International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:6, 2008

Neural Networks: From Black Box towards Transparent Box
Application to Evapotranspiration Modeling

A. Johannet, B. Vayssade, and D. Bertin

***Abstract*—**Neural networks are well known for their ability to model non linear functions, but as statistical methods usually does, they use a no parametric approach thus, *a priori* knowledge is not obvious to be taken into account no more than the *a posteriori* knowledge. In order to deal with these problematics, an original way to encode the knowledge inside the architecture is proposed. This method is applied to the problem of the evapotranspiration inside karstic aquifer which is a problem of huge utility in order to deal with water resource.

***Keywords*—**Neural-Networks, Hydrology, Evapotranpiration, Hidden Function Modeling.

I. Introduction

NEURAL networks are known for their ability to identify non linear relations. For this reason they are used in the field of hydrology with an increasing success. Nevertheless the fact that the model obtained with neural network is not understandable in terms of physical parameters (black box model) is a brake to their use in this field. On another hand, researches have been done in order to implement possibly non linear differential equations inside the network in order to use available knowledge [1]. The model is then called "gray box". The approach presented in this paper has the same goal than the previous one but starts from the naturalist approach: how can we constrain the network, with naturalist knowledge, in order to oblige it to deliver the information of interest? The problem of evapotranspiration is addressed in this paper and after a presentation of the notation and algorithm in the first part, we introduce the problem of the evapotranspiration estimations in a second part. Starting from the standard multilayer perceptron presented in a third part, we present, at the end, how by successive approximations a hidden estimation of evapotranspiration is carried out. Moreover the estimation seems to be constant even if the number of hidden neurons changes.

II. Neural Networks in Water Sciences Modeling

During the last twenty years there has been considerable research devoted, on the one hand, to the field of nonlinear and adaptive modeling, and on the other hand to the study of neural networks in order to perform such tasks. Nevertheless, the idea of using neural networks' ability to model nonlinear and non-stationary behaviours in hydrological systems emerged only about ten years ago [2]. Currently, several theoretical results and many different learning schemes have proven that neural networks are becoming a very effective tool in hydrological applications.

Neural networks are firstly devices able to learn. In the case of signal processing, or system identification, the set of examples consists of sampled input and output signals.

The second fundamental property of neural networks is that they can implement non linear functions [3]. This property is a necessary one for systems such as catchment areas which may have different responses even when the input is the same (for example, the behaviour during summer or winter is very different- See Figure for illustration).

Clearly, Neural Networks are statistical models. They have been used for about ten years in an increasing number of applications for elaborate rainfall-runoff models using Self-Organized networks [4], or multilayer networks [5-7]. Other approaches are also used, such as fuzzy logic [8] or sequential automata [9]. Moreover, because of their ability to identify non linear dynamical models, recurrent non-directed neural networks are good candidates for simulating fast floods [10]. All these approaches put in evidence that Neural Networks work better than others models, sometimes just a little better, in other cases with a significant improvement.

Neural Networks are efficient in modeling water transfer, and because of the complexity of the phenomenon, we hope that neural network models may significantly improve not only flood forecasting, but also scientific knowledge. A way to deal with this goal is to add constraints into the neural model in order to lead it to implement rainfall runoff relation, in such a way that we can interpret the model. According to this goal, modeling non measurable processing had yet been done [2][10]. This point is at the heart of this work.

A. The Model of Neuron and Multilayer Network

An artificial neuron is a mathematical operator which generally computes two actions: first the linear weighted sum of its inputs, and second the non-linear evaluation of its output. Various models of neurons have been proposed depending on the evaluation function. The formula is:

$$o_l = f\left(\sum_{inputs\_m} c_{lm}.i_m \right) \quad (1)$$

where o_l is the output of the neuron l, i_m is one of its inputs, c_{lm} is the synaptic coefficient linking this input to the neuron under consideration, and f(.) is the evaluation function. For example it is possible to choose f(.) = tanh(.). Linear neurons may exist, they have an Identity function.

A neural network is a set of interconnected neurons. These connections (defined by the set of coefficients c_{lm}) are computed during the learning phase.

It has been demonstrated that any non linear, smooth function can be identified by such a network [2]. The accuracy of the identification depends on the number of hidden neurons. This result is of course very important, but it only constitutes a proof of the existence of the solution; therefore the difficulty is to find the solution using the appropriate learning method operating on an architecture which includes a sufficient number of neurons. In this study we firstly consider the well known two-layer perceptron, and secondly an *ad hoc* network, coding in its architecture the function we want to implement.

B. Learning

The neural network learning phase is the computation of the synaptic weights in order to minimise a *"cost function"*. Different learning rules can be derived taking into account different *cost functions* and different minimising methods. Let us consider only identification and forecasting applications; the cost function J is more understandable in the case of supervised learning, since this function is generally the sum of the squared errors between the measured outputs and the computed values, for each input-output couple of interest. It is possible to consider this "cost" function J as follows (only one output neuron):

$$J(C,k) = \frac{1}{2}\sum_{\{k\}}\left(o^k(C) - d^k\right)^2 \tag{2}$$

where {k} is the set of input-output couples taken for k past values, and C is the set of synaptic coefficients.

Starting from this *cost function*, several learning rules have been proposed depending on the chosen minimising method. The most popular method has been the backpropagation learning rule introduced by D. Rumelhart [11] which uses the steepest gradient descent. However, other more efficient rules have been proposed, for example a descent inspired by second order minimisation methods [12-13]. Amongst these second order methods the "Levenberg-Marquardt" learning rule [14-15] is at present the most powerful and leads in a few iterations to a very satisfactory solution.

1) Backpropagation learning rule

The backpropagation learning rule provides a method for modifying the network's synaptic weights according to the gradient of the quadratic error. It was the first learning rule which enabled learning on nonlinear networks, and which could also operate on multilayer networks.

Let us consider the network shown in Fig. 3. An input-output couple is presented to the network which has to associate the input vector i^k $\{i^k_1, i^k_2, i^k_3, \ldots\}$ to the desired output d^k (scalar value in case of one output neuron). It can be noticed that the intermediate, or hidden, neurons have no desired value. After computation of the network's output o^k, the modification to apply to the coefficients, at time t, using a gradient method with a constant step μ is:

$$c^k_m(t+1) = c^k_m(t) - \mu\frac{\partial J(C,t)}{\partial c^k_m} \tag{3}$$

Therefore, using the backpropagation learning rule, the synaptic coefficients of a multilayered neural network can be computed. Its principal drawbacks are the sensitivity of the result to the initialisation of the synaptic weights, and the slowness of the convergence rate toward a minimum of the *cost function*.

2) Levenberg-Marquardt Learning Rule

Because of its efficiency, the Levenberg-Marquardt rule should be used whenever possible. Nevertheless, the Levenberg-Marquardt learning rule suffers from two drawbacks: first it has to invert a matrix which is an approximation of the Hessian: the second order derivative of the *cost function* relative to the synaptic coefficients, *i.e* a matrix the dimension of which is equal to nc.nc if nc is the number of synaptic coefficients. Sometimes this matrix is too huge to be inverted; sometimes this Hessian matrix may be non-invertible [15].

In some words (see [14-15] for full presentation), Levenberg-Marquardt algorithm starts, as bakpropagation, from a problem of cost function minimization. The principle of the rule is to apply to the coefficients an increment taking into account the first and second order of the Taylor decomposition of the *cost function* (notes that Levenberg-Marquardt addresses the *cost function*, taking into account the whole set of learning couple at the same time t). Noting that the second term of the Taylor decomposition needs the computation of the Hessian Matrix, Levenberg–Marquardt method considers an approximation of the Hessian:

$H=\Delta^T\Delta$, where Δ is the vector composed of the first order derivative of the *cost function* (computed by the backpropagation), the formula is:

$$[H]_{ij,lm} \cong \sum_{\{k\}}\frac{\partial J}{\partial c_{ij}}\frac{\partial J}{\partial c_{lm}} \tag{4}$$

The Levenberg-Marquardt rule assumes that at each presentation t of the whole set of learning couples $\{i^k,o^k\}$, an increment to the coefficients is computed in the direction of the gradient: Δ, with amplitude μ(C,t) such that:

$$\mu(C,t) = \left(\Delta^T\Delta + \lambda(t).Id\right)^{-1} \tag{5}$$

where Id is the Identity matrix.

The interpretation is the following: at the beginning of the learning process, a high value of factor $\lambda(t)$ is chosen in order to lead the matrix $\mu(C)$ to be diagonal dominant. The rule is therefore close to a first order gradient descent rule.

The factor $\lambda(t)$ is then decreased in order to be neglected in relation to the approximation of the Hessian part : $\Delta^T\Delta$. At the end of learning, the computation essentially uses the second order information and in a few iterations comes close to the *cost function* minimum.

This presentation of the Levenberg-Marquartd rule shows that backpropragation is necessarily computed in order to estimate the derivatives Δ.

Starting from the previous considerations, the identification of a dynamic system can be addressed by neural networks in computing learning with input-output couples. It is well known that the behaviour of a dynamic system depends not only on external inputs but also on some internal variables that represent the "state" of the system. Under the condition of observability of the system, these state variables are assumed to be past outputs of the real process. However expertise may indicate that another choice may be to select the most relevant state variables (see S. Narendra in [16] for further considerations).

C. Learning of a Discrete-Time Feedback Network

Considering a network at a given instant, learning is performed using the previous external inputs: {i(t)} plus the state variables: the previous output or complementary state variables. Learning on recurrent networks can be performed in at least two ways: the first one consists in taking into account all the previous values using a recurrent method, see for example K. Narendra [16] and P. J. Werbos [17]; the second way takes into account only a few time events, and formulates the backpropagation on a small window of time as proposed by L. Personnaz [18]. The second way was chosen in this study because of its simplicity.

1) Schemes of identification

Two strategies are possible in order to implement the learning: in the first one the objective is to capture the dynamics of the process. Then the errors coming from the network are taken into account during the learning. The looped input is initialised with the past estimated value of the network. This scheme of identification is called "non directed".

The second way of learning uses measured values coming from the system. This mode is termed "directed".

It is immediately clear that in the case of a neural model with feedback operating on non measurable state variables, the previous discussion is not relevant; the only solution is the non directed model. The identification of the underground flow of water was approached in this way [10].

III. Classical Approach for Rainfall-Runoff Modeling by Multilayer Perceptron

As usual in the neural network field, the first approach is the multilayered perceptron with one hidden layer. We applied rainfall as inputs and runoff as output.

We have applied the model to a well studied underground system in the Ariège (FRANCE): the system of Baget. The work performed by the "Laboratoire Souterrain du CNRS" supplies an extensive set of data: the measures of daily rainfall and daily flow of the river for 20 years.

The phenomena of water infiltration into a karstic system is complicated and still partially unknown. A karstic system is a fractured limestone with a very irregular outflow. Its modeling by neural networks is therefore interesting in the two fields of neural networks sciences and karstology.

The Baget is a Pyrenean river interesting for this study because it is well known [18]. The Baget catchment has an area of 13,25 km^2. Its mean altitude is 950 m and its pluviometry quite important: 1700 mm of water by year. Its annual evapotranspiration is estimated to 54 mm. The evapotranspiration is the quantity of water which is evaporated in the atmosphere or which is consumed by the vegetation. As other Pyrenean springs, the Baget spring presents a low water period during autumn and great flows during winter and spring. Usually the snow melt doesn't give great flows.

The data base has daily flow measurements from august, 31, 1973 to December the 30 1999, and daily rainfall at the "Balagué" station for the same duration. Rainfall and flow are represented in Fig. 1 for the last year: 1999.

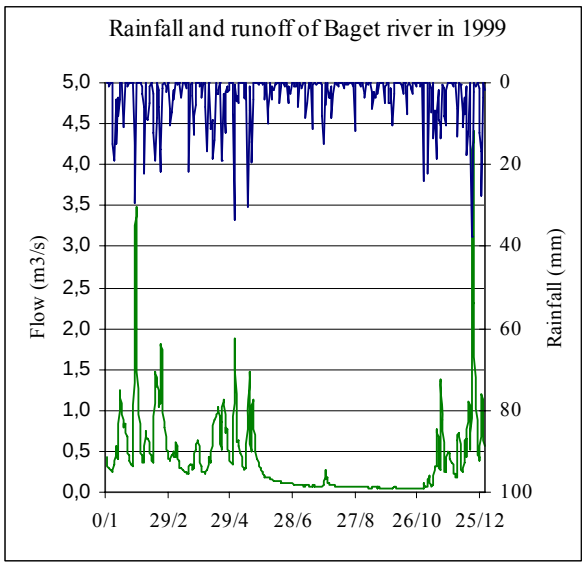

Fig. 1 Rainfall and flows of the Baget river for 1999. One can notices two high level water periods: during winter and at beginning of spring. The low level period is during summer and autumn. One can notice that, due to evapotranspiration, the flow is very low during summer even if some rainfall occurs

In order to compute rainfall-runoff relation by a two layer Neural network, we chose the rainfall measured by a rain gauge at "Balagué" station. An input bias is necessary in order to represent the base flow. Its value is not 1, as is usually applied, but a lower value due to the great number of very low values of the flow during the rainfall recording. This adjustment is necessary in order not to saturate the sigmoids during learning. The mean of the inputs was shown to be a good value. As shown in Fig. 2 we apply the rainfall to the network in a temporal window. This temporal window is essential in order to capture the temporal behaviour of the catchment. Seventeen time steps were chosen for rain recorder, they correspond to the duration of groudwater transfer .

The learning is processed upon twenty years, from 1974 to 1994, validation is estimated on five years beginning in 1995 and stopping in 1999.

At the output of the network we measure the quality of the response using a criterion used in hydrology and called the Nash criterion [19]. The Nash criterion is analogous to the coefficient of determination and is calculated as:

$$\text{Nash} = 1 - \frac{\sum_k \left(o^k - d^k\right)^2}{\sigma^2} \qquad (5)$$

where σ is the standard deviation of the test signal.

The Nash criterion takes into accounts the quadratic error and normalises this error by the variance of the signal. The closer the criterion to the value 1, the better the model is. If forecasting is limited to predict the mean value, the criterion is equal to zero; negative values are very bad.

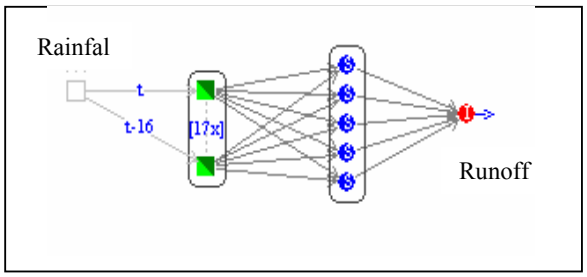

Fig. 2 Two layer perceptron for rainfall-runoff relation. Five hidden neurons are sufficient

One can see on Fig. 3 a typical sequence of runoff prediction, for the year 1999. The mean value of Nash criteria upon five simulations for the validation set is 0.66. This value is not a very good one, and one can note on the hydrogram that the peak values are not well predicted. Also, during summer the network "invents" floods when the water level is very low. The interpretation is that the network has no information about the temperature (evaporation) and the consumption of the vegetation. So it can't evaluate the evapotranspiration.

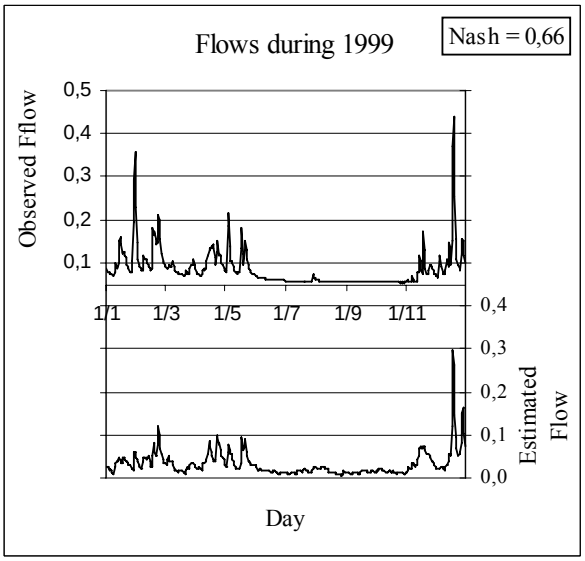

Fig. 3 Observed and estimated flow for Baget river with a standard multilayer perceptron. Because of the evapotranspiration which cannot be taken into account, the prediction is not very good

IV. Adding a Priori Evapotranspiration

A. Taking into account of Evapotranspiration

Even if the phenomenon of evapotranspiration is different in karstic systems compared to other aquifers, it is logical to think that it has an important seasonal component: maximal during summer and minimal during winter.

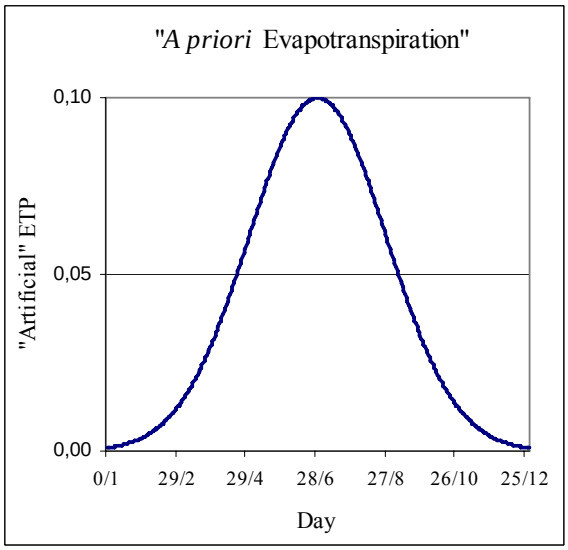

Fig. 4 *A priori* ETP inputted to the network

The classical approach would consists in using a estimation of the ETP via a formula; for example the Turc formula computes an estimation of the evapotranspiration using climatic data as temperature, insolation, … [20].

In lack of climatic data, particularly temperatures, it is

possible to build a signal so called "*a priori* ETP" which would give the information about seasonal variations, and to apply this signal in input of the network

Thus one can compute a "bell" function which is maximal in summer, and minimal in winter. If this signal is inputted to the network, this one would be able to take the information onto account in order to computes the flow. Such a signal is represented in Fig. 4.

New simulations are then computed adding this input to the network with a temporal width of 17 delays as chosen for the rainfall window. It had to be noticed that the architecture of the network has to be modified in order to deliver good forecasts. A three layers network is necessary as shown in Fig. 5.

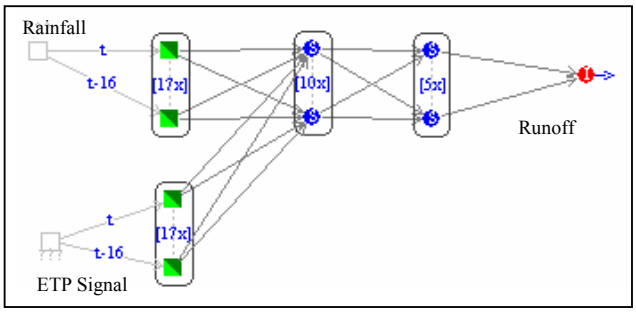

Fig. 5 Architecture of the network with a priori ETP in input

Using such architecture with such inputs leads to better prediction: one can obtain a mean Nash criteria of 0.71 (mean upon five trial with various initialization of coefficients). A good prediction with a Nash criteria of 0,73 is shown in Fig. 6.

One can note on the hydrogram (Fig. 6) that the evapotranspiration has been taken into account: the summer rainfall contributes very little to the water flow.

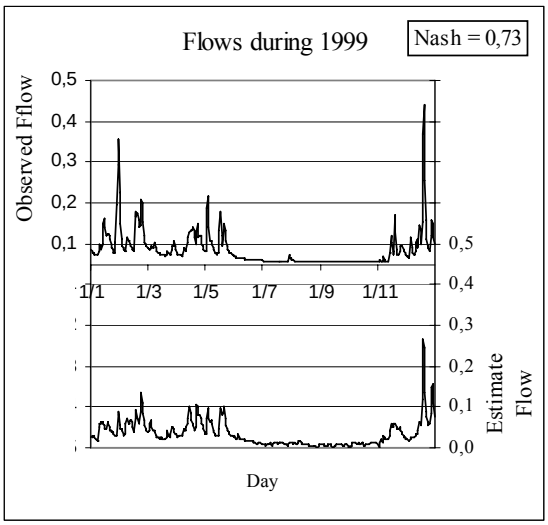

Fig. 6 Observed and estimated flow for Baget river with a standard multilayer perceptron. Inputs are rainfall and a priori evapotranspiration

Nevertheless it is not obvious to know exactly how the neural network have done signal processing on "*a priori* ETP". This information would be very interesting but we are limited by the "black box" behaviour of the neural networks.

V. Extracting Evapotranspiration from Network

A. Not Constrained Network

The improvement observed in forecasting when an "*a priori*" evapotranspiration input is added to the network suggests that the network is able to take this information in account. Nevertheless it is not obvious to extract such information in a "black box" model: the coefficients are not understandable as known parameters. However, in order to perform such a task we have constrained the architecture of the network as shown in Fig. 7.

The idea is as follows: a universal identificator is built in order to identify the evapotranspiration signal: it is composed of a standard two layers neural network with a linear output neuron. Its inputs are the "*a priori*" evapotranspiration taken into account upon a temporal window. Its output: the real evapotranspiration is unknown, but is applied to another network receiving the rainfall as inputs and delivering runoff as output. The estimated evapotranspiration output is called hereafter "hidden ETP". We hope that the network "rainfall-runoff", computing its learning and having a need of information about evapotranspiration, would compute the real function of evapotranspiration. An observer of the output of the neuron called ETP neuron on the Fig. 7, or hidden ETP, would deliver an estimation of the evapotranspiration.

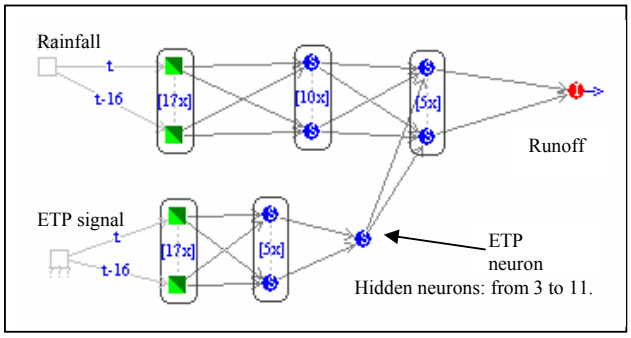

Fig. 7 Constrained architecture in order to "isolate" the evapotranspiration contribution

Simulations are realized with standard neurons with sigmoïdal function (between {+1,-1}) as usually done, and it is very interesting to note that the output of the ETP neuron has always the same shape. The number of hidden neuron of the ETP sub-network has been increased from 3 to 11, and it appears that the forecasting quality is not increased from 7 hidden neurons. Without surprise it appears also that the magnitude of the function can vary a lot: it is the "black box effect". The mean estimation of evapotranspiration is shown

in Fig. 8. Mean is computed upon twenty years of learning and five trials with different initialization.

Even if the quantification of the curve is not possible, it is very interesting to note that the maximum of the curve is not at mid-year as it is supposed with classical models of evapotranspiration, but at the beginning of September. This observation is an important result, and need to be studied on other catchments, karstic or not.

Another observation is possible: the mean estimated ETP may have negative values which is not possible. Both observations: different magnitude and negatives values let us think that the modeling of ETP by the network is not very realistic. Nevertheless, because of the interest of the method we continue in the same way in increasing the constraints by introduction of knowledge in the network.

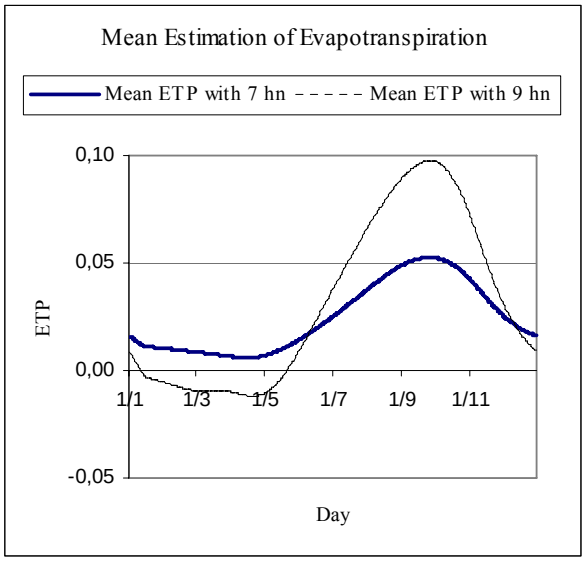

Fig. 8 Estimation of Evapotranspiration. The neurons are coded in {-1;1}

B. Constrained Neurons in {01}

The most obvious constraint to take into account is to force the ETP to be between 0 (all the water goes to the spring) to 1: all the water is consumed). This is achievable by choosing for the ETP neuron a sigmoid function between {0;1}, in place of identity function. Other neurons of the network are coded in {-1;1}. This modification is performed and, as for the previous simulation, we present on Fig. 9 the mean of the estimated ETP upon 20 years and 5 trials.

It is very interesting to note that the shapes are still identical and that the maximum is still in September (the minimum in February). It is surprising to note that another "local maximum" appears in April.

Unfortunately the magnitudes of the functions are still different and could discourage any temptation to quantify the ETP. Nevertheless an estimation of the magnitude of the "impact" of the ETP upon the computation of the flow is possible by multiplying the hidden ETP estimation by the magnitude of the coefficients linking this hidden output to the rest of the network. Unfortunately the "impact" of the hidden ETP is not constant in all cases: with 5, 7 or 9 neurons on input layer of the ETP network as shown in Table I. It is thus difficult to conclude.

TABLE I
ESTIMATION OF THE "IMPACT" OF HIDDEN ETP UPON THE NETWORK

100*Max ETP*Moy(abs(Cij))	5 hidden neuron	7 hidden neuron	9 hidden neuron
	3	2.9	3.3

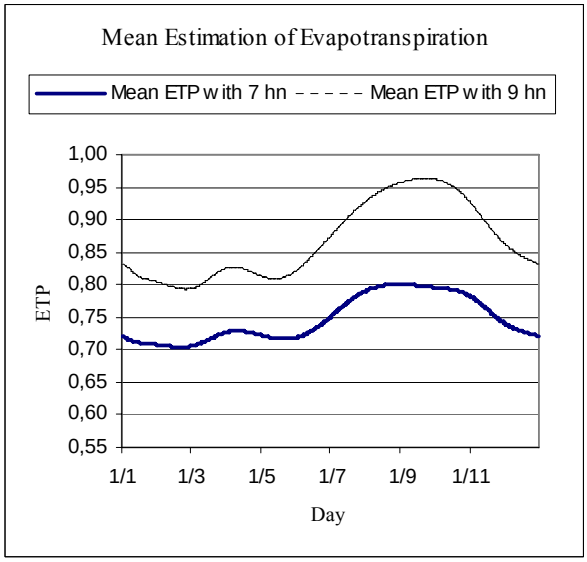

Fig. 9 Estimation of Evapotranspiration. The neurons are coded in {0;1}

C. Constrained Neurons with Regularization

Continuing the reasoning and in order to constrain the magnitude of the curves we have then applied to the goal function of the network a term of weight decay [21]. The weight decay is introduced to prevent networks to overfit during learning: we add in the goal function a term which limits the magnitude of the coefficients during learning. We hope that this term will equalize the magnitude of the coefficients and by consequence the magnitude of the hidden ETP. Typically this term is the squared sum of the coefficient. More precisely we have used in this work the following goal function:

$$G(C,k) = \alpha.\frac{1}{2}\sum_{\{k\}}\left(o^k(C) - d^k\right)^2 + \beta.\frac{1}{2}\sum\left(C_{ij}\right)^2 \qquad (6)$$

Where α and β are *ad hoc* coefficients (0.95 and 0.05 respectively).

The mean estimation of evapotranspiration is still computed upon 20 years and 5 initializations and delivers the signals shown in Fig. 10. It is remarkable to note that the curves with decay or without decay have really the same shape, but the magnitude is still different.

As computed previously an estimation of the "impact" of

the hidden ETP upon the network can be done using the coefficient's magnitude and is reported in Table II.

TABLE II
ESTIMATION OF THE "IMPACT" OF HIDDEN ETP UPON THE NETWORK

100*Max ETP*Moy(abs(C_{ij}))	5 hidden neuron	7 hidden neuron	9 hidden neuron
	2,0	2.1	2.0

It appears in this table that the "impact" of hidden ETP is quasi constant which is very interesting: we have probably obtained a realistic value for the ETP.

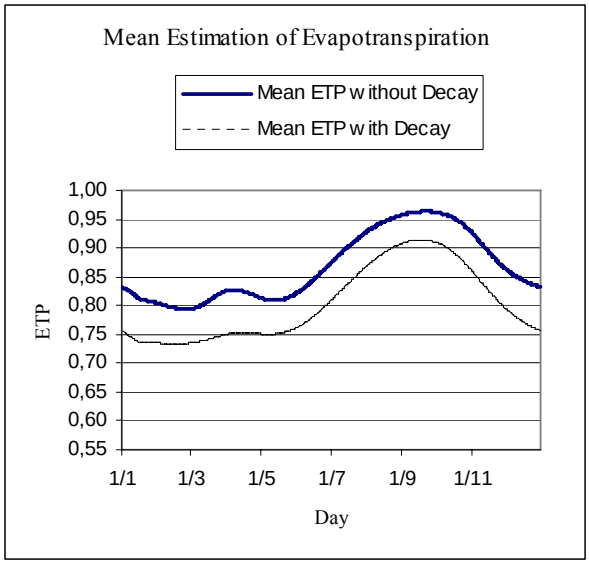

Fig. 10 Estimation of Evapotranspiration, with and without decay. Neurons are coded in {0;1}, and the hidden ETP layer has 7 hidden neurons

D. Best fit with the Help of the Real Evapotranspiration

At this stage of the exposal, we can consider that the neural network gave a fiable modelling of the shape of evapotranspiration curve, but not of its amplitude. Nevertheless, this last parameter can be adjusted taking into account the mean measured value of the real evapotranspiration (ETR) on one year. This value is simply expressed as the difference between the total amount of water in rainfall on the catchment and the total outflow at the outlet of this area. It can be noticed that the ETR can only be measured by this way and only cumulated upon an entire year. If we do this calculation on the 21 years being used to realize the learning, we get a mean value of the real evapotranspiration of 405 mm. On the other hand the estimated evapotranspiration with the weight decay gives us a value of 239 (without unity because neural network has normalized signals). So, the whole curve has to be multiplied by the factor 405/239 in order to model the evapotranspiration. When this adjustment is made, we obtain the modeling of ETP which has been calculated by the network and quantified by ETR and which is presented in Fig. 11.

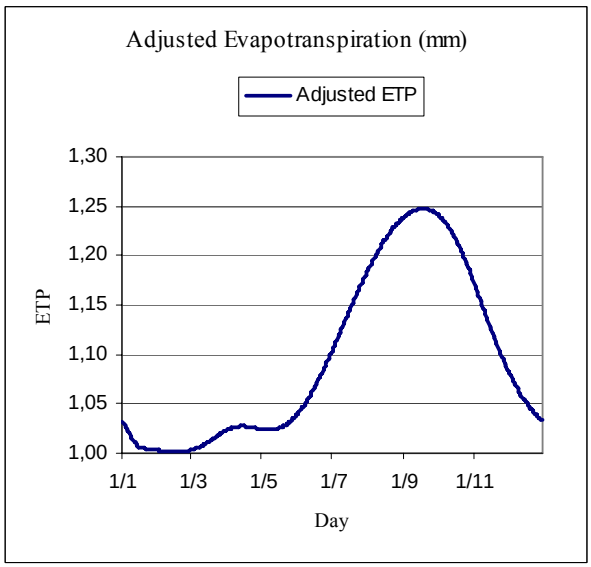

Fig. 11 Estimeted evapotranspiration after adjustment using real, measured over a year, evapotranspiration

Of course, such estimation of the evapotranspiration needs to be studied deeply in order to give sense to the double information that the network have extracted: first the date of the maximum of the ETP and secondly the value of the magnitude. Nevertheless, the fact that the network delivers constant estimations, independently of the number of hidden neurons is very satisfactorily. Moreover, because we can observe the shape of ETP signal inside the network, the presented method produces a network which can be called as "transparent" network.

VI. CONCLUSION

We first showed that because of their ability to identify non linear dynamical models, neural networks are good candidates for simulating rainfall-runoff relation. Moreover, using a specific architecture we showed that static models can be interpreted in terms of hydrogeology and provide an estimation of hidden variables like evapotranspiration. This last property is really innovative and opens up a wide field of fruitful research in earth science.

ACKNOWLEDGMENT

Authors want to thank warmly A. Mangin and D. D'Hulst from "Laboratoire Souterrain du CNRS" for helpfull discussions and providing data sets.

REFERENCES

[1] Y. Oussar, G. Dreyfus. "How to be a Gray Box: Dynamic Semi-physical Modeling". Neural Networks, invited paper, vol. 14, 2001, pp. 1161-1172.
[2] A. Johannet, A. Mangin, D. D'Hulst. "Subterranean Water Infiltration Modelling by Neural Networks: Use of Water Source Flow". In Proc. of ICANN, M. Marinaro and P.G. Morasso eds, Springer, 1994, pp. 1033-1036.
[3] G. Cybenko. "Approximation by Superposition of a Sigmoidal Function". Math. Ctrl Signal Syst, 2, 1989, pp. 293-342.
[4] H. Moradkhani, K. Hsu, H. V. Gupta, S. Sorooshian. "Improved Streamflow Forecasting Using Self-Organizing Radial Basis Function Artificial Neural Networks". Journal of Hydrology, 295, 2004, pp. 246-262.
[5] I. N. Daliakopoulos, P. Coulibaly, I. K. Tsanis. "Groundwater Level Forecasting Using Artificial Neural Networks". Journal of Hydrology 309, 2005, pp. 229-240.
[6] D. I. Jeong, Y. O. Kim. "Rainfall-Runoff models using artificial neural networks for ensemble streamflow prediction". Hydrological Processes, 19, 2005, pp. 3819-3835.
[7] B. Kurtulus M. Rasac. "Evaluation of the ability of an artificial neural network model to simulate the input-outpout responses of a large karstic aquifer : the Larochefoucault aquifer (Charente – France)". Hydrogeological Journal, 2006.
[8] A. P. Jaquin, A. Y. Shamseldin. "Development of rainfall-runoff models using Takagi-Sugeno fuzzy inference systems". Journal of Hydrology, 329, 2006, pp. 145-173.
[9] A.-L. Courbis, E. Touraud and B. Vayssade. "Water balance diagnosis based on a simulation tool". ENVIROSOFT'98, 1998, pp. 199-208.
[10] A. Johannet, P-A. Ayral, B. Vayssade. "Modelling non Measurable Processes by Neural Networks: Forecasting Underground Flow Case Study of the Cèze Basin (Gard – France)". CISSE, 2006.
[11] D. Rumelhart, G. Hinton, R. Williams. "Learning Internal Representation by Error Propagation". PDP, MIT Press, 1988.
[12] E.A. Bender. "Mathematical Method for Artificial Intelligence". IEEE Computer Society Press, 1996.
[13] A.J. Shepherd. "Second-Order Methods for Neural Networks". Springer, 1997.
[14] D.W. Marquardt. Journal of the Society for Industrial and Applied Mathematics, vol 11, pp. 431-441.
[15] W.H. Press, S.A.Teukolsky, W.T. Vetterling, B.P. Flannery. "Numerical recipies in C". Cambridge University Press, 1992.
[16] Narendra K. S., Parthasarathy K. "Gradient Methods for the Optimization of Dynamical Systems Containning Neural Networks". IEEE trans. neur. net., vol 2, 1991, n°2, pp. 252-262.
[17] Werbos P.J. "Backpropagation Throught Time : What it Does and How to Do It". Proc. IEEE, 78, N°10, 1990, pp. 1550-1560.
[18] Mangin A. (1970). "Le système karstique du Baget (Ariège)". Annales de Spéléologie, vol 25, fasc. 3.
[19] J.E. Nash, J. V. Sutcliffe. "River Flow Forecasting through Conceptual Model. Part I – A Discussion of Principles". Journal of Hydrology, 10, 1970, pp. 282-290.
[20] L. Oudin et al. "Which potential evapotranspiration input for a lumped rainfall-runoff model? Part 2 Towards a simple and efficient potential evapotranspiration model for rainfall-runoff modelling". Journal of hydrology, 303, 2005, pp. 290-306.
[21] Geman S. Bienenstock E. & Doursat R. "Neural networks and the bias/variance dilemma". Neural Computation 4, 1992, pp. 1-58.