



WARSAW UNIVERSITY OF TECHNOLOGY  
FACULTY OF MATHEMATICS  
AND INFORMATION SCIENCE



# Neural Networks

Lecture 4



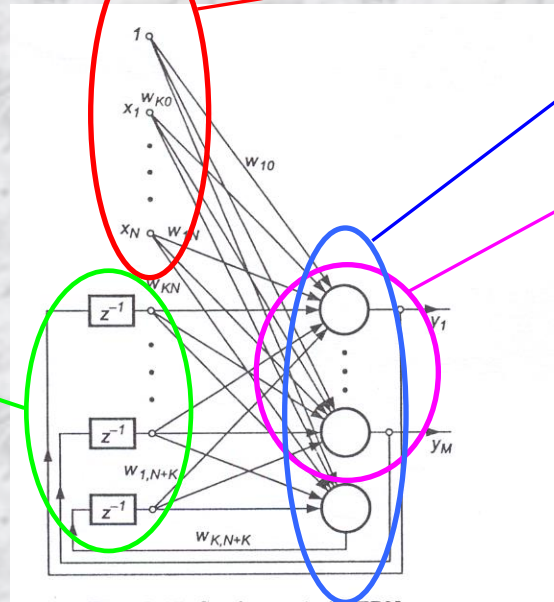
***Real Time Recurrent  
Network (RTRN)***

# RTRN Network

RTRN Network (Real Time Recurrent Network) is the network for real-time signal processing

Network structure

$K$  element context layer



$N$  - number of inputs  
 $K$  - number of neurons in hidden layer  
 $M$  - number of output neurons (from the  $K$  hidden elements)

# RTRN Network

$$u_i(t) = \sum_{j=0}^{N+K} w_{ij}^{(1)} x_j(t)$$

weighted input signal of  $i$ -th neuron in hidden layer (12)

$$y_i = f(u_i(t))$$

output signal from  $i$ -th neuron in hidden layer (13)

Vector input signal  $\mathbf{x}(t)$  and delayed one cycle vector  $\mathbf{y}(t-1)$  create the excitation network signal

$$[1, x_1(t), \dots, x_2(t), x_N(t), y_1(t-1), \dots, y_K(t-1)]$$
 (14)



# RTRN Network

This is the special case of the Elman network with constant values of output weights ( $w_{\alpha}^{(2)}$ ) and defined by

$$w_{ij}^{(2)} = \delta_{ij} = \begin{cases} 1 & \text{for } i = j \\ 0 & \text{for } i \neq j \end{cases}$$

Modifying the Elman's algorithms one obtain the new learning algorithm for that network

# RTRN Network

## RTRN training algorithm

1. Initialization of weights (usually uniform distribution from  $[-1;1]$ ).
2. Calculation of neurons' state in  $(t=0,1,\dots)$  (signals  $u_j$  and  $y_j$  formulas (12) and (13) and next vector excitation signal (14).

3. Calculation of  $\frac{dy_j(t)}{dw_{\alpha\beta}} = 0$  according to

$$\frac{dy_j(t)}{dw_{\alpha\beta}} = \frac{df(u_j)}{du_j} \left[ \delta_{j\alpha} + \sum_{t=1}^K \frac{dy_j(t-1)}{dw_{\alpha\beta}} w_{j,t+N} \right]$$

# RTRN Network

## RTRN training algorithm

4. Upgrades of weights according to steepest descent algorithm according to

$$w_{\alpha\beta}(t+1) = w_{\alpha\beta}(t) - \eta \sum_{j=1}^M [y_j(t) - d_j(t)] \frac{dy_j(t)}{dw_{\alpha\beta}}$$

for  $\alpha=1,2,\dots,K$ ,  $\beta=0,1,2,\dots,N+K$

5. Go to step 2.

# ***Radial Networks***



# Network structure

---

## Radial networks

For special purposes sometimes we are using the radial neurons i.e. neurons with *Radial Basis Function RBF*. They have non typical aggregation methods of input data, they uses non typical transfer function (Gauss function) and they learned by the special way.

Data aggregation consist on the calculation of a distance between an input signal (vector  $\mathbf{X}$ ), and established in a learning process centroid of a certain set  $\mathbf{T}$ .

# Network structure

## Sigmoid and radial neuron

Sigmoidal neuron represents in the multidimensional space hyperplane separating space into two categories (Fig.A). Radial neuron represents hypersphere performing circular separation around the central point (Fig.B).



Fig. A

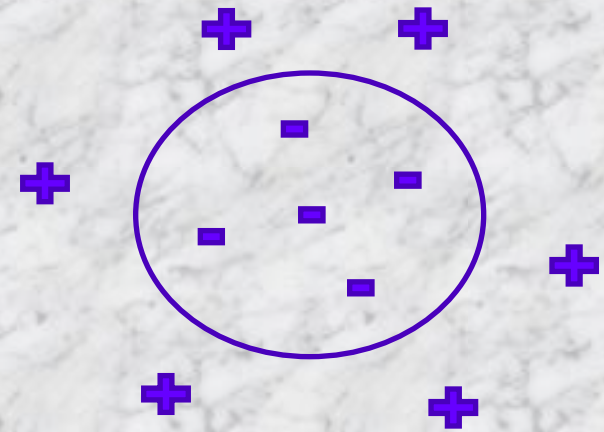


Fig. B

# Radial basis function network

---

*Radial basis function network (RBF)* uses radial basis functions as activation functions typically have three layers: an input layer, a hidden layer with a non-linear **RBF** activation function and a linear output layer.

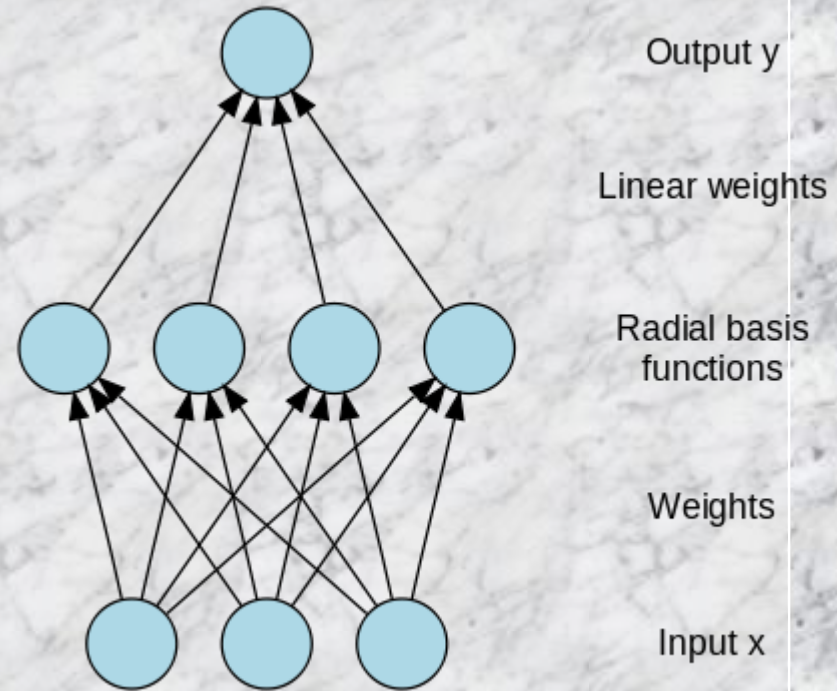
Functions that depend only on the distance from a center vector are radially symmetric about that vector, hence the name *radial basis function*. In the basic form all inputs are connected to each hidden neuron. The norm is typically taken to be the Euclidean distance and the radial basis function is commonly taken to be Gaussian.

# Radial basis function network

The output of the network is then a scalar function of the input vector, and is given by

$$\varphi(\mathbf{x}) = \sum_{i=1}^N a_i \rho(\|\mathbf{x} - \mathbf{c}_i\|)$$

where  $N$  is the number of neurons in the hidden layer,  $\mathbf{c}_i$  is the center vector for neuron  $i$ , and  $a_i$  is the weight of neuron  $i$  in the linear output neuron. Functions that depend only on the distance from a center vector are radially symmetric about that vector. The radial basis function is commonly taken to be Gaussian.



$$\rho(\|\mathbf{x} - \mathbf{c}_i\|) = \exp[-\beta \|\mathbf{x} - \mathbf{c}_i\|^2]$$





***Kohonen Self - Organizing  
Networks***



# Kohonen Networks

---

In the previous part we discussed *a supervised learning* techniques based among others on the back propagation technique. Now we will speak about *unsupervised learning methods*, and in particular Kohonen's self organizing maps. The idea of self - organization was proposed in 1973 by von der Malsburg and was based on close studies of the topology of the brain's cortex region.

# Kohonen Networks

---

It is well known that learning or adaptation is the chemical process changing the effectiveness of the synaptic connections at the cell input.

The self - organization network has two main assumptions:

- the input patterns that share common features belong to the same class,
- the network will be able to identify common features across the range of input patterns.

# Kohonen Networks

---

Kohonen used the idea that the brain uses spatial mapping to model complex data structures internally.

It allows him to perform data compression on the vectors to be stored in the network, using a technique known as vector quantization.

Data compression means that multi - dimensional data can be represented in a much lower dimensional space. The implementation of Kohonen's is two - dimensional.

# Kohonen Networks

---

The perceptron was the network model where neurons were acting independently. Now, we will speak about system performing *feature maps*, generalizing the self organizing process by means of geometrical organization mutually competing cells.



# Kohonen Networks

---

Two types of a learning by competition:

**winner-takes-all**

or

**winner-takes-most**



# Kohonen Networks

---

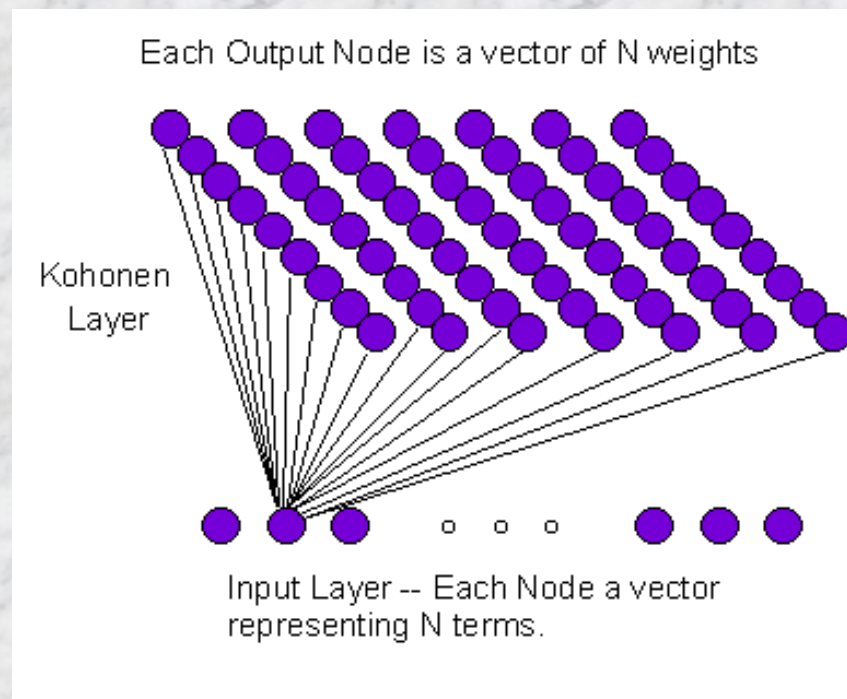
Kohonen uses the idea that some regions in a brain are responsible for certain functions – i.e. used the von der Malsburg model

The part of a one-layer, two-dimensional Kohonen network with the connections of between only two input elements and the neurons are shown ( in practice all input are connected to all nodes).

# Kohonen Networks

The neurons are not arranged in layers (as in the multilayer perceptron) but in the flat grid. Feedback is restricted to lateral interconnections to intermediate neighboring nodes

Note also, that there is no special output layer - each of the elements is itself and output element.



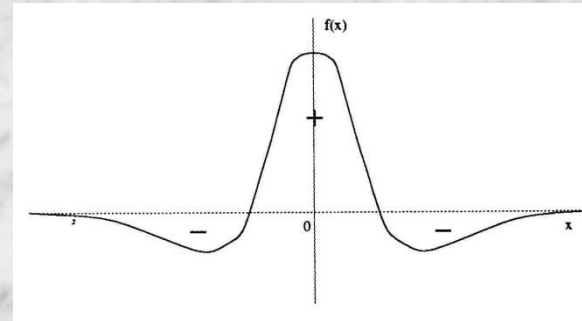
# Kohonen Networks

---

We have seen from the previous lectures that activation in a nervous cell is propagated to other cells via axon links – which may have an inhibitory or excitatory effect at the input of another cell. However, we have not considered the question of how the axon links are affected by lateral distance from propagating neuron.

# Kohonen Networks

A simplified model of the effect is illustrated by the *Mexican hat* function



Cells physically close to the active cell have strongest links. Those of certain distance even switch to inhibitory links. Kohonen modeled this effect by using only locally interconnected networks and restricting the adaptation the weights values to localized "neighborhoods".



# Kohonen Networks

---

The other Kohonens' assumptions !!

- input signals are normalized (i.e.  $|X| = 1$ )
- the element to be learned is selected by the special rule (see algorithm),
- the weights of elements in the neighborhood of this selected neuron are also modified



# Kohonen Networks

---

**The idea of neighborhood is introduced and defined.**

The examples of neighborhood:

- 1-D chain with numbered elements
- 2-D array with elements indexed by  $a(i,j)$ .

The location (distance) of neighbor elements differs less than a certain value.

# Kohonen Networks

For 1-D the function defining the distance between the  $i^{\text{th}}$  and  $j^{\text{th}}$  element

$$h(i, j) = \begin{cases} 1 & \text{dla } i = j \\ 1/2 & \text{dla } |i - j| = 1 \\ 0 & \text{dla } |i - j| > 1 \end{cases}$$

or  $h(i, j) = 1/\rho(i, j)$

where  $\rho(i, j)$  is the distance between elements

or  $h(i, j) = \exp[-\rho^2(i, j)]$ .

# Kohonen Networks

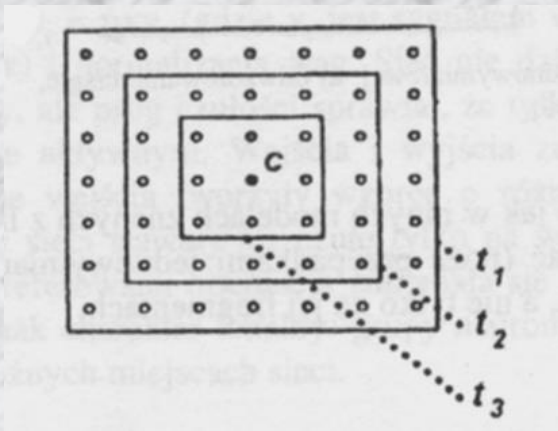
---

Two proposals of neighborhood:

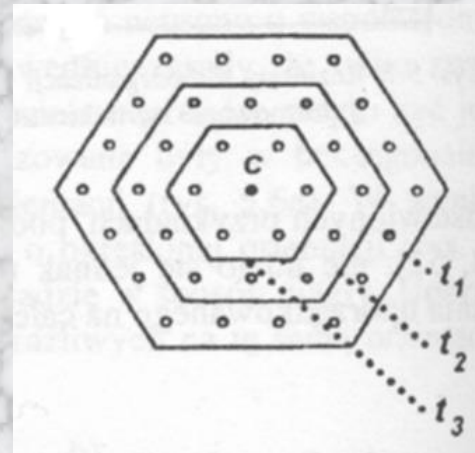
- a) rectangular grid
- b) hexagonal grid

Kohonen introduce the idea that number of neurons surrounding the winning node and size of the neighborhood is reduced with time during the training sequence to its final size.

# Kohonen Networks



rectangular grid



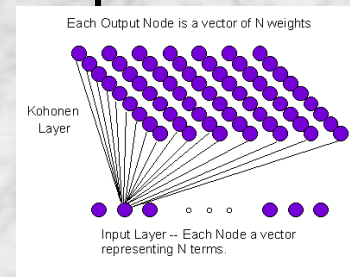
hexagonal grid



# Kohonen Networks

## Algorithm description

At the beginning all weights are set to be usually small random values. Each node have the unique weight vector, the dimensionality of which is defined by the number of components in the input vector. During the learning cycle, a set of training patterns is shown tom the network. Comparison is made between each input pattern and the weight vectors.



# Kohonen Networks

---

The node with the weight vector closest to the input pattern is selected to as the „winner“. This node modifies its own weight vector to align with the input. The node has now become more sensitive to the particular training input and will provide maximum response from the network if this input is applied again.

Also the nodes in the neighborhood of the winning node are also modified.

# Kohonen Networks

---

The network is trying to create regions that will respond to a spread of values around the training input. The nodes around are given similar alignment.

**As the result, vectors that are close spatially to the training values will still be classified correctly. This demonstrates generalization properties of the network.**

# Kohonen Networks

The change of the connection weight between the  $j^{\text{th}}$  input element and the  $i^{\text{th}}$  Kohonen element (in the time  $t$ ) is defined by

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)h(i, j)[x_i(t) - w_{ij}(t)]$$

where  $w_{ij}(t)$  is the connection weight of the  $j^{\text{th}}$  input with the  $i^{\text{th}}$  neuron

$\eta$  is the learning rate coefficient



# Kohonen Networks

---

The learning rate coefficient  $\eta$  (unit of proportionality) decreases the adaption rate with time (where „time“ means the number of passes through the training set).

The training process attempts to cluster the nodes on the topological map to reflect the range of class types found in the training data. At the beginning the adaptation is kept high ( $>0.5$ ) and is reduced as training progresses. Typically fine tuning stage will take between 100 and 1000 times as many steps as finding the coarse representation.

# Kohonen Networks

---

The training algorithm will produce clusters for all the class types found in the training data. The ordering of the clusters on the map and the convergence times for training are dependent on the way the training data are presented to the network.

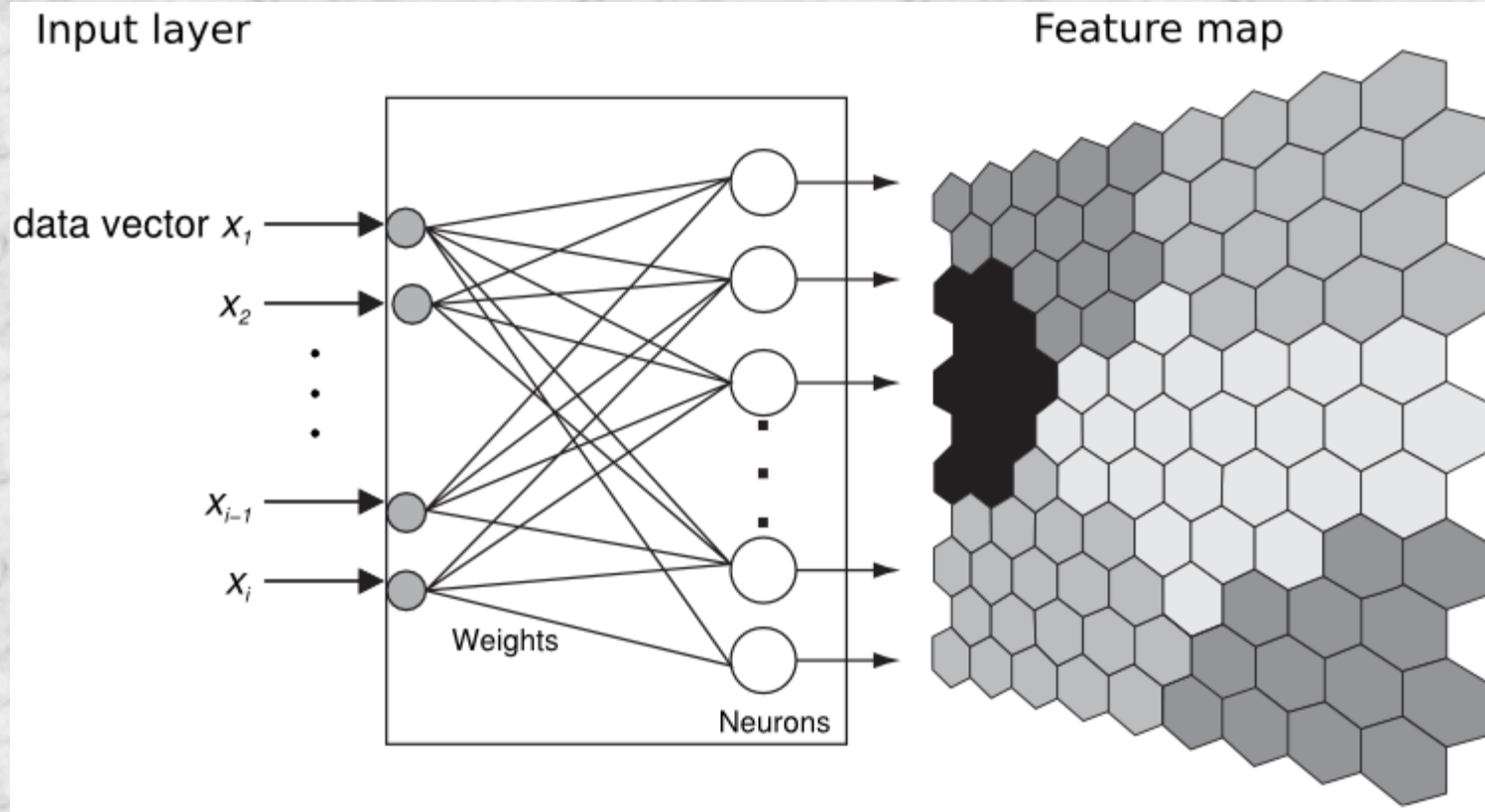
Once the network has self-organized the internal organization the clusters can be labeled to indicate their class so that the network can be used to classify unknown inputs.

# Kohonen Networks

---

The network forms the internal features without supervision, but the classification labeling must be done by hand, once the network is fully trained.

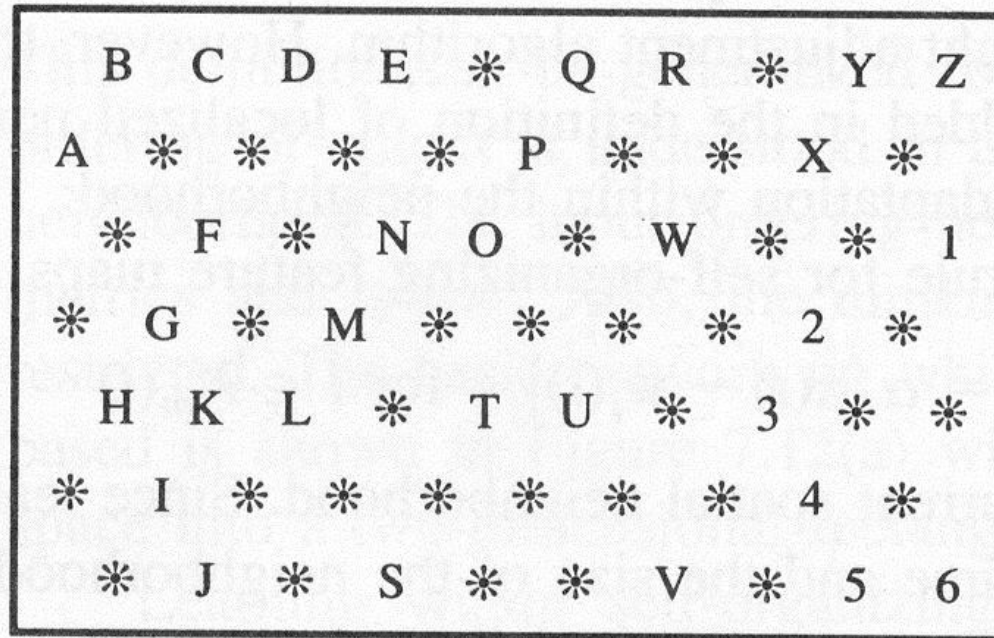
# Kohonen Networks



Feature map produced after training



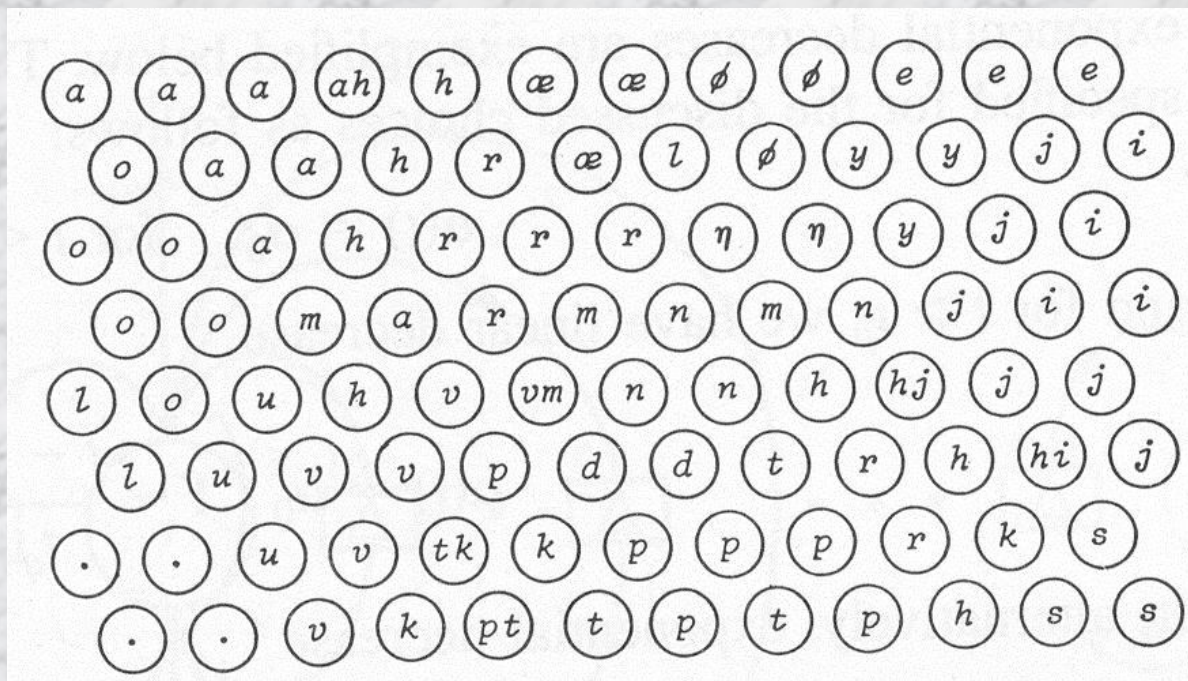
# Kohonen Networks



Feature map produced after training

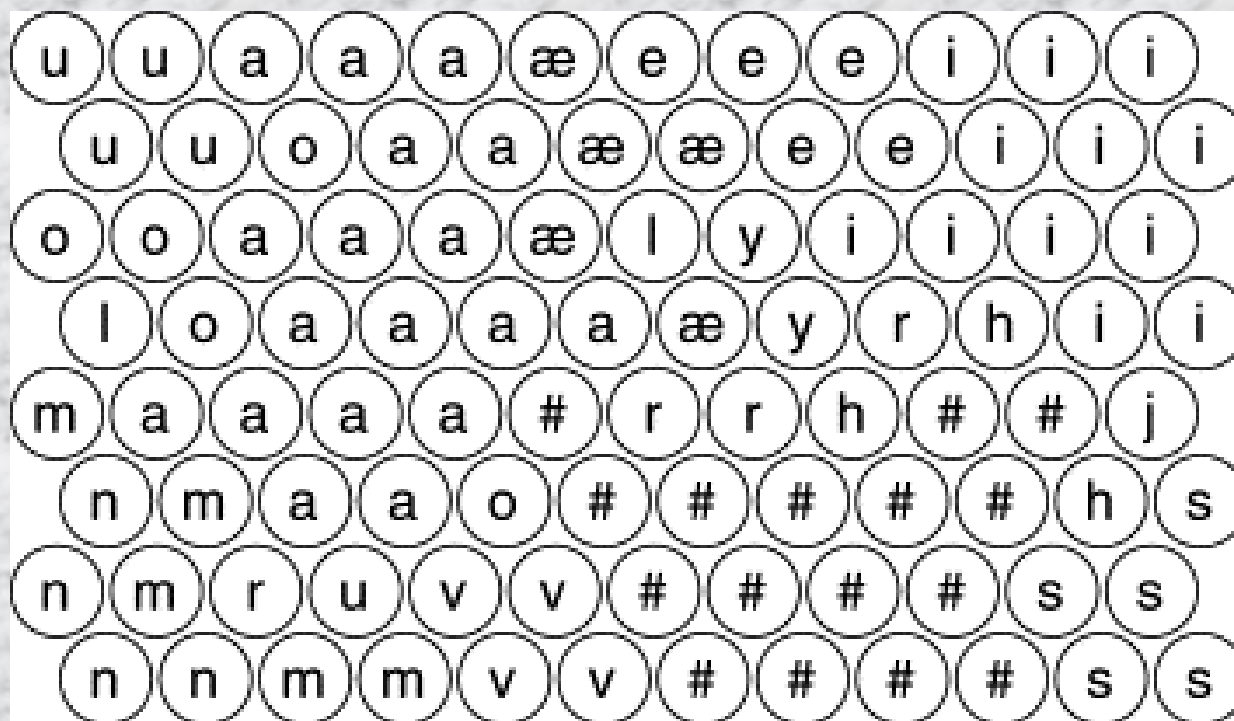
# Kohonen Networks

A phoneme typological feature map. The network was trained on voice data (Finish language)




# Kohonen Networks

A characters' recognition typological feature map.







**We'll take a  
5-minute  
break now**





***Model ART***

# Adaptive Resonance Theory

---

The problem of stability is very important in a self-organizing systems. Most of neural networks models faces the problem known as the *stability-plasticity dilemma*.

Stability means that already learned templates will not be affected by the new data.

Plasticity means ability to accept new data.

# Adaptive Resonance Theory

---

The network unable to learn new information on the top of old is the great problem. In a multilayer perceptron trying to add a new template to an already trained network may destroy all previous learning by interfering with weight values.

# Adaptive Resonance Theory

---

Decreasing the learning coefficient (i.e. influence of successive input signals) the already existing classes can be locked up. But such a mechanism yields lack of plasticity – ability to react for the new data.



# Adaptive Resonance Theory

---

**Stability** - vs- **plasticity** it was Grossbergs' dilemma, how both can be achieved simultaneously?

Stability problem is connected with the number of output elements (the number of classes). If this number is fixed there is no possibility to create the new classes for new templates.

# Adaptive Resonance Theory

---

When the number of classes is not limited it yields to the very fine resolution of the space of input signals up to the limit

1 template = 1 class

One of possible solution is the possibility to add new output elements to create the new stable class (category).

# Adaptive Resonance Theory

---

The major feature of ART model, proposed by Gail Carpenter and Stephen Grossberg from MIT is the ability to switch modes between plastic (the learning state where the internal parameters can be modified) and stable (a fixed classification set), without detriment to any previous learning.

# Model A R T – Authors

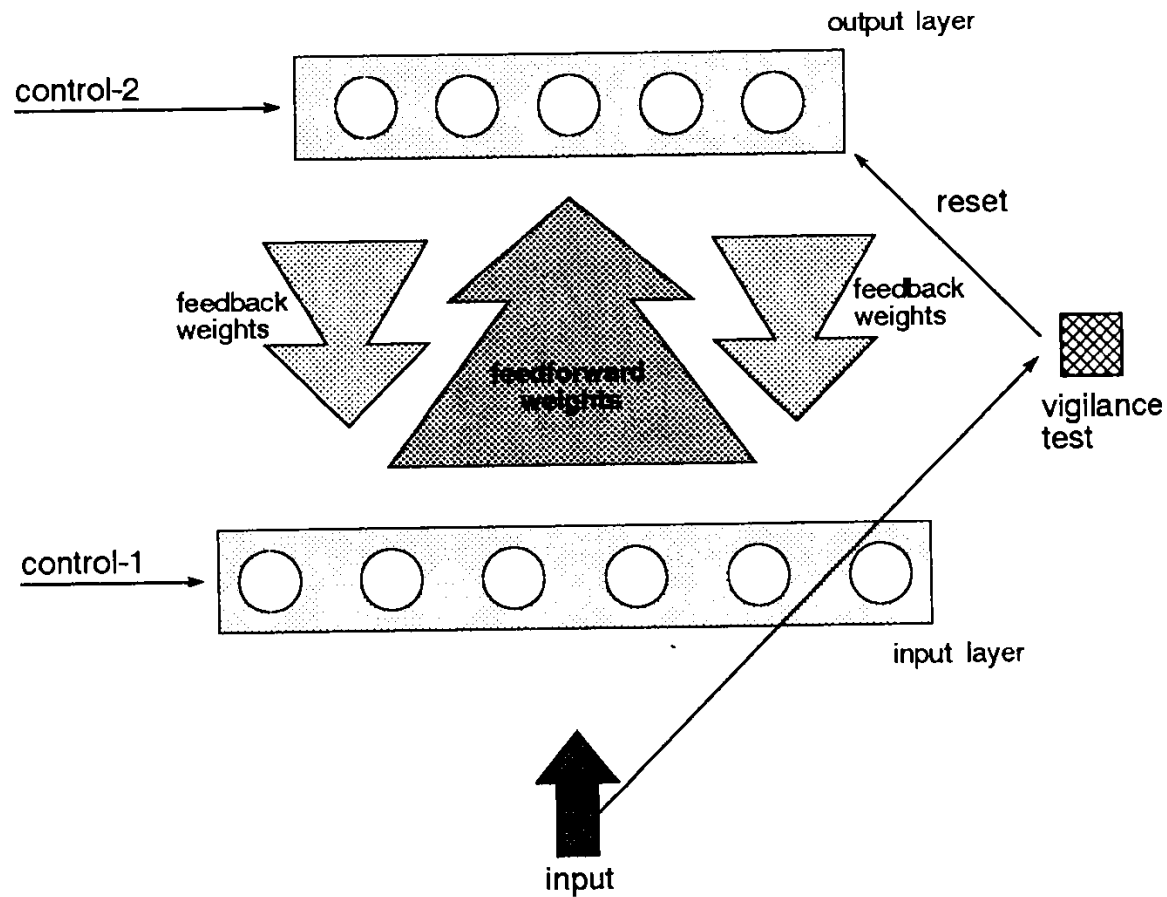
---

- Stephen Grossberg – professor of mathematics, psychology and biomedical engineering
- Gail Carpenter – professor of mathematics





# Model ART – network description



# Adaptive Resonance Theory

---

The concept of the network **ART** –  
**A**daptive **R**esonance **T**heory –

**The model main idea:** *two signals - input and memorized template are in a resonance if they are sufficiently similar.*

If the input signal is not similar to any of memorized templates (it can not be included into existing classes) – the new class, new category is formed – and its template is this signal. The new output element is added.

# Adaptive Resonance Theory

---

When an input learning vector is presented:

- If the network has learned it previously, a resonant state is achieved
- quickly.
- Otherwise, it searches to the stored patterns for a match.
- If no match is found, a new pattern is stored.
- The previously learned patterns remains without change

# Adaptive Resonance Theory

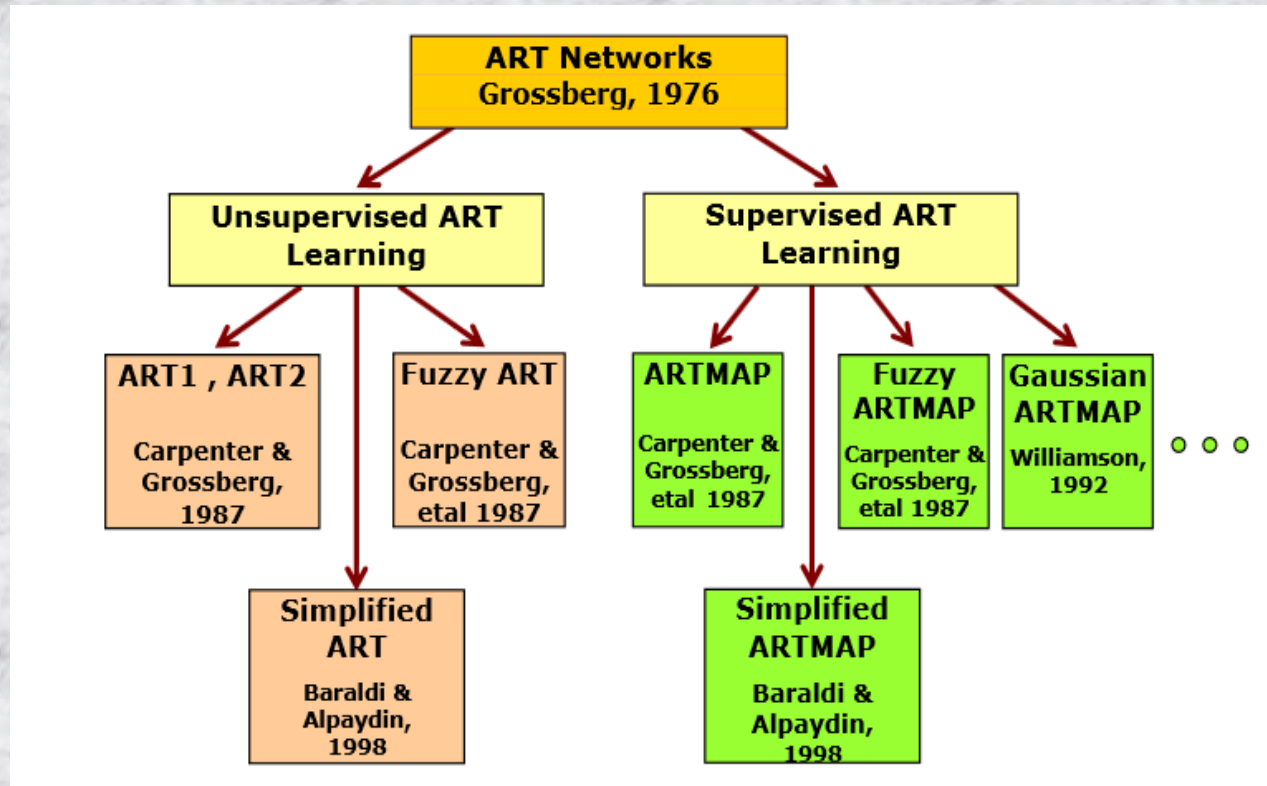
---

The similarity depends of the *vigilance threshold*  $\rho \in \langle 0;1 \rangle$ .

This parameter controls the resolution of the classification process. A low threshold ( $<0.4$ ) will produce a low resolution creating fewer class types. A high vigilance threshold (tending to 1) will produce a very fine resolution classification, meaning that even slight variations between input patterns will force a new class to be made.



# Important ART Networks



# Model A R T – network description

---

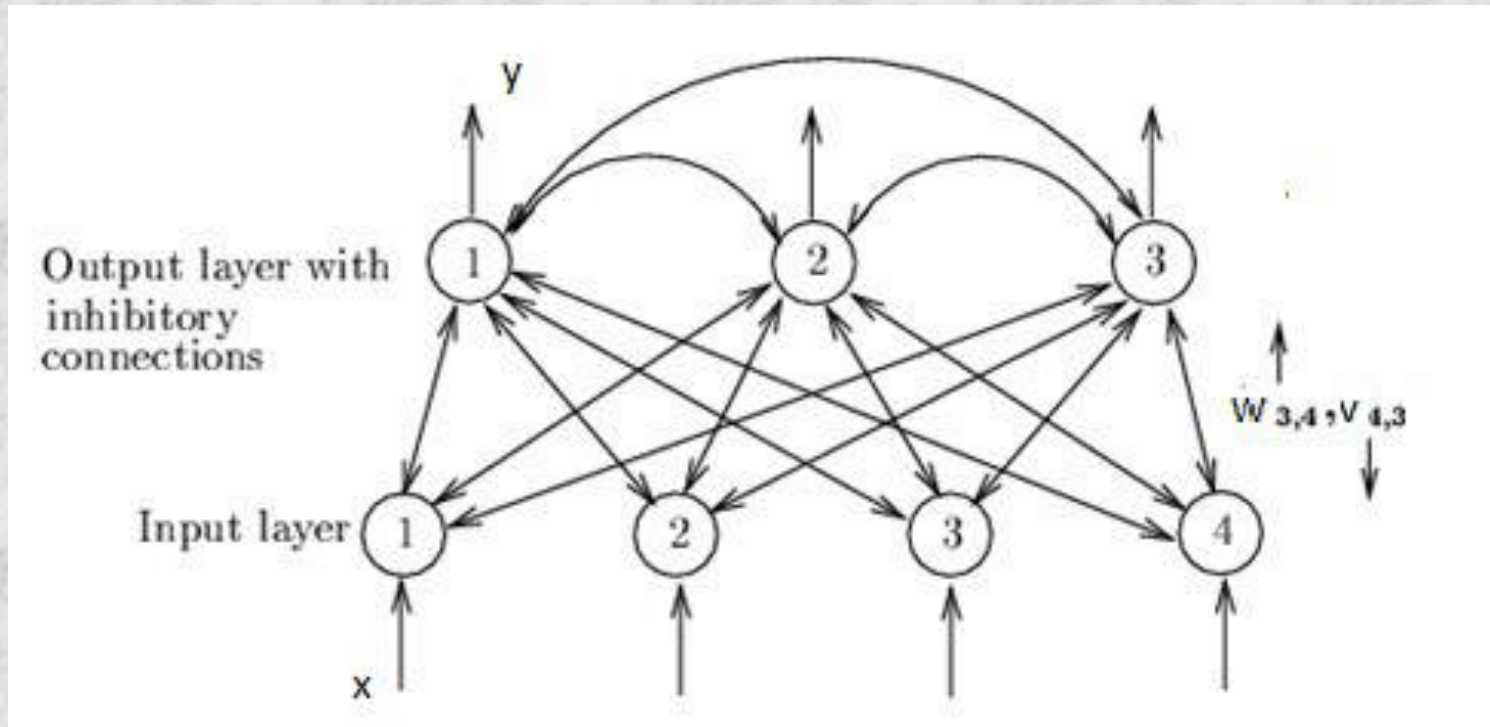
## The ART network has two layers:

The first is the *input–comparison* layer,

The second is the *output–recognition (category)* layer.

These layers are connected together with *feedforward connections* from input layer to the output layer, *feedback connections* – from the output layer to the input layer. There are also connections between the nodes of the output layer as *lateral inhibition*.

# Model A R T – network description



Input

# Model A R T – network description

---

The ART network has *feedforward* weight matrix from the input layer to the output layer and *feedback* weight matrix from the output to the input layer. These paths will be marked **W** and **B** respectively.

For each layer there are also logic control signals that control the data flow through layers at each stage of network operation: *control-1* and *control-2*.



# Model A R T – network description

---

***Control-1*** determines the course of data flow for the input layer – its' binary value toggles the first layer of nodes between two modes: *input* and *comparison*.

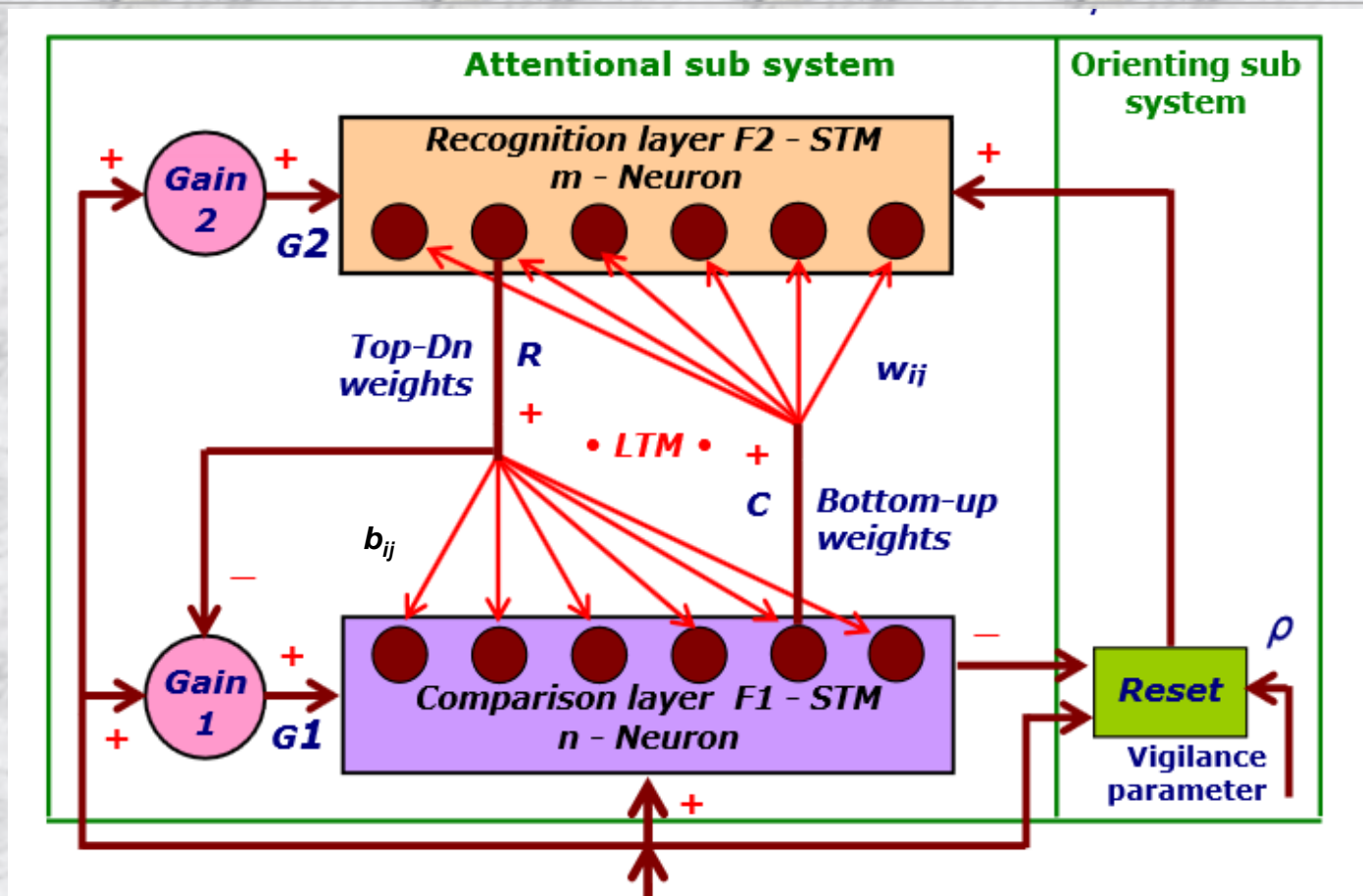
The state of control-1 is *one* whenever a valid input is presented to the network but is forced to *zero* if any node in the recognition (output) layer is active.

# Model A R T – network description

---

The binary value of ***Control-2*** enables or disables the nodes in the recognition (output) layer. It is *one* for any valid input pattern but *zero* after a failed vigilance test (this disables the recognition layer nodes and resets their activation levels to zero).

# Model A R T – network description



# Model A R T – network description

---

Between the input and output layers there is also a *reset circuit*. It performs a reset function for output nodes – it is actually responsible for comparing the input signals to a vigilance threshold that determines whether a new class pattern should be created for an input pattern.



# ART1 – the phases

---

There are several phases to learning or classification in an ART network.

The main paradigm is that continually modified input vector is passed forward and backward between the layers in a cyclic process.

The network action and activity of separate layers can be described.

The phases: *an initialization phase, a recognition phase, a comparison phase* and *a search phase.*

# ART1

## 1. The initialization phase.

The weight vectors **W** and **B** must be initialized.

The feedback links (**B**) are all set to one, so every node in the output layer is initially connected to every node in the input layer.

The feedforward links (**W**) are set to constant value, for example

$$w_i = \frac{1}{1+n}$$

where  $n$  is the number of input nodes.

The vigilance threshold is set in a range  $0 < \rho < 1$ .

# ART1

---

## 2. The recognition phase.

In this phase the input vector is passed through the network and its value is matched against the classification represented at each node in the output layer. The neuron with largest input has weights that best match the input vector. It wins the competition.

# ART1

---

Each weight vector  $\mathbf{W}$  at each recognition node (output) is a „*stored template*“, or exemplar class. The input vector is compared to the exemplar at each node and the best match is found. The best match comparison is done by the dot product of the input vector and a node's weight vector. The node with closest weight vector will yield the largest result.

(of course several nodes in the output layer may have high level of activation).



# ART1

---

The lateral inhibition between the nodes will turn off each node except the maximum response node – only this node remain active in the output layer. This node passes its stored pattern ***B*** (the class exemplar) back to the comparison (input) layer.

# ART1

---

## 3. The comparison phase

Two vectors are present at the input layer where each node has three inputs. A comparison between the input vector  $\mathbf{X}$  and the  $\mathbf{B}$  is done and if degree of similarity is less than vigilance parameter the network causes reset. The effect of the reset is to force the output of firing neuron in the recognition layer to zero, disabling it for the duration of current classification.

## 4. The search phase

$S < \rho$  means that the node was the best match, but the classification was wrong.

If there is no reset signal generated, the match is adequate and the classification is finished. Otherwise, the other stored pattern must be researched to seek a correct match. This process repeats, until one of the two events occurs.

If no such node is found the network declares the input vector an unknown class and allocates it to a unused node in the output layer.

# ART1 Algorithm Description

**1 Step.** Initialize

$$b_{ij}(0) = 1$$

$$w_{ij}(0) = 1/(1+n)$$

$$0 \leq i \leq n-1, 0 \leq j \leq M-1$$

Set  $\rho$  where  $0 \leq \rho \leq 1$

where  $t_{ij}(\tau)$  is the top-down and  $w_{ij}(\tau)$  is the bottom-up connection weight between node  $i$  and node  $j$  at time  $\tau$ .  $\rho$  is the vigilance threshold which determines how close an input has to be to correctly match a stored exemplar;  $M$  number of output nodes,  $n$  number of input nodes.

all output elements are enable.



# ART1 Algorithm Description

**Step 2.** Apply new input

**Step 3.** Computing matching

$$\mu_j = \sum_{i=1}^N w_{ij}(\tau) x_i$$

$$i = 0, 1, 2, \dots, M-1, \quad j = 0, 1, 2, \dots, N-1$$

$\mu_j$  is the output node  $j$  and  $x_i$  is element of the input which can be either 0 or 1.

**Step 4.** Select best matching element

$$\mu_{j^*} = \max_j [\mu_j]$$

# ART1 Algorithm Description

## Step 5. Test

$$\|\mathbf{X}\| = \sum_i \mathbf{x}_i$$

$$\|\mathbf{BX}\| = \sum_i \mathbf{b}_{k^*i}(\tau)\mathbf{x}_i$$

$$i = 0, 1, 2, \dots, N-1$$

$$\frac{\|\mathbf{BX}\|}{\|\mathbf{X}\|} > \rho \quad \begin{cases} \text{yes} \rightarrow \text{go to Step 7} \\ \text{no} \rightarrow \text{go to Step 6} \end{cases}$$

# ART1 Algorithm Description

## Step 6.

Disable best match

Set output of best match node to 0. Go to Step 3

$$\mathbf{b}_{j^*i}(\tau + 1) = \mathbf{b}_{j^*i}(\tau)\mathbf{x}_i$$

## Step 7.

Adapt best match

$$\mathbf{w}_{ij^*} = \mathbf{b}_{j^*i}(\tau) / (0.5 + \sum_i \mathbf{b}_{j^*i}(\tau)\mathbf{x}_i)$$

**Step 8.** Repeat. Enable and disabled nodes, then go to Step 2

# Model ART

---

The procedure can be stopped if:

1. The stored exemplar correctly match with the input signal  $\mathbf{X}$ , then the signal  $\mathbf{X}$  is included into this class and its exemplar (described by the vector  $\mathbf{W}$ ) is modified (if necessary)
2. There is no possibility to include input signal into existing class, then the 1<sup>st</sup> unused element creates a new class with  $\mathbf{W} = \mathbf{X}$ ;



# Model A R T

---

**3.** If there is no possibility to include input signal into existing class and there is no more unused elements the input signal is not taken into account (omitted).

# Model A R T

---

This algorithm solves both problems of **stability** and **plasticity**. It kept plasticity as long so all reserve nodes are not used, and stability – because the procedure of changes in the net structure (changes of weights  $\mathbf{W}^k$  values) can be performed only a limited number of times defined by the set of input signals.

It is the result of the algorithm where in the **Step 7**, we can only remove bits (i.e. put some weights to zero), but the bits can not be added. The stored exemplars can not be reconstructed in a cyclic way.

# Model A R T

The loop Step 3 → ... → Step 6 → Step 3 is searching through the set of stored exemplars:

- computing matching
- selecting the best matching, next second in line etc. according to the criterion

*maximum value of  $B^k * X$  until the condition*

$$\frac{\|BX\|}{\|X\|} > \rho$$

*is fulfilled*

# Model A R T – network description

---

The network is operating automatically. It does not need external signals and is able to cope even with the infinite stream of input data.

Basic features:

- fast access to classes (categories),
- possibility to create a new classes if necessary
- rejection of unknown inputs after exceeding the network capacity

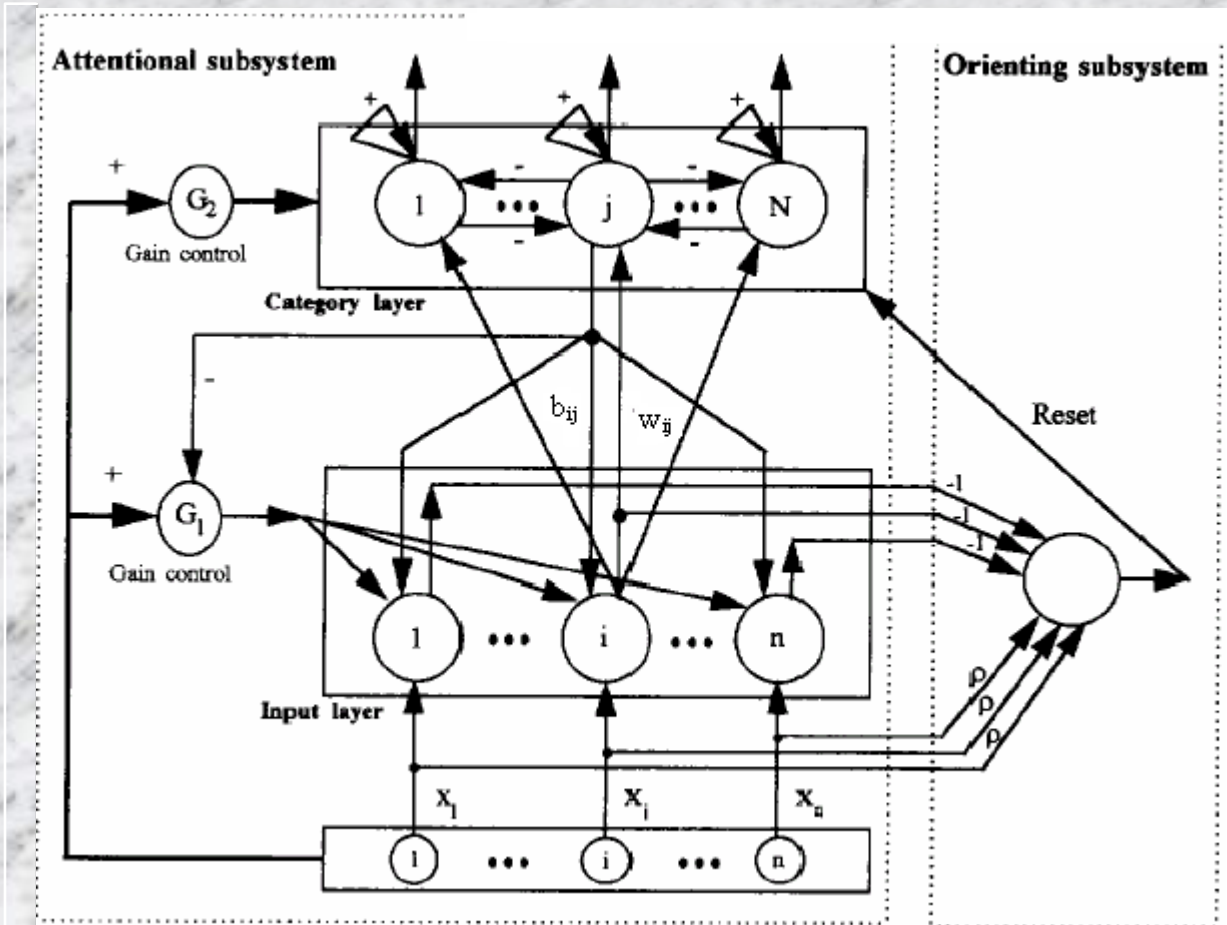


# Model A R T – network description

---

- can be used to memorize the binary patterns
- the self-organizing and self stabilizing network
- learn with the assumed accuracy
- can learn  $2^n$  different patterns, where  $n$  – pattern dimension

# Model A R T – network description



where:

$n$  – dimension of an input

$N$  – number of category

$w_{ji}$  – weights

bottom – up

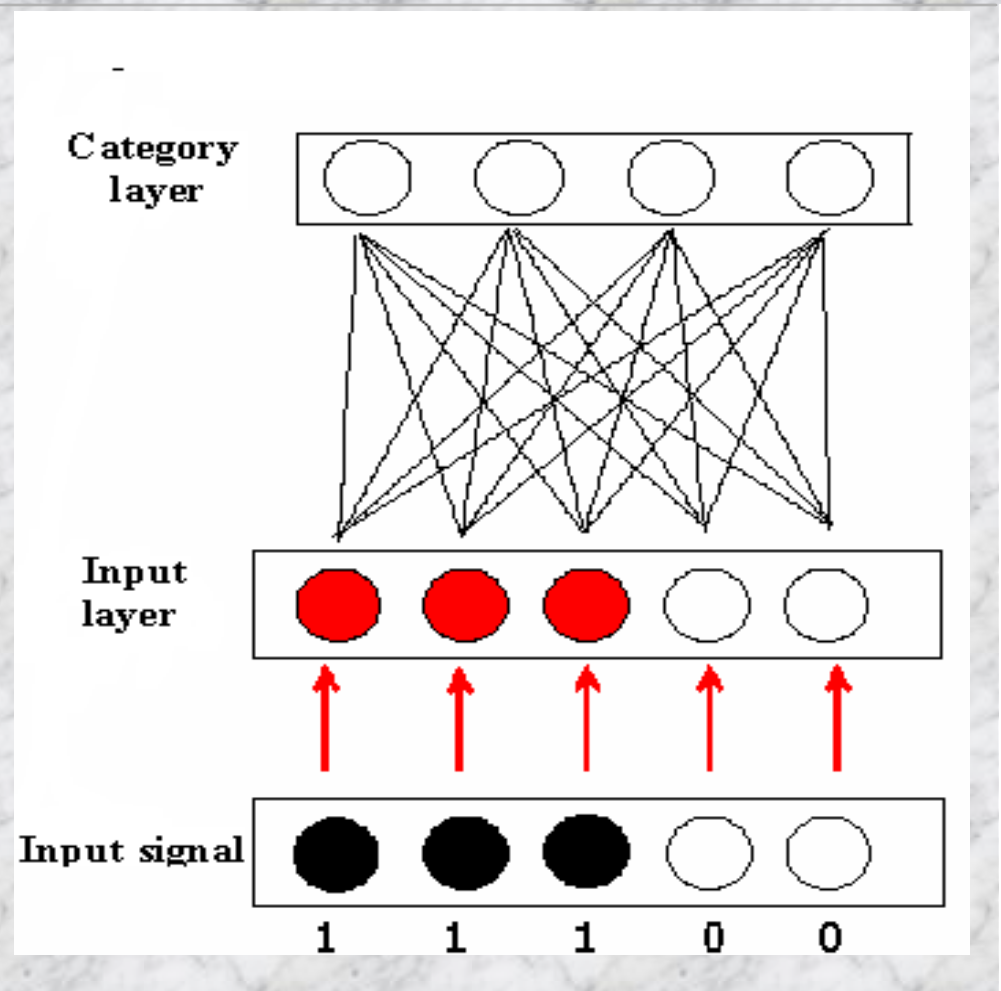
$b_{ji}$  – weights

top-down

$\rho$  – vigilance threshold

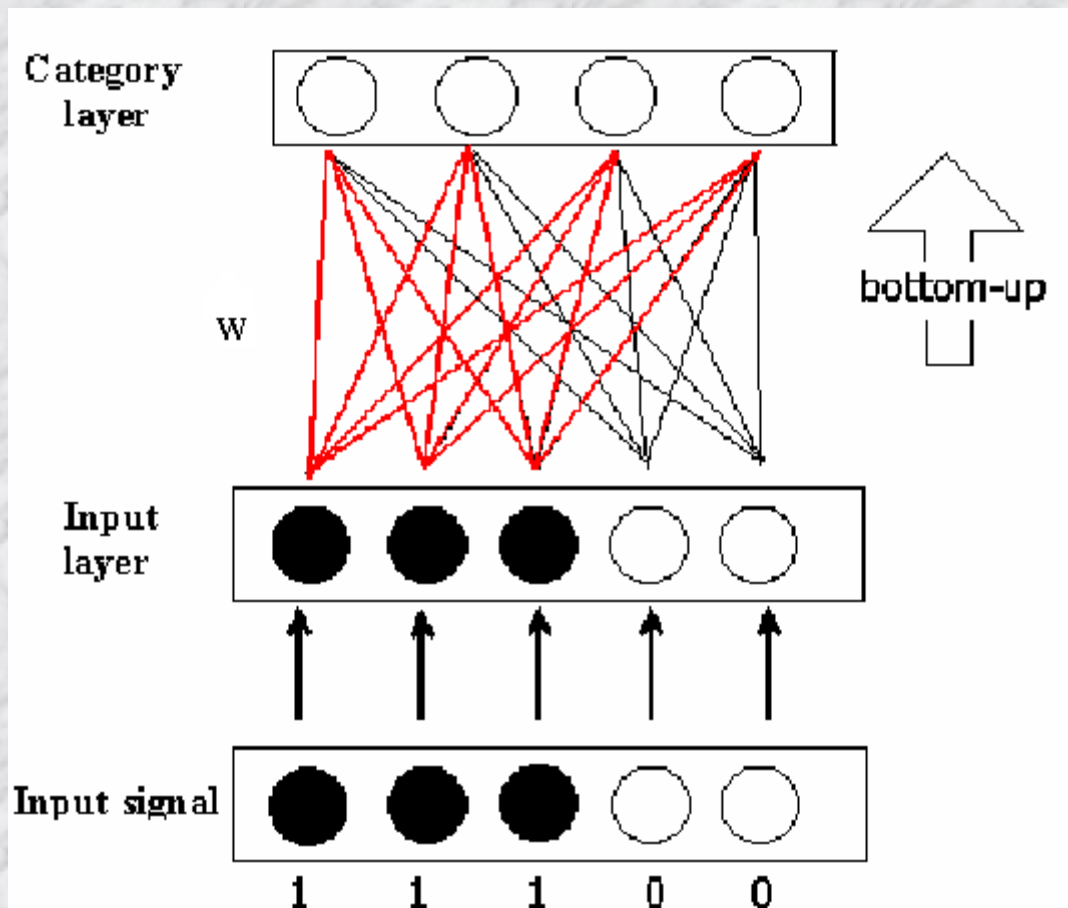
# Model A R T – network operation

- Input signal  
 $X=[1,1,1,0,0]$



# Model A R T – network operation

- Signal goes through weight matrix  $W$

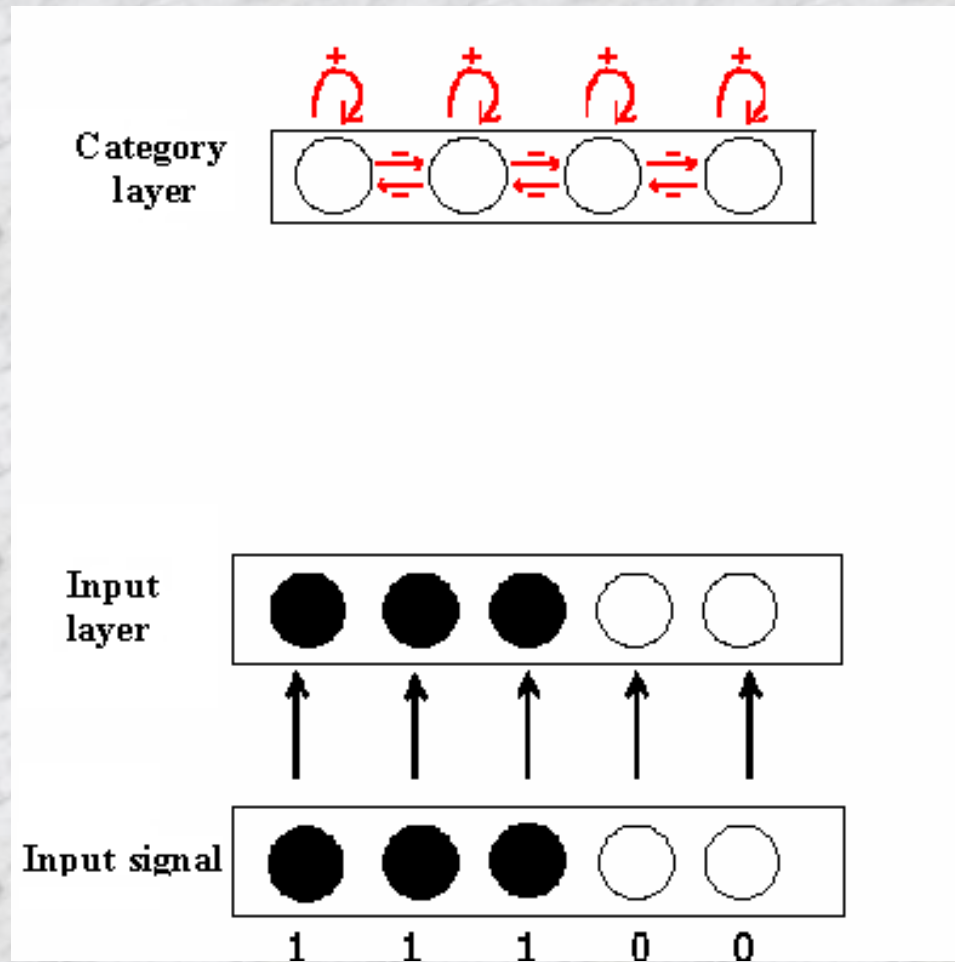




# Model A R T – network operation

- Selection of best matching category

$$J = \arg \max_j u_j \quad j=1 \dots N$$



# Model A R T – network operation

- Comparison of input signal and category template, if

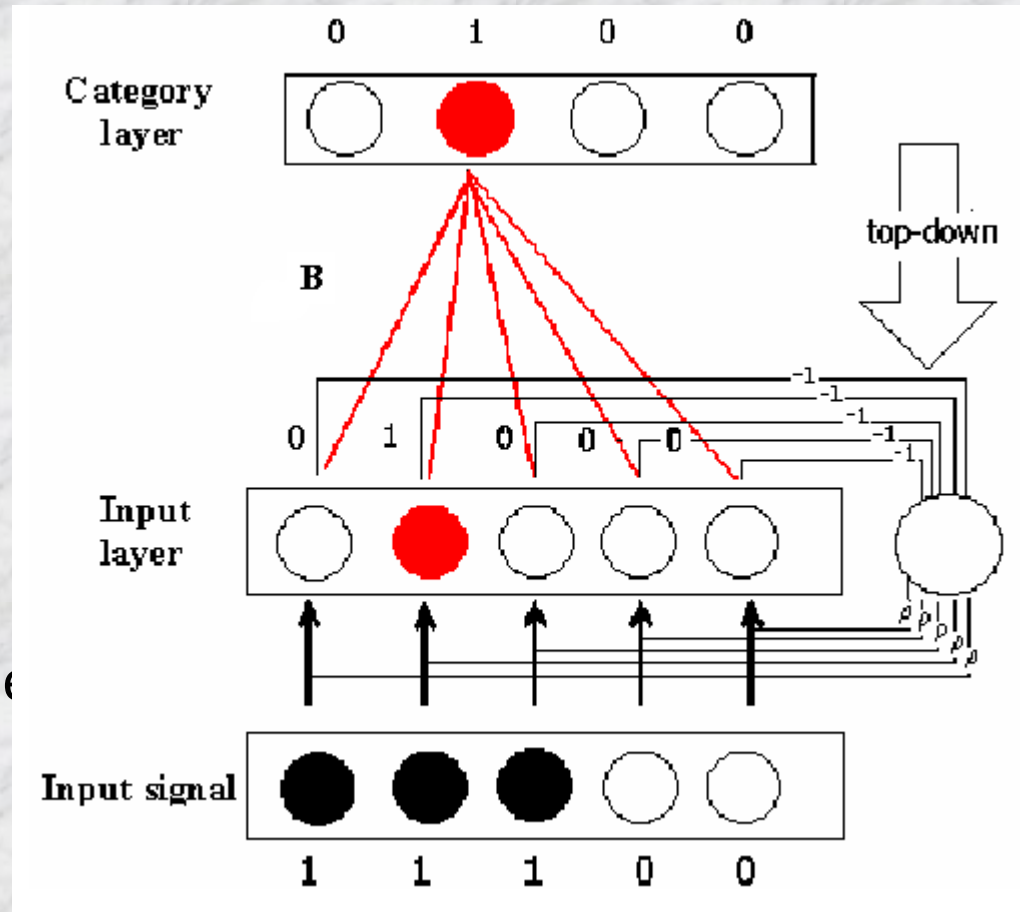
$$\frac{|B_J \cap X|}{|X|} = \frac{\sum_{i=1}^n b_{Ji} X_i}{\sum_{i=1}^n X_i} \leq \rho$$

reset category

otherwise category

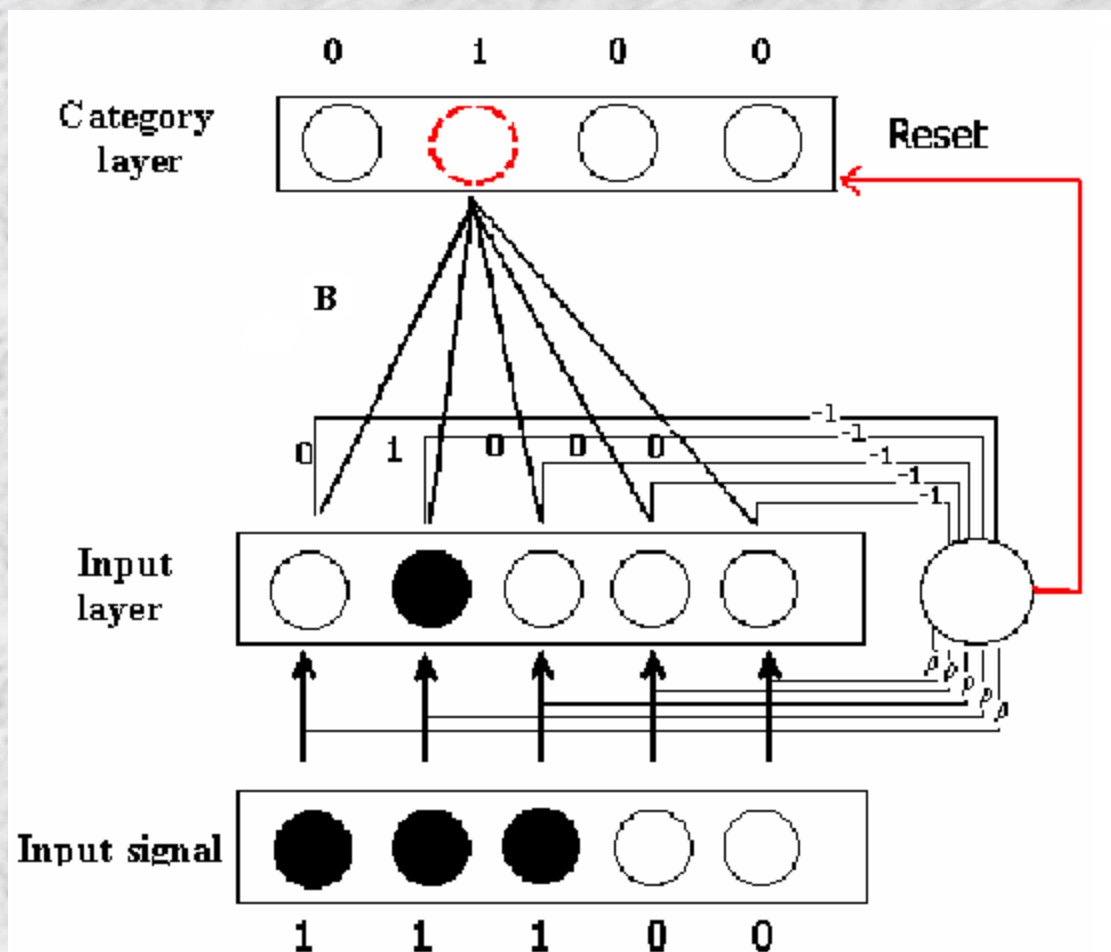
was correctly recognized

(resonance), weight correction



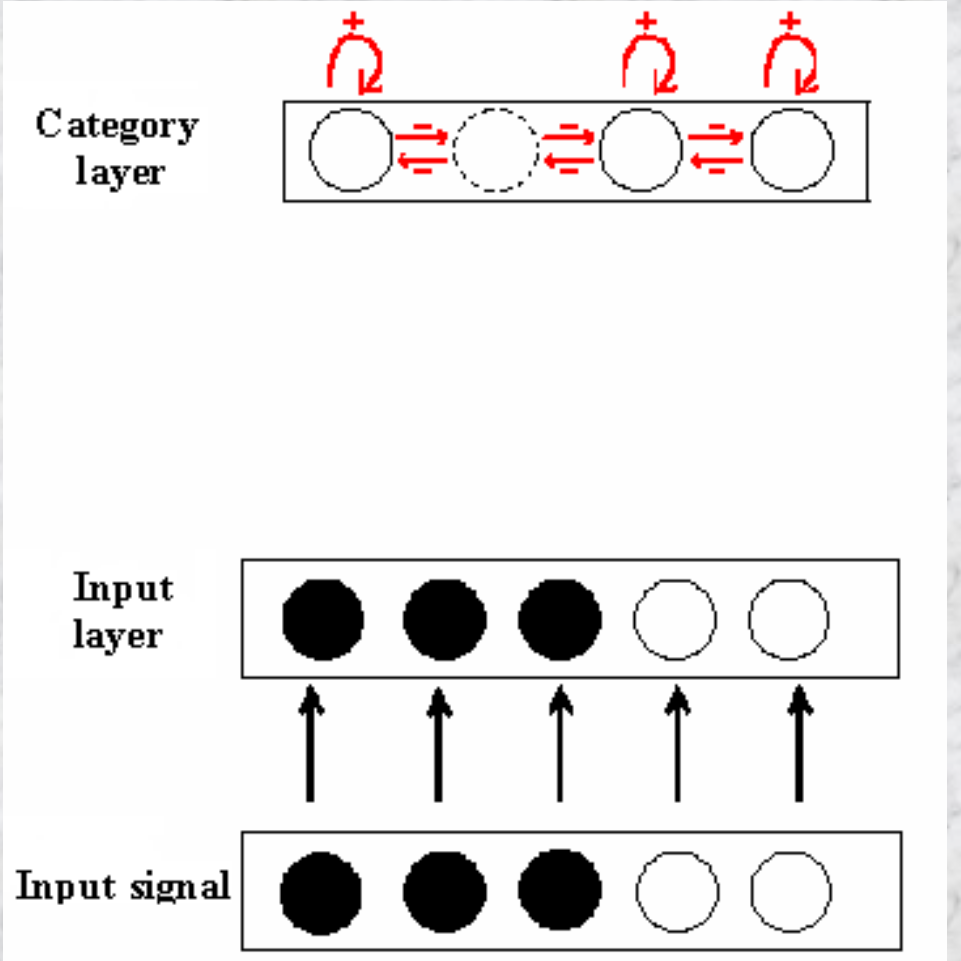
# Model A R T – network operation

Reset  
category



# Model A R T – network operation

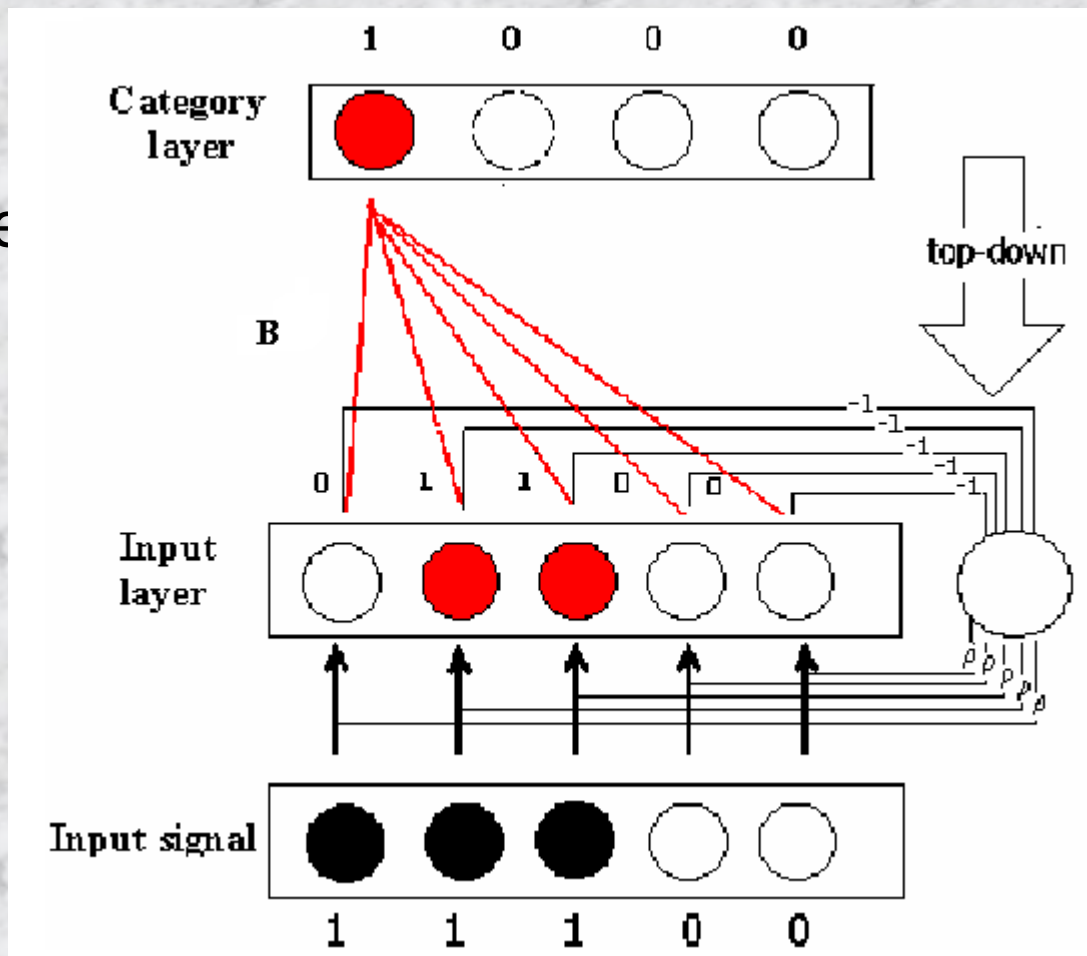
Selection of  
best category





# Model A R T – network operation

Comparison of  
input signal and  
category template



# Connection weights

Correction of the weight values, matrices **B** and **W**

$$b_{ji}^{(new)} = \begin{cases} \frac{w_{ji}^{(old)} x_i}{\alpha + \sum_{i=1}^n w_{ji}^{(old)} x_i} & j = J \\ b_{ji}^{(old)} & j \neq J \end{cases}$$

$$w_{ji}^{(new)} = \begin{cases} w_{ji}^{(old)} x_i & j = J \\ w_{ji}^{(old)} & j \neq J \end{cases}$$

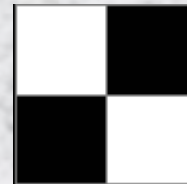
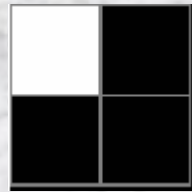
# Learning precision

---

- Learning accuracy depends of the parameter  $\rho$ ;  $0 \leq \rho \leq 1$
- Quality of categories depends if  $\rho$ 
  - wide categories for small  $\rho$  values
  - precise (narrow) categories for  $\rho$  close to 1

# ART1 modifications

Input signals  $x_1 = [0,1,1,1]$ ,  $x_2 = [0,1,1,0]$



if shown in this sequence will be classified into one category, because at the beginning  $B_0 = [0,1,1,1]$ , and next

$$\frac{|B_0 \cap X_2|}{|X_2|} = \frac{\sum_{i=1}^n b_{0i} x_{2i}}{\sum_{i=1}^n x_{2i}} = \frac{2}{2} = 1$$

we get the exemplar  $B_0 = [0,1,1,0]$ ,



# ART1 modifications

To avoid such problems the inputs can be extended adding the complementary vectors

$$x_1 = [0,1,1,1, | 1,0,0,0 ], x_2 = [0,1,1,0, | 1,0,0,1],$$

then

$$B_0 = [0,1,1,1,1,0,0,0]$$

$$|B_0 \cap X_2| = 3$$

and next

hence

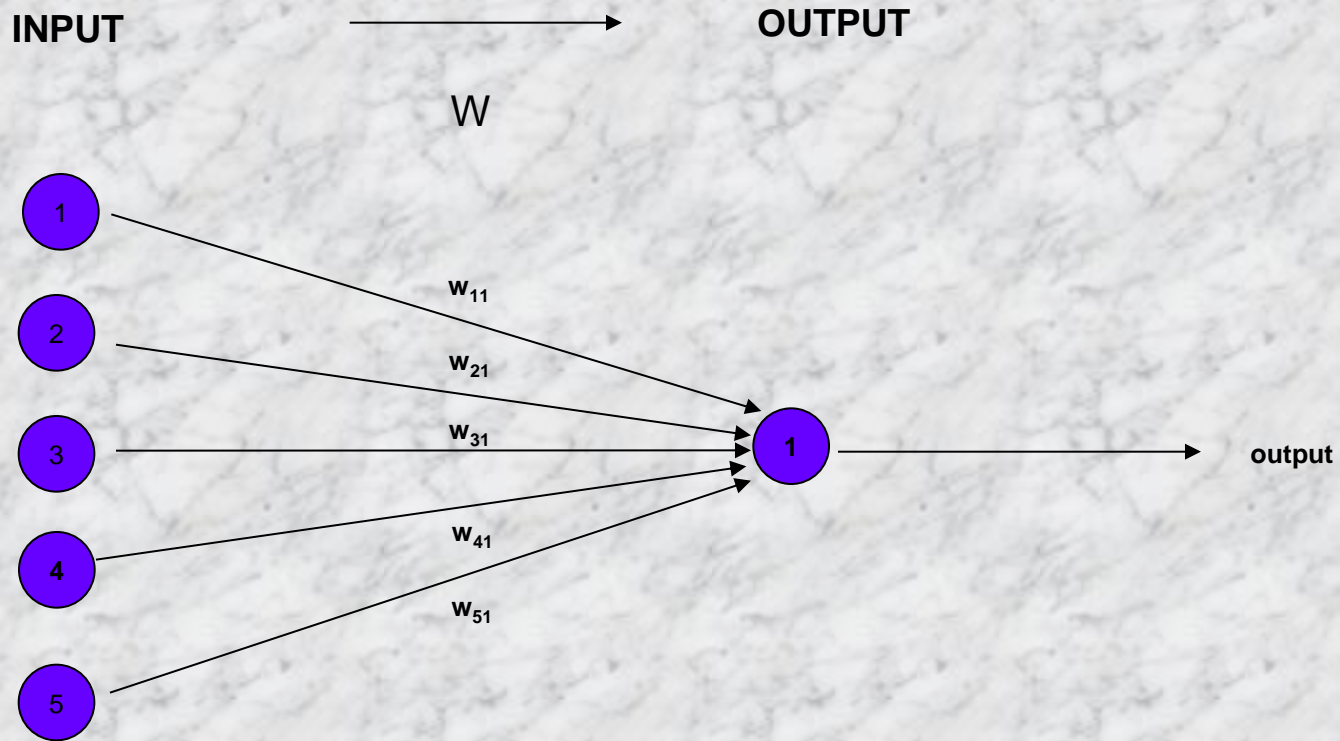
$$\frac{|B_0 \cap X_2|}{|X_2|} = \frac{\sum_{i=1}^n b_{0i} x_{2i}}{\sum_{i=1}^n x_{2i}} = \frac{3}{4}$$



**We'll take a  
5-minute  
break now**

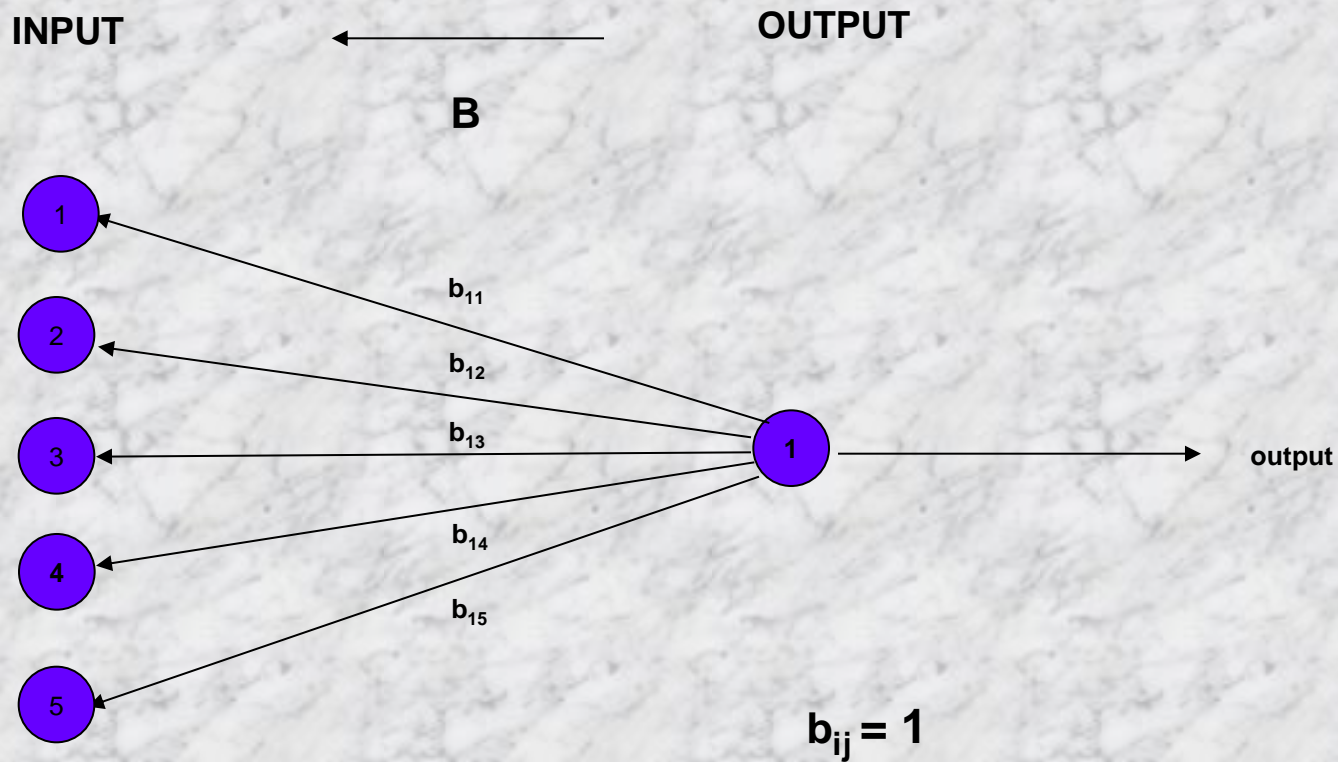
# ART Example

## Initial structure



# ART Example

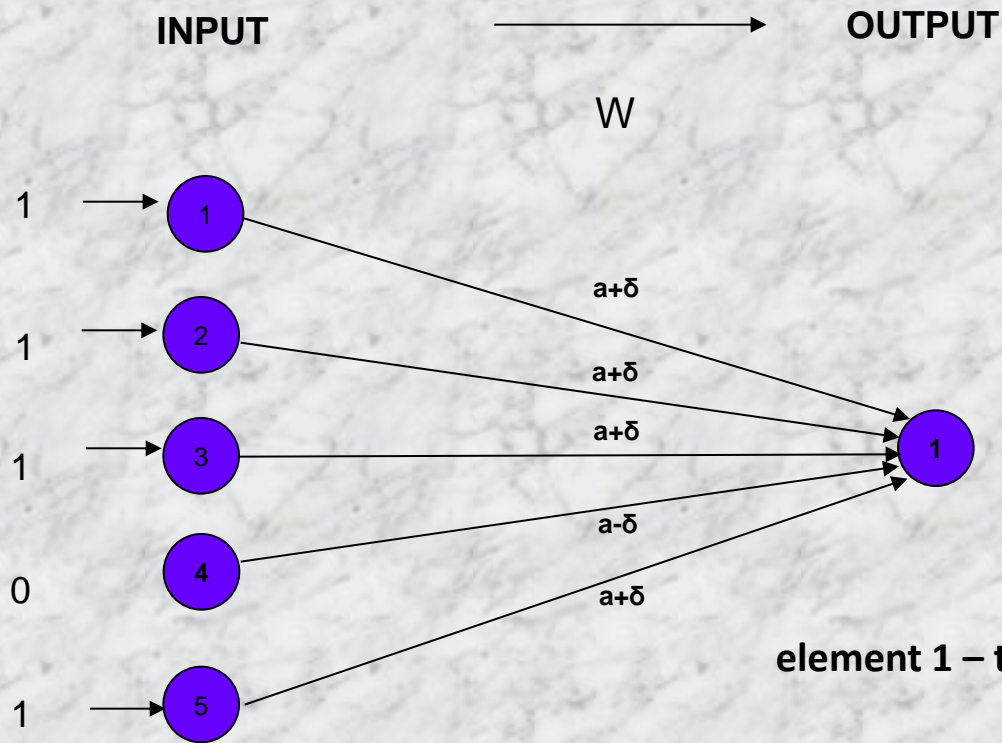
## Initial structure





# ART Example

1<sup>st</sup> input signal  $X_1 = [1 \ 1 \ 1 \ 0 \ 1]$



$$W^1 = \begin{bmatrix} a + \delta \\ a + \delta \\ a + \delta \\ a - \delta \\ a + \delta \end{bmatrix}$$

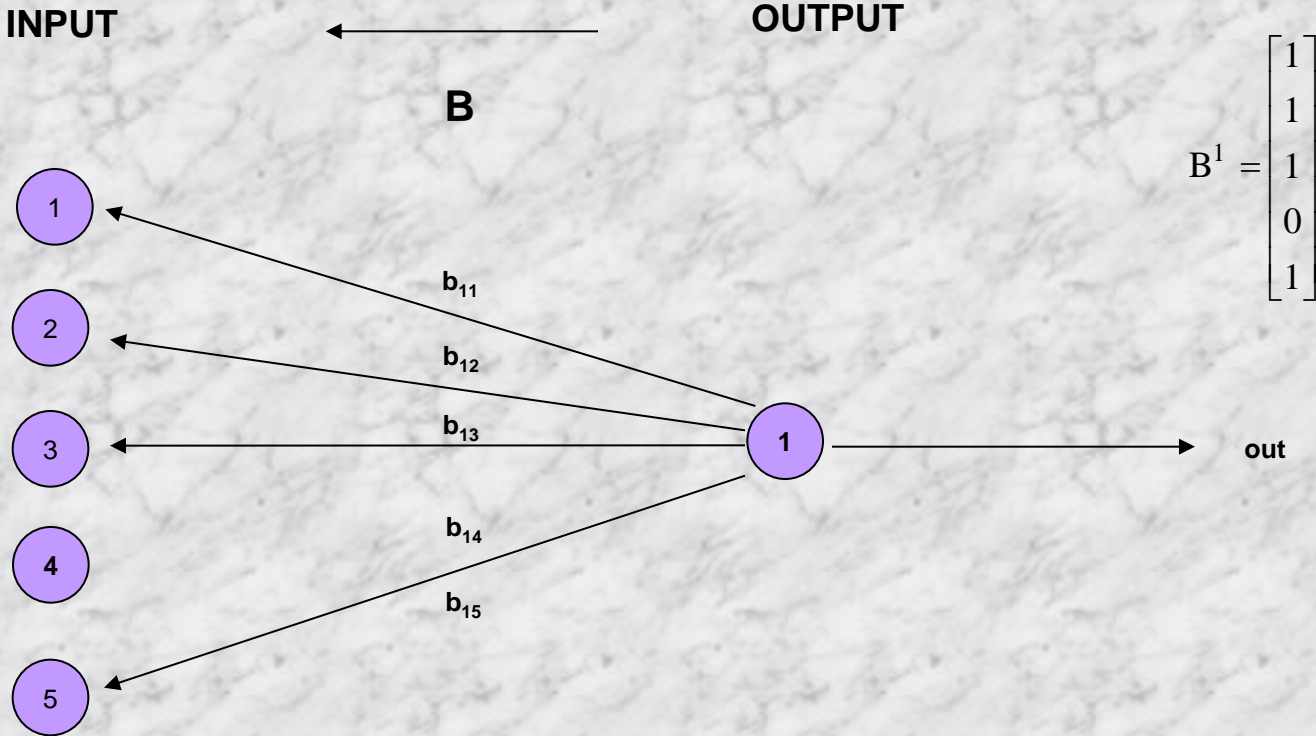
element 1 - the winner

# ART Example

1<sup>st</sup> input signal

INPUT

OUTPUT



# ART Example

2<sup>nd</sup> input signal  $X_2 = [0 \ 0 \ 1 \ 1 \ 1]$

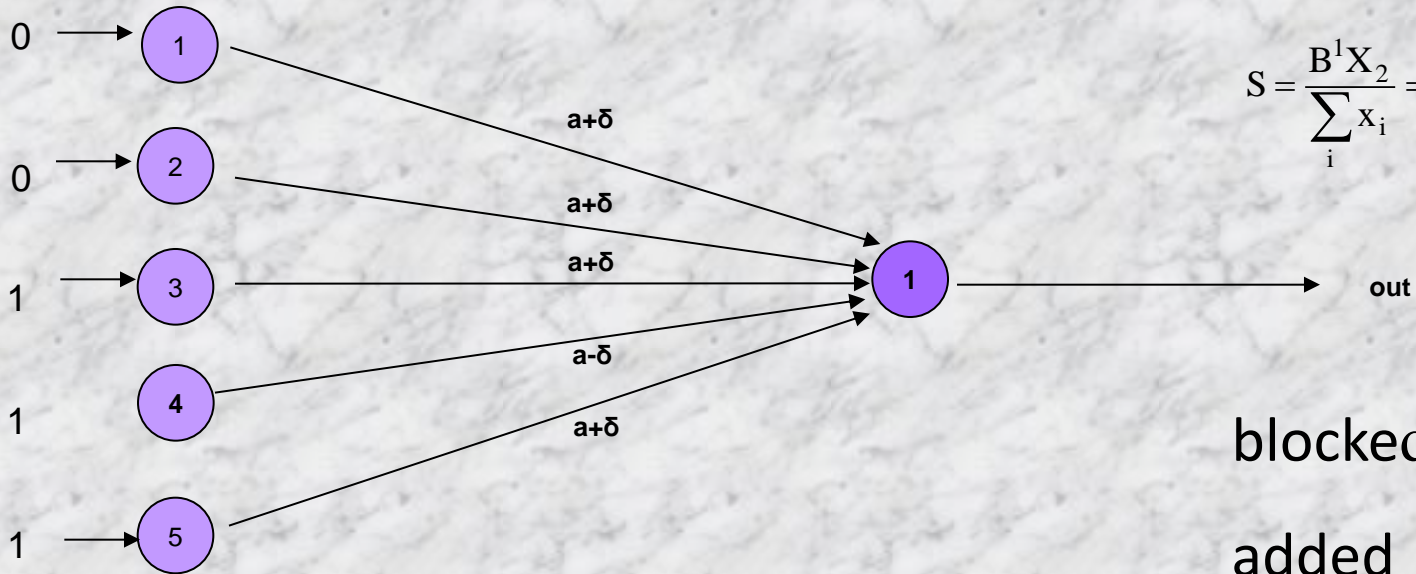
$$W^1 X_2 = 3a + \delta$$

INPUT

OUTPUT

$W^1$

vigilance test



$$S = \frac{B^1 X_2}{\sum_i x_i} = \frac{2}{3} < \rho = 0.9$$

blocked

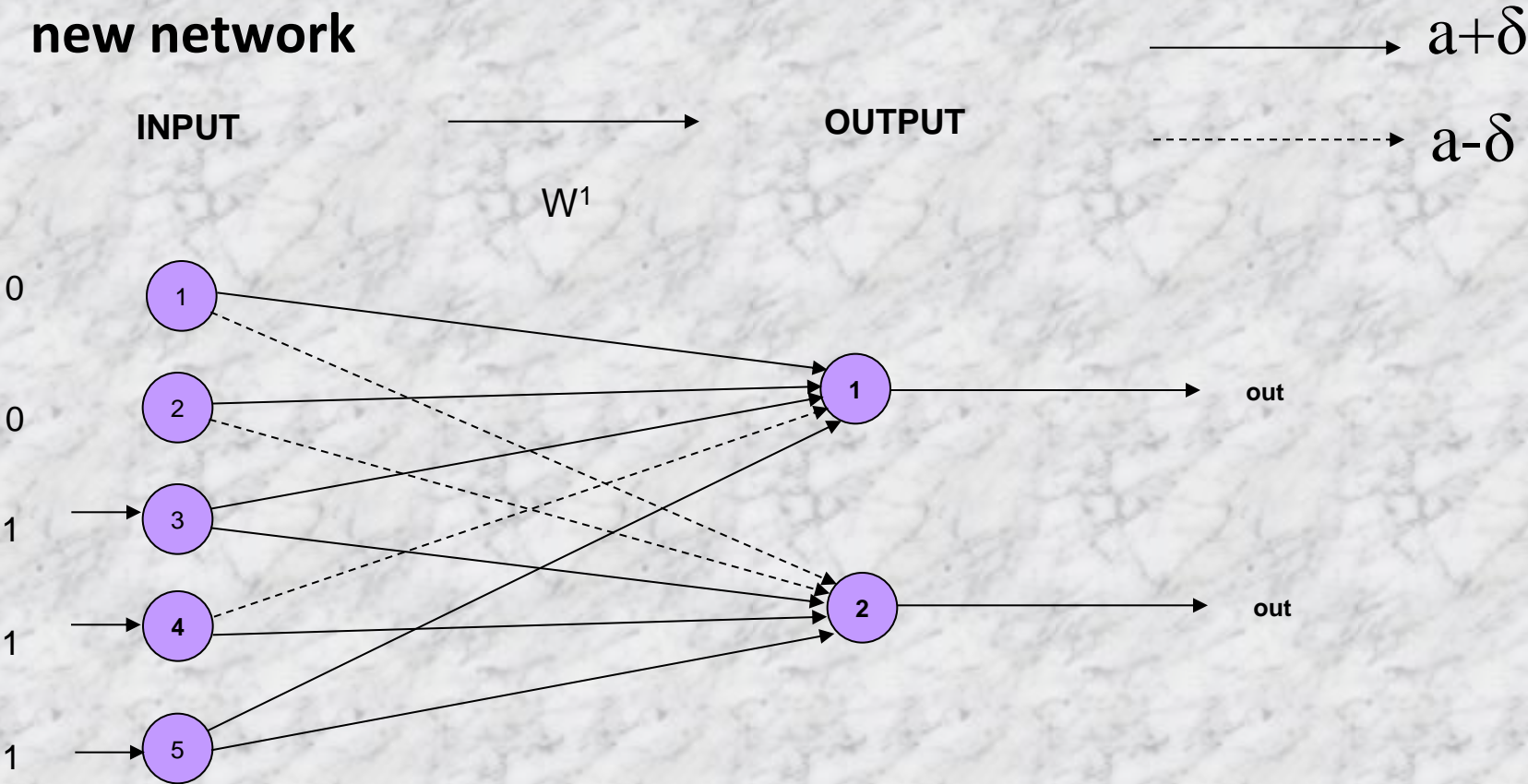


added



# ART Example

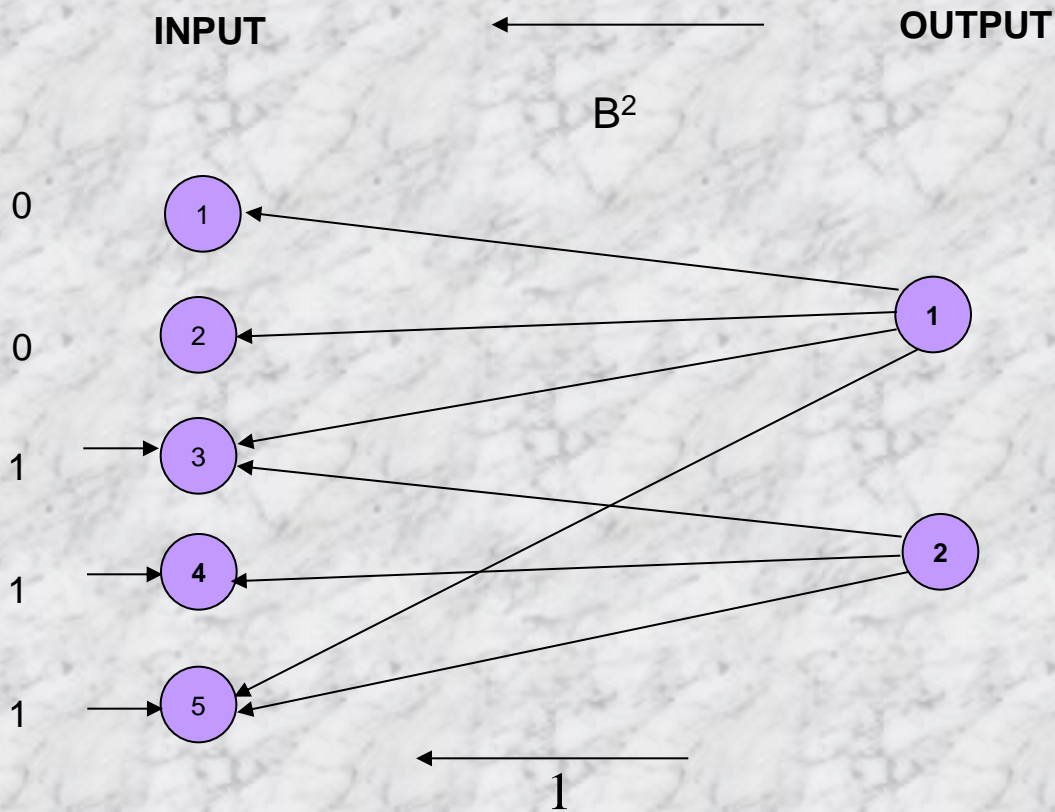
new network





# ART Example

new network



$$W^2 = \begin{bmatrix} a + \delta & a - \delta \\ a + \delta & a - \delta \\ a + \delta & a + \delta \\ a - \delta & a + \delta \\ a + \delta & a - \delta \end{bmatrix}$$

$$B^2 = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$$

# ART Example

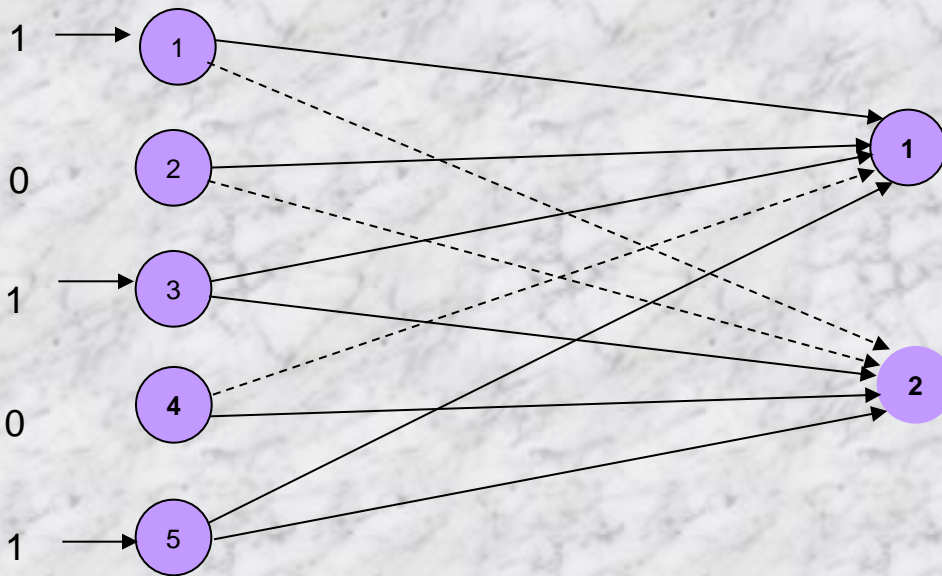
3<sup>rd</sup> input signal  $X_3 = [1 \ 0 \ 1 \ 0 \ 1]$

—————→  $a + \delta$   
 - - - - -→  $a - \delta$

INPUT

OUTPUT

$W^2$



$$W^2 X_3 = \begin{bmatrix} 3a + 3\delta \\ 3a + \delta \end{bmatrix}$$

**element 1 –  
the winner**

vigilance test

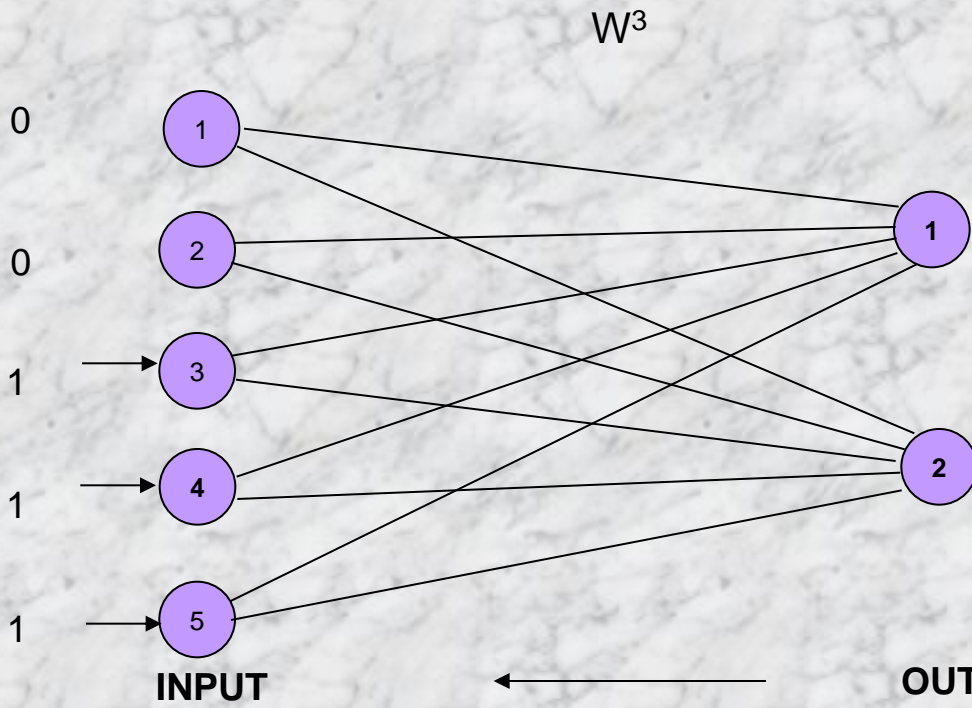
$$S = \frac{B^1 X_3}{\sum_i x_i} = \frac{3}{3} > \rho = 0.9$$

input  $X_3$  is added to the class  $X_1$

# ART Example

new network

INPUT  $\longrightarrow$  OUTPUT



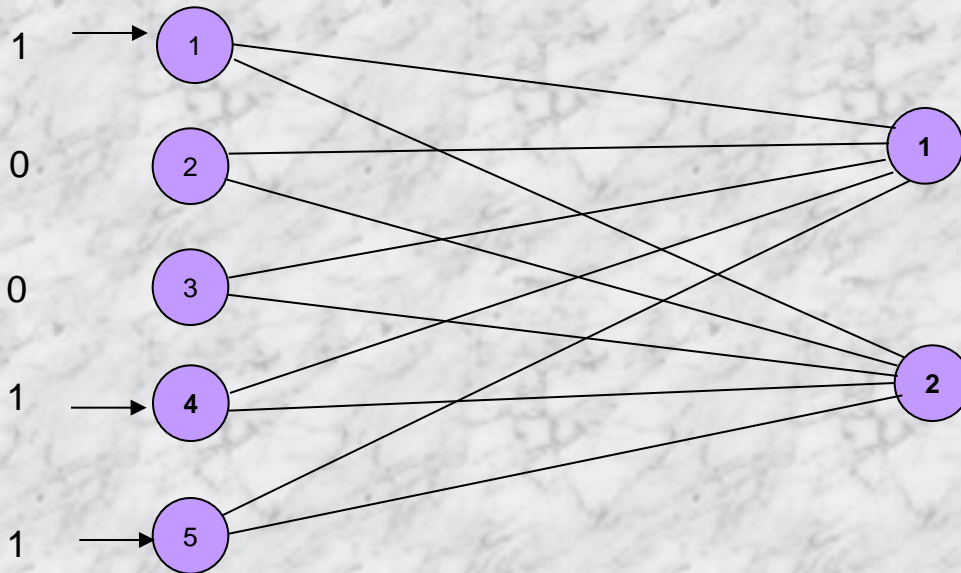
$$W^3 = \begin{bmatrix} a + 2\delta & a - \delta \\ a & a - \delta \\ a + 2\delta & a + \delta \\ a - 2\delta & a + \delta \\ a + 2\delta & a - \delta \end{bmatrix}$$

$$B^3 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 1 & 1 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$$

$B^3$

# ART Example

4th input signal  $X_4 = [1 \ 0 \ 0 \ 1 \ 1]$



$$W^3 X_4 = \begin{bmatrix} 3a + 2\delta \\ 3a + \delta \end{bmatrix} \quad \text{element 1 - the winner}$$

vigilance test

$$S = \frac{B_1^3 X_4}{\sum_i x_i} = \frac{2}{3} < \rho = 0.9 \quad \text{element 1 blocked}$$

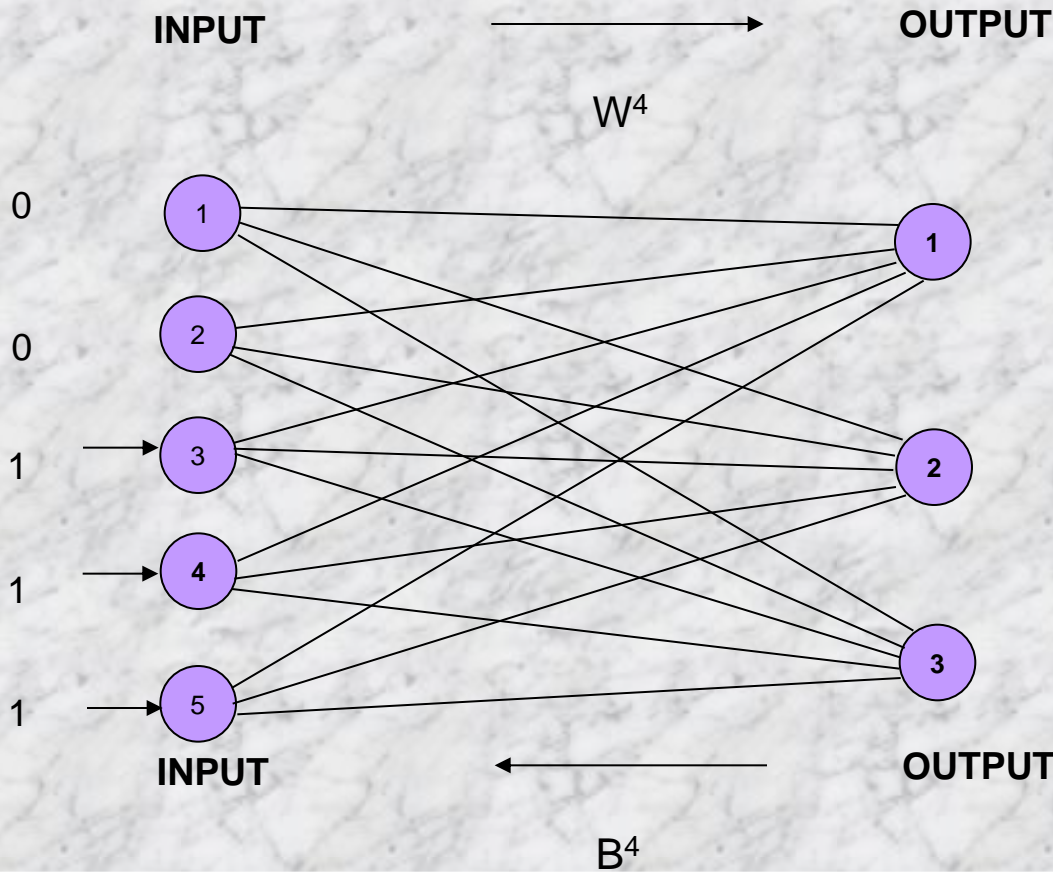
$$S = \frac{B_2^3 X_4}{\sum_i x_i} = \frac{2}{3} < \rho = 0.9 \quad \text{element 2 blocked}$$

new element 3 is added



# ART Example

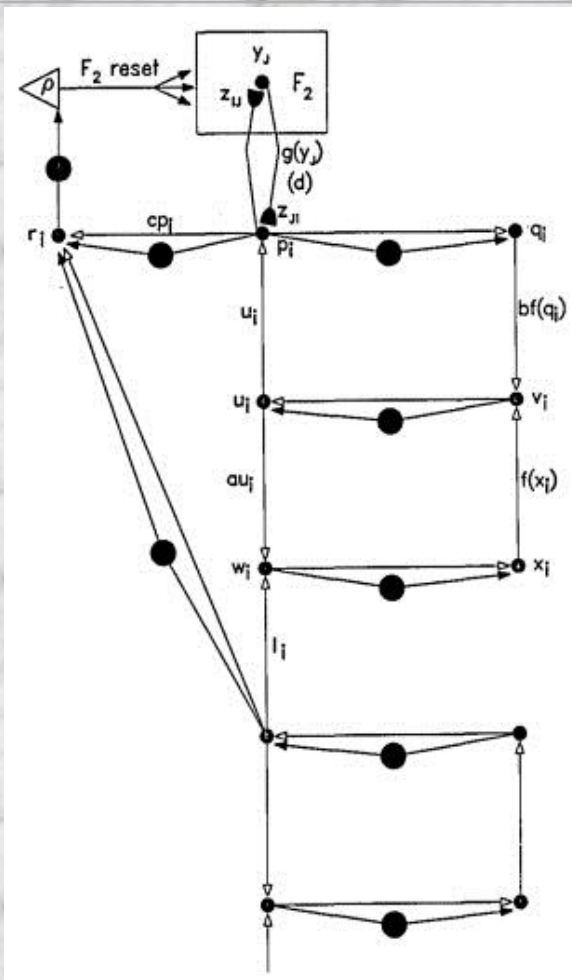
new network



$$W^4 = \begin{bmatrix} a+2\delta & a-\delta & a+\delta \\ a & a-\delta & a-\delta \\ a+2\delta & a+\delta & a-\delta \\ a-2\delta & a+\delta & a+\delta \\ a+2\delta & a+\delta & a+\delta \end{bmatrix}$$

$$B^4 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

# ART2



F2

F2 (category field) is similar to the recognition layer in ART1, which is responsible for competitive matching to current input modes representation.

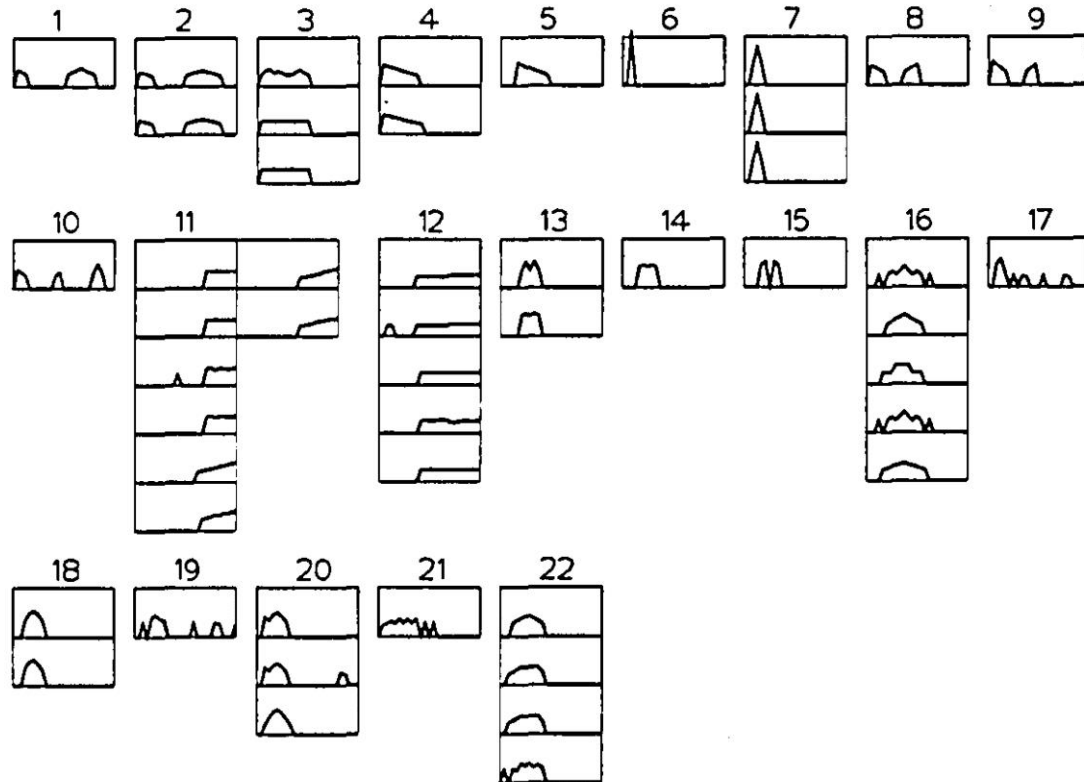
F1

F1 (feature representation field) is similar to the comparison layer in ART1, which includes some calculating tiers and gains controller.

F0

F0 (preprocessing layer) perform normalization, contrast enhancement, noise reduction

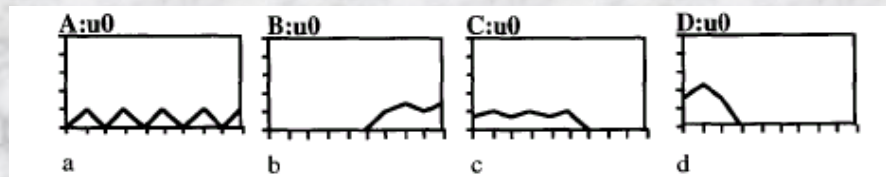
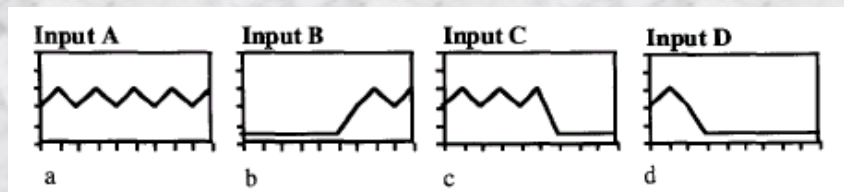
# ART2



Recognition category summary for the ART 2 system.

# ART2

- Can classify the analog pattern (gray scale)
- Preprocessing in the layer F0: normalization, contrast enhancement, noise reduction
- Comparison with input in the layer F1
- Patterns are similar if are proportional

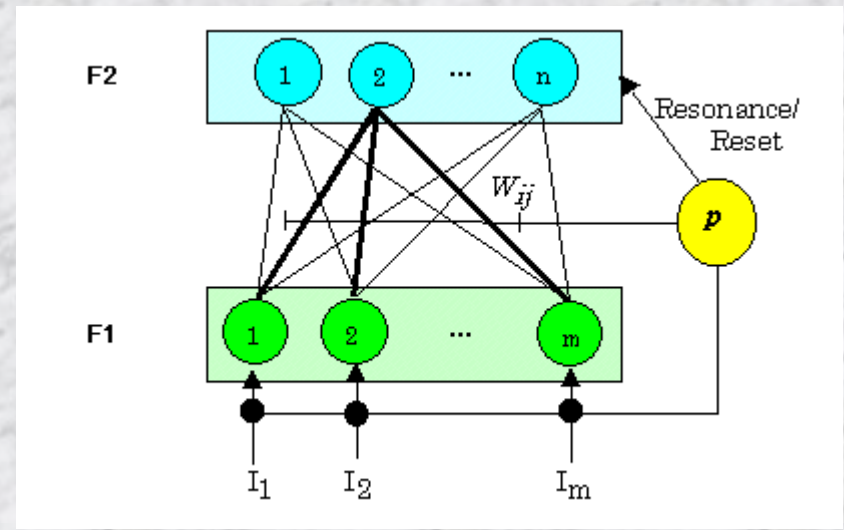




# Fuzzy ART

A Fuzzy Adaptive Resonance Theory (ART) model capable of rapid stable learning of recognition categories in response to arbitrary sequences of analog or binary input patterns is described. Fuzzy ART incorporates computations from fuzzy set theory into the ART 1 neural network.

The generalization to learning both analog and binary input patterns is achieved by replacing appearances of the **AND** operator in ART1 by the **MIN** operator of fuzzy set theory.





# ARTMAP

- Calculation of input in the category (output) layer

$$u_i = \frac{\sum_{j=1}^N \min(w_{ji}, x_i)}{\alpha + \sum_{j=1}^N w_{ji}}$$

- Vigilance test

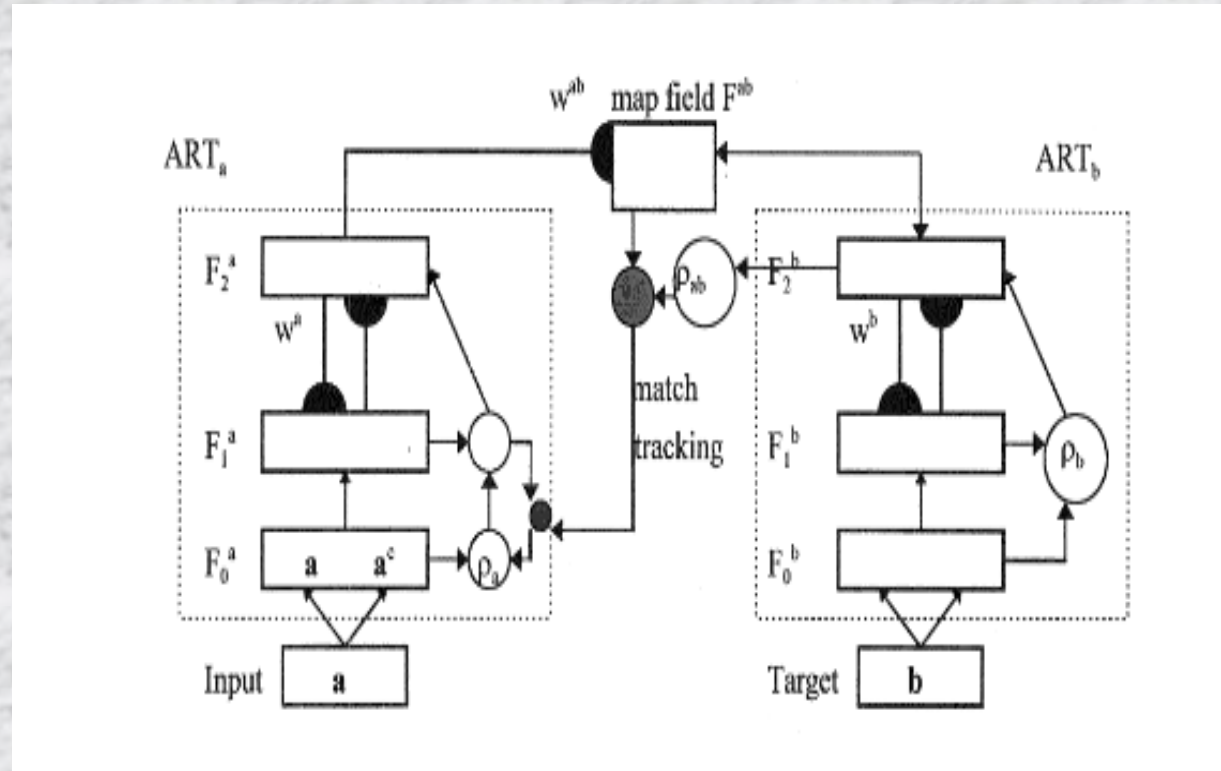
$$\frac{\sum_{i=1}^N \min(w_{Ji}, x_i)}{\sum_{i=1}^N w_{Ji}} < \rho$$

- Change weight values

$$w_{Ji}^{(\text{new})} = \begin{cases} w_{Ji}^{(\text{old})} + \bar{\beta}(\min(w_{Ji}^{(\text{old})}, x_i) - w_{Ji}^{(\text{old})}) & \text{for existing category} \\ x_i & \text{for new category} \end{cases}$$

$0 \leq \beta \leq 1$  learning factor ( $\beta=1$  fast learning)

# Fuzzy ARTMAP



In a Fuzzy ARTMAP, categories formed by two Fuzzy ART units  $ART_a$  and  $ART_b$ , are associated via a MAP field as category and class respectively.



# Fuzzy ART and Fuzzy ART1

- Calculation of input in the category (output) layer

$$u_i = \frac{\sum_{j=1}^N \min(w_{ji}, x_i)}{\alpha + \sum_{j=1}^N w_{ji}}$$

- Vigilance test

$$\frac{\sum_{i=1}^N \min(w_{Ji}, x_i)}{\sum_{i=1}^N w_{Ji}} < \rho$$

- Change weight values

$$w_{Ji}^{(\text{new})} = \begin{cases} w_{Ji}^{(\text{old})} + \bar{\beta}(\min(w_{Ji}^{(\text{old})}, x_i) - w_{Ji}^{(\text{old})}) & \text{for existing category} \\ x_i & \text{for new category} \end{cases}$$

$0 \leq \beta \leq 1$  learning factor ( $\beta=1$  fast learning)

# Applications

---

- Recognition within the pictures
- Recognition objects from the radar
- Speech recognition
- .....