Advanced Studies in Theoretical Physics
Vol. 9, 2015, no. 13, 629 - 658
HIKARI Ltd, www.m-hikari.com
http://dx.doi.org/10.12988/astp.2015.5775

Invariant in Neural Morpho Computing

Germano Resconi[1], Xu Guanglin[2] and Xu Xiaolin[3]

[1] Dept. of Mathematics and Physics, Catholic University Brescia, Italy

[2] School of Mathematics and Information
Shanghai Lixin University of Commerce, China

[3] College of International Vocational Education,
Shanghai Second Polytechnic University, Shanghai, China

Abstract

In this paper we present an unification theory denoted morpho computing that controls data with rules. This new theory gives us an unification model of the brain theories. Brain appears as a special system where we can control and produce rules for our necessity. Physical nature is a system similar to the brain but with fixed rules. In physical universe samples are the experiments that give us an idea how the universal rules are without symbolic methods. Unsupervised neural network and supervised neural network are parts of the morpho computing where invariance or rules are parts of the computation. So we not only have memory separated from rules as in the digital computer, but rules and data are mixed in associative memory or in more sophisticate morpho computing memory. We show that supervised neuron is implemented in the new theory without most of the problems in the back propagation.

Keywords: Unsupervised neuron, supervised neuron, associative memory, Hopfield neural network, Kohonen self-organizing maps, pattern recognition

1. Introduction

This paper begins with associative memory as a prototype element of morpho computing. Associative memory is a good example of the rules and data fusion. In the classical associative memory we use orthonormal property of the input samples. Now we show that this property is not always necessary. So we move from associative memory to propagators with certain properties. After the associative

memory we study self associative memory as Hopfield neural network. The unsupervised Kohonen self-organizing maps is included in the morpho computing as special cases. We also study the involution neural network with the morpho computing instrument. The last part and also the most important part is the supervised neuron where we go beyond the back propagation method in a way to solve most of its problems. In conclusion the fusion of rules and data is the winning method to understand how the brain works in comparison with physical nature. We show that the morpho computing can be used also for physical nature (electronical circuit). This gives us the idea that physical nature and the brain are not so different but both are parts of the same family of systems. Physical rules are fixed and give us the support to create the brain which can join aims (intentions) and rules. Aims or intentions can be represented by special samples, boolean functions, images, input vectors or other systems that we want to implement in the physical model of the brain which is a special part of the universe where rules can change.

2 Neural Associative Memory

Neural associative memories (NAM) are neural network models consisting of neuron-like and synapse-like elements. At any given point the state of the neural network is given by the activity pattern produced by input and output vectors. Neurons update their activity values based on the inputs they receive (over the synapses).Fig.1 shows the neural network with three input vectors and one output vector on the right.

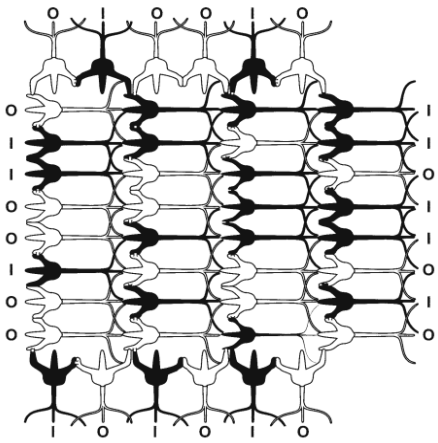


Fig.1 the neural network with neuron-like and synapse-like

The linear associator is one of the simplest and first studied associative memory model. Fig.2 is the network architecture of the linear associator.

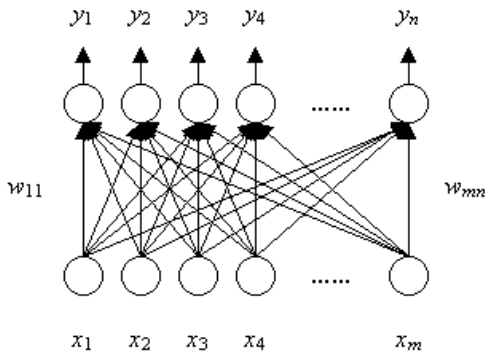


Fig.2 the network of the linear associator

In Fig.2, all the *m* input units are connected to all the *n* output units via the connection weight matrix $\boldsymbol{W} = [w_{ij}]_{m \times n}$ where w_{ij} denotes the synaptic strength of the unidirectional connection from the *i*th input unit to the *j*th output unit.

Given the representation $\overrightarrow{x_k} = [x_{k1}, x_{k2}, ...x_{km}]^T$, $\overrightarrow{y_k} = [y_{k1}, y_{k2}, ...y_{kn}]^T$,for the jth component $y_{kj} \quad j = 1,2,..n$,we have (1).

$$y_{kj} = [w_{j1}(k), w_{j2}(k), ..., w_{jm}(k)] \begin{bmatrix} x_{k1} \\ x_{k2} \\ ... \\ x_{km} \end{bmatrix} \tag{1}$$

(1) can be written in the form of (2).

$$y_{kj} = \sum_{i=1}^{m} w_{ji}(k) x_{ki} \qquad (2)$$

Or (3)

$$\begin{bmatrix} y_{k1} \\ y_{k2} \\ ... \\ y_{kn} \end{bmatrix} = \begin{bmatrix} w_{11}(k) & w_{12}(k) & ... & w_{1m}(k) \\ w_{21}(k) & w_{22}(k) & ... & w_{2m}(k) \\ ... & ... & ... & ... \\ w_{n1}(k) & w_{n2}(k) & ... & w_{nm}(k) \end{bmatrix} \begin{bmatrix} x_{k1} \\ x_{k2} \\ ... \\ x_{km} \end{bmatrix} \tag{3}$$

It is the connection weight matrix W that stores the q different associated pattern pairs $\{(\boldsymbol{x}_k, \boldsymbol{y}_k) \mid k = 1, 2, ..., q\}$. For every associative pattern k, $\overrightarrow{x_k} \to \overrightarrow{y_k}$,We have (4).

$$\overrightarrow{y_k} = \overrightarrow{w(k)}\,\overrightarrow{x_k} \tag{4}$$

Building an associative memory is nothing but constructing the connection weight matrix $\boldsymbol{W}$ that when an input pattern is presented, the stored pattern associated with the input pattern is retrieved. So we call the set of components $\boldsymbol{W}$ correlation memory matrix $\overrightarrow{M}$, which can be represented as (5).

$$\overrightarrow{M} = \sum_{k=1}^{q} \overrightarrow{w(k)} \tag{5}$$

So we have $\vec{Y} = \overrightarrow{M}\,\overrightarrow{X}$.

The process of constructing the connection weight matrix is called encoding. During encoding the weight values of the correlation matrix $\overrightarrow{M}$ for a particular associated pattern pair $(\boldsymbol{x}_k, \boldsymbol{y}_k)$ are computed as (6).

$$W = \sum \overrightarrow{y_k}\,\overrightarrow{x_k^T} \tag{6}$$

Or it can be also written like (7).

$$\overrightarrow{M} = [\overrightarrow{y_1}, \overrightarrow{y_2}, ..., \overrightarrow{y_q}] \begin{bmatrix} \overrightarrow{x_1^T} \\ \overrightarrow{x_2^T} \\ \cdots \\ \overrightarrow{x_q^T} \end{bmatrix} = \vec{Y}\,\overrightarrow{X^T} \tag{7}$$

So $\vec{Y} = \overrightarrow{M}\,\overrightarrow{X} = \vec{Y}\,\overrightarrow{X^T}\,\overrightarrow{X}$

And $\overrightarrow{X^T}\,\overrightarrow{X}$ =Identity.

For example, if $X = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ \cdots & \cdots & \cdots \\ x_{n1} & x_{n2} & x_{n3} \end{bmatrix}$,and $\varphi_1 = \begin{bmatrix} x_{11} \\ x_{21} \\ \cdots \\ x_{n1} \end{bmatrix}, \varphi_2 = \begin{bmatrix} x_{12} \\ x_{22} \\ \cdots \\ x_{n2} \end{bmatrix}, \varphi_3 = \begin{bmatrix} x_{13} \\ x_{23} \\ \cdots \\ x_{n3} \end{bmatrix}$, X is

represented $X = [\varphi_1 \quad \varphi_2 \quad \varphi_3]$.Since $\overrightarrow{X^T}\overrightarrow{X}$ =Identity,

$$\begin{bmatrix} \varphi_1 \\ \varphi_2 \\ \varphi_3 \end{bmatrix} [\varphi_1 \quad \varphi_2 \quad \varphi_3] = Identity$$

So we have $\varphi_i \varphi_i = 1$ and $\varphi_i \varphi_j = 0 \quad i \neq j$,which means if the input patterns are mutually orthogonal, perfect retrieval can happen.

3 Associative Memory and Morpho Computing

The associative memory is a propagator that propagates the information form input X to output Y with the rule Y= WX. Now given the sample $Y_k = WX_k$,if $Y_k = A, X_k = B$, we have $A = WB$ .Since A can be written in form (8).

$$A = A(B^T B)^{-1} B^T B \qquad (8)$$

We have (9)

$$W = A(B^T B)^{-1} B^T \qquad (9)$$

When the input samples B are orthogonal we have (10).

$$W = AB^T \qquad (10)$$

This is the classical associative expression.

Example 1

$$A = \begin{bmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \\ y_{31} & y_{32} \end{bmatrix}, B = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{bmatrix}$$

So we have (11)

$$W = A(B^T B)^{-1} B^T = \begin{bmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \\ y_{31} & y_{32} \end{bmatrix} \left(\begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{bmatrix}^T \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{bmatrix} \right)^{-1} \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{bmatrix}^T = w_{i,j} \quad (11)$$

The samples in input B define the invariants or rules to be satisfied in the transformation and the samples in output give us the type of transformation that we want to generate.
In fact given the inputs

$$B = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{bmatrix}$$ we have the input invariance (12).

,

$$(x_{32} - x_{22}) - (x_{22} - x_{12}) = x_{32} - 2x_{22} + x_{12} = 0$$
$$(x_{31} - x_{21}) - (x_{21} - x_{11}) = x_{31} - 2x_{21} + x_{11} = 0 \qquad (12)$$

That means the components of the input vectors are in the straight line. Since the samples are points in the straight line, the output vectors are points in the straight line.

Now when the output samples are transformations of the inputs as $A = \Omega B$, We have (13).

$$W = \Omega B(B^T B)^{-1} B^T = \Omega B(B^T B)^{-1} B^T \quad (13)$$

The parameters W of the associative memory can be written in (14).

$$W = \Omega B(B^T B)^{-1} B^T = \Omega Q \quad (14)$$

Where Q is the projection operator of the vector p into the input space sample B.
In fact we have

$$Q = B(B^T B)^{-1} B^T$$
$$Q^2 = B(B^T B)^{-1} B^T B(B^T B)^{-1} B^T = Q \quad (15)$$

For A = B we have (16).

$$Wy = B(B^T B)^{-1} B^T y = B(B^T B)^{-1} B^T y \qquad (16)$$

For $y = \begin{bmatrix} 1 \\ 1 \\ 4 \end{bmatrix}$, We give the association of external vector and its projection as Fig.3.

Blue line represents external vector($X_{k)}$, and red line is the vector (Y_k) with the straight line property generated by projection operator.

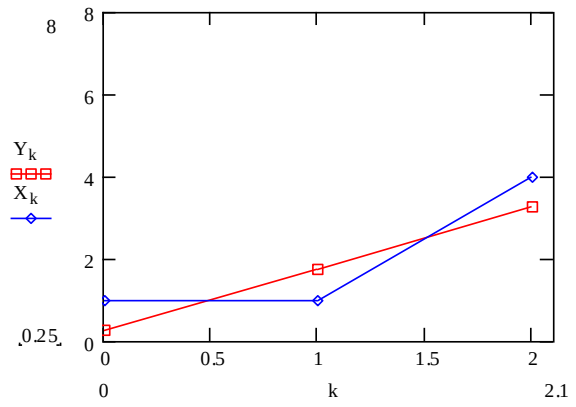


Fig.3 the association of external vector and its projection

Where the X_k and Y_k are associated. They are not equal because the external vector is not be in agreement with the rules in B that are straight lines.
For A=2B, we have Fig.4. We also have a red straight line but with the expansion of the coordinate space ($\Omega = 2$).

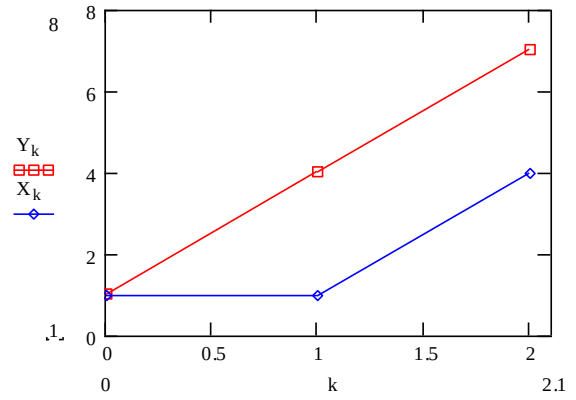


Fig.4 the projection with the expansion of the coordinate

Example 2

If $\Omega = R(\alpha) = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}$ is the rotation operator and

$$B = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{bmatrix}$$

We have (17)

$$W = A(B^T B)^{-1} B^T = \Omega Q =$$

$$\begin{bmatrix} \cos(\alpha) & \sin(\alpha) & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{bmatrix} \left(\begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{bmatrix}^T \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{bmatrix} \right)^{-1} \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{bmatrix}^T$$

$$= \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{bmatrix} =$$

$$\begin{bmatrix} q_{11}\cos(\alpha) + q_{21}\sin(\alpha) & q_{12}\cos(\alpha) + q_{22}\sin(\alpha) & q_{13}\cos(\alpha) + q_{23}\sin(\alpha) \\ -\sin(\alpha)q_{11} + \cos(\alpha)q_{21} & -\sin(\alpha)q_{12} + \cos(\alpha)q_{22} & -\sin(\alpha)q_{13} + \cos(\alpha)q_{23} \\ q_{31} & q_{32} & q_{33} \end{bmatrix} \quad (17)$$

In the associative memory we compute the weight as an operator (memory) that includes invariance. So any association generates output elements of the same universe with the same rules or invariance. In digital computer we only have passive memory, but in the brain or morpho computing we have data and also rules. Any input to the brain changes in a way to have new data associated to the input with the internal rules in the memory.

4 Hopfield neural network and Morpho computing

The Hopfield Network (1982) can be represented by Fig.5.

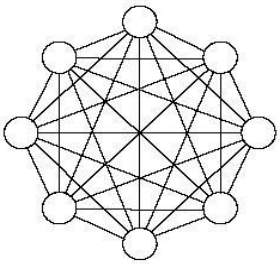


Fig.5 Hopfield Network

Where we can see that Hopfield neural network is a self or bidirectional (loop) of neural network. As described above, in the associative memory we have

$$W = YX^T$$

In Hopfield neural network X=Y so we have

$$W = XX^T$$

Because

$$X = \Gamma(x_k) = \textit{field vector}$$

Hopfield Network Basis $\Gamma_s(x_k) = \textit{set of field vectors}$ fields for n neurons (positions x_k) and correlations as projection operators for orthogonal set of fields is (18).

$$w_{i,j} = \begin{bmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,n} \\ w_{2,1} & w_{2,2} & \dots & w_{2,n} \\ \dots & \dots & \dots & \dots \\ w_{n,1} & w_{n,2} & \dots & w_{n,n} \end{bmatrix}$$

$$= \begin{bmatrix} \Gamma_1(x_1) & \Gamma_2(x_1) & \dots & \Gamma_p(x_1) \\ \Gamma_1(x_2) & \Gamma_2(x_2) & \dots & \Gamma_p(x_2) \\ \Gamma_1(x_3) & \Gamma_2(x_3) & \dots & \Gamma_p(x_3) \\ \dots & \dots & \dots & \dots \\ \Gamma_1(x_n) & \Gamma_2(x_n) & \dots & \Gamma_p(x_n) \end{bmatrix} \begin{bmatrix} \Gamma_1(x_1) & \Gamma_2(x_1) & \dots & \Gamma_p(x_1) \\ \Gamma_1(x_2) & \Gamma_2(x_2) & \dots & \Gamma_p(x_2) \\ \Gamma_1(x_3) & \Gamma_2(x_3) & \dots & \Gamma_p(x_3) \\ \dots & \dots & \dots & \dots \\ \Gamma_1(x_n) & \Gamma_2(x_n) & \dots & \Gamma_p(x_n) \end{bmatrix}^T$$

$$= \sum_s \Gamma_s(x_i)\Gamma_s(x_j) \qquad (18)$$

We remember that w_{ij} is the connection element or self associative memory.

$$Aw = \begin{bmatrix} F(x_1) \\ F(x_2) \\ F(x_3) \\ \dots \\ F(x_n) \end{bmatrix} = \begin{bmatrix} \Gamma_1(x_1) & \Gamma_2(x_1) & \dots & \Gamma_p(x_1) \\ \Gamma_1(x_2) & \Gamma_2(x_2) & \dots & \Gamma_p(x_2) \\ \Gamma_1(x_3) & \Gamma_2(x_3) & \dots & \Gamma_p(x_3) \\ \dots & \dots & \dots & \dots \\ \Gamma_1(x_n) & \Gamma_2(x_n) & \dots & \Gamma_p(x_n) \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_p \end{bmatrix}$$

$$= w_1 \begin{bmatrix} \Gamma_1(x_1) \\ \Gamma_1(x_2) \\ \Gamma_1(x_3) \\ \dots \\ \Gamma_1(x_n) \end{bmatrix} + w_2 \begin{bmatrix} \Gamma_2(x_1) \\ \Gamma_2(x_2) \\ \Gamma_2(x_3) \\ \dots \\ \Gamma_2(x_n) \end{bmatrix} + \dots + w_p \begin{bmatrix} \Gamma_n(x_1) \\ \Gamma_n(x_2) \\ \Gamma_n(x_3) \\ \dots \\ \Gamma_n(x_n) \end{bmatrix} \qquad (19)$$

For the extension of the associative memory we can enlarge the Hopfield model with a more complex projection operator where X are not orthonormal set of vectors.

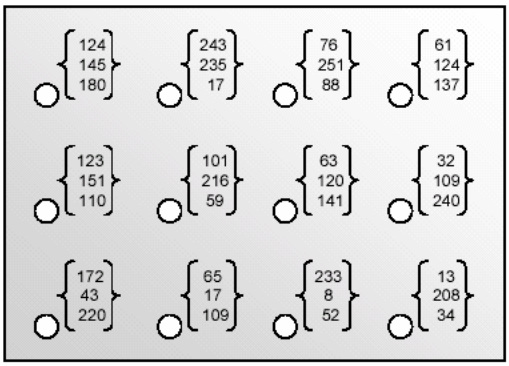
$$w = (A^T A)^{-1} AF$$

$$F^* = A(A^T A)^{-1} AF = WF = QF$$

In conclusion, the novel Hopfield–Like network has the property to be non Euclidean,memeory matrix. Self associative memory is given by general projection operator where the elementary basis vectors X are dependent on each other and there is the correlation among the elementary basis fields.

5 Kohonen Self Organizing Maps by Morpho computing

Given the set of points with three dimensions vectors of weights w_{ij},



The basic principle of the Self-Organizing Map is to adjust these weight vectors until the map represents a picture of the input data set. The goal of learning in the Self-Organizing Map is to cause different parts of the network to respond similarly to certain input patterns. Fig.6 shows the principle of input and output in Self-Organizing Map.

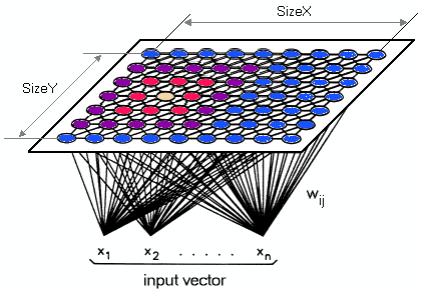


Fig.6 the principle of input and output in Self-Organizing Map

In morpho computing the similarity between input vectors X and weights vectors W can be obtained by the projection operator Q.

$$QW = X(X^T X)^{-1} X^T W \qquad (20)$$

For the orthogonality of the projection operator the QW vectors of weights is at the minimum of the projection space of the inputs.

Example 3

Given the input vectors

$$x_1 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, x_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, X = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

$$w_1 = \begin{bmatrix} 0.7 \\ 0.2 \\ 0.5 \\ 0.9 \end{bmatrix}, w_2 = \begin{bmatrix} 0.1 \\ 0.7 \\ 0.3 \\ 0.6 \end{bmatrix}, w_3 = \begin{bmatrix} 0.32 \\ 0.4 \\ 0.3 \\ 0.63 \end{bmatrix}, W = \begin{bmatrix} 0.7 & 0.1 & 0.32 \\ 0.2 & 0.7 & 0.4 \\ 0.5 & 0.3 & 0.3 \\ 0.9 & 0.6 & 0.63 \end{bmatrix}$$

So we have

$$QW = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \left(\begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}^T \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}^T \begin{bmatrix} 0.7 & 0.1 & 0.32 \\ 0.2 & 0.7 & 0.4 \\ 0.5 & 0.3 & 0.3 \\ 0.9 & 0.6 & 0.63 \end{bmatrix}$$

$$= \begin{bmatrix} 0.45 & 0.4 & 0.36 \\ 0.49 & 0.4 & 0.36 \\ 0 & 0 & 0 \\ 0.9 & 0.6 & 0.63 \end{bmatrix}$$

The weights $w_{ij} = \begin{bmatrix} 0.45 & 0.4 & 0.36 \\ 0.49 & 0.4 & 0.36 \\ 0 & 0 & 0 \\ 0.9 & 0.6 & 0.63 \end{bmatrix}$ are the most similar weights to the input data given the initial random values.

Example 4

The following is the canonical form by projection operator as a particular case of Kohonen network modeled by morpho computing.

Given the input vectors $X = \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ -1 & 1 \\ -1 & -1 \end{bmatrix}$ and the weights

$$W = \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ -1 & 1 \\ -1 & -1 \end{bmatrix} + \begin{bmatrix} 0.5 & 1 \\ 0.5 & 1 \\ 0.5 & 1 \\ 0.5 & 1 \end{bmatrix} = X + D$$

We have the new weights

$$QW = \begin{bmatrix} 1 & 1 \\ -1 & 1 \\ 1 & -1 \\ -1 & -1 \end{bmatrix} \left(\begin{bmatrix} 1 & 1 \\ -1 & 1 \\ 1 & -1 \\ -1 & -1 \end{bmatrix}^T \begin{bmatrix} 1 & 1 \\ -1 & 1 \\ 1 & -1 \\ -1 & -1 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & 1 \\ -1 & 1 \\ 1 & -1 \\ -1 & -1 \end{bmatrix}^T \left(\begin{bmatrix} 1 & 1 \\ 1 & -1 \\ -1 & 1 \\ -1 & -1 \end{bmatrix} + \begin{bmatrix} 0.5 & 1 \\ 0.5 & 1 \\ 0.5 & 1 \\ 0.5 & 1 \end{bmatrix} \right) = \begin{bmatrix} 1 & 1 \\ -1 & 1 \\ 1 & -1 \\ -1 & -1 \end{bmatrix}$$

The projection operator changes the weights in a way to eliminate the given translation to be equal to input vectors.

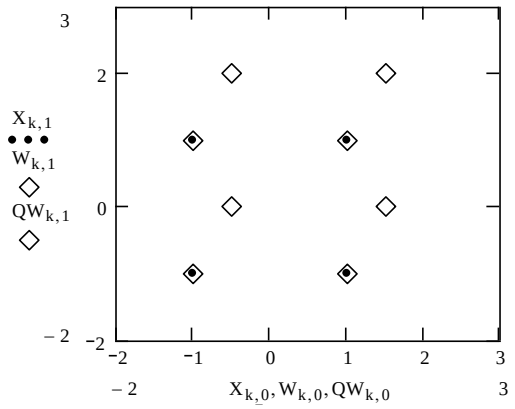


Fig.7 the position before rotation of W

In Fig.7, the input vectors are the black points symmetric to the origin. The diamond points are non-symmetric and the Kohonen neurons that assume the same black point position after the projection operator. Now we rotate and translate *W* to get Fig.8. The non-symmetric point or neurons assume the same property (symmetry) of the input vectors but are not equal (similar) to the input symmetric vectors or points.

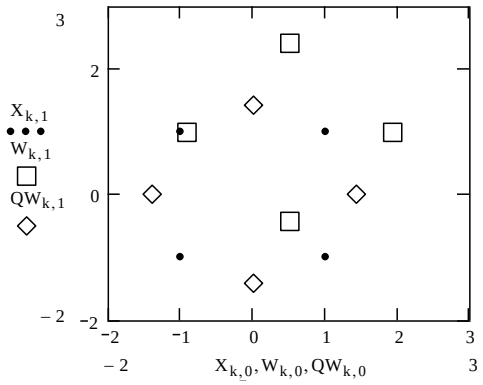


Fig.8 the position after rotation of W

After the projection, QW has the same symmetry of the input vector. In conclusion, the projection operator is like the self organizing maps that preserves the form or properties (symmetry) of the input vector. The similarity means the same morphology in the morpho computing.

6 Morpho computing Learning with noising to learn patterns and retrieve patterns

In many cases we have to compute the weights that are orthogonal to the given input data. In fact for neural associative memory the goal is to design a neural network capable of memorizing a large set of patterns from a data set X (learning phase), and recalling them later in presence of noise (recall phase). Each pattern $x = \{x_1, x_2, ..., x_n\}$ is a vector of length *n* or a field, our focus is to memorize the patterns with strong local correlation among the entries. More specially, we divide the entries of each pattern x into L overlapping subpatterns due to overlaps, a pattern node can be a member of multiple subpatterns. We denote the *i*-th subpattern by $x^i = \left\{ x_1{}^i, x_2{}^i, ..., x_{n_i}{}^i \right\}$

Fig.9 gives the example of the pattern and sub pattern.

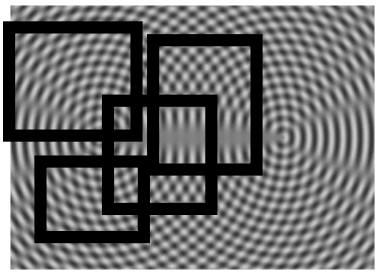


Fig.9 Any box is sub pattern of the wave interference figure.

The learning process is given by the weight matrix W for which we have the orthogonal condition (21).

$$W^i x^i = 0 \qquad (21)$$

The weights $w_{i,j}$ is the dual space of the subpatterns system. Now with the projection operator it is possible to compute these weights with only one step as shown in (22).

$$W = w_{i,j} = I - X[X^T X]^{-1} X^T \qquad (22)$$

In fact we have (23)

$$WX = (I - X[X^T X]^{-1} X^T)X = 0 \qquad (23)$$

Example 5

Given the pattern $x = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$,It includes two sub patterns $x^1 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, x^2 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$

Now the weights are

$$W = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \left(\begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}^T \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \right)^{-1} \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}^T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

To retrieve the sub pattern from the weights *W*, we compute the inverse process.

$$X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \left(\begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}^T \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}^T = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

Now when we have a pattern with noise given by expression (24)

$$y^i = x^i + \varepsilon^i \qquad (24)$$

We can detect the noise because we have (25)

$$Wy = Wx + W\varepsilon^{i} = W\varepsilon^{i} \qquad (25)$$

So we can clean the sub-partner form its noise and retrieve the original pattern x. The task of a neural associative memory is to retrieve a set of previously memorized patterns from their noisy versions by using a network of neurons. We show that with the projection operator we can easily solve the previous problems that are similar to the Convolutional Neural Associative Memories.

7 Supervised neural network by morpho computing

Given one neuron with two inputs and one output, shown in Fig.10.

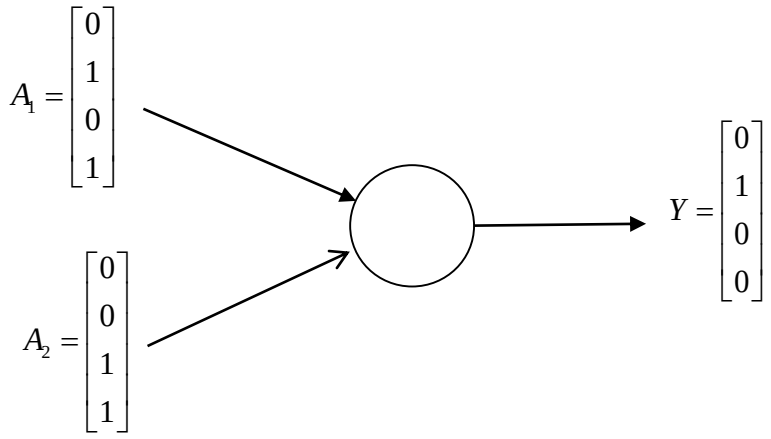


Fig.10 the neuron with two inputs and one output

we compute the weights and threshold of the neuron. Given A the input vectors, *w* the weights, and Y the desired vector,

$$A = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}, Y = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix},$$

we have the equation (26).

$$Aw = Y \qquad (26)$$

or

$$\begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

What we want to solve is to get the weights by the input and desired function Y. According to the previous equation, we get the weights *w* by (27).

$$\begin{aligned} &A^T A w = A^T Y \ , \\ &w = (A^T A)^{-1} A^T Y \end{aligned} \tag{27}$$

$$\text{And } w = (A^T A)^{-1} A^T Y = \begin{bmatrix} \frac{2}{3} \\ -\frac{1}{3} \end{bmatrix}$$

With *w* we can compute the projection *QY* of *Y* into the input world *A* by (28).

$$A\, w = A(A^T A)^{-1} A^T Y = QY \tag{28}$$

$$QY = A\, w = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \frac{2}{3} \\ -\frac{1}{3} \end{bmatrix} = \frac{2}{3} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} - \frac{1}{3} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} = w_1 A_1 + w_2 A_2 = \begin{bmatrix} 0 \\ \frac{2}{3} \\ -\frac{1}{3} \\ \frac{1}{3} \end{bmatrix} \tag{29}$$

For the projection operator the linear combination of the column vectors in (29) assumes the minimal value of difference QY-Y among all possible linear combinations.

Because the neuron-like neuron can be given biases by introducing an extra input to each unit which always has a value of 1, the weight θ on this extra input is called the bias and is equivalent to a threshold of the opposite sign. To compute the threshold we use the expression (30).

$$\theta = \frac{\min[(QY)Y] + \max[(QY)(1-Y)]}{2} \tag{30}$$

In the previous example we have

$$\theta = \frac{\min[0, \frac{2}{3}, 0, 0] - \max[0, 0, -\frac{1}{3}, \frac{1}{3}]}{2} + \max[0, 0, -\frac{1}{3}, \frac{1}{3}] = \frac{\frac{2}{3} - \frac{1}{3}}{2} + \frac{1}{3} = \frac{1}{2}$$

Calling the output y_j we can write (31).

$$y_j = f\ [\sum_i w_i X_{i,j} - \theta] \tag{31}$$

Where f is the step function (actually known as the Heaviside function) and

$$f(x) = 1 \quad x > 0$$
$$f(x) = 0 \quad x \leq 0$$

In this situation no hidden neurons are necessary. With projection operator we find the strength of the connections with the desired output without iteration process. Among the eight desired functions Y in (32), only $Y = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$ cannot be solved by projection operator.

$$\mathrm{Y} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} \tag{32}$$

Now,

$$A = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}, Y = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

This is a XOR Boolean function. We compute QY by the projection operator.

$$QY = A(A^T A)^{-1} A^T = \begin{bmatrix} 0 \\ \frac{1}{3} \\ \frac{1}{3} \\ \frac{2}{3} \end{bmatrix}$$

We compare the vectors in (33). We know that in any neuron we have that the outputs are separate in two parts. The first are the independent parts given by the output function

$$\begin{bmatrix} F_1 \\ F_2 \end{bmatrix} = \begin{bmatrix} w_1 - \theta \\ w_2 - \theta \end{bmatrix} \tag{33}$$

And the other part is dependent output given by the linear superposition of the independent parts. So we have the dependent part is

$$F_1 + F_2 = w_1 + w_2 - 2\theta$$

We can compute all the possible output of the neuron in this way

$$\begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \end{bmatrix} = \begin{bmatrix} 0 \\ F_1 \\ F_2 \\ F_1 + F_2 \end{bmatrix}$$

To obtain the wanted Boolean function 0110, we must put a constrains to the possible output in this way

$$\begin{bmatrix} 0 \\ F_1 > 0 \\ F_2 > 0 \\ F_1 + F_2 < 0 \end{bmatrix}$$

This Boolean constrain is in conflict with the linear superposition of the outputs. In fact we have

$$if\,(F_1 > 0) and (F_2 > 0)\ \text{then}\ F_1 + F_2 > 0$$

But the logic conclusion is in conflict with the Boolean constrain for which

$$F_1 + F_2 < 0$$

Now we have the conflicting between Y and QY. We can eliminate the conflict situation by an extension of the input space in this way

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

That is a new layer. We remark that we adjoin a new colon that assume value 1 in the exact point where the conflict grow up. Now if we use the use the projection operators we have the weights

$$w = (A^T A)^{-1} A^T = \begin{bmatrix} 1 \\ 1 \\ -2 \end{bmatrix}$$

So we have (34).

$$QY = A(A^T A)^{-1} A^T = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \qquad (34)$$

For

$$A = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}, Y = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

With the projection operator we have

$$QY = \begin{bmatrix} 0 \\ \frac{1}{3} \\ \frac{1}{3} \\ \frac{2}{3} \end{bmatrix}, w = \begin{bmatrix} \frac{1}{3} \\ \frac{1}{3} \end{bmatrix}$$

With the threshold

$$\theta = \frac{\frac{2}{3} + \frac{1}{3}}{2} = \frac{1}{2}$$

We can solve the Boolean function 0001.
In conclusion with the solution of the conflict and the solution for the Boolean function 0001, we can built the neural network that solve the XOR problem. So we have Fig.11.

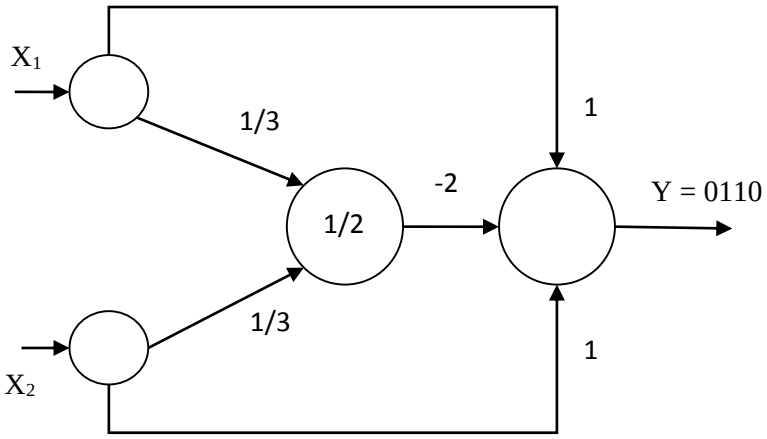


Fig.11 Weights and threshold for XOR Boolean function 0110 of the hidden neuron that solve the original conflict and the three dimension input to the mother neuron.

The following is a more complex example.

$$Y=\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \end{bmatrix}=\begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, QY=\begin{bmatrix} 0 \\ 0 \\ 0.5 \\ 0 \\ 0.5 \\ 0 \\ 0,5 \\ 0.5 \end{bmatrix}, X=\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

As in the two inputs neuron we have this form of the Boolean function

$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}\begin{bmatrix} F_1 \\ F_2 \\ F_3 \end{bmatrix}=\begin{bmatrix} 0 \\ F_1 \\ F_2 \\ F_1+F_2 \\ F_3 \\ F_1+F_3 \\ F_2+F_3 \\ F_1+F_2+F_3 \end{bmatrix}=\begin{bmatrix} 0 \\ F_1 \\ F_2 \\ F_1+F_2 \\ F_3 \\ F_1+F_3 \\ F_2+F_3 \\ \frac{1}{2}(F_1+F_2)+\frac{1}{2}(F_1+F_3)+\frac{1}{2}(F_2+F_3) \end{bmatrix}$$

Now for the Boolean function 00110101 we create this constrain

$$\begin{bmatrix} 0 \\ F_1 < 0 \\ F_2 > 0 \\ F_1 + F_2 > 0 \\ F_3 < 0 \\ F_1 + F_3 > 0 \\ F_2 + F_3 < 0 \\ \frac{1}{2}(F_1 + F_2) + \frac{1}{2}(F_1 + F_3) + \frac{1}{2}(F_2 + F_3) > 0 \end{bmatrix}$$

So we have

$$F_2 > F_1 \qquad (35)$$

and for $F_1 + F_3 > 0$, $F_2 + F_3 < 0$

We have

$$F_1 + F_3 > F_2 + F_3$$

And

$$F_1 > F_2 \qquad (36)$$

But (35) and (36) are in conflict so the previous Boolean function cannot solved. Elimination of the conflict by compensation. To eliminate the conflict we enlarge the input space by a new input which value 1 is put in the same position where we have the conflict. So we have the new input structure

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix}, Y = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

With the projection operator we have

$$QY = A(A^T A)^{-1} A^T Y = \begin{bmatrix} 0 \\ 0 \\ 0.4 \\ 1.2 \\ -0.1 \\ 0.7 \\ 0.3 \\ 1.1 \end{bmatrix}, W = (A^T A)^{-1} A^T Y = \begin{bmatrix} 0.8 \\ 0.4 \\ -0.1 \\ -0.8 \end{bmatrix}$$

We know that the Boolean function 01000000 has no conflicts. In fact we have

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, QY = \begin{bmatrix} 0 \\ 0.373 \\ -0.125 \\ 0.25 \\ -0.125 \\ 0.25 \\ -0.25 \\ 0.125 \end{bmatrix}, X = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, w = \begin{bmatrix} 0.375 \\ -0.125 \\ -0.125 \end{bmatrix}$$

The neural network for the Boolean function 00110101 is in Fig.12.

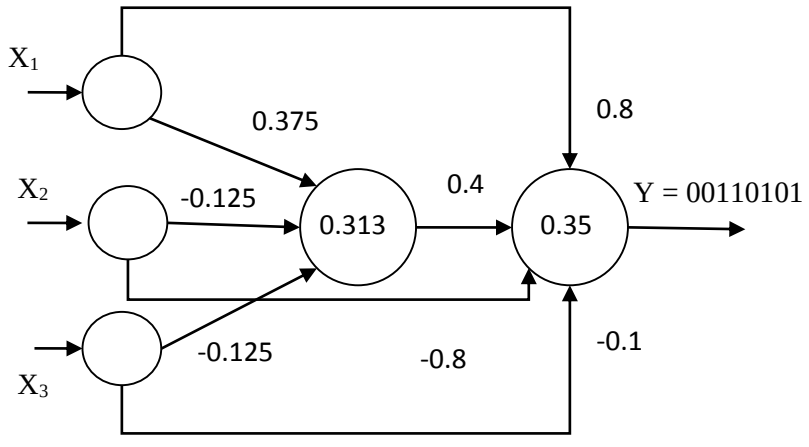


Fig.12 Weights and threshold for Boolean function 00110101 of the hidden neuron that solve the original conflict and the four dimension input to the mother neuron.

8 Electronic systems and Morpho Computing

Given Fig.13 which is an electronic system with impedance matrix Z and connection matrix X we have the relation between voltages and currents as an associative memory or MIMO, we have (37).

$$i = X(X^T ZX)^{-1} X^T v \quad (37)$$

Now we can see (38)

$$v^* = Zi = ZX(X^T ZX)^{-1} X^T v = Qv \quad (38)$$

In fact we have (39)

$$Q = ZX(X^T ZX)^{-1} X^T$$
$$Q^2 = ZX(X^T ZX)^{-1} X^T ZX(X^T ZX)^{-1} X^T = ZX(X^T ZX)^{-1} X^T \quad (39)$$

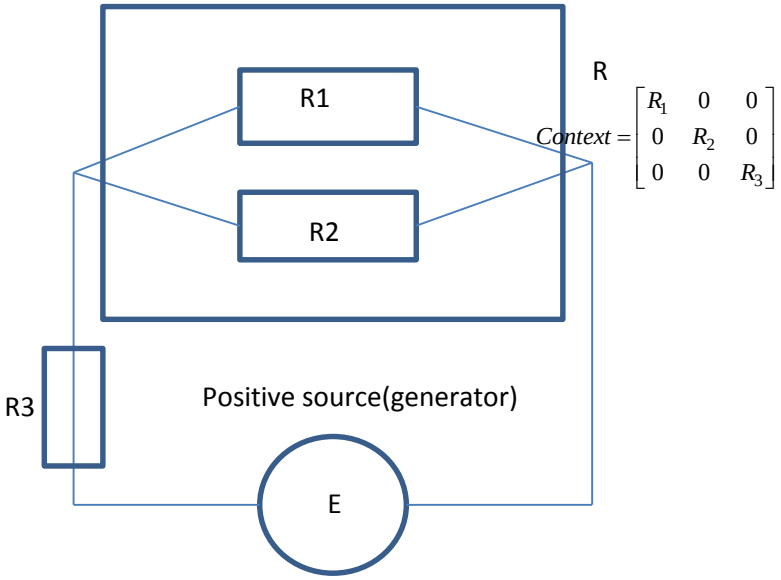


Fig.13 electronic system

$$I = \begin{bmatrix} i_1 \\ i_2 \\ i_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \end{bmatrix} = \begin{bmatrix} E \\ E \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}^T \begin{bmatrix} R_1 & 0 & 0 \\ 0 & R_2 & 0 \\ 0 & 0 & R_3 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}^T \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

$$v^* = \begin{bmatrix} v_1 * \\ v_2 * \\ v_3 * \end{bmatrix} = \begin{bmatrix} R_1 & 0 & 0 \\ 0 & R_2 & 0 \\ 0 & 0 & R_3 \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \\ i_1 + i_2 \end{bmatrix} = \begin{bmatrix} R_1 & 0 & 0 \\ 0 & R_2 & 0 \\ 0 & 0 & R_3 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \end{bmatrix}$$

Where $\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$ is the connection matrix and $\begin{bmatrix} R_1 & 0 & 0 \\ 0 & R_2 & 0 \\ 0 & 0 & R_3 \end{bmatrix}$ is the impedance matrix Z.

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}^T \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} E \\ E \end{bmatrix} = \textit{Source Force}$$

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}^T \begin{bmatrix} R_1 & 0 & 0 \\ 0 & R_2 & 0 \\ 0 & 0 & R_3 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \end{bmatrix} = \begin{bmatrix} E \\ E \end{bmatrix} = \textit{source of flow}$$

Now with the same model we can compare unsupervised and supervised neural network and also the electronic system. Because the brain is an electronic system we can establish a bridge between artificial neural network and natural neural network.

9 Conclusion

In this paper we show that supervised and unsupervised neural network can be computed by a new model that takes care of the invariance or rules in the samples as parts of special universe. In traditional physical system rules are modeled by symbolic differential equations that we solve to find the behavior of elements in the physical universe. Now we know that these differential equations are very abstract and difficult to find the wanted behavior, and the brain cannot use differential equations or symbolic expressions as we know in traditional mathematical sense. The brain uses samples that include the universal rules in the implicit way. With samples we can build associative memory with traditional orthonormal property or propagator that extends the traditional associative memory. Now the brain takes sensor input information and associative internal structure that changes the sensor information in a way to have internal data that satisfy the rules which are previously learned by samples of input and output (associative memory or memory with rules). So we think that for digital memory in the computer and associative memory in the morpho computing, the former is a passive memory and the latter is an active memory that changes the data in a way to include the rules (morpho) of the external universe or environment in an implicit way. With extension of the associative memory and morpho computing we include Hopfield neural network, Kohonen self-organizing maps, pattern recognition and supervised neuron. For supervised neuron we create a new algorithm by which it is possible to avoid most of the problems of back propagation. To show the analogy with physical system we implement the same

morpho computing also for a physical device as an electrical circuit where rules are included in the process of the computation. The difference between the brain and universal physics is that rules in the physical part are fixed and unable to be changed, and in the brain there are also rules as in the physical universe but we can control the rules to obtain our aims.

References

[1] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice-Hall, USA, 1999.

[2] I. T. Jolliffe, *Principal Component Analysis*, Springer, 2002. http://dx.doi.org/10.1007/b98835

[3] H.-A. Loeliger, An introduction to factor graphs, *IEEE Signal Processing Magazine*, **21** (2004), 28-41. http://dx.doi.org/10.1109/msp.2004.1267047

[4] T. Kohonen, *Self-Organizing Maps*, Springer, 2001.

[5] T. Kohonen, The self-organizing map, *Proceedings of the IEEE*, **78** (1990), no. 9, 1464-1480. http://dx.doi.org/10.1109/5.58325

[6] T. Kohonen et al., Engineering applications of the self-organizing map, *Proceedings of the IEEE*, **84** (1996), no. 10, 1358-1384. http://dx.doi.org/10.1109/5.537105

[7] Y. H. Hu, *Associative Learning and Principal Component Analysis*, Lecture 6 Notes, 2003.

[8] R. P. Lippmann, An introduction to computing with neural nets, *IEEE Transactions of Acoustics, Speech, and Signal Processing, ASSP*, **4** (1987), 4-22. http://dx.doi.org/10.1109/massp.1987.1165576

[9] R. McEliece and et. al., The capacity of Hopfield associative memory, *IEEE Transactions of Information Theory*, **33** (1987), 461-482. http://dx.doi.org/10.1109/tit.1987.1057328

[10] G. X. Ritter and P. Sussner, An introduction to morphological neural networks, *Proceedings of the 13th International Conference on Pattern Recognition*, Vienna, Austria, **4** (1996), 709-711. http://dx.doi.org/10.1109/icpr.1996.547657

[11] G. X. Ritter, P. Sussner and J. L. Diaz de Leon, Morphological associative memories, *IEEE Transactions on Neural Networks*, **9** (1998), no.2, 281-293.

http://dx.doi.org/10.1109/72.661123

[12] Germano Resconi, Robert Kozma, Geometry image of neurodynamics, *NCTA 2012*, paper no. 13.

[13] Walter J. Freeman, *Mass Action in The Nervous System*, Academic Press Inc., New York, San Francisco, London, 1975.

[14] R. Kozma W. J. Freeman, Intermittent spatio – temporal desynchronization and sequenced synchrony in ECoG signals, *Chaos: An Interdisciplinary J. of Nonlinear Science*, **18** (2008), 037131. http://dx.doi.org/10.1063/1.2979694

[15] J. Rinzel and B. Ermentrout, Analysis of Neural Excitability and Oscillations, *Methods in Neural Modeling*, ed. C. Koch and I. Segev, 251-291, MIT Press, 1998.

[16] Germano Resconi, Modelling fuzzy cognitive map by electrical and chemical equivalent circuits, *Information Science 2007, Proceedings of the 10th Joint Conferenc*e, Salt lake City, USA, 18-24 July (2007), 1029-1035. http://dx.doi.org/10.1142/9789812709677_0144

[17] Germano Resconi, Vason P. Srini, Electrical circuit as a morphogenetic system, *GEST International Transactions on Computer Science and Engineering*, **53** (2009), no. 1, 47-92.

[18] Greg S. Snider of Hewlett Packard Laboratory, *First Memristor and Memristive Systems Symposium*, University of California, Berkeley, 2008.

[19] Carver Mead, Neuromoprhic Electronic Systems, *Proceeding of the IEEE*, **78 (**1990), no.10, 1629-1636. http://dx.doi.org/10.1109/5.58356

[20] Antonio B. Torralba, *Analogue Architectures for Vision Cellular Neural Networks and Neuromorphic Circuits*, Doctorat thesis, , Laboratory of Images and Signals, Institute national Polytechnique de Grenoble, 1999.

[21] Pavel Pokorny, Geodesics revisited, *Chaotic Modeling and Simulation (CMSIM)*, 281 - 298, 2012.

[22] Bob Rink, *Lecture notes on Geometric Mechanics and Dynamics*, December 14, 2007.

[23] David E. Johnson, Johnny R. Johnson, John L. Hilburn , Peter D. Scott, *Electrical Circuit Analysis*, John Wiley &Sons Inc., 1999.

[24] J. Awrejcewicz, D. Sendkowski, Geometric Analysis of the dynamics of a

double pendulum, *Journal of Mechanics of Materials and Structures*, **2** (2007), no. 8, 1421-1430. http://dx.doi.org/10.2140/jomms.2007.2.1421

[25] Karl H. Pribram, *Languages of the Brain: Experimental paradoxes and principles in neuropsychology*, Brandon House, New York, 1971.

[26] Karl H. Pribram, *Brain and Perception: Holonomy and Structure in Figural Processing*, Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1991. http://dx.doi.org/10.4324/9780203728390

[27] J. L. Hindmarsh and R. M. Rose, A model of neuronal bursting using three coupled first order differential equations, *Proceedings of the Royal Society B: Biologic. Sciences*, **221** (1984), 87-102. http://dx.doi.org/10.1098/rspb.1984.0024

[28] J. L. Hindmarsh and R. M. Rose. A model of the nerve impulse using two first-order differential equations, *Nature*, **296** (1982), 162-164. http://dx.doi.org/10.1038/296162a0

[29] Walter J. Freeman, *Mass Action In The Nervous System- Examination of the Neurophysiological Basis of Adaptive Behavior through the EEG*, Academic Press, New York, San Francisco, London, 1975. http://dx.doi.org/10.1016/0022-510x(77)90054-5

[30] Carver Mead, M. Ismail, *Analog VLSI and Neural Systems*, Kluwer Academic Publishers, 1989. http://dx.doi.org/10.1007/978-1-4613-1639-8

Mathematical Appendix

Loop Transformation

Given the propagator (1)

$$B \to A, P_1 = A(B^T B)^{-1} B^T$$

$$A \to B, P_2 = B(A^T A)^{-1} A^T \qquad (1)$$

This is a loop of morpho computing network where A and B are samples of two different categories or universe with different internal rules, as shown in Fig.1.

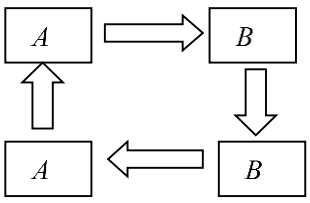


Fig.1 a loop of morpho computing network

When we begin with A and get to B, after from B we come back to A, we have the chain of operations (2).

$$P_2P_1 = A(B^TB)^{-1}B^TB(A^TA)^{-1}A^T = A(A^TA)^{-1}A^T = Q_A \quad (2)$$

That gives us the projection operator of A into itself. So we have the operation (3).

$$A(A^TA)^{-1}A^Ty = Q_Ay \qquad (3)$$

Where y does not belong to the category or set with rules where there is the sample A. Now Q_Ay belongs to the category where there is the sample A and has the minimum distance from y.

Similarly, when we begin with B, propagate to A, and end with the same B we have (4).

$$P_1P_2 = B(A^TA)^{-1}A^TA(B^TB)^{-1}B^T = B(B^TB)^{-1}B^T = Q_B \quad (4)$$

Operation (5) shows any external element y is changed by the projection operator Q_B has the same rules of the universe that includes the sample B with the minimum distance.

$$B(B^TB)^{-1}B^Ty = Q_By \qquad (5)$$

A more complex associative memory loop can be given by Fig.2.

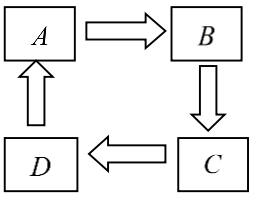


Fig.2 a more complex loop of morpho computing network

We have (6).

$$P_1P_2P_3P_4 = A(D^TD)^{-1}D^TD(C^TC)^{-1}C^TC(B^TB)^{-1}B^TB(A^TA)^{-1}A^T = A(A^TA)^{-1}A^T \quad (6)$$

We can also give the image of the projection operator or loop by boxes. (Fig.3), which transforms an external element y to an internal element belonging to the category (set with rules or morphisms).

Sample A of a category

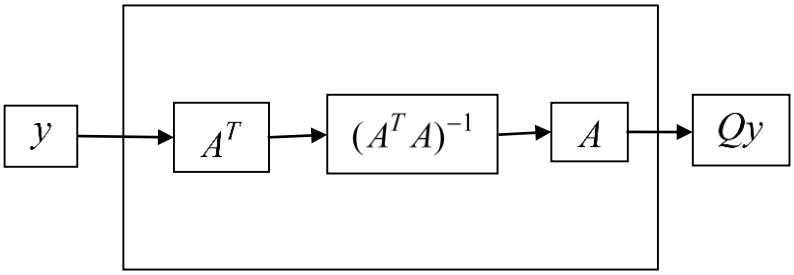


Fig. 3 loop representation of projection operator

The morpho computing loop is completely different from feedback loop. In fact in the feedback we have input and output of data, and in the morpho computing we have input and output vectors with rules. So the projection operator is a loop that changes the external vectors with external rules to the internal vectors with rules given by samples.

Example 1

Given $A = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$ we have $Aw = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 0 \\ w_1 \\ w_2 \\ w_1 + w_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = y$

, .

y is a vector where y_1 is always zero , y_2 and y_3 are independent variables and y_4 is a dependent variable. Now we solve the previous equation and we move from y so

$$A^T Aw = A^T y$$
$$w = (A^T A)^{-1} A^T y$$

So now we come back to $y*$ to close the cycle so we have

$$Aw = A(A^T A)^{-1} A^T y = Qy = y*$$

So if *y* is a free vector, y* is not free but only components are free. Q is the projection operator as we can see in Fig.4 where y is the desired vector.

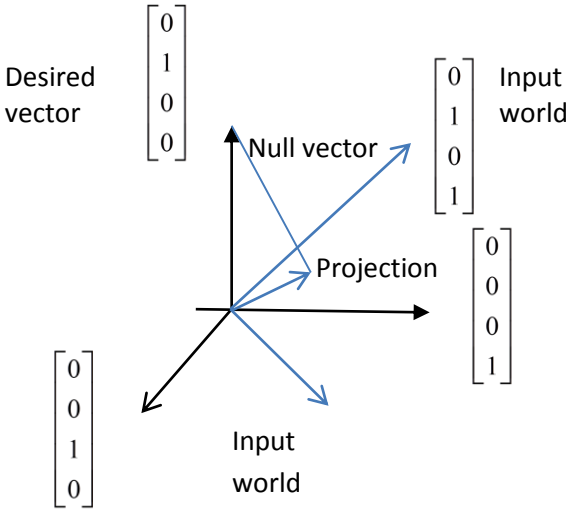


Fig. 4 the relation among input, desired and projection vectors

Received: July 30, 2015; Published: September 25, 2015