# Neuronal Networks: Algorithms and Architectures for Ecologists and Evolutionary Ecologists

S. Lek · J.L. Giraudel · J.F. Guégan

## 1.1 Introduction

In ecological research, the processing and interpretation of data play an important role. The ecologist disposes of many methods, ranging from numerical, mathematical, and statistical methods to techniques originating from artificial intelligence (Ackley et al. 1985) like expert systems (Bradshaw et al. 1991; Recknagel et al. 1994), genetic algorithms (d'Angelo et al. 1995; Golikov et al. 1995) and artificial neuronal networks, i.e. ANN (Colasanti 1991; Edwards and Morse 1995).

ANNs were initially developed as models of biological neurons. They are intelligent, thinking machines, working in the same way as the animal brain. They learn from experience in a way that no conventional computer can, and they rapidly solve hard computational problems. With the increasing use of computers, these models could be simulated, and later research was also directed at exploring the possibilities of using and improving these models for performing specific tasks. Research into ANNs has led to the development of various types of neuronal networks, suitable for resolving different types of problems including auto-associative memory, generalization, optimization, data reduction, control and prediction tasks in various scenarios and architectures. Chronologically, we can cite the Perceptron (Rosenblatt 1958), ADALINE, i.e. ADAptive LINear Element (Widrow and Hoff 1960), Hopfield network (Hopfield 1982), Kohonen network (Kohonen 1984), Boltzmann machine (Ackley et al. 1985), and multilayer feed-forward neuronal networks learned by back propagation algorithm (Rumelhart et al. 1986). Descriptions of these methods can be found in various books such as Freeman and Skapura (1992), Gallant (1993), Smith (1994), Bishop (1995), Ripley (1996), etc. The choice of the type of network depends on the type of the problem to be solved. At present, two popular ANNs are multilayer feed-forward neuronal networks, both trained by back propagation algorithm, i.e. back propagation network (BPN) and Kohonen self-organizing mapping, i.e. Kohonen network (SOM). The BPN are the most often used, but other networks are also gaining in popularity nowadays with the emergence of new techniques in various areas of the sciences.

In the last decade research into ANNs has shown explosive growth. They are often applied in physics research like in speech recognition (Rahim et al. 1993; Chu and Bose 1998) or image recognition (Dekruger and Hunt 1994; Cosatto and Graf 1995; Kung and Taur 1995) and in chemical research (Kvasnicka 1990; Wythoff et al. 1990; Smits et al. 1992). In biology most applications of ANNs have been in medicine and molecular biology (Lerner et al. 1994; Albiol et al. 1995; Faraggi and Simon 1995; Lo et al. 1995). Nevertheless, a few applications of this method were reported in ecological and environmental sciences at the beginning of the 1990s. For instance, Colasanti (1991) found

similarities between ANNs and ecosystems and recommended the utilization of this tool in ecological modelling. In a review of computer-aided research in biodiversity, Edwards and Morse (1995) underlined that ANNs have an important potential. Relevant examples are found in very different fields in applied ecology, such as modelling the greenhouse effect (Seginer et al. 1994), predicting various parameters in brown trout management (Baran et al. 1996; Lek et al. 1996a,b), modelling spatial dynamics of fish (Giske et al. 1998), predicting phytoplankton production (Scardi 1996; Recknagel et al. 1997), predicting fish diversity (Guégan et al. 1998), predicting the production/biomass (P/B) ratio of animal populations (Brey et al. 1996), predicting farmer risk preferences (Kastens and Featherstone 1996), etc. Most of these works showed that ANNs performed better than more classical modelling methods.

This book contains working examples of ANN solutions to real ecological problems in various areas. The tasks are as diverse as the neuronal architectures and algorithms themselves, although no attempt has been made to include an example of every shape and form of ANN. We have organized this book so every chapter deals with an interesting example, in which an ANN has been shown to offer a good solution or not. The present textbook is the result of the experiences of leading practitioners in ANN ecological modelling, and we thank them all most warmly.

This book is organized in several chapters. In Chapter 1, two very popular ANN algorithms will be presented: a back propagation neuronal network (BPN) and a Kohonen self-organizing mapping (SOM) network. This chapter offers the reader new to ANN, an introduction with illustrative ecological examples and a comparison to the more classical statistical methods. The following chapters are gathered by ecological theme, from ecosystem studies to evolutionary ecology and even including topics such as remote sensing. The papers show how to obtain solutions for each ecological problem, often making reference to the more classical statistical or mathematical methods like linear or logistic regressions, discriminant analysis, or models based on differential equations, etc.

## 1.2 Back Propagation Neuronal Network (BPN)

The back propagation neuronal networks, also called multilayer feed-forward neuronal networks or multilayer perceptron, are very popular and are used more than other types of neuronal networks for a wide variety of problems. The BPN is based on the supervised procedure, i.e. the network builds a model based on examples in data with known outputs (Fig. 1.1). It has to extract the input-output relation solely from the examples presented, which together are implicitly assumed to contain the information necessary for this relation. The relationship between problem (input) and solution (output) may be quite general, e.g. the simulation of species richness (where the problem is defined by the characteristic of environment and the solution by the value of species richness) or the abundance of animal expressed by the quality of habitat. A BPN is a powerful system, often capable of modelling complex relationships between variables. It allows one to predict an output object for a given input object.

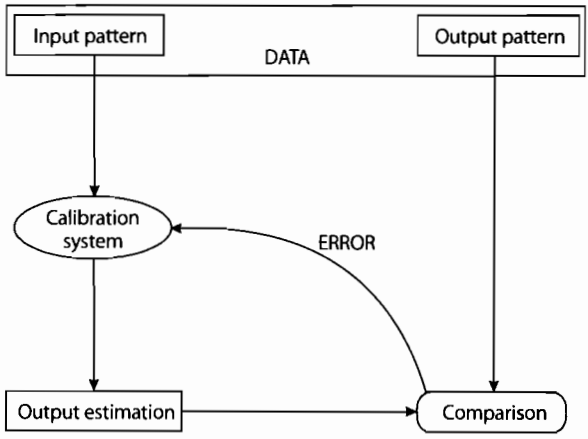


**Fig. 1.1.** Diagram showing how data are used to establish the model calibration in the supervised modelling procedure. The goal of supervised learning is to find a model, or mapping, that will correctly associate the inputs with the output (or target) data

### 1.2.1 Structure of BPN

The BPN architecture is a layered feed-forward neuronal network, in which the nonlinear elements (neurons) are arranged in successive layers, and the information flows unidirectionally, from input layer to output layer, through the hidden layer(s) (Fig. 1.2). As can be seen in this figure, nodes from one layer are connected (using interconnections or links) to all the nodes in the adjacent layer(s), but no lateral connection between nodes within one layer, or feedback connection(s) are possible. This is in contrast with recurrent networks where feedback connections are also permitted. The number of input and output units depends on the representations of the input and the output objects, respectively. The hidden layer(s) is (are) an important parameter in the network. The BPN with an arbitrary number of hidden units has been shown to be a universal approximate (Cybenko 1989; Hornick et al. 1989) for continuous maps and can therefore be used to implement any function defined in these terms.

### 1.2.2 BPN Algorithm

BPN is one of the easiest networks to understand. Its learning and update procedure is based on a relatively simple concept: if the network gives the wrong answer, then the weights are corrected so that the error lessens, so future responses of the network are more likely to be correct. The conceptual basis of the back propagation algorithm was first presented in 1974 by Webos, then independently reinvented by Parker (1982), and presented to a wide readership by Rumelhart et al. (1986).

In a training phase, a set of input/target pattern pairs is used for training, which is presented to the network many times. After training is stopped, the performance of the network is tested. The BPN learning algorithm involves a forward-propagating step followed by a backward-propagating step.

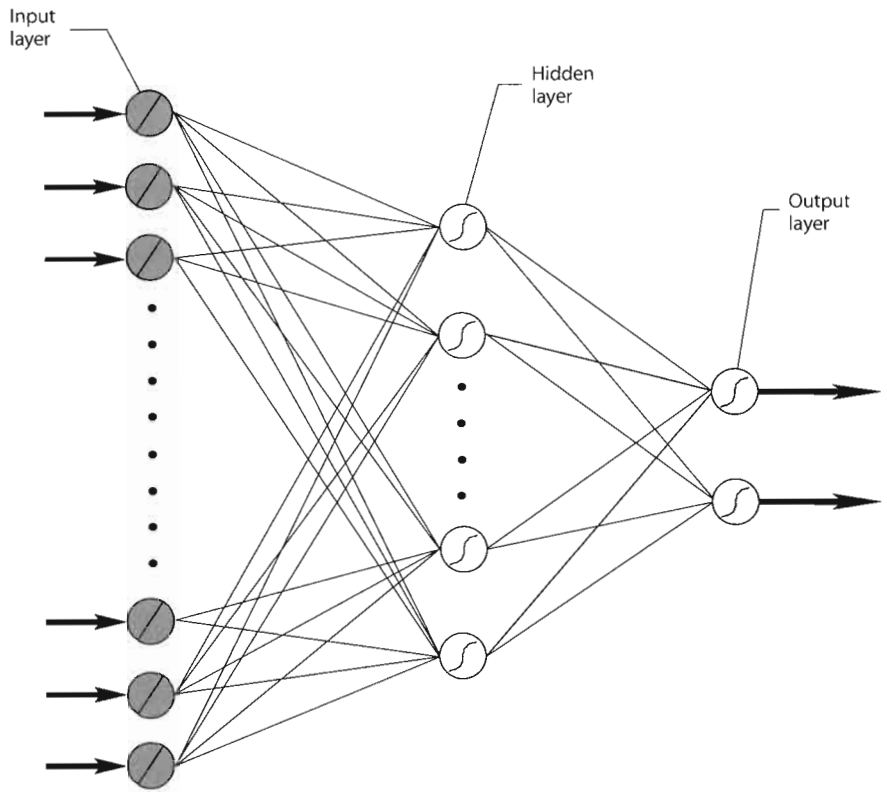


**Fig. 1.2.** Schematic illustration of a three-layered feed-forward neuronal network, with one input layer, one hidden layer and one output layer

### 1.2.2.1
### *Forward-Propagating Step*

In a natural neuron, the dendrites receive signals from other neurons and send them to the cell body, which elaborates a response. The axon receives response signals from the cell body and carries them away through the synapse to the dendrites of neighbouring neurons. In ANNs, the computational element, i.e. the processing element, is called a neuron (sometimes referred to as node or unit). Figure 1.3 shows the general appearance of a neuron with its connections. Each neuron is numbered; the one in the figure is the $j$th. Like a real neuron, the artificial neuron has many inputs, but only a single output, which can stimulate many other neurons in the network. The input the $j$th neuron receives from the $i$th neurons is indicated as $x$. Each connection to the $j$th neuron is associated to a quantity called weight or connection strength. The weight on the connection from the $i$th neuron to the $j$th neuron is denoted $w_{ji}$. An input connection may be excitatory (positive weight) or inhibitory (negative weight). A net input

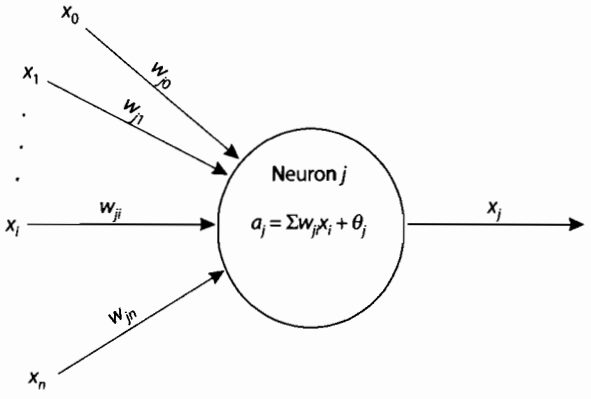


**Fig. 1.3.** Basic processing element (neuron) in a network. Each input connection value ($x_i$) is associated with a weight ($w_{ji}$). The output value can fan out to other units

(called activation) for each neuron is the sum of all its input values multiplied by their corresponding connection weights, expressed by the formula:

$$a_j = \sum_i w_{ji} x_i + \theta_j \tag{1.1}$$

where $i$ is the total number of neurons in the previous layer, $\theta_j$ is a bias term, which influences the horizontal offset of the function. The bias $\theta_j$ may be treated as the weight from the supplementary input unit, which has a fixed output value of 1. Once the activation of the neuron is calculated, we can determine the output value (i.e. the response) by applying a transfer function:

$$x_j = f(a_j) \tag{1.2}$$

Many transfer functions may be used, e.g. linear function, a threshold function, a sigmoid function, etc. (Fig. 1.4). A sigmoid function is often used. Its formula is:

$$x_j = f(a_j) = \frac{1}{1 + e^{-a_j}} \tag{1.3}$$

The weights play an important role in propagation of the signal in the network. They establish a link between an input pattern and the associated output pattern, i.e. they contain the knowledge of the neuronal network about the problem/solution relationship.

The forward-propagating step begins with the presentation of an input pattern to the input layer, and continues as activation level calculations propagate forward till the output layer through the hidden layer(s). In each successive layer, every neuron sums its inputs and then applies a transfer function to compute its output. The output layer of the network then produces the final response, i.e. the estimated target value.

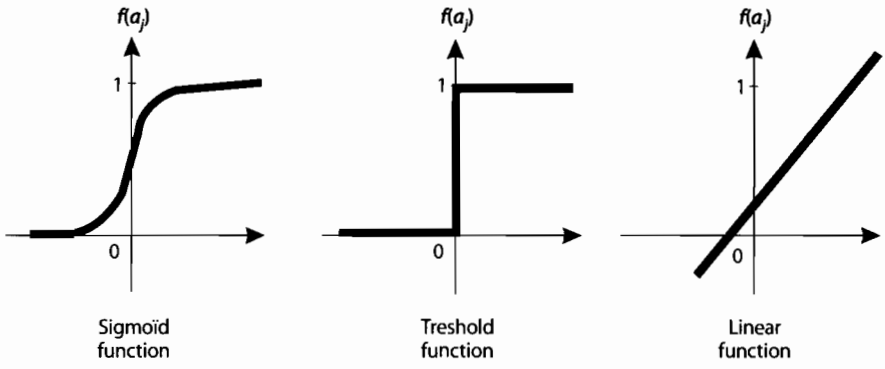


**Fig. 1.4.** Three types of transfer functions

### 1.2.2.2 *Backward-Propagating Step*

The backward-propagating step begins with the comparison of the network output pattern to the target value, when the difference (or error) is calculated. The backward-propagating step then calculates error values and changes the incoming weights, starting with the output layer and moving backward through the successive hidden layers.

The error signal ($\delta$) associated with each processing unit indicates the amount of error associated with that unit. This parameter is used during the weight-correction procedure, while learning is taking place. A large value for $\delta$ indicates that a large correction should be made to the incoming weights; its sign reflects the direction in which the weights should be changed.

If the output layer is designated by $k$, then its error signal is:

$$\delta_k = (t_k - x_k) f'(a_k) \tag{1.4}$$

with $t_k$: the target value of unit $k$, $x_k$: the output value for unit $k$, $f'$: the derivative of the sigmoid function, $a_k$: the weighted sum of the input to $k$, and $(t_k - x_k)$: the amount of error. (The $f'$ part of the term forces a stronger correction when the sum $a_k$ is near the rapid rise in the sigmoid curve.)

For the hidden layer ($j$), the error signal is computed as:

$$\delta_j = \left[\sum_k \delta_k w_{kj}\right] f'(a_j) \tag{1.5}$$

The adjustment of the connection weights is done using the $\delta$ values of the processing unit. Each weight is adjusted by taking into account the $\delta$ value of the unit that receives input from that interconnection. The connection weight is adjusted as follows:

$$\Delta w_{kj} = \eta \delta_k x_j \tag{1.6}$$

The adjustment of weight $w_{kj}$, which goes to unit $k$ from unit $j$, depends on three factors: $\delta_k$ (error value of the target unit), $x_j$ (output value for the source unit) and $\eta$. This weight adjustment equation is known as the generalized $\delta$ rule (Rumelhart et al. 1986). $\eta$ is a learning rate, commonly between 0 and 1, chosen by the user, and determines the rate of learning of the network. A very large value of $\eta$ can lead to instability in the network, and unsatisfactory learning. Too small values of $\eta$ can lead to excessively slow learning. Sometimes, the learning rate is varied to produce efficient learning of the network during the training procedure. For example, to obtain the best learning performance, the value of $\eta$ can be high at the beginning of the procedure, and decrease during the learning session.

### 1.2.3 Training the Network

Before training commences, the connection weights are set to small random values. Values between −0.3 and 0.3 are often used. Next the input patterns are applied to the network, which is allowed to run until an output is produced at each output node. The differences between the output calculations and the targets expected, taken over the entire set of patterns, are used to modify the weights. One complete calculation in the network is called an epoch or iteration of training or learning procedure. This process (epoch) is repeated until a suitable level of error is achieved. The BPN algorithm performs gradient descent on this error surface by modifying each weight in proportion to the gradient of the surface at its location (Fig. 1.5). It is known that gradient descent can sometimes cause networks to get stuck in a depression in the error surface where such a depression exists. These are called "local minima" which correspond to a partial solution for the network in response to the training data. Ideally, we seek a global minimum (lowest error value possible); nevertheless, the local minima are surrounded and the network usually does not leave it by the standard BPN algorithm. Special techniques should be used to get out of a local minimum: changing the learning parameter or the number of hidden units, but notably by the use of momentum term ($\alpha$) in the algorithm. The momentum term is chosen generally between 0 and 1. Taking this last term into account, the formula for weight modification at epoch $t + 1$ is given by:

$$\Delta w_{kj}(t+1) = \eta \delta_k(t+1)\, x_k(t+1) + \alpha \Delta w_{kj}(t) \tag{1.7}$$

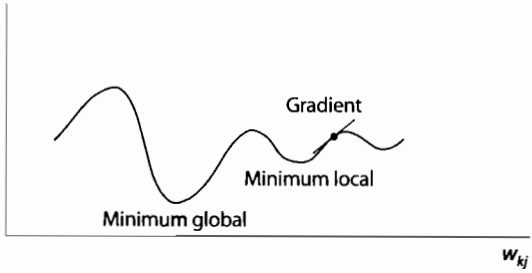


**Fig. 1.5.** Error surface as function of a weight showing gradient and local and global minima

The learning rate ($\eta$) and the momentum term ($\alpha$) play important roles in the learning process of BPN. If the values of these parameters are wrong, the network can oscillate, or more seriously it can get stuck in a local minimum. In our example (see Section 1.2.7), we obtained good convergence of the networks by initially making $\alpha = 0.7$ and $\eta = 0.01$; then they were modified according to the size of the error by the following algorithm:

```
If present_error > previous_error * 1.04
then η = η * 0.75,
     α = 0,
else  η = η * 1.05,
     α = 0.95,
EndIf
```

A training set must have enough examples of data to be representative for the overall problem. However, the training phase can be time-consuming depending on the network structure (number of input and output variables, number of hidden layers and number of nodes in each of them), the number of examples in the training set and the number of iterations.

### 1.2.4 Testing the Network

Typically the use of a BPN requires both training and test sets. Both sets contain input/output pattern pairs taken from real data. The first is used to train the network, and the second one serves to assess the performance of the network after training is complete. In the testing phase, the input patterns are fed into the network and the desired output patterns compared with those given by the neuronal network. The agreement or the disagreement of these two sets gives an indication of the performance of the neuronal network model.

Another decision that has to be made is the subdivision of the data set into different subsets which are used for training and testing the BPN. The best solution is to have separate data bases, to be able use the first set for training and testing the model, and the second independent set for validation of the model (Mastrorillo et al. 1998). This situation is rarely observed in ecological studies, and partitioning the data set may be applied for testing the validity of the model. We present here two partitioning procedures: *(i)* if enough data sets are available, the data may be divided randomly in two parts to give a training and a test set. The proportion may be 1 : 1, 2 : 1, 3 : 1, etc. for the two sets. However, the training set still has to be large enough to be representative of the problem and the test set has to be large enough to allow correct validation of the network. This procedure of partitioning the data is called k-fold cross-validation, sometimes named the hold-out procedure (Utans and Moody 1991; Efron and Tibshirani 1995; Kohavi and Wolpert 1996; Friedman 1997); *(ii)* if there are not enough examples available to permit the data set to be split into a representative training and test set, other strategies may be used, like cross-validation. In this case, the data set is divided into $n$ parts, usually small, i.e. containing few examples of data. The BPN may now be trained with $n - 1$ parts, and tested with the last part. The same network struc-

ture may be repeated to use every test set once in one of the $n$ procedures. The result of these tests together provide the performance of the model. Sometimes, in extreme cases, the test set can have only one example, and this is called the leave-one-out procedure (Efron 1983; Kohavi 1995). The case is often used in ecology when either we have a small database or each observation is a unique piece of information different from the others.

### 1.2.5 Overtraining or Overfitting the Network

If a network is overfitted (or overtrained), it has a good memory in the detail of data. In such cases, the network will not learn the general features inherently present in the training set, but it will perfectly learn more and more of the specific details of the training set. Thus the network loses its capacity to generalize. Several rules have been developed by many researchers regarding approximate determinations of the required network parameters to avoid overfitting. Three parameters are responsible for this phenomenon: the number of epochs, the number of hidden layers and the number of neurons in each hidden layer. The determination of the appropriate numbers of these elements is the most crucial parameter in BPN modelling. Previously, the optimum size of epochs or hidden layers or hidden nodes were determined by trial and error using training and test sets of data. A typical graph of training and generalization errors versus number of parameters is shown in Fig. 1.6. We can show the errors decrease rapidly as function of parameter complexity. If the error in the training set decreases steadily, the error of the test set can increase after minimal values, i.e. the model is no longer able to generalize. The training procedure must be stopped when the error on the test set is lowest, i.e. the zone corresponding to the best compromise between bias

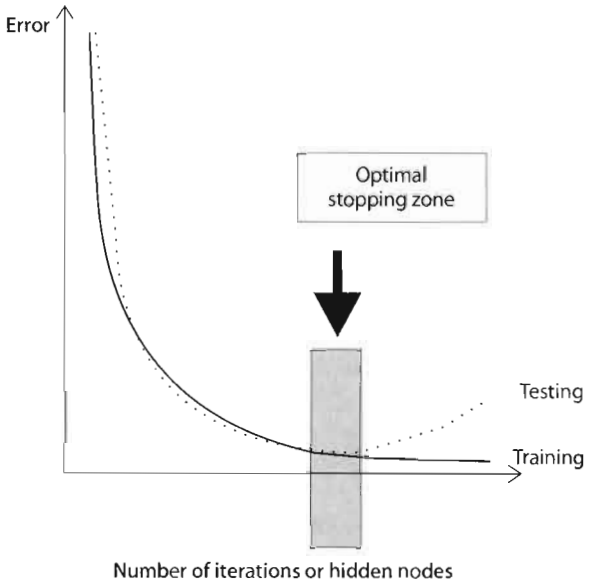


**Fig. 1.6.** Criteria of determination of training stop and selection the optimum network architecture

and variance. For an excellent summary of the issues affecting generalization in neuronal networks see Geman et al. (1992).

### 1.2.6 Use Aspects

In ANN modelling, as mentioned above, many parameters are difficult to grasp and their understanding by the model is often based to some extent on heuristics. We propose to now illustrate this situation through an example taken from fish ecology, prediction of the food consumption by fish relative to their biomass (Q/B) (Palomares and Pauly 1989; Palomares 1991; Lek et al. 1995). Palomares (1991) made a census of and standardized 108 direct evaluations of Q/B involving 65 species and 25 families of fish throughout the world (see data in Appendix). Using the multiple linear regression (MLR) model, we can explain 51% of the variance of Q/B after log transformation of some of the variables:

$$\log(Q/B) = 0.372 - 0.205\log(W_\infty) + 0.936\log(T) + 0.209\log(A) + 0.529h + 0.425d - 0.019p - 0.165D - 0.477P$$

This model is built with 8 independent variables: the asymptotic weight of the species ($W_\infty$), the morphological ratio $A$ representing the motor activity of the fish, the mean annual temperature ($T$), three discrete variables defining the diet, herbivorous ($h = 1$), detritivorous ($d = 1$), farmed fish ($p = 1$) and carnivorous ($h = d = p = 0$), and two morphological measurements: $D$ = standard length / height of the body and $P$ = height of the tail / height of the body.

As we can see (Appendix), the variables have different ranges of values. For example, $W_\infty$ has relatively high values and it might dominate or paralyse the model. Scaling of the input variables is then necessary. Different methods may be used, but the best results are often obtained by autoscaling, i.e. centred and reduced variable by this formula:

$$z = \frac{X - \bar{x}}{\sigma_x} \tag{1.8}$$

where $z$ is the scaled value, $X$ is the unscaled value, $\bar{x}$ and $\sigma_x$ are the mean and standard deviation for the specific variable. The effect of the autoscaling of the variables in the data set is shown in Fig. 1.7.

The output variable(s) also need to be scaled according to the transfer function used. If the sigmoid function is used, the range of variables must be scaled into the interval [0, 1], as given by the formula:

$$z = \frac{x - \min}{\max - \min}(\text{high} - \text{low}) + \text{low} \tag{1.9}$$

where $z$ is the scaled value, $x$ the unscaled value, min and max: the minimum and maximum of the unscaled values for the variable, high and low: lower bound and up-

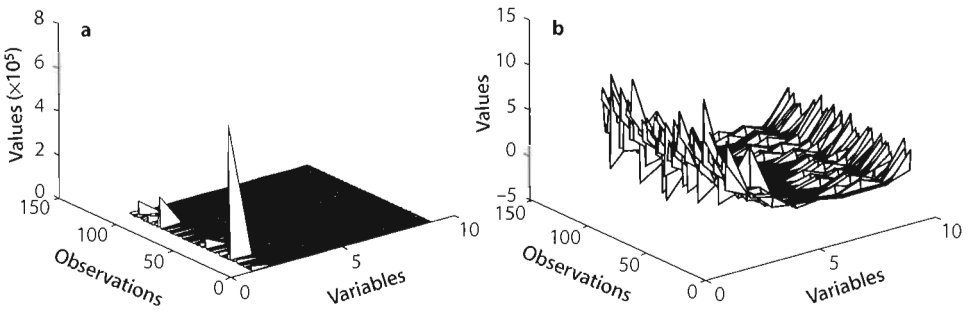


**Fig. 1.7.** Effect of autoscaling on a data set. The orders of variables (1 to 8) are the same in the table in Appendix; **a** before scaling, only one variable is dominant ($W_\infty$); **b** after scaling, all variables have more or less the same reach

**Table 1.1.** Values of the synaptic weights linking different independent variables to the dependent variable (Q/B) by using a linear function as the transfer function. The experimentation is repeated 5 times (±*SD*: Standard deviation)

| Experiment | $W_\infty$ | *T* | *A* | *d* | *p* | *h* | *P* | *D* |
|---|---|---|---|---|---|---|---|---|
| 1 | −0.0131 | 0.0481 | −0.0172 | 0.0069 | −0.0056 | 0.0540 | 0.0157 | 0.0613 |
| 2 | −0.0096 | 0.0452 | −0.0133 | 0.0045 | −0.0077 | 0.0538 | 0.0128 | 0.0611 |
| 3 | −0.0138 | 0.0487 | −0.0179 | 0.0074 | −0.0052 | 0.0540 | 0.0163 | 0.0614 |
| 4 | −0.0040 | 0.0405 | −0.0073 | 0.0007 | −0.0110 | 0.0536 | 0.0081 | 0.0606 |
| 5 | −0.0039 | 0.0405 | −0.0073 | 0.0006 | −0.0110 | 0.0536 | 0.0081 | 0.0606 |
| Mean | −0.0089 | 0.0446 | −0.0126 | 0.0040 | −0.0081 | 0.0538 | 0.0122 | 0.0610 |
| ±*SD* | 0.0048 | 0.0040 | 0.0051 | 0.0033 | 0.0028 | 0.0002 | 0.0040 | 0.0004 |

per bound of the range over which the data is scaled. Commonly, the bounds used were between 0 and 1, or between 0.1 and 0.9. The latter choice can often speed the rate of convergence.

### 1.2.7 BPN versus MLR

Our purpose in this section is to compare BPN with MLR, which are largely used in ecological modelling. Other methods are more scarcely applied to ecology and related fields (e.g. General additive models (GAM), Alternating Conditional Expectations (ACE), etc.), but we chose not to include them in this comparative analysis. So, we used the simplest neuronal network and shall now try to see whether any common points can be found with multiple regression.

First, we used a network with no hidden layers and a linear transfer function. To avoid overfitting, the learning procedure was stopped at 1 000 epochs. We repeated this operation 5 times, with different synaptic weights randomly choosing them between

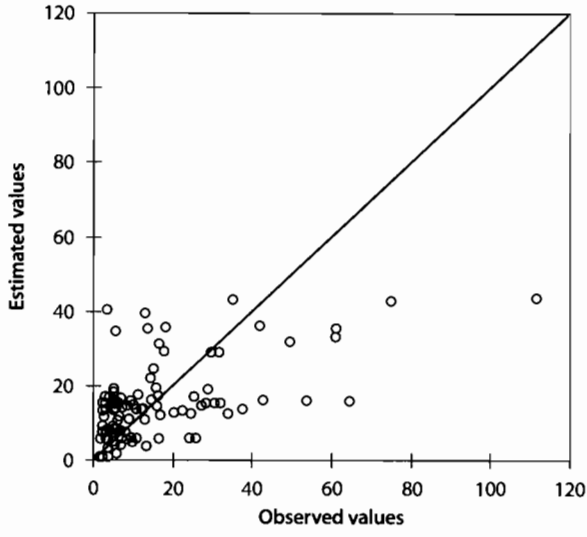


**Fig. 1.8.** Performance of back propagation neuronal network without hidden layer after 1 000 epochs of learning procedure, by using linear function as transfer function

−0.3 and 0.3. In spite of the different starting weights, the final synaptic weights were very close (very low standard deviation), in value and sign (Table 1.1).

Taking into account the mean of the 5 experiments, we can obtain the following model:

$$Q / B = f(-0.0089W_{\infty} + 0.0446T - 0.0126A + 0.0040d - 0.0081p + 0.0538h + 0.0122P + 0.0610D) \quad (1.10)$$

This equation allows the value of Q/B to be computed with the values of other parameters with $f$ as linear function, in the same way as the MLR model. The correlation coefficient between observed and estimated values is $r = 0.57$ (Fig. 1.8), i.e. the same value obtained by the MLR model performed without transformation of variables.

**Table 1.2.** Values of the synaptic weights linking different independent variables to the dependent variable (Q/B) by using a sigmoid function as the transfer function. The experimentation is repeated 5 times (*SD*: Standard deviation)

| Experiment | $W_{\infty}$ | *T* | *A* | *d* | *p* | *H* | *P* | *D* |
|---|---|---|---|---|---|---|---|---|
| 1 | −6.3935 | 0.4529 | −0.0689 | 0.1912 | −0.2671 | 0.3030 | 0.0364 | 0.4671 |
| 2 | −6.7578 | 0.4442 | −0.0561 | 0.1817 | −0.2693 | 0.3091 | 0.0438 | 0.4182 |
| 3 | −6.7647 | 0.4442 | −0.0561 | 0.1818 | −0.2693 | 0.3090 | 0.0437 | 0.4181 |
| 4 | −6.3932 | 0.4529 | −0.0689 | 0.1912 | −0.2671 | 0.3030 | 0.0364 | 0.4671 |
| 5 | −6.6569 | 0.4425 | −0.057 | 0.1806 | −0.2718 | 0.3114 | 0.0460 | 0.4068 |
| Mean | −6.5932 | 0.4473 | −0.0614 | 0.1853 | −0.2689 | 0.3071 | 0.0413 | 0.4355 |
| ±*SD* | 0.1874 | 0.0051 | 0.0069 | 0.0054 | 0.0019 | 0.0039 | 0.0045 | 0.0293 |

In a second experiment, by applying the sigmoid function $f$, we obtained an improvement of the model ($r = 0.67$), given by the following equation (see Table 11.2 for coefficients):

$$Q / B = f(-6.5932W_{\infty} + 0.4473T - 0.0614A + 0.1853d - 0.2689p + 0.3071h + 0.0413P + 0.4355D) \quad (1.11)$$

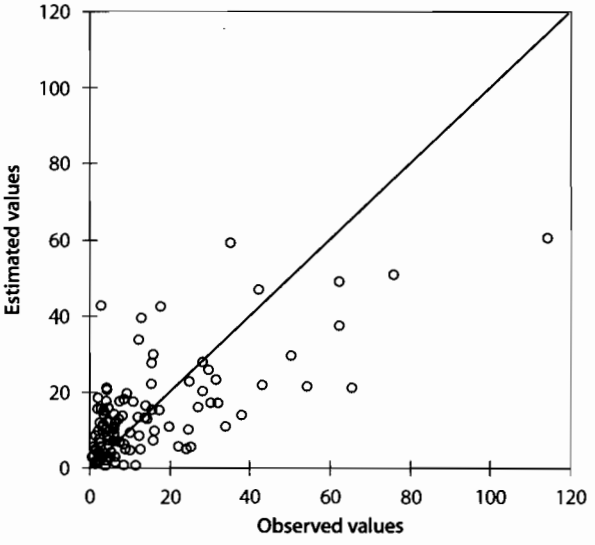


**Fig. 1.9.** Performance of back propagation neuronal network without hidden layer after 1000 epochs of learning procedure, by using sigmoid function as transfer function

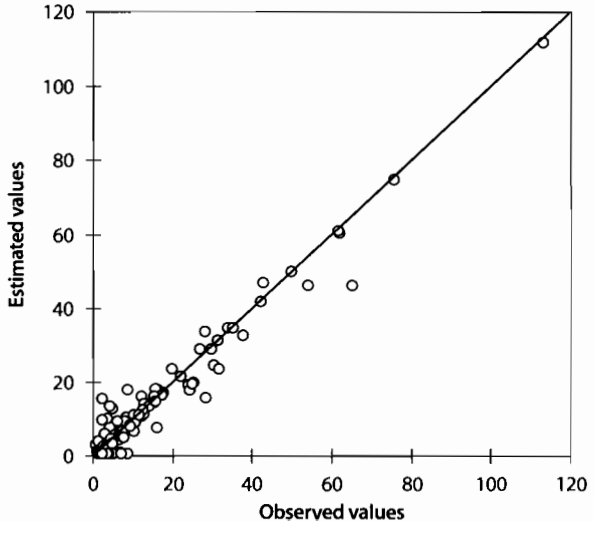


**Fig. 1.10.** Performance of back propagation neuronal network with 8 neurons in hidden layer, after 1000 epochs of learning procedure, by using sigmoid function as transfer function

The predictive quality is better, but the errors are high compared to the perfect line of prediction (Fig. 1.9). The synaptic weights are high and constant for $W_{\infty}$ variable.

Using one hidden layer, we improved the quality of prediction (Fig. 1.10): practically all observations are aligned on the perfect line (coordinate 1 : 1). For more details, see Lek et al. (1995).

## 1.3 Kohonen Self-Organizing Mapping (SOM)

### 1.3.1 Algorithm

The Kohonen SOM falls into the category of unsupervised competitive learning (Fig. 1.11) methodology, in which the relevant multivariate algorithms seek clusters in the data (Everitt 1993). Conventionally, at least in ecology, the reduction of the multivariate data is usually carried out using principal components analysis or hierarchical clustering analysis (Jongman et al. 1995). Unsupervised learning allows the investigator to group objects together on the basis of their perceived closeness in $n$-dimensional hyperspace (where $n$ is the number of variables or observations made on each object).

Formally, a Kohonen network consists of two types of units: an input layer and an output layer. The array of input units operates simply as a flow-through layer for the input vectors and has no further significance. In the output layer, SOM often consists of a two-dimensional network of neurons arranged on a square (or other geometrical form) grid laid out in a lattice. A hexagonal lattice is preferred, because it does not favour horizontal or vertical directions. Each neuron is connected to its nearest neighbours on the grid (Fig. 1.12). The neurons store a set of weights (weight vector), an $n$-dimensional vector if input data are $n$-dimensional.

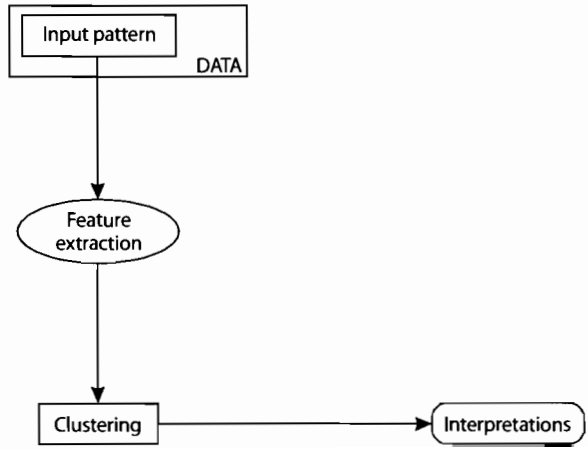


**Fig. 1.11.** Diagram showing how data are used to establish the model calibration in the unsupervised learning procedure. The goal of the unsupervised learning is to obtain a cluster or mapping in order that people can easily explain the data

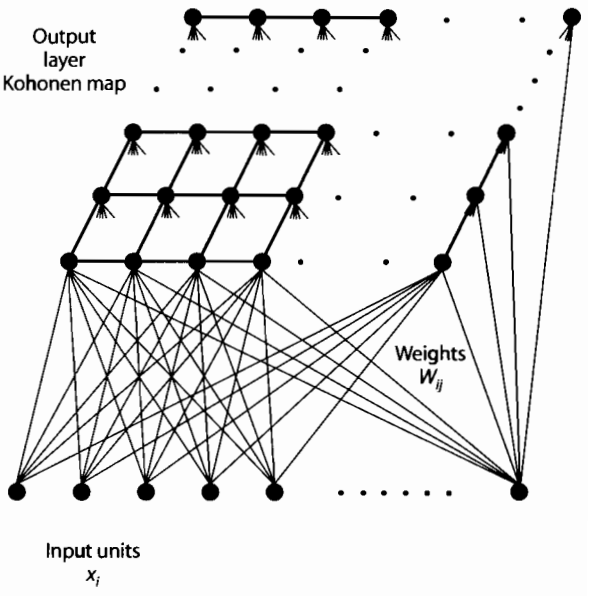


**Fig. 1.12.** A two dimensional Kohonen self-organizing feature map network

Since the introduction of the Kohonen neuronal network (Kohonen 1982), several training strategies have been proposed (see e.g. Hecht-Nielsen 1990; Freeman and Skapura 1992) which deal with different aspects of use of the Kohonen network. In this section, we will keep to the neuronal network proposed by Kohonen (1984). For an input $x$, each neuron $j$ (weights: $w^j$) calculates its activation level, defined as:

$$\left\| w^j - x \right\| = \sqrt{\sum_{i=0}^{n} \left( w_i^j - x_i \right)^2} \tag{1.12}$$

Thus, this is simply the Euclidean distance between the points represented by the weight vector and the input in $n$-dimensional space. A node whose weight vector closely matches the input vector will have a small activation level, and a node whose weight vector is very different from the input vector will have a large activation level. The node in the network with the smallest activation level is deemed to be the winner for the current input vector.

During the training process the network is presented with each input pattern, and all the nodes calculate their activation levels as described above. The winning node and some of the node around it are then allowed to adjust their weight vectors to match the current input vector more closely. The nodes included in the set, which are allowed to adjust their weights, are said to belong to the neighbourhood of the winner. The size of the winner's neighbourhood is decreased linearly after each presentation of the complete training set (all available data being analysed), until it includes only the winner itself. The amount by which the nodes in the neighbourhood are allowed to adjust their weights is also reduced linearly through the training period.

The factor which governs the size of the weight variations is known as the learning rate. The adjustments to each item in the weight vector are made in accordance with:

$$\delta w_i = -\alpha(w_i - x_i) \tag{1.13}$$

where $\alpha$ is the learning rate, $\delta w$: the change in the weight. This is carried out for $i = 1$ to $i = n$, the dimension of the data. The learning is decomposed into two phases. During the first one (ordering phase), $\alpha$ shrinks linearly from 1 to the final value 0 and the neighbourhood radius decreases in order to initially contain the whole map and finally only the nearest neighbours of the winner. During the second phase, tuning takes place: $\alpha$ attains small values (for example 0.02) during a long period and the neighbourhood radius keeps the value 1.

The effect of the weight updating algorithm is to distribute the neurons evenly throughout the region of $n$-dimensional space populated by the training set (Kohonen 1984; Hecht-Nielsen 1990). This effect is displayed and shows the distribution of a square network over an evenly populated two-dimensional square input space, and a more complex input space. The neuron with the weight vector closest to a given input pattern will win for that pattern and for any other input patterns that it is closest to. Input patterns which allow the same node to win are then judged to be in the same cell, and when a map of their relationships is drawn, a line encloses them. By training with networks of increasing size, a map with several levels of groups or contours can be drawn. These contours, however, may sometimes cross, which appears to be due to a failure of the SOM to converge to an even distribution of the neuron over the input space (Erwin et al. 1992). Construction of these maps allows close examination of the relationships between the items in the training set.

### 1.3.2 Missing Data

In ecology, a difficult problem arises from missing data. For some data items, certain components of the data vectors are unknown. SOM accepts the fact that data may be missing and two approaches can be used. Firstly, data items with too many missing components are discarded during the learning process and are then mapped on the organizing map (Kaski and Kohonen 1996). Secondly, if only few components of a data vector are missing, a convenient solution consists in only using available components in Eqs. 1.12 and 1.13 (Samad and Harp 1992).

### 1.3.3 Outliers

Due to measurement errors or to values really different from the rest, outliers are often present in ecological data. Classical methods of clustering are sensitive to the presence of outliers in the data. And generally, it is useful to detect outliers before computing clusters. With SOM, the process is quite different. Each outlier takes its place in one unit of the map, and only the weights of that neuron and its nearest neighbours

are affected. There is no effect on the other neurons. Moreover, the observation of scattered data in an area of the map should suggest the presence of an outlier.

### 1.3.4
### Use of Different Metrics

Measuring ecological likeness often leads to the use of various similarity or distance coefficients. For example, the presence/absence measure or the computation of genetic data calls for the choice of distances other than the classical Euclidean distance. Describing community structure (for plants or animals) often leads the ecologist to use various distance measures between sample units (Ludwig and Reynolds 1988). These problems can be solved with SOM but it is necessary to pay attention. Not only should the activation level of each neuron (Eq. 1.12) be computed with the appropriate distance, but also a compatible metrics have to be used in the adjustment of the weights (Eq. 1.13) (Kohonen 1995).

### 1.3.5
### Aspects of Use

The iris data published by Fisher (1936) have been widely used for examples in discriminant and cluster analysis. Sepal length, sepal width, petal length and petal width were measured in millimetres on 50 flower specimens from each of three species, *Iris setosa*, *I. virsicolor*, and *I. virginica*. The graphical representation of the two first PCA axes (Fig. 1.13) shows complete separation of the first class (*I. setosa*), and the two other classes are very close to each other. Using discriminant analysis, we obtain 98% of good

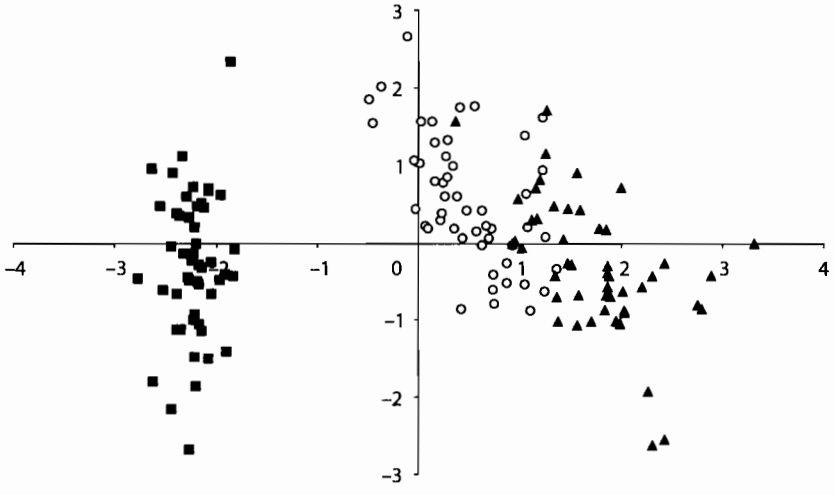


**Fig. 1.13.** Iris data projected linearly onto the two-dimensional subspace obtained with PCA. Variety: *I. setosa*, ■; *I. versicolor*, ○; *I. virginica*, ▲

classification (i.e. 3 misclassified observations, 2 *I. virsicolor* and 1 *I. virginia* specimens).

This data set was used to illustrate the SOM method. A Kohonen map with $8 \times 10$ neurons in a hexagonal lattice is trained with the 150 observations presented randomly and iteratively. The components of the data items were scaled to mean 0 and variance 1. The different varieties of iris, *I. setosa, I. versicola, I. virginica,* are not used during the learning (unsupervised learning). 2 000 iterations are made during the ordering phase then 40 000 iterations during the tuning phase. At the end, individuals are set in the appropriate unit of the SOM (Fig. 1.14). SOM allows a first clustering: individuals are present in only 61 hexagons. It is worth noting that only two hexagons contain irises of different species and *I. setosa* are present in the left lower part of the map, *I. versicolor* in the middle part and *I. virginica* in the right lower part.

Then, it is necessary to represent the relative distances between their neighbouring units. A scale of shades of grey is used (Iivarinen et al. 1994). Light shades indicate small distances and dark shades, large distances between two neighbouring hexagons. In that way, a "cluster landscape" is formed and clusters can be seen better (Fig. 1.15). Three plains appear (light areas) separated by hills or mountains (dark areas): *I. setosa* individuals residing mainly in the left lower plain, *I. versicolor* in the right upper plain and some *I. virginica* in a little plain area in the middle of the right side. The mountainous area from the upper left to the lower right part of the map mainly groups *I. versicolor* and *I. virginica.*

Another interesting representation with SOM is the distribution of each variable on the map (Fig. 1.16). SOM is coloured for each component of weight vectors, namely

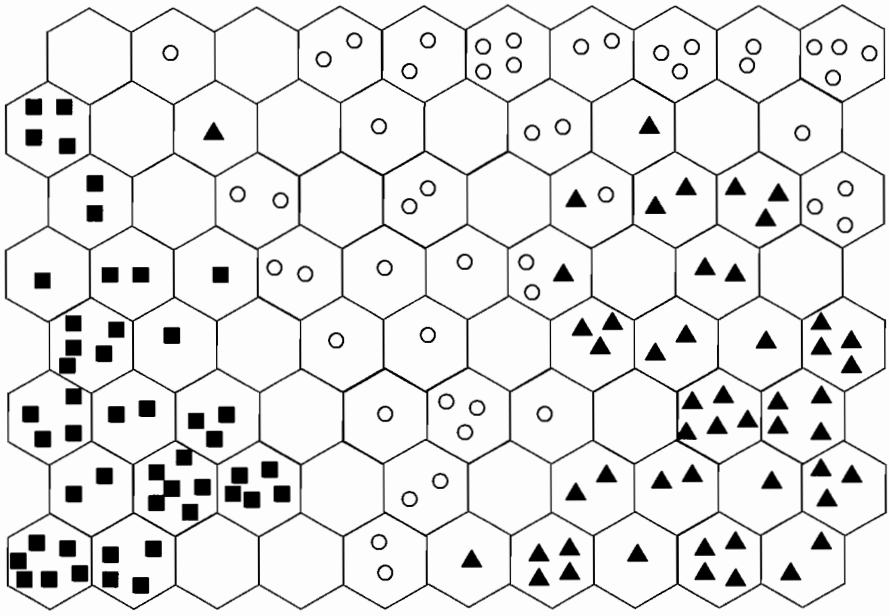

**Fig. 1.14.** Iris data mapped on the organizing map. Variety: *I. setosa,* ■; *I. versicolor,* ○; *I. virginica,* ▲

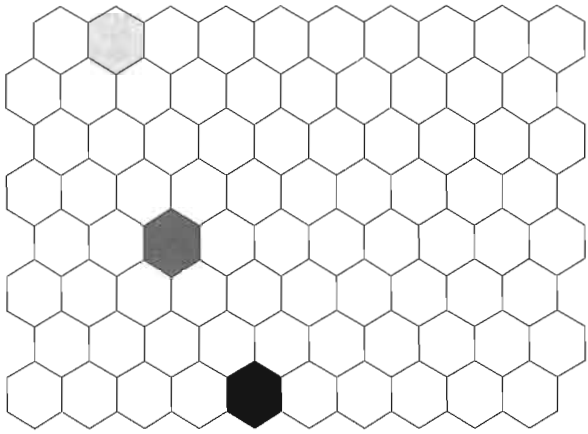

**Fig. 1.15.** Self-organizing map for iris data with shades of grey indicating the degree of clustering

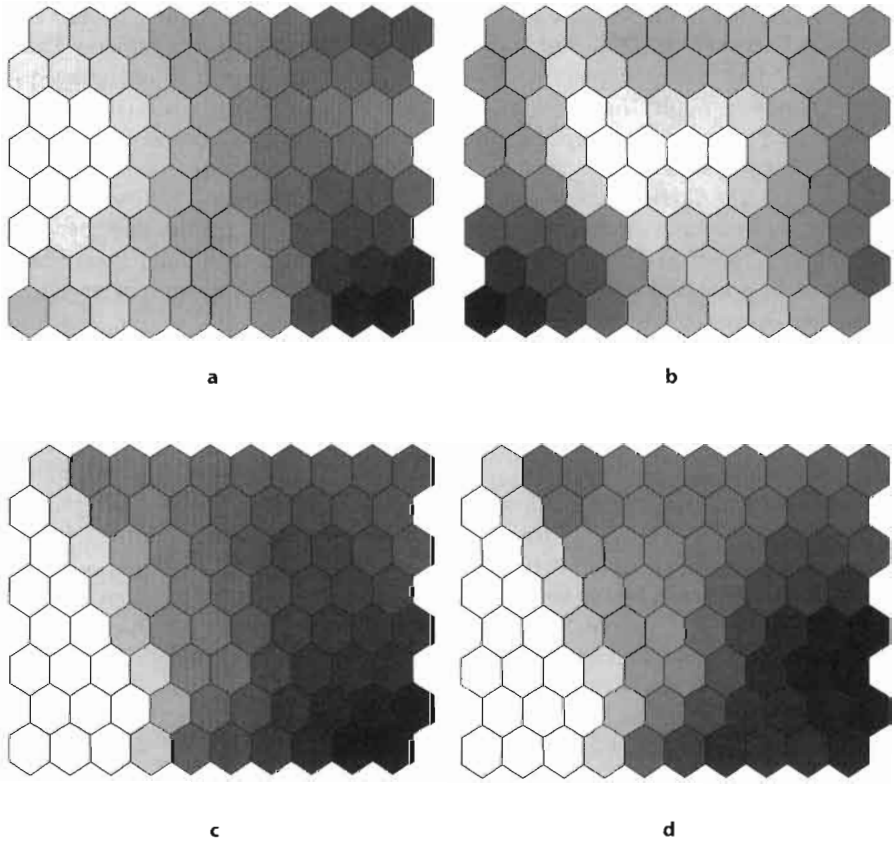


**Fig. 1.16.** Representation of the components of the weight vectors for each neuron; **a** sepal length; **b** sepal width; **c** petal length; **d** petal width. In each map, white colour indicates the smallest value and black colour the largest ones

sepal length, sepal width, petal length, petal width. In each display, two hexagons with similar grey level contain individuals of the same kind for this variable. For example, with sepal length, *I. setosa* individuals are in bright area (small values) and *I. virginica* individuals in dark area (large values). With sepal width, bright areas correspond to *I. versicola* individuals and dark areas to *I. setosa*. Then, well known characteristics of the different species are visualised on the map which can be useful for interpretation.

## 1.4 Conclusion

In this introductory part, we have highlighted the potential use of artificial neuronal networks in ecology. Using known examples (iris data or Q/B ratios, in annexe) we showed that one can obtain better results with the ANN. During the last two decades of the current century, the growing development of computer-aided analysis, which is easily accessible to all researchers, has facilitated the use of ANNs in ecological modelling. To apply an ANN program, ecologists can obtain freeware or shareware from various web sites in the world. Users interested can find these programs by filling in "neuronal network" as a keyword in the search procedure of the web explorer. Thus, they can obtain many computer ANN programs functioning with all operating systems (Windows, Apple, Unix stations, etc.). Moreover, increasingly specialized ANN packages are proposed at acceptable prices for personal computers and most professional statistical software now includes proposes ANN procedures (e.g. SAS, S-Plus, Matlab, etc.). The development of computers and ANN software must allow ecologists to apply ANN methods more easily to resolve the complexity of relationships between variables in ecological data. In the following chapters, readers will find papers which illustrate the ecological application of ANNs in several fields, ranging from terrestrial to aquatic ecosystems, remote sensing and evolutionary ecology.

## Acknowledgements

We thank the following referees for their help in accepting to review the different papers: Ambroise, Arino, Auger, Aussem, Balls, Beacham, Belaud, Bourret, Canu, Capblancq, Careaux, Cereghino, Charles, Chau, Chon, Cisneros, Comrie, Culverhouse, Dimopoulos, Dreyfus, Dubois, Ehrman, Flanagan, Foody, Fuhs, Gan, Gascuel, Gedeon, Grandjean, Halls, Hanser, Huntingford, Jarre-Teichman, Kapetsky, Kaski, Kimes, Kok, Komatsu, Kropp, Lewis, Manel, Marzban, Masson, Megrey, Melssen, Morimoto, Morlini, Murase, Oberdorff, O'Connor, Park, Prasher, Puig, Sanchez Manes, Recknagel, Roush, Scardi, Schultz, Shamseldin, Starret, Teriokhin, Thibault, Thiria, Tomasel, Touzet, Vila, Wang, Werner, William, Wong, Yang.

## References

Ackley DH, Hinton GE, Sejnowski TJ (1985) A learning algorithm for Boltzmann machines. Cognitive science 9:147–169

Albiol J, Campmajo C, Casas C, Poch M (1995) Biomass estimation in plant cell cultures: A neuronal network approach. Biotechnology Progress 11:88–92

Baran P, Lek S, Delacoste M, Belaud A (1996) Stochastic models that predict trout population densities or biomass on macrohabitat scale. Hydrobiologia 337:1–9

Bishop MC (1995) Neuronal networks for pattern recognition. Clarendon Press, Oxford
Bradshaw JA Carden KJ, Riordan D (1991) Ecological applications using a Novel expert system shell. Computer Applications in the Biosciences 7:79–83
Brey T, Jarre-Teichmann A, Borlich O (1996) Artificial neuronal network versus multiple linear regression: Predicting P/B ratios from empirical data. Mar Ecol Prog Ser 140:251–256
Chu WC, Bose NK (1998) speech signal prediction using feedforward neuronal-network. Electronics Letters 34:999–1001
Colasanti RL (1991) Discussions of the possible use of neuronal network algorithms in ecological modelling. Binary 3:13–15
Cosatto E, Graf HP (1995) A neuronal-network accelerator for image-analysis. IEEE Micro 15:32–38
Cybenko G (1989) Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals and Systems 2:304–314
d'Angelo DJ, Howard LM, Meyer JL, Gregory SV, Ashkenas LR (1995) Ecological use for genetic algorithms: Predicting fish distributions in complex physical habitats. Can J Fish Aquat Sc 52:1893–1908
Dekruger D, Hunt BR (1994) image-processing and neuronal networks for recognition of cartographic area features. Pattern Recognition 27:461–483
Edwards M, Morse DR (1995) The potential for computer-aided identification in biodiversity research. Trends in Ecology and Evolution 10:153–158
Efron B (1983) Estimating the error rate of a prediction rule: Improvement on cross-validation. J Am Statistical Association 78(382):316–330
Efron B, Tibshirani RJ (1995) Cross-validation and the bootstrap: Estimating the error rate of the prediction rule. Rep Tech Univ Toronto
Erwin E, Obermayer K, Schulten K (1992) Self-organizing maps: Ordering, convergence properties and energy functions. Biol Cyb 67(1):47–55
Everitt BS (1993) Cluster analysis. Edward Arnold, London
Faraggi D, Simon R (1995) A neuronal network model for survival data. Statistics in Medicine 14:73–82
Fisher RA (1936) The use of multiple measurements in axonomic problems. Annals of Eugenics 7:179–188
Freeman JA, Skapura DM (1992) Neuronal networks: Algorithms, applications and programming techniques. Addison-Wesley Publishing Company, Reading
Friedman JH (1997) On bias, variance, o/1-loss and the curse-of-dimensionality. Data Mining and Knowledge Discovery 1:55–77
Gallant SI. (1993) Neuronal network learning and expert systems. The MIT Press, Cambridge
Geman S, Bienenstock E, Doursat R (1992) Neuronal networks and the bias/variance dilemma. Neuronal Computation 4:1–58
Giske J, Huse G, Fiksen O (1998) Modelling spatial dynamics of fish. Rev. Fish. Biol. Fish. 8:57–91
Golikov SY, Sakuramoto K, Kitahara T, Harada Y (1995) Length-frequency analysis using the genetic algorithms. Fisheries Science 61:37–42
Guégan JF, Lek S, Oberdorff T (1998) Energy availability and habitat heterogeneity predict global riverine fish diversity. Nature 391:382–384
Hecht-Nielsen R (1990) Neurocomputing. Addison-Wesley, Massachusetts
Hopfield JJ (1982) Neuronal networks and physical systems with emergent collective computational abilities. Proc Natl Acad Sci USA 79:2554–2558
Hornick K, Stinchcombe M, White H (1989) Multilayer feedforward networks are universal approximators. Neuronal Networks 2:359–366
Iivarinen J, Kohonen T, Kankas J, Kaski S (1994) Visualizing the clusters on the self-organizing map. In: Carlsson C, Järvi T, Reponen T (eds) Proceedings of the Conference on Artificial Intelligence Research in Finland. Finnish Artificial Intelligence Society, Helsinki, pp 122–126
Jongman RHG, ter Braak CJF, van Tongeren OFR (1995) Data analysis in community and landscape ecology. Cambridge University Press, Cambridge
Kaski S, Kohonen T (1996) Exploratory data analysis by the self-organzing map: Structures of welfare and poverty in the world. In: Refenes A-PN, Abu-Mostafa Y, Moody J, Weigend A (eds) Neuronal Networks in Financial Engineering. Proceedings of the Third International Conference on Neuronal Networks in the Capital Markets, World Scientific, Singapore, pp 498–507
Kastens TL, Featherstone AM (1996) Feedforward backpropagation neuronal networks in prediction of farmer risk preference. American Journal of Agricultural Economics 78:400–415
Kohavi R (1995) A study of cross-validation and bootstrap for estimation and model selection. Proc. of the 14th Int. Joint Conf. on Artificial Intelligence, Morgan Kaufmann Publishers Inc., Bari
Kohavi R, Wolpert DH (1996) Bias plus variance decomposition for zero-one loss functions. In: Saitta L (ed) Machine learning. Proceedings of the Thirteenth International Conference, Morgan Kaufmann Publishers Inc., Bari, pp 275–283
Kohonen T (1982) Self-organized formation of topologically correct feature maps. Biol Cybern 43:59–69
Kohonen T (1984) Self-organization and associative memory. Springer-Verlag, Berlin

Kohonen T (1995) Self-organizing maps. Springer-Verlag Berlin
Kung SY, Taur JS (1995) Decision-based neuronal networks with signal image classification applications. IEEE Transactions on Neuronal Networks 6:170–181
Kvasnicka V (1990) An application of neuronal networks in chemistry. Chemical Papers 44(6):775–792
Lek S, Belaud A, Lauga J, Dimopoulos I, Moreau J (1995) An improved estimation of the animal food of fish populations using neuronal networks. Marine and Freshwater Research, 46(8):1229–1236
Lek S, Belaud A, Baran P, Dimopoulos I, Delacoste M (1996a) Role of some environmental variables in trout abundance models using neuronal networks. Aquatic Living Resources 9:23–29
Lek S, Delacoste M, Baran P, Dimopoulos I, Lauga J, Aulagnier S (1996b) Application of neuronal networks to modelling nonlinear relationships in ecology. Ecol Model 90:39–52
Lerner B, Levinstein M, Rosenberg B, Guterman H, Dinstein I, Romem Y (1994) Feature selection and chromosomes classification using a multilayer perceptron neuronal network. IEEE International conference on Neuronal Networks, Orlando (Florida), pp 3540–3545
Lo JY, Baker JA, Kornguth PJ, Floyd CE (1995) Application of artificial neuronal networks to interpretation of mammograms on the basis of the radiologists impression and optimized image features. Radiology 197:242
Ludwig J A, Reynolds JF (1988) Statistical ecology: A primer on methods and computing. John Wiley & Sons, New York
Mastrorillo S, Dauba F, Oberdorff T, Guégan JF, Lek S (1998) Predicting local fish species richness in the Garonne river basin. C R Acad Sci Paris, Life Sciences 321:423–428
Palomares ML (1991) La consommation de nourriture chez les poissons: étude comparative, mise au point d'un modèle prédictif et application à l'étude des relations trophiques. Ph D thesis, Institut National Polytechnique de Toulouse, Toulouse
Palomares ML, Pauly D (1989) A multiple regression model for predicting the food consumption of marine fish populations. Australian Journal of Marine and Freshwater Research 40:259–73
Parker DB (1982) Learning logic. Invention report S81–64, File 1, Office of Technology Licensing, Stanford University
Rahim MG, Goodyear CC, Kleijn WB, Schroeter J, Sondhi MM (1993) On the use of neuronal networks in articulatory speech synthesis. Journal of the Acoustical Society of America 93:1109–1121
Recknagel F, Petzoldt T, Jaeke O, Krusche F (1994) Hybrid expert-system delaqua – a toolkit for water-quality control of lakes and reservoirs. Ecol Model 71:17–36
Recknagel F, French M, Harkonen P, Yabunaka KI (1997) Artificial neuronal network approach for modelling and prediction of algal blooms. Ecol Model 96:11–28
Ripley BD (1996) Pattern recognition and neuronal networks. Cambridge University Press, Cambridge
Rosenblatt F (1958) The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review 65:386–408
Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propagating errors. Nature 323:533–536
Samad T, Harp SA (1992) Self-organizing with partial data. Network: Computation in Neuronal Systems 3:205–212
Scardi M (1996) Artificial neuronal networks as empirical models for estimating phytoplankton production. Mar Ecol Prog Ser 139:289–299
Seginer I, Boulard T, Bailey BJ (1994) Neuronal network models of the greenhouse climate. Journal of Agricultural Engineering Research 59:203–216
Smith M (1994) Neuronal networks for statistical modelling. Van Nostrand Reinhold, New York
Smits JRM, Breedveld LW, Derksen MWJ, Katerman G, Balfoort HW, Snoek J, Hofstraat JW (1992) Pattern classification with artificial neuronal networks: Classification of algae, based upon flow cytometer data. Analytica Chimica Acta 258:11–25
Utans J, Moody JE (1991) Selecting neuronal network architectures via the prediction risk: application to corporate bond rating prediction. Proceedings of the First International Conference on Artificial Intelligence Applications on Wall Street, IEEE Computer Society Press, Los Alamitos
Webos P (1974) Beyond regression: New tools for prediction and analysis in the behavioral sciences. Thesis, Harvard University
Widrow B, Hoff ME (1960) Adaptive switching circuits. IRE Wescon conference record 4:96–104
Wythoff BJ, Levine SP, Tomellini SA (1990) Spectral peak verification and recognition using a multilayered neuronal network. Analytical Chemistry 62(24):2702–2709

# Appendix

**Table A1.1.** 108 records of Q/B ratio data (for variable symbols, see text)

| *W* | *T* | *A* | *d* | *p* | *H* | *P* | *D* | Q/B |
|---|---|---|---|---|---|---|---|---|
| 63 | 20 | 2.32 | 0.334 | 0.271 | 0 | 0 | 0 | 8.23 |
| 362 | 25 | 1.41 | 0.35 | 0.267 | 0 | 0 | 0 | 8.1 |
| 1216 | 18 | 1.89 | 0.326 | 0.263 | 1 | 0 | 0 | 31.4 |
| 28 | 15 | 1.31 | 0.161 | 0.486 | 0 | 0 | 0 | 9.13 |
| 7500 | 13 | 2.4 | 0.261 | 0.34 | 0 | 0 | 0 | 6.49 |
| 10541 | 13 | 2.4 | 0.261 | 0.34 | 0 | 0 | 0 | 4.08 |
| 100 | 10 | 2.06 | 0.25 | 0.324 | 0 | 1 | 0 | 7.41 |
| 230 | 15.5 | 2.21 | 0.217 | 0.417 | 0 | 0 | 0 | 8.63 |
| 605 | 9 | 2.21 | 0.217 | 0.417 | 0 | 0 | 0 | 5.22 |
| 1824 | 16 | 2.21 | 0.217 | 0.417 | 0 | 0 | 0 | 23.9 |
| 1206 | 16 | 2.21 | 0.217 | 0.417 | 0 | 0 | 0 | 25.1 |
| 13312 | 15 | 1.5 | 0.182 | 0.429 | 0 | 0 | 0 | 2.03 |
| 10925 | 15 | 1.5 | 0.182 | 0.429 | 0 | 0 | 0 | 6.43 |
| 6049 | 7 | 1.5 | 0.182 | 0.429 | 0 | 0 | 0 | 2.95 |
| 8810 | 7 | 1.5 | 0.182 | 0.429 | 0 | 0 | 0 | 1.61 |
| 3288 | 7 | 1.5 | 0.182 | 0.429 | 0 | 0 | 0 | 0.581 |
| 2 | 25 | 1.03 | 0.216 | 0.409 | 0 | 0 | 0 | 33.9 |
| 1 | 25 | 0.78 | 0.183 | 0.425 | 0 | 0 | 0 | 24.3 |
| 2 | 25 | 0.75 | 0.211 | 0.524 | 0 | 0 | 0 | 12.1 |
| 8 | 25 | 1.02 | 0.238 | 0.326 | 0 | 0 | 0 | 37.6 |
| 10 | 25 | 0.83 | 0.188 | 0.444 | 0 | 0 | 0 | 16.2 |
| 11 | 25 | 0.81 | 0.235 | 0.354 | 0 | 0 | 0 | 12 |
| 13 | 25 | 1 | 0.238 | 0.28 | 0 | 0 | 0 | 26.9 |
| 4 | 25 | 0.93 | 0.187 | 0.4 | 0 | 0 | 0 | 19.9 |
| 250 | 25.8 | 2.52 | 0.232 | 0.437 | 0 | 0 | 0 | 5.93 |
| 32 | 15.4 | 2.13 | 0.443 | 0.3 | 0 | 0 | 0 | 2.5 |
| 32000 | 24.5 | 2.37 | 0.489 | 0.231 | 0 | 1 | 0 | 1.61 |
| 615 | 31.7 | 1.94 | 0.378 | 0.364 | 0 | 1 | 0 | 4.24 |
| 808 | 30.8 | 1.51 | 0.221 | 0.524 | 0 | 1 | 0 | 5.58 |
| 883 | 30.8 | 1.98 | 0.289 | 0.436 | 0 | 1 | 0 | 6.26 |
| 60 | 14 | 1.41 | 0.273 | 0.39 | 0 | 0 | 0 | 15.9 |
| 316 | 9 | 1.49 | 0.303 | 0.357 | 1 | 0 | 0 | 13.6 |
| 766 | 14 | 1.49 | 0.303 | 0.357 | 0 | 1 | 0 | 2.68 |
| 1769 | 12.4 | 1.49 | 0.303 | 0.357 | 1 | 0 | 0 | 14.5 |
| 710 | 12.4 | 1.34 | 0.256 | 0.456 | 0 | 0 | 0 | 12.7 |
| 6650 | 25.5 | 1.44 | 0.247 | 0.352 | 0 | 0 | 0 | 22.1 |
| 21887 | 25 | 1.26 | 0.161 | 0.353 | 0 | 0 | 0 | 1.32 |

**Table A1.1.** *Continued*

| W | T | A | d | p | H | P | D | Q/B |
|---|---|---|---|---|---|---|---|---|
| 6074 | 22.5 | 1.26 | 0.161 | 0.353 | 0 | 0 | 0 | 1.73 |
| 1688 | 22.5 | 1.01 | 0.178 | 0.595 | 0 | 0 | 0 | 1.33 |
| 12356 | 10 | 0.77 | 0.181 | 0.25 | 0 | 0 | 0 | 2.59 |
| 15714 | 12 | 0.77 | 0.181 | 0.25 | 0 | 0 | 0 | 2.26 |
| 2 | 10 | 1.69 | 0.3 | 0.166 | 0 | 0 | 0 | 3.85 |
| 3776 | 13 | 1.78 | 0.361 | 0.331 | 0 | 0 | 0 | 1.1 |
| 16595 | 25 | 1.09 | 0.365 | 0.357 | 0 | 0 | 0 | 4.26 |
| 1006 | 26 | 2.4 | 0.214 | 0.306 | 0 | 0 | 0 | 4.31 |
| 3067 | 15 | 1.76 | 0.25 | 0.4 | 0 | 0 | 0 | 10.2 |
| 3067 | 15 | 1.76 | 0.25 | 0.4 | 0 | 1 | 0 | 1.52 |
| 47000 | 19 | 0.69 | 0.275 | 0.395 | 0 | 0 | 0 | 4.02 |
| 12338 | 28 | 1.54 | 0.314 | 0.308 | 0 | 0 | 0 | 4.02 |
| 1880 | 28 | 1.07 | 0.312 | 0.302 | 0 | 0 | 0 | 2.77 |
| 17940 | 28 | 0.92 | 0.516 | 0.341 | 0 | 0 | 0 | 2.34 |
| 702 | 27 | 1.49 | 0.307 | 0.333 | 0 | 0 | 0 | 15.4 |
| 2296 | 27 | 1.69 | 0.307 | 0.388 | 0 | 0 | 0 | 6.22 |
| 3290 | 27 | 1.69 | 0.304 | 0.388 | 0 | 0 | 0 | 10.1 |
| 380 | 10 | 1.64 | 0.285 | 0.368 | 0 | 0 | 0 | 2.79 |
| 173 | 9 | 1.94 | 0.285 | 0.274 | 0 | 0 | 0 | 5.99 |
| 897 | 15.4 | 1.94 | 0.285 | 0.274 | 0 | 0 | 0 | 6.25 |
| 154 | 9 | 1.94 | 0.285 | 0.274 | 0 | 0 | 0 | 4.57 |
| 336 | 16.5 | 1.94 | 0.285 | 0.274 | 0 | 0 | 0 | 5.06 |
| 3036 | 27 | 2.21 | 0.292 | 0.123 | 0 | 0 | 0 | 10.6 |
| 147000 | 25 | 1.21 | 0.206 | 0.185 | 0 | 0 | 0 | 8.47 |
| 13000 | 20 | 1.28 | 0.412 | 0.309 | 0 | 0 | 0 | 5.26 |
| 3229 | 27 | 1.68 | 0.387 | 0.338 | 0 | 0 | 0 | 6.64 |
| 7400 | 24 | 1.19 | 0.369 | 0.292 | 0 | 0 | 0 | 2.34 |
| 9617 | 24 | 1.97 | 0.415 | 0.232 | 0 | 0 | 0 | 4.67 |
| 4000 | 16 | 1.97 | 0.415 | 0.232 | 0 | 0 | 0 | 1.61 |
| 3555 | 15 | 1.97 | 0.415 | 0.232 | 0 | 0 | 0 | 4.74 |
| 1093 | 27 | 0.91 | 0.267 | 0.267 | 0 | 0 | 0 | 13.9 |
| 16 | 30 | 0.76 | 0.361 | 0.331 | 1 | 0 | 0 | 17.5 |
| 153 | 20.5 | 1.32 | 0.381 | 0.367 | 1 | 0 | 0 | 29.6 |
| 479 | 25 | 1.2 | 0.454 | 0.367 | 0 | 1 | 0 | 7.5 |
| 1144 | 22.5 | 1.17 | 0.413 | 0.4 | 0 | 0 | 1 | 2.7 |
| 242 | 27 | 1.17 | 0.413 | 0.4 | 0 | 1 | 0 | 30.3 |
| 348 | 27 | 1.17 | 0.413 | 0.4 | 0 | 1 | 0 | 31.6 |

**Table A1.1.** *Continued*

| *W* | *T* | *A* | *d* | *p* | *H* | *P* | *D* | Q/B |
|---|---|---|---|---|---|---|---|---|
| 1193 | 27 | 1.17 | 0.413 | 0.4 | 0 | 1 | 0 | 2.24 |
| 996 | 27 | 1.17 | 0.413 | 0.4 | 0 | 0 | 1 | 75.5 |
| 271 | 26 | 1.28 | 0.457 | 0.337 | 0 | 1 | 0 | 28 |
| 2495 | 26 | 1.28 | 0.457 | 0.337 | 0 | 1 | 0 | 3.56 |
| 95 | 26.5 | 1.28 | 0.457 | 0.337 | 0 | 1 | 0 | 65.1 |
| 5700 | 28.5 | 1.28 | 0.457 | 0.337 | 0 | 1 | 0 | 3.3 |
| 361 | 28.5 | 1.28 | 0.457 | 0.337 | 0 | 1 | 0 | 24.8 |
| 2036 | 24.5 | 1.28 | 0.457 | 0.337 | 1 | 0 | 0 | 49.9 |
| 1517 | 30 | 1.28 | 0.457 | 0.337 | 1 | 0 | 0 | 12.8 |
| 2056 | 28.5 | 1.28 | 0.457 | 0.337 | 0 | 1 | 0 | 2.21 |
| 545 | 28.5 | 1.28 | 0.457 | 0.337 | 0 | 1 | 0 | 15.6 |
| 5700 | 20.5 | 1.28 | 0.457 | 0.337 | 1 | 0 | 0 | 17.2 |
| 431 | 26 | 1.28 | 0.457 | 0.337 | 1 | 0 | 0 | 61.8 |
| 95 | 27 | 1.28 | 0.457 | 0.337 | 0 | 1 | 0 | 42.8 |
| 101 | 32 | 1.28 | 0.457 | 0.337 | 0 | 1 | 0 | 15.3 |
| 145 | 32 | 1.28 | 0.457 | 0.337 | 0 | 1 | 0 | 28.4 |
| 145 | 27 | 1.28 | 0.457 | 0.337 | 0 | 1 | 0 | 54 |
| 2495 | 27 | 1.28 | 0.457 | 0.337 | 0 | 1 | 0 | 4.81 |
| 2495 | 32 | 1.28 | 0.457 | 0.337 | 0 | 1 | 0 | 4.15 |
| 1396 | 25.8 | 1.56 | 0.489 | 0.41 | 1 | 0 | 0 | 15.7 |
| 215 | 27 | 1.21 | 0.451 | 0.366 | 0 | 0 | 1 | 35.1 |
| 360 | 26 | 1.48 | 0.479 | 0.35 | 0 | 1 | 0 | 9.28 |
| 1265 | 26 | 1.48 | 0.479 | 0.35 | 0 | 1 | 0 | 4.46 |
| 429 | 27.5 | 1.65 | 0.458 | 0.344 | 0 | 0 | 1 | 113 |
| 5877 | 15 | 2.55 | 0.232 | 0.417 | 0 | 0 | 1 | 4.74 |
| 787 | 23 | 2.55 | 0.232 | 0.417 | 0 | 0 | 1 | 12.3 |
| 215 | 27 | 2.81 | 0.392 | 0.157 | 1 | 0 | 0 | 61.7 |
| 234 | 27 | 1.92 | 0.353 | 0.171 | 1 | 0 | 0 | 42 |
| 81920 | 24 | 5.8 | 0.26 | 0.088 | 0 | 0 | 0 | 11.6 |
| 622000 | 15 | 6.7 | 0.296 | 0.13 | 0 | 0 | 0 | 3.94 |
| 756 | 12 | 0.66 | 0.448 | 0.233 | 0 | 0 | 0 | 3.69 |
| 149 | 12.1 | 0.66 | 0.448 | 0.233 | 0 | 0 | 0 | 7.04 |
| 910 | 12 | 1.01 | 0.511 | 0.183 | 0 | 0 | 0 | 3.43 |
| 3430 | 12 | 1.01 | 0.511 | 0.183 | 0 | 0 | 0 | 2.12 |

Lek S., Giraudel J.L., Guégan Jean-François. (2000).

Neuronal networks : algorithms and architectures for ecologists and evolutionary ecologists.

In : Lek S. (ed.), Guégan Jean-François (ed.). Artificial neuronal networks : application to ecology and evolution.

Berlin : Springer, 3-27.

(Environmental Science). Applications of Artificial Neural Networks to Ecological Modelling : Workshop, Toulouse (FRA), 1998/12.

ISBN 3-540-66921-3