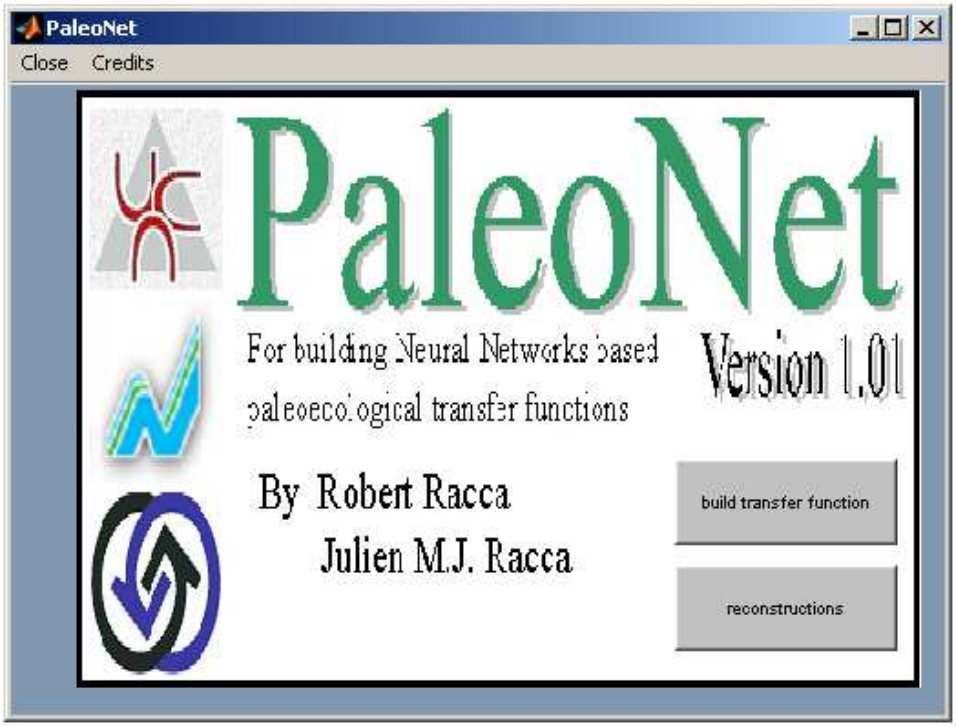
PaleoNet

Software for building Neural Network-based
paleoecological transfer functions

Julien M.J. Racca
Robert Racca

User guide Version 1.01
September 2005

Introduction



Welcome to PaleoNet, a program designed for the development and use of neural network-based paleoecological transfer functions. PaleoNet was developed at the University of Nouvelle Calédonie. It was a joint effort between the Computer Science and Mathematics groups of the University of Nouvelle Calédonie (directed by Prof. Robert Racca), the Paleolimnology-Paleoecology Laboratory (directed by Prof. Reinhard Pienitz) of the Centre d'Étude Nordiques (University Laval, Quebec, Canada), and the Interuniversity research group in Limnology (Prof. Yves Prairie; GRIL, Montreal, Canada).

The following features of neural network analyses are currently implemented.

Standard algorithms:

Backpropagation (gradient descendent algorithm)
Backpropagation with momentum (gradient descendent algorithm with momentum)

Pruning algorithm:

HVS: Yacoub and Bennani (1997)

Cross-validation techniques:

K-fold cross-validation
Jack-knifing cross-validation

Illustration of results:

X-Y scatter and line plots with options to control symbol and line styles, axis scaling.

Program limits

There are no limits to this version of PaleoNet.

Contact

Please send comments / bug reports to racca.julien@courrier.uqam.ca

Julien M.J. Racca
Paleolimnology-Paleoecology Laboratory
Centre d'études nordiques & Département de géographie
Université Laval
Québec (QC)
G1K 7P4 Canada

Suggested citation

Racca J.M.J and Racca. R. 2005. PaleoNet User guide. Software for building Neural Network-based paleoecological transfer functions. Université Laval. Québec, Canada. xxpp

Getting started

Introduction

PaleoNet is an autonomous program developed by Matlab. You must, nevertheless, install the Matlab MCR Installer before launching the PaleoNet.exe program. PaleoNet v.1.01 and Matlab MCR Installer are downloadable free of charge from: …..
PaleoNet is divided into 2 units. The first unit is dedicated to the development of neural networks transfer functions. In order to use the first unit you require modern calibration data composed of both species data and environmental data. The second unit is dedicated to the application of the transfer functions developed in the first unit. Therefore, the use of the second unit requires fossil data on which you want to use the developed models. PaleoNet enables you to eliminate non-contributing species from the transfer functions. Indeed, the HVS pruning method of Yacoub et Bennani (1997) is implemented in the process

Principles of neural network modelling

This chapter provides an introduction to the general principles of quantitative paleoecological modelling based on neural networks.

Artificial neuron: An artificial neuron is a processing element like a biological neuron (Fig. 1a). It works as follows: (1) it receives input (from the original data or from the output of other neurons in the network). Each input comes via a connection, which has a given strength (weight); these weights correspond to the synaptic efficiency in a biological neuron. The weighted sum of the inputs is formed to compose the activation of the neuron. (2) The activation signal is passed through an activation function (sigmoid, tan sigmoid, linear or step function) to produce the output of the neuron. The output is then duplicated as many times as needed.
Back-propagation neural networks: In this type of network, neurons are arranged in a distinct layered topology: one input layer, one or more hidden layers and one output layer (Fig. 1b). The input layer is not really neural at all: these units simply serve to introduce the value of the input variables. The hidden and output layer neurons are each connected to all of the units in the preceding layer.
The back-propagation algorithm (descending gradient algorithm) is based on supervised learning, namely to learn, the system has to know, for each example, the output (environmental variable) associated with the input (species data). The learning phase consists of adjusting the weights of the network connections by feeding a set of input/target pattern pairs (examples) many times. The back-propagation algorithm works as follows: (1) the network is initialized by assigning a learning rate, a maximum number of iterations and random values to the synaptic weights; (2) a training pattern is fed and propagated forward through the network to compute an output value for each output unit; (3) the computed output is compared with the expected output; (4) a backward pass

through the network is performed, changing the synaptic weight on the basis of the observed output errors. Steps 2 through 4 are iterated for each pattern in a training set, then the network performance is checked and a new set of training patterns is submitted to the network (i.e., a new epoch is started) if it needs further optimization. This dynamic

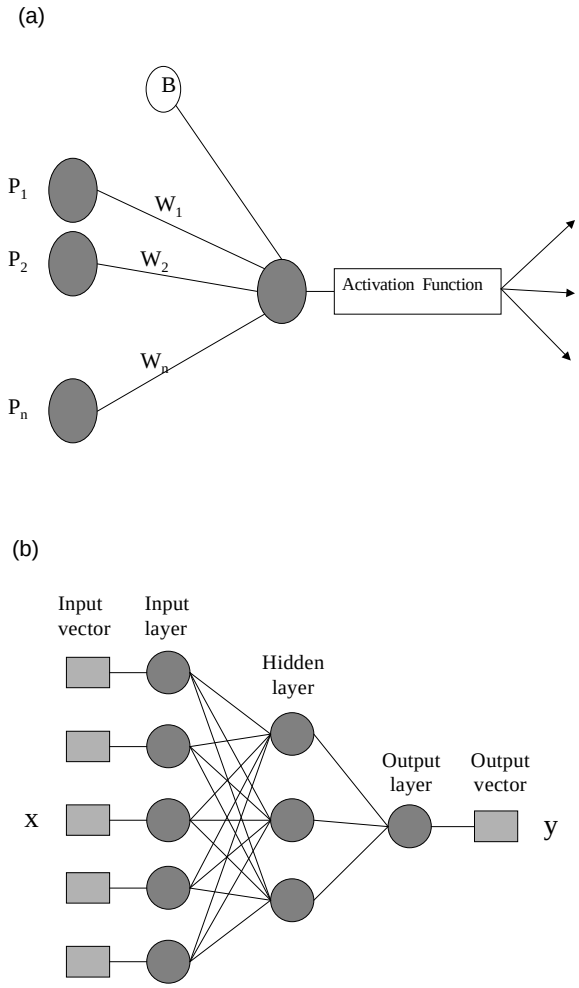


Figure 1. (a) Schematic representation of a simple processing element. The incoming signals (p) are multiplied by the weight of the connections (W) and summed. The bias (B) is then added, and the resulting sum is filtered through the activation function to produce the activity of the neuron.

(b) Schematic representation of the general architecture of a 3-layer back-propagation network with five elements in the input layer, three neurons in the hidden layer, and one neuron in the output layer.

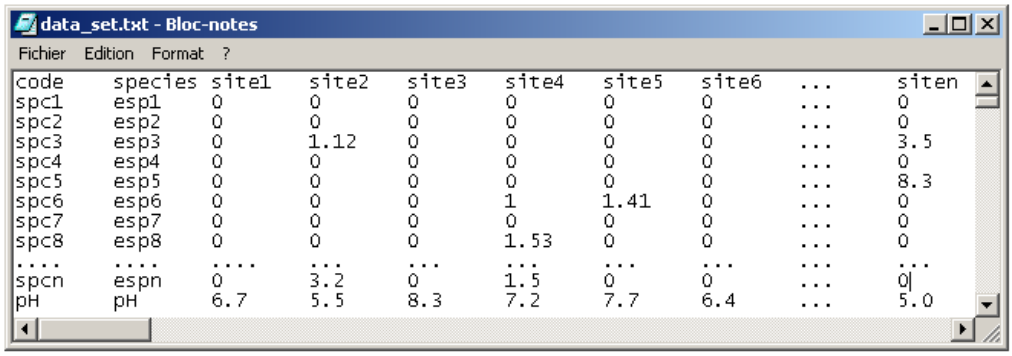
procedure allows the difference between the predicted output and observed output to converge towards a minimal value.

Back-propagation networks are also called "universal approximators" and, as such, they are ultimately able to learn any pattern perfectly. These networks are only really useful if they are capable, after a learning period, of generalizing. In order to generalize, a network must be able to produce the correct output data on samples not included in the learning set. A well-built neural network will, after training with a learning set, give a high proportion of correct predictions when fed a validation set. Several types of validation techniques exist, but the most commonly used in paleoecology involve jack-knifing or K-fold cross-validation principles.

Building neural network transfer functions

File formats

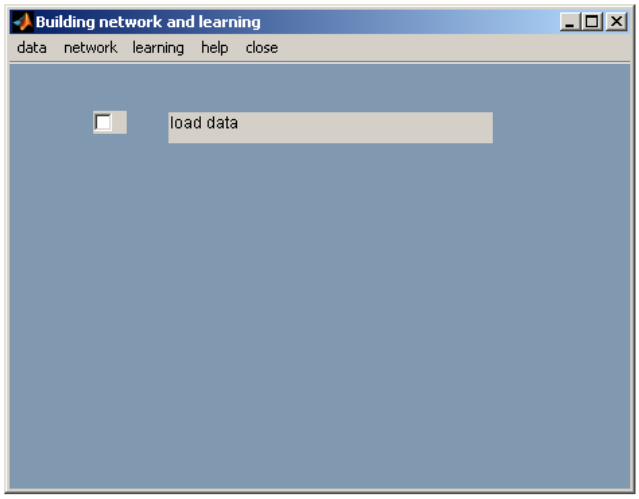
PaleoNet uses text files (tab delimited). The structure of the calibration files must be precisely as follows (see table):



code	species	site1	site2	site3	site4	site5	site6	...	siten
spc1	esp1	0	0	0	0	0	0	...	0
spc2	esp2	0	0	0	0	0	0	...	0
spc3	esp3	0	1.12	0	0	0	0	...	3.5
spc4	esp4	0	0	0	0	0	0	...	0
spc5	esp5	0	0	0	0	0	0	...	8.3
spc6	esp6	0	0	0	1	1.41	0	...	0
spc7	esp7	0	0	0	0	0	0	...	0
spc8	esp8	0	0	0	1.53	0	0	...	0
....	....	....	...	...	...	...	...	...	...
spcn	espn	0	3.2	0	1.5	0	0	...	0
pH	pH	6.7	5.5	8.3	7.2	7.7	6.4	...	5.0

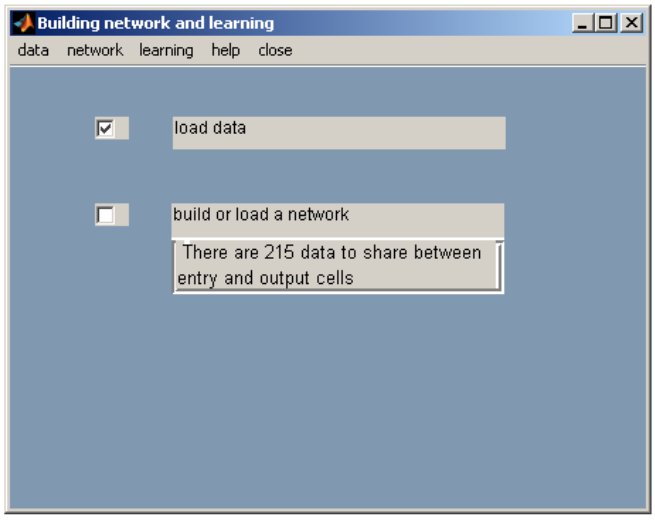
The first column represents the species code (ex: spc1)
The second column represents the name of the species (ex: esp1)
The following columns represent the relative abundance value of the species of each sample (there are as many columns as samples, ex: lakes, site, etc.)
The last row of the file contains the environmental variable to be represented (ex:pH)
(Example: if there are 200 species in 100 lakes, the matrix will have 102 columns and 201 rows)
Note 1: there cannot be any missing data in the text matrix.
Note 2: species code and name must be less than 8 characters
Note 3: in contrast to other programs generally used to develop transfer functions (e.g. C^2, Calibrate, WAPLS, etc) the species and environmental variable data are in the same file.

Load and display data

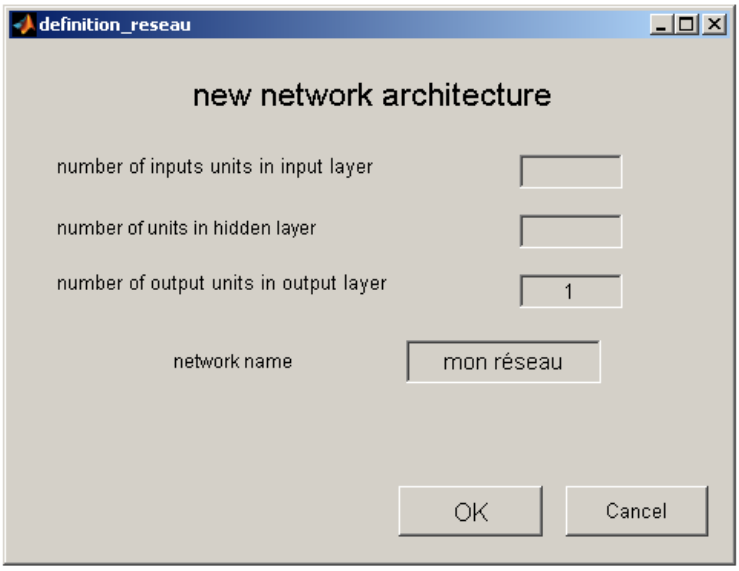


After editing the data, as mentionned in the "file formats" section, click on "load data" in the "data" menu to upload the data. The uploaded data can be visualized by clicking on "show data". An Excel sheet will open and the uploaded data will appear on the screen (**Note**: no changes nor modifications can be made at this stage.)

Build or load a network architecture



Before chosing the type of algorithm and the method of cross-validation, you must define the network architecture according to the calibration data set. In the menu "network" you can upload an existing architecture or you can define a new one.



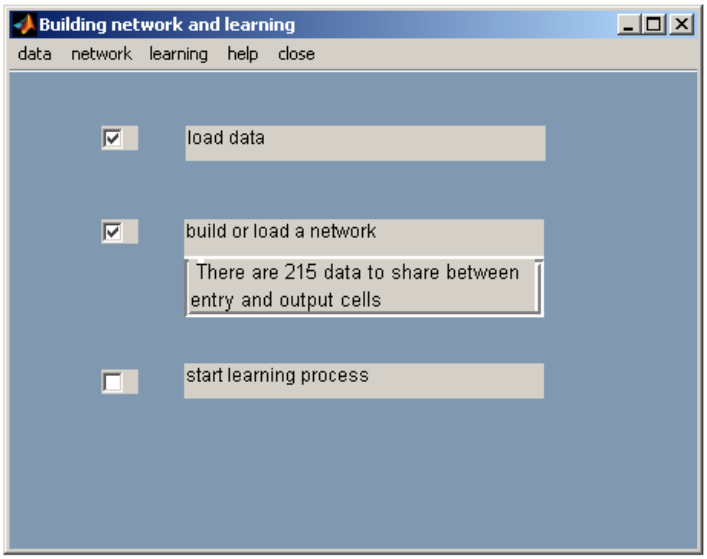
Many types of architectures in neural modelling exist but for the calibration of paleoecological data, we will use a 3 layer architecture. In this type of architecture, the first layer (network input) represents the species (so there will be as many neurons in the input layer as species in the calibration data set.) The last layer (network output) represents the variable to be processed (thus there will be only one neuron on this layer because we are processing one variable at a time.) The intermediate layer (hidden layer) allows us to indirectly connect the network inputs to the network outputs. While the number of neurons in the input and output layers is defined by the data caracteristics to be analysed (essentially number of species), the number of neurons in the intermediate layer is more difficult to define. It is an important parameter in the network structure but there is no formula enabling us to determine this number in order to obtain optimal results. In general, however, the larger the number of neurons in the intermediate layer, the easier the network will converge during the learning process. Conversly, the smaller the number of neurons on the intermediate layer, the less the network will converge. On the other hand, the generalization is less accurate when the number of neurons on the intermediate layer is too large, and vice versa. Thus, the number of neurons on the intermediate layer must be determined in order to obtain a valid adjustment between the capacities to converge and to generalize. Concerning the paleoecological problems, we have established that 3 neurons in the intermediate layer offered the best results (see Racca et al. 2000 or Bishop and al 1995 for further details.)

Note 1: when the number of neurons in each layer is define, then PaleoNet will automaticaly connect the neurons between layers.

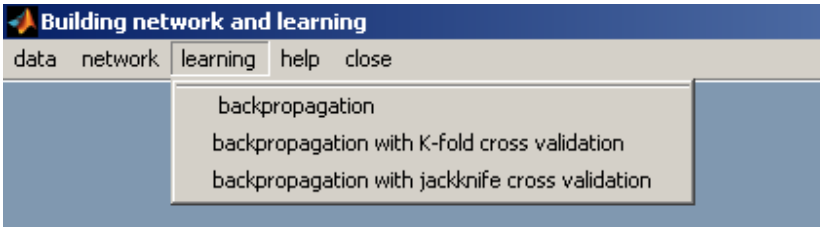
Note 2: the network architecture and the network itself are two different things: the architecture refers to the arrangement of the neurons on the layers, while the network is the model itself (i.e. all the weights of the connexions that are determined after the learning periode.)

Note 3: according to Note 2, it is possible to upload an existing network but not an archtiecture network. The extension of the network files is "net".

Learning



At this level, three choices are possible.



The first choice (backpropagation) enables you to accomplish the learning process but does not allow you to do the cross validation (Figure 1). Knowing that the neural model requires determining various parameters, it may be useful to insure that they are well defined before launching the cross validation routines (jackknife or K-fold) that sometimes require long time periods. This is what you can do with the first choice.

The second choice (backpropagation with K-fold) enables you to use the backpropagation learning process with momentum and to achieve the cross validation K-fold (see cross validation section for details.)

The third choice (backpropagation with jackknife) enables you to use the backpropagation learning process with momentum and to achieve the cross validation jackknife (see cross validation section for details.)

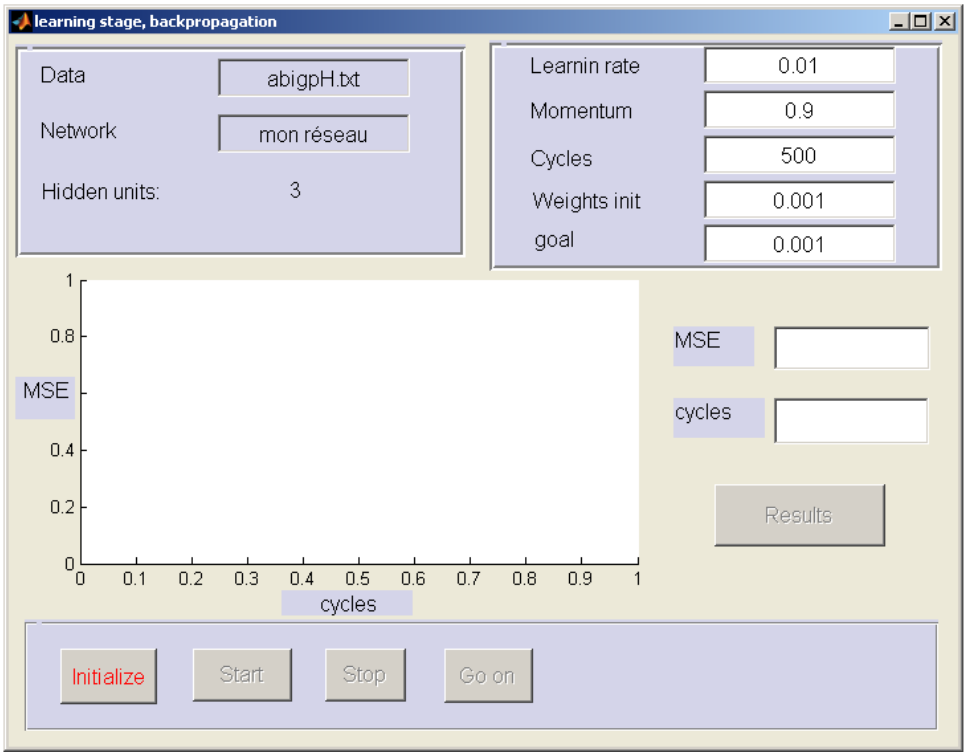


Figure 1:

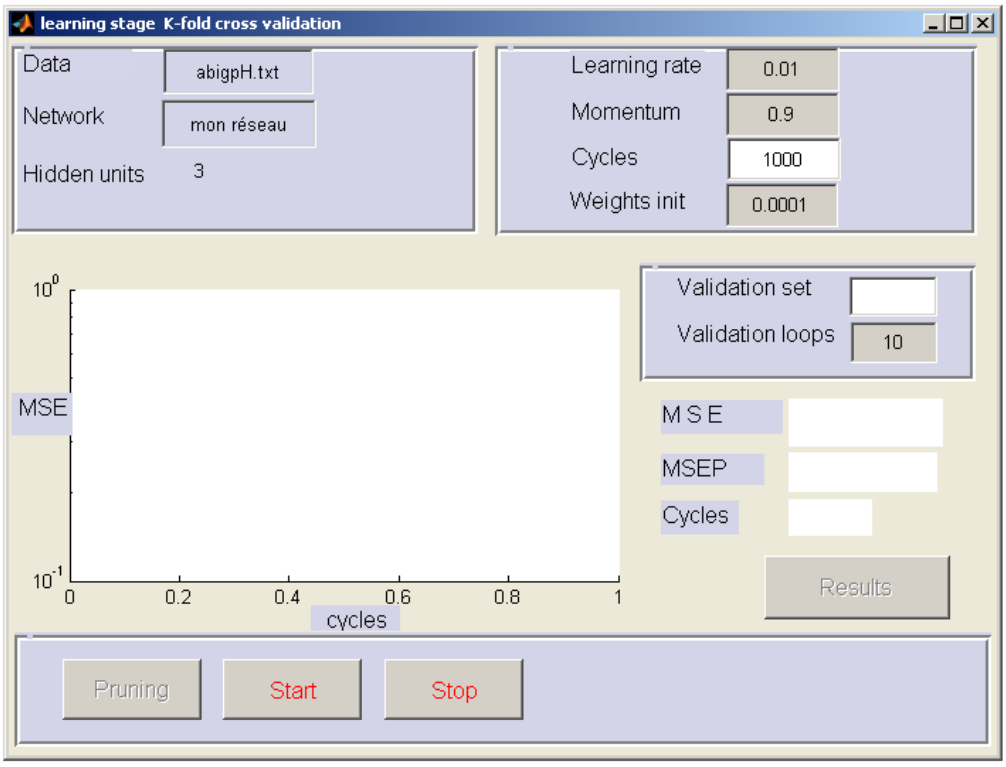


Figure 2:

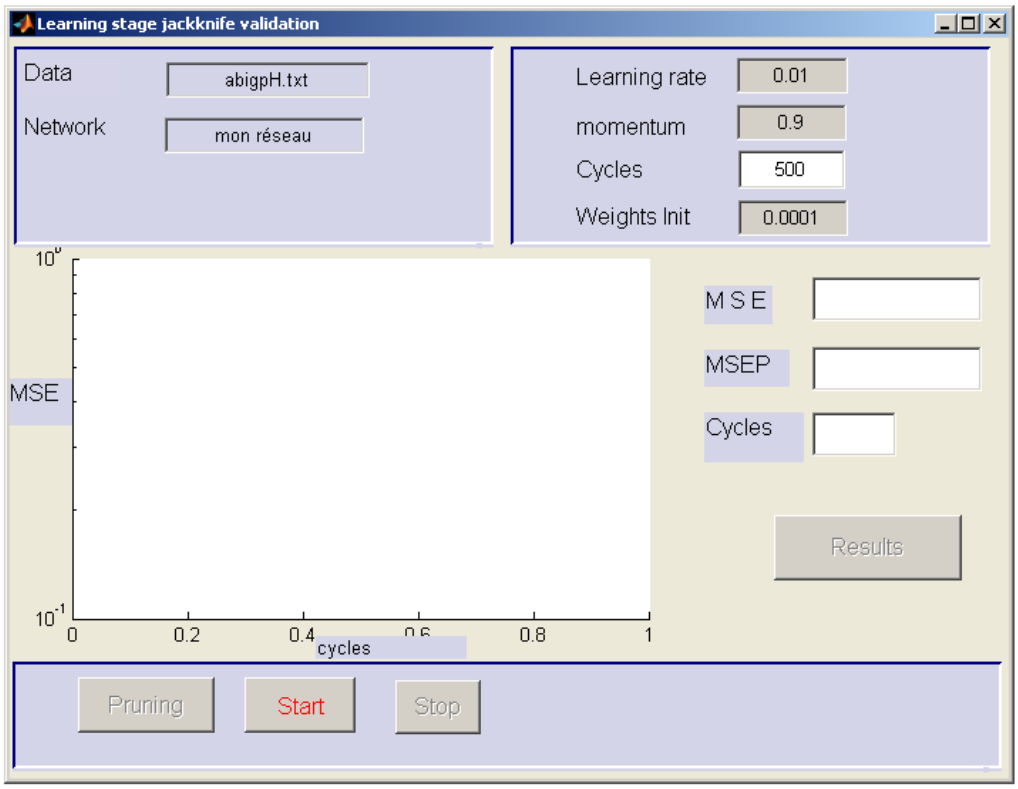


Figure 3:

Depending on the cross validation you wish to achieve, you have the choice between two windows (figures 2 and 3) that correspond to the choices mentioned above. In both cases, you will have to determine certain learning parameters. Moreover, if you choose the K-fold cross-validation method, you will have to define the size of the validation set as well as the number of validation loops.

Defining the learning parameters

The **learning "step"**: it defines the decreasing "speed" of the gradient (i.e. the learning speed.) It can be situated between 10-5 and 1. The default value is 0.001. **Warning**: it is possible that a "step" is too large and therefore will not allow the convergence. On the contrary, a smaller step requires very long learning processes. It is important to find the "step" that allows a rapid convergence (i.e. the error curve of the training set according to the iterations must always decrease. See figures 4 et 5)

The **momentum**: it allows the "smoothing out of the gradient decrease". It can be between 0 and 1. The default value is 0.9.

The **weights** of connections: are determined randomly at the beginning of the process. The default values are voluntarily weak. They are not to be changed.

The **number of cycles**: it represents the length of the learning process. At each cycle, all the learning vectors (input-output couples) are presented to the network. The default value is 1000.

Note 1: An appropriate architecture and well-determined learning parameters, combined with a long enough learning process should allow you to converge your network so that your error is almost null or null (MSE=0) If the network does not converge, increase the number of learning cycles (see figures 1 to 3)

Note 2: The optimal number of cycles, which is the one that allows you to have the best model for the paleoenvironmental reconstructions, is defined during the cross validation. See following section.

Defining the validation groups

Validation group: If you have chosen the learning process with cross validation k-fold, you must define the size of the validation group in the learning window. That is, you must specify the number of samples that will be used to validate the learning process. The data will be divided into a learning group and a validation group. The network will adjust its weights from the learning data and the network obtained will be explained to the validation group. The learning MSE (MSE apparent) and the validation one (MSEP) are calculated from the observed predictions of each group. The suggested choice of proportions for the learning and validation "k-fold" paleoecological data is respectively 80-20% or 90-10%.

Note: If you choose the "jackknife" validation type, the data will automatically be split into two groups according to the proportions n-1 (learning) and 1 (validation) (where n is the total number of samples.) The same principle applies: the network will adjust its weights according to the learning data and the obtained network will be applied to the validation group.

Number of loops: in order to accurately validate your models, it is recommended to perform several learning processes and validations with different groups in order to obtain an average of MSE and MSEP. The number of loops represents the number of consecutive learning processes and validations. When the number of loops is finished,

the average of MSE and MSEP is calculated according to the MSE and the MSEP of each loop.

Note1: the proportion of the learning process and validation groups is the same in every loop. The groups are determined randomly with possible return.

Note 2: in the "jackknife" cross validation type, the number of loops is equal to the number of samples: every sample is used once as a validation group. When the simulation is completed, the average MSE apparent and MSEP is calculated according to the MSE apparent and MSEP of every loop.

The learning tools:

Start learning: allows you to launch a learning process with parameters (step, momentum, weights of initial connections, number of loops, size of validation group and number of loops) that you have defined. During the learning process, a progression indicator is presented.

Stop learning: allows you to stop a running learning process (before attaining the number of loops.) Useful if you notice a convergence problem, for example. It enables you to redefine the learning parameters without waiting for the end of the learning process and validation.

The displayed results:

The learning curve: at all times, you can visualize the learning curve(s). The curve(s) represent(s) the average apparent error of the MSE model according as a function of the number of cycles.

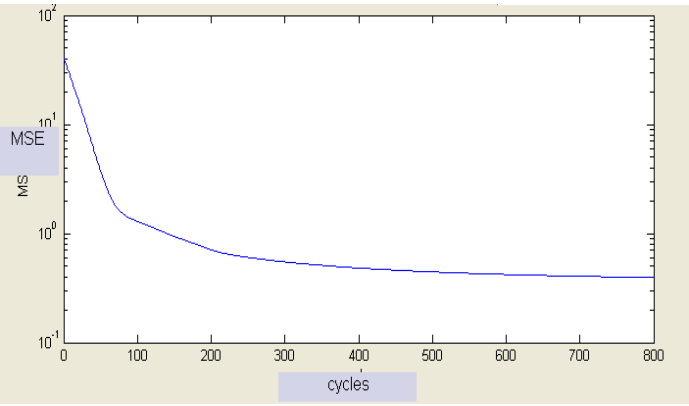


Figure 4

Note1: at the end of a learning process with cross validation, an average MSE curve of each learning process will be presented.

Note2: as mentioned above, it is important that each learning curve be decreasing and rapidly converging to 0. If the curve shows successive decreasing-increasing phases, you must decrease the learning "**step**". If it does not converge quickly enough, you must, on the contrary, increase the "**step**" or the number of cycles.

Note 3: it is probable that the system may not be able to converge in spite of the "**step**" or the number of cycles. If that is the case, you can increase the **number of neurons in the hidden layer**. If there is no improvement, it means that there is no link between the inputs and outputs that can be approached by a mathematical function. This is almost impossible.

The validation curve: at the end of a learning process with cross validation, a curve showing the validated error curve of the model (mean MSEP) will be added to the average MSE curve (figure. 5)

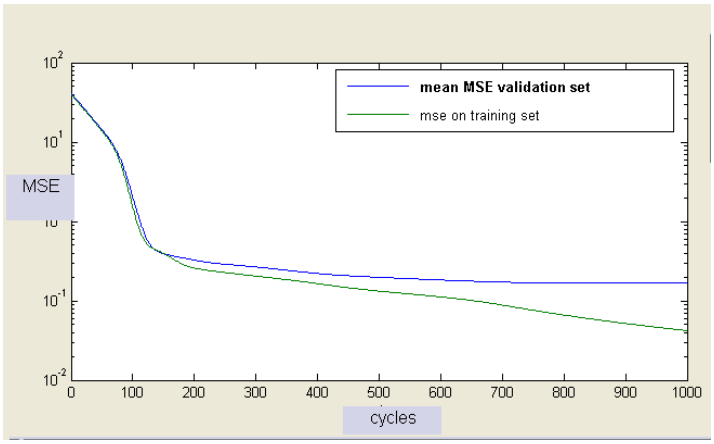
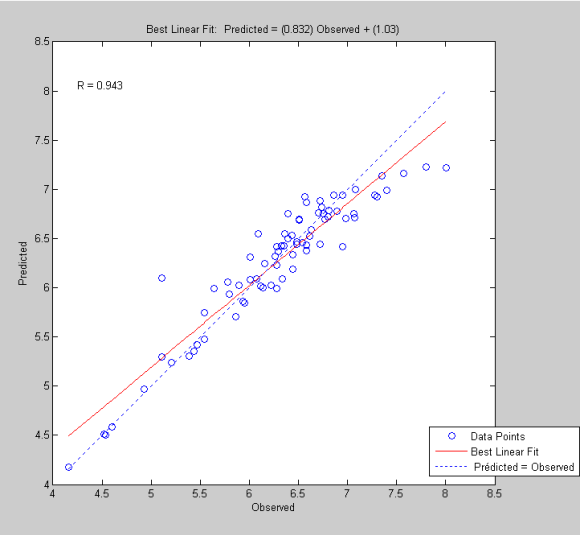


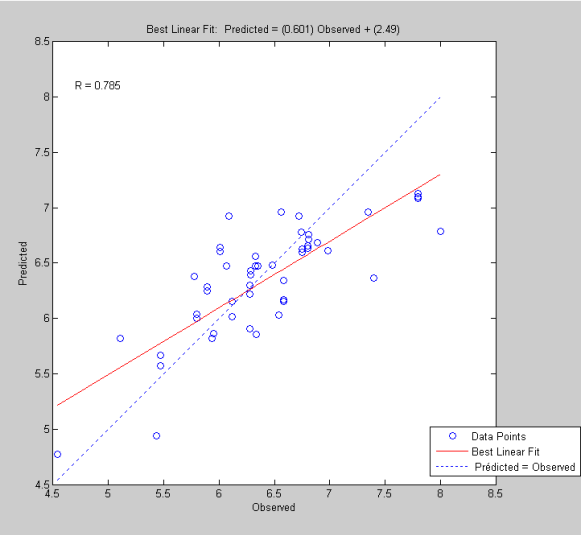
Figure 5

Note1: the validation curve enables you to determine the number of optimal cycles. That is, the one that offers the best capacity of generalizing. A learning process too long induces a low MSE apparent but a higher MSEP. That is called overfitting. Thus, stopping the learning process when the MSEP is at its lowest will prevent overfitting. PaleoNet automatically determines the optimal number of cycles and will automatically relaunch a learning process with that number of cycles as the stopping point.

Note2: if the optimal number of cycles determined by PaleoNet corresponds to the one you established before launching the learning process, it indicates that the number of cycles is too short: the model is capable to converge more and to better generalize. In that case, you can repeat the learning process with a larger number of cycles, or increase the learning "step".

The post regression: by clicking on "**results**" you can visualize the apparent and the real performances for the optimal number of learning cycles. The performances are presented as a regression between observed values and predicted values. The regression characteristics (r^2 apparent and r^2 cross-val, equations and slopes) will be presented separately.

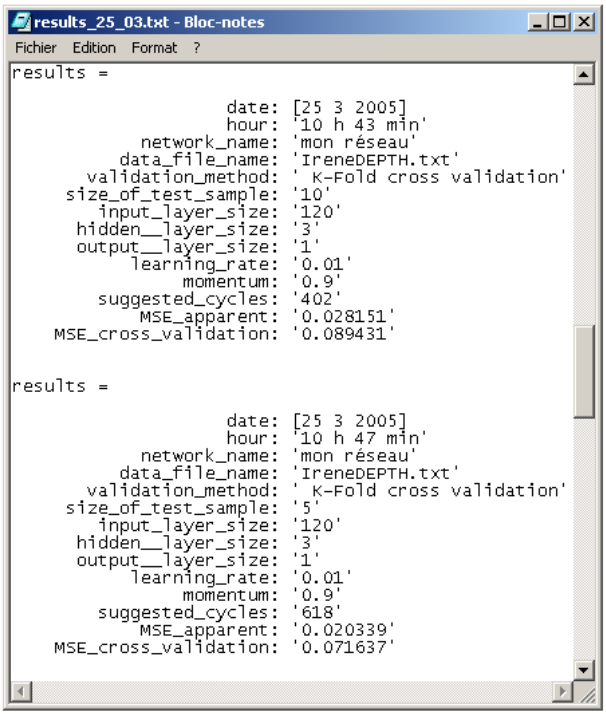




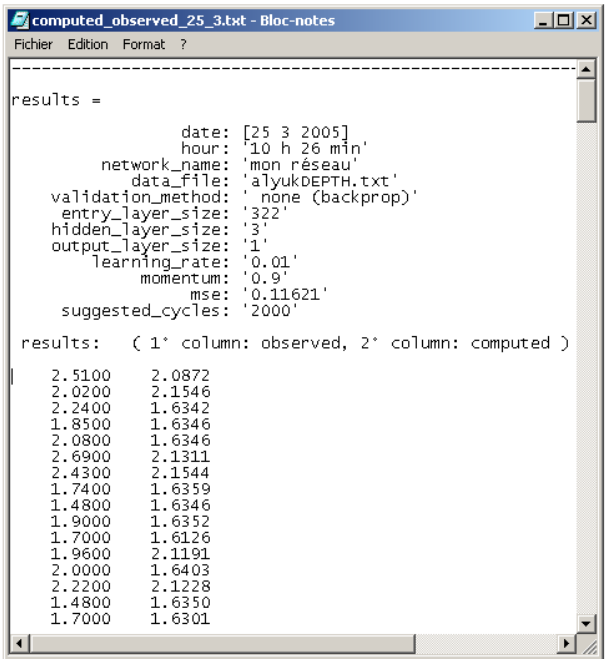
Note: before carrying on, you will have to close the windows showing the post regression diagrams. You can save them in any format.

The saved results:

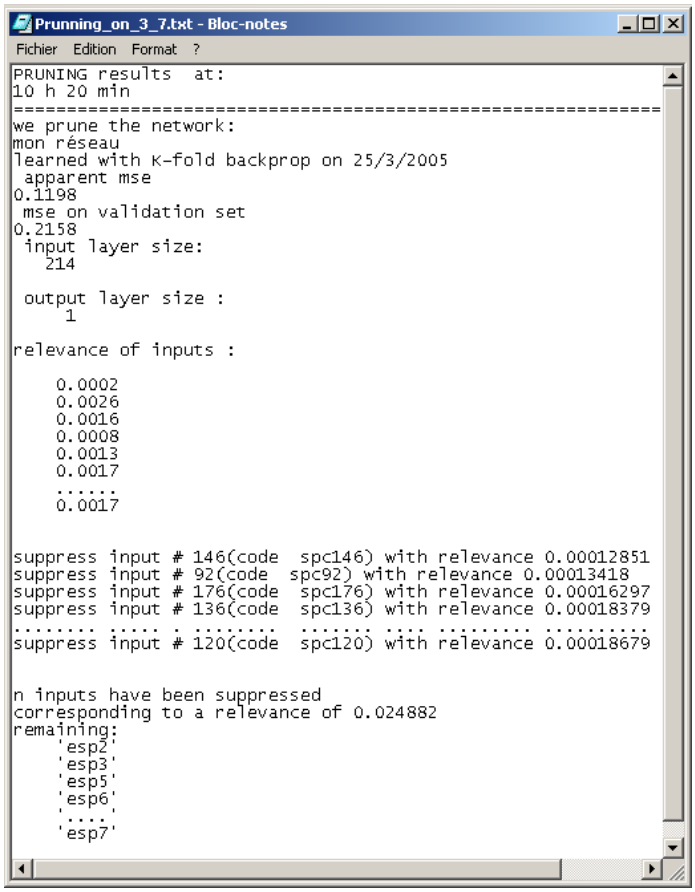
The result file: at the end of each simulation, a text file (results_month_day) showing the principle characteristics and the performances of the network will be created in a "results" folder. Only one folder is created per day (its name includes the date.). If several simulations are performed the same day, the results will be displayed in a sequence (the time when you have saved the results will be indicated in the file.)



The predicted / observed file: a second file is created at the end of a simulation (predicted_observed_month_day). This file groups together the predicted and the observed values for each sample. Apparent and cross validated values are presented. Again, if several simulations are performed the same day, these results will be displayed in a sequence (the time when you have saved the results will be indicated in the file.)



Note: the pruning results are displayed in another file (pruning_results_month_day), also displayed in the "results" folder. Only one pruning file is created per day. If several simulations implying pruning are performed the same day, the results will be displayed in a sequence.



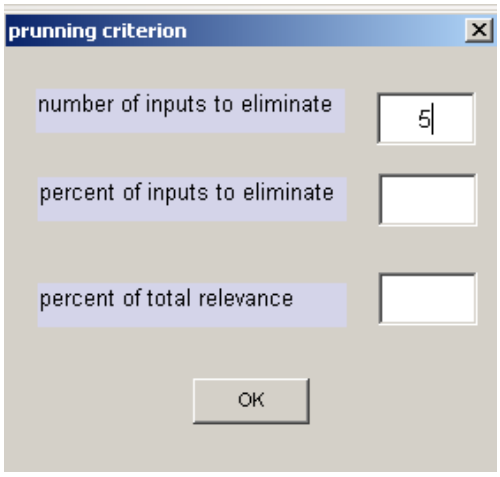
Pruning

The pruning function implemented in PaleoNet was proposed by Yacoub and Bennani (1997). It allows you to determine the relative contributions of the inputs after a learning period and to eliminate the less relevant inputs. When you call the function by clicking on "pruning", you will be asked the pruning criterion. You will have the choice between these three criterions:

The percentage of inputs: you must specify the percentage of inputs (species) that you wish to eliminate. The "x %" less relevant will be eliminated.

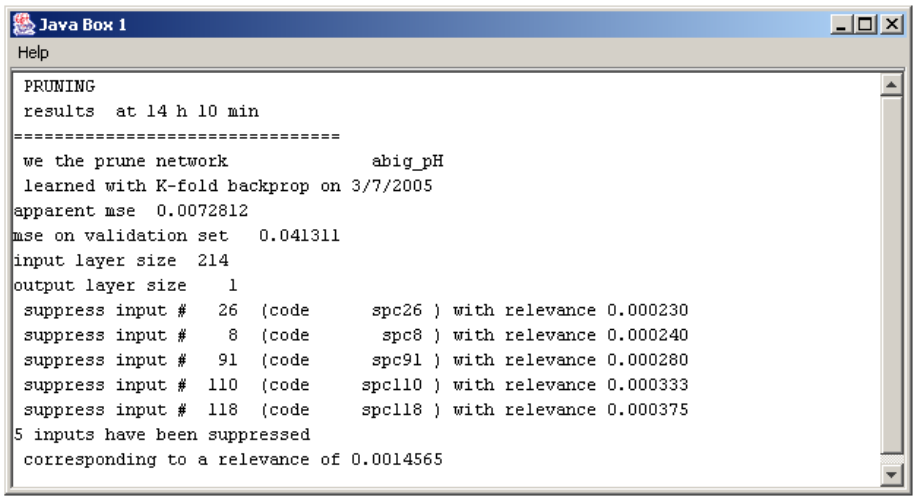
The number of inputs: you must enter the number of inputs (species) you wish to eliminate. The "x" less relevant will be eliminated.

The percentage of relevance: you must specify which percentage of total relevance, to which the inputs are associated, you wish to eliminate. The inputs that represent x % of total relevance will be eliminated.



Note 1: Pruning often allows you to improve the models. Thus, eliminating inputs (species) will induce a decrease in MSEP. On the other hand, after several pruning, the MSEP will not decrease; on the contrary it will increase. Therefore, it is recommended to prune progressively (a little at each time) and verify that the MSEP decreases at each stage. You will thus obtain an optimal model. That is, one that will offer the best predicted performances and that includes only the necessary inputs (species).

Note 2: A file indicating the pruned species is displayed when the pruning is completed



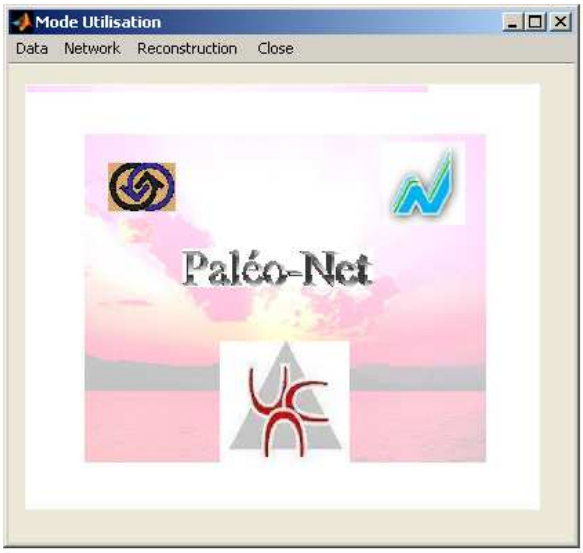
Architecture and network saving:

After a simulation period, it possible to save the architecture and the networks obtained (weights.)
Saving the architecture is useful when pruning, especially when you wish to prune later the same data set. In that case you will not have to redefine the new architecture

corresponding to the number of inputs remaining. (Because the number of inputs and consequently the number of neurons on the input layer will have decreased.) The architecture files have the suffix "....."
Saving the network is even more useful because it allows you to use it subsequently to achieve reconstructions. The network files have the suffix "res"

Utilisation of transfer functions

Once you have calibrated and validated you models (networks) according to your modern data and once you have pruned (or not) the species non-contributive and/or harmful, you are ready to create reconstructions using your fossil data.



Fossil File Formats:

The fossil files must have the same structure than the modern data ones (see page 2.) However, not all the fossil data files contain environmental data (it is what we are trying to reconstruct!) Also, there are no more lines in the fossil files than ther is in the moder ones. Moreover, the columns now represent the stratigraphic sequence and not the different sample sites.

Correspondence between modern and fossil files:

It is frequent that the species of modern data and those of fossil data do not correspond. In fact, it is also frequent that certain species that were used to develop models are not present in the fossil data and vice versa. PaleoNet automatically adjusts the fossil data matrix before using them. A correspondence report is created in the file "reconstructions_month_day" of the file "reconstruction" (see following.) Although the adjustments here do not allow you to determine if the models can be applied on the fossil data (ANALOG tests must be done) it allows you to visualize the specific differences between modern and fossil data. It is especially useful when pruning.

Reconstruction:

After loading the fossil data and the desired network, you can apply the model (network) on the fossil data by clicking on "reconstruction".

Saving:

The values gathered by the model as well as certain useful information (name of the network and of the fossil file) will be saved in the “reconstructions_month_day” file in the “reconstructions” folder. Only one file containing the reconstructions can be created per day. If several reconstructions are completed the same day, they will be presented in a sequence.

References

YACOUB M. & BENNANI Y. (1997),“HVS: A Heuristic for Variable Selection in Multilayer Artificial Neural Network Classifier", International Conference on Artificial Neural Networks and Intelligent Engineering, ANNIE '97, Missouri, USA.