Orthogonal considerations in the design of neural networks for function approximation

Bruno Francois

Orthogonal considerations in the design of neural networks for function approximation

Article *in* Mathematics and Computers in Simulation · June 1996

DOI: 10.1016/0378-4754(95)00062-3

CITATIONS
13

READS
259

1 author:

Bruno Francois
École Centrale de Lille
253 PUBLICATIONS **6,033** CITATIONS

SEE PROFILE

IMACS, Mathematics and Computers in Simulation, 41 (5-6), July 1996

Orthogonal Considerations in the Design of Neural Networks For Function Approximation

FRANCOIS B.

L2EP, E.C. Lille, Cité Scientifique BP48, 59651 Villeneuve d'Ascq, France
Tel: 20 33 53 88, Fax: 20 33 54 54, E_mail: bruno@leepi1.ec-lille.fr

ABSTRACT

Two problems occur in the design of feedforward neural networks: the choice of the optimal architecture and the initialization. Generally, input and output data of a system (or a function) are measured and recorded. Then, experimenters wish to design a neural network to map exactly these output values.

By formulating this problem as a continuous approximation problem, this paper shows that the use of orthogonal functions is a partial optimization in the choice of hidden functions.

Parameters initialization is obtained by using the knowledge of input and output data in the calculation of a discrete approximation. The hidden weights are found by constructing orthogonal directions on which the input values are represented. The pseudo inverse is used to determine output weights such that the Euclidean distances between neural responses and output values are minimized.

1 Introduction

Conventional supervised learning consists, in a first step, by setting randomly weights and biases, and, after, updating them to reduce a quadratic cost function with a gradient descent procedure. This technique requires a training set, of m desired output data (written $\vec{F}$), for which the corresponding input data ($\vec{I}$) are known. A cost function, E, is usually defined as a sum of the distances between $\vec{F}$ and $\vec{R}$, the responses of the neural net fed with $\vec{I}$. Any configuration of weights and biases, producing a global minima of E, guarantees a correct neural network output $\vec{R}$ for every input values belonging to the training set.

It's well known that finding this configuration with a learning algorithm can be prohibitively slow. More over, the main method for finding the hidden layer size is to increase the hidden layer (and so weight parameters) in a minimal network according some appropriate criteria on E ("Constructive algorithms") [1], or, to eliminate some of the weights in an oversize network according the same criteria ("Pruning technics") [8].

By adjusting the number of hidden units, users modify the shape of the error in the purpose of getting a smaller global minima. Unfortunately, while the global minima is decreasing, more local minima are created and, then, increase (again) the algorithm convergence time. A solution to this problem is to initialize the weights near the global minima before applying weights adaptation. But, this weights calculation is complex mainly because many equal minima may exist.

In this paper, we determine a particular class of hidden functions which involves an unique global minima**.** Afterwards, we present a weights initialization which places the cost function in the attractive domain of the global minima. This initialization is based on the orthogonal representation, in the hidden layer, of the information holding in $\vec{I}$, and, on the minimization of E. A learning algorithm is detailed in section 4 to make this error fall into the global minima.

2 Particular hidden units for continuous approximation

2.1) Mathematical formulation

Approximation theory addresses the problem of interpolating or approximating a function by a selected function[12]. For a multilayer neural network, this selected function is a finite sum of non linear functions. This sum is characterized by:

_ the number of hidden units, which causes the number of parameters: weights and biases

_ the non linearity of unit functions.

Before giving the expression of this sum, let us remind that a unit "j" from a layer "a" indexed with the character ^{a} (see figure 1) performs two tasks:

_ a linear transformation of its inputs: $S_j^a = \sum_{i=1}^{Np} W_{ij}^a . O_i^p + B_j^a$

where S_j^a is the summation into the unit j from the layer ^{a}

B_j^a is the bias of unit j in layer ^{a}

O_i^p is the output of unit i in the previous layer ^{p}

N_P is the number of units in this layer

W_{ij}^a is the connection weight between the unit i in the previous layer ^{p} and the unit "j" in layer ^{a}.

_ a non linear transformation of the resulting summation through the function (φ_j) of unit j:

$O_j^a = \varphi_j\left[S_j^a\right]$.

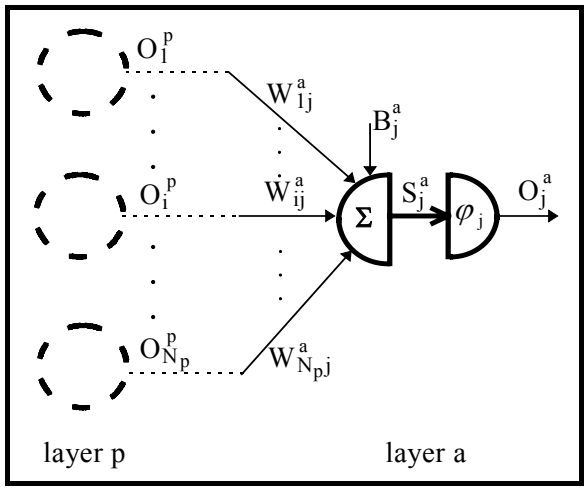

figure 1: Functional description of a unit

Consider the feedforward network (figure 2), designed to approximate a function with N_I inputs variables: $I_1 \ldots I_{N_I}$ and a single output R.

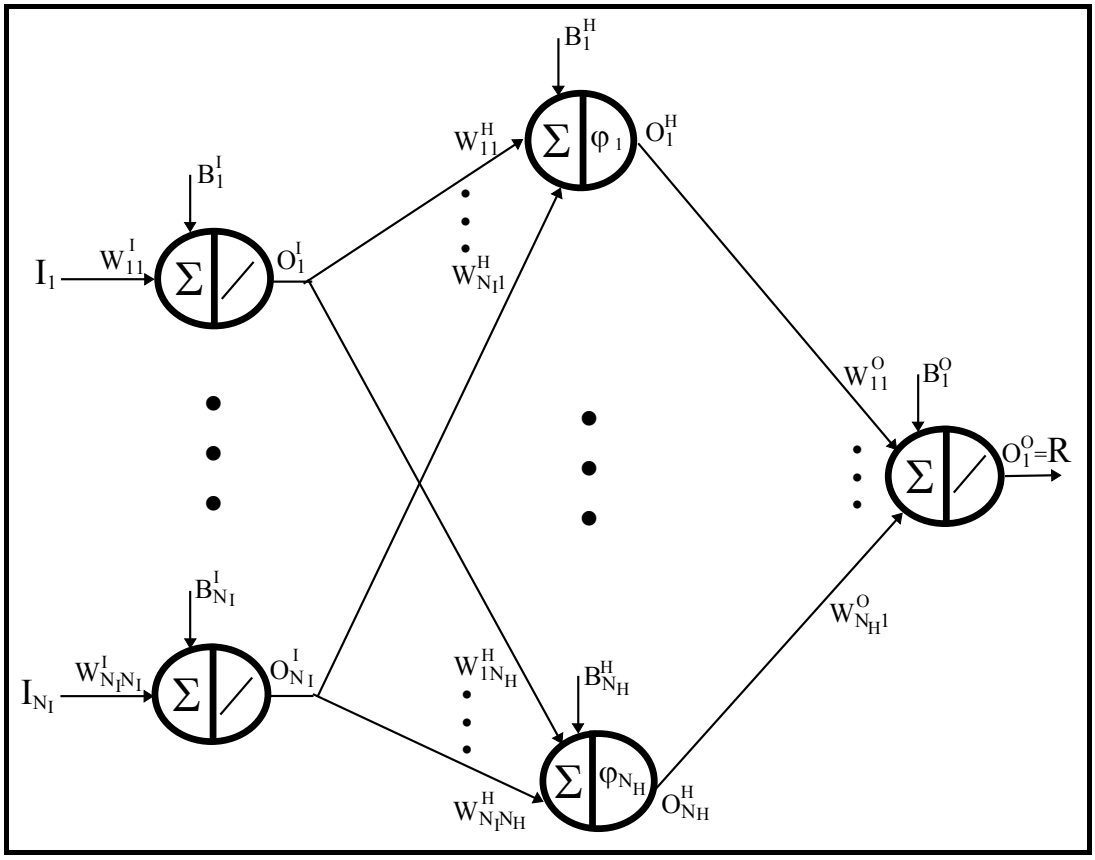

figure 2: Functional description of a neural network

All units from the input layer I have a linear function φ_j, their outputs are a weighted and biased copy of the input variable I_i **(**for $1 \le i \le N_I$**):**

$$O_i^I\left(I_i\right) = S_i^I = B_i^I + W_{ii}^I . I_i \quad (1)$$

These unit outputs O_i^I are mixed with the hidden weights W_{ij}^H as

$$S_j^H\left(I_1 \ldots I_{N_I}\right) = B_j^H + \sum_{i=1}^{N_I} W_{ij}^H . O_i^I\left(I_i\right) \quad (2)$$

and cross the non linear function φ_j:

$$O_j^H\left(I_i \ldots I_{N_I}\right) = \varphi_j\left[\, S_j^H\left(I_i \ldots I_{N_I}\right)\right] \quad (3)$$

As the unit from the output layer has a linear function φ_j, this unit output is

$$O_1^O\left(I_1 \ldots I_{N_I}\right) = S_1^O\left(I_1 \ldots I_{N_I}\right) = B_1^O + \sum_{j=1}^{N_H} W_{j1}^O . O_j^H\left(I_1 \ldots I_{N_I}\right) \quad (4)$$

Then, the response of the neural network can be expressed as a limit sum of non linear functions

$$R\left(I_1 \dots I_{N_I}\right) = \sum_{j=0}^{N_H} W_{j1}^O . \varphi_j \left[\; S_j^H \left(I_1 \dots I_{N_I}\right)\right] \quad (5)$$

with $\varphi_0[\ ...] = 1$ and $W_{01}^O = B_1^O$

2.2) Approximation with orthogonal functions

Providing that φ_j are continuous, monotone, non-linear and bounded functions, a number of papers [2], [3], [4], have proved that this feedforward network may uniformly approximate any continuous function to an arbitrary degree of accuracy.

The accuracy can be measured with the squared Euclidean distance:

$$E\left(I_1 \dots I_{N_I}\right) = \left[f\left(I_1 \dots I_{N_I}\right) - R\left(I_1 \dots I_{N_I}\right)\right]^2 \quad (6)$$

$$= f\left(I_1 \dots I_{N_I}\right)^2 - 2. \sum_{j=0}^{N_H} W_{j1}^O . f\left(I_1 \dots I_{N_I}\right) . \varphi_j \left[S_j^H \left(I_1 \dots I_{N_I}\right)\right]$$

$$+ \sum_{j=0}^{N_H} \sum_{k=0}^{N_H} W_{j1}^O . W_{k1}^O . \varphi_j \left[S_j^H \left(I_1 \dots I_{N_I}\right)\right] . \varphi_k \left[S_k^H \left(I_1 \dots I_{N_I}\right)\right]$$

For a continuous variation of the input variables $I_1 \dots I_{N_I}$ in respective domains $\Delta I_1 \dots \Delta I_{N_I}$, the sum of this distance defines the L_2 norm:

$$\overline{E} = \int_{\Delta I_1} \dots \int_{\Delta I_{N_I}} f\left(I_1 \dots I_{N_I}\right)^2 dI_1 \dots dI_{N_I} \quad - 2. \sum_{j=0}^{N_H} W_{j1}^O . C_j$$

$$+ \sum_{j=0}^{N_H} \sum_{k=0}^{N_H} W_{j1}^O . W_{k1}^O . \int_{\Delta I_1} \dots \int_{\Delta I_{N_I}} \varphi_j \left[S_j^H \left(I_1 \dots I_{N_I}\right)\right] . \varphi_k \left[S_k^H \left(I_1 \dots I_{N_I}\right)\right] \, dI_1 \dots dI_{N_I} \quad (7)$$

with $C_j = \int_{\Delta I_1} \dots \int_{\Delta I_{N_I}} f\left(I_1 \dots I_{N_I}\right) . \varphi_j \left[S_j^H \left(I_1 \dots I_{N_I}\right)\right] dI_1 \dots dI_{N_I}$ (8),

We want to determine the functions φ_j and parameters W_{j1}^O such that $\overline{E}$ is a minimum. A suboptimal solution to this problem is to minimize $\overline{E}$ among the parameters W_{j1}^O and, after, to deduce the functions φ_j.

$$\frac{\partial \overline{E}}{\partial W_{j1}^O} = 0 \Leftrightarrow C_j = \sum_{k=0}^{N_H} W_{k1}^O . \int_{\Delta I_1} \dots \int_{\Delta I_{N_I}} \varphi_j \left[S_j^H \left(I_1 \dots I_{N_I}\right)\right] . \varphi_k \left[S_k^H \left(I_1 \dots I_{N_I}\right)\right] dI_1 \dots dI_{N_I}$$

This system is easily solved if all functions φ_j are orthogonal:

$$\iint_{\Delta S_k \, \Delta S_j} \varphi_j \left[S_j^H \left(I_1 \dots I_{N_I}\right)\right] . \varphi_k \left[S_k^H \left(I_1 \dots I_{N_I}\right)\right] dS_j dS_k = \delta_{jk} . \iint_{\Delta S_k \, \Delta S_j} dS_j dS_k \text{, where}$$

– δ_{jk} is the Kronecker symbol ($\delta_{jk} = 0$ if $j \neq k$, $\delta_{jj} = 1$)

_ ΔS_j and ΔS_k are the variation domains of each hidden units j and k, due to the variation of $I_1 \dots I_{N_I}$ in $\Delta I_1 \dots \Delta I_{N_I}$ following equation *(2)*.Then, the output weights have the following expression: $W_{j1}^O = \dfrac{C_j}{\int_{\Delta I_1} \dots \int_{\Delta I_{N_I}} dI_1 \dots dI_{N_I}}$ for $0 \le j \le N_H$.

By replacing this expression in *(7)*, we find the value of the unique global minima:

$$\overline{E}_{min} = \int_{\Delta I_1} \dots \int_{\Delta I_{N_I}} f\left(I_1 \dots I_{N_I}\right)^2 dI_1 \dots dI_{N_I} - \sum_{j=0}^{N_H} \left(W_{j1}^O\right)^2 . \int_{\Delta I_1} \dots \int_{\Delta I_{N_I}} dI_1 \dots dI_{N_I}$$

If the number of expansion terms ($\mathbf{N_H}$) increases, the global minima $\overline{E}_{min}$ decreases [6]. More over, if $\int_{\Delta I_1} \dots \int_{\Delta I_{N_I}} dI_1 \dots dI_{N_I} = 1$, $\overline{E}_{min}$ goes to the well known Parceval's equality.

In summarize, the Euclidean distance *(6)* can be reduced by choosing an infinite sum of hidden functions φ_j such that:

_ $\varphi_0\left[S_j^H\right] = 1$

_ $\varphi_j\left[S_j^H\right]$ are orthogonal in the domains ΔS_j induced by $\Delta I_1 \dots \Delta I_{N_I}$

_. $W_{j1}^O = \dfrac{C_j}{\int_{\Delta I_1} \dots \int_{\Delta I_{N_I}} dI_1 \dots dI_{N_I}}$ for $0 \le j \le N_H$.

Legendre polynomials belong to this class of functions and are defined, on the domain $\Delta S_j = [-1,1]$ by: $\varphi_0\left[S_j^H\right] = 1$, $\varphi_1\left[S_j^H\right] = S_j^H$ and $\varphi_{n+1}\left[S_j^H\right] = \dfrac{2.n+1}{n+1} . S_j^H . \varphi_n\left[S_j^H\right] - \dfrac{n}{n+1} . \varphi_{n-1}\left[S_j^H\right]$.

Unfortunately, since the expression of $f\left(I_1 \dots I_{N_I}\right)$ is supposed not known, the C_j terms (equation 8) can't be calculated. More over, it's excluded to consider an infinite layer of hidden units in practice. As the only knowledge of $f\left(I_1 \dots I_{N_I}\right)$ is a set of discrete values coming from the training set, we are going to determine the number and the values of parameters W_{j1}^O by minimizing $\overline{E}$ for discrete variations of $\left(I_1 \dots I_{N_I}\right)$.

3 Initialization of parameters by discrete function representation

3.1) Vectorial formulation of the problem

All values of an input scalar variable I_i can be included into a vector $\vec{I}_i = \left[\dots I_i(\alpha) \dots\right]^T$ with $1 \le \alpha \le m$, where α is an index and m is the number of values. For each value of independent input

scalar variables $\{I_1(\alpha)...I_{N_I}(\alpha)\}$ from the training set, we can associate the corresponding output scalar value $f(\alpha)$. The set of all output values constitutes the vector $\vec{F} = [...f(\alpha)...]^T$ and we say that this vector is a function of $[\vec{I}_1...\vec{I}_{N_I}]$. The basis defined by the independent input vectors, $[\vec{I}_1...\vec{I}_{N_I}]$, defines a N_I dimensional space I, where the vector $\vec{F}$ spans a subspace written F (figure 3).

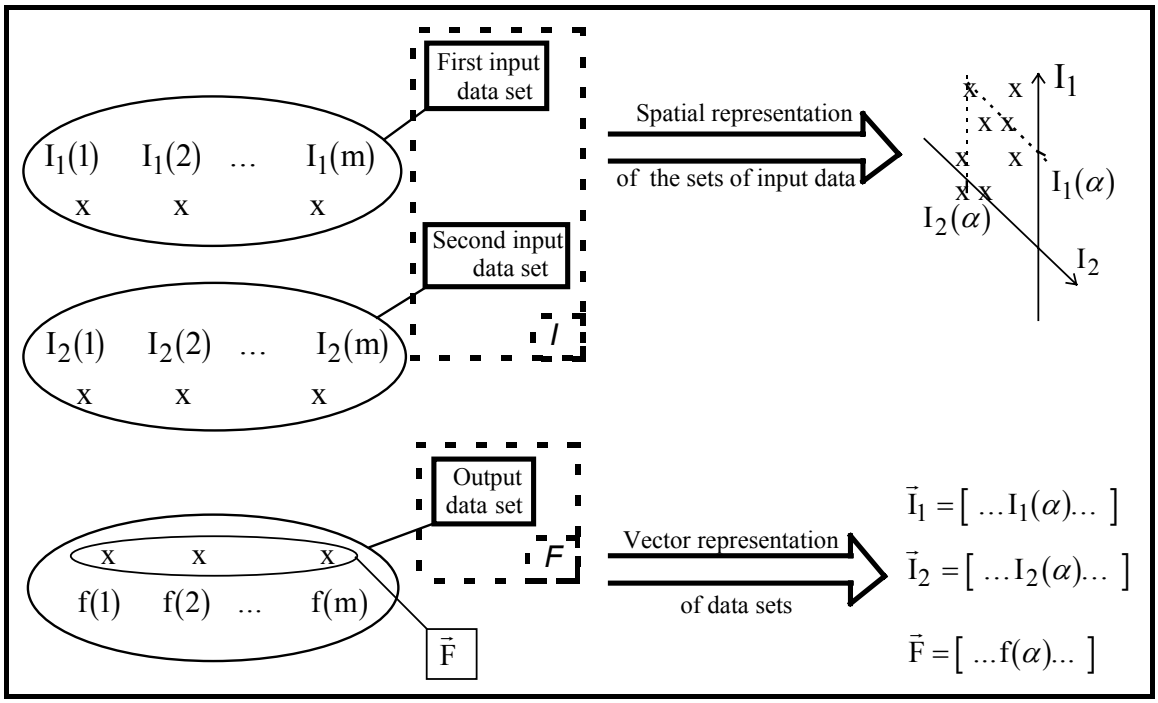

figure 3: Representations of a two inputs function

In a same way, we can associate to the input vectors a vector $\vec{R}$ containing the responses of the neural network when fed with the input values from $[\vec{I}_1...\vec{I}_{N_I}]$.
Then, the initialization of a neural network applied to function approximation can be formulated as:
" Assuming that $\vec{F}$ contains a full representation of the function output set, the problem is to find a new basis, determined by the weights and the bias, in which $\vec{R}$ is similar to $\vec{F}$ on the restricted input domain defined by the basis $[\vec{I}_1...\vec{I}_{N_I}]$ ".

3.2) Reduction of the variation domain of input variables

The number of input units is the number of input variables of the function: N_I. Elementary statistical arrangement, when manipulating input sets of data, is to center them with their mean

value $\bar{I}_i$: $\bar{I}_i = \frac{1}{m}.\sum_{\alpha=1}^{m} I_i(\alpha)$, and to divide them with the squared root of their variance σ_i: $\sigma_i = \sum_{\alpha=1}^{m}\left(I_i(\alpha) - \bar{I}_i\right)^2$ to get a better scattering.

This calculation can be implemented on the first layer by setting $\boxed{W_{ii}^I = \frac{1}{\sqrt{\sigma_i}}}$ and $\boxed{B_i^I = \frac{-\bar{I}_i}{\sqrt{\sigma_i}}}$.

By replacing I_i with $I_i(\alpha)$ in equation (1) and writing it for all α, we find

$$\begin{bmatrix} . \\ . \\ . \\ O_i^I(\alpha) \\ . \\ . \\ . \end{bmatrix} = \begin{bmatrix} . \\ . \\ . \\ B_i^I \\ . \\ . \\ . \end{bmatrix} + W_{ii}^I . \begin{bmatrix} . \\ . \\ . \\ I_i(\alpha) \\ . \\ . \\ . \end{bmatrix} \quad \text{with } 1 \le \alpha \le m \quad \text{for } 1 \le i \le N_I, \text{ then we get } \vec{O}_i^I = \vec{B}_i^I + W_{ii}^I . \vec{I}_i .$$

The first level of processing layer converts the training input sets by data translation into a best data representation (figure 4).

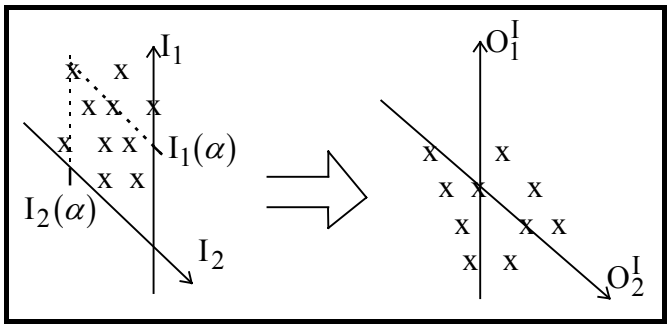

figure 4: Representation of two sets of input variables after processed by the input layer

3.3) <u>Orientation of the spatial representation of input data</u>

3.31) Construction of an orthogonal basis

The set of the input unit responses can form a $(m \times N_I)$ matrix as $O^I = \left[\vec{O}_1^I \ldots \vec{O}_{N_I}^I\right]$. The covariance matrix, $C = O^{I^T}.O^I$, is a positive semi definite matrix and, then, admits non negative real eigenvalues: $\lambda_1^2 \ge \lambda_2^2 \ge \ldots \ge \lambda_{N_I}^2 \ge 0$. We'll write the N_I corresponding orthogonal unit eigenvectors: $\vec{V}_1, \vec{V}_2, \ldots, \vec{V}_{N_I}$. Using these vectors as an orthogonal basis, O^I can be represented as $O^I = \vec{P}_1.\vec{V}_1^T + \vec{P}_2.\vec{V}_2^T + \ldots + \vec{P}_{N_I}.\vec{V}_{N_I}^T$, where $\vec{P}_i = O^I.\vec{V}_i$ is called a component vector.

The subspace spanned by the component vectors (corresponding to non zero magnitude) represents the linear relationships (independent of α) which exist among the inputs contained in O^I.

3.32) Linear dependencies between input and hidden units

Each input $S_j^H(\alpha)$ of hidden units j is a point fixed by N_I coordinates: $O_1^I(\alpha),\ldots,O_{N_I}^I(\alpha)$. The set of values ($1\le\alpha\le m$) is considered as a vector $\vec{S}_j^H$ in the vectorial space $\Re^{N_I}$.

For example, consider $N_I=2$. When presenting $I_1(\alpha)$ and $I_2(\alpha)$, we obtain on the first layer $O_1^I(\alpha)$ and $O_2^I(\alpha)$. Assume that $B_1^H=0$, the summation made by the first hidden unit is given by equation (2): $\begin{bmatrix}.\\.\\.\\S_1^H(\alpha)\\.\\.\\.\end{bmatrix}=W_{11}^H.\begin{bmatrix}.\\.\\.\\O_1^I(\alpha)\\.\\.\\.\end{bmatrix}+W_{21}^H.\begin{bmatrix}.\\.\\.\\O_2^I(\alpha)\\.\\.\\.\end{bmatrix}$ for $1\le\alpha\le m$, then we get

$\vec{S}_1^H=\left[\vec{O}_1^I\ \vec{O}_2^I\right].\vec{W}_1^H$. $\vec{W}_1^H=\left[W_{11}^H\ W_{21}^H\right]^T$ is a projection vector among an axis guided by $\vec{S}_1^H$.

More generally, for a neural network with N_I inputs and N_H hidden units, the internal summation of the hidden unit j $(1\le j\le N_H)$ is: $\vec{S}_j^H=O^I.\vec{W}_j^H$ with $\vec{W}_j^H=\left[W_{1j}^H\ W_{2j}^H\ \ldots\ W_{N_I j}^H\right]^T$.

From a geometric point of view, the hidden layer contains a particular representation of O_1^I that will be written: $O^H=\vec{S}_1^H.\vec{W}_1^{H^T}+\vec{S}_2^H.\vec{W}_2^{H^T}+\ldots+\vec{S}_{N_H}^H.\vec{W}_{N_H}^{H^T}$.

3.33) Application of principal component analysis to hidden initialization

If we consider the particular case where $(N_H\le N_I)$, the initialization of these hidden weights can be found by using the Principal Component Analysis. Consequently, to get the best approximation of the linear dependencies between the inputs, we must minimize the critical error: $J=\left\|O^I-O^H\right\|^2$ *(9)* by choosing $O^H=\sum_{j=1}^{N_H}\vec{P}_j.\vec{V}_j^T$ *(10)*.

By replacing *(10)* in *(9)*, it is easy to find $J=\sum_{j=N_H+1}^{N_I}\lambda_j^2$ for fixed N_H and N_I.

This error will be null if:

_ the number of hidden units is equal to the number of input variables $(N_H = N_I)$

_ the hidden weights are collinear to the eigenvectors $\vec{W}_j^H = K_j . \vec{V}_j$

_ the bias, B_j^H, are null.

The normalization scalars K_j will be calculated such that the domains $\Delta S_i = \left[\mathrm{Min}(S_i), \mathrm{Max}(S_i)\right]$ fit into the domain of the orthogonal functions ([-1,1] for Legendre polynomials).

Now, let us expose another interpretation of this hidden initialization.

When we present input values out of the training set, the neural network makes a generalization. This one will be better if we find a linear combination which separates the most possible the inputs. So, this is equivalent to find the weights W_{ij}^H which express the largest variance on a component $\vec{S}_j^H$. In this case, we remark that the first diagonal element of the diagonal matrix: $\left[\vec{S}_1^H \ldots \vec{S}_{N_H}^H\right]^T . \left[\vec{S}_1^H \ldots \vec{S}_{N_H}^H\right]$ is maximum. We can develop each $\vec{S}_j^H$ as:

$$\left[\vec{S}_1^H \ldots \vec{S}_{N_H}^H\right]^T . \left[\vec{S}_1^H \ldots \vec{S}_{N_H}^H\right] = \left[\ldots \vec{W}_j^{H^T} \ldots\right] . O^{I^T} . O^I . \left[\ldots \vec{W}_j^H \ldots\right].$$

As $O^{I^T} . O^I$ is a symmetric matrix, it admits the following decomposition [10]:

$O^{I^T} . O^I = V . \Delta . V^T$ *(11)*, where Δ is a diagonal matrix containing the eigenvalues of $C = O^{I^T} . O^I$,

$$\Delta = \begin{bmatrix} . & 0 & 0 & 0 & 0 \\ 0 & . & 0 & 0 & 0 \\ 0 & 0 & \lambda_i & 0 & 0 \\ 0 & 0 & 0 & . & 0 \\ 0 & 0 & 0 & 0 & . \end{bmatrix}$$ and V is a matrix containing the corresponding eigenvectors of C, $V = \left[\ldots, \vec{V}_i, \ldots\right]$. Then, if we set $\vec{W}_j^H = \vec{V}_j$, $\left[\vec{S}_1 \ldots \vec{S}_{N_H}\right]^T . \left[\vec{S}_1 \ldots \vec{S}_{N_H}\right] = \Delta$. Following the previous remark, Δ must contain the eigenvalues in a decreasing order.

From relation *(11)*, we can show that component vectors are orthogonal: $\vec{S}_i^{H^T} . \vec{S}_j^H = 0$ (for $i \neq j$), $\left\|\vec{S}_i^H\right\|^2 = \lambda_i^2$.

3.34) Dimensionally expansion of the input space

Following this initialization, the data are projected from their original N_I dimensional space I, spanned by $\left\{\vec{I}_1 \ldots \vec{I}_{N_I}\right\} \in \Re^{N_I}$, onto the N_H dimensional space S, spanned by $\left\{\vec{S}_1 \ldots \vec{S}_{N_H}\right\} \in \Re^{N_H}$.

This particular orthogonal representation, known as principal component analysis, is characterized:

_ by $\vec{S}_1$, which orients the data representation among the input data vectors holding the maximum eigenvalue λ_1 of C .

_ by the number of hidden units N_H, which has been supposed to be $1 \le N_H \le N_I$.

It has been demonstrated by recent rigorous results that a minimal hidden layer conditions the best generalization properties [7]. For specific applications, it may be interesting to construct a large hidden layer. Firstly, it improves the accuracy of the approximation ($\overline{E}_{min}$ decreases see section 22). Secondly, it increases the robustness (of hardware implementation) in the sense that if one connection is cut ($W_{ij}^a = 0$), the neural network still gives a good approximation.

> For this class of neural networks ($N_H > N_I$), our initialization consists to recalculate the covariance matrix without including the input vector which has given the largest eigenvalue and, so, has given the orientation of the previous representation.

The component vectors of this reduced covariance matrix spans a new ($N_I - 1$) dimensional space in which nevertheless the inputs can be represented.

By implanting at least two representations in the hidden layer, we have realized a dimensionally expansion of the input space. So, we can find no more than N_I representations and, then, the maximum number of hidden units (possible for the proposed initialization) is: $N_I + (N_I - 1) + \ldots + 1$

For higher number of hidden units, we calculate the biases B_i^H ($i > N_I + (N_I - 1) + \ldots + 1$) such that the domains ΔS_i^H don't overlap.

3.4) <u>Output initialization</u>

3.41) Data transformation

Each values $S_j^H(\alpha)$ from $\vec{S}_j^H$ is transformed by the functions $\varphi_j\left[S_j^H(\alpha)\right]$. And, all output values of hidden units are ranged in $\vec{\varphi}_j$: $\vec{\varphi}_j = \left[\ldots \varphi_j(\alpha) \ldots\right]^T$ for $(1 \le j \le N_H)$.

Consider a linear combination of the N_H vectors $\vec{\varphi}_j$,this is a subspace of $\Re^{N_H}$ called R. This subset is spanned by $\vec{R}$ which contains the responses of the neural network:

$$\begin{bmatrix} . \\ . \\ . \\ R(\alpha) \\ . \\ . \\ . \end{bmatrix} = W_{01}^O . \begin{bmatrix} . \\ . \\ . \\ \varphi_0(\alpha) \\ . \\ . \\ . \end{bmatrix} + W_{N_H 1}^O . \begin{bmatrix} . \\ . \\ . \\ \varphi_{N_H}(\alpha) \\ . \\ . \\ . \end{bmatrix} \qquad 1 \le \alpha \le m, \text{ then } \vec{R} = W_{01}^O . \vec{\varphi}_0 + \ldots + W_{N_H 1}^O . \vec{\varphi}_{N_H}$$

(12)

<u>Remark</u>: If $\varphi_0(\alpha) = 1$, $W_{01}^O = B_1^O$

3.42) Output initialization with the pseudo inverse

Equation *(12)* can be rewritten as $\vec{R} = \left[\vec{\varphi}_0 \ \ldots \ \vec{\varphi}_{N_H}\right] . \vec{W}_1^O = \Phi . \vec{W}_1^O$ *(13)*

where $\vec{W}_1^O = \left[W_{01}^O, \ldots, W_{N_H 1}^O\right]^T$ contains the output parameters and Φ is a $(m \times N_H)$ matrix. The problem is then to find $\vec{W}_1^O$ to have $\vec{F} = \Phi \,.\, \vec{W}_1^O$ *(14)*. This problem is linear in the vector $\vec{W}_1^O$, although non linear with respect to the input $\left[\vec{I}_1 \ldots \vec{I}_{N_H}\right]$.

The general solution of *(14)* is $\boxed{\vec{W}_1^O = \Phi^+ . \vec{F}}$

where the pseudo inverse of Φ is called Φ^+ and is given by $\Phi^+ = \Phi^T\left[\Phi.\Phi^T\right]^{-1}$. The computation of the pseudo inverse of a general matrix based on the Greville's theorem can be viewed in [5]. Assume that rank$[\Phi]$ = r, the table 1 gives the meaning of Φ^+.

r=m			r ≠ m
m= N_H	m < N_H	m> N_H	∀ m
Φ^+ is the full inverse of the full rank and squared matrix Φ	*(14)* have many solutions but $\Phi^+.\vec{F}$ is the one which minimizes the norm of $\vec{W}_1^O$	*(14)* have not exact solutions but $\Phi^+.\vec{F}$ is the one which minimizes the Euclidean distance: $\left(\vec{F}-\vec{R}\right)^T.\left(\vec{F}-\vec{R}\right)$ and the norm of $\vec{W}_1^O$	

Table 1

In practice, we have usually more patterns than hidden units: $m \geq N_H$. Suppose any function $\varphi_j[\ldots]$, then, linear dependencies between $\vec{\varphi}_0, \ldots, \vec{\varphi}_{N_H}$ can appear, and, $\mathbf{N_H > r}$. So, another way to draw near to $\bar{E}$ is to increase the linearly independent subset of ϕ by increasing the number of hidden units until that r = m. If φ_j**[…]** are chosen non linear and different each other (It is the case of Legendre polynomials), the linearly independent subset is the all set ϕ. Then, N_H = r is guaranteed, and, so, less added hidden units are required to reach m = r.

4) <u>Learning algorithm</u>

Since adaptation to new dynamics or new functionnement domain is required in practice, we develop now a learning algorithm based on the generalized delta rule [11]. The well known backpropagation algorithm performs a steepest descent on the surface $\frac{1}{2}E(\alpha)^2$ by adjusting the weights as New W_{ij}^a = Old W_{ij}^a + $\eta.\Delta W_{ij}^a$, where η is the learning rate and $\Delta W_{ij}^a = \frac{\partial[\frac{1}{2}E(\alpha)^2]}{\partial W_{ij}^a}$.

For the output weights, we have then: $\Delta W_{i1}^O = \frac{\partial[\frac{1}{2}E(\alpha)^2]}{\partial S_1^O} \,.\, \frac{\partial S_1^O}{\partial W_{i1}^O} = -E(\alpha) \,.\, O_1^H$.

For other weights, the chain rule is used to write:

$$\Delta W_{ij}^{H} = \frac{\partial[\frac{1}{2}E(\alpha)^2]}{\partial S_j^H} \cdot \frac{\partial S_j^H}{\partial W_{ij}^H} = \sum_{j=1}^{N_H}\left[\frac{\partial[\frac{1}{2}E(\alpha)^2]}{\partial S_1^O} \cdot \frac{\partial S_1^O}{\partial O_j^H} \cdot \frac{\partial O_j^H}{\partial S_j^H}\right] \cdot \frac{\partial S_j^H}{\partial W_{ij}^H} \cdot$$

$$\Delta W_{ij}^{H} = \sum_{j=1}^{N_H}\left[\frac{\partial[\frac{1}{2}E(\alpha)^2]}{\partial S_1^O} \cdot W_{j1}^O \cdot \varphi_j'\left(S_j^H\right)\right] \cdot O_i^I$$

$\varphi_j'\left(S_j^H\right)$ are then the derivative of the Legendre polynomials.

5) Example and simulation

The proposed example is the approximation of the function $f = I_1 * I_2$ with a 2 hidden units neural network. We have represented, figure 5, the m=18 sampled input output data and, figure 6, the same points but after processed by the hidden orthogonalization .

The sum of all Euclidean distances: $E\left(I_1 \ldots I_{N_i}\right)^2 = \left[f\left(I_1 \ldots I_{N_i}\right) - R\left(I_1 \ldots I_{N_i}\right)\right]^2$ measures the dissimilarity between f and the neural network approximation on the restricted domain $\left[\vec{I}_1 \ldots \vec{I}_{N_I}\right]$.

Table 2 compares a unity hidden initialization and the hidden orthogonalization with the Euclidean distance $E(18)$ for different hidden functions. No great differences appear when choosing linear hidden functions or sigmoid hidden functions (It is certainly due to the small number of patterns).

Keeping the calculated initializations, we present, now, data outside the training set. A global error performance can be measured with the following error function:

$\bar{E}(n) = \sum_{\alpha=1}^{n} E(\alpha)^2 = \sum_{\alpha=1}^{m} E(\alpha)^2 + \sum_{\alpha=m+1}^{n} E(\alpha)^2$. The first term includes all patterns used to initialize the network and the second measures the generalization ability. We calculate this distance for n=38.

Table 3 shows the good generalization behavior of Legendre polynomials. The corresponding initialization is given by: $\vec{W}_{11}^I = 1.01,\ \vec{W}_{22}^I = 0.65,\ B_1^I = -0.02,\ B_2^I = 0.02,$

$\vec{W}_1^H = [-0.31\ \ 0.2]^T,\ \vec{W}_2^H = [-0.2\ \ 0.31]^T,\ \theta = [-1.61\ -0.12\ -3.38]^T.$

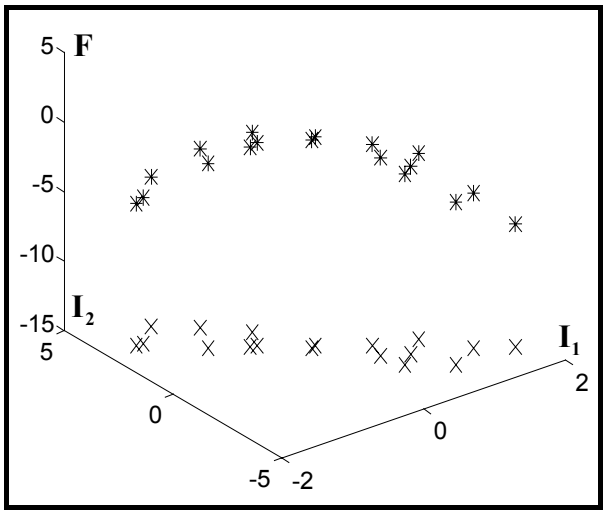

figure 5: Input output data

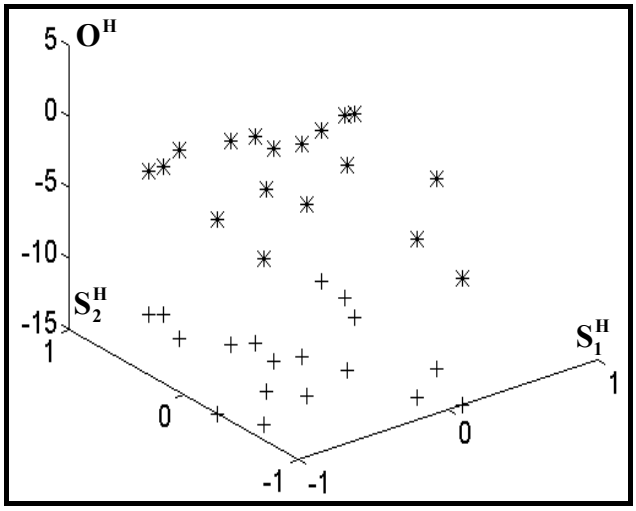

figure 6: Representation of these points in the hidden layer

E(18)	linear	sigmoid	Legendre
$\vec{W}_j^H = \vec{1}$	59.99	60.9	11.08
$\vec{W}_j^H = \vec{V}_j$	59.99	59.98	2.87

table 2: Effect of the hidden orthogonalization

E(59)	linear	sigmoid	Legendre
$\vec{W}_j^H = \vec{1}$	609	558	450
$\vec{W}_j^H = \vec{V}_j$	609	548	279

table 3: Generalization capacity of Legendre polynomials

6) Conclusion

We have presented a neural network architecture and the corresponding initialization for approximating relationship between input and output data. Multi output function approximation can be executed by placing orthogonal neural networks in parallel on input variables as many as it is necessary. The initialization applies three treatments to the training set:

_ a statistical arrangement to make appear the maximum of information hold in the input data sets

_ a hyperspace construction such that the projected input data get generalization abilities

_ an interpolation of output data from the training set with the output values from hidden units.

The hidden initialization based on principal component analysis can be easily extended to continuous variation of α in $O_j^I(\alpha)$ by considering the grammian written $G^2 = \int_{\alpha_1}^{\alpha_2} O^{I^T}(\alpha).O^I(\alpha)\ d\alpha$ rather than the covariance matrix [9].

The output initialization depends on the definition of a distance $\overline{E}(\alpha)$ and on a training set. Possible improvements can be achieved by hanging this measurement, for example the Bernoulli error: $\overline{E}(\alpha) = \sum_{\alpha=1}^{n} f(\alpha).\log[R(\alpha)] + [1-f(\alpha)].\log[1-R(\alpha)]$.

The initialization is greatly depending of the training set (like the convergence of a learning algorithm). It will be more precise if the training set is closest to the function dynamics.

Considerations about the hidden layer size have been done among generalization abilities, approximation accuracy and rank calculation. In future, generalization abilities will be study by using the information theory (Quantification of the knowledge holding in the data training) and the selectivity (Quantification of the network storage capacity).

Finally, practical implantation of orthogonal functions must be considered on parallel chips (A.S.I.C) and on pipeline algorithm (applied on Digital Signal Processors).

REFERENCES

[1] Francois B., Borne P., "Design and initialization of a multilayer neural network applied to function approximation", 3rd IMACS Int. Workshop on "Qualitative Reasoning and Decision Technologies", QUARDET'93, 1993, Barcelona (Spain)

[2] Cybenko G., "Approximation by Superpositions of a Sigmoïdal function", Mathematics of Control, Signals and Systems, 1989, 2, p. 306-314

[3] Hornik, K., Stinchcombe, M., and White, H., " Multilayer Neural Feedforward Neural Networks are Universal Approximators", Neural Networks, 1981, vol. 2, p. 359-366

[4] Cotter, E.N., " The Stone-Weierstrass Theorem and its Application to Neural Networks", IEEE Trans. on Neural Networks, vol. 1, No 4, Dec. 1990

[5] Ben-Israel, A., Greville, T.,N.E., "Generalized Inverses: Theory and Applications", 1974, a Wiley-Interscience Publication, John WileySons, NEW YORK, ISBN 0-471-06577-3

[6] Murray R. Spiegel, "Analyse de Fourier", serie Schaum, 1980, France

[7] Saratchandran P., " Effect of hidden layers on generalization properties of feedforward neural networks ", Neural Parallel & Scientific Computations, 1993, vol.1, p. 227-240

[8] Ehud D. Karnin, " A Simple Procedure for Pruning Back-Propagation Trained neural networks", IEEE Trans. on neural networks, vol. 1, No 2, June 90, p. 239-242

[9] Bruce C. Moore, " Principal Component Analysis in Linear Systems: Controllability, Observability and Model Reduction ", IEEE Trans. on Automatic Control, vol. AC-26, No 1, Feb. 81, p.17-32

[10] Kosko B., " Neural Networks and Fuzzy Systems ", Chap. 5, Prentice Hall International, 1992

[11] Rumelhart D.E., HintonG.E., and Williams R.J. , "Parallel distributed processing", Vol.1, M.I.T. Press, Cambridge, Mass. 1988, Chap.8

[12] Chen T., Chen H., "Approximation of Continuous Functionnals by Neural Networks with Application to Dynamic System", IEEE Trans. on Neural Networks, Vol.4, No 6, June 93, p. 910-918