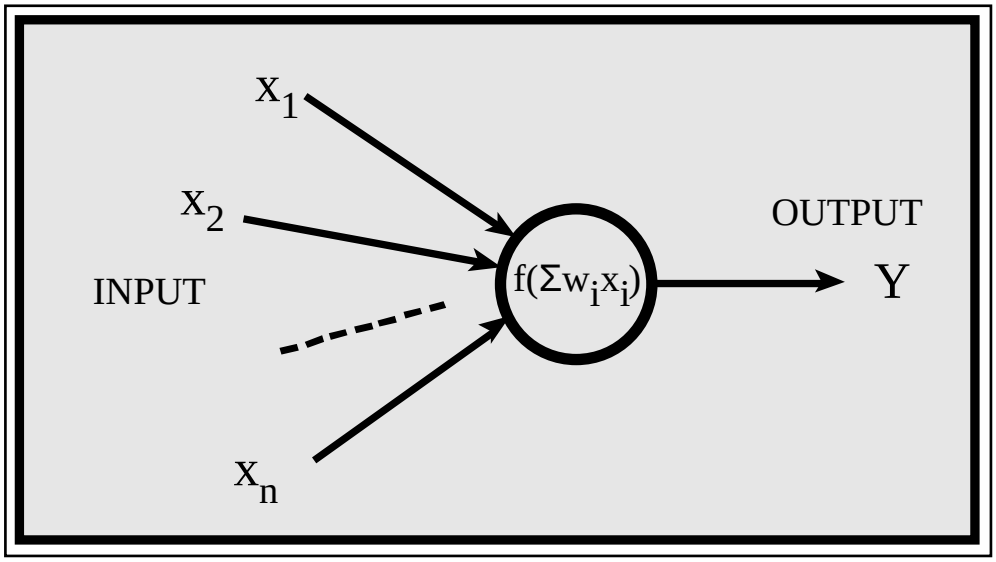
NEURAL NETWORKS



SOME EXAMPLES OF f FUNCTIONS

1. **Threshold sign function**: $f(x) = \begin{cases} 1 & x \geq \mu \\ 0 & x < \mu \end{cases}$

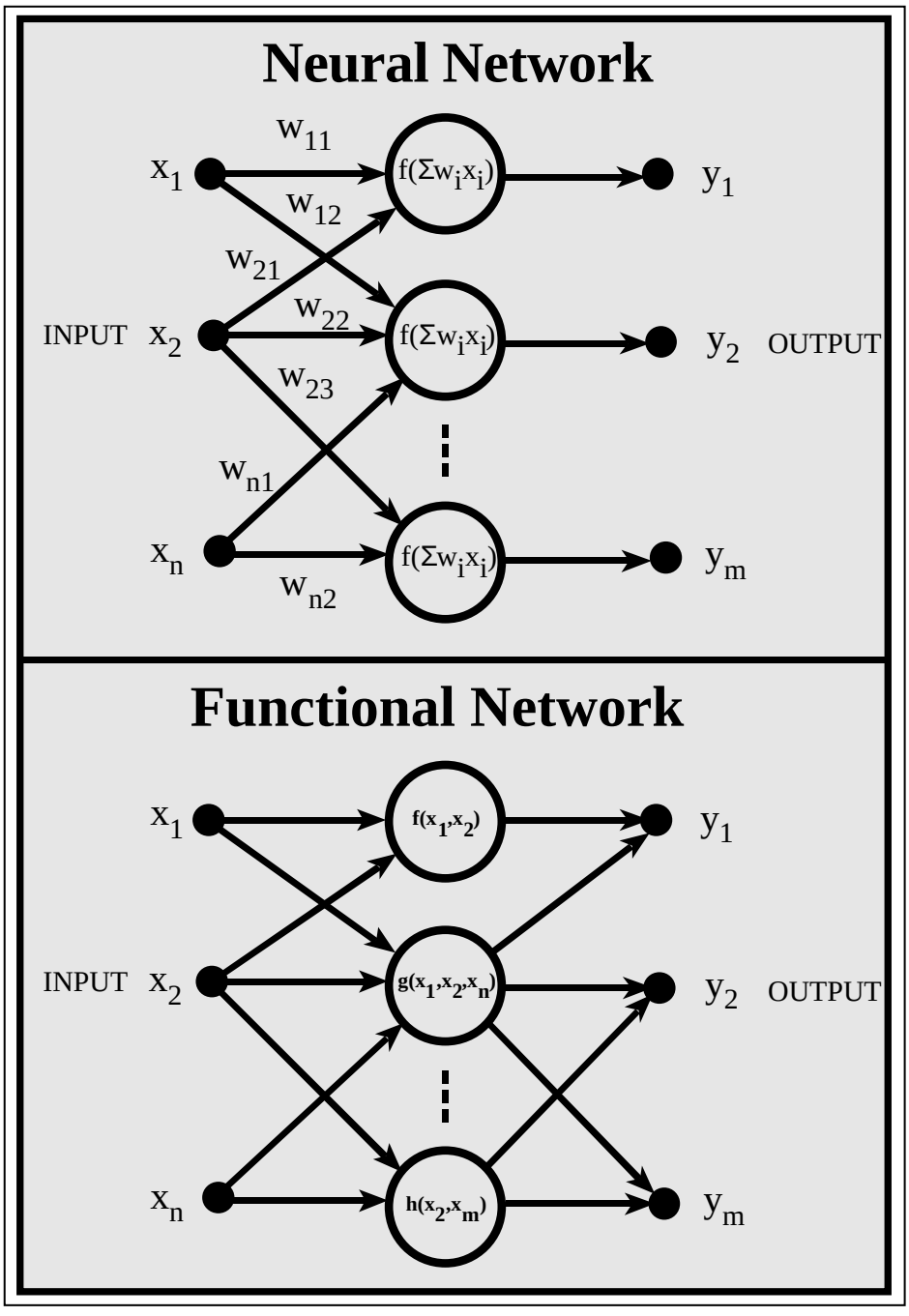
2. **Logistic function**: $f(x) = \dfrac{1}{1+e^{-x}}$

3. **Hyperbolic tangent**: $f(x) = tanh(x)$

4. **Exponential threshold function**:

$$f(x) = \begin{cases} 1 - \dfrac{1}{1+x} & x \geq 0 \\ \dfrac{1}{1-x} - 1 & x < 0 \end{cases}$$

DIFFERENCES BETWEEN NEURAL AND FUNCTIONAL NETWORKS



DIFFERENCES BETWEEN NEURAL AND FUNCTIONAL NETWORKS

The main differences are:

1. In functional networks, neuron functions are **multivariate**, while in neural networks they are **univariate**.
2. In functional networks, the neuron functions can be **different**, while in neural networks they are **identical**.
3. In functional networks **there are no weights**, while in neural networks **there are weights**.
4. In functional networks the **outputs can be coincident**, while in neural networks **outputs cannot be coincident**.
5. In functional networks **neural functions are learned**, while in neural networks **weights are learned**.

FUNCTIONAL UNITS AND CELLS

Definition 1 (Functional Unit) *Let* $X = \{X_i | i \in \mathcal{I}\}$ *be a set of nodes. A functional unit (also called a neuron)* U *over* X*, is a triplet* $< Y, f, Z >$*, where* $Y, Z \subset X$*;* $Y \neq \emptyset, Z \neq \emptyset, Y \cap Z = \emptyset$ *are subsets of* X *and* $f : \mathcal{Y} \rightarrow \mathcal{Z}$ *is a given function. We say that* Y*,* Z *and* f *are the set of input and output nodes, and the processing function of the functional unit* U*, respectively.*

Functional units can be reduced to simpler functional units called *cells*.

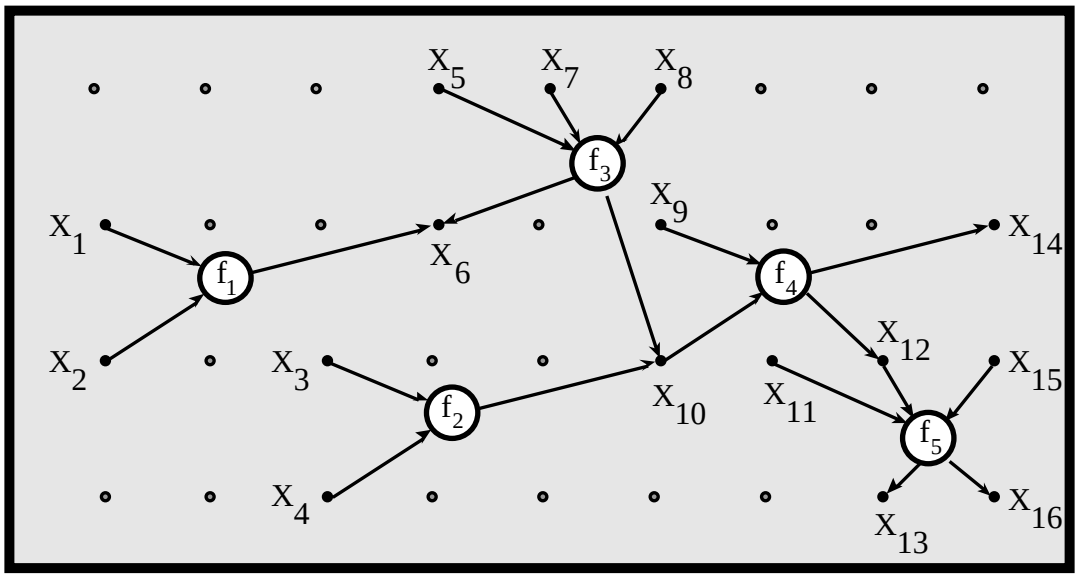
Definition 2 (Functional cell) *A functional unit* $U =< Y, f, Z >$ *such that* $|Z| = 1$*, where* $|Z|$ *is the cardinal of* Z *(single node output), is called a functional cell.*

It is clear that a functional unit $< Y, f, Z >$ can be reduced to the set $\{< Y, f_i, Z_i > | i \in \mathcal{I}_Z\}$ of cells by considering the $|\mathcal{I}_Z|$ components f_i of the corresponding function f.

FORMAL DEFINITION OF A FUNCTIONAL NETWORK

Definition 3 (Functional Network) *A functional network is a pair* $< X, \Gamma >$, *where* X *is a set of nodes and* $\Gamma = \{U_j =< Y_j, f_j, Z_j > | j \in \mathcal{J}\}$ *is a set of functional units over* X, *which satisfies the following condition: Every node* $X_i \in X$ *must be either an input or an output node of at least one functional unit in* Γ, *i.e.,*

$$\forall X_i \in X, \exists j \in \mathcal{J} \text{ such that } X_i \in Y_j \text{ or } X_i \in Z_j$$



An example of functional network showing the set of nodes (printed circuit board) and five functional units (electronic components).

INPUT, OUTPUT AND INTERMEDIATE NODES

Definition 4 (Input Node of a Functional Network) *A node is said to be an input node of a functional network* $< X, \Gamma >$*, if it is the input node of at least one functional unit in* Γ *and is not the output of any functional unit in* Γ*.*

Definition 5 (Output Node of a Functional Network) *A node is said to be an output node of a functional network* $< X, \Gamma >$*, if it is the output node of at least one functional unit in* Γ *and is not the input of any functional unit in* Γ*.*

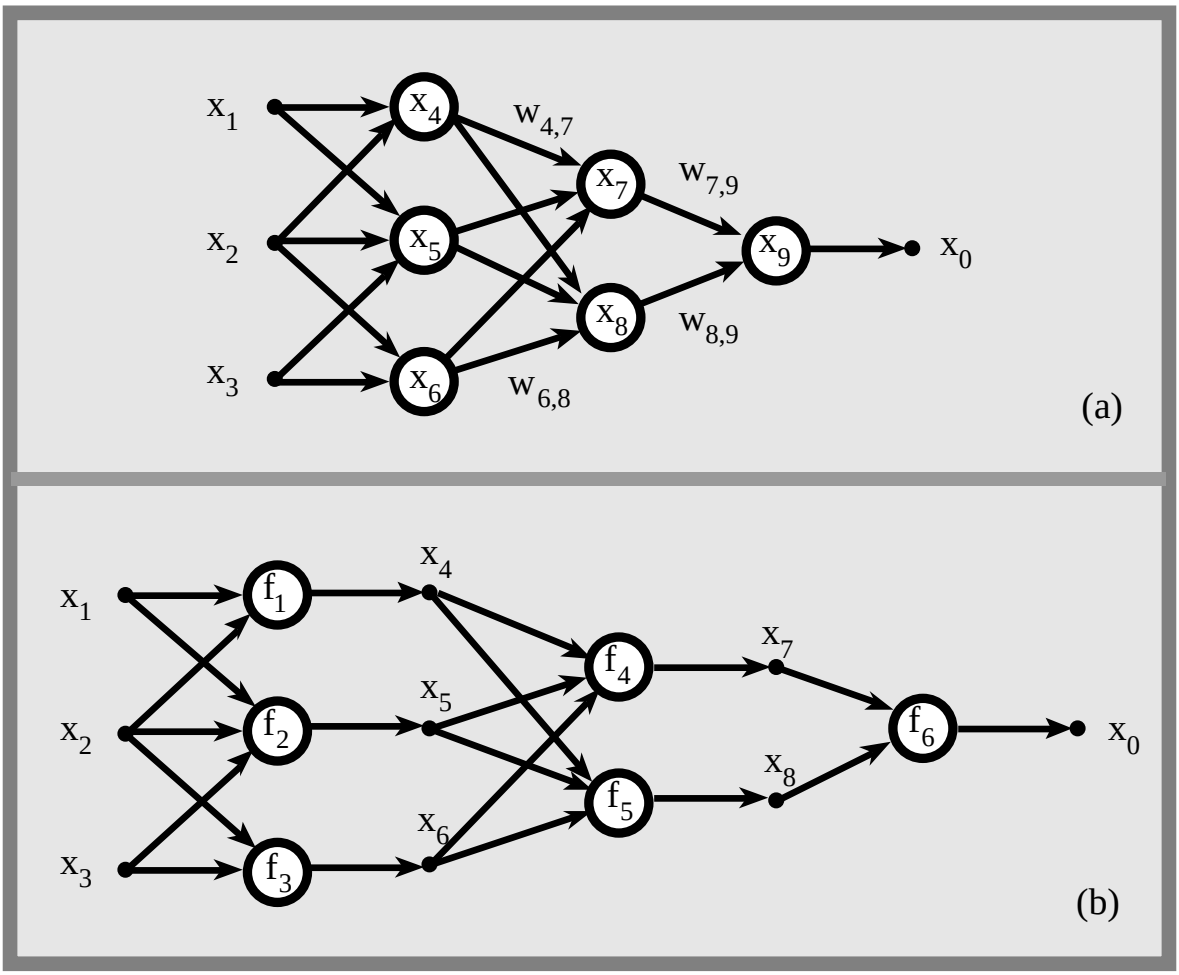
Definition 6 (Intermediate Node of a Functional Network) *A node is said to be an intermediate node of a functional network* $< X, \Gamma >$*, if it is the input node of at least one functional unit in* Γ *and, at the same time, is the output node of at least one functional unit in* Γ*.*

ELEMENTS OF A FUNCTIONAL NETWORK

1. **Several layers of storing units**.
   (a) *A layer of input units*. This first layer contains the input information.
   (b) *A set of intermediate layers of storing units*. They are not neurons but units storing intermediate information. This set is optional and allows connecting more than one neuron output to the same unit.
   (c) *A layer of output units*. This last layer contains the output information.
2. **One or more layers of neurons or computing units**. A neuron is a computing unit which evaluates a set of input values and returns a set of output values to the next layer of units.
3. **A set of directed links**. They connect the input or intermediate layers to neurons, and neurons to intermediate or output units.

NEURAL NETWORK AND EQUIVALENT FUNCTIONAL NETWORK

Neural Network



Equivalent Functional Network

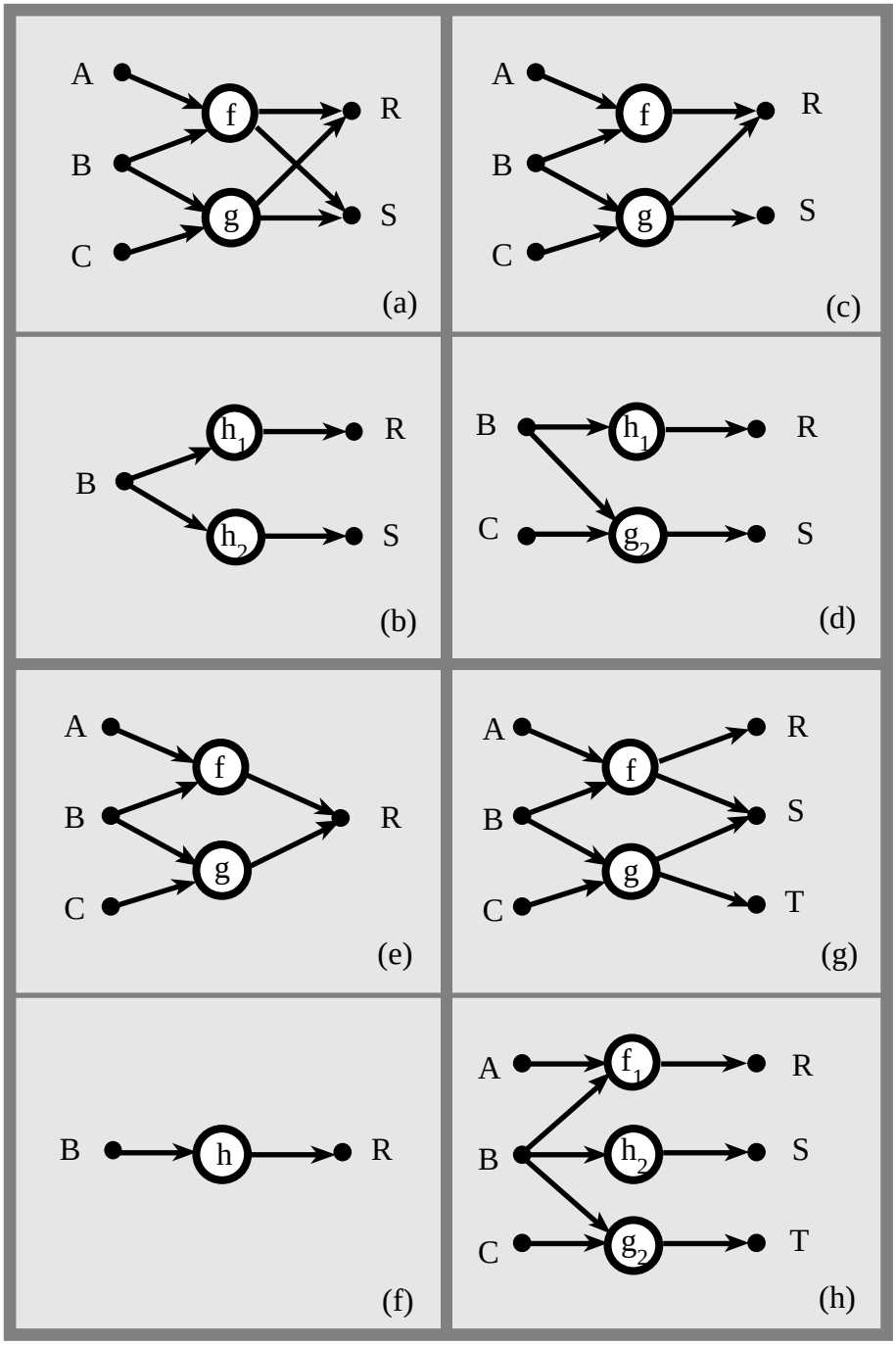
SIMPLIFYING FUNCTIONAL NETWORKS

Given a functional network, an interesting problem consists of determining whether or not there exists another functional network giving the same output for any given input. This leads to the concept of equivalent functional networks.

Definition 7 (Equivalent networks). *Two functional networks are said to be equivalent if they have the same input and output layers and they give the same output for any given input.*

The practical importance of the concept of equivalent functional networks is that we can define equivalent classes of functional networks, that is, sets of equivalent functional networks, and then choose the simplest in each class to be used in applications.

SIMPLIFYING ONE-LAYER FUNCTIONAL NETWORKS

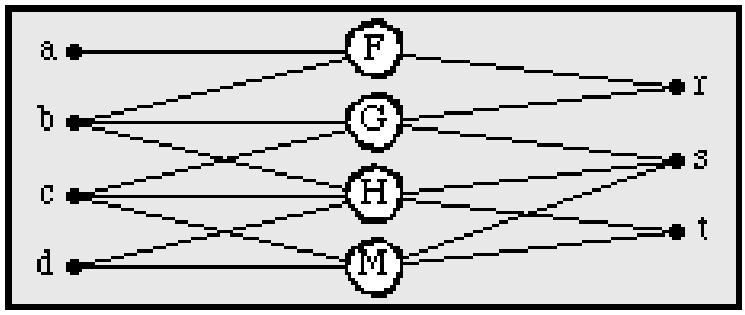


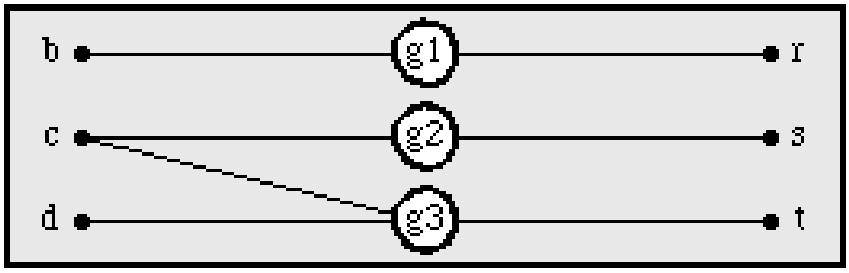
SIMPLIFYING ONE-LAYER FUNCTIONAL NETWORKS (ALGORITHM)

Input: A functional network $N =< X, \Gamma >$.

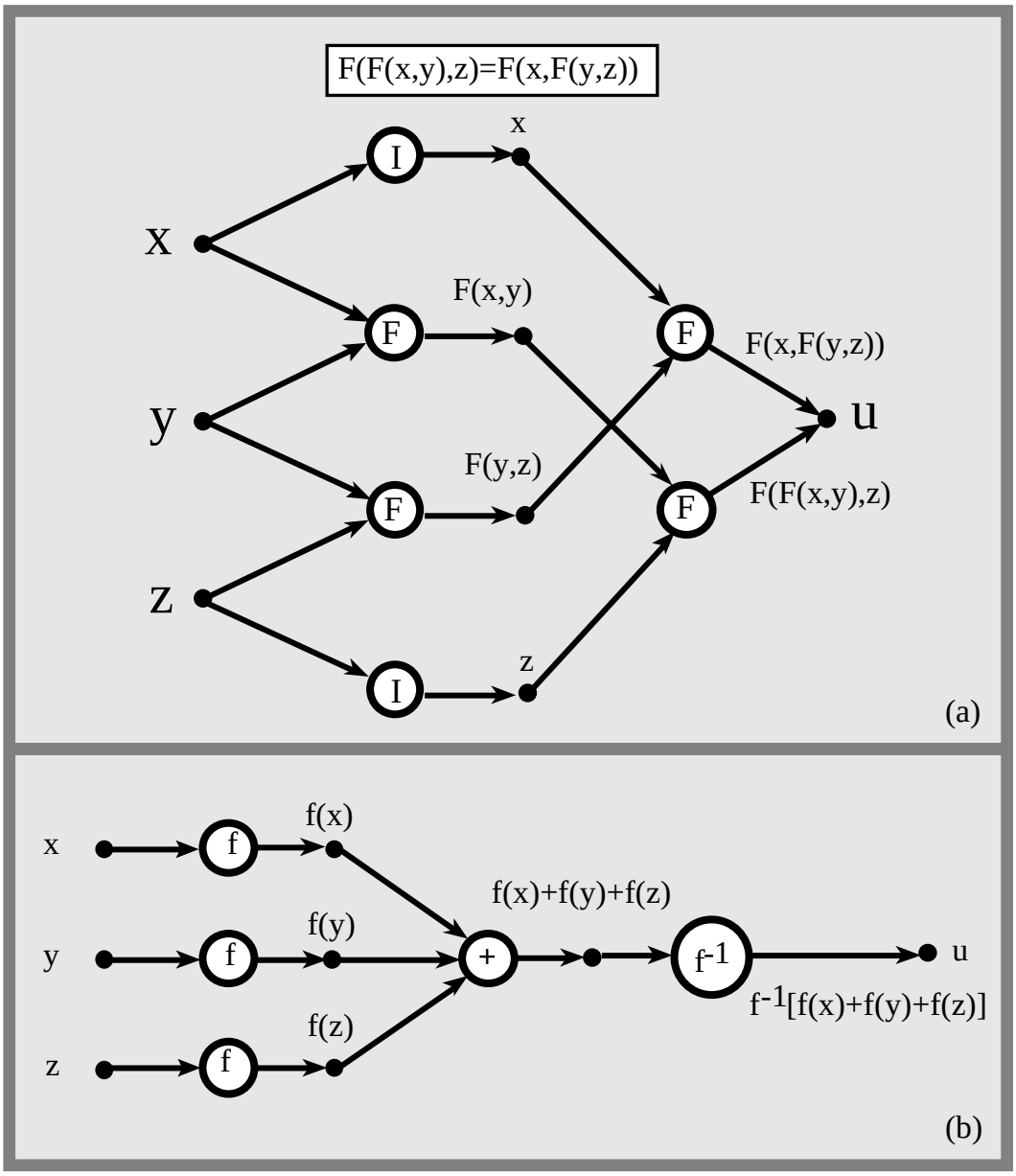
Output: The simplified network.

- **Step 1**: For each output node Z_i in N connected to more than one neuron do:
  - **Step 1.1**: Obtain the intersection set Y_i of all input nodes connected to Z_i.
  - **Step 1.2**: Replace all implied cells by the single cell $< Y_i, h_i, Z_i >$.
- **Step 2**: Repeat Step 1 until no change.





THE ASSOCIATIVITY FUNCTIONAL NETWORK



THE ASSOCIATIVITY FUNCTIONAL EQUATION

Theorem 1 *The general solution continuous and invertible in both variables on a real rectangle of the functional equation*

$$F(F(x,y),z) = F(x,F(y,z)), \tag{1}$$

is

$$F(x,y) = f^{-1}[f(x)+f(y)], \tag{2}$$

where f is an arbitrary continuous and strictly monotonic function, which can be replaced only by $cf(x)$, where c is an arbitrary constant. ■

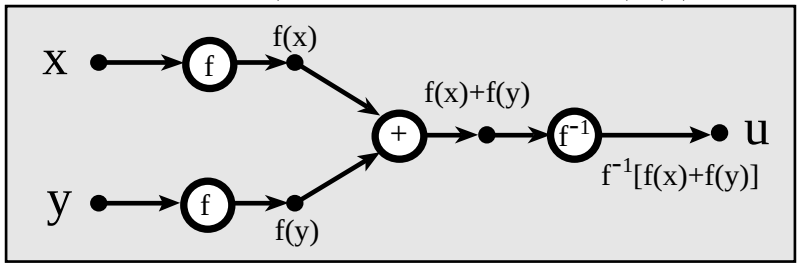
Replacing (2) in (1), we can see that the two sides of (1) can be written as

$$f^{-1}[f(x)+f(y)+f(z)], \tag{3}$$

and then, the functional network (a) in the Figure is equivalent to the functional network (b), where only a one-argument function f need to be learned.

THE ASSOCIATIVITY FUNCTIONAL EQUATION

The Figure shows a network able to reproduce any associative operation (see Equation (2)).



Three conclusions can be derived:

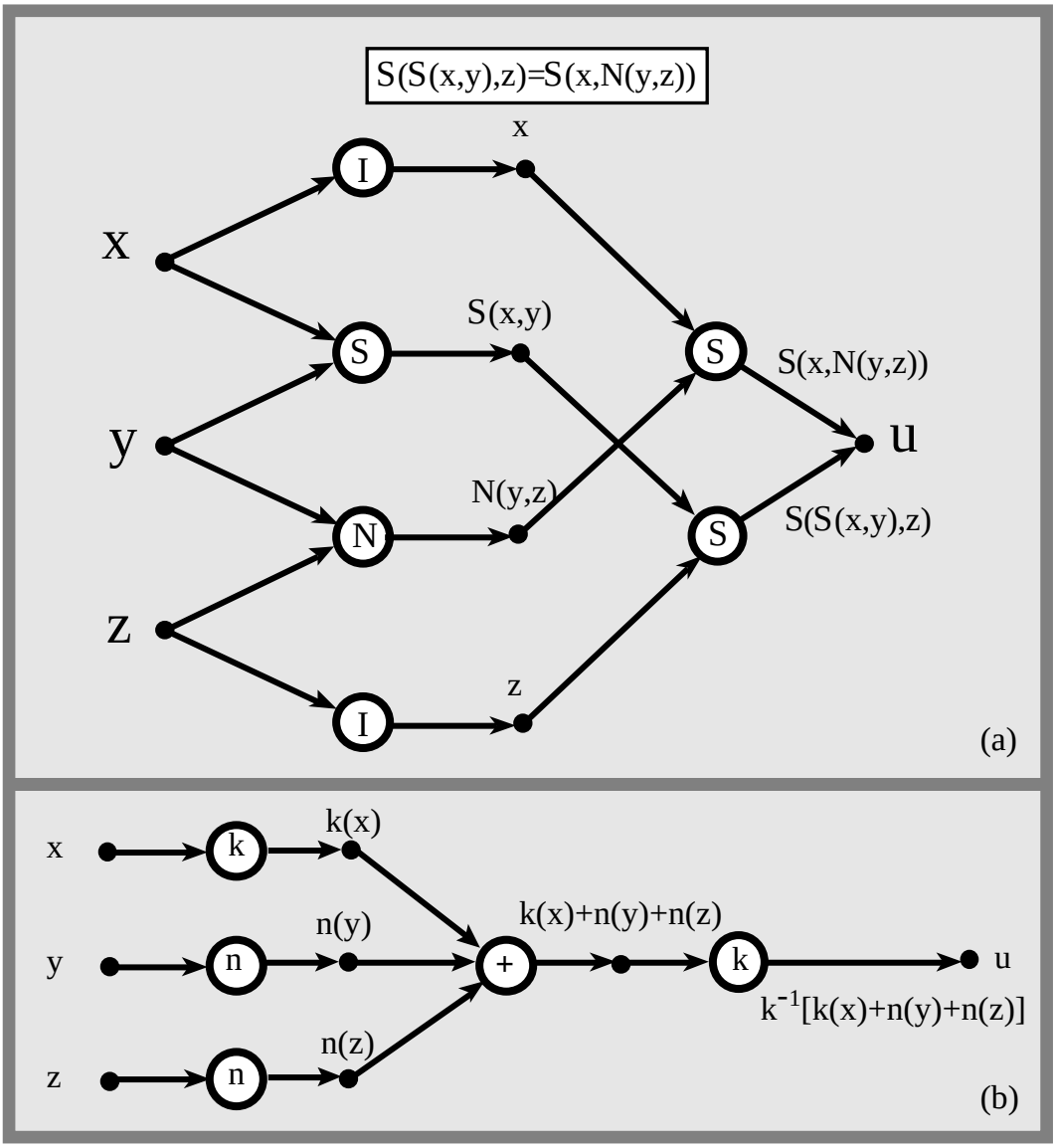
1. No other functional forms for F satisfy equation (1). So, no other F neurons can be used.
2. The functional structure of the solution is (2), where f is determined up to a constant.
3. The functional equation (1) reduces the initial degrees of freedom of $F(.,.)$ from a two-argument to a single-argument function f.

Finally, from (2) we get

$$z = F(x, y) \Leftrightarrow f(z) = f(x) + f(y), \qquad (4)$$

an interesting relation to be exploited for learning $f(x)$.

THE TRANSFORMATION FUNCTIONAL NETWORK



THE TRANSFORMATION FUNCTIONAL EQUATION

Theorem 2 (Transformation equation)
The general solution continuous and invertible in both variables on a real rectangle of the functional equation

$$S[S(x,y),z] = S[x,N(y,z)], \tag{5}$$

is

$$\begin{aligned} S(x,y) &= k^{-1}[k(x)+n(y)]; \\ N(x,y) &= n^{-1}[n(x)+n(y)], \end{aligned} \tag{6}$$

where k and n are arbitrary continuous and strictly monotonic functions, which can be replaced only by

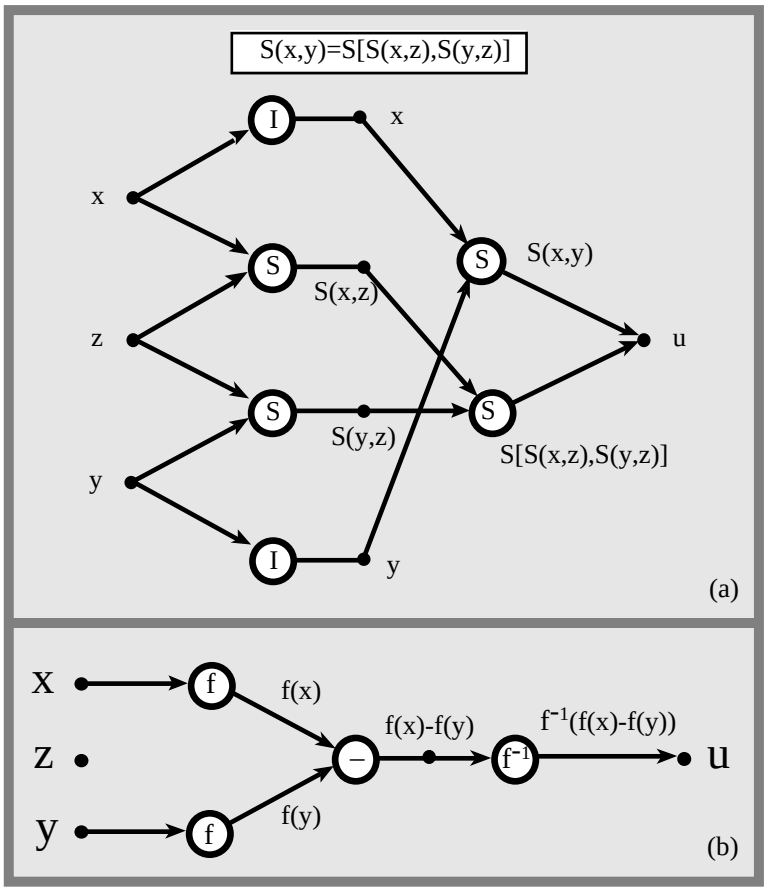
$$\begin{aligned} k^*(x) &= k[(x-a)/c] \\ n^*(x) &= cn(x), \end{aligned}$$

where a and c are arbitrary constants.
The two sides of (5) can be written as

$$k^{-1}[k(x)+n(y)+n(z)], \tag{7}$$

Expression (7) shows that the functional networks in Figures (a) and (b) are equivalent. ■

THE TRANSITIVITY FUNCTIONAL NETWORK



Theorem 3 *The general continuous and invertible solution of the functional equation*

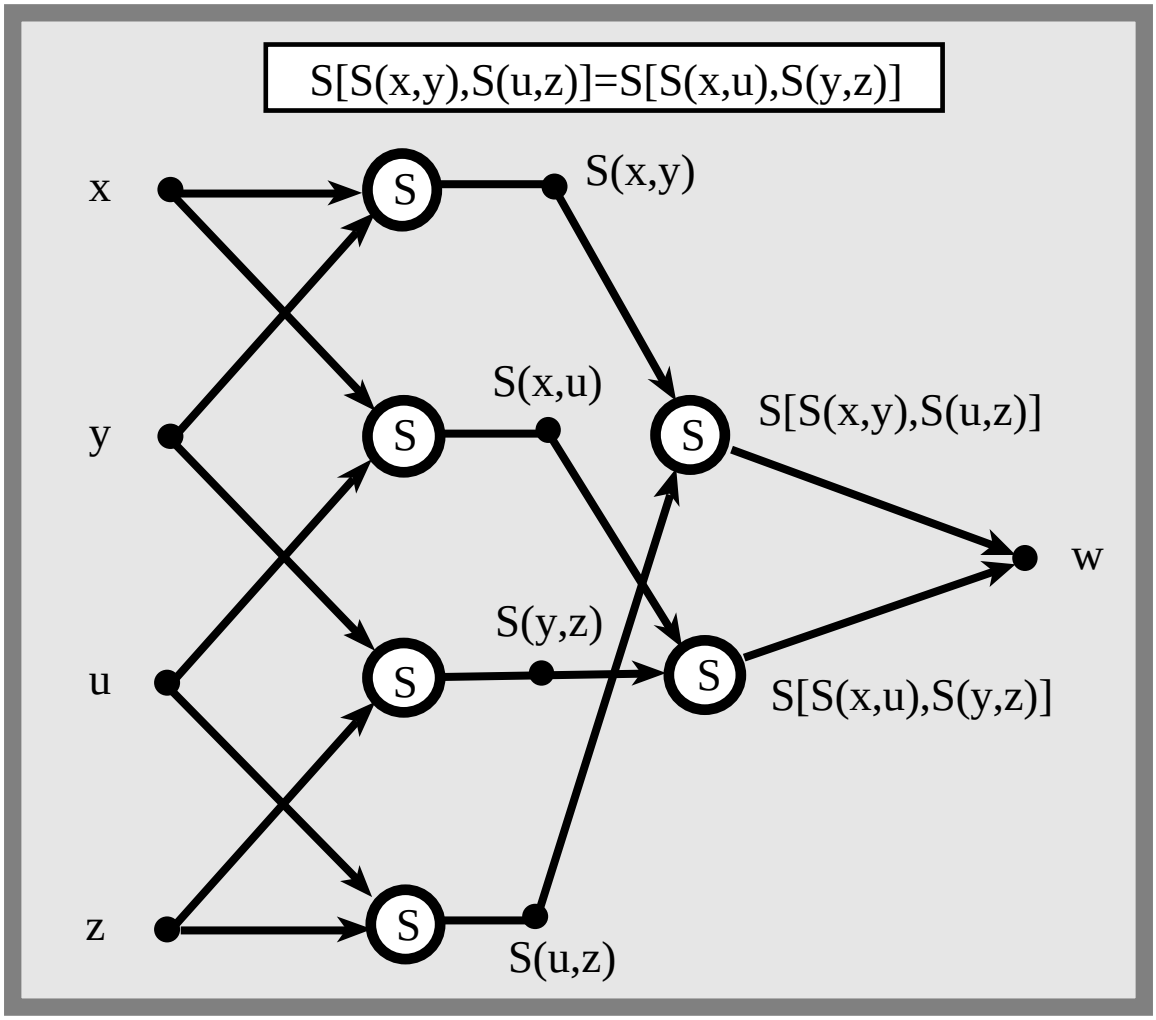
$$S(x,y) = S[S(x,z), S(y,z)], \tag{8}$$

is

$$S(x,y) = f^{-1}[f(x) - f(y)] \tag{9}$$

where f is an arbitrary continuous and strictly monotonic function which can be replaced only by $g(x) = cf(x)$, with c arbitrary. ■

THE BISYMMETRY FUNCTIONAL NETWORK



Theorem 4 *The general solution continuous and invertible in both variables on a real rectangle of the functional equation*

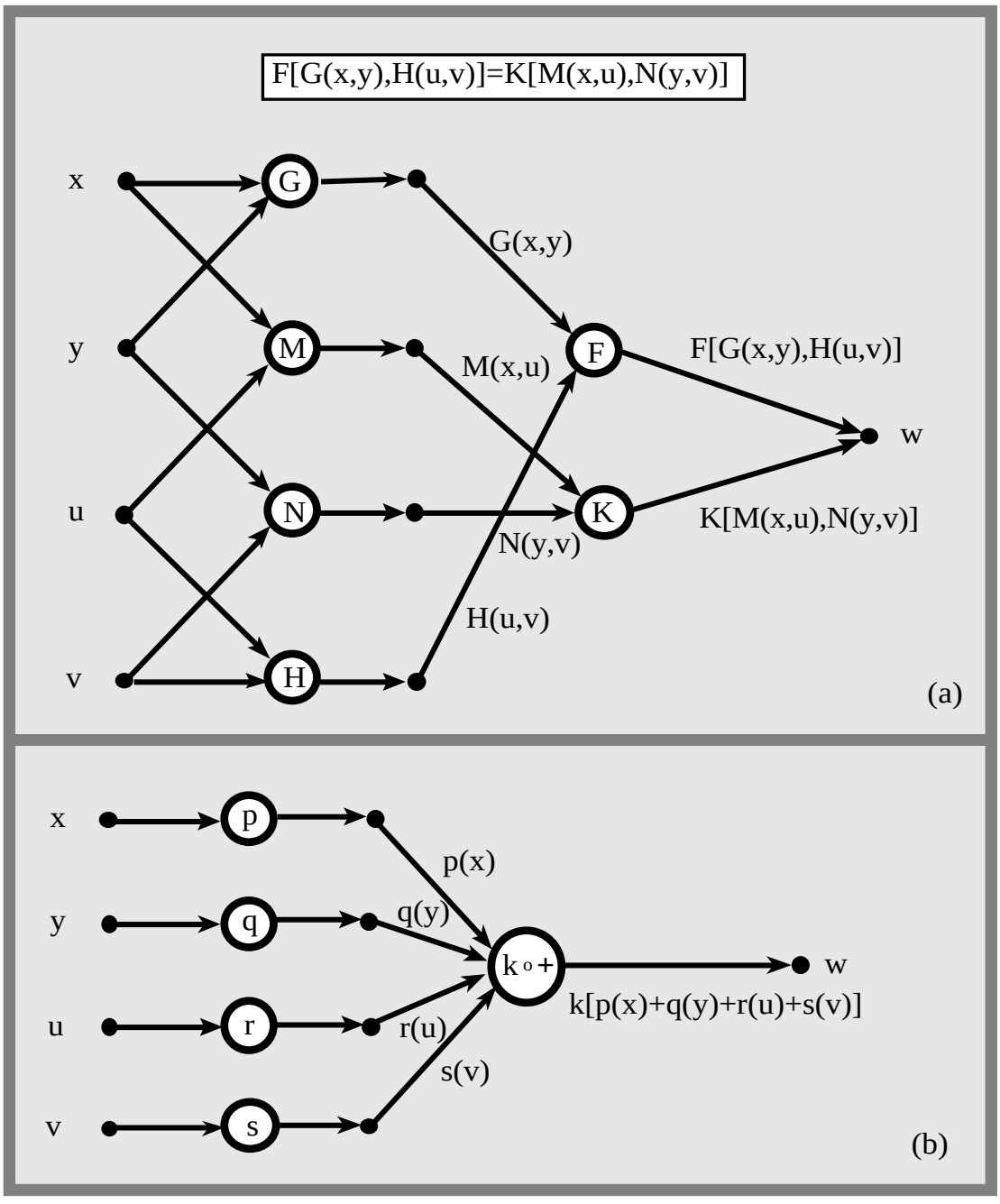
$$S[S(x,y),S(u,z)] = S[S(x,u),S(y,z)]$$

is

$$S(x,y) = g^{-1}[Ag(x) + Bg(y) + C],$$

where g is an arbitrary continuous and strictly monotonic function that can be replaced only by $g_1(x) = cg(x) + d$, where c and d and A, B and C are arbitrary constants. ■

THE GENERALIZED BISYMMETRY FUNCTIONAL NETWORK



THE GENERALIZED BISYMMETRY FUNCTIONAL EQUATION

Theorem 5 *The general solution continuous on a real rectangle of the functional equation*

$$F[G(x,y),H(u,v)] = K[M(x,u),N(y,v)] \qquad (10)$$

with G invertible in both variables, F and M invertible in the first variable for a fixed value of the second variable and H, K and N invertible in the second variable for a fixed value of the first, is

$$\begin{array}{ll} F(x,y) = k[f(x)+g(y)], & G(x,y) = f^{-1}[p(x)+q(y)], \\ K(x,y) = k[l(x)+m(y)], & H(x,y) = g^{-1}[r(x)+s(y)], \\ M(x,y) = l^{-1}[p(x)+r(y)], & N(x,y) = m^{-1}[q(x)+s(y)], \end{array}$$

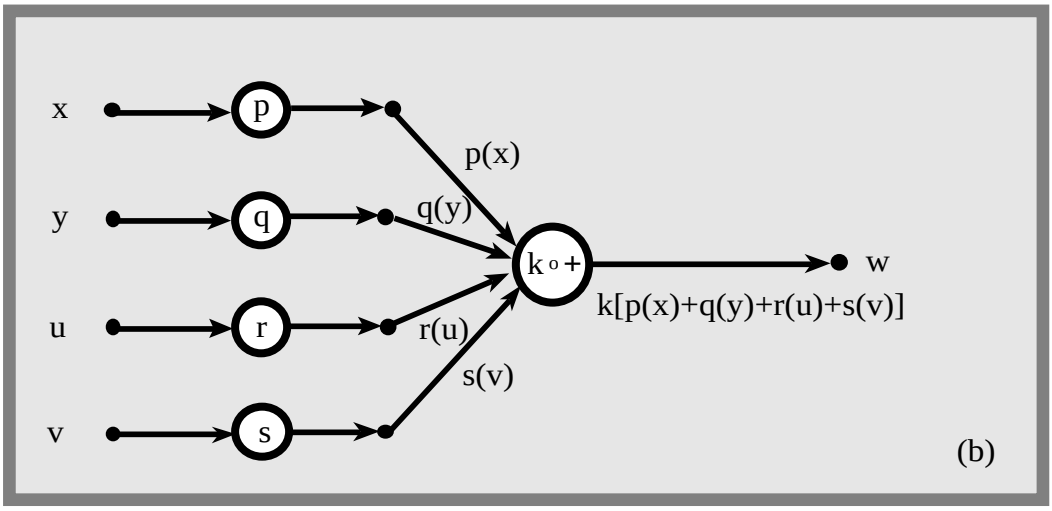
where f, g, k, l, m, p, q and s are arbitrary continuous and strictly monotonic functions and r is an arbitrary continuous function. The lower case functions are not uniquely determined.
The two sides of (10) can be written as

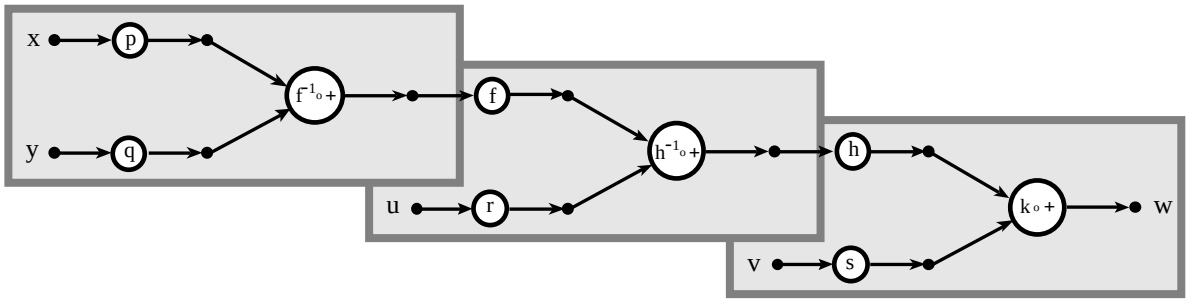
$$k[p(x)+q(y)+r(u)+s(v)], \qquad (11)$$

which proves the equivalence of the two functional networks in the previous Figure.

THE GENERALIZED BISYMMETRY FUNCTIONAL NETWORK

The Figure below shows how the network (b) can be replaced by three simpler neurons.





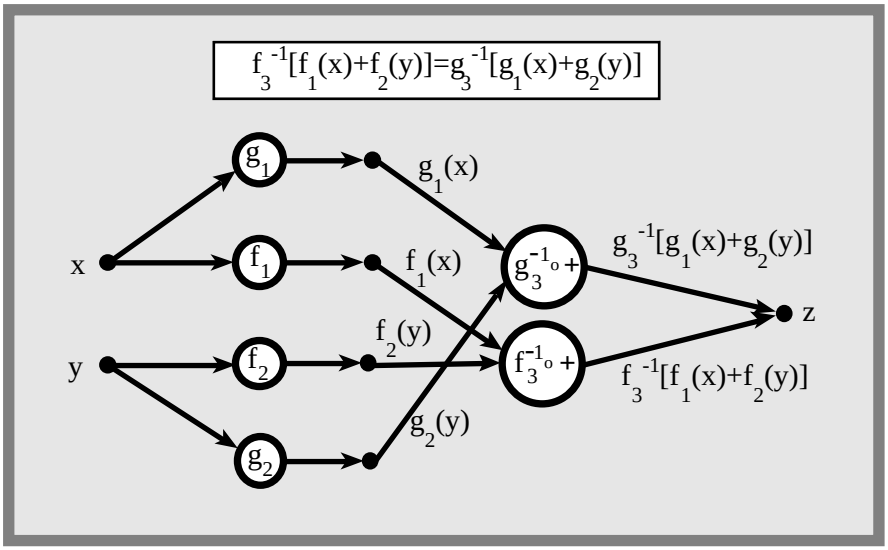
THE GENERALIZED BISYMMETRY FUNCTIONAL EQUATION (UNIQUENESS)

The lower case functions in the solution are not uniquely determined. This implies that there are many different functional networks which are equivalent. In fact, Castillo and Ruiz-Cobo, 1992, page 86, show that if there are two sets of $f_i, g_i, k_i,_i, m_i, p_i, q_i, s_i; i = 1, 2$ such that the solution holds for both sets, then they must be connected by the relations :

$$\begin{array}{lll} l_2(x) = cl_1(x) + a + e - d & ; & f_2(x) = cf_1(x) + a \\ k_2(x) = k_1(\frac{x-a-b}{c}) & ; & g_2(x) = cg_1(x) + b \\ m_2(x) = cm_1(x) + b + d - e & ; & q_2(x) = cq_1(x) + d \ , \\ p_2(x) = cp_1(x) + a - d & ; & r_2(x) = cr_1(x) + e \\ s_2(x) = cs_1(x) + b - e & & \end{array}$$

where a, b, c, d and e are arbitrary constants.

UNIQUENESS OF REPRESENTATION



Theorem 6 *If* $F(x,y)$ *can be written as:*

$$F(x,y) = f_3^{-1}[f_1(x) + f_2(y)] = g_3^{-1}[g_1(x) + g_2(y)]$$

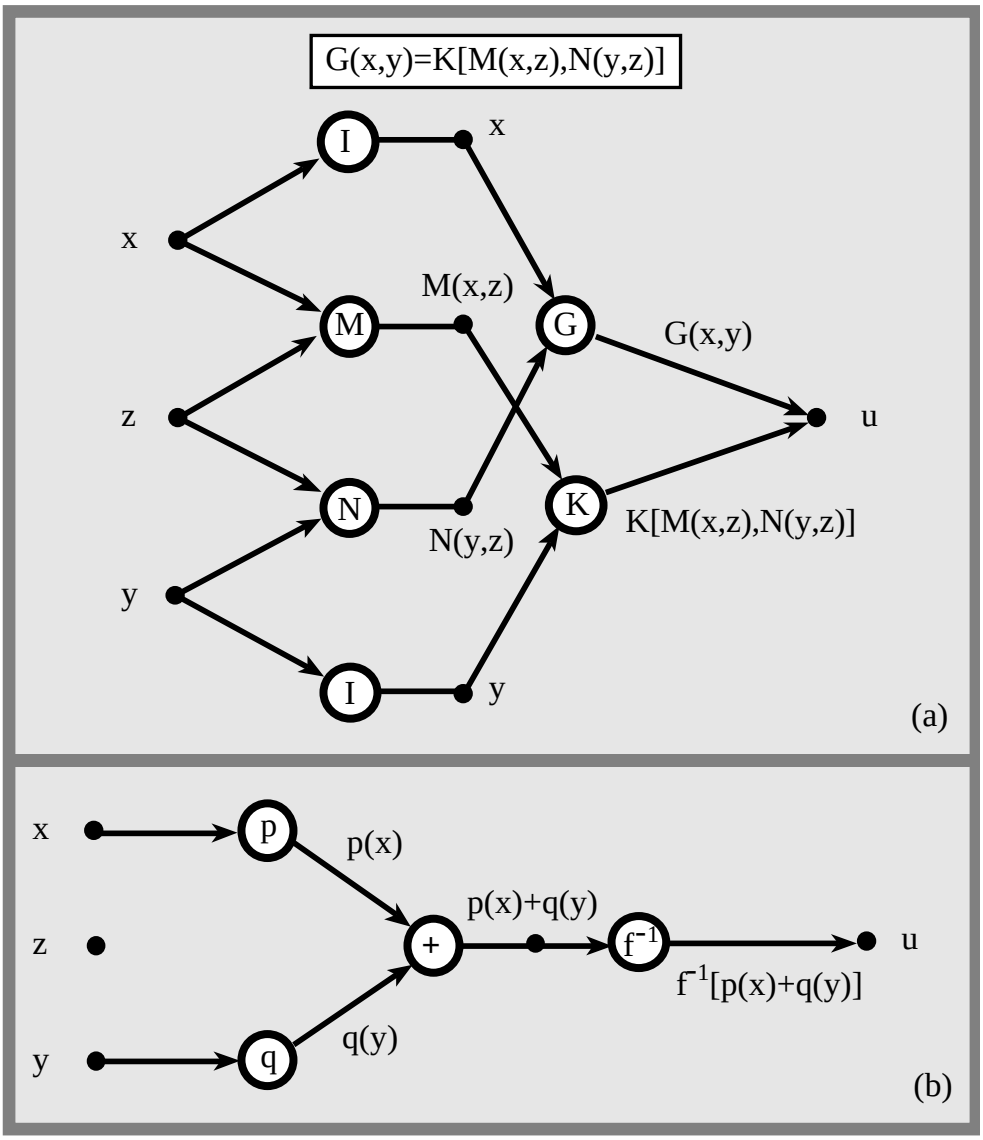
where the functions f_i, g_i *(i=1,2,3) are continuous and strictly monotonic functions, then*

$$\begin{aligned} g_3^{-1}(x) &= f_3^{-1}(\tfrac{x-a-b}{c}), \\ g_1(x) &= cf_1(x) + a, \\ g_2(y) &= cf_2(y) + b, \end{aligned}$$

where a, b *and* $c \neq 0$ *are arbitrary constants.* ■

Constants a, b and c are not identifiable. Any set of values of (a, b, c) leads to the same $F(x, y)$.

ILL-DESIGNED FUNCTIONAL NETWORKS



Some initial functional networks show a false dependence of the output units from some input units.

Theorem 7 *The general solution continuous on a real rectangle of the functional equation*

$$G(x,y) = K[M(x,z), N(y,z)], \qquad (12)$$

with N invertible in the first argument for any value of the second, M invertible in both arguments, K invertible in the first argument for some fixed value of the second and G invertible in the second for a fixed value of the first, is

$$G(x,y) = f^{-1}[p(x) + q(y)] \ , \ K(x,y) = f^{-1}[l(x) + n(y)]$$
$$M(x,y) = l^{-1}[p(x) + r(y)] \ , \ N(x,y) = n^{-1}[q(x) - r(y)]$$

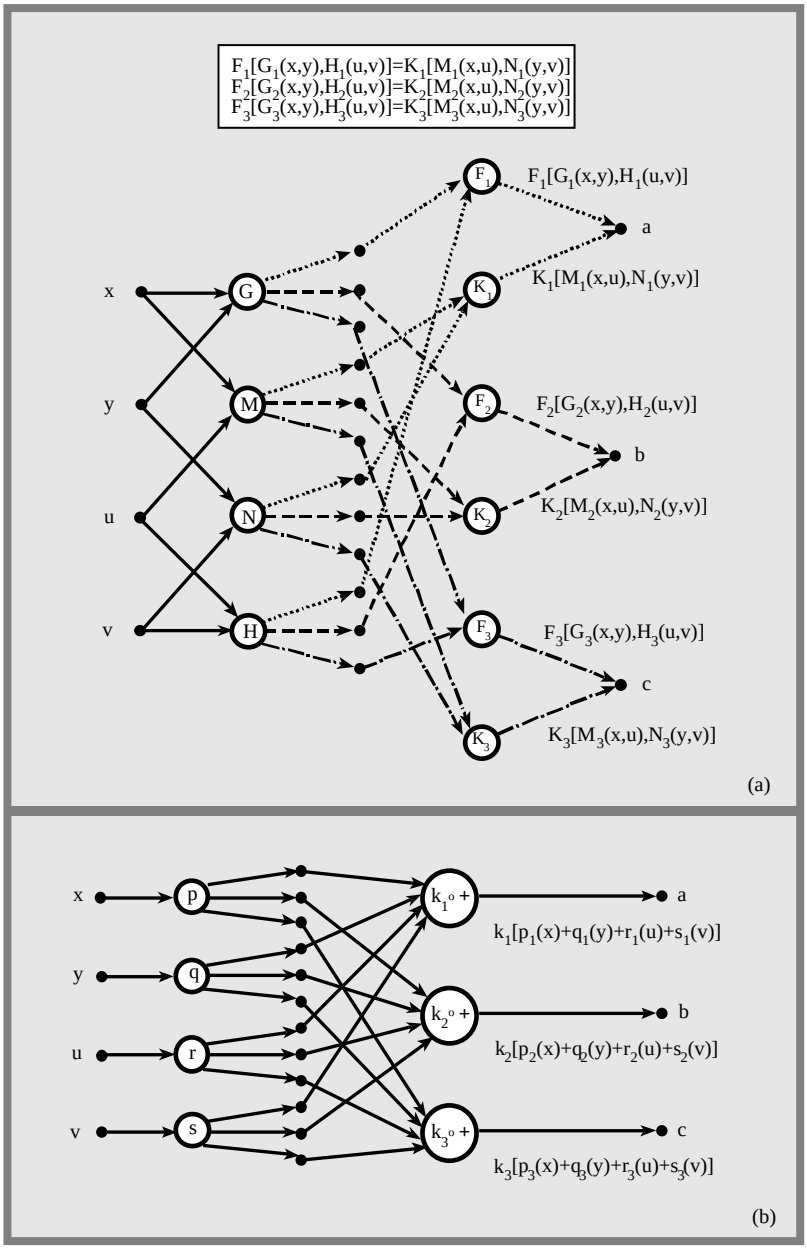
where f, l, n, p, q and r are arbitrary continuous and strictly monotonic functions which are determined up to the following relations

$$f_2(x) = c_1 f_1(x) + a_1 + b_1 \quad ; \quad p_2(x) = c_1 p_1(x) + a_1$$
$$q_2(x) = c_1 q_1(x) + b_1 \quad ; \quad l_2(x) = c_1 l_1(x) + a_4$$
$$n_2(x) = c_1 n_1(x) + b_1 + a_1 - a_4 \quad ; \quad r_2(x) = c_1 r_1(x) + a_4 - a_1$$

where the a's and b's are arbitrary constants. ■

It is clear that the output w does not depend on the input z.

INDEPENDENT MULTIPLE OUTPUT NETWORKS



An independent multiple output network with its corresponding equivalent simplified network. Different dashed lines are for different outputs.

INDEPENDENT MULTIPLE OUTPUT NETWORKS

The compatibility conditions are the following (one per output unit):

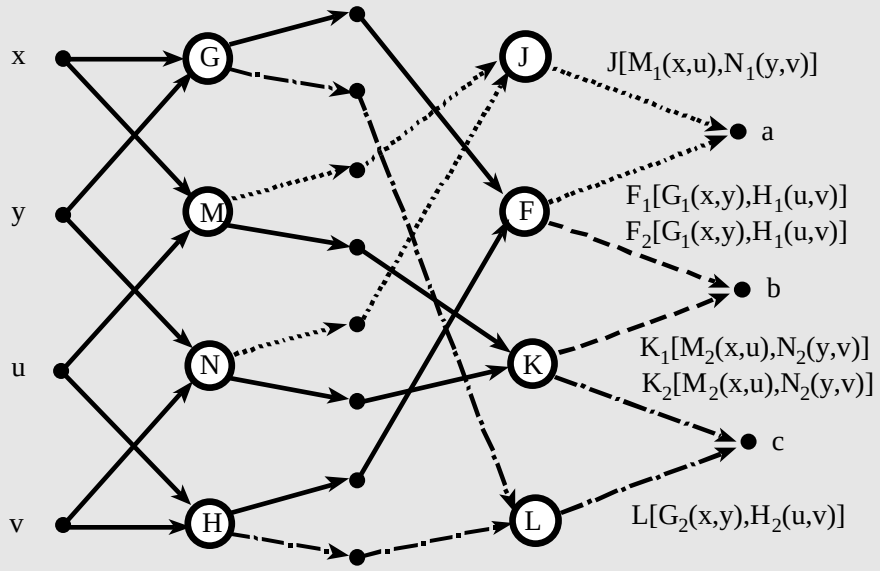
$$\begin{aligned} F_1[G_1(x,y),H_1(u,v)] &= K_1[M_1(x,u),N_1(y,v)] \\ F_2[G_2(x,y),H_2(u,v)] &= K_2[M_2(x,u),N_2(y,v)] \\ F_3[G_3(x,y),H_3(u,v)] &= K_3[M_3(x,u),N_3(y,v)]. \end{aligned}$$

Note that this is equivalent to three functional networks, because in the resulting systems of functional equations associated with the network, all the involved functions are different.

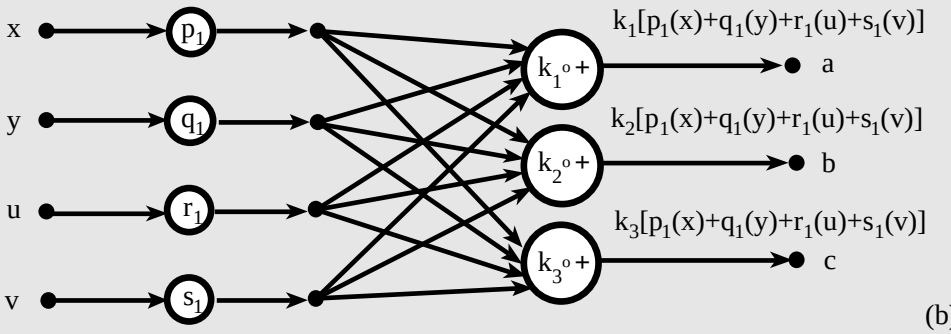
As we shall see, other networks exist in which this does not occur and the same functions appear in several places.

DEPENDENT MULTIPLE OUTPUT NETWORKS I

$$F_1[G_1(x,y),H_1(u,v)] = J[M_1(x,u),N_1(y,v)]$$
$$F_2[G_1(x,y),H_1(u,v)] = K_1[M_2(x,u),N_2(y,v)]$$
$$L[G_2(x,y),H_2(u,v)] = K_2[M_2(x,u),N_2(y,v)]$$



(a)



(b)

DEPENDENT MULTIPLE OUTPUT NETWORKS II

The functional network in the above Figure leads to the following system of functional equations:

$$\begin{aligned} F_1[G_1(x,y), H_1(u,v)] &= J[M_1(x,u), N_1(y,v)] \\ F_2[G_1(x,y), H_1(u,v)] &= K_1[M_2(x,u), N_2(y,v)] \\ L[G_2(x,y), H_2(u,v)] &= K_2[M_2(x,u), N_2(y,v)] \end{aligned}$$

Taking into account that the resulting functional equations are of the form (10), we have

$$\begin{aligned} &F_1(x,y) = k_1[f_1(x) + g_1(y)]; \quad G_1(x,y) = f_1^{-1}[p_1(x) + q_1(y)], \\ &J(x,y) = k_1[l_1(x) + m_1(y)]; \quad H_1(x,y) = g_1^{-1}[r_1(x) + s_1(y)], \\ &M_1(x,y) = l_1^{-1}[p_1(x) + r_1(y)]; \quad N_1(x,y) = m_1^{-1}[q_1(x) + s_1(y)] \end{aligned}$$

$$\begin{aligned} &F_2(x,y) = k_2[f_2(x) + g_2(y)]; \quad G_1(x,y) = f_2^{-1}[p_2(x) + q_2(y)], \\ &K_1(x,y) = k_2[l_2(x) + m_2(y)]; \quad H_1(x,y) = g_2^{-1}[r_2(x) + s_2(y)], \\ &M_2(x,y) = l_2^{-1}[p_2(x) + r_2(y)]; \quad N_2(x,y) = m_2^{-1}[q_2(x) + s_2(y)] \end{aligned}$$

$$\begin{aligned} &K_2(x,y) = k_3[f_3(x) + g_3(y)]; \quad M_2(x,y) = f_3^{-1}[p_3(x) + q_3(y)], \\ &L(x,y) = k_3[l_3(x) + m_3(y)]; \quad N_2(x,y) = g_3^{-1}[r_3(x) + s_3(y)], \\ &G_2(x,y) = l_3^{-1}[p_3(x) + r_3(y)]; \quad H_2(x,y) = m_3^{-1}[q_3(x) + s_3(y)] \end{aligned}$$

DEPENDENT MULTIPLE OUTPUT NETWORKS III

Since two different expressions exist for G_1, H_1, M_2 and N_2, they must coincide, that is:

$$\begin{aligned} f_1^{-1}[p_1(x)+q_1(y)] &= f_2^{-1}[p_2(x)+q_2(y)] \\ g_1^{-1}[r_1(x)+s_1(y)] &= g_2^{-1}[r_2(x)+s_2(y)] \\ l_2^{-1}[p_2(x)+r_2(y)] &= f_3^{-1}[p_3(x)+q_3(y)] \\ m_2^{-1}[q_2(x)+s_2(y)] &= g_3^{-1}[r_3(x)+s_3(y)] \end{aligned} \tag{13}$$

and taking into account the general solution of equations of the form (12) we get:

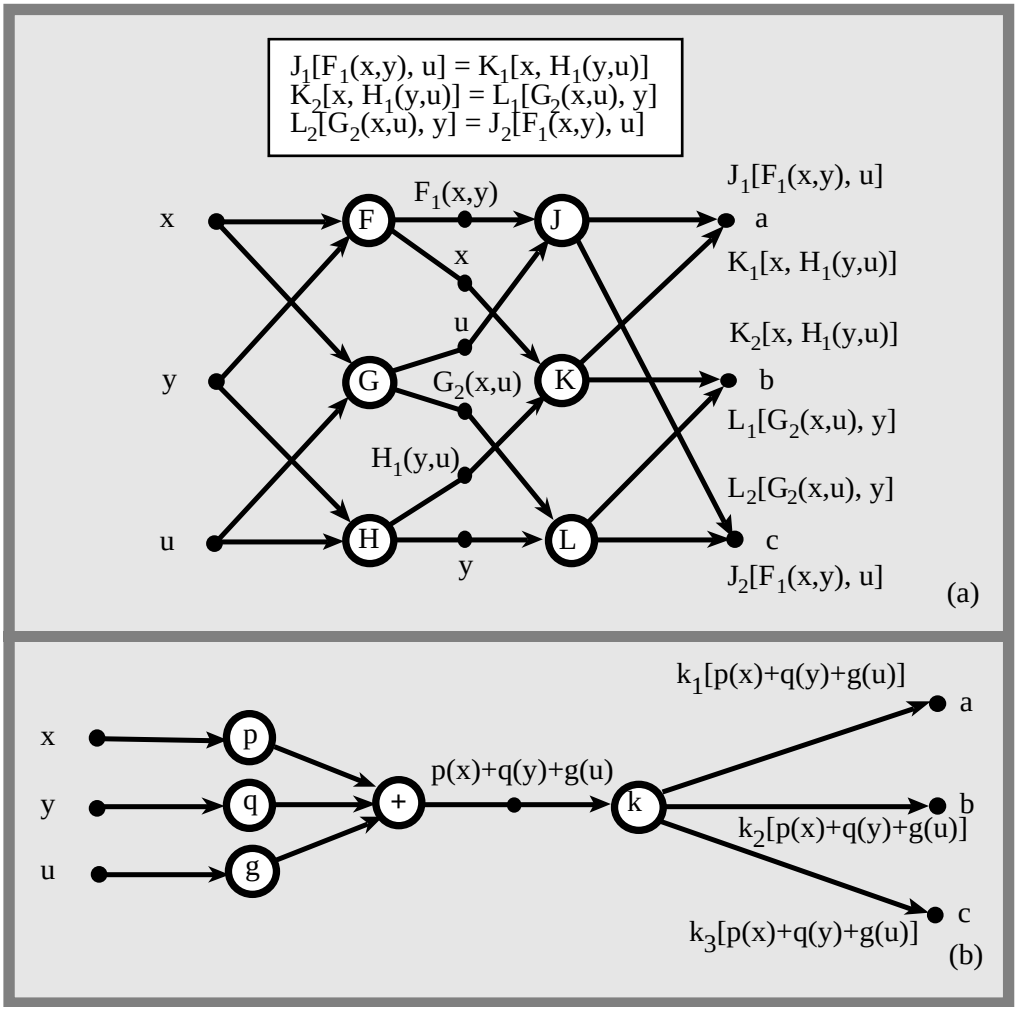
$$\begin{aligned} p_3(x) &= c_1c_3p_1(x)+c_3a_1+a_3; \\ q_3(x) &= c_2c_3r_1(x)+c_3a_2+b_3; \\ r_3(x) &= c_1c_4q_1(x)+c_4b_1+a_4; \\ s_3(x) &= c_2c_4s_1(x)+c_4b_2+b_4; \end{aligned} \tag{14}$$

which finally leads to

$$\begin{aligned} a &= k_1[p_1(x)+q_1(y)+r_1(u)+s_1(v)] \\ b &= k_2[p_1(x)+q_1(y)+r_1(u)+s_1(v)] \\ c &= k_3[p_1(x)+q_1(y)+r_1(u)+s_1(v)], \end{aligned} \tag{15}$$

and proves the equivalence of the two nets in the Figure.

DEPENDENT MULTIPLE OUTPUT NETWORKS IV



$$
\begin{aligned}
J_1[F_1(x,y),u] &= K_1[x,H_1(y,u)] \\
K_2[x,H_1(y,u)] &= L_1[G_2(x,u),y] \\
L_2[G_2(x,u),y] &= J_2[F_1(x,y),u]
\end{aligned}
$$

DEPENDENT MULTIPLE OUTPUT NETWORKS V

Since they are generalized associativity equations:

$$
\begin{array}{ll}
J_1(x,y) = k_1[f_1(x) + g_1(y)]; & F_1(x,y) = f_1^{-1}[p_1(x) + q_1(y)], \\
K_1(x,y) = k_1[p_1(x) + n_1(y)]; & H_1(x,y) = n_1^{-1}[q_1(x) + g_1(y)], \\
L_1(x,y) = k_2[f_2(x) + g_2(y)]; & G_2(x,y) = f_2^{-1}[p_2(x) + q_2(y)], \\
K_2(x,y) = k_2[p_2(x) + n_2(y)]; & H_1(x,y) = n_2^{-1}[g_2(x) + q_2(y)], \\
L_2(x,y) = k_3[f_3(x) + g_3(y)]; & G_2(x,y) = f_3^{-1}[q_3(x) + p_3(y)], \\
J_2(x,y) = k_3[n_3(x) + p_3(y)]; & F_1(x,y) = n_3^{-1}[q_3(x) + g_3(y)],
\end{array}
$$

Coincidence of F_1, H_1 and G_2 leads to:

$$
\begin{aligned}
f_1^{-1}[p_1(x) + q_1(y)] &= n_3^{-1}[q_3(x) + g_3(y)] \\
n_1^{-1}[q_1(x) + g_1(y)] &= n_2^{-1}[g_2(x) + q_2(y)] \\
f_2^{-1}[p_2(x) + q_2(y)] &= f_3^{-1}[q_3(x) + p_3(y)]
\end{aligned}
$$

which implies:

$$
\begin{array}{ll}
g_2(x) = c_2 q_1(x) + a_2; & g_3(x) = c_1 q_1(x) + b_1; \\
p_2(x) = \dfrac{c_1 p_1(x) + a_1 - a_3}{c_3}; & p_3(x) = c_3 c_2 g_1(x) + c_3 b_2 + b_3; \\
q_2(x) = c_2 g_1(x) + b_2; & q_3(x) = c_1 p_1(x) + a_1.
\end{array}
$$

and

$$
\begin{aligned}
a &= k_1[p_1(x) + q_1(y) + g_1(u)] \\
b &= k_2[p_1(x) + q_1(y) + g_1(u)] \\
c &= k_3[p_1(x) + q_1(y) + g_1(u)],
\end{aligned}
$$

EXACT LEARNING: CLONING

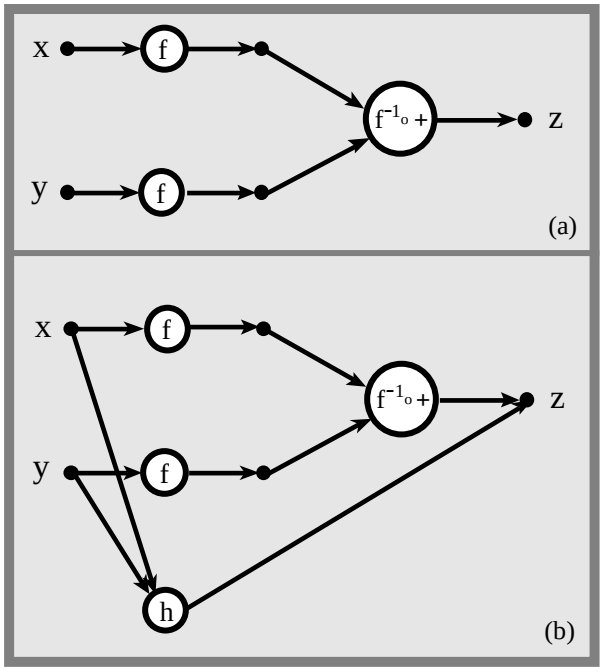
We want to reproduce the associative operation

$$x \oplus y = h(x, y) = xy. \tag{16}$$

Any associative operation $\oplus$ can be written as:

$$x \oplus y = f^{-1}[f(x) + f(y)], \tag{17}$$

where $f(x)$ is an invertible function. Learning the operation $\oplus$ means identifying f. To this end, we can add one more functional unit to the functional net above to get the functional net in the Figure, where $h(x, y) = xy$.



EXACT LEARNING ALGORITHM

This forces the function f to satisfy the equation

$$f^{-1}[f(x)+f(y)] = xy \Leftrightarrow f(xy) = f(x)+f(y),$$

which is a Cauchy equation with solution

$$f(x) = c\log x \Leftrightarrow f^{-1}(x) = \exp\left(\frac{x}{c}\right), c \in \mathbb{R}.$$

ALGORITHM

Input: A functional net N_1 to be cloned.

Output: A clone N_2 of the net N_1.

- **Step 1**: Select a family $\mathcal{N}$ of functional nets with the same input and output nodes as N_1.
- **Step 2**: Connect the functional N_1 to the family $\mathcal{N}$.
- **Step 3**: Learn the implied unknown functions in $\mathcal{N}$ by solving the corresponding functional equations.
- **Step 4**: Disconnect N_1 to recover the desired member N_2 of the original family $\mathcal{N}$.

LEARNING FUNCTIONAL NETWORKS

Learning the neuron function

$$F(x,y) = f_3^{-1}[f_1(x) + f_2(y)]$$

is equivalent to learning $f_1(x), f_2(x)$ and $f_3(x)$ or $f_3^{-1}(x)$.

To learn, we use some data consisting of triplets $\{(x_{1j}, x_{2j}, x_{3j}) | x_{3j} = F(x_{1j}, x_{2j}); j = 1, \ldots, n\}$.

The following estimation methods can be used:

1. **Linear Method**: It is called linear because the associated optimization function leads to a system of linear equations in the parameter estimates. One shortcoming of this method is that it is necessary to invert the $f_3(x)$ function to obtain the output values.
2. **Non-Linear Method**: It leads to a function which is non-linear in the parameters. A clear advantage of this method with respect to the linear method is that the $f_3(x)$ function need not be inverted to obtain the output values.

LEARNING FUNCTIONAL NETWORKS
LINEAR ESTIMATION METHOD

We approximate $f_s(x); s = 1, 2, 3$ by

$$\begin{aligned} \hat{f}_s(x) &= \sum_{i=1}^{m_s} a_{si}\phi_{si}(x);\ s = 1, 2; \\ \hat{f}_3(x) &= -\sum_{i=1}^{m_3} a_{3i}\phi_{3i}(x) \end{aligned}$$

where the $\{\phi_{si}(x); i = 1, \ldots, m_s; s = 1, 2, 3\}$ are sets of given linearly independent functions.

Since we must have

$$f_3(x_{3j}) = f_1(x_{1j}) + f_2(x_{2j}); j = 1, \ldots, n,$$

the error can be measured by

$$e_j = \hat{f}_3(x_{3j}) - \hat{f}_1(x_{1j}) - \hat{f}_2(x_{2j}); j = 1, \ldots, n. \quad (18)$$

Thus, we minimize the sum of square errors

$$Q = \sum_{j=1}^{n} e_j^2 = \sum_{j=1}^{n} \left(\sum_{s=1}^{3} \sum_{i=1}^{m_s} a_{si}\phi_{si}(x_{sj}) \right)^2 \quad (19)$$

subject to

$$\hat{f}_k(x_0) \equiv \sum_{i=1}^{m_k} a_{ki}\phi_{ki}(x_0) = \alpha_k;\ k = 1, 2, 3, \quad (20)$$

where α_k and x_0 are constants, necessary to have uniqueness of solution.

Using the Lagrange multipliers we define the auxiliary function

$$Q_\lambda = \sum_{j=1}^{n}\left(\sum_{s=1}^{3}\sum_{i=1}^{m_s} a_{si}\phi_{si}(x_{sj})\right)^2 + \sum_{k=1}^{3}\lambda_k\left(\sum_{i=1}^{m_k} a_{ki}\phi_{ki}(x_0) - \alpha_k\right)$$

The minimum corresponds to

$$\frac{\partial Q_\lambda}{\partial a_{tr}} = 2\sum_{j=1}^{n}\left(\sum_{s=1}^{3}\sum_{i=1}^{m_s} a_{si}\phi_{si}(x_{sj})\right)\phi_{tr}(x_{jt}) + \lambda_t\phi_{tr}(x_0) = 0$$

$$\frac{\partial Q_\lambda}{\partial \lambda_t} = \sum_{i=1}^{m_t} a_{ti}\phi_{ti}(x_0) - \alpha_t = 0;$$

which is valid for $t = 1, 2, 3$.

An interesting simplification consists of assuming $f_3(x) = x$, which implies an error

$$e_j = x_{3j} - \hat{f}_1(x_{1j}) - \hat{f}_2(x_{2j}); j = 1, \ldots, n. \qquad (21)$$

Note that in order to get a linear system of equations we measure the error in (18) in terms of the transformed variable $f_3(x)$. This is the price we need to pay for linearity.

On the contrary, with the error in (21), we measure the errors in the variable scale.

We approximate $f_s(x); s = 1, 2$ by

$$\hat{f}_s(x) \;=\; \sum_{i=1}^{m_s} a_{si}\phi_{si}(x);\;\; s = 1, 2;$$

and $f_3^{-1}(x)$ by

$$\hat{f}_3^{-1}(x) \;=\; \sum_{k=1}^{m_3} a_{3k}\phi_{3k}(x),$$

where the $\{\phi_{si}(x); i = 1, \ldots, m_s; s = 1, 2, 3\}$ are sets of given linearly independent functions.
We must have

$$x_{3j} = f_3^{-1}\left(f_1(x_{1j}) + f_2(x_{2j})\right); j = 1, \ldots, n,$$

thus, the error can be measured by

$$e_j = x_{3j} - \hat{f}_3^{-1}\left(\hat{f}_1(x_{1j}) + \hat{f}_2(x_{2j})\right); j = 1, \ldots, n. \quad (22)$$

Thus, we minimize the sum of square errors

$$Q = \sum_{j=1}^{n} e_j^2 = \sum_{j=1}^{n}\left(x_{3j} - \sum_{k=1}^{m_3} a_{3k}\phi_{3k}\left(\sum_{s=1}^{2}\sum_{i=1}^{m_s} a_{si}\phi_{si}(x_{sj})\right)\right)^2, \quad (23)$$

subject to

$$\hat{f}_k(x_0) \equiv \sum_{i=1}^{m_k} a_{ki}\phi_{ki}(x_0) = \alpha_k;\;\; k = 1, 2, 3, \quad (24)$$

where α_k and x_0 are constants, necessary to have uniqueness of solution.

Thus, we minimize the sum of square errors

$$Q = \sum_{j=1}^{n} e_j^2 = \sum_{j=1}^{n} \left(x_{3j} - \sum_{k=1}^{m_3} a_{3k}\phi_{3k} \left(\sum_{s=1}^{2} \sum_{i=1}^{m_s} a_{si}\phi_{si}(x_{sj}) \right) \right)^2, \tag{25}$$

subject to

$$\hat{f}_k(x_0) \equiv \sum_{i=1}^{m_k} a_{ki}\phi_{ki}(x_0) = \alpha_k; \; k = 1, 2, 3, \tag{26}$$

where α_k and x_0 are constants, necessary to have uniqueness of solution.

One way of considering this constraint consists of solving (24) in their first coefficients and replace them in (23), that is,

$$a_{k1} = \frac{\alpha_k - \sum_{i=2}^{m_k} a_{ki}\phi_{ki}(x_0)}{\phi_{k1}(x_0)}; \; k = 1, 2, 3. \tag{27}$$

Note that in this case we always measure the errors in the variable scale.

LEARNING FUNCTIONAL NETWORKS (EXAMPLE)

Consider the data

x_{j1}	x_{j2}	x_{j3}	x_{j1}	x_{j2}	x_{j3}
0.890	0.462	1.172	0.935	0.21	1.065
0.695	0.313	0.755	0.861	0.741	1.296
0.069	0.083	0.085	0.026	0.256	0.228
0.167	0.006	0.034	0.888	0.939	1.451
0.701	0.529	0.916			

and assume that for $s = 1, 2, 3$:

$$(\phi_{s1}(x), \ldots, \phi_{s4}(x)) = (1, x, x^2, \log(1+x))$$

and

$$(\alpha_1, \alpha_2, \alpha_3) = (1, \log 2, -1).$$

Then, minimizing we obtain

$$\begin{array}{llllllll} a_{11} = 0; & a_{12} = 0; & a_{13} = 1; & a_{14} = 0; \\ a_{21} = 0; & a_{22} = 0; & a_{23} = 0; & a_{24} = 1; \\ a_{31} = 0; & a_{32} = 1; & a_{33} = 0; & a_{34} = 0, \end{array}$$

which was due to the fact that the data in the above Table was simulated from

$$x_{j1} \oplus x_{j2} = x_{j1}^2 + \log(1 + x_{j2}) = x_{j3};\ j = 1, \ldots, n$$

EXAMPLE OF LEARNING FUNCTIONAL NETWORKS

To illustrate the unidentifiability of the constants a, b and c, we consider the same case above but using

$$(\alpha_1, \alpha_2, \alpha_3) = (1, 1, 1).$$

Then, the a coefficients become

$$\begin{array}{llll} a_{11} = -3.33; & a_{12} = 0.0; & a_{13} = 4.3; & a_{14} = 0.0; \\ a_{21} = -2.0; & a_{22} = 0.0; & a_{23} = 0.0; & a_{24} = 4.3; \\ a_{31} = 5.33; & a_{32} = -4.3; & a_{33} = 0.0; & a_{34} = 0.0, \end{array}$$

which shows that both solutions are related by

$$\begin{aligned} g_3^{-1}(x) &= f_3^{-1}(\tfrac{x-a-b}{c}), \\ g_1(x) &= cf_1(x) + a, \\ g_2(y) &= cf_2(y) + b, \end{aligned}$$

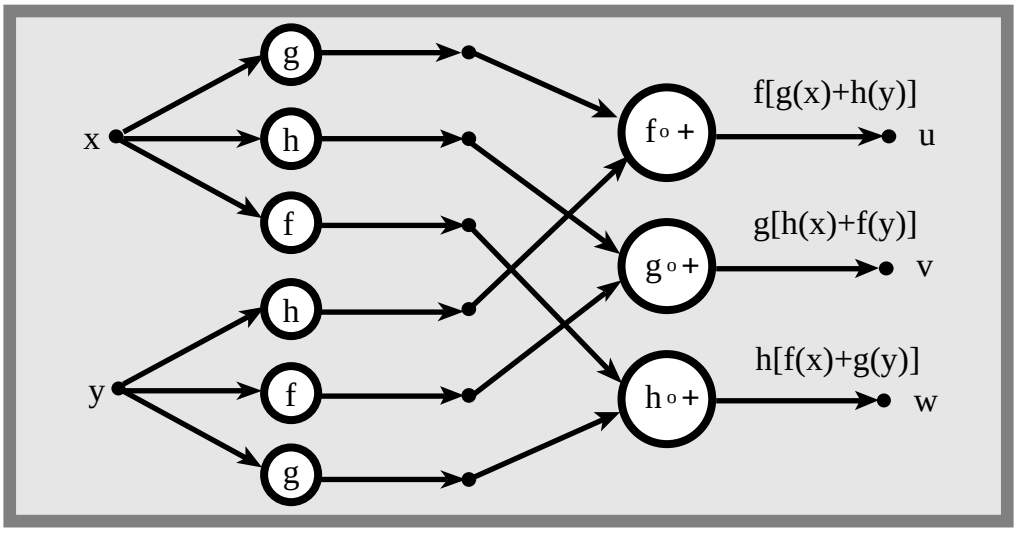
with $a = -3.3, b = -2$ and $c = 4.3$.

EXAMPLE OF LEARNING FUNCTIONAL NETWORKS

x	y	u	v	w	x	y	u	v	w
0.987	0.977	3.63	13.4	6.97	0.548	0.682	2.28	5.82	2.76
0.831	0.341	2.1	6.95	2.58	0.777	0.0759	1.68	5.06	2.19
0.547	0.00965	1.31	3.02	1.73	0.53	0.401	1.77	4.41	1.99
0.247	0.435	1.61	2.94	1.55	0.524	0.715	2.32	5.78	2.82
0.773	0.205	1.82	5.62	2.26	0.113	0.929	2.55	4.2	2.66
0.764	0.0629	1.65	4.89	2.16	0.693	0.52	2.16	6.35	2.62
0.777	0.0862	1.69	5.11	2.19	0.145	0.838	2.33	3.98	2.33
0.946	0.745	3.	11.	4.49	0.369	0.762	2.28	4.87	2.58
0.399	0.736	2.25	4.96	2.56	0.839	0.361	2.14	7.16	2.64
0.152	0.301	1.37	2.15	1.27	0.315	0.646	2.01	4.06	2.08

These data are simulated with:

$$u = x^2 + \exp(y); v = (\exp(x) + y)^2; w = \exp(x + y^2).$$



$$f(x) = x,\ g(x) = x^2 \text{ and } h(x) = \exp(x)$$

EXAMPLE OF LEARNING FUNCTIONAL NETWORKS

To estimate the functions f, g and h, we use

$$\begin{aligned}\hat{f}(x) &= a_1 + a_2 x + a_3 x^2 + a_4 \exp(x)\\ \hat{g}(x) &= b_1 + b_2 x + b_3 x^2 + b_4 \exp(x)\\ \hat{h}(x) &= c_1 + c_2 x + c_3 x^2 + c_4 \exp(x)\end{aligned}$$

Minimizing the sum of squares we get

$$\begin{array}{llll} a_1 = 0.0126, & a_2 = 1.002, & a_3 = 0.0035, & a_4 = -0.0014\\ b_1 = -0.0013, & b_2 = 0.0019, & b_3 = 1.0139, & b_4 = -0.0051\\ c_1 = 0.1434, & c_2 = 0.095, & c_3 = 0.1610, & c_4 = 0.8476 \end{array}$$

With the aim of validating the model, a different sample of size $m = 1000$ has been used and the values of the variables u, v and w have been predicted from the values of the x and y variables using the estimated model. The following mean errors have been obtained:

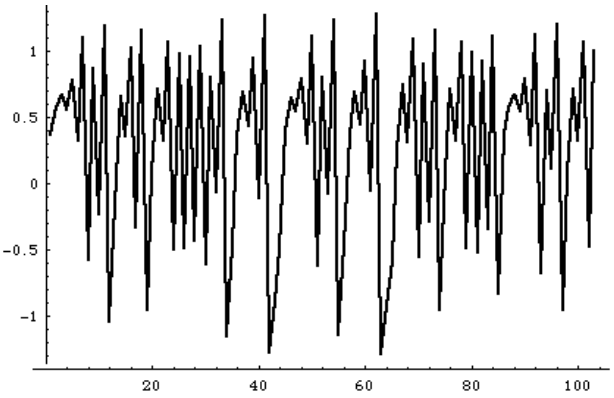
$$\begin{aligned} e_1 &= \frac{1}{m}\sum_{i=1}^{m} |\hat{f}(\hat{g}(x_i) + \hat{h}(y_i)) - u_i| = 0.00397426,\\ e_2 &= \frac{1}{m}\sum_{i=1}^{m} |\hat{g}(\hat{h}(x_i) + \hat{f}(y_i)) - v_i| = 0.00800446,\\ e_3 &= \frac{1}{m}\sum_{i=1}^{m} |\hat{h}(\hat{f}(x_i) + \hat{g}(y_i)) - w_i| = 0.00986862, \end{aligned}$$

which show that the predictions are excellent.

THE HENON SERIES EXAMPLE

The Hénon's time series with starting values 0.3638 and 0.5630 is (see Figure):

$$x_n = 1.0 - 1.4x_{n-1}^2 + 0.3x_{n-2}. \tag{28}$$



To reproduce these series, we use the model:

$$x_n = f_3^{-1}\left[f_1(x_{n-1}) + f_2(x_{n-2})\right], \tag{29}$$

with the set $\{\phi_{s1}(x), \ldots, \phi_{s6}(x)\}; s = 1, 2, 3$:

$$\{1, \log(2+x), \ldots, \log(6+x)\}. \tag{30}$$

To be fair, we have not used the exact functions x or x^2.

We get $RMSE = 0.00074$ and $MaxErr = 0.00252$, much better than Stern values (0.00471 and 0.0187, respectively).

THE HENON SERIES EXAMPLE

The Table below also shows the root mean squared error (RMSE) and the single largest prediction error on the 100 training samples and the 5000 testing examples, where the number t after the word $Network$ refers to the number of functions used in the set (30). For $t = 6$ we used the preceding set of functions, for $t = 5$ we removed the last function $\log(6 + x)$, and for $t = 7$ we added the function $\log(7 + x)$.
We have evaluated the model with the following 5000 terms, and obtained $RMSE = 0.00083$ and $MaxErr = 0.00314$. These values are consistent with the errors on the training data. Thus, overfitting was not a problem even though we used 18 parameters. Again the results are much better than those given by Stern.

	Stern's	Network 5	Network 6	Network 7
Parameters	29	15	18	21
Train. RMSE	0.00471	0.0045	0.00074	0.00065
Test RMSE	0.00549	0.0045	0.00083	0.00072
Train. MaxErr	0.0187	0.0177	0.00252	0.00204
Test MaxErr	0.0218	0.0189	0.00314	0.00269

THE AR(2) SERIES EXAMPLE

A set of 100 data points for each of 4 time series $AR(2)$ models of the following form (see Figures)

$$x_n = \alpha_1 x_{n-1} + \alpha_2 x_{n-2} + \epsilon_n \tag{31}$$

are analyzed, where ϵ_n are i.i.d. normal deviates with standard deviation σ_e and coefficients α_1 and α_2 given in the Table below.

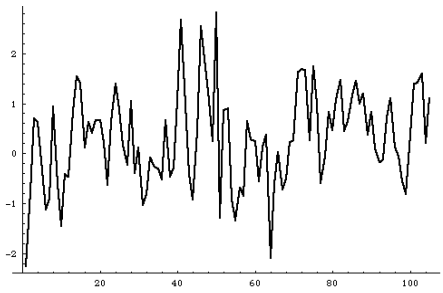
			Training set RMSE		Test set RMSE	
α_1	α_2	σ_e	NN	FN	NN	FN
0.70	-0.49	0.7695	0.571	0.867	1.018	0.898
0.90	-0.81	0.5088	0.422	0.527	0.625	0.513
0.90	-0.97	0.2163	0.177	0.424	0.295	0.234
1.4	-0.99	0.1002	0.084	0.096	0.133	0.102

The Table below shows a summary of the main results obtained for the neural network proposed by Stern, and the functional network associated with the model

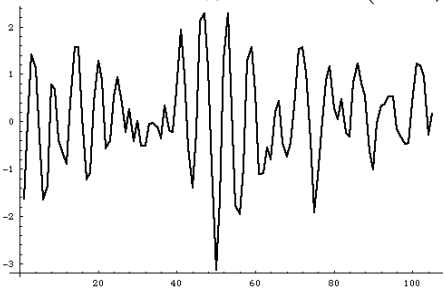
$$x_n = f_1(x_{n-1}) + f_2(x_{n-2}) + \epsilon_n, \tag{32}$$

where the f_1 and f_2 functions have been approximated by the set $\phi = \{1, x\}$.

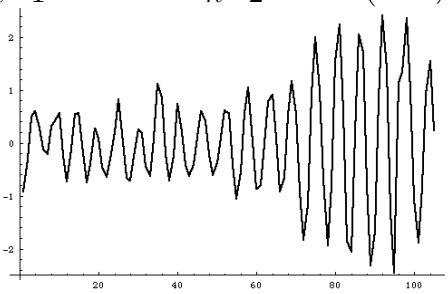
THE AR(2) SERIES EXAMPLE



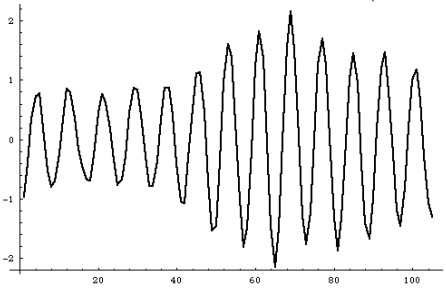
$$x_n = 0.70x_{n-1} - 0.49x_{n-2} + N(0.0, 0.7695^2).$$



$$x_n = 0.90x_{n-1} - 0.81x_{n-2} + N(0.0, 0.5088^2).$$



$$x_n = 0.90x_{n-1} - 0.97x_{n-2} + N(0.0, 0.2163^2).$$



$$x_n = 1.40x_{n-1} - 0.99x_{n-2} + N(0.0, 0.1002^2).$$

THE AR(2) SERIES EXAMPLE

Note that the associative functional network is only adequate for the case $\alpha_1 = \alpha_2 = 1$.

To test the quality, a simulated series of 5000 data points have been used with the results in the Table.

There is some evidence of overfitting to the training sample for the Stern neural network because the fitted RMSE is smaller than the standard deviations of the stochastic disturbance used to generate the time series and much larger for the test series (see Table). However, the results for the functional network do not show this problem of overfitting (the RMSEs are larger or similar to the standard deviations (see Table)) and, in addition, it gives much better predictions (smaller RMSE for the test sample).

THE SPANISH DATA EXAMPLE

As a final example we use real data from Alegre, Arcarons, Bolancé and Díaz, 1995.

Year	Stock index	Price index	% of savings
1964	103.55	112.67	23.18
1965	108.64	123.22	22.35
1966	111.62	129.71	22.54
1967	111.86	138.22	21.89
1968	146.29	142.19	22.54
1969	215.14	147.07	24.37
1970	192.36	157.03	24.37
1971	220.88	172.18	24.36
1972	291.74	184.81	24.59
1973	328.94	211.06	24.99
1974	294.47	248.80	24.32
1975	306.28	283.85	23.29
1976	218.80	340.00	21.30
1977	147.23	429.79	20.75
1978	131.58	501.01	21.72
1979	109.98	578.82	20.40
1980	116.60	667.03	19.40

THE SPANISH DATA EXAMPLE

To reproduce these series, we use the model:

$$s_t = f_1(m_t) + f_2(c_t);\ \ f_3(x) = x, \qquad (33)$$

where s_t is the percentage of savings, m_t is the general Madrid stock index, and c_t is the consumer price index.

We selected the set of functions for $s = 1, 2$:

$$\{\phi_{s1}(x), \phi_{s2}(x), \phi_{s3}(x), \phi_{s4}(x)\} = \{1, x, \log(x), x^2\}.$$

The resulting approximating functions are:

$$f_1(x) = 1.09 - 0.089x + 13.33 \log x + 0.00011x^2$$
$$f_2(x) = 54.76 + 0.11x - 20.72 \log x - 0.000064x^2,$$

leading to a root mean squared error (RMSE) of 0.523661 and a largest prediction error of 1.019.

THE SPANISH DATA EXAMPLE

The Figure shows the observed (continuous line) and the estimated values (dashed line) of the percentage of savings.

The results are good even though the selected functional network is simple.

LEARNING THE ASSOCIATIVITY FUNCTIONAL NETWORK

Learning the neuron function

$$F(x,y) = f^{-1}[f(x) + f(y)]$$

is equivalent to learning $f(x)$.

We can approximate $f(x)$ by

$$\hat{f}(x) = \sum_{i=1}^{m} a_i \phi_i(x) \tag{34}$$

where the $\{\phi_i(x); i = 1, \ldots, m\}$ are given linearly independent functions, capable of approximating $f(x)$ to the desired accuracy.

To estimate $\{a_i; i = 1, \ldots, m\}$, we use some data consisting of triplets $\{(x_{j1}, x_{j2}, y_j) | y_j = F(x_{j1}, x_{j2}) = x_{j1} \oplus x_{j2}; j = 1, \ldots, n\}$.

We must have

$$f(y_j) = f(x_{j1}) + f(x_{j2}); j = 1, \ldots, n \tag{35}$$

thus, the error can be measured by

$$e_j = \hat{f}(x_{j1}) + \hat{f}(x_{j2}) - \hat{f}(y_j); j = 1, \ldots, n. \tag{36}$$

Thus, we minimize the sum of squared errors

$$Q = \sum_{j=1}^{n} e_j^2 = \sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_i \left[\phi_i(x_{j1}) + \phi_i(x_{j2}) - \phi_i(y_j) \right] \right)^2$$

subject to

$$f(x_0) \equiv \sum_{i=1}^{m} a_i \phi_i(x_0) = \alpha,$$

where α is an arbitrary but given real constant, which is necessary to identify c.
Using the Lagrange multipliers we build

$$Q_\lambda = \sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_i b_{ij} \right)^2 + \lambda \left(\sum_{i=1}^{m} a_i \phi_i(x_0) - \alpha \right) \qquad (37)$$

where

$$b_{ij} = \phi_i(x_{j1}) + \phi_i(x_{j2}) - \phi_i(y_j). \qquad (38)$$

The minimum corresponds to

$$\frac{\partial Q_\lambda}{\partial a_r} = 2 \sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_i b_{ij} \right) b_{rj} + \lambda \phi_r(x_0) = 0; \ \forall r$$
$$\frac{\partial Q_\lambda}{\partial \lambda} = \sum_{i=1}^{m} a_i \phi_i(x_0) - \alpha = 0,$$

THE ASSOCIATIVE OPERATION EXAMPLE

To learn an associative operation of two real numbers, in the interval $(0, 5)$, we can take pairs of numbers in that interval and their operated values as triplets

$$\{(x_{j1}, x_{j2}, y_j) | y_j = F(x_{j1}, x_{j2}) = x_{j1} \oplus x_{j2}; \forall j\}.$$

We can do this deterministically or we can simulate a set of triplets. Assume that we simulate 10 of these triplets and obtain the values:

x_1	x_2	y	x_1	x_2	y
0.433	0.842	1.275	2.263	1.634	3.897
1.111	2.066	3.177	3.071	1.147	4.218
1.928	2.381	4.309	2.292	2.113	4.405
0.691	4.527	5.218	2.747	2.210	4.957
2.801	3.658	6.459	3.050	3.951	7.001

Since we know that the operation is associative, we have:

$$f(x \oplus y) = f(x) + f(y).$$

THE ASSOCIATIVE OPERATION EXAMPLE

Using the set $\{\phi_i(x); i = 1, \ldots, m\}$ of functions:

$$\{1, \log(x+1), \log(x+2), \log(x+3), \log(x+4)\},$$

with $m = 5$, where we have removed the function x, because we know it is the exact solution (the sum operation), and solve the linear system of equations (51), we obtain

$$(a_1, a_2, a_3, a_4, a_5, \lambda)$$
$$= (-55.3, -21.8, 200.4, -468.4, 311.1, -0.0002).$$

To check the quality of the approximation, we have simulated 1000 pairs of random numbers in the interval $(0, 3)$, calculated their exact and approximate values, and obtained the resulting RMSE (root mean squared error) for the cases of the preceding set of functions $\phi_i(x)$ and the two sets obtained by sequentially adding the new functions $\log(x+5)$ and $\log(x+6)$.

Number of functions (m)	5	6	7
RMSE	0.0476	0.0177	0.0021

APPLICATIONS TO REGRESSION

Regression models reproduced by this network:

1. **Linear Regression Models**: They are of the type
$$z = \sum_{i=1}^{m} f_i(x) + \epsilon, \tag{39}$$
where the f_i functions are to be estimated.

2. **Non-Linear Regression Models**: of the type
$$z = h[\sum_{i=1}^{m} f_i(x)] + \epsilon, \tag{40}$$
where the f_i and h functions are to be estimated.

3. **Other Non-Linear Regression Models**: For example,
$$z = [\sum_{i=1}^{m} a_i x_i^{\alpha_i}]^{\beta} + \epsilon, \tag{41}$$
where the functions $a_i x_i^{\alpha_i}; i = 1, \ldots, m$ and u^{β} are estimated.

APPROXIMATE LEARNING I

Assume now that instead of knowing the h function, we have a set of available data (triplets):

$$D = \{(x_t, y_t, (x \oplus y)_t) | t \in \mathcal{T}\}.$$

Then, we can approximate the f function by

$$\hat{f}(x) = \sum_{i=0}^{k} c_i x^i.$$

Since we have

$$f^{-1}[f(x) + f(y)] = x \oplus y \Leftrightarrow f(x \oplus y) = f(x) + f(y),$$

to estimate $\{c_i | i = 0, \ldots, k\}$, we can minimize, with respect to c_i, the sum of squared errors

$$\begin{aligned} Q &= \sum_{t \in \mathcal{T}} e_t^2 = \sum_{t \in \mathcal{T}} \left(f((x \oplus y)_t) - f(x_t) - f(y_t)\right)^2 \\ &= \sum_{t \in \mathcal{T}} \sum_{i=0}^{k} \left(c_i (x \oplus y)_t^i - c_i x_t^i - c_i y_t^i\right)^2, \end{aligned}$$

where e_t is the error for the data point t. From this minimization process, we obtain the c_i values, that is, an estimate of the f function.

Since in this case, the estimated function $\hat{f}$ does not reproduce exactly the $\oplus$ operation, we say that we have an approximate learning.

APPROXIMATE LEARNING II

The approximate learning process consists of selecting the "best" function g_i in a given class $\mathcal{C}$, using $D = \{(x^{(t)}_{input}, x^{(t)}_{output}) | t \in \mathcal{T}\}$.

To this end, a measure $d(D; D_i(g_i))$ is selected to measure the discrepancy between the observed data D and the functional net values $D_i(g_i)$. Since the aim of the estimation process consists of obtaining the optimal function g_i, we minimize the discrepancy with respect to $g_i \in \mathcal{C}$, i.e.,

$$\min_{g_i \in \mathcal{C}} d(D; D_i(g_i)).$$

As one example, we can use

$$d(D; D_i(g_i)) = \sum_{t \in \mathcal{T}} \| f_{c_i}(x^{(t)}_{input}) - x^{(t)}_{output} \|^2, \qquad (42)$$

where $\| \, . \, \|$ is a norm in $\mathbb{R}^{|X_{output}|}$ and f_{c_i} is the processing function of the compact form of the functional net associated with g_i.

Note that some of the neural functions g_i can be known, and then, the learning can be partial.

THE ITERATOR EXAMPLE I

Consider the function

$$f(x) = \log(1 + \exp(x)). \tag{43}$$

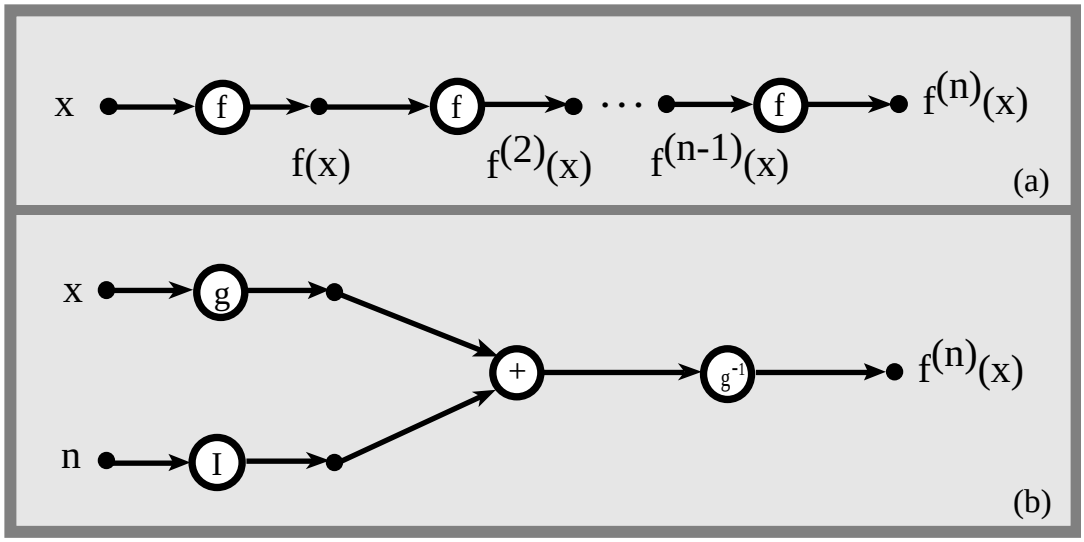
and assume that we are interested in calculating its n-th f-iterate $F(x, n) = f^{(n)}(x)$.
It is well known that $F(x, n)$ can be written as

$$f^{(n)}(x) = F(x, n) = g^{-1}[g(x) + n] \tag{44}$$

for some $g(.)$, and then

$$f(x) = g^{-1}[g(x) + 1]. \tag{45}$$

To this end we can use the functional network in the Figure.



THE ITERATOR EXAMPLE II

To estimate g we have the data pairs in Table,

x_t	y_t	x_t	y_t	x_t	y_t
0.681	1.091	0.919	1.255	0.612	1.045
0.156	0.774	0.728	1.122	0.140	0.766
0.167	0.780	0.888	1.233	0.968	1.290
0.444	0.940	0.925	1.259	0.787	1.162
0.126	0.758	0.493	0.969	0.175	0.785
0.579	1.024	0.519	0.986	0.061	0.724
0.883	1.229	0.436	0.935		

where $y_t = f(x_t)$, and from (45) we consider that

$$g(y_t) = g(x_t) + 1; \forall t = 1, \ldots, 20, \qquad (46)$$

and make $g(x) = \sum_{i=1}^{4} c_i x^i$, that is, we approximate $g(x)$ by a fourth degree polynomial.

Coefficient c_0 has no influence and can be assumed to be zero.

The error in each data point is

$$e_t = g(y_t) - g(x_t) - 1 = \sum_{i=0}^{k} c_i (y_t^i - x_t^i) - 1 \qquad (47)$$

THE ITERATOR EXAMPLE III

Thus, to estimate $c_1, \ldots, c_4$, we minimize

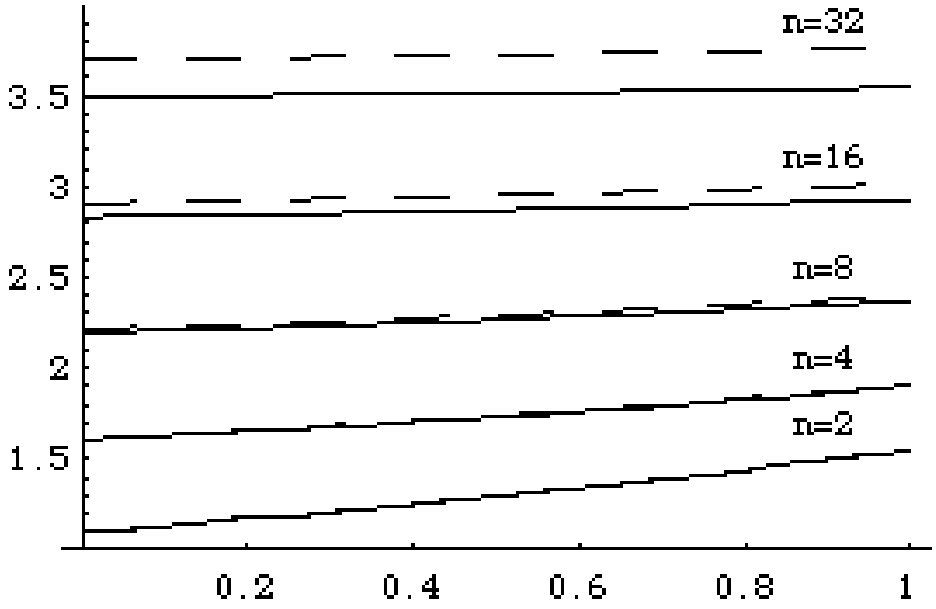
$$Q = \sum_{t=1}^{20} (\sum_{i=0}^{k} c_i (y_t^i - x_t^i) - 1)^2. \qquad (48)$$

The resulting polynomial approximation is

$$\hat{g}(x) = 0.992x + 0.536x^2 + 0.106x^3 + 0.0844x^4 \qquad (49)$$

The Figure shows the exact (continuous lines) and approximate (dashed lines) f-iterates of orders $2, 4, 8, 16$ and 32:

$$\hat{f}^{(n)}(x) = \hat{g}^{-1}[\hat{g}(x) + n]. \qquad (50)$$



THE BEAM EXAMPLE I

Castillo (1996) shows how the the usual mathematical model in terms of differential equations:

$$\begin{aligned} q'(x) &= p(x) \\ m'(x) &= q(x) \\ w'(x) &= \frac{m(x)}{EI} \\ z'(x) &= w(x), \end{aligned} \tag{51}$$

can be written in terms of functional equations:

$$\begin{aligned} q(x+u) &= q(x) + A(x,u) \\ m(x+u) &= m(x) + uq(x) + B(x,u) \\ w(x+u) &= w(x) + \tfrac{1}{EI}\left[m(x)u + q(x)\tfrac{u^2}{2} + C(x,u)\right] \\ z(x+u) &= z(x) + w(x)u + \\ &\quad \tfrac{1}{EI}\left[m(x)\tfrac{u^2}{2} + q(x)\tfrac{u^3}{6} + D(x,u)\right] \end{aligned}$$

where

$$\begin{aligned} A(x,u) &= \int_x^{x+u} p(s)ds \\ B(x,u) &= \int_x^{x+u} (x+u-s)p(s)ds \\ C(x,u) &= \int_x^{x+u} B(x,s-x)ds \\ D(x,u) &= \int_x^{x+u} C(x,s-x)ds \end{aligned} \tag{52}$$

THE BEAM EXAMPLE II

Alternatively, from (51) we have.

$$EIz^{(iv)}(x) = p(x). \tag{53}$$

To obtain an equivalent functional equation in $z(x)$, we can write $z(x+u)$ for three different values of u and eliminate $w(x), m(x)$ and $q(x)$. For example, for $u, 2u, 3u$ and $4u$, we get:

$$\begin{aligned} z(x+4u) &= 4z(x+u) - 6z(x+2u) + 4z(x+3u) \\ &\quad -z(x) + (-4D(x,u) + 6D(x,2u) \\ &\quad -4D(x,3u) + D(x,4u))/EI, \end{aligned}$$

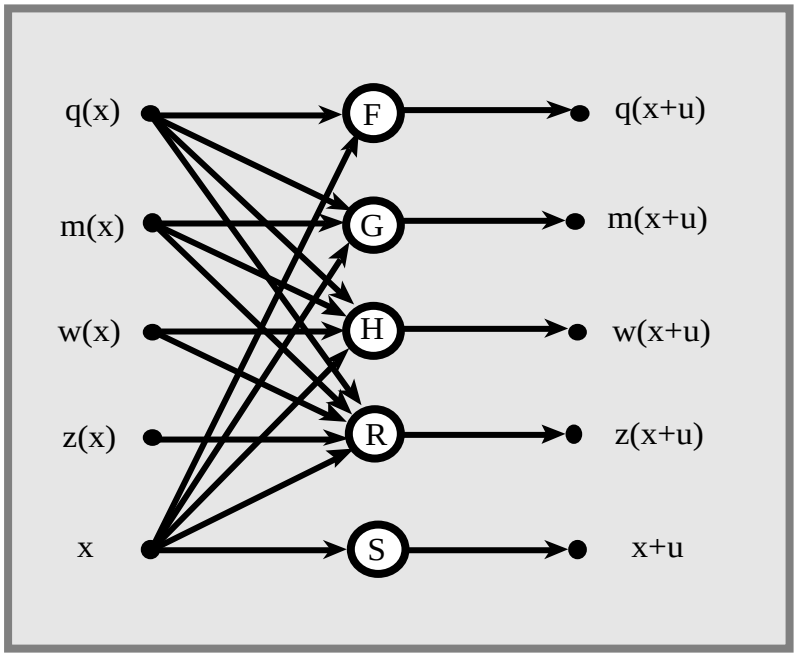
which is equivalent to (53). Similarly:

$$\begin{aligned} w(x+3u) &= w(x) - 3w(x+u) + 3w(x+2u) \\ &\quad + \frac{[3C(x,u) - 3C(x,2u) + C(x,3u)]}{(EI)} \end{aligned}$$

$$\begin{aligned} m(x+2u) &= 2m(x+u) - m(x) - B(x,u) \\ q(x+u) &= q(x) + A(x,u), \end{aligned}$$

which can be interpreted as finite difference equations. In this case, they give the exact solution at the interpolating points.

THE BEAM EXAMPLE III



$$
\begin{aligned}
q(x+u) &= q(x) + A(x,u) \\
m(x+u) &= m(x) + uq(x) + B(x,u) \\
w(x+u) &= w(x) + \tfrac{1}{EI}\left[m(x)u + q(x)\tfrac{u^2}{2} + C(x,u)\right] \\
z(x+u) &= z(x) + w(x)u + \\
&\quad \tfrac{1}{EI}\left[m(x)\tfrac{u^2}{2} + q(x)\tfrac{u^3}{6} + D(x,u)\right],
\end{aligned}
$$

We use the approximation:

$$
\begin{aligned}
A(x,u) &= a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 \\
B(x,u) &= b_0 + b_1 x + b_2 x^2 + b_3 x^3 + b_4 x^4 \\
C(x,u) &= c_0 + c_1 x + c_2 x^2 + c_3 x^3 + c_4 x^4 \\
D(x,u) &= d_0 + d_1 x + d_2 x^2 + d_3 x^3 + d_4 x^4
\end{aligned}
$$

THE BEAM EXAMPLE IV

The vectors $(q(x), m(x), w(x), z(x), x)$ measured in a real clamped at both ends beam corresponding to a load $p(x) = -1 - \exp(-x)$, which is unknown to the analyst are:

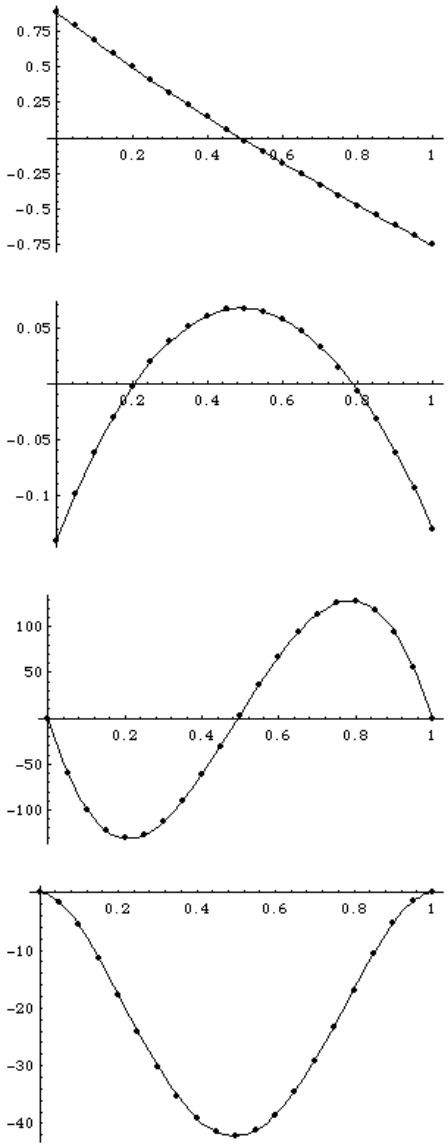
$q(x)$	$m(x)$	$w(x)$	$z(x)$	x
0.878	-0.14	0.	0.	0
0.779	-0.0989	-59.6	-1.58	0.05
0.683	-0.0623	-99.7	-5.63	0.1
0.589	-0.0305	-123.	-11.3	0.15
0.497	-0.00339	-131.	-17.7	0.2
...	...	...	...	...
-0.754	-0.13	0.00	0.0	1.0

To estimate the 20 parameters we minimize

$$\begin{aligned} Q = & \sum_x \left(F(q(x), x) - q(x+u)\right)^2 + \\ & \sum_x \left(G(m(x), q(x), x) - m(x+u)\right)^2 + \\ & \sum_x \left(H(w(x), m(x), q(x), x) - w(x+u)\right)^2 + \\ & \sum_x \left(R(z(x), w(x), q(x), x) - z(x+u)\right)^2 . \end{aligned}$$

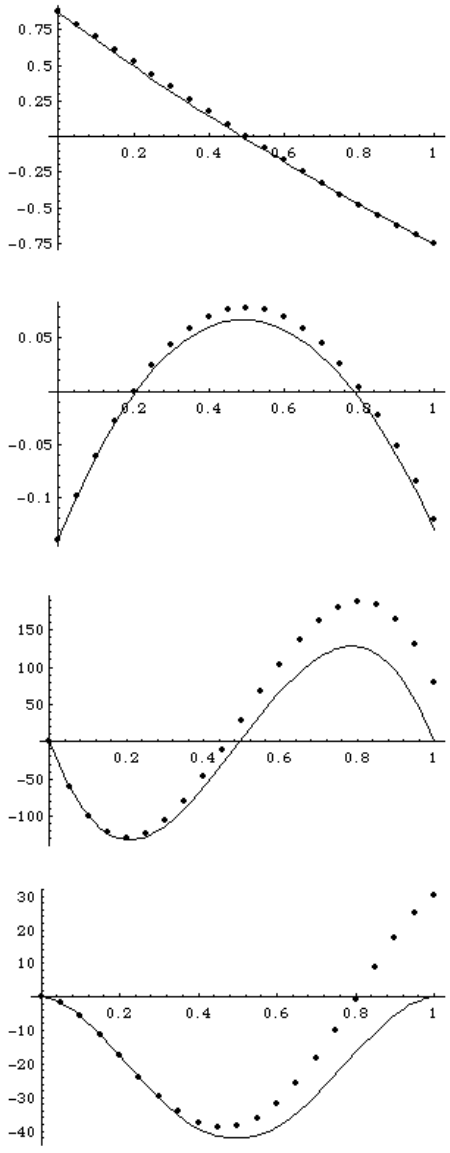
with respect to $a_0, \ldots, b_0, \ldots, c_0, \ldots, d_0, \ldots$.

THE BEAM EXAMPLE V



Measured vectors of the beam and the predictions using the model above. Vectors are predicted based on the previous observed vectors.

THE BEAM EXAMPLE VI



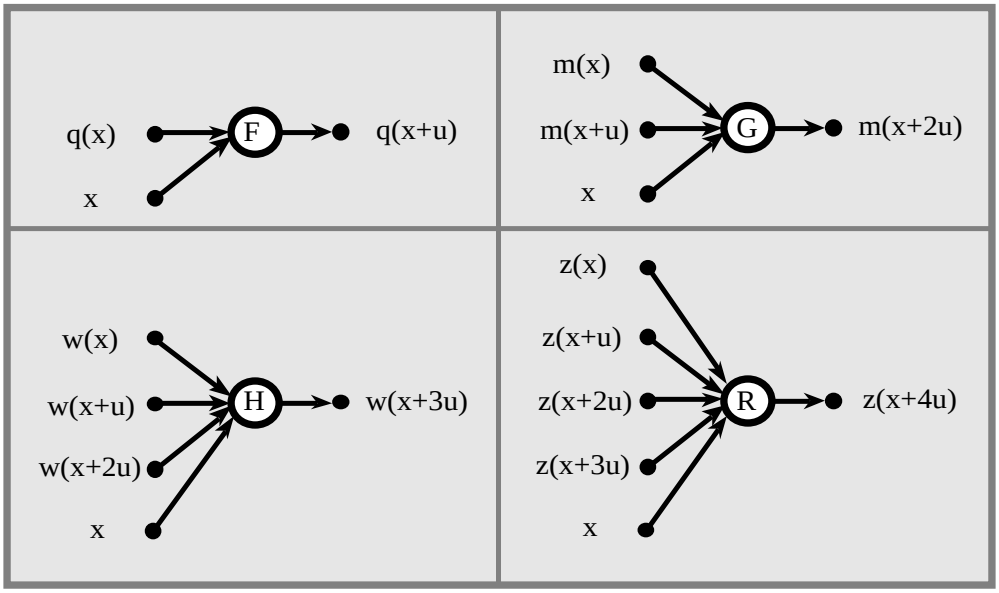
Measured deflection of the beam and the prediction using the model above. Vectors are predicted based on the left most observed vectors.

THE BEAM EXAMPLE VII

As a second alternative we take

$$(-4D(x,u) + 6D(x,2u) - 4D(x,3u) + D(x,4u))/EI$$
$$= c_0 + c_1 x + c_2 x^2 + c_3 x^3 + c_4 x^4$$

which corresponds to the functional network in the lower right corner of the Figure.



For estimating the parameters we minimize

$$Q = \sum_x [z(x+4u)$$
$$-4z(x+u) + 6z(x+2u) - 4z(x+3u) + z(x)$$
$$-c_0 - c_1 x - c_2 x^2 - c_3 x^3 - c_4 x^4]^2,$$

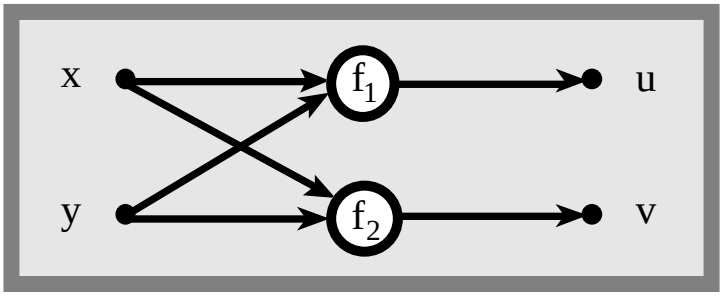
with respect to $\{c_0, c_1, c_2, c_3, c_4\}$.

SIMULATING NORMAL VARIABLES

If X and Y are two variables $U(0,1)$, then,

$$\begin{aligned} U &= (-2\log X)^{\frac{1}{2}} sin(2\pi Y) \\ V &= (-2\log X)^{\frac{1}{2}} cos(2\pi Y) \end{aligned} \qquad (54)$$

are normal $N(0,1)$ and independent. Thus, the functional net in the Figure allows obtaining the values u and v of two independent $N(0,1)$.



The functional net is invertible, because

$$\begin{aligned} x &= g_1(u,v) = e^{-\frac{u^2+v^2}{2}} \\ y &= g_2(u,v) = \frac{1}{2\pi} arctg(\frac{u}{v}) \end{aligned}$$

and its jacobian J satisfies:

$$J = \frac{1}{2\pi} \begin{vmatrix} \frac{v}{u^2+v^2} & -ue^{-\frac{u^2+v^2}{2}} \\ \frac{-u}{u^2+v^2} & -ve^{-\frac{u^2+v^2}{2}} \end{vmatrix} = \frac{1}{2\pi} e^{-\frac{u^2+v^2}{2}} \neq 0 \quad ; \forall\, u, v$$

THE LORENZ ATTRACTOR I

The Lorenz Attractor has differential equations:

$$\begin{aligned} \dot{x} &= a(y-x) \\ \dot{y} &= x(c-z)-y \\ \dot{z} &= xy-bz, \end{aligned} \tag{55}$$

or difference equations:

$$\begin{aligned} x_{t+\Delta} &= a\Delta(y_t - x_t) + x_t \\ y_{t+\Delta} &= \Delta x_t(c-z_t) + y_t(1-\Delta) \\ z_{t+\Delta} &= \Delta x_t y_t + z_t(1-b\Delta), \end{aligned} \tag{56}$$

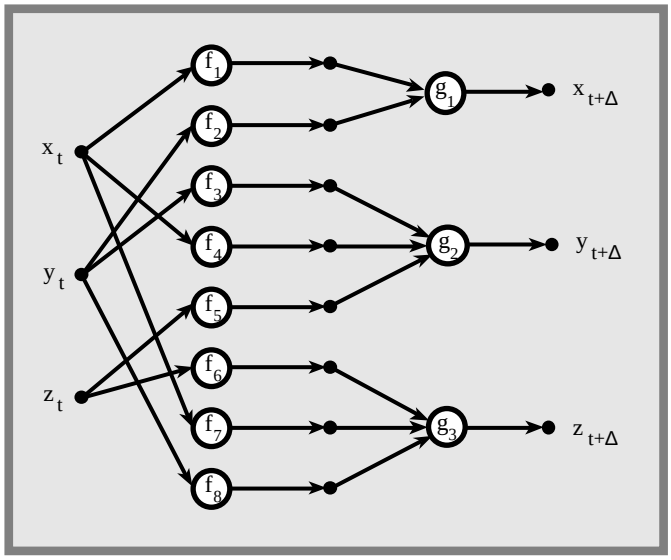
where Δ is the step size.
We use the functional network in Figure, where

$$\begin{cases} x_{t+\Delta} = g_1[f_1(x_t), f_2(y_t)] \\ y_{t+\Delta} = g_2[f_3(y_t), f_4(x_t), f_5(z_t)] \\ z_{t+\Delta} = g_3[f_6(z_t), f_7(x_t), f_8(y_t)], \end{cases} \tag{57}$$

and $f_1, \ldots, f_8$ and $g_1, \ldots, g_3$ are to be estimated.
We have approximated the f_i, g_i functions by:

$$\begin{aligned} \hat{f}_i(x) &= a_{0i} + a_{1i}x; \quad i = 1, \ldots, 8 \\ \hat{g}_1(x,y) &= b_0 + b_1 x + b_2 y + b_3 xy, \\ \hat{g}_i(x,y,z) &= c_{0i} + c_{1i}x + c_{2i}y + c_{3i}z + c_{4i}xy + c_{5i}xz \\ &\quad + c_{6i}yz; \quad i = 1, 2 \end{aligned}$$

THE LORENZ ATTRACTOR II



and used a randomly generated set of data $\{(x_t, y_t, z_t, x_{t+\Delta}, y_{t+\Delta}, z_{t+\Delta})|t = 1, \ldots, 50\}$.
To estimate the polynomial coefficients we minimized the discrepancy measure

$$\begin{aligned} Q \;=\; & \sum_{t=1}^{50}(x_{t+\Delta} - \hat{g}_1(\hat{f}_1(x_t), \hat{f}_2(y_t)))^2 \\ & + \sum_{t=1}^{50}(y_{t+\Delta} - \hat{g}_2(\hat{f}_3(x_t), \hat{f}_4(y_t), \hat{f}_5(z_t)))^2 \\ & + \sum_{t=1}^{50}(z_{t+\Delta} - \hat{g}_3(\hat{f}_6(x_t), \hat{f}_7(y_t), \hat{f}_8(z_t)))^2 \end{aligned} \qquad (58)$$

To validate the estimation, we simulated 1000 more data points and calculated the mean error $Q/1000$, obtaining a value of 0.000077, which shows the high quality of the fit.

IMPLEMENTING FUNCTIONAL NETWORKS

The implementation of functional networks reduces to:

1. **Selecting the set of basic functions** $\phi_{si}(x); i = 1, \ldots, m; s = 1, \ldots, r$.
2. **Building the matrix of the linear system of equations**.
3. **Solving the system of linear equations**: Any of the well known and efficient methods can be used to this end.
4. **Calculate the value Q of the error function**.
5. **Check the quality of the approximation**: If the error Q is large start again.
6. **Build the neural functions**: Based on the resulting functions $\phi_{si}(x)$.
7. **Use the model**.

STEPS FOR WORKING WITH FUNCTIONAL NETWORKS

1. **Understanding of the problem to be solved**: This is a crucial step.
2. **Selecting the topology of the initial functional network**: Based on the knowledge of the problem.
3. **Simplifying the initial functional network using functional equations**: Functional equations allow simplifying networks.
4. **Obtaining the required data to learn the functional network**: Data must be collected to learn the neural functions.
5. **Estimating or learning the neural functions**: The minimization methods allow estimating the neural functions.
6. **Cross validation of the model**: A cross validation of the model is convenient.
7. **Use of the model**: If the validation process is satisfactory, the model is ready to be used.