

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

Giorgio Corani(*), Giorgio Guariso(*)

(*) Dipartimento di Elettronica ed Informazione
Politecnico di Milano
corani(guariso)@elet.polimi.it

==PREPRINT==
Neural Computing and Applications 14(1): 66-77 (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 performances issues, pruning is useful to provide an evidence of inputs relevance, removing measuring station 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 be not 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 to link for real time operations just the gauges of the pruned predictor, thus considerably saving costs, a critical issue in developing countries.

keywords: pruning, feed-forward neural networks, optimal brain surgeon, time series prediction, flood forecast

Introduction

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

Since the flood warning system does not aim at providing an explicit knowledge of the rainfall-runoff process, black box models have been widely used besides 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 the hydrological forecasting practice (see, for instances [1], where tens of paper on the topic are quoted) and they were recognized to be able to provide more accurate flow predictions than traditional models ([2, 3]). Furthermore they are very fast in simulating and forecasting, as required for real time operations.

However, as 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 performances is attained. Such kind of 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 also partially connected models would lead to a combinatorial explosion of the number of trials needed. On the contrary, it is quite likely that 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 10 variables may contain few hundreds parameters, even if it has only an output variable.

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 measure of the contribution of each parameter to the problem solution, and consequently prune the less influential one from the network, to generate a new partially connected model, containing one parameter less. In this way, weights and neurons 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 less 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 modelling 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 networks¹.

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

Let us define \mathcal{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 \mathcal{G} . If the basin is instrumented through 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 \mathcal{G} is equivalent to that of $(\mathcal{G} - g)$. 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 requiring to poll 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 to reduce network management costs if they are a critical issue: provided that long enough time series are available, and that the exercise costs of the forecast system grows up with the number of the linked gauges (for example, because of the fee required to access the data of the measuring network, or of transmission lines installation), the Authority in charge of the forecast system can first analyze off line all the available data, 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 welcomed in developing countries; an example of a similar situation can be found in the upper Bangladesh [13].

In this paper we first recall how ANN can be used to properly model the rainfall runoff process, presenting also 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.

¹In [8] it is also pointed out that, among neuroscientists, such approach is currently adversed 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.

Neural network modelling of the rainfall runoff relationship

The proposed forecast system is based, according to a typical hydrological modelling approach, on rainfall runoff models which require the availability of rain gauges distributed within or near the watershed. The forecast are issued after the arrival of the rainfall events; since rainfall-runoff response times are in order of few hours for small and medium sized basins (i.e. under or about 1000 km^2), this is a natural bound on the forecast lead times. In a recent work [14], lead times have been successfully increased up to 24 hours even on small basins, acquiring data from radar, radiosondes and satellite and hence monitoring the synoptic evolution of the atmospheric conditions on a wide area. It should be remarked, however, that just in few cases such an advanced and expensive instrumentation is 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.

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 levels-flows rates 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, \dots, 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 $\tau_1, \tau_2, \dots, \tau_m$ 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 $[r_1(t - \tau_1), r_1(t - \tau_1 - 1), \dots, r_1(t - \tau_1 - n + 1), \dots, r_m(t - \tau_m), r_m(t - \tau_m - n + 1)]$, 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 8 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 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 delays configuration

and is given by $k_{max} = (\min_m(\tau_1, \tau_2, \dots, \tau_m) + 1)$; 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:

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

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:

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

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:

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

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 to the output neuron.

Time delays configuration

Rainfall time delays have to be carefully configured, given their influence on the maximum reachable forecast horizon, as previously explained, and their importance on the rainfall-runoff modelling. 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 a hydrological point of view, the determination of travel times is a very difficult task, since they may vary heavily depending on the saturation state of the basin: intuitively, the rainfall reaches quicker the river 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 would lead to a very time consuming process, if one would try exhaustively all the possible time delays 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 rainfalls 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 starting point for the neural network configuration. Thanks to the speed of their calibration, such linear models allow in fact to analyze exhaustively all the feasible values of the delays vector $\bar{\tau} = [\tau_1, \tau_2, \dots, \tau_m]$.

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

Split sample approach

It is common knowledge that ANN can suffer from either underfitting or overfitting: a not sufficiently complex network can fail to detect the relationships in complicated data sets, leading to underfitting; a too complex network may lead on the contrary to overfitting, representing also 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 the similarity of statistical properties (mean, variance, maximum) between them:

- 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 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 $x_{std} = \frac{x_i - \mu(x)}{\sigma(x)}$.

Training objective function

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

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

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 through 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 eq. (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.

We use instead 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 removing non useful connections. Weights are eliminated one at a time, and thus the pruning algorithm continues generating partially connected networks, the latter containing one parameter less 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 on the training error resulting from the removal of each parameter (*parameter saliency*), and removes that 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 non-regularized training function $E_{tr} = \frac{1}{2N} \sum_{i \in S_{Tr}}^N (y_i - \hat{y}_i)^2$; 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 $\bar{\theta}$:

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

where $\nabla E_{tr}(\bar{\theta})$ is the gradient of the objective function evaluated in $\bar{\theta}$, and H is its Hessian, i.e. $H = \frac{\partial^2 E_{tr}}{\partial \theta^2}$. Assuming 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:

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

The weight whose elimination leads to the minimum error increase can be identified 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 constrain, the only allowed adjustments in θ is the deletion of a unique weight θ_i . The problem can be solved by means of Lagrange multipliers and returns the following estimate of the error increase, due to the elimination of the j -th weight:

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

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:

$$\partial\theta = -\frac{\theta_j}{[H^{-1}]_{jj}} H^{-1} e_j \quad (8)$$

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 obtained network (in principle, re-training could 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 to retrain the network every time a weight or a small part of them (3-5%) has been eliminated), and evaluation of its performances on training and validation set;
5. back to step 2, until there are parameters left.

Given the non linearity 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 required to train several times the initial network 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 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:

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

Though such a criterion contains also 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 contained usually 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 farther, rainfall increases its influence 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 modelling effort, requiring to configure many different predictors. Our

findings show that including a training error term (which favor 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 multi-step ahead forecasts. We cannot however provide a theoretical proof of such an empirical evidence.

The architecture selected at the end of the procedure contains very few parameters (75% to 95% less 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" 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 to improve the generalization of the networks, whose statistical performances increases of some points 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.

Results

Two different Italian catchments have been 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-hours time window.

In order to run the pruning algorithm, we developed for each basin an initial oversized completely connected network, able to properly represent the basin specific rainfall-runoff relationship. We trained several times such initial network 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 minimizes 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 damages from an incorrect forecast (a false or a missing alarm). However, one has always to resort to statistical evaluations since benefits and costs also depends upon a number of external factors such as the way the information is distributed and how people reacts to it. Several indicators of the forecast effectiveness have been proposed besides the classical measures of the root mean square error

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

(*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 largely adopted. The “prediction” of the average value, which can be considered as the prediction available even in the worst case, has then 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 claim for careful investigation of the model reliability in very high flows 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 take into account only 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 about the average relative error rate on high flows:

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

Olona River

The first case study refers to the basin of river Olona (see Figure 3), located in Lombardia, Northern Italy. The average flow is about $2.5 \text{ m}^3/\text{sec}$, while the maximum expected flow over a time period of 10 years is about $108 \text{ m}^3/\text{sec}$ [27]. The area sizes 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 rainfalls are 30% lower compared to other basins located in the same region [28]; this is due to many artificial inflows which the river receives in the lower part of the basin and to a series of small reservoirs (with an overall capacity of about $208 * 10^3 \text{ mc}$), built in order to mitigate flood events, which alter the natural behavior of the basin.

Besides Castellanza hydrometer, which measures water levels, three rain gauges are available on the basin (Fig. 3). They are located in:

- 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 1100 hourly steps) occurred within the period 1999-2001; training, validation and testing sets contain respectively about 500, 200, 400 patterns. 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.

We will evaluate the model performances on a three hours forecast horizon, judged as suitable by Civil Protection technicians.

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 hours, we had to take into account only time delays equal or higher than 2 hours. The ARX model which minimizes the criterion J presents an identical two-hours 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 exploit 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 minimizes the calibration objective J . The selected model (Fig. 4a) has 3 nodes in the hidden layer; its input variables have order 3 for an overall input set comprising 12 elements. Such a model will constitute 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 is partially connected and contains 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 does not contain any parameter related to Varese rainfall gauge (network input r_2 in figure 4b), and requires to acquire real time data just from Arcisate and Vedano rain gauges and from Castellanza hydrometer. Varese is 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, show a lower cross-correlation. Hence, correlation analysis suggests that the information content of Varese gauge is mostly redundant, as demonstrated by pruning results.

The “autoregressive memory” of the basin appears to be of order 1, since the model retains only one term, namely $y(t)$, out of five initially provided. However, such an autoregressive term has a great modelling relevance: it is the very last input removed from the network before pruning algorithm ends up.

The two predictors show the same forecast efficiency on the testing set, which is the most significant for performances evaluation. Remarkably, the pruned model overperforms the connected one on the high-flow indicator, thus showing the effectiveness of the re-training on the extended dataset.

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

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 year (8040 time steps).

Models using a single gauge have a forecast availability of about 94% on average, while the models with two or three gauges 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 hence 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 less in real time operations results in an improvement between 2.5 – 5% in the forecast availability. Such an improvement may actually be underestimated, since it is computed on a continuous time series, containing hence 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 “emergency predictors” which, using

for example just a single gauge as input data, would allow 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 to decrease the need for emergency predictors.

Tagliamento River

The second case study refers to the basin of river Tagliamento, located in Friuli, North-East of Italy (Fig. 6). The average flow is about $90 m^3/sec$, while the maximum flood peak over the last decades reached $4000 m^3/sec$ in 1966.

The basin closed at Venzone measures about $1950 km^2$; the monitoring system comprises the 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 occurred over the years 1978-1996, for an overall length of 2000 hourly time steps. Training, validation and testing sets contain respectively about 1000, 600, 400 time steps; water level average values on the same sets are 111.7, 109.8, 134 *cm* respectively, while ratios between water level averages and standard deviations are 1.5, 2.3, 2.2.

A feed forward neural network for flood forecasting has been already proposed in [15]. Such network is completely connected, and its input variables set includes all the gauges on the basin. The model architecture comprises 10 nodes in the hidden layer and contains 5 autoregressive terms and 15 terms for each rain gauge in the input layer. The model is built to directly forecast at k time steps ahead (*direct predictor*). The same paper also states that a forecast horizon of 5 hours in advance can be considered satisfactory on this basin, and that no seasonality is clearly detectable in the available data. We will use such results as reference in the evaluation of the proposed pruned predictor.

Using initially the results of ARX models analysis, time delays of 8, 14, 10, 11, 6 hours have been finally set, for the rain gauges located at Ampezzo, Moggio, Paularo, Pesariis and Resia respectively. Then, we ran the pruning algorithm, starting from a network with 30 input variable (5 autoregressive and 5 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 the about 500 of the initial architecture, and to the about 800 in [15].

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

The results on the 5-hours-ahead forecast for the completely connected model and the pruned one are compared in Tab.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 has been obtained retraining the network on the merging of training and validation sets.

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

Data on the reliability of the gauges on the Tagliamento river were not available and this prevents 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 prune here two gauges instead of one, and because the higher distances between the gauges certainly increases the statistical independence of the malfunctioning events at different gauges.

Conclusions

Designing a neural network through pruning instead of trial and error allows to lower the number of parameters of 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 they required to poll just a limited part (60-70%) of the available rain gauges on the basins. Though relying on less information, their forecast accuracy is 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 re-training on the merge of training and validation data, no longer requiring 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 shows that the removal of a single gauge leads to a forecast availability improvement which ranges 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 to issue a forecast, with a reduced level of accuracy, until at least some gauge is reachable on the basin. Operating the pruned predictor instead of the 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 long enough time series are available, and that the exercise costs of the forecast system grows up with the number of the linked gauges (for example, because of the fee required to access the data of the measuring network, or of the transmission lines installation), the Authority in charge of the forecast system can link just the stations retained in predictor designed by pruning, with clear reduction of 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 modelling.

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

References

- [1] H.R. Maier and C.G. Dandy, “Neural networks for the prediction and forecasting of water resources variables: a review of modelling issues and applications,” *Environmental Mod. & Soft.*, , no. 15, pp. 101–124, 2000.
- [2] V.K. Hsu, S. Gupta, and S. Sorooshian, “Artificial neural network modeling of the rainfall runoff process,” *Water Resour.Res.*, vol. 31, no. 10, pp. 2517–2530, 1995.
- [3] A.Y. Shamseldin, K.M. O’Connor, and K.M. Liang, “Methods for combining the outputs of different rainfall-runoff models,” *J. Hydrol.*, vol. 245, pp. 196–217, 2001.
- [4] R. Reed, “Pruning algorithms-a survey,” *IEEE Transactions on Neural Networks*, vol. 4, no. 5, pp. 740–747, 1993.
- [5] H. M. Henrique, E. L. Lima, and D. E. Seborg, “Model structure determination in neural network models,” *Chemical Engineering Science*, vol. 55, no. 22, pp. 5457–5469, 2000.
- [6] R. García-Gimeno, C. Hervás-Martínez, and M. I. de Silóniz, “Improving artificial neural networks with a pruning methodology and genetic algorithms for their application in microbial growth prediction in food,” *International Journal of Food Microbiology*, vol. 72, no. 1-2, pp. 19–30, 2002.
- [7] R.J. Poppi and D.L. Massart, “The optimal brain surgeon for pruning neural network architecture applied to multivariate calibration,” *Analytica Chimica Acta*, vol. 375, no. 1-2, pp. 187–195, 1998.
- [8] P.T. Quinlan, “Structural change and development in real and artificial neural networks,” *Neural Networks*, , no. 11, pp. 577–599, 1998.
- [9] G. Edelman, *Neural Darwinism: the theory of neuronal group selection*, Basic Books, 1987.
- [10] J.-P. Changeux, P. Courrége, and A. Danchin, “A theory of the epigenesis of neuronal networks by slective stabilisation of synapses,” in *Proceedings of the National Acaddemy of Science USA*, 1973, pp. 2974–2978.
- [11] S.R. Quartz and T.J. Sejnowski, “The neural basis of cognitive development: a constructivist manifesto,” *J. of Behavioral and Brain Sciences*, vol. 45, pp. 35–41, 1997.

- [12] J. Moody and P.J. Antsaklis, “The dependence identification neural networks construction algorithm,” *IEEE Trans. Neural Networks*, vol. 7, pp. 3–15, 1996.
- [13] S.Y. Liong, W.H. Lim, and G.N. Paudyal, “River stage forecasting in Bangladesh: neural network approach,” *J. Comput. in Civil Eng.*, vol. 14, no. 1, pp. 1–8, 2000.
- [14] G. Kim and A.P Barros, “Quantitative flood forecasting using multisensor data and neural networks,” *J. Hydrol.*, vol. 246, pp. 45–62, 2001.
- [15] M. Campolo, P. Andreussi, and A. Soldati, “River flood forecasting with a neural network model,” *Water Resour. Res.*, vol. 35, no. 4, pp. 1191–1197, 1999.
- [16] Inc. MathWorks, *System Identification Toolbox User Guide*, 2000.
- [17] C.M. Bishop, *Neural networks for pattern recognition*, Oxford University Press, 1995.
- [18] S Haykin, *Neural networks: A comprehensive Foundation*, Macmillan Coll., 1995.
- [19] Y. Le Cun, I. Kanter, and S. Solla, “Eigenvalues of covariance matrices: application to neural network learning,” *Phys. Rev. Letters*, vol. 14, no. 1, pp. 2396–2399, 1991.
- [20] F. Girosi, M. Jones, and T. Poggio, “Regularization theory and neural networks architectures,” *Neural Comput.*, vol. 7, pp. 219–269, 1995.
- [21] J. Hertz, A. Krogh, and R.G. Palmer, *Introduction to the Theory of Neural Computation*, Addison Wesley, 1991.
- [22] Y. Le Cun, J.S. Denker, and S.A. Solla, “Optimal brain damage,” in *Advances in neural information processing systems*, D.S. Touretzky, Ed. 1990, vol. 2, pp. 598–605, Morgan Kaufmann.
- [23] B. Hassibi and D.G. Stork, “Second order derivatives for network pruning: Optimal brain surgeon,” in *Proceedings of Advances in Neural Information Processing System*, C.L. Giles S.J. Hanson, J.D. Cowan, Ed., 1993, pp. 164–171.
- [24] M. Norgaard, O. Ravn, N.K. Poulsen, and L.K. Hansen, *Neural networks for modelling and control of dynamic systems*, Springer-Verlag, London, 2000.

- [25] M. Norgaard, “Neural network based system identification toolbox,” Tech. Rep. 00-E-891, Department of Automation, Technical University of Denmark, 2000.
- [26] F.J Chang, J. Liang, and Y. Chen, “Flood forecasting using radial basis function neural networks,” *IEEE Trans. Syst. Man. Cybern. (C)*, vol. 31, no. 4, pp. 530–535, 2001.
- [27] Zampaglione, “Progetto di massima per il riequilibrio idraulico ambientale del fiume Olona: Relazione Idrologica,” 1995.
- [28] G. Corani and G. Guariso, “Stochastic models for flood forecasting on rivers Brembo and Olona,” Tech. Rep. 2001.50, Department of Electronics and Information, Polytechnic of Milan, 2001, (italian).

Tables

<i>gauges couple</i>	<i>cross-correlation</i>
(Arcisate - Varese)	.75
(Arcisate - Vedano)	.65
(Varese - Vedano)	.69

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

	<i>Connected network</i> (3 rain gauges)		<i>Pruned network</i> (2 rain gauges)	
	Training	Testing	Training- Validation	Testing
Time series averaged indicators				
<i>RMSE</i>	.012	.046	.011	.035
<i>R</i> ²	.92	.84	.89	.85
ρ	.96	.92	.94	.93
High flows analysis				
<i>hf</i>	.12	.24	.19	.19

Table 2: Results for the 3 hours ahead prediction on the Olona basin.

<i>gauges couple</i>	<i>cross - correlation</i>	<i>gauges couple</i>	<i>cross - correlation</i>
(Ampezzo-Moggio)	.43	(Moggio-Pesariis)	.43
(Ampezzo-Paularo)	.55	(Moggio-Resia)	.48
(Ampezzo-Pesariis)	.69	(Paularo-Pesariis)	.48
(Ampezzo-Resia)	.42	(Paularo-Resia)	.49
(Moggio-Paularo)	.46	(Pesariis-Resia)	.41

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

	<i>Connected network</i> (Campolo et al., 1999) 5 rain gauges		<i>Pruned network</i> (retrained) 3 rain gauges	
Time series averaged indicators				
	Training	Testing	Training-Validation	Testing
<i>RMSE</i>	-	-	.0087	.0130
<i>R</i> ²	.89	.85	.87	.88
ρ	-	-	.93	.94
High flows analysis				
<i>hf</i>	-	-	.26	.18

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

Figure Captions

Figure 1. Structure of a fully connected neural network.

Figure 2. Values of objective J (eq. 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.

Figure 3. The Olona catchments

Figure 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, r_3 to the Arcisate, Varese, Vedano rainfall gauges. Each variable in a group refers to a specific relative time instant (t, t-1 etc.).

Figure 5. 3-hours-ahead simulations. The reversed y axis (right side) in figures a,b is an estimate of the average rainfall on the basin.

Figure 6. The Tagliamento catchment

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

Figure 8. 5-hours-ahead simulations. The reversed y axis (right side) provided in figures a,b is an estimate of the average rainfall on the basin.

Figures

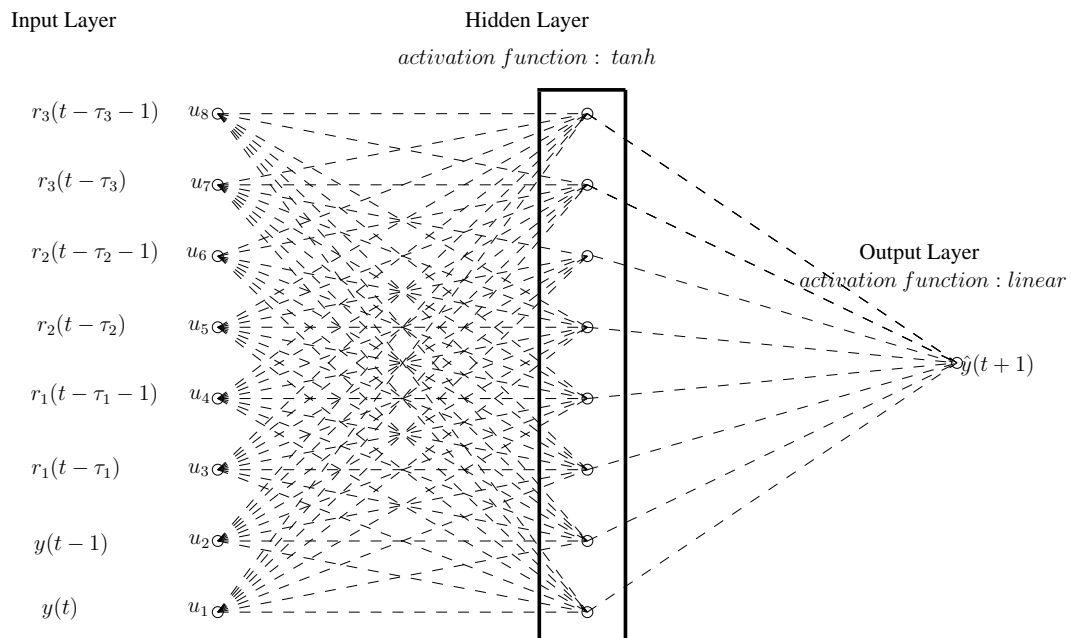


Figure 1: Structure of a fully connected neural network.

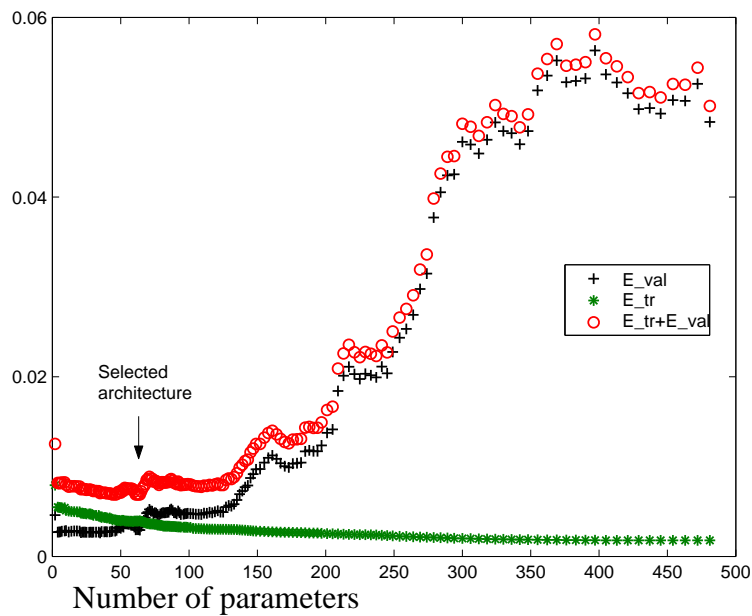


Figure 2: Values of objective J (eq. 9) as a function of the number of parameters during a sample pruning sessions. E_{val} is the root mean squared error on the validation set.

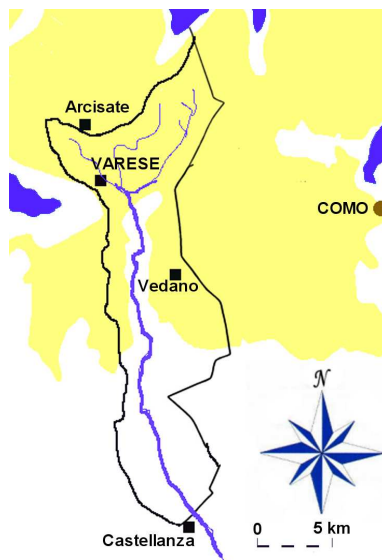


Figure 3: The Olona catchments

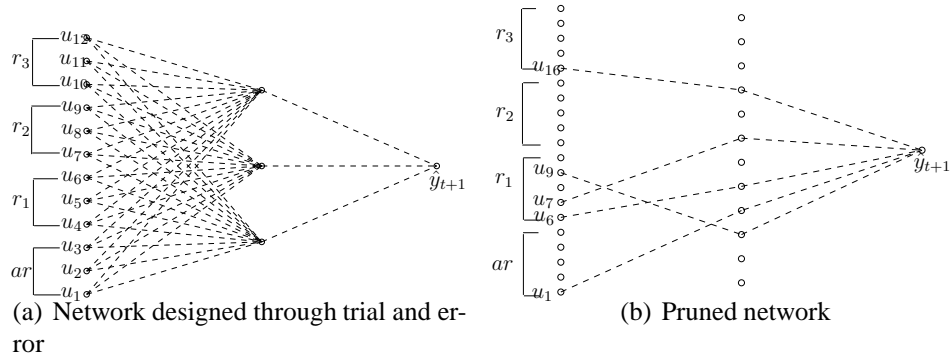
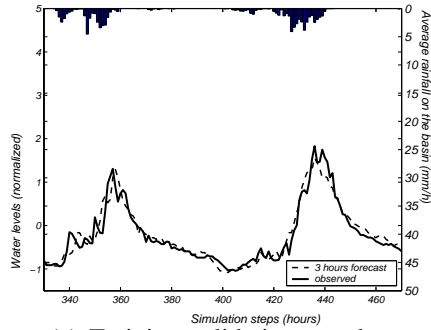
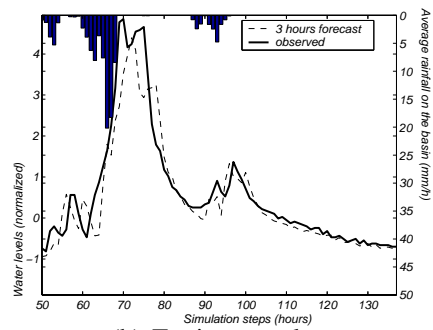


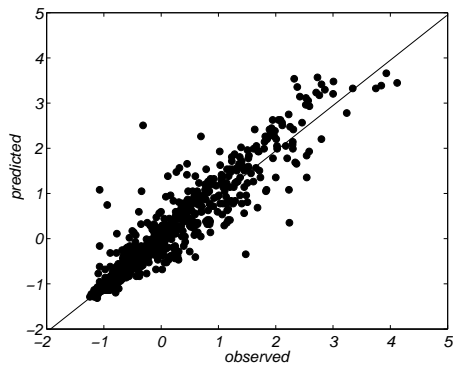
Figure 4: Neural network architectures for the Olona river. Input variables in group ar are of autoregressive type, while those in groups r_1, r_2, r_3 to the Arcisate, Varese, Vedano rainfall gauges. Each variable in a group refers to a specific relative time instant ($t, t-1$ etc.).



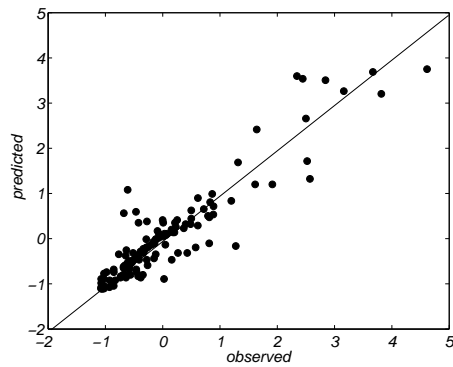
(a) Training-validation sample



(b) Testing sample



(c) Training-validation scatter plot



(d) Testing scatter plot

Figure 5: 3-hours-ahead simulations. The reversed y axis (right side) in figures a,b is an estimate of the average rainfall on the basin.

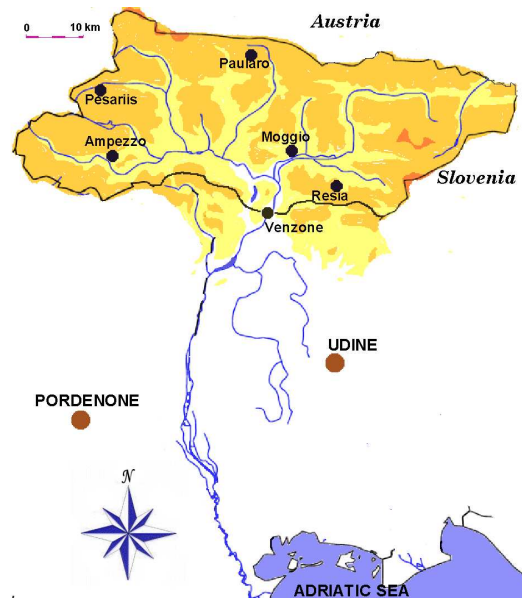


Figure 6: The Tagliamento catchment

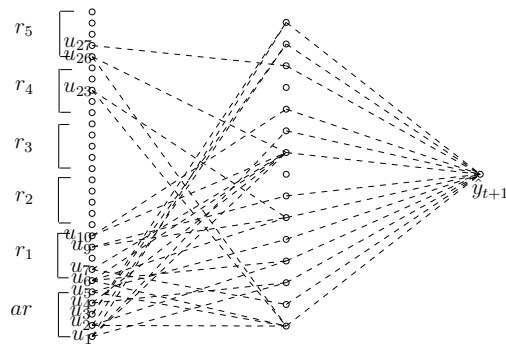
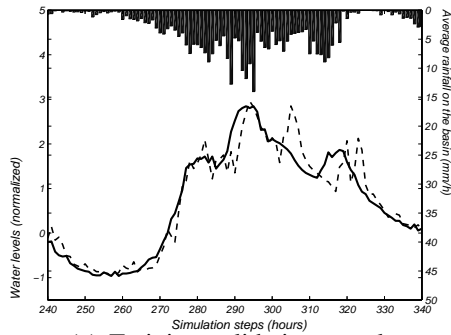
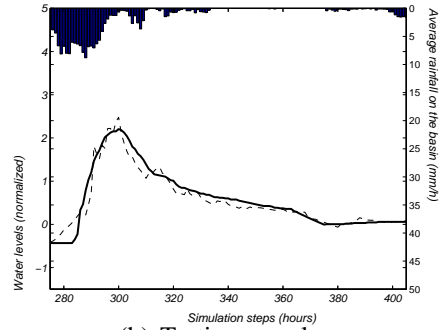


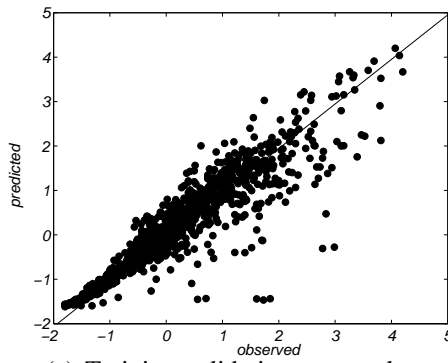
Figure 7: The architecture of the pruned neural predictor. ar refer to autoregressive water level measurements (Venzone), r_1, r_2, r_3, r_4, r_5 to Ampezzo, Moggio, Paularo, Pesariis, Resia rainfall measurements.



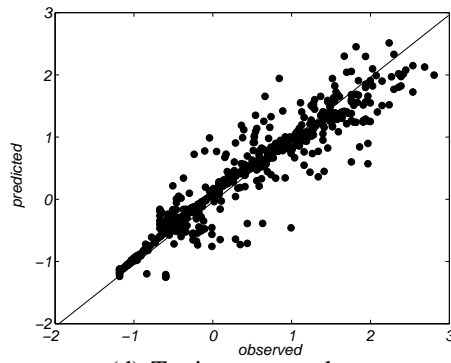
(a) Training-validation sample



(b) Testing sample



(c) Training-validation scatter plot



(d) Testing scatter plot

Figure 8: 5-hours-ahead simulations. The reversed y axis (right side) provided in figures a,b is an estimate of the average rainfall on the basin.