net is able to store *p*, N-dimensional patterns $\mathbf{s}^{(m)}$.

Each element in the I[st] layer is „*responsible*" for the one stored pattern. The incoming weights to that *m*-th element

$$\mathbf{w}_m = [w_{m1}, w_{m2}, \dots, w_{mN}]$$

connects input nodes with that element.

The classifierer has *p* class, *p* elements and *p* outputs.

11

Hamming Model

The nonlinear characteristic of an element produce at the output the signal is equal to 1 if and only if the input signal is identical with the stored pattern.

The incoming weights of the element (*m*), that one where the *m*-ty pattern is stored, are equal

$$\mathbf{w}_m = \mathbf{s}^{(m)}$$

The input signals of network elements are

$$\mathbf{X}^T\mathbf{s}^{(1)}, \mathbf{X}^T\mathbf{s}^{(2)}, \dots, \mathbf{X}^T\mathbf{s}^{(m)}, \dots, \mathbf{X}^T\mathbf{s}^{(p)}$$

12

Hamming Model

If the input signal $\mathbf{X} = \mathbf{s}^{(m)}$, the only one weighted input is equal to **N,** and the rest belongs to the (-N;+N) (the input signals x_i are equal to -1 or +1).

The inner (scalar) products $\mathbf{X}^T\mathbf{s}^{(m)}$ are used to calculate the similarities between the input signal and stored patterns.

The inner product $\mathbf{X}^T\mathbf{s}^{(m)}$ can be written as:

the number of positions (bits) where they agree – minus the number of positions where they disagree.

13

Hamming Model

The number of positions they disagree – it is Hammings' distance

$$H(\mathbf{X},\mathbf{s}^{(m)})$$

so, the number where they agree

$$N - H(\mathbf{X},\mathbf{s}^{(m)})$$

hence $\mathbf{X}^T\mathbf{s}^{(m)} = \{N - H(\mathbf{X},\mathbf{s}^{(m)})\} - H(\mathbf{X},\mathbf{s}^{(m)})$,

$$\mathbf{X}^T\mathbf{s}^{(m)}/2 = N/2 - H(\mathbf{X},\mathbf{s}^{(m)})$$

14

Hamming Model

The weight matrix of Hammings' net **W** (connections with the I[st] layer)

$$W = \frac{1}{2}\begin{bmatrix} s_1^{(1)} & s_2^{(1)} & \circ & \circ & s_N^{(1)} \\ s_1^{(2)} & s_2^{(2)} & \circ & \circ & s_N^{(2)} \\ \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ \\ s_1^{(p)} & s_2^{(p)} & \circ & \circ & s_N^{(p)} \end{bmatrix}$$

The input signal **X** produce at the input of each element signal

$$\frac{1}{2} X s^{(m)}$$

15

Hamming Model

Plus the additional constant bias signal of N/2,

$$E^H = \frac{1}{2} X s^{(m)} + \frac{N}{2} = N - H(X, s^{(m)})$$

The nonlinear characteristic

$$f(E^H) = \frac{1}{N} E^H$$

produce at the output signal of value <0; 1>. The element of a greater output signal indicates the class (the number of a class) where the input signal **X** has the smallest Hamming distance. The best matching

H = 0 i ***f(E***^{H}***) =1***

16

Hamming Model

The iterative procedure of MAXNET have to suppress the rest (smaller) output signals

$$W_{MAXNET} = \begin{bmatrix} 1 & -\varepsilon & \circ & \circ & -\varepsilon \\ -\varepsilon & 1 & \circ & \circ & -\varepsilon \\ \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ \\ -\varepsilon & -\varepsilon & \circ & \circ & 1 \end{bmatrix} = W_M$$

where $0 < \varepsilon < 1/p$

the coefficient of lateral inhibition

17

Hamming Model

The recursive procedure

$$y(t+1) = \Psi\left[W_{MAXNET} \cdot y(t)\right]$$

where $\Psi[a] = \begin{cases} 0 & \text{if } a < 0 \\ a & \text{if } a \geq 0 \end{cases}$

18

Hamming Model

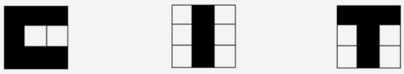
Example

$$s^{(1)} = [+1 \quad +1 \quad +1 \quad +1 \quad -1 \quad -1 \quad +1 \quad +1 \quad +1]$$

$$s^{(2)} = [-1 \quad +1 \quad -1 \quad -1 \quad +1 \quad -1 \quad -1 \quad +1 \quad -1]$$

$$s^{(3)} = [+1 \quad +1 \quad +1 \quad -1 \quad +1 \quad -1 \quad -1 \quad +1 \quad -1]$$

these are patterns of C, I and T



19

Hamming Model

the weight vector

$$W = \begin{bmatrix} s^{(1)} \\ s^{(2)} \\ s^{(3)} \end{bmatrix}$$

the input signals to the Hamming net elements

$$E^H = \frac{1}{2} W \cdot X + \begin{bmatrix} 9/2 \\ 9/2 \\ 9/2 \end{bmatrix}$$

20

Hamming Model

Let us assume that the input signal

$$X = [+1 \quad +1 \quad +1 \quad +1 \quad +1 \quad +1 \quad +1 \quad +1 \quad +1]$$

it has to be classified to one of three classes ***C,I,T***
Let ε = 0,2 (< 1/3 = 1/p), hence

$$E^H = \begin{bmatrix} 7 \\ 3 \\ 5 \end{bmatrix} \qquad f(E^H) = \begin{bmatrix} 7/9 \\ 3/9 \\ 5/9 \end{bmatrix}$$

$f(E^H) = y(0)$ is the first input to the MAXNET

21

Hamming Model

Iterative procedure

$$y(t+1) = \Psi\left[E^M \cdot y(t)\right]$$

yields

$$W_M \cdot y(t) = \begin{bmatrix} 1 & -0.2 & -0.2 \\ -0.2 & 1 & -0.2 \\ -0.2 & -0.2 & 1 \end{bmatrix} \begin{bmatrix} y_1(t) \\ y_2(t) \\ y_3(t) \end{bmatrix}$$

and next

$$E^M(1) = \left[6/10 \quad 1/15 \quad 1/3\right] = y(1)$$

22

Hamming Model

next

$$E^M(2) = \begin{bmatrix} \frac{13}{25} & \frac{-3}{25} & \frac{1}{5} \end{bmatrix} \Rightarrow$$

$$y(2) = \begin{bmatrix} \frac{13}{25} & 0 & \frac{1}{5} \end{bmatrix}$$

$$E^M(3) = \begin{bmatrix} \frac{12}{25} & \frac{-18}{125} & \frac{12}{125} \end{bmatrix} \Rightarrow$$

$$y(3) = \begin{bmatrix} \frac{12}{25} & 0 & \frac{12}{125} \end{bmatrix}$$

$$E^M(4) = \begin{bmatrix} \frac{576}{1250} & \frac{-144}{1250} & 0 \end{bmatrix} \Rightarrow$$

$$y(4) = \begin{bmatrix} \frac{576}{1250} & 0 & 0 \end{bmatrix}$$

23

Hamming Model

The last result stops the procedure – the stable state is achieved.

<u>Conclussion:</u>

The unknown input signal ***X*** had the smallest Hamming distance from the pattern $s^{(1)}$, so it belonged to the

class - **C**

24

The Hopfield Model

The Hopfield Model

W 1982 J. J. Hopfield
Neural Networks and Physical Systems with Emergent Collective Computational Abilities

Model similar to the perceptron – but with many differences.

It is not only the model – it is the ideology.

The Hopfield Model

Hopfield exploited an analogy to energy states in physics and introduced the *computational energy function*. Like a physical system, the network seeks its lowest energy state and with the iteration procedure converges to the stable state.

The Hopfield network is able to *memorize* and next *reproduce* the information on the base of an incomplete or noisy input signal.

The Hopfield Model

The system associates the input information with this stored which is the ”*closest*” in accordance to the measure of similarity.

The algorithm realized by the network is called

nearest neighbour algorithm

The Hopfield model has a shortage of precise mathematical description and precise convergence conditions.

The Hopfield Model

Network description

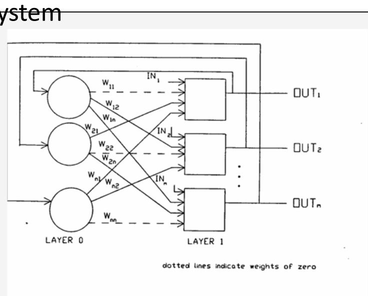
The Hopfield net consists of a number of elements, each connected to every other element - it is fully connected network (but no self feedback loops).

It is also symmetrically-weighted network, since the weights on the connections from one element to another are the same in both directions.

Each element has, like the single-layer perceptron, a threshold and each element calculates the weighted sum of their inputs minus the threshold value.

The Hopfield Model

The system



The Hopfield Model

Network operation:

The input and output signals can be binary e.g. $x \in \{-1,+1\}$ (the bipolar case) or $x \in \{0,1\}$ (the unipolar case) or continuous valued.

Next an unknown object is input to the network which proceeds to cycle (the first network output is taken as the new input, which produces an output and so on.) through a succession of states, until it converges on a stable solution, which happens when the output values of elements no longer alter.

31

The Hopfield Model

The network is prepared during the initialization (or learning) phase when the interconnecting matrix is calculated.

The interconnection weights w_{ij}, $i,j = 1,2,\ldots,n$ form the $n \times n$ symmetric interconnection matrix $\boldsymbol{W}$, which is defined by the outer - product learning rule

$$w_{ij} = \begin{cases} \sum_{s=1}^{N} x_i^s x_j^s & \text{dla} \quad i \neq j \\ 0 & \text{dla} \quad i = j \end{cases}$$

N is the number of stored objects,

x_j^s is the *j* element of object *s*.

32

The Hopfield Model

Comparison Perceptron - Hopfield

❖ in a perceptron network is learned through the repeated adjustment of weights

❖ in a Hopfield model network is prepared during the initialization (or learning) phase when the interconnecting matrix is calculated.

33

The Hopfield Model

Comparison Perceptron - Hopfield

❖ in a perceptron network is addressed by the input signal – and generates the appropriate output signal

❖ in a Hopfield model the first output signal is used as a new input signal etc. (until it converges to the stable state).

34

The Hopfield Model

Analysis of system energy:

Network "calculates" an error (calculate energy)

$$E = -\frac{1}{2}\sum_i (y_i - y_j^*)^2$$

E determines the value the actual network output signal **Y** differs from required signal $\mathbf{Y^*}$

Big difference – big energy. Small difference – small energy

35

The Hopfield Model

The network output signal is a function of the weights values and an input signal.

Assuming the network with two weights only – the geometrical interpretation is a surface in 3D

Each next weight increase the problem dimension.

Generally – all weights are the subject of correction which lead to multidimensional energy function

36

The Hopfield Model

Learning rule.

The network updates its weights such that the euclidean (?) distance of the output vector and the target vector is minimized minimizing the Energy E.

Learning method – a gradient descent method

A knowledge of Y^* and Y are necessary. In the Hopfield's model we do not have such a knowledge – in the consecutive steps – an algorithm has to be changed.

The Hopfield Model

For the Hopfield network the energy has the form:

$$E = -\frac{1}{2}\sum_i \sum_{j \neq i} w_{ij} x_i x_j + \sum_i x_i T_i$$

where

w_{ij} is the weight between the i^{th} and j^{th} element,

x_i is the input signal of element i^{th},

T_i is the threshold value of the element i^{th}.

and $w_{ij} = w_{ji}$ and $w_{ii} = 0$

Associative memory

Associative memories

The massively parallel models of associative or content associative memory have been developed.

Some of these models are: Kohonen, Grossberg, Hamming and widely known Hopfield model.

The most interesting aspect of the most of these models is that they specify a learning rule which can be used to train network to associate input and output patterns.

Associative memories

The associative network is a computational model emphasizing local and synchronous or asynchronous control, high parallelism, and redundancy. Such a network is a connectionist architecture and shares some common features with the Rosenblatt's Perceptron. However, that is much more powerful and flexible than the Perceptron.

Associative memory model

The model has its origin both in the Hamming and Grossberg models.
The network model is composed of 3 layers or slabs: and input layer, an intermediate layer, and an output layer. The intermediate layer is a modified totally interconnected memoryless Grossberg slab with recurrent shunting on-center off-surround subnets, whose purpose is to achieve a majority vote so that only one neuron from this level, the one with the highest input value, will send its output to the next layer.

43

Associative memory model

The similarities to Grossbergs' model: interconnections between input layer and intermediate layer
The similarities to Hammings' model interconnections (feedback) in the intermediate layer.

The connections between the input layer and intermediate layer contain all the information about one stored vector. The network is implementing the nearest-neighbor algorithm.

44

Associative memory model

The number of elements in the intermediate layer defines the number of stored patterns..
All feedback connections within the intermediate layer are based on the rule of **lateral inhibition.**
The network is performing a
winner-takes-all
operation.
The elements of input signals (and stored vectors) are the binary values 0 and 1.

$X = [x_1, x_2, x_3, ..., x_n] \qquad x_i \in \{0,1\}$

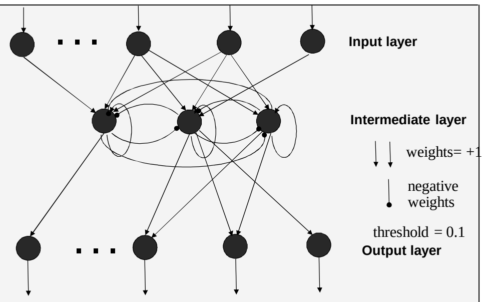
45

Associative memory model

The input and output elements (neurons) are only nodes whose purpose is to connect the inputs and outputs respectively to the intermediate slab. The network can be programmed to function as an autoassociative content-addressable memory or as symbolic substitution system which yields an arbitrary defined output for any input – it depends from the connections between the intermediate slab and the output layer.

46

Associative memory model



47

Associative memory model

<u>Programming the network</u>
The interconnections (weights) between the input elements and each intermediate neuron are independent to each other. Each intermediate element has its weights programmed to one input signal and these connections are left unchanged while the other neurons are programmed.

Adding or removing a new pattern does not influence to the existing network structure and weights.

48

Associative memory model

The connection weights between the elements of the input layer and j^{th} element of intermediate slab are:

- if the i^{th} element of the input vector is equal to zero

$$w_i^j = 0$$

- if the i^{th} element of the input vector is equal to one

$$w_i^j = \frac{1}{b_j}$$

where b_j is the number of non-zero elements in the j^{th} input vector to be stored.

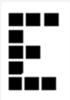
Associative memory model

This procedure normalizes the total input to each element of the intermediate slab to the interval <0;1>, and takes not account the ***relative*** number of stored elements equal to the input elements, instead of the ***absolute*** number. It allows to distinguish between signals if one is included in another one.

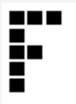
Associative memory model

Example:

pattern ❶



pattern ❷



weights of intermediate slab element where the pattern is recorded

$$w_i^j = \frac{1}{10} \qquad w_i^j = \frac{1}{8}$$

Associative memory model

1. In the input signal is ❶, the output from both elements is equal to one.
2. If the input signal is ❷, the output signal from element 1 is equal to 0.8 hence from element 2 is equal to 1.0

The ambiguous output signal in the first case can be solved by the proper network structure.

Associative memory model

This learning procedure is repeated for each input vector, each time with a new intermediate neuron.

The total number of different vectors that can be stored with this prescription in the net with n – elements in the input layer is

$$\sum_{k=1}^{n} \binom{n}{k} = 2^n - 1$$

Associative memory model

Each neuron in the intermediate slab is connected to all other neurons of this slab. The weight on the self feedback loop is equal to one, and all the other values depend on the correlation between stored vectors. The weight between the output of j^{th} neuron and input of the k^{th} neuron is given by

$$w(k,j) = \frac{1 + cor(k,j)w^k}{2(M-1)}$$

where $cor(k,j)$ is correlation (inner product) between k^{th} and j^{th} stored vectors.

w^k is one of the identical positive weight from the input slab to the k^{th} neuron,

M is equal to the number of neurons in the intermediate slab with non-zero inputs.

Associative memory model

The denominator ensures that the total lateral inhibition for the element with the greatest value is smaller that its input.
This procedure realizes the rule winner-takes-all.
The intermediate slab selects the maximum input, and drives all the other intermediate neurons to zero. If more then one intermediate neuron has the same maximum value, the slab will select the one that is less correlated to the remaining stored vectors.

55

Associative memory model

The structure of connections in the intermediate slab **is not symmetrical**

$w(k,j) \neq w(j,k),$ hence $w^j \neq w^k$

If two or more neurons will have the same input signal, and the outputs may not be discriminated by the criterion, then the slab will be unable to distinguish between them and the outputs will be driven to zero or will be a superposition of the twp or more outputs.

56

Associative memory model

Retrieval of stored vectors
At the input layer the unknown signal is applied and the network has to „recognize” it.

If the stored vectors are orthogonal, any full of or partial input corresponding to one stored vector would cause only one neuron in the intermediate slab to have a non-zero output in the first iteration. When the stored vectors are not orthogonal, a certain number of neurons will be excited.

57

Associative memory model

Let $\boldsymbol{f}$ is the unknown input signal

The elements of the vector $\boldsymbol{X}$ define the total input to the elements of the intermediate layer

$$\boldsymbol{X} = \boldsymbol{f} * \boldsymbol{W}^1$$

W^1 is the matrix of connections between the input layer and intermediate layer (the columns are equal to the input weights w_k^j of each stored vector.

58

Associative memory model

The output to of the first iteration is equal to

$$\boldsymbol{G} = \boldsymbol{W}^2 * \boldsymbol{X}^T$$

where $\boldsymbol{W}^2$ is square matrix of connections between elements of the intermediate slab

$$W^2 = \begin{bmatrix} 1 & -w(1,2) & -w(1,3) & \dots & -w(1,n) \\ -w(2,1) & 1 & -w(2,3) & \dots & -w(2,n) \\ \dots & \dots & \dots & \dots & \dots \\ -w(n,1) & -w(n,2) & -w(n,3) & \dots & 1 \end{bmatrix}$$

59

Associative memory model

the iterative formula

$$\boldsymbol{G}(\boldsymbol{t}+1) = \boldsymbol{W}^2 * \boldsymbol{G}(\boldsymbol{t}) = \left(\boldsymbol{W}^2\right)^t \left(\boldsymbol{f} * \boldsymbol{W}^1\right)^T$$

the output values are calculated by formula

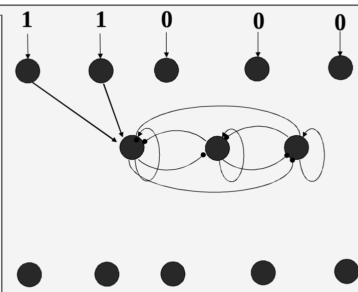
$$\boldsymbol{Y} = \boldsymbol{W}^3 * \boldsymbol{G}$$

$\boldsymbol{W}^3$ matrix of connections between the intermediate slab and the output layer; for the associative memory

$$\boldsymbol{W}^3 = \boldsymbol{W}^1$$

60

Associative memory model

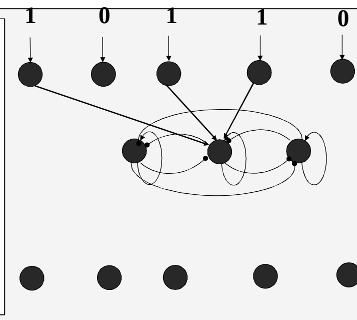


V^1=[1 1 0 0 0]

$$w_i^1 = \begin{cases} 0 & \text{for } i = 3,4,5 \\ 0.5 & \text{for } i = 1,2 \end{cases}$$

61

Associative memory model

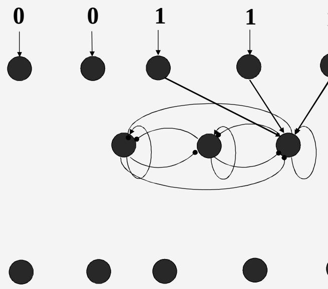


V^2=[1 0 1 1 0]

$$w_i^2 = \begin{cases} 0 & \text{for } i = 2,5 \\ 1/3 & \text{for } i = 1,3,4 \end{cases}$$

62

Associative memory model

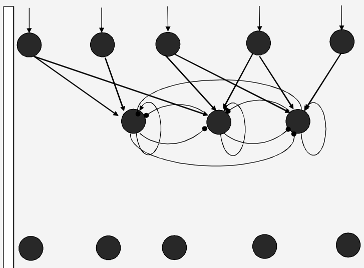


V^3=[0 0 1 1 1]

$$w_i^2 = \begin{cases} 0 & \text{for } i = 1,2 \\ 1/3 & \text{for } i = 3,4,5 \end{cases}$$

63

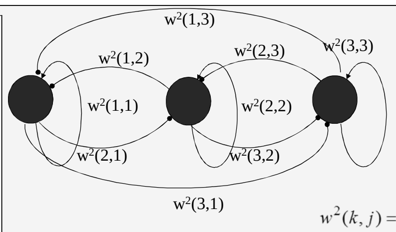
Associative memory model



$$\boldsymbol{W}^1 = \begin{bmatrix} 1/2 & 1/3 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/3 & 1/3 \\ 0 & 1/3 & 1/3 \\ 0 & 0 & 1/3 \end{bmatrix}$$

64

Associative memory model



$$w^2(j,j) = 1$$

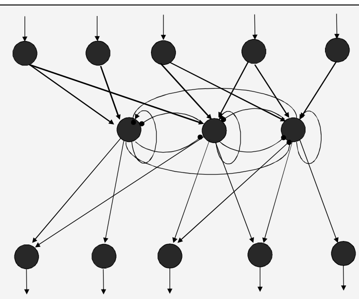
$$w^2(k,j) = -\frac{1 + cor(k,j)w^k}{2(n-1)}$$

$$\boldsymbol{w}^k = \boldsymbol{w}_j^k > 0$$

$$\boldsymbol{W}^2 = \begin{bmatrix} 1 & -3/4 & -1/2 \\ -2/3 & 1 & -2/3 \\ -1/2 & -2/3 & 1 \end{bmatrix}$$

65

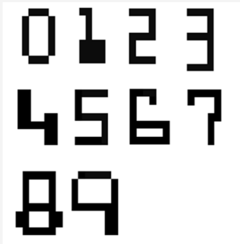
Associative memory model



66

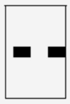
Associative memory model

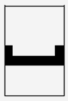
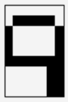
2D patterns stored in the network (9x6)



Associative memory model

Input signal | Output signal

 after 3 iterations 

 after 4 iterations 

Logic operations with neural networks

Logic networks

Most publications on neural networks focus on pattern recognition and associative memories. Here will be presented new area – logic operations. A multilayer system composed of simple identical elements **can perform any Boolean function of two, three or more variables.**

Logic networks

Long ago, M. Minsky and S. Pappert describing perceptron, or rather describing its faults used the **XOR** function as the example of operation cannot be performed by the one-layer perceptron.
This simple logical function can be realized on many ways

Logic networks

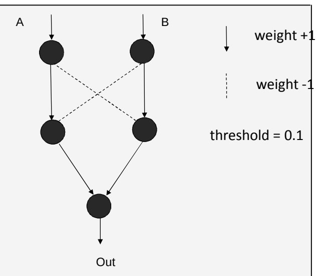
Example
The connections with an arrow have the positive weight equal +1, connections without arrows have the weights equal to -1. All elements are identical, with the nonlinear characteristics and threshold equal to 0.1. Input signals components are equal to one or zero.

73

Logic networks

Example of the network able to perform XOR operation

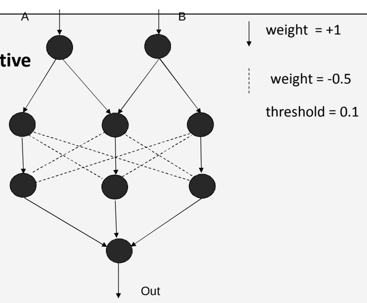
A	B	Wy
0	0	0
1	0	1
0	1	1
1	1	0



74

Logic networks

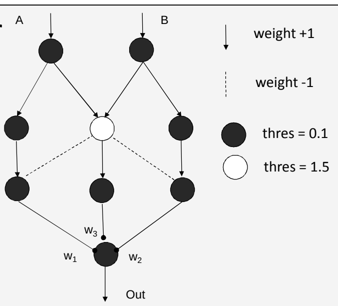
Model of associative memory type



75

Logic networks

Logic module for many functions



76

Logic networks

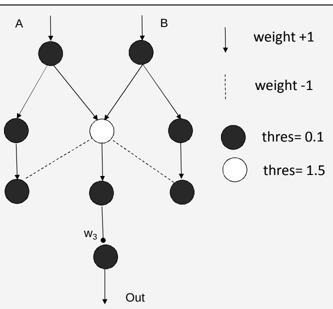
Examples of functions

function	w_1	w_2	w_3
A OR **B**	1	1	1
A AND **B**	0	0	1
A XOR **B**	1	1	0
A AND (NOT **B**)	1	0	0
A AND (**B** OR (NOT **B**))	1	0	1

77

Logic networks

Example
A and **B**



78

Logic networks

Logical operations
For *n* logical variables one can creates 2^{2^n} different functions.

number of variables n	number of functions of n variables
1	4
2	16
3	256
4	65 536

79

Logic networks

Any logical function can be written in ***a canonical form***.
The canonical form: An expression is said to be in a canonical *sum-of-product* form when variables are logically ANDed into groups (called minterms), that are logically ORed to form a function.
Every variable appears in every minterm once in the canonical sum-of-product form. All 2^n minterms of n variables can be generated in a network of *n*+1 levels, and the minterm can be combined into arbitrary function in an additional level..

80

Logic networks

Functions of two variables

The Canonical form

$$f = \overline{A}\,\overline{B}f_0 + \overline{A}Bf_2 + A\overline{B}f_1 + ABf_3$$

81

Logic networks

Table of 16 possible two-element logical operations

	Function	Descr	coefficients			
			f_0	f_1	f_2	f_3
1	$\overline{A}\,\overline{B}$	NOR	1	0	0	0
2	$A\overline{B}$		0	1	0	0
3	$\overline{A}B$		0	0	1	0
4	AB	AND	0	0	0	1

82

Logic networks

Table of 16 possible two-element logical operations

	Function	Descr	coefficients			
			f_0	f_1	f_2	f_3
5	$\overline{A}\,\overline{B} + A\overline{B}$	~B	1	1	0	0
6	$\overline{A}\,\overline{B} + \overline{A}B$	~A	1	0	1	0
7	$A\overline{B} + AB$	A	0	1	0	1
8	$\overline{A}B + AB$	B	0	0	1	1

83

Logic networks

Table of 16 possible two-element logical operations

	Function	Descr	coefficients			
			f_0	f_1	f_2	f_3
9	$\overline{A}B + A\overline{B}$	XOR	0	1	1	0
10	$A\overline{B} + \overline{A}B + AB$	OR	0	1	1	1
11	$\overline{A}\,\overline{B} + AB$	~XOR	1	0	0	1
12	$\overline{A}\,\overline{B} + A\overline{B} + AB$	A+~B	1	1	0	1

84

Logic networks

Table of 16 possible two-element logical operations

	Function	Descr	coefficients			
			f_0	f_1	f_2	f_3
13	$\overline{A}\,\overline{B}+A\overline{B}+\overline{A}B$	NAND	1	1	1	0
14	$\overline{A}\,\overline{B}+\overline{A}B+AB$	~A+B	1	0	1	1
15	out always = 0	FALSE	0	0	0	0
16	out always = 1	TRUE	1	1	1	1

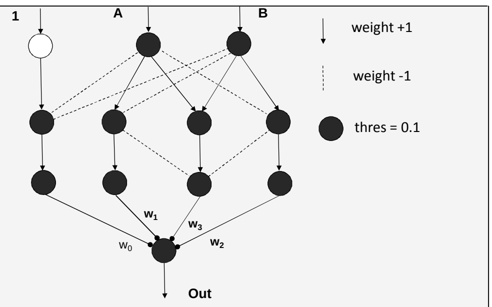
Logic networks

Any out of 16 two-element logic operations can be programmable by a universal logic module.

Model assumptions:

- Input signals ar equal to 1 or 0.
- Connections with arrow are equal to +1.
- Connections without arrows are equal to -1.
- The element shown white is always activated by the input signal equal to +1.

Logic networks – Universal logic module



Logic networks

Description of network operation

The network input signal IN = [1,A,B,]

Input signal to the elements of the 1st intermediate layer X = IN * W^1

W^1 matrix of connections between input elements and elements of the 1st intermediate layers

$$W^1 = \begin{bmatrix} +1 & 0 & 0 & 0 \\ -1 & +1 & +1 & -1 \\ -1 & -1 & +1 & +1 \end{bmatrix}$$

Nonlinear threshold function Φ $\hat{X} = \Phi(X) = \begin{cases} 1 & \text{for } x_i > 0 \\ 0 & \text{for } x_i \le 0 \end{cases}$

Logic networks

Description of network operation

Input signal to the element of the 2nd intermediate layer Y = $\hat{X}$ * W^2

W^2 matrix of connections between elements of the 1st and 2nd intermediate layers

$$W^2 = \begin{bmatrix} +1 & 0 & 0 & 0 \\ 0 & +1 & -1 & 0 \\ 0 & 0 & +1 & 0 \\ 0 & 0 & -1 & +1 \end{bmatrix}$$

Nonlinear threshold function Φ

$$\hat{Y} = \Phi(Y)$$

Logic networks

Description of network operation

Network output signal OUT = Φ($\hat{Y}$ * W^3)

W^3 matrix of connections between the elements of the 2nd intermediate layer and the output element

$$W^3 = \begin{bmatrix} w_0 & w_1 & w_3 & w_2 \end{bmatrix}$$

Finally, for the network

$$OUT = \Phi\left\{\Phi\left[\Phi(IN * W^1) * W^2\right] * W^3\right\} =$$
$$= \Phi\{\Phi(1 - A - B)w_0 + \Phi(A - B)w_1 + \Phi(B - A)w_2 +$$
$$\Phi[\Phi(A + B) - \Phi(A - B) - \Phi(B - A)]w_3\}$$

Logic networks

Example

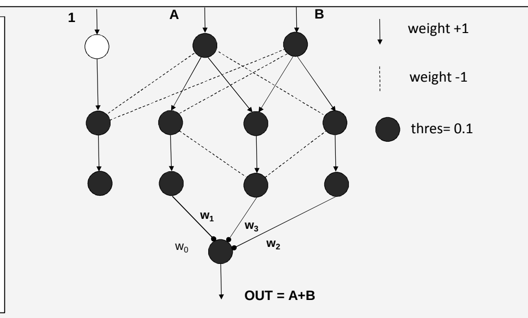
The well-known operation **OR (A+B)** using logical theorems (expansion, distributive, commutative, De Morgan's etc)**,** can be rewritten into a canonical form

$$\mathbf{A+B=A(B+\overline{B})+B(A+\overline{A})=}$$
$$\mathbf{=AB+A\overline{B}+BA+B\overline{A}=}$$
$$\mathbf{=AB+A\overline{B}+\overline{A}B}$$

The universal logic module can perform this operation by setting of weights

$w_0 = 0 \quad w_1 = 1 \quad w_2 = 1 \quad w_3 = 1$

Logic networks – Universal logic module



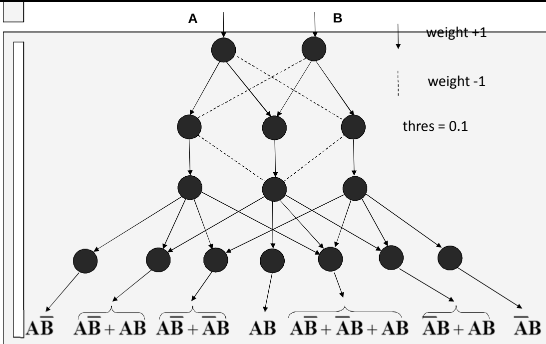
Logic networks

The other solutions

By replacing a single output element by a layer of 2^{2^n} elements (for n=2 by 16 elements), and by fixing the interconnections to the output layer we get the network where each output element corresponds to one logical function. Each element of the second intermediate slab reacts to only one term of the canonical form of a logical function.

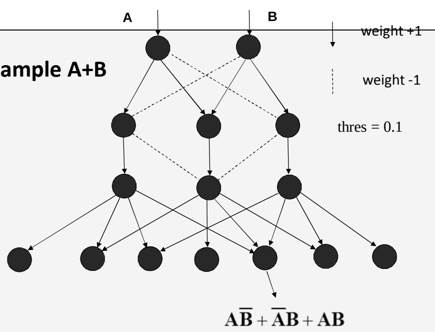
A simplified version of the network which can perform seven of all 16 two element logical operations, neglecting only those for which a total zero input lead to a non-zero output is shown.

Logic networks – Universal logic module



Logic networks – Universal logic module

Example A+B



Logic networks – Universal logic module

Example

A XOR B