Water Resources Management 9: 299–313, 1995.
© 1995 *Kluwer Academic Publishers. Printed in the Netherlands.*

Neural Nets for Modelling Rainfall-Runoff Transformations

M. LORRAI[1] and G. M. SECHI[2]
[1] *Hydrocontrol, Cagliari, Italy*
[2] *D.I.T. Sezione Idraulica, Universita' di Cagliari, Piazza D'Armi, 09123 Cagliari, Italy*

(Received: 27 July 1994; accepted: 25 January 1995)

Abstract. To obtain river flow data, a neural network (NN) is developed and applied to rainfall-runoff transformation. The NN has been built considering a hidden two layer net and the sigmoidal has been used as a response function. Training is conducted using a back-propagation learning rule. In the input layer, both areal and point data values may be considered. The capability to provide a suitable forecast of river runoff has been examined for the Araxisi watershed in Sardinia. Experiments have been made dividing the total extension of observed data into three ten-year periods, assuming each as a training set, learning the NN and simulating the other two decades over the same period. The obtained model efficiency confirms the capability of this approach to supplying a useful tool in the evaluation of rainfall-runoff transformations.

Key words: hydrology, rainfall-runoff models, neural networks.

1. Introduction

River flow forecasts are required to provide basic information on a wide range of problems in the design and control of river operations. The availability of extended records of rainfall and climatic data that can be used to obtain stream flow data is the major reason for the origin of rainfall-runoff modelling. Starting from the end of the last century rainfall-runoff modelling was applied in a large variety of methods and techniques. Following Dooge (1986), the levels of interest in explanatory theories and predictive models in hydrology include observation, understanding, prediction and control fields. Within this framework models which provide a physically sound description of hydrological processes occurring in a catchment would have significant advantages over a purely empirical model. However, as reported by Kachroo (1992), the development of physical models is still in its early stages, and in operational hydrology two alternative approaches are usually singled out in rainfall-runoff modelling. One is the conceptual modelling method which attempts to represent in a simplified manner the known physical process occurring in the rainfall-runoff transformation. The other approach is systems analysis or the 'black box' approach, which affords general and flexible relationships by its application to input and output records.

Within the latter class, a method to predict the runoff response of the watershed on the basis of known series could be based on the application of a neural net to the previously observed data. Neural nets (NN) are constructed to obtain a prediction of system response without attempting to reach understanding or to provide insight into the nature of the phenomena.

The use of NN in geoscience is relatively new. A recent work by French, Krajewski and Cuykendall (1992) indicates that an NN is capable of performing the complex relationship by describing the space-time evolution of rainfall as the one inherent in a complex rainfall simulation model.

In the 1980's neural nets have also been applied to difficult and generally poorly understood tasks (Dennis and Phillips, 1991). The limit with this kind of approach, which is essentially a black box analysis, is that it is not understood how the NN has solved the prediction problem. Statistical analysis, as principal components analysis and canonical discriminant analysis, is a tool that can be used to interpret the behaviour of NN results. However, using NN in rainfall-runoff transformation, the amount of available information is generally reduced to basic climatic variables, all useful, in such a way that the model-sizing problem frequently reduces to determining the number of preceding time steps to consider in evaluating the actual watershed response.

The aim of this study is to verify the possibility of utilising NN to predict the runoff when only information about the variation of the basic input variables, namely rainfall and temperature, is available. In this first approach to the problem, the time step has been established in month periods. It is clear that with this time unit the reconstructed hydrologic behaviour of the catchment is suitable for water resource studies where storage-yield sequences are usually related to monthly periods but not adequate to deal with other proposals such as those related to flood flow problems. Furthermore, it is known (Pitman, 1978; Piga and Piras, 1983) that if month periods are sufficient to describe the hydrologic behaviour for a design or control problem, there is often no benefit in utilising more complex models based on daily, even hourly rainfall inputs, since total monthly flow predictions of these shorter step models are not more accurate. Undoubtedly, NN modelling could be used with any other time step and with as much more information with respect to the rainfall-runoff process.

2. Model Efficiency

A first requirement is that the model must generate runoff values with first moments as close as possible to the observed ones. However, following this criterion, mean and variance cannot tell us by how much the synthetic and observed data differ in each period. An index of the residual error is the quantity:

$$E = \sum_{i=1}^{N} (x_i - \hat{x}_i)^2, \tag{1}$$

where x_i is the model output in the *i*th period and $\hat{x}_i$ the observed data. Even if this criterion can be used to compare different models for the same watershed, it is not possible to utilise E to compare the reliability of modelling in different watersheds. To overcome this limitation, the model efficiency R^2 has been defined by Nash and Sutcliffe (1970) according to the coefficient of determination in the regression theory:

$$R^2 = \frac{E_0 - E}{E_0}, \tag{2}$$

where

$$E_0 = \sum_{i=1}^{N} (x_i - \overline{x})^2 \tag{3}$$

and $\overline{x}$ is the mean of observed periods. Dealing with hydrological regimes with marked seasonal variations, it has been pointed out by Cao and Piga (1978) and Garrick *et al.* (1978) that the efficiency (2) gives an optimistic evaluation of the runoff reliability prediction. The intrinsic periodic variability of the process represents an important part of its total variability and in this situation efficiency can be redefined as

$$R^2 = \frac{\left(\sum_{d=1}^{D} E_d\right) - E}{\sum_{d=1}^{D} E_d}, \tag{4}$$

where

$$E_d = \sum_{i=1}^{N_d} (x_i - \overline{x}_d)^2 \tag{5}$$

is the square variation of the observed runoff in the season $d = 1, D$.

In the following for comparison we shall use the latter efficiency expression (4) that seems to provide a better evaluation of the model's ability in forecasting flows for the studied catchment where seasonal behaviour represents a main portion of total flow variation.

3. Neural Network Modelling

The NN modelling technique started with an attempt to create a computer methodology that can simulate some important features of the human nervous system; in other words, the ability to solve problems that cannot be solved with the classical programming techniques available today, the possibility of dealing with a great amount of information, and especially, the ability to learn from examples. A lot has been said about these features in nervous systems but all lead to the conclusion that the human brain works in a different way to modern computers. For instance, it seems that computational tasks are not delegated to one complex central processing unit but to many, different, simple, and widely distributed processing units. Even

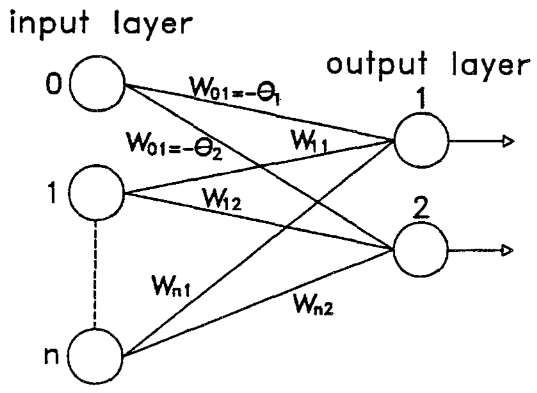


Fig. 1. Perceptron model.

the way in which information is stored is different: it is a 'content-addressable memory' (CAM), so that data is found using content rather than address. Finally the feature of learning from examples seems to derive from the ability to modify nervous associations depending on the problem it has to solve. An extended introduction to basic concepts and features underlying the NN modelling approach can be found in Beale and Jackson (1990).

A first attempt at reproducing the basic functions of a biological neuron by McCulloch and Pitts (1943), simply modelled the sum of electrical inputs as compared to a threshold value θ:

$$x_j = f_h\left[\sum_{i=1,N} w_{ij}x_i - \theta_j\right], \qquad (6)$$

where f_h is the response step function which produces only 1 (positive function argument) or 0 (negative or zero function argument) as output x_j; w_{ji} is the effectiveness index of the transmission ability of the 'link', connecting neuron i to neuron j, called link weight; x_i is the input value from neuron i; N the number of neurons connected to neuron j. Another way of achieving the same effect is by adding an extra input (input 0) that is fixed to be on all the time ($x_0 = 1$) and has a link weight equal to minus the threshold value. This value is called the neurons' bias or offset.

In the first artificial neural network, created by Frank Rosenblatt in 1962 called perceptron, neurons were arranged within one active layer (output layer). Figure 1 shows an example of perceptron with two active neurons in the output layer. The input layer is only responsible for distributing inputs and does not carry out any threshold calculation. The imposition of arbitrary link weights allows the network to obtain an output, but, at first run, this is most likely to be very different from the desired output.

Widrow and Hoff (1960) proposed a learning rule, known as delta rule, by which the link weights are modified using an error term equal to the difference between

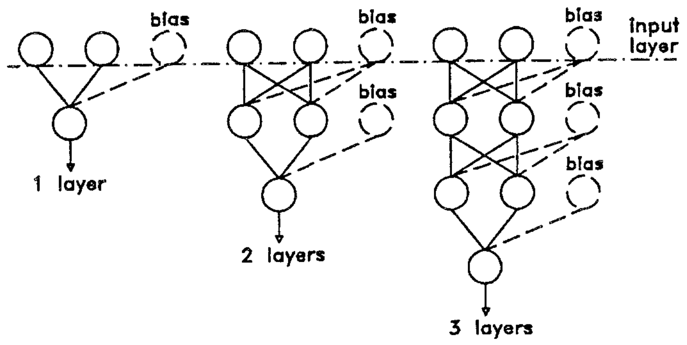


Fig. 2. Examples of NN with different layers.

desired output and actual output. This network link weight adjustment, guided by the knowledge of what is to be achieved, is known as 'supervised learning'. The error term can be written

$$\Delta_j = \hat{x}_j - x_j, \tag{7}$$

where x_j is the actual response of the system to a known input at learning cycle t. The learning rule is

$$w_{ij}(t+1) = w_{ij}(t) + \eta\Delta_j x_i. \tag{8}$$

The gain $0 \leq \eta \leq 1$ controls the gradual process of adjustment to new link values moving towards best values. Minsky and Papert (1969) proved that the simple perceptron model is only able to classify a pattern if a linear boundary between the pattern classes can be found for it. For instance, it is impossible to find a solution to the classic exclusive-or (XOR) problem. The paper effectively put an end to research in NN for many years.

To overcome this, the next step was to arrange the NN neurons in several layers: the input layer, the output layer and one or more hidden layers between the two. The response function also changed so as to exceed the learning problems showed by the original step function for multi-layer perceptrons. A commonly used response function for NN is the sigmoid:

$$f_h = 1/\left(1 + \mathrm{e}^{-\sum w_{ij}\, x_i}\right). \tag{9}$$

For different problems it might be convenient and would be possible to use different response functions such as linear, hyperbolic tangent, sine, Gaussian, etc.

To choose a proper network configuration for the problem, more layers means ability to solve more complex problems, but also more computational complexity. However complex, we never need more than three layers in a network (Kolmogorov Theorem). In Figure 2 we can see examples of different layer NN.

The new learning rule for this multi-layer perceptron is due to Rumelhart and McClelland (1986) and is called the 'generalised delta rule' or 'back-propagation rule'. This rule essentially implements descendent gradient method in sum-squared error function for finding weights. It starts with the definition of the error term

$$E_p = 1/2 \sum_{j=1,N} (\hat{x}_j - x_j)^2, \tag{10}$$

where E_p is the error function for pattern p. The output x_j is the value given by the response function f_h acting on the weighted sum

$$s_j = \sum_{i=1,m} w_{ij} x_i, \tag{11}$$

$$x_j = f_h(s_j), \tag{12}$$

where *m* is the neuron number in the previous layer. Following Rumelhart and McClelland (1986), we can write:

$$\frac{\partial E_p}{\partial w_{ij}} = \frac{\partial E_p}{\partial s_j} \frac{\partial s_j}{\partial w_{ij}}. \tag{13}$$

We also have

$$\frac{\partial s_j}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \sum_{k=1,m} w_{kj} x_k = \sum_k \frac{\partial w_{jk}}{\partial w_{ij}} x_k = x_i, \tag{14}$$

since $\partial w_{kj}/\partial w_{ij} = 0$ except when $k = i$ when it equals 1. If we put

$$-\frac{\partial E_p}{\partial s_j} = \delta_j \tag{15}$$

substituting (14) and (15) in (13), we can write:

$$-\frac{\partial E_p}{\partial w_{ij}} = \delta_j x_i. \tag{16}$$

Decreasing the value of E_p therefore means making the weight changes proportional to $\delta_j x_i$

$$\Delta w_{ij} = \eta \delta_j x_i, \tag{17}$$

where η is a gain term similar to that used in (8), seen above. To know the value of δ_j for each of the nodes we can write

$$\delta_j = -\frac{\partial E_p}{\partial s_j} = -\frac{\partial E_p}{\partial x_j} \frac{\partial x_j}{\partial s_j}. \tag{18}$$

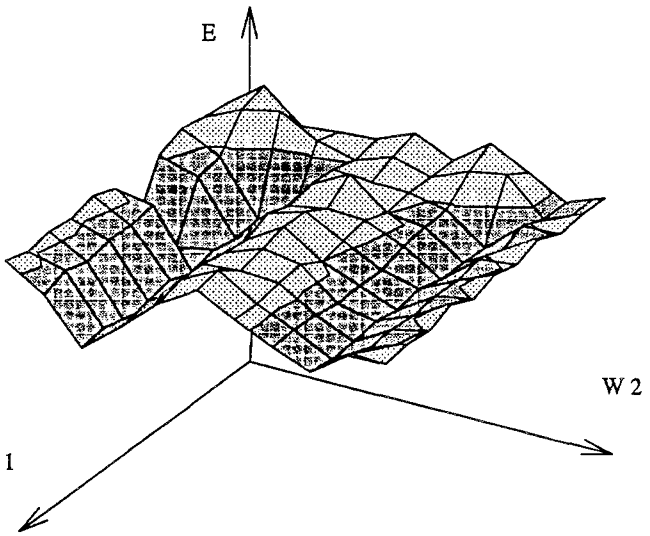


Fig. 3. Behaviour of error function.

If both target and actual outputs are available (output layer), considering Equations (10) and (12), we find

$$\delta_j = f'_h(s_j)(\hat{x}_j - x_j). \tag{19}$$

For the hidden layers we need other considerations since their target output is not known. Therefore, if j is a node in a hidden layer, we can write:

$$\frac{\partial E_p}{\partial x_j} = \sum_k \frac{\partial E_p}{\partial s_k}\frac{\partial s_k}{\partial x_j} = \sum_k \left(\frac{\partial E_p}{\partial s_k}\frac{\partial}{\partial x_j}\sum_i w_{ik}x_i\right) = -\sum_k \delta_k w_{jk}, \tag{20}$$

where the sum index k ranges over the next layer and i over the same layer and since the term $\partial x_i/\partial x_j = 0$ except when $i = j$ when it equals 1. For a hidden layer we have:

$$\delta_j = f'_h(s_j)\sum_k \delta_k w_{jk}. \tag{21}$$

The function is proportional to the terms δ_k in subsequent units, so the error has to be calculated in the output units first, and then passed back throughout the net to the previous hidden units in order to allow them to alter their connection weights. For this reason the rule is called 'back propagation'.

If we suppose that only two weights can vary, it is possible to plot a three dimensional graph of the error function for a particular pattern versus the two weights, which could look as in Figure 3.

The general behaviour of error function gives a rippling surface with rising and falling zones that correspond to points in which the error is lower or higher. The aim of the generalised delta rule is to find the minimum error corresponding to the deeper surface zone. We can use different approaches to improve this research. Lowering the gain term η during training allows to overcome local minima with a large step at first and, as the gain decreases, the solution is reached without wide oscillations. By introducing an extra term called 'momentum' in the learning rule, the updated weights become:

$$w_{ij}(t+1) = w_{ij}(t) + \eta\delta_j x_i + \alpha(w_{ij}(t) - w_{ij}(t-1)), \tag{22}$$

where α is the momentum factor, $0 < \alpha < 1$. The greater the weight changes produced by this term, the larger the changes so that convergence may speed up.

4. The Rainfall-Runoff Neural Model

The NN approach to forecast rainfall-runoff transformation has been implemented by linking modules that create specialised self-adaptive NN to the data structure here considered, with more general modules utilised in training the net. In the present work for this second phase a tool called NeuroWindows (1993) has been tested. It is a dynamic link library (DLL) designed for use with Microsoft's Visual Basic, Access Basic and C^{++} programming languages.

The developed software allows to create and memorise for the basin under study several kinds of two- or three-layered neural networks. Depending on the input variables, it is possible to choose for each network the neuron number for the input layer and for the hidden layers, whilst the software puts one neuron in the output layer (runoff).

The training period and the evaluation period must be specified before the training starts. As the parameter to evaluate training performance the software calculates efficiency R^2, and the algorithm runs until a satisfactory value is reached.

In the following there is a general description of the main routines utilised in the rainfall-runoff neural model. These routines have been here developed in Visual Basic. The code can be divided in three main parts as follows:

1) creation of neural network:

```
network general characteristic definition
procedure MakeNet (net_num, Bp=back_propagation)
layers definition
procedure MakeLayer (net_num, layer_num, neuron_number,layer_type)
links definition
procedure MakeLink(net_num, from_layer, to_layer, init_weight)
```

2) training of the network:

```
while (end_training)
begin
```

```
        for i=1 to training_set
        begin
            training value assignment to the input layer
            procedure PutLayer (net_num, training_values, input layer)
            input value propagation towards output neuron
            procedure Propagate (net_num)
            error calculation and back propagation
            procedure train (net_num, desired_value)
        end
        efficiency evaluation
        procedure evaluate (net_num, evaluating_value, end_training)
    end
```

3) evaluation over simulation period:

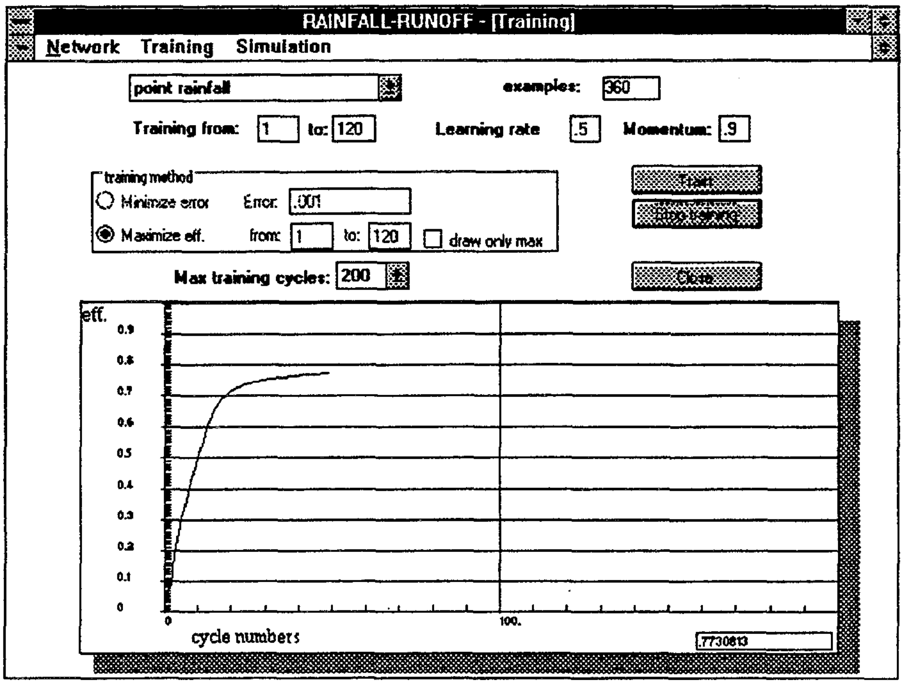
```
    simulating value assignment to the input layer
    procedure PutLayer(net_num, training_values, input_layer)
    input value propagation towards output neuron
    procedure Propagate(net_num)
    get the output value from the output layer
    procedure GetValue(net_num)
```

All the input work and the output results are given in a friendly to use and graphically-assisted manner. An example of a training frame is given in Figure 4.

5. Application of the NN Model

The capability of NN to provide a suitable forecast of river runoff has been examined for the Araxisi watershed in Sardinia (Italy). The catchment area is 121.0 km^2 and the mean quota 740 m above sea level. The basin is almost impermeable and the hydrologic regime quite Mediterranean with a dry season during the summer and a rainy period from late autumn to the beginning of spring. For this basin the measured series of rainfall and runoff has been considered for 30 years from 1946 to 1975. Figure 5 shows a schematic representation of the basin and the location of gauges. Mean areal month values of rainfall over the basin have been simply evaluated by giving fixed values to each station weight. Temperature data are available for only three stations in the basin.

The NN created for modelling rainfall runoff transformation in this basin has been built considering a two hidden layer NN for each simulation trial. The sigmoidal has been used as a response function. The initial weight of each link was chosen at random in the range [-0.5; $+0.5$] while the learning rate was 0.5 and the momentum 0.9. During training the momentum term was automatically halved each time the efficiency over the training set started to decrease or remain steady after a significant number of cycles.


RAINFALL-RUNOFF - [Training]

point rainfall

examples: 360

Training from: 1 to: 120

Learning rate .5

Momentum: .9

training method

Minimize error Error: .001

[✓ Maximize eff.] from: 1 to: 120

draw only max

Max training cycles: 200

.7730813

Fig. 4. Example of interactive NN training procedure.

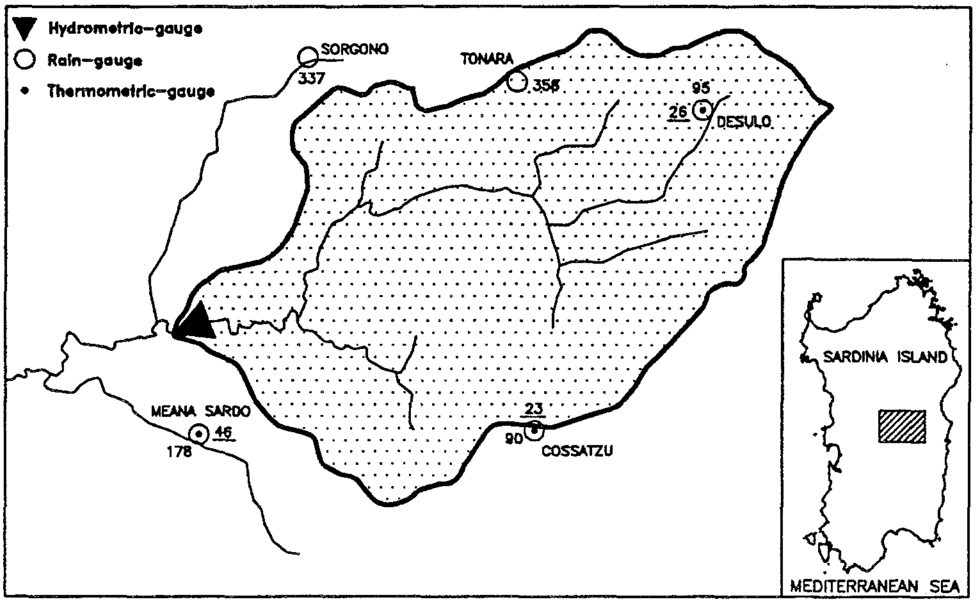


Fig. 5. Araxisi basin.

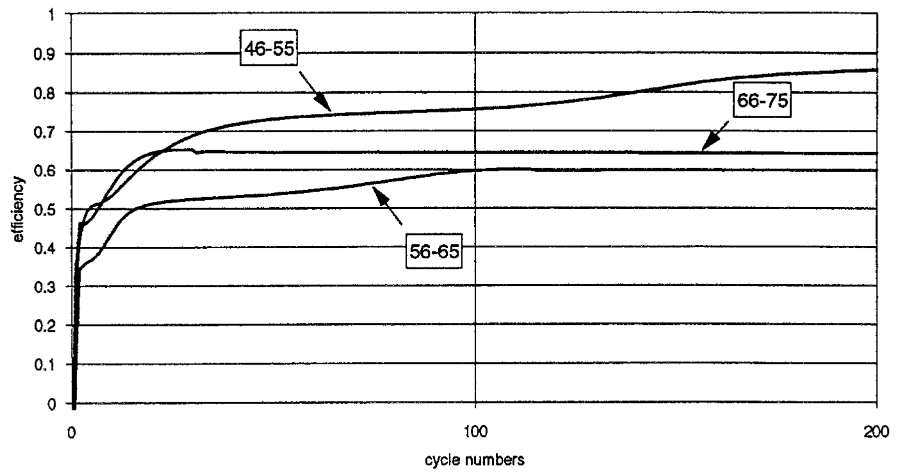


Fig. 6. Efficiency assuming training period 1946–55.

Preliminary experiments have been carried out to examine and compare the efficiency of NN, on varying the number of previous month time steps included in the input layer. The code will construct a differently-sized NN, as the input and hidden layers spread on increasing input variables. With regard to mean areal values, the NN model still remains within a very small dimension, considering as input nodes, rainfall, temperature and runoff of the preceding months as well as rainfall and temperature of the examined month.

Experiments have been made dividing the total extension of observed data in three ten-year periods, assuming each of the ten years as training periods and running the training phase up to 400 cycles to reach the best efficiency values. It is shown that there is no special advantage in taking into account more than the data of the previous month, as for fixed and limited learning cycles this condition frequently assures best results. However, the efficiency reached in the training set of the 1946–55 ten year period is always more than 0.8 with 400 learning cycles. Clearly the validity of the model decreases when evaluating the efficiency over an external period.

Figure 6 shows the behaviour of the efficiency reached on increasing the cycles used in learning the NN when considering the training period 1946–55, and when evaluating with the same parameters (link weights) also the runoff for the simulating periods 1956–65 and 1966–75. The final values reached are 0.856 for the training period, and 0.597 and 0.641 for the simulating periods.

These results have been obtained after 200 learning cycles with a reasonable computing time even on a small PC 386. On increasing the number of cycles a sort of over-training can often be observed so that when the 'internal' efficiency is increased slightly in the training period, a corresponding drop is observed in 'external' efficiency in the simulating period.

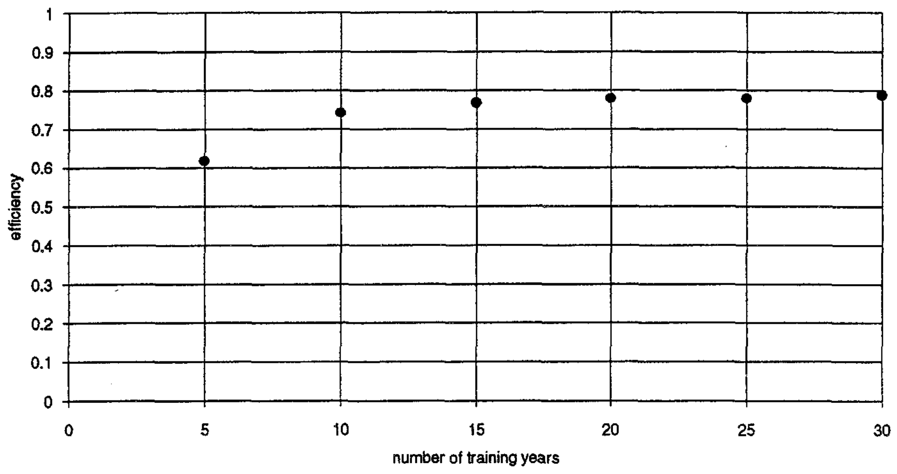


Fig. 7. Efficiency in 1946–75 on varying the length of the training period.

Regarding the increase in efficiency obtained using a longer training period, Figure 7 shows the behaviour of R^2 evaluated on the 30 available years but training the NN model on different lengths of the training period. As expected, the increase remains significant until the total length is reached but the values are already significantly high for the ten year training assumed here for the testing simulation.

With the NN model it is easy to take into account point rainfall and temperature instead of mean areal values. Clearly the NN becomes greater because each rain and thermometric gauge determines the necessity of considering a node for each input month and a node for the hidden layer. In the Araxisi basin taking into account the gauges in Figure 5 and considering the actual and preceding month data in the input layer, the NN has 17 nodes in both the input and the hidden layers. The computing time remains acceptable also on a small computer. Moreover NN gains in efficiency as can be shown in Figure 8 where the results obtained with mean areal and point data input are compared. The point-data NN model reaches an efficiency of 0.899 for the ten years of the training period.

The user can carry out this kind of approach even more easily since there is no need to evaluate an aggregate of point values and since considerations about the geographical position of the gauge can be left out because NN makes an automatic evaluation of the best weights for each station in the model.

With point input data Table I shows the values of efficiency R^2 obtained by changing the training period in the three ten year periods available. In Figures 9 and 10 the behaviour of the observed and calculated runoff flows are compared for the best ($R^2 = 0.899$) and the worst ($R^2 = 0.454$) case.

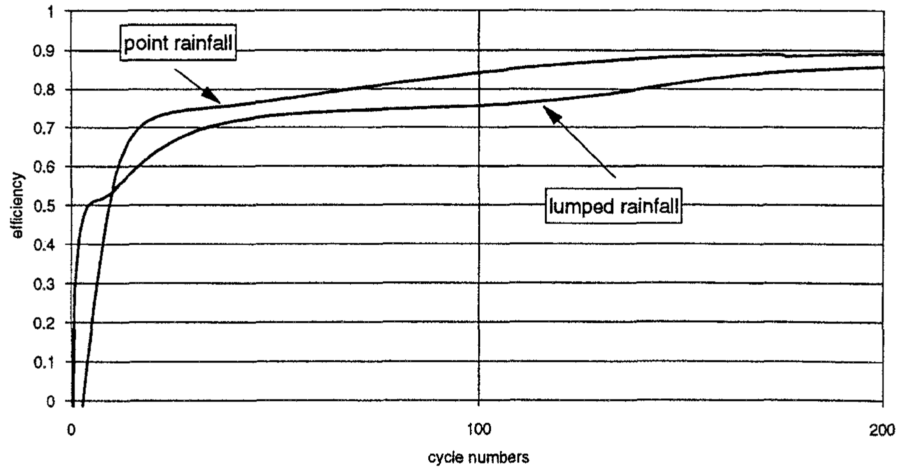


Fig. 8. NN model efficiency considering point and lumped rainfall data.

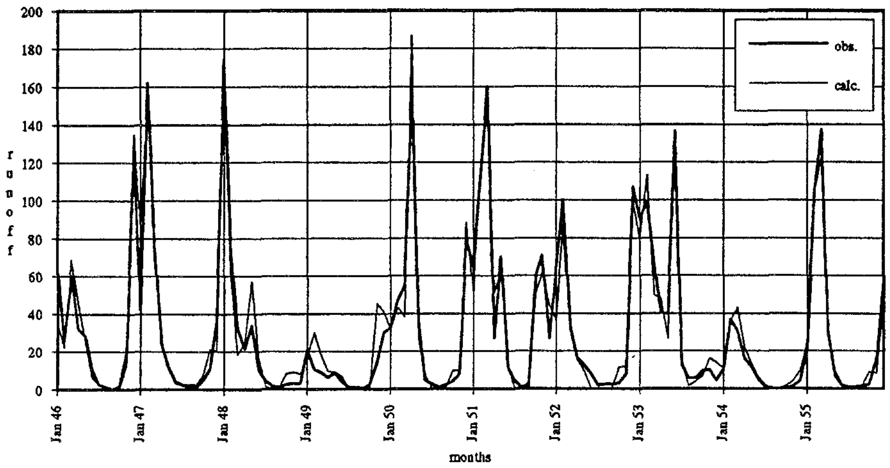


Fig. 9. Observed and evaluated runoff flows ($R^2 = 0.899$).

6. Final Remarks and Conclusions

The NN model applied to the rainfall-runoff transformation problem seems to reach encouraging results for the basin under examination. Moreover, working

TABLE I. Evaluation period

training per.	1946–55	1956–65	1966–75
1946–55	0.899	0.480	0.685
1956–65	0.597	0.699	0.472
1966–75	0.559	0.454	0.864

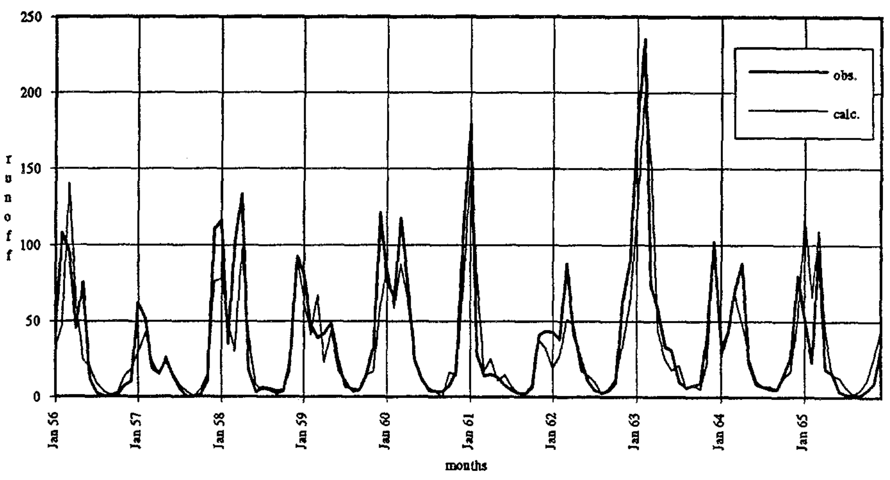


Fig. 10. Observed and calculated runoff flows ($R^2 = 0.454$).

with this simple basin the main indications about NN modelling features find confirmation: i.e. their ability to reach self organising well balanced link weights and the possibility of being applied at the real system level.

As pointed out by Beale and Jackson (1990) NN are far from being a universal panacea for all computing situations. Nevertheless, they are an efficient tool in solving many problems where the relationships between inputs and outputs are not explained thoroughly. Finally, to give weight to above features we can compare them with the results reported by Piga and Piras (1983) on the ability of a conceptual model to simulate monthly stream flows for the Araxisi basin. The best results obtained considering the calibration period 1951–75 gave an internal efficiency of 0.558 while in the 12-year periods 1951–63 and 1963–75 internal efficiency was 0.632 and 0.537.

Other comparison values are the efficiency obtained from a simple multivariate AR model built essentially by taking into account the same variables as in the NN model. The internal efficiencies reached in the 3 decades 1946–1955, 1956–1965 and 1966–1975 are, respectively, 0.601, 0.429 and 0.504.

The results obtained with NN are significant better than those reached in the two above models and confirm the ability of this approach to provide a useful tool in solving problems in hydrology.

References

Beale, R. and Jackson, T.: 1990, *Neural Computing: An introduction*, Institute of Physics, Bristol, Philadelphia.

Cao, C. and Piga, E.: 1978, Considerazioni su un modello deterministico operante su base mensile del processo di trasformazione afflussi-deflussi di un bacino imbrifero impermeabile, *Atti Facolta' di Ingegneria di Cagliari* **10**, 55–82.

Dennis, S. and Phillips, S.: 1991, Analysis tools for neural networks, Technical Report 207, Department of Computer, University of Queensland.

Dooge, J.C.I.: 1986, Looking for hydrologic laws, *Water Resour. Res.* **22**(9), 46S–58S.

Frenk, M.N., Krajewski, W.F., and Cuykendall, R.R.: 1992, Rainfall forecasting in space and time using a neural network, *J. Hydrology* **137**, 1–31.

Garrick, M., Cunnane, C., and Nash, J.E., 1978, A criterion of efficiency for rainfall runoff modelling, *J. Hydrology* **36**, 375–381.

Kachroo, R.K.: 1992, River flow forecasting. Part 1: A discussion of the principles, *J. Hydrology* **133**, 1–15.

McCulloch, W.S. and Pitts, W.H.: 1943, A logical calculus of the ideas immanent in neural nets, *Bull. Math. Biophys.* **5**, 115–133.

Minsky, M. and Papert, S.: 1969, *Perceptrons: An introduction to Computational Geometry*, MIT Press, Cambridge, MA.

Nash, J.E. and Sutcliffe, J.: 1970, River flow forecasting through conceptual models. Part 1. A discussion of principles, *J. Hydrology* **10**, 282–290.

Piga, E. and Piras, A.: 1983, Influenza del passo di calcolo e della durata del periodo di taratura sulle prestazioni di un modello concettuale, *Atti Facolta' di Ingegneria di Cagliari* **22**, 5–33.

Pitman, W.V., Flow generation by catchment models of differing complexity – A comparison of performance, *J. Hydrology* **38**, 59–70.

Rosenblatt, F.: 1962, *Principles of Neurodynamics*, Spartan.

Rumelhart, D.E. and McClelland, J.L.: 1986, *Parallel Distributed Processing*, Volume 1, MIT Bradford Press.

Ward Systems Group: 1993, *NeuroWindows*, Frederick, Maryland.

Widrow, B. and Hoff, M.: 1960, Adaptive switching circuits, IRE WESCON Convention Record, Part 4, 96–104.