Neural Comput & Applic (2005) 14: 66–77
DOI 10.1007/s00521-004-0450-z

ORIGINAL ARTICLE

Giorgio Corani · Giorgio Guariso

An application of pruning in the design of neural networks for real time flood forecasting

Received: 18 December 2003 / Accepted: 20 July 2004 / Published online: 28 January 2005

Abstract We propose the application of pruning in the design of neural networks for hydrological prediction. The basic idea of pruning algorithms, which have not been used in water resources problems yet, is to start from a network which is larger than necessary, and then remove the parameters that are less influential one at a time, designing a much more parameter-parsimonious model. We compare pruned and complete predictors on two quite different Italian catchments. Remarkably, pruned models may provide better generalization than fully connected ones, thus improving the quality of the forecast. Besides the performance issues, pruning is useful to provide evidence of inputs relevance, removing measuring stations identified as redundant (30–40% in our case studies) from the input set. This is a desirable property in the system exercise since data may not be available in extreme situations such as floods; the smaller the set of measuring stations the model depends on, the lower the probability of system downtimes due to missing data. Furthermore, the Authority in charge of the forecast system may decide for real-time operations just to link the gauges of the pruned predictor, thus saving costs considerably, a critical issue in developing countries.

Keywords Pruning · Feedforward neural networks · Optimal brain surgeon · Time-series prediction · Flood forecast

1 Introduction

An efficient flood alarm system may significantly improve public safety, and mitigate economical damage caused by inundations. Flood forecasting is undoubtedly a challenging field of operational hydrology, and a huge amount of literature has been published over the years; in particular, the rainfall-runoff relationship has been recognized to be nonlinear.

Since the flood warning system does not aim at providing explicit knowledge of the rainfall-runoff process, black box models have been widely used in addition to the traditional physically based models, which include a great number of parameters and require a fine-grained physical description of the area under study. In particular, over the last decade, artificial neural networks (ANN) have been increasingly used in hydrological forecasting practice (see, for instances of this [1], where tens of papers on the topic are quoted) and they were recognized as being able to provide more accurate flow predictions than traditional models [2, 3]. Furthermore, they are very fast at simulating and forecasting, as required for real-time operations.

However, as is well known, finding the optimal network architecture for a given problem is not a trivial task. Modelers usually work by trial and error, starting from an initial hypothesis about input variables, and then trying to determine the best network architecture for the current choice of inputs, assessing the fitness of many different model prototypes. Afterwards, they modify somehow the input set and restart searching for the architecture until a satisfactory degree of model performance is attained. Such a procedure is often time consuming and requires a great amount of experience and guesswork.

A second drawback of this procedure is that only fully connected architectures can be taken into account, since testing even partially connected models would lead to a combinatorial explosion of the number of trials needed. In all probability some of the many weights that translate the relations between inputs and outputs inside the network are less relevant. As a matter of fact, fully connected networks are not parsimonious: for example, a fully connected network with an input set of ten variables may contain few hundreds parameters, even if it has only one output variable.

G. Corani (✉) · G. Guariso
Dipartimento di Elettronica ed Informazione,
Politecnico di Milano, Via Ponzio 34/5,
20133 Milan, Italy
E-mail: corani@elet.polimi.it
Tel.: +39-02-23993562
Fax: +39-02-23993412

In this paper, we address the two criticisms mentioned above by means of pruning algorithms. Although they constitute a recognized research field in the theoretical neural networks area (see [4] for a review), they are still not widely used for applications. However, they have been exploited in fields different from water resources, such as in chemical processes identification [5], predictive microbiology [6] and near-infrared spectroscopy [7]. The basic idea of pruning algorithms is to start from a fully connected network, considered large enough to capture the desired input–output relationship. Then, they compute some measurements of the contribution made by an individual parameter to the problem solution, and consequently prune it from the network if it is the least influential one, to generate a new partially connected model, containing one parameter less. In this way, weights and neurons that are considered redundant are eliminated, thus significantly reducing the amount of guesswork needed for the model selection. Network architectures selected by pruning are very parsimonious because they may contain one order of magnitude fewer parameters than the initial one, and are highly optimized since they retain the representation power of the fully connected models.

Although pruning algorithms address computational issues in artificial neural networks, they have also a biological plausibility. During the learning process, functional modifications occur in the existing neural connections within the brain [8]. The modeling counterpart of such functional modifications is constituted, for artificial neural networks, by training algorithms, which adjust weights and biases during the calibration, keeping the network architecture unchanged. However, besides modifying the existing connections, the brain prunes some of them. In particular, according to the "*selectionist*" approach [9, 10], brain development comprises an initial period of over-production of neurons and connections, followed by a more prolonged period of eliminations of redundant ones. Indeed, pruning algorithms implement selectionism in artificial neural networks[1].

In hydrological forecasting practice, pruned predictors may lead to relevant advantages over those designed by trial and error.

Let us define $\mathscr{G}$ as the set of the measuring gauges in the basin; a "*complete predictor*" is a model whose input variables set includes some terms for each gauge in $\mathscr{G}$. If the basin has two rain gauges A and B, network inputs at time t may for example be constituted by past water levels $[y(t),\ y(t-1)]$ and rainfall measurements $[r_A(t),\ r_A(t-1),\ r_B(t),\ r_B(t-1)]$. If, for example, all the weights related to $r_A(t-1)$ are pruned from the network, such a variable is no longer required by the predictor. If both $r_A(t)$ and $r_A(t-1)$ are removed, gauge A does not contribute to the representation of the phenomenon given by the model. If the pruned predictor removes a subset g of measuring stations without decreasing the forecast accuracy, we can state that the informative content of $\mathscr{G}$ is equivalent to that of $(\mathscr{G}-g)$. Two main practical consequences follow from such a statement.

It is known that, especially during floods, gauges may experience measurement or transmission problems. A pruned predictor, which provides the same forecast accuracy of the complete one polling a smaller set of gauges, could have a definite advantage, in that it would be less subject to downtimes due to missing data, thus increasing the forecast availability.

Additionally, pruning may allow one to reduce network management costs if that is a critical issue: provided that a sufficiently long time series is available, because operating costs of the forecast system increase with the number of linked gauges (e.g., the fee required to access the data of the measuring network, or the cost of transmission line installation), the Authority in charge of the forecast system can first analyze all the available data off line, and then connect only the stations retained in the pruned predictor, with a considerable saving in costs. The situation is in fact quite common and such an approach may be particularly welcome in developing countries; an example of a similar situation can be found in upper Bangladesh [13].

In this paper, we first recall how ANN can be used to properly model the rainfall runoff process, and also present the pruning algorithm used. Then, we compare the performances and forecast availabilities of predictors designed through pruning and trial and error on two quite different Italian river basins.

2 Neural network modeling of the rainfall runoff relationship

The proposed forecast system is based, according to a typical hydrological modeling approach, on rainfall runoff models, which require the availability of rain gauges distributed within or near the watershed. The forecast is issued after the arrival of the rainfall events; since rainfall-runoff response times are of the order of few hours for small- and medium-sized basins (i.e., up to about 1,000 km^2), this is a natural boundary on the forecast lead times. In a recent work [14], lead times have been successfully increased up to 24 h even on small basins, acquiring data from radar, radiosondes and satellite and hence monitoring the synoptic evolution of the atmospheric conditions over a wide area. However, it should be noted that only in a few cases is such an advanced and expensive instrumentation available: in fact, most catchments are gauged simply with much cheaper hydrometers and rain-gauges, as is in the case studies presented in the paper.

[1] In [8] it is also pointed out that, among neuroscientists, such an approach is currently contradicted by "*constructivism*" [11], which describes the brain development as an increase, rather than a decrease, in the number of synapses. Constructive neural networks algorithms (see for instance [12]), which start with a small network and grow the network until a satisfactory is found, may be interpreted as implementing constructivism in artificial neural networks.

We model the rainfall-runoff process through a *feed forward* neural network with one hidden layer. The water level is the variable y to be predicted, but the approach would be exactly the same using flow rates, if water level-flow rate relationships were available.

The model input set comprises an *autoregressive* part of order p (i.e., p past water level measurement taken at the hydrometer), and several rainfall variables associated with the m available rain gauges $r_1,r_2,...,r_m$. It is evident that the number of rainfall terms used in the model may differ from one gauge to the other, but we assume here an equal number n of terms to simplify the notation. Such rainfall inputs are delayed by time lags $\tau_1,\tau_2,...,\tau_m$, respectively, in order to take into account the travel times from the rain gauges to the river. Hence, the *exogenous* part of input contains the variables $[r_1(t-\tau_1),r_1(t-\tau_1-1),...,r_1(t-\tau_1-n+1),\ ...,\ r_m(t-\tau_m),...,r_m(t-\tau_m-n+1)]$, and the network input layer comprises $(p+m*n)$ variables. We define u as the vector containing all such input variables.

Figure 1 shows a sample neural network structure with eight nodes in the hidden layer, in the case $p=n=2$, $m=3$. The model is a one-step-ahead predictor; thus, its output at time t is the water-level estimate $\hat{y}(t+1)$. In order to issue the forecasts of k steps ahead, one has to apply recursively the model, using some of the forecasts obtained at previous steps as autoregressive inputs, and updating of one unit at each step the time window spanned by the rainfall data. The maximum reachable forecast horizon $(t+k_{\max})$ is limited by the time-delay configuration and is given by $k_{\max}=(\min_m(\tau_1,\tau_2,...,\tau_m)+1)$; forecasting on farther temporal horizons would in fact require to know rainfall values after t.

At forecast time, all the data in the input layer are sent to each node in the hidden layer. As is well known, each node treats all the input layer data in the same manner, i.e., computing first a weighted sum of them and then processing the result through its activation function. For example, the j-th node weights the input layer data as follows:

$$z_j = \sum_{k=1}^{k=p+m*n} w'_{kj}u_k - \sigma_j \tag{1}$$

where w_{kj}' is the weight of input u_k, and σ_j is the neuron bias. The computed signal z_j is the argument of the activation function (namely, hyperbolic tangent) of the j-th neuron:

$$f(z_j) = 1 - \frac{2}{\exp(2z_j)+1} \tag{2}$$

Then, the outputs of all the hidden neurons are sent to the output layer, which contains a unique node. Here, the outputs of the hidden layer are weighted, returning the forecast:

$$\hat{y}(t+1) = \sum_j w''_j f(z_j) - \sigma_p \tag{3}$$

where w''_j is the weight of the output of the j-th hidden neuron at the output neuron, and σ_p represents the *bias* associated with the output neuron.

2.1 Time-delay configuration

Rainfall time delays have to be carefully configured, given their influence on the maximum reachable forecast horizon, as previously explained, and their importance to the rainfall-runoff modeling. The interaction with river authorities is an important step in order to establish the minimum forecast horizon useful for alarm purposes and hence to put some constraints on minimum time delays to be taken into account. From the

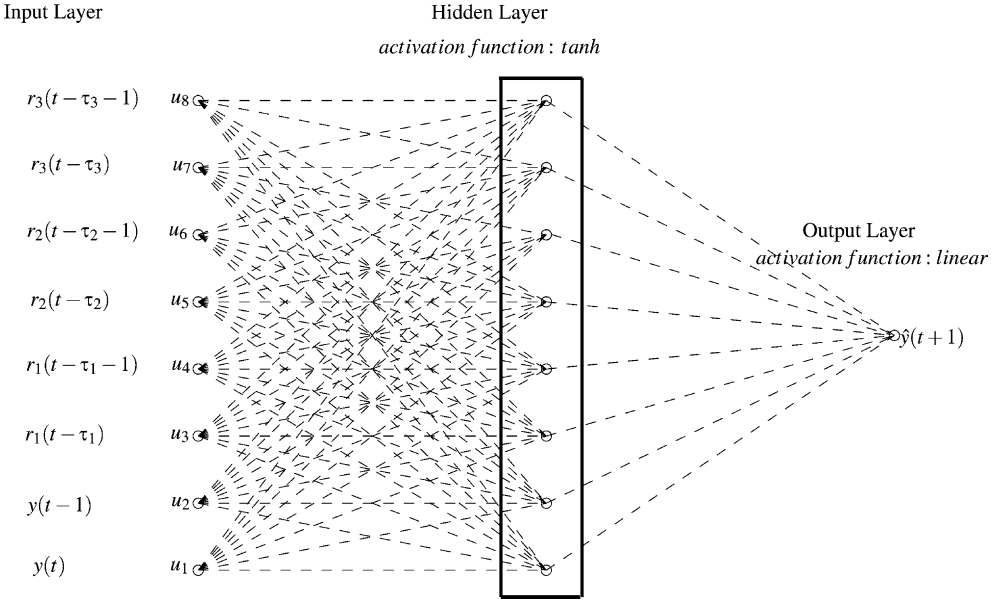

Fig. 1 Structure of a fully connected neural network

hydrological point of view, the determination of travel times is a very difficult task, since they may vary greatly depending on the saturation state of the basin: intuitively, the rainfall reaches the river more rapidly as the basin becomes wetter and the rainfall infiltration process less effective. Existing empirical hydrological formulas allow a rough estimate of the travel times, providing an idea of their order of magnitude in standard situations; however, they are not really useful in flood forecasting.

Analyzing time delays by trial and error is a very time-consuming process, if one tries all the possible time-delay combinations on all the available rain gauges, optimizing in each experiment the network architecture and the parameters estimate.

A possible approach is to calculate the cross correlations between rainfall and water level and then to configure the delays corresponding to the correlation peaks [15]. Using a more data-driven approach, we chose to select time delays on linear ARX models, and use them as a starting point for the neural network configuration. Thanks to the speed of their calibration, such linear models allow one in fact to analyze exhaustively all the feasible values of the delays vector $\bar{\tau} = [\tau_1, \tau_2, \ldots, \tau_m]$.

The complete procedure took a few minutes on a standard PC and was accomplished using the Matlab System Identification Toolbox [16].

2.2 Split sample approach

It is common knowledge that ANN can suffer from either underfitting or overfitting: an insufficiently complex network can fail to detect the relationships in complicated data sets, leading to underfitting; a too-complex network may lead, in contrast, to overfitting, also representing the noise in the calibration data. To tackle these problems, we adopted the split sample approach [17, 18], dividing the available data into three different subsets, and ensuring, as far as possible, that the statistical properties (mean, variance, maximum) are similar:

- A training set S_{Tr}, with cardinality N, used to estimate the parameters of the neural networks architectures
- A validation set S_{val}, with cardinality M, used to compare the architecture performances, and hence to select the optimal one. The union of S_{Tr} and S_{val} corresponds to the whole calibration set, in that the two sets are jointly exploited in the predictor choice
- A testing set S_{Te}, with cardinality Q, used to assess the model performances on previously unused data. Running the model on this set allows us to get a unbiased estimate of its generalization error.

Since data standardization makes the training algorithm numerically robust and leads to a faster convergence [19], means and variances of training time series are used to standardize the three data sets, according to the classical formula $x_{\mathrm{std}} = (x_i - \mu(x))/(\sigma(x))$.

2.3 Training objective function

We adopt a regularized objective function to be minimized during the training:

$$W(\theta) = \frac{1}{2N}\sum_{i\in \mathrm{Tr}}^{N} [y_i - \hat{y}_i(\theta)]^2 + \frac{1}{2N} D||\theta|| \tag{4}$$

where a term (*weight decay*) proportional to the norm of the weights $||\theta||$ is added to the mean-squared error criterion to improve the generalization capability of the model [20]. A convenient value of the weight-decay factor D has to be selected by trial and error.

The Levenberg–Marquardt training algorithm, recognized as suitable, both for speed and robustness, in the estimate of small- and medium-size networks, i.e., containing some hundreds of weights, has been used to minimize the objective in (4).

2.4 Architecture selection

Although the optimal architecture selection task is in principle combinatorial, it is usually accomplished by trial and error, varying the number of nodes in the model; however, as already anticipated, this kind of search cannot address partial connectivity, because of computational unfeasibility.

Instead, we use a pruning algorithm to select the neural network architecture. The basic idea of pruning algorithms is to start from a fully connected, over-parametrized network and then remove nonuseful connections. Weights are eliminated one at a time, and thus the pruning algorithm continues generating partially connected networks, the latter containing one parameter fewer than the former. A key issue of such algorithms is clearly the criterion which determines which weights should be eliminated from the network and how the remaining weights should be adjusted for best performances. Several approaches can be followed to this purpose. The simplest techniques are based on weight values analysis: for example, *magnitude-based* pruning [21] assumes that small weights are irrelevant. More complex algorithms try to quantify the relevance of a parameter through its influence on the performance function. Optimal brain damage (OBD) [22] estimates the increase in the training error resulting from the removal of each parameter (*parameter saliency*), and removes the one with the lowest saliency.

In this paper, we exploit the optimal brain surgeon (OBS) algorithm, which is a variant of OBD and which has been demonstrated to be better than both OBD and magnitude-based approaches, thanks to a far more reliable saliency estimate [23].

In the following, we describe the algorithm assuming, for the sake of simplicity, to adopt the nonregularized training function $E_{\mathrm{tr}} = 1/2N \sum_{i \in S_{\mathrm{Tr}}}^{N} (y_i - \hat{y}_i)^2$; the generalization for the regularized case can be found in [24].

The training error function is approximated by means of a second-order Taylor series expansion around the current parameter estimate $\bar{\theta}$:

$$E_{\mathrm{tr}} = E_{\mathrm{tr}}(\bar{\theta}) + \nabla E_{\mathrm{tr}}(\bar{\theta})\delta\theta + \frac{H}{2}(\delta\theta)^2 \tag{5}$$

where $\nabla E_{\mathrm{tr}}(\bar{\theta})$ is the gradient of the objective function evaluated in $\bar{\theta}$, and H is its Hessian, i.e., $H = (\partial^2 E_{tr}/\partial\theta^2)$. Assuming that the network is already trained, the gradient term may be neglected since $\bar{\theta}$ represents a minimum for E_{tr}. Therefore, we can estimate the error surface around $\bar{\theta}$ looking just at the quadratic term; the change in E_{tr} is given by

$$\partial E_{\mathrm{tr}} = E_{\mathrm{tr}}(\theta) - E_{\mathrm{tr}}(\bar{\theta}) = \delta\theta^T \frac{H}{2}\delta\theta \tag{6}$$

The weight whose elimination leads to the minimum error increase can be identified by finding the value of $\delta\theta$ which minimizes ∂E_{tr}, subject to constraint $e_i^T\delta\theta + \theta_i = 0$, where e_i is a unit vector in the weight space parallel to the w_i axis. According to such a constraint, the only adjustment allowed in θ is the deletion of a unique weight θ_i. The problem can be solved by means of Lagrange multipliers and gives the following estimate of the error increase , due to the elimination of the j-th weight:

$$\partial E_{\mathrm{tr}}(j) = \frac{1}{2}\frac{\hat{\theta}_j^2}{[H^{-1}]_{jj}} \tag{7}$$

where $[H^{-1}]_{jj}$ is the (j,j) element of the inverse of the Hessian matrix. Formula (7) is actually the saliency estimate for the *j-th* parameter. The consequent change to be applied to the weights vector is given by

$$\partial\theta = -\frac{\theta_j}{[H^{-1}]_{jj}} H^{-1} e_j. \tag{8}$$

The overall architecture selection task can be automated as follows:

1. Training of the initial fully connected architecture (in order to run the pruning algorithm, the initial network which has to be "large enough" to capture the input–output relationship; thus, a convenient starting point for the algorithm is a slightly over-parametrized network).
2. Ranking of parameters on the base of their saliences.
3. Elimination of the weight with the lowest saliency and generation of a new architecture.
4. Re-training of the network obtained (in principle, re-training should not be necessary, since formula (8) already provides an update rule for θ; however, since the parameters update is based on the series expansion (5), it is recommended that the network be retrained every time a weight or a small part of it (3–5%) has been eliminated), and evaluation of its performance on the training and validation set.
5. Back to step 2, until there are parameters left.

Given the nonlinearity of the problem, the estimation of the weights and their saliences may vary significantly depending on the weights of the initial fully connected network. In order to reduce the probability that the algorithm falls into a local minimum, it is necessary to train the initial network several times and to run a pruning session for every different set of weights. Figure 2 shows a sample of the error function behavior during an OBS session; the algorithm starts from an over-parametrized, fully connected network at the very right of the figure and moves to the left eliminating the parameters one at a time. E_{tr} shows, from right to left, a

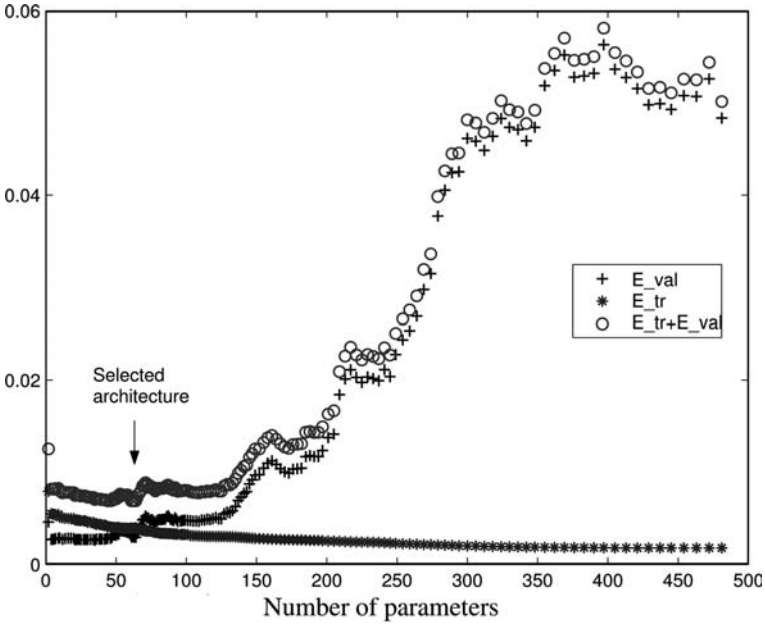

Fig. 2 Values of objective J (9) as a function of the number of parameters during a sample pruning sessions. E_{val} is the root-mean squared error on the validation set

monotonically increasing behavior, whereas the validation error E_{val} shows a roughly convex behavior.

Finally, the optimal model architecture is selected taking into account both training and validation performances, according to the following criterion:

$$J = \frac{1}{2N}\sum_{i\in S_{tr}}(y_i - \hat{y}_i)^2 + \frac{1}{2M}\sum_{i\in S_{val}}(y_i - \hat{y}_i)^2. \quad (9)$$

Although such a criterion also contains a training error term and may hence seem optimistically biased, selecting the architecture just on the base of the validation performances, as usually recommended [24], leads to poor architecture choices in our experiments. Since network performances are assessed during pruning by means of the 1-step prediction error, architectures minimizing the validation error usually contained just the very first autoregressive term $y(t)$. Indeed, using just $y(t)$ as input can lead to satisfactory results if $y(t+1)$ has to be predicted: rainfall takes some time (''concentration time'' in hydrologic terms) to affect the flow values. However, the aim of the system is to provide a prediction of the flood behavior over several time steps. As the forecast horizon becomes greater, rainfall increases its influence on the flow trend.

One could, in principle, set up different models (possibly relying on different input sets), each targeted to a specific forecast horizon h, minimizing the validation error on the h-steps prediction. However, such an approach involves a much greater modeling effort, requiring one to configure many different predictors. Our findings show that including a training error term (which favors complex architectures, since it is a decreasing function of the number of parameters) in the model selection criterion is quite effective in optimizing the generalization of models over multistep forecasts. However, we cannot provide a theoretical proof of such empirical evidence.

The architecture selected at the end of the procedure contains very few parameters (75–95% fewer than the initial one) and is therefore no longer prone to overfitting.

Usually, validation data cannot be used directly to improve the parameters estimate: they are in fact somewhat ''wasted'' when used just to prevent overfitting. On the contrary, one can take advantage of the parameter parsimony of pruned models, retraining the selected optimal architecture on the merging of both training and validation sets, without using early stopping. Such a procedure allows us to improve the generalization of the networks, whose statistical performances increase of some percent on the testing set.

Neural networks have been implemented, calibrated and pruned through the Neural Network Based System Identification Toolbox for Matlab [25][2] computation time required by a pruning session is about few minutes on a PC.

[2]The toolbox is freely available on the Internet: http://www.iau.dtu.dk/research/control/nnsysid.html

3 Results

Two different Italian catchments were considered in order to compare pruned and fully connected neural networks. We used hourly time series, containing those episodes showing a water-level increase higher than 100% within a 24-h time window.

In order to run the pruning algorithm, we developed an initial oversized completely connected network for each basin, each one capable of representing the basin-specific rainfall-runoff relationship properly. We trained such an initial network several times and, after each training, we ran the OBS algorithm many times, varying the weight decay factor between 0.0001 and 1. At the end, we chose the network which minimized the generalization criterion J and retrained it.

Judging the effectiveness of a flood forecasting system is a quite complex task. In principle, a correct measure would be a cost-benefit analysis, including the damage from an incorrect forecast (a false or a missing alarm). However, one has always to resort to statistical evaluations since benefits and costs also depend upon a number of external factors such as the way the information is distributed and how people react to it. Several indicators of the forecast effectiveness have been proposed in addition to the classical measures of the root mean-square error (RMSE) and correlation ρ between predicted and real flows. The model efficiency R^2, defined as

$$R^2 = \frac{F_0 - F}{F_0}$$

where F_0 is the variance of water levels $\sum_i (y_i - \mu(y))^2$ and F is the mean-square error $\sum_i (y_i - \hat{y}_i)^2$, is widely adopted. The ''prediction'' of the average value, which can be considered as the prediction available even in the worst case, then has an efficiency of 0. An efficiency value of 90% indicates a very satisfactory model performance while a value in the range 80–90% indicates a fairly good model [3].

Clearly, flood forecasting applications require a careful investigation of the model reliability in very high flow situations; indexes based on the error rate between observed and predicted peak are often used [26], though they do not take into consideration the delays of the predictions, which may be of critical relevance. Furthermore, such indicators only take into account peaks data, and their estimate may be biased given the small dataset. We adopted a more general ''high-flows-error rate'' criterion hf, computed taking into account only the K flows data exceeding the average value plus twice the square deviation. Such data cover about 2–5% of the whole time series, thus allowing a more reliable estimate.

The indicator provides an idea of the average relative error rate on high flows:

$$hf = \frac{1}{K} \sum_{y_i > \mu + 2\sigma} \left| \frac{y_i - \hat{y}_i}{y_i} \right|$$

3.1 Olona river

The first case study refers to the basin of river Olona (see Fig. 3), located in Lombardy, Northern Italy. The average flow is about 2.5 m^3/s, while the maximum expected flow over a time period of 10 years is about 108 m^3/s [27]. The size of the area is about 200 km^2 at the closing section (Castellanza), and the basin is divided into two parts: the upper one, mountainous and weakly anthropized, and the lower one, flat and strongly urbanized. The cross correlations between water levels and rainfall are 30% lower than those of other basins located in the same region [28]; this is due to many artificial inflows that the river receives in the lower part of the basin and to a series of small reservoirs (with an overall capacity of about 208×10^3 m^3), built in order to mitigate flood events, which alter the natural behavior of the basin.

In addition to the Castellanza hydrometer, which measures water levels, three rain gauges are available on the basin (Fig. 3). They are located at:

- Arcisate, in the mountainous part of the basin
- Varese, at the beginning of the urbanized area
- Vedano, in the urbanized area.

Data refer to 13 events (with an overall length of about 1,100 hourly steps) that occurred within the period 1999–2001; training, validation and testing sets contain about 500, 200, 400 patterns, respectively. Water level averages over training, validation and testing sets are, respectively, 82.3, 89.6, 83.2 cm; ratios between water levels averages and standard deviations are, respectively, 3.1, 2.2, 2.3.

Fig. 3 The Olona catchment

We evaluated the model performances on a 3 h-forecast horizon, judged as suitable by civil protection technicians.

3.1.1 Models

We started selecting the time delays on the ARX models, as described previously in the paper. In order to provide the forecast on a time horizon of at least 3 h, we had to take into account only time delays equal to or longer than 2 h. The ARX model which minimizes the criterion J presents an identical 2-h time delay for all the rain gauges.

Then, we trained many different neural networks, fixing such time delays and looking for the optimal complexity by trial and error, as usually done in the literature. In this case, we exploited the early stopping technique [17] in order to avoid overfitting, evaluating the objective function at each iteration on training and validation sets, and stopping the training in correspondence with the minimum validation error. Finally, we chose the network architecture which minimized the calibration objective J. The selected model (Fig. 4a) had three nodes in the hidden layer; its input variables had an order of 3 for an overall input set comprising twelve elements. Such a model constituted the term of comparison for the pruned predictor.

In order to initialize the pruning algorithm, we provided a completely connected network, slightly oversized both in the input and in the hidden layer with respect to the optimal predictor previously found. The optimal neural network selected through pruning is presented in Fig. 4b. The model was partially connected and contained only 10 parameters, compared to the about 180 of the initial architecture and to the 43 of the completely connected predictor.

The structure of the pruned predictor did not contain any parameter related to the Varese rainfall gauge (network input r_2 in Fig. 4b), and required us to acquire real-time data just from the Arcisate and Vedano rain gauges and from Castellanza hydrometer. Varese was highly cross correlated with both Arcisate and Vedano, since it is located between them; on the other hand, Arcisate and Vedano, located at the opposite sides of the basin, showed a lower cross correlation. Hence, correlation analysis suggests that the information content of Varese gauge was mostly redundant, as demonstrated by pruning results (Table 1).

The ''autoregressive memory'' of the basin appears to be of order 1, since the model retained only one term, namely $y(t)$, out of five initially provided. However, such an autoregressive term has a great modeling relevance: it was the very last input removed from the network before the pruning algorithm was finished.

The two predictors showed the same forecast efficiency on the testing set, which was the most significant

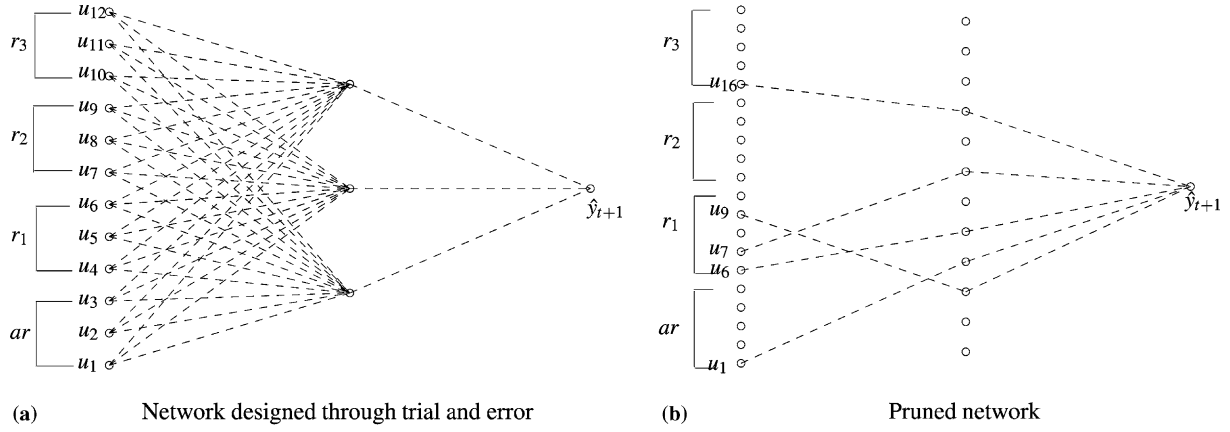

Fig. 4 Neural network architectures for the Olona river. Input variables in group *ar* are of autoregressive type, while those in groups r_1, r_2 and r_3 to the Arcisate, Varese and Vedano rainfall gauges. Each variable in a group refers to a specific relative time instant (t, t-1, etc.)

Table 1 Cross correlations between rainfall data evaluated in correspondence of cross-correlograms maxima

Gauges couple	Cross–correlation
(Arcisate–Varese)	0.75
(Arcisate–Vedano)	0.65
(Varese–Vedano)	0.69

for performances evaluation. Remarkably, the pruned model overperformed the connected one on the high-flow indicator, thus showing the effectiveness of the re-training on the extended dataset (Table 2).

Sample plots of observed versus 3-h ahead forecasted water level are shown in Fig. 5a, b for training and testing respectively, while Fig. 5c, d presents the scatter plots on the whole time series.

3.2 Forecast availability

Since a predictor cannot issue the forecast if any of its input variables is missing, we investigate in this section in which way the number of gauges included in a model affects the forecast availability. To this purpose, we used the whole available time series, containing data recorded continuously for about one and a half years (8,040 time steps).

Table 2 Results for the 3 h-ahead prediction on the Olona basin

	Connected network (three rain gauges)		Pruned network (two rain gauges)	
	Training	Testing	Training-validation	Testing
Time series averaged indicators				
RMSE	0.012	0.046	0.011	0.035
R^2	0.92	0.84	0.89	0.85
ρ	0.96	0.92	0.94	0.93
High flows analysis				
hf	0.12	0.24	0.19	0.19

Models using a single gauge have a forecast availability of about 94% on average, while the models with two or three gauges have about 89% and 86.5%. If malfunctioning events were actually statistically independent, forecast availabilities of models with two or three gauges would be $94^2 = 88\%$ and $94^3 = 83\%$, respectively. It therefore appears that forecast availabilities are higher than they would be in the case of pure statistical independence. This is physically explainable for the small area of the basin, which increases the statistical dependence (and hence the temporal overlap) between malfunctioning episodes at different gauges. One could roughly conclude that polling one gauge fewer in real-time operations results in an improvement between 2.5% and 5% in the forecast availability. Such an improvement may actually be underestimated, since it is computed on a continuous time series, therefore containing many low flow periods, while the forecast system is expected to work in more difficult situations, which stress the transmission and measuring system, thus increasing the probability of missing data.

Although the pruned predictor allows a higher availability than the complete one, it should be completed by a set of of "emergency predictors" which, using for example just a single gauge as input data, would allow us to issue the forecast, with a lower degree of accuracy, until at least one gauge on the basin is reachable. Within this framework, the pruned predictor would still remain preferable to the complete one, allowing us to lessen the need for emergency predictors.

3.3 Tagliamento river

The second case study refers to the basin of river Tagliamento, located in Friuli, North-East Italy (Fig. 6). The average flow is about 90 m^3/s, while the maximum flood peak over the last decades reached 4,000 m^3/s in 1966.

The basin closed at Venzone measures about 1,950 km^2; the monitoring system comprises the

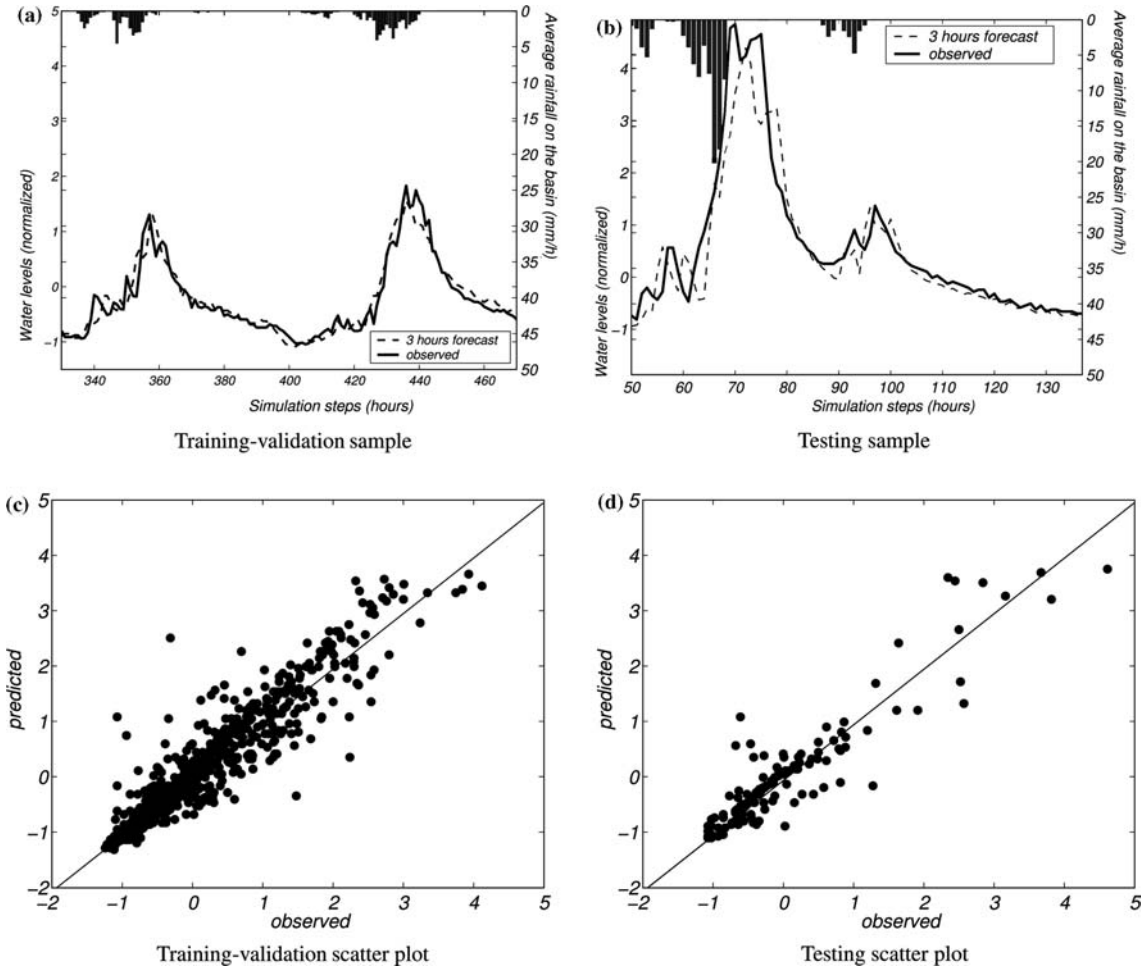

Fig. 5 Three-hour-ahead simulations. The reversed *y*-axis (*right-hand side*) in **a**, **b** is an estimate of the average rainfall on the basin

Venzone hydrometer, at the end of the mountain district, and five rain gauges (Paularo, Ampezzo, Pesariis, Resia and Moggio), each located in a different subbasin. The dataset comprises 20 flood events that occurred over the years 1978–1996 for an overall length of 2,000 hourly time steps. Training, validation and testing sets contain, respectively, about 1,000, 600 and 400 time steps; water level average values on the same sets are 111.7, 109.8 and 134 cm respectively, while ratios between water level averages and standard deviations are 1.5, 2.3 and 2.2.

A feedforward neural network for flood forecasting has been already proposed in [15]. Such a network is completely connected, and its input variables set includes all the gauges in the basin. The model architecture comprises ten nodes in the hidden layer and contains five autoregressive terms and 15 terms for each rain gauge in the input layer. The model is constructed to directly forecast at k time steps ahead (*direct predictor*). The same paper also states that a forecast horizon of 5 h in advance can be considered satisfactory for this basin, and that no seasonality is clearly detectable in the available data. We used such results as a reference in the evaluation of the proposed pruned predictor.

Initially using the results of ARX models analysis, time delays of 8, 14, 10, 11 and 6 h were set for the rain gauges located at Ampezzo, Moggio, Paularo, Pesariis

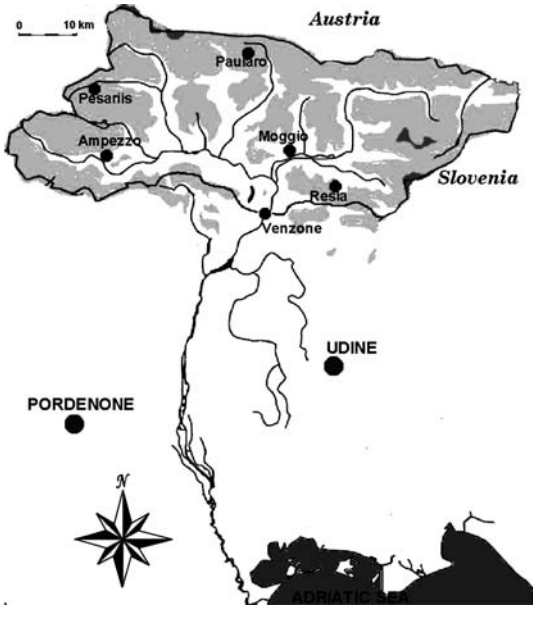

Fig. 6 The Tagliamento catchment

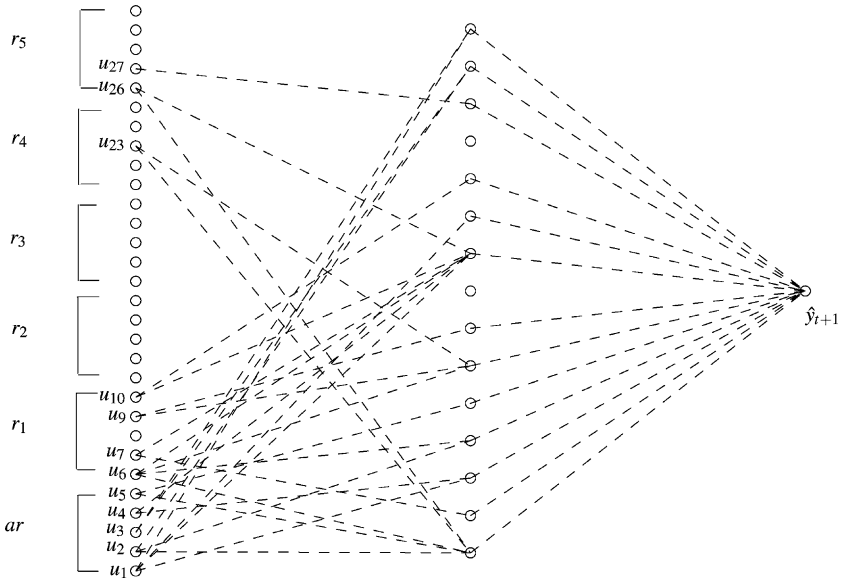

Fig. 7 The architecture of the pruned neural predictor. *ar* refers to autoregressive water-level measurements (Venzone), r_1, r_2, r_3, r_4 and r_5 to Ampezzo, Moggio, Paularo, Pesariis and Resia rainfall measurements

and Resia, respectively. Then, we ran the pruning algorithm, starting from a network with 30 input variables (five autoregressive and five terms for each rain gauge) and 15 nodes in the hidden layer. The optimal pruned predictor is presented in Fig. 7. It contains 44 parameters, compared to about 500 of the initial architecture, and to about 800 in [15].

In this case, it is difficult to explain pruning results analyzing rainfall data through cross correlation (Table 3). However, it should be pointed out that correlation coefficients provide information about linear relationships between two variables, while here we were interested in comparing the informative content of different sets of variables: a quite different matter. The first evidence is that, since gauges were spread on a much wider basin than Olona, the cross correlations are significantly lower. Moggio and Paularo gauges were pruned from the model, even if they did not show high cross-correlation coefficients with the other stations, and hence they seemed to provide a valuable fresh informative content. Furthermore, Ampezzo and Pesariis gauges were retained in the model, although they were the most cross correlated ones. The correlations between water levels and the rainfall data were around 0.5 for all the gauges, and hence did not allow us to determine the gauges' relevance to the variable to be predicted. In this case the pruned network, thanks to its nonlinearity, captured an informative redundancy between the rain gauges data which the correlation analysis, able to detect only linear relationships, did not recognize. The autoregressive memory of the basin appeared to be of the order of 5.

Table 3 Cross correlations between rainfall data evaluated in correspondence of cross-correlograms maxima

Gauges couple	Cross correlation	Gauges couple	Cross correlation
(Ampezzo–Moggio)	0.43	(Moggio–Pesariis)	0.43
(Ampezzo–Paularo)	0.55	(Moggio–Resia)	0.48
(Ampezzo–Pesariis)	0.69	(Paularo–Pesariis)	0.48
(Ampezzo–Resia)	0.42	(Paularo–Resia)	0.49
(Moggio–Paularo)	0.46	(Pesariis–Resia)	0.41

The results on the 5-h-ahead forecast for the completely connected model and the pruned one are compared in Table 4. The two models show very close performances on both training and testing set. Similarly to the Olona case study, an improvement of about 1–2% of the testing performances was obtained retraining the network on the merging of training and validation sets.

Plots of observed versus 5-h ahead forecasted water level are shown in Fig. 8a, b for training and testing, respectively, while Fig. 8c, d presents the scatter plots on the whole time series.

Data on the reliability of the gauges on the Tagliamento river were not available and this prevented any quantitative evaluation of the improvement in the forecast availability between the complete and the pruned predictor. However, advantages may be expected to be higher than in the Olona case, since we pruned two gauges here instead of one, and because the higher distances between the gauges certainly increased the statistical independence of the malfunctioning events at different gauges.

4 Conclusions

Designing a neural network through pruning instead of trial and error allows us to lower the number of parameters by about one order of magnitude, overcoming overfitting problems. Furthermore, pruned models deliver a clear evidence of input variables relevance, removing redundant inputs; indeed in our case studies, we only had to poll just a limited number (60–70%) of the available rain gauges in the basins. Although relying on less information, their forecast

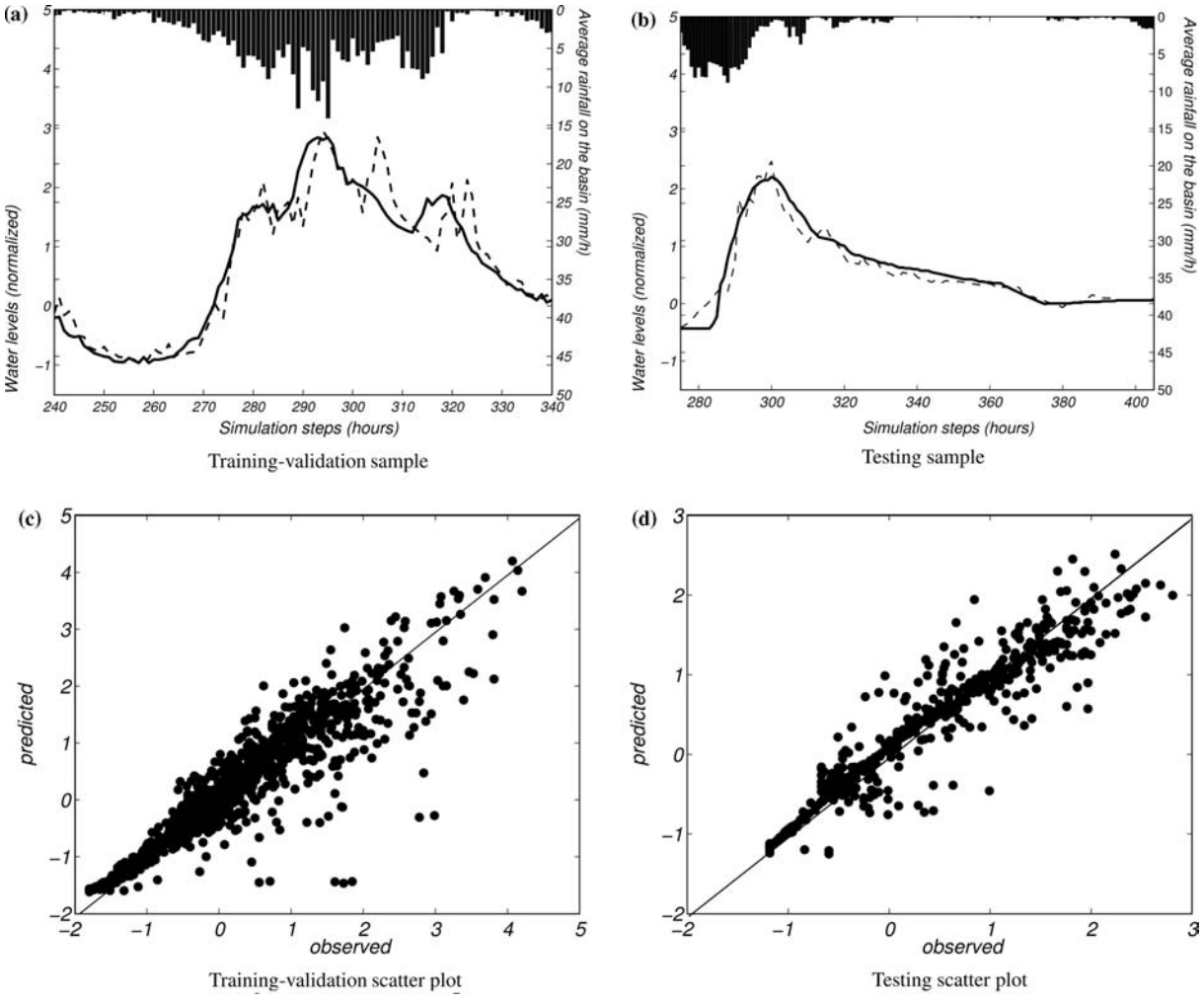

Fig. 8 Five-hour-ahead simulations. The reversed *y*-axis (*right-hand side*) provided in **a**, **b** is an estimate of the average rainfall on the basin

accuracy was equivalent to fully connected models containing many more parameters and linked to each available rain gauge on the basin. Remarkably, pruned models may also generalize better than fully connected networks, taking advantage of a final retraining on the merge of training and validation data, no longer requiring us to waste precious data for early stopping purposes. Using a reduced set of rainfall gauges can be very convenient in real-time operations, making the system less subject to downtimes due to missing data. A rough estimate for the Olona basin showed that the removal of a single gauge led to a forecast availability improvement ranging between 2.5% and 5%. These values are expected to increase on wider basins, where the statistical dependence of malfunctioning episodes at different gauges decreases.

To cope with these situations of data insufficiency, one must set up a set of emergency models. Such emergency models may use just one or few of the available gauges and thus allow one to issue a forecast, with a reduced level of accuracy, until at least some gauge is reachable in the basin. Operating the pruned predictor instead of the

Table 4 Results on the 5-h-ahead prediction on the Tagliamento basin. Training performances of the re-trained network are obtained on the concatenation of training and validation set

	Connected network [15] five rain gauges		Pruned network (retrained) threes rain gauges	
	Training	Testing	Training-validation	Testing
Time series averaged indicators				
RMSE	–	–	0.0087	0.0130
R^2	0.89	0.85	0.87	0.88
ρ	–	–	0.93	0.94
High flows analysis				
hf	–	–	0.26	0.18

complete one provides equivalent performances and higher reliability, reducing the need for the less precise emergency models, and thus increasing the average accuracy of the overall system.

Pruning results may be important also for cost reductions: provided that adequately long time series are available. Since the operating costs of the forecast system increase with the number of the linked gauges (e.g., because of the fee required to access the data of the measuring network, or the cost of transmission line installation), the Authority in charge of the forecast system can link just the stations retained in a predictor designed by pruning, with a clear reduction in the budget.

Explaining why certain gauges are pruned from the model is not always possible through correlation analysis, since it captures only linear relationships.

A desirable development of the algorithm would be the possibility of removing groups of parameters instead of a unique one; in this way, pruning could actually be used for feature selection, thus constituting a great tool in neural network modeling.

Acknowledegments The authors thank M. Molari and N. Quaranta, Civil Protection Service of the Lombardy Region, for supplying the data of the Olona river, M. Campolo and A. Soldati, University of Udine, for the data of the Tagliamento river.

References

1. Maier HR, Dandy CG (2000) Neural networks for the prediction and forecasting of water resources variables: a review of modelling issues and applications. Environ Mod Soft 15:101–124
2. Hsu VK, Gupta S, Sorooshian S (1995) Artificial neural network modeling of the rainfall runoff process. Water Resour Res 31(10):2517–2530
3. Shamseldin AY, O'Connor KM, Liang KM (2001) Methods for combining the outputs of different rainfall-runoff models. J Hydrol 245:196–217
4. Reed R (1993) Pruning algorithms-a survey. IEEE Trans Neural Netw 4(5):740–747
5. Henrique HM, Lima EL, Seborg DE (2000) Model structure determination in neural network models. Chem Eng Sci 55(22):5457–5469
6. García-Gimeno R, Hervás-Martínez C, de Silóniz MI (2002) Improving artificial neural networks with a pruning methodology and genetic algorithms for their application in microbial growth prediction in food. Int J Food Microbiol 72(1–2):19–30
7. Poppi RJ, Massart DL (1998) The optimal brain surgeon for pruning neural network architecture applied to multivariate calibration. Anal Chim Acta 375(1–2):187–195
8. Quinlan PT (1998) Structural change and development in real and artificial neural networks. Neural Netw 11:577–599
9. Edelman G (1987) Neural Darwinism: the theory of neuronal group selection. Basic Books, New York
10. Changeux J-P, Courrége P, Danchin A (1973) A theory of the epigenesis of neuronal networks by slective stabilisation of synapses. Proc Natl Acad Sci USA 70(10):2974–2978
11. Quartz SR, Sejnowski TJ (1997) The neural basis of cognitive development: a constructivist manifesto. J Behav Brain Sci 45:35–41
12. Moody J, Antsaklis PJ (1996) The dependence identification neural networks construction algorithm. IEEE Trans Neural Netw 7:3–15
13. Liong SY, Lim WH, Paudyal GN (2000) River stage forecasting in Bangladesh: neural network approach. J Comput Civil Eng 14(1):1–8
14. Kim G, Barros AP (2001) Quantitative flood forecasting using multisensor data and neural networks. J Hydrol 246:45–62
15. Campolo M, Andreussi P, Soldati A (1999) River flood forecasting with a neural network model. Water Resour Res 35(4):1191–1197
16. Inc MathWorks (2000) System identification toolbox user guide
17. Bishop CM (1995) Neural networks for pattern recognition. Oxford University Press, Oxford
18. Haykin S (1995) Neural networks: a comprehensive foundation. Macmillan Coll, New York
19. Le Cun Y, Kanter I, Solla S (1991) Eigenvalues of covariance matrices: application to neural network learning. Phys Rev Lett 14(1):2396–2399
20. Girosi F, Jones M, Poggio T (1995) Regularization theory and neural networks architectures. Neural Comput 7:219–269
21. Hertz J, Krogh A, Palmer RG (1991) Introduction to the theory of neural computation. Addison Wesley, Reading
22. Le Cun Y, Denker JS, Solla SA (1990) Optimal brain damage. In: Touretzky DS (ed) Advances in neural information processing systems, vol 2. Morgan Kaufmann, San Francisco, pp 598–605
23. Hassibi B, Stork DG (1993) Second order derivatives for network pruning: optimal brain surgeon. In: Giles CL, Hanson SJ, Cowan JD (eds) Proceedings of advances in neural information processing system, pp 164–171
24. Norgaard M, Ravn O, Poulsen NK, Hansen LK (2000) Neural networks for modelling and control of dynamic systems. Springer, London
25. Norgaard M (2000) Neural network based system identification toolbox. Technical Report 00-E-891, Department of Automation, Technical University of Denmark
26. Chang FJ, Liang J, Chen Y (2001) Flood forecasting using radial basis function neural networks. IEEE Trans Syst Man Cybern C 31(4):530–535
27. Zampaglione (1995) Progetto di massima per il riequilibrio idraulico ambientale del fiume Olona: Relazione Idrologica
28. Corani G, Guariso G (2001) Stocasthic models for flood forecasting on rivers Brembo and Olona. Technical Report 2001.50, Department of Electronics and Information, Polytechnic of Milan