Generalisation for neural networks through data sampling and training procedures, with applications to streamflow predictions

F. Anctil, N. Lauzon

Hydrology and Earth System Sciences, 8(5), *940–958* (2004) © EGU

Hydrology & Earth System Sciences

Generalisation for neural networks through data sampling and training procedures, with applications to streamflow predictions

François Anctil[1] and Nicolas Lauzon[2]

[1]Department of Civil Engineering, Université Laval, Pavillon PouliotQuebec City, QC G1K 7P4, Canada
[2]Golder Associates Ltd., 1000, 940-6th Avenue SW, Calgary, Alberta T2P 3T1, Canada

Email for corresponding author: nicolas.lauzon@golder.com

Abstract

Since the 1990s, neural networks have been applied to many studies in hydrology and water resources. Extensive reviews on neural network modelling have identified the major issues affecting modelling performance; one of the most important is generalisation, which refers to building models that can infer the behaviour of the system under study for conditions represented not only in the data employed for training and testing but also for those conditions not present in the data sets but inherent to the system. This work compares five generalisation approaches: stop training, Bayesian regularisation, stacking, bagging and boosting. All have been tested with neural networks in various scientific domains; stop training and stacking having been applied regularly in hydrology and water resources for some years, while Bayesian regularisation, bagging and boosting have been less common. The comparison is applied to streamflow modelling with multi-layer perceptron neural networks and the Levenberg-Marquardt algorithm as training procedure. Six catchments, with diverse hydrological behaviours, are employed as test cases to draw general conclusions and guidelines on the use of the generalisation techniques for practitioners in hydrology and water resources. All generalisation approaches provide improved performance compared with standard neural networks without generalisation. Stacking, bagging and boosting, which affect the construction of training sets, provide the best improvement from standard models, compared with stop-training and Bayesian regularisation, which regulate the training algorithm. Stacking performs better than the others although the benefit in performance is slight compared with bagging and boosting; furthermore, it is not consistent from one catchment to another. For a good combination of improvement and stability in modelling performance, the joint use of stop training or Bayesian regularisation with either bagging or boosting is recommended.

Keywords: neural networks, generalisation, stacking, bagging, boosting, stop-training, Bayesian regularisation, streamflow modelling

Introduction and literature review

Neural networks have become accepted tools for fast efficient modelling for a large range of scientific domains. In hydrology and water resources, the issues that have been assessed traditionally are the proper selection and pre-processing of the inputs and outputs, and the choice of the architecture of the neural networks (Coulibaly *et al*., 1999; Dawson and Wilby, 2001). The necessity of determining representative training and test data sets is also highlighted by ASCE (2000a) and Maier and Dandy (2000) as critical for the construction of optimal neural network models. All these issues are part of the difficult-to-resolve problem of generalisation (ASCE, 200b), which refers to building models (i.e. any kind of models) that can infer the behaviour of the system under study for conditions represented in the data used for training and testing as well as for conditions absent from the data sets but inherent to the system.

Generalisation is a standard issue for all types of models and aims at reducing modelling errors, which can be decomposed into three parts, bias, variance and noise (Geman *et al.,* 1992; Wolpert and Macready, 1999; Zhou *et al.*, 2002). Bias is a measure of how much a function deviates from the desired function over the data set. Variance indicates the extent to which the model is sensitive to the choice of the data set. Noise is intrinsic to the system under study and is expected under any circumstances. Reducing bias and variance is the goal of generalisation. Here, the primary focus is on the gain in modelling performance

through the use of generalisation techniques; this, implicitly, relates to reducing bias and variance errors. This paper explores some approaches to generalisation so as to promote them for applications of neural networks to hydrology and water resources and, more particularly in this work, to streamflow modelling.

GENERALISATION THROUGH TRAINING PROCEDURES

A first group of approaches (i.e. stop training and Bayesian regularisation) relates to the training procedures, and they have been used to some extent in hydrology and water resources. Stop training is a long standing practice by which a training set is used simultaneously with a validation set. The weights of the networks are optimised with the training set while the progression of the optimisation process is regulated by the performance of the resulting network on the validation set. The process is stopped when an optimum (e.g. minimisation of the sum of squared errors) is reached on the performance of the network with this validation set. Also known as cross-validation (ASCE, 2000a), this concept aims at stopping the optimisation process before over-fitting occurs; then, the weights of the networks become too fixed to accommodate system conditions present in the training data set, likely reducing the validity of the weights for other possible system conditions (Amari *et al.*, 1997; Gençai and Qi, 2001; Vlassides *et al.*, 2001; Drucker, 2002; Chan *et al.*, 2003). Examples of application of early stopping to water resources and hydrology can be found in Lopez-Sabater *et al.* (2002), for hydraulic parameters; Coulibaly *et al.* (2001), for groundwater modelling; Maier and Dandy (1997) and Yabunaka *et al.* (1997), for algae concentration; Xiao and Chandrasekar (1997), Kuligowski and Barros (1998) and Luk *et al.* (2000), for rainfall; and Raman and Sunilkumar (1995), Golob *et al.* (1998) and Coulibaly *et al.* (2000) for streamflow modelling.

Bayesian regularisation is a more recent approach, and consists in reducing the impacts of the network weights. Bayesian regularisation involves a multiple objective optimisation in which both the sum of squared errors and the sum of the squared weights must be minimised (MacKay, 1992; Foresee and Hagan 1997; Anctil *et al.*, 2004). Bayesian regularisation often leads to more parsimonious networks than stop training (Medeiros *et al.*, 2001). This approach is particularly suited to reduce variance errors, because the minimisation constrains the weight to small values, making less likely the possibility of large fluctuations in the response of the network given large inputs. Examples of application of this approach to water resources and hydrology can be found in Anctil *et al.* (2004), for streamflow modelling, or Morse *et al.* (2003), for estimating ice parameters.

GENERALISATION THROUGH DATA SAMPLING

A second group of generalisation approaches (i.e. bagging and boosting) relates to techniques used for constructing training data sets, and they have not really been used extensively in hydrology and water resources. One available application of bagging in water resources is the work of Cannon and Whitfield (2002) on streamflow modelling. Bagging, which is also known as bootstrap aggregating (Breiman, 1996), is a procedure where several training data sets are created from an original training data set by bootstrap (i.e. random pick with replacement). Each of these training sets is employed to train a neural network, or any other types of models for that matter. Hence, a pool of models is created, and a global estimate can be obtained by the aggregation (e.g. the mean) of their estimates for a given input vector. It has been demonstrated that bagging can reduce variance errors (Breiman, 1996, 2001), since aggregation has the effect of smoothing fluctuations from the estimates of all the models. The distribution of the estimates also allows the construction of confidence intervals (Lajbcygier and Connor, 1997; Papadopoulos *et al.*, 2001), hence giving a margin of errors in addition to the global estimate. Many applications of bagging have implicated synthetically generated data or simple real-application cases for basic analyses of the approach (Breiman, 1996, 2000, 2001; Wolpert and Macready, 1999; Zhou *et al.*, 2002). Examples of the use of bagging with more complex real-application cases can be found in Raviv and Intrator (1996), with medical data; Lajbcygier and Connor (1997) and Gençai and Qi (2001), with economic data; Papadopoulos *et al.* (2001), with industrial process data (paper curl); Sohn and Lee (2003), with an application to road traffic accidents; and Agrafiotis *et al.* (2002), with a case in chemistry. Boosting also demands the construction of several models, but usually involves a step-by-step procedure. In the first step, all the training input vectors have equal impact on the optimisation of the weights or parameters of the models. This impact is updated in the subsequent steps so as to focus on the input vectors for which the previous models have been less efficient in terms of performance. The subsequent models are, thus, calibrated based on these disadvantaged input vectors, providing more specific responses that compensate for the weaknesses of the previous models. Quite often, the impact of the input vectors is determined by weights given to them in the creation of a set of vectors for the training set (Pal and Mather, 2003). At other times, the weights for each input vectors are imposed on the

objective function that regulate the training procedure (Schwenk and Bengio, 2000). Global estimates can be obtained from the aggregation of the estimates of models built in this step by step process. Adaboost (Freund and Schapire, 1997) is a very common computational tool that offers several boosting procedures, for it is mentioned in many applications, including simple and theoretical cases (Drucker, 2002; Zhou *et al.*, 2002), as well as some more complex examples such as classification processes (Schwenk and Bengio, 2000), pharmaceutical drug dissolution (Goh *et al.*, 2003) and application to road traffic accidents (Sohn and Lee, 2003). Most of the time, with boosting, the impact (weights) of the input vectors is set automatically, although nothing would forbid a more manual determination. Such is the case of stratified sampling (MacNamee *et al.*, 2002; Chawla *et al.*, 2003), where input vectors are classified according to specific patterns, and then the construction of a sample of vectors for the training set is accomplished by sampling equally from each of the classes of vectors. It implies that less frequent patterns would have a greater representation than if a totally random sampling where performed on the set of input vectors put all together. Stratified sampling could have a significant impact on cases of streamflow modelling, where extreme, but rare, high flows would have a larger representation.

The last approach mentioned in this work is called stacking (or ensemble), and implies the use of a pool of models (Wolpert, 1992). In a first version, several different models are used in parallel and a global estimate is obtained from the aggregation of the estimates of the models (e.g. the mean of the results of the models). In a second version, several models are used in series, each model using outputs from the previous model as inputs. This approach has been applied on many occasions in hydrology and water resources, and not only with neural networks. Refining a streamflow model by using another model to estimate its errors is a common and long-standing practice, as attested by WMO (1992), which describes several deterministic streamflow models for the purpose of a comparison, many of them using autoregressive functions to model their errors so as to improve streamflow predictions. Using neural networks instead of autoregressive functions has also been attempted by Babovic *et al.* (2001) with a hydrodynamic model and by Anctil *et al.* (2003) with several rainfall-runoff models. Similarly, the aggregation of the estimates of several streamflow models is a long-standing practice. The work of Cavadias and Morin (1986) is an early example of the aggregation of several deterministic streamflow models. More recently, several versions of model aggregation involving neural networks have been performed by Abrahart and See (2000, 2002), Hu *et al.* (2001), Kim and Barros (2001), See and Abrahart (2001), and Shamseldin *et al.* (2002). Model aggregation is the version of stacking employed in this work.

GENERALISATION APPLIED TO STREAMFLOW MODELLING WITH NEURAL NETWORKS

Stop training, Bayesian regularisation, bagging and particular forms of boosting (stratified sampling) and stacking (aggregation of models) are the generalisation approaches compared in this paper in the context of streamflow modelling. Multi-layer perceptron neural networks, calibrated with the Levendberg Marquardt algorithm, are employed as streamflow models. The Levendberg Marquardt algorithm is chosen as the optimisation tool because of its proven efficiency, as demonstrated in Tan and Van Cauwenberghe (1999). Systematic comparison and analysis of several generalisation approaches have been accomplished on some occasions (see for example Yang *et al.*, 1998; Bauer and Kohavi, 1999; Cunningham *et al.*, 2000; Zhou *et al.*, 2002; Sohn and Lee, 2003). Shu and Burn (2004) are arguably the first to offer a systematic comparison of generalisation techniques for neural networks in hydrology, with an application to flood frequency analysis using stacking, bagging and boosting. In this paper, the intent is to explore a wider range of generalisation techniques with a very common application with neural networks: streamflow modelling. The first two approaches control the training process, and generalisation is achieved by avoiding extreme situations, that is, over-fitting in the case of stop training and exceedingly large weights in the case of Bayesian regularisation. In the last three approaches, generalisation is established based on the sampling process employed to build training sets. A diversity of training sets ensures that models are trained to respond to a larger number of behaviours, in the expectation that the aggregation of the models covers all the possible behaviours of the system under study. Bagging involves that all input vectors have an equal chance at all time to be selected for the training sets. Boosting, in this work, is a constrained form of sampling, where classes of input vectors have an equal chance to be represented in the training sets. The determination of the classes is performed with self organised maps (i.e. Kohonen neural networks), as explained later in the paper. Stacking, here, employs the same original training set for all models. Generalisation is achieved only through the aggregation of the models, and consequently serves as reference when compared with bagging and boosting. These approaches are tested with the data from six different catchments, for one-step ahead predictions of daily

streamflow. The context of application is presented in the next section and includes details on the Kohonen networks for the classification of the input vectors and a description of the protocol of experiment for comparing the approaches. The analysis of the results from the application of stop training, Bayesian regularisation, bagging, boosting and stacking is detailed in the following section. This analysis leads to the formulation of guidelines that may orient practitioners in the use of these approaches.

Context of application

RIVERS UNDER STUDY

The selection of the rivers is driven by the necessity to encompass a large array of hydrological behaviours so as to ensure that the results from the approaches tested here are as general as possible. The rivers listed in Table 1 come from different hydro-climatic regions, which respectively generate different hydrological behaviours in terms of streamflow production. Figure 1 illustrates the daily mean, max and min streamflow for all six rivers. At the low extreme is the catchment of the Salt Fork River in the semi-arid region of the U.S. Midwest. Streamflow on this river is generally low, even intermittent, presenting a very weak seasonal cycle. Heavy precipitations do not imply the occurrence of high streamflow peaks, on account of likely long periods of low soil moisture in the catchment. The annual period of high flow is intermittent on this catchment, with some years showing no significant high flows. At the high extreme is the catchment of the San Juan River, which is very humid, has a very obvious seasonal cycle, and possesses the highest daily streamflow mean and standard deviation of all the catchments under study (see Table 1). Located on the Canadian Pacific Coast, this catchment is fed by sustained heavy precipitations, particularly between November and April. Soil moisture is likely to be high for most of the year, ensuring a high streamflow production from this catchment. In terms of hydrological conditions, the other four catchments (i.e. Kavi, Leaf, Serein and Volpajola) lie between these two extremes with respect to streamflow production. A seasonal cycle is quite apparent for Kavi and Serein, while this cycle is less pronounced for Leaf and Volpajola. The high flow period for Kavi occurs in summer and it is usually very dry in winter. Winter is the period of high flow for Leaf, Serein and Volpajola, while summer is relatively dry. Very high streamflows compared with the average follow heavy precipitations on Leaf, while moderately high streamflows are common in Serein and Volpajola in winter. On some occasions, Volpajola generates relatively high streamflow in summer. The common feature of all six catchments is the almost entire absence of snow. Actually, snow falls briefly almost every year in San Juan and occasionally in Serein but is assumed to have a negligible effect on the rainfall-runoff relationship. For all catchments, rainfall is virtually the sole generator of streamflow. Daily streamflow and rainfall observations are available for all of these six catchments over a period of from 18 to 43 years (see Table 1).

DETERMINATION OF THE TRAINING AND TEST DATA SETS

Although it is secondary to the main objective of this work, that is, generalisation, careful attention must be given to the construction of the training and test data sets. For most applications in general and for those in hydrology and water resources in particular, it must be ensured that all of these sets contain an adequate proportion of all the system behaviours available in the database (Coulibaly *et al.,* 1999; ASCE, 2000b; Maier and Dandy, 2000). Here a classification algorithm, the Kohonen neural network, is employed for an objective identification of the patterns present in each database. The Kohonen neural network is a clustering

Table 1. Basic characteristics of the catchments under study.

Catchment	*Area* (km^2)	*Daily streamflow (mm)* *Mean*	*St. dev.*	*Location*	*Database length (year)*
Kavi	975	0.39	0.85	Ivory Coast	32
Leaf	1949	1.37	2.90	United States	40
Salt Fork	2217	0.10	0.45	United States, Midwest	29
San Juan	580	7.10	11.23	Canada, Pacific coast	34
Serein	1120	0.61	0.86	France	43
Volpajola	930	2.40	2.42	France, Corsica	18

Note: St. dev. stands for standard deviation.

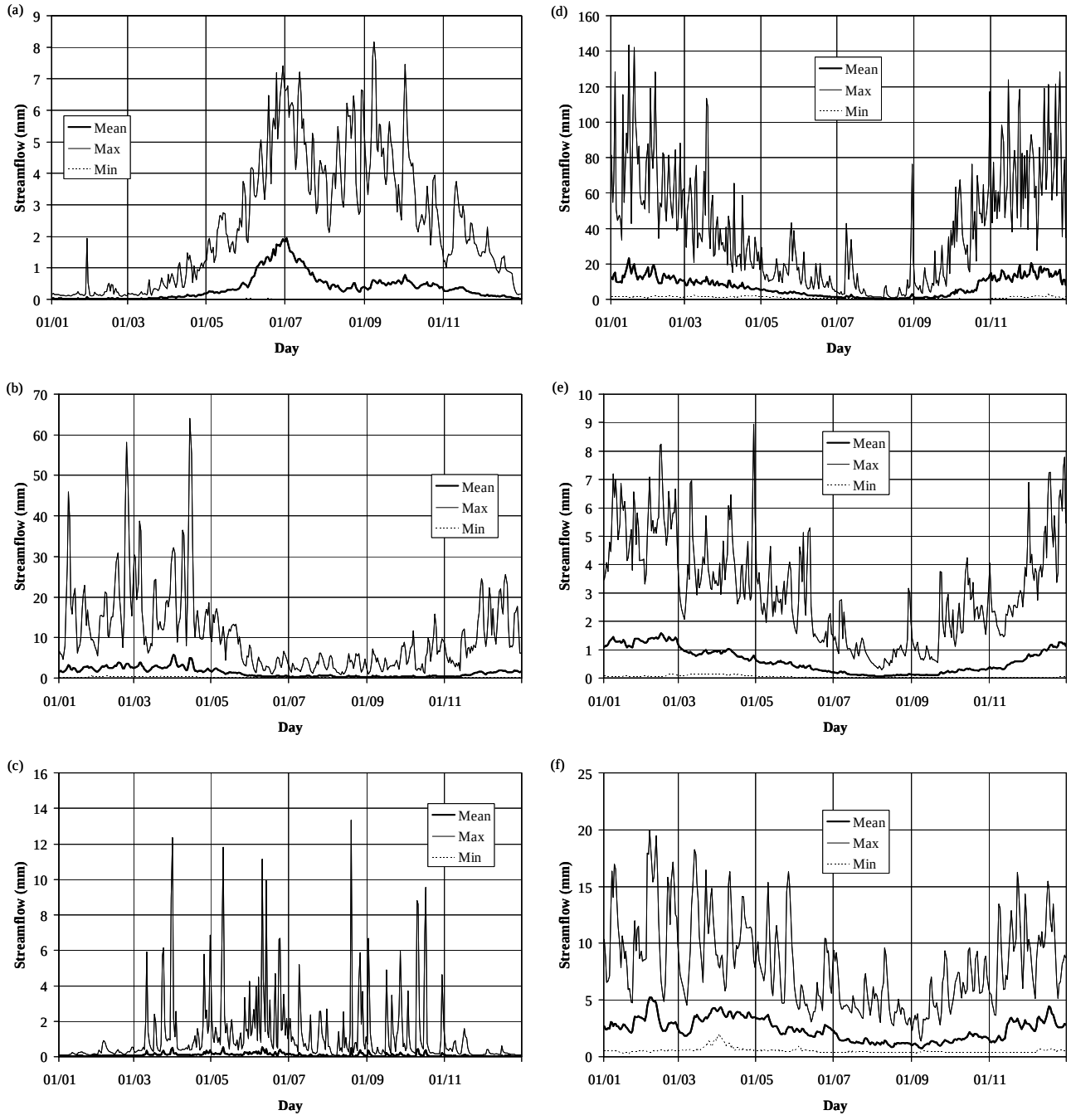

Fig. 1. *Daily mean, maximum and minimum streamflows for the (a) Kavi, (b) Leaf, (c) Salt Fork, (d) San Juan, (e) Serein and (f) Volpajola rivers.*

technique that is used increasingly in hydrology and water resources, such as for the classification of catchment conditions (Liong *et al.*, 2000); the determination of hydrological homogeneous regions (Hall and Minns, 1999); the identification of river pollutant sources (Gots *et al.*, 1998); and the study of algae bloom (Bowden *et al.*, 2002). The network is made of an input layer of neurons that receive the data and of an output layer often structured in a planar surface. The weight vector of each output neuron is of the same scale as the inputs, and consequently can be considered as the mass centre of a class. Each output neuron is, thus, the equivalent of a class, and it is said to be activated when its weight vector is the closest in distance to the input vector given to the network. The elements of all the weight vectors must be calibrated so as to cover the whole data domain. The calibration process ensures that all the patterns present

in the data are defined in a meaningful coordinate system, and this is why the Kohonen network is often called a self-organised map. The Kohonen network reduces the dimension of a problem, from an *n*-dimension input vector to 2-dimension solution, so as to produce a clearer view of the data patterns (Kohonen, 1990 and 1997).

The classification strategy used here is similar to that of Abrahart and See (2000), where the classification is performed on the input vectors. A few output layer configurations have been tested (3×3, 3×4 and 4×4), and the smallest one has been kept because: (1) it ensures that a large number of input vectors is associated with each class of behaviour (i.e. output neuron); and (2) it provides information on the behaviour of the system that is as meaningful as that provided by the larger configuration. Once the classification is complete, the input vectors in each class are divided randomly into three equal subsets. The first subset represents the original training set prior to the use of bagging and boosting. The second one is another training set used for validation when stop training is employed. The first and second subsets are put together when stop training is not employed. The third subset is the test set on which the performance of the streamflow models is evaluated after training. A global test subset is obtained from the combination of the test subset of each class, and is performed with both types of training sets (first and second) as well. This construction process ensures that each class is equally represented in all data sets. This increases the likelihood that an equal proportion of all the system behaviours available in the database is present in all the data sets.

The classification made with the Kohonen neural network is also useful when boosting is applied. Here, boosting implies stratified sampling of the original training set with respect to the classes of input vectors. The sampling requires that a training set be composed of an equal number of input vectors from each class. Hence, classes with a smaller number of input vectors would have a larger representation in the resulting training set than as would be the case with a purely random selection of input vectors, regardless of the classes. Bagging employs this purely random selection of input vectors for the construction of training sets. For this work, the effect of boosting is that extreme flood events are more frequently available for the training of neural networks than as is the case with bagging and stacking.

PROTOCOL OF EXPERIMENT

Multi-layer perceptron neural networks, calibrated with the Levendberg Marquardt algorithm, are employed as streamflow models to perform one-step ahead predictions (i.e. daily streamflow at time t+1). For all six catchments, the input vectors are a daily streamflow at time t and daily rainfall at times t, t-1 and t-2. Such inputs are relatively standard for neural network streamflow models and for the size of the catchments. While one would expect a distinct input vector structure for each catchment, an identical structure has been applied to all so as to create an efficient automated process to evaluate the models. In the input vectors, there are enough precipitation and streamflow inputs to accommodate the catchment with the slowest response time, and it is assumed that the calibration process is efficient enough to generate parsimonious models in spite of the potentially large number of inputs. One hidden layer is included in the network architecture, and tests are made to determine the effect of the number of neurons in that layer. The number of neurons in the hidden layer has been allowed to vary between 2 and 15, making a total of 14 possible hidden layer configurations. Once the input vectors are classified and the training and test sets are built, neural network streamflow models are developed for all the catchments, using bagging (i.e. random sampling of the original training set), boosting (i.e. stratified sampling of the original training set), and stacking (i.e. training of several neural networks with the original training sets). Fifty structurally identical, neural networks are created each time one of these approaches is used. These networks differ only by virtue of their distinct training set with bagging and boosting, and by the initial seeds employed in their training in all cases. Bagging, boosting and stacking are also used in conjunction with stop training or Bayesian regularisation. Thus, a total of 9 modelling configurations (e.g. bagging with stop training, boosting alone, and so on) are tested for each of the six catchments so that 6300 neural network models are trained for this work (i.e. 9 modelling configurations, by 14 hidden layer configurations, by 50 replications).

With the test data sets, the prediction performance of the streamflow models is evaluated with the persistence index, which is expressed as follows (Kitanidis and Bras, 1980):

$$PERS = 1 - \frac{SSE}{SSE_{naïve}} \tag{1}$$

where:

$$SSE = \sum_{t} \left(Q_{t,obs} - Q_{t,est}\right)^2 \tag{2}$$

and:

$$SSE_{naïve} = \sum_{t} \left(Q_{t,obs} - Q_{t-1,obs}\right)^2 \tag{3}$$

in which the *SSE* terms are the sum of square errors, and

$Q_{t,obs}$ and $Q_{t,est}$ are the observed and estimated streamflows, respectively. Persistence consists of a comparison between the model under study, and the naïve model where the estimate of the observed streamflow at time t ($Q_{t,obs}$) is simply the streamflow at time t-1 ($Q_{t-1,obs}$). A value of *PERS* smaller or equal to 0 indicates that the model under study performs worse or no better than the easy to implement naïve model. A *PERS* value of 1 is obtained when the model under study provides exact estimates of observed streamflows.

Numerous models are tested in this application and the results may spawn a large number of analyses. Only one performance measure (i.e. *PERS*) is employed here so as to provide a concise account of the results obtained from the models. However the persistence index on any given model provides an indication of bias error, while the values of persistence index obtained from a pool of models may provide indication of variance error.

Results

CLASSIFICATION OF INPUT VECTORS

Classification of input vectors is performed with the Kohonen neural network for all rivers so as to allow an objective division of the database into three sets (i.e. two as training sets and one as the test set). Nine classes are determined for each river; this small number of classes ensures that a large number of input vectors is associated with each class of behaviour. The division is accomplished so that each class of input vectors is represented equally in all data sets. Table 2 shows an example of the results obtained from the classification of the Leaf River database. For each of the classes, from 1 to 9, the averages of the elements of the input vectors (i.e. streamflow at time t and rainfall at times t, t-1 and t-2) as well as the average of the corresponding output (i.e. streamflow at time t+1) are given in Table 2. The averages for the whole database for Leaf are shown for comparison (i.e. Total in Table 2). The classification highlights the different patterns present in the database, where each class exhibits distinct input vectors. The common feature of the classification for all rivers is the presence of a large class (e.g. class 7 in Table 2), which contains the numerous cases of low to medium flows with low to moderate rainfall. These cases can be considered as relatively uniform, with low variability. The other classes are dedicated to the discrimination of the various patterns of streamflows and rainfalls present in the database. For all rivers, there is always a class of high flows with relatively high rainfall (e.g. class 3 in Table 2).

The effect of the classification is determinant in streamflow modelling configurations that involve the use of boosting which, in this work, comes down to a stratified sampling for the construction of data sets for training models. Stratified sampling implies that each class is represented by an equal number of input vectors, which means that classes with a small number of input vectors in the original training set are over represented in the data sets constructed for training. Extreme events such as high floods occur more often and are more likely to influence to their advantage the determination of the weights of the models, even though the increase in their presence is made through replication of the same events (i.e. sampling with replacement).

Table 2. Characteristics of the classes of input vectors for Leaf.

Class	*Mean (in mm) for*					*Number of input vectors for*			
	Q_{t+1}	*Input vector*				*T1*	*T2*	*V*	*Total*
		Q_t	P_t	P_{t-1}	P_{t-2}				
1	2.68	1.32	30.15	4.69	1.79	429	396	361	1186
2	2.53	2.39	12.50	2.97	14.66	75	82	82	239
3	8.15	8.64	3.22	8.21	39.23	200	220	208	628
4	0.62	0.57	7.05	1.03	0.67	404	438	377	1219
5	1.47	1.97	0.69	0.97	6.91	487	492	571	1550
6	2.87	3.87	0.79	4.95	15.66	169	182	166	517
7	0.41	0.48	0.19	0.16	0.35	2428	2293	2421	7142
8	0.61	0.64	1.02	6.95	1.17	360	350	341	1051
9	2.82	2.44	3.70	29.84	3.37	343	340	392	1075
Total	1.37	1.37	3.92	3.92	3.92	4895	4793	4919	14607

Note: T1 is the first training set, T2 is the second training set, and V is the test set.

Once the classification is established, the database is divided randomly to ensure that about one third of the input vectors of any class is allocated to each of the three data sets, that is, two for training and one for testing, respectively T1, T2 and V in Table 2. As a reminder, T1 and T2 are merged when stop training is not employed.

STAND-ALONE MODELS

This study is concerned primarily by improvement in the performance of streamflow models when a generalisation approach is employed. This improvement is compared with the performance obtained in the traditional situation where only one model is developed (i.e. without replications). Here, these single models are called stand-alone models and are trained using the Levendberg-Marquardt algorithm only (i.e. no stop training and no Bayesian regularisation). Note that these stand-alone models are also used in the stacking modelling configuration. Figure 2 shows the performance of these reference models. The results are classified in terms of the number of hidden neurons, and persistence is established with the test data set. The number of hidden

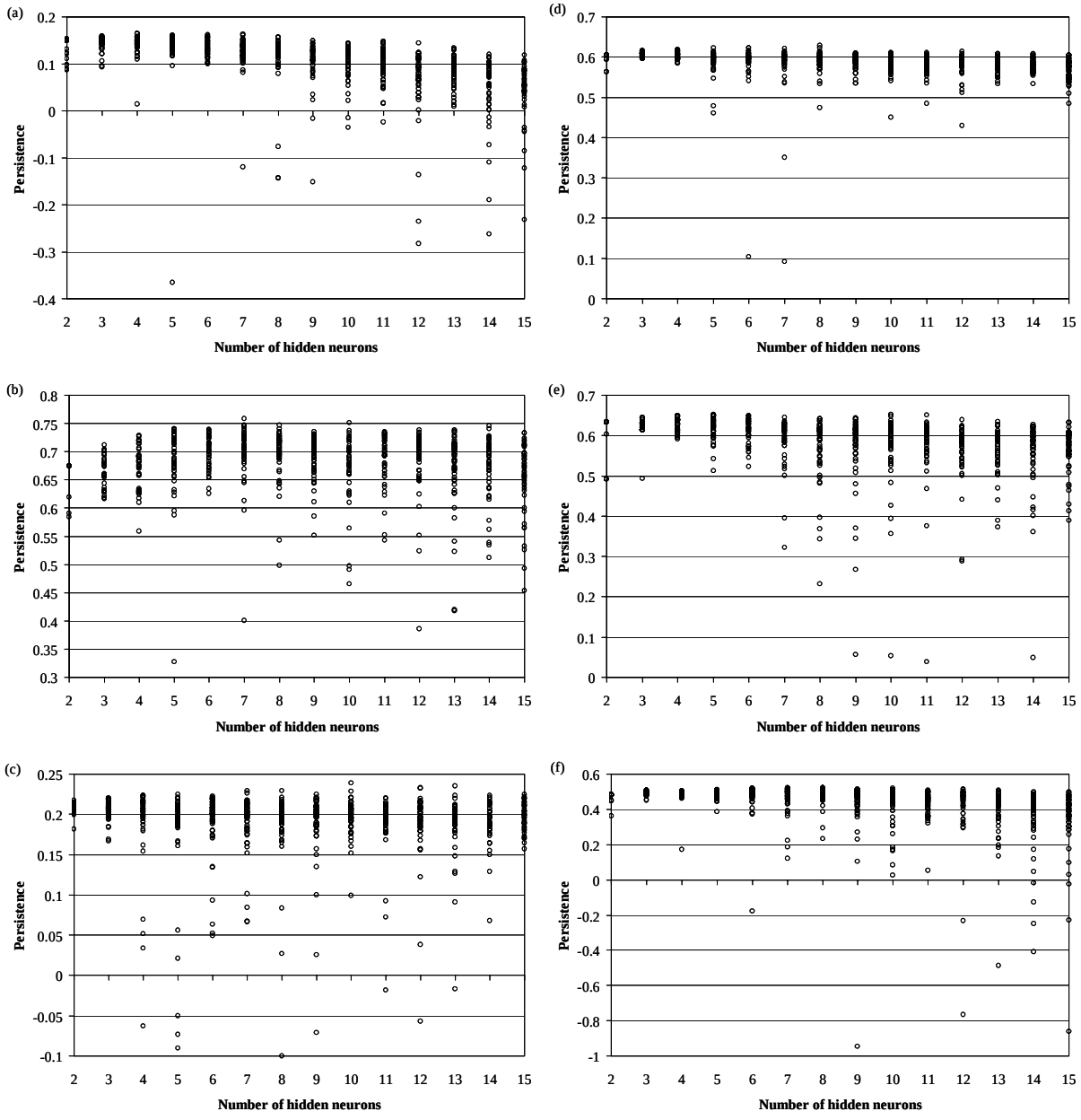

Fig. 2. *Variation of modelling performance in terms of persistence for stand-alone models for the (a) Kavi, (b) Leaf, (c) Salt Fork, (d) San Juan, (e) Serein and (f) Volpajola rivers.*

neurons is assumed to have little effect on modelling performance overall and this is confirmed by the results in Fig. 2, as the median performance for any given river is relatively constant with respect to the number of hidden neurons.

However, the performance varies considerably from one replicate to the other, exemplifying the risk of stand-alone models, as mentioned by Iyer and Rhinehart (1999). If exception is made of models with isolated extremely low performance, this variability tends to increase with the number of hidden neurons for most rivers in Fig. 2. The complexity of the model, as expressed by the number of hidden neurons, affects the search of a global optimum by the training algorithm. The greater the number of hidden neurons, the more difficult it is to find the global optimum. Ideally, all models for a river should exhibit the same performance, the limitation on the global optimum being regulated only by the capacity of the input vectors to explain the outputs. It is obvious that the input vectors do not explain the outputs for Kavi or Salt Fork (Fig. 2a and c) as clearly as do those for Leaf, Serein, San Juan and even Volpajola (Fig. 2b, e, d and f, respectively).

For the rest of this study, because stand-alone model performance is highly sensitive to the initialisation of the weights of the networks during optimisation, the median performance of the stand-alone models is compared with the performance of the streamflow models based on a generalisation approach.

EFFECT OF GENERALISATION ON MODELLING PERFORMANCE

Stacking, bagging and boosting have a beneficial effect on the performance of streamflow models. All three cases involve the aggregation of the estimates of several streamflow models (i.e. 50 in this work) into one global estimate. The mean and the median of the model streamflow estimates are considered here as global estimates. Also, the estimates of the stand-alone model that produces the best individual performance in the modelling configuration for stacking are also considered global estimates. Figure 3 shows the performance of these three estimates (Mean, Median, Max) for all six rivers, when stacking is employed and with respect to the number of hidden neurons. In Fig. 3, the median performance of the stand-alone models is also illustrated for comparison. Clearly, streamflow estimates from the use of stacking (Mean, Median) lead to better performances than estimates from most stand-alone models, particularly as the number of hidden neurons increases. This conclusion also applies to bagging and boosting, as confirmed by Figs. 4 and 5, respectively. The performance of the best individual models (Max) is not presented in Figs. 4 and 5 because the results are not consistently as good as with the mean and median estimates. A generalisation approach helps to reduce the uncertainties related to the difficulty of the training algorithm to find the global optimum as the complexity (i.e. the number of hidden neurons) of the streamflow models increases.

Stacking, bagging and boosting perform equally well, as shown in Figs. 3, 4 and 5. Either method could be used with favourable results, although stacking may be a preferred choice where the models perform poorly, such as Salt Fork River (item c in Figs. 3 to 5). Kavi and Salt Fork are cases for which neural networks are less convincing in terms of performance. Salt Fork is an intermittent system, dependent largely on detailed soil moisture conditions to explain streamflows. These soil moisture conditions are not available in the data given to the models. There is very little dependence between the input vectors and the outputs, and no sampling method can possibly correct that shortfall.

On a global scale, Table 3 summarises the performance results for all modelling configuration, thus exhibiting the behaviour of all generalisation approaches considered in this work. Each element of Table 3 is the average persistence with respect to the number of hidden neurons. For example, the value of the third column, third row for the Kavi River (0.147) is the average of the performance values shown in Fig. 3a, with the median employed as estimator (square symbols). Table 3 demonstrates that stacking, bagging and boosting provide consistent improvements over stand-alone models, and this confirms the results in Figs. 3 to 5. Stand-alone models benefiting from stop-training or Bayesian regularisation show some improvement from stand-alone models where neither of these two generalisation approaches is used (i.e. column 6 or 9 compared with column 3 in Table 3). However, the improvement may sometimes be only marginal. The best estimator with stacking is the stand-alone model having the maximum performance (Max), while the mean is a better estimator for bagging and boosting than the median. The combination of stop-training or Bayesian regularisation with either one of stacking, bagging or boosting always improves performance from the median of stand-alone models (i.e. Alone versus the Mean or Median in Table 3). Of course, there might be instances of stand-alone models that outperform any form of generalisation (see Max versus the Mean or Median in Table 3). However, to find such instances, a very large number of stand-alone models would have to be produced, and this adds an element of uncertainty in the strategy of using stand-alone models versus a generalisation technique.

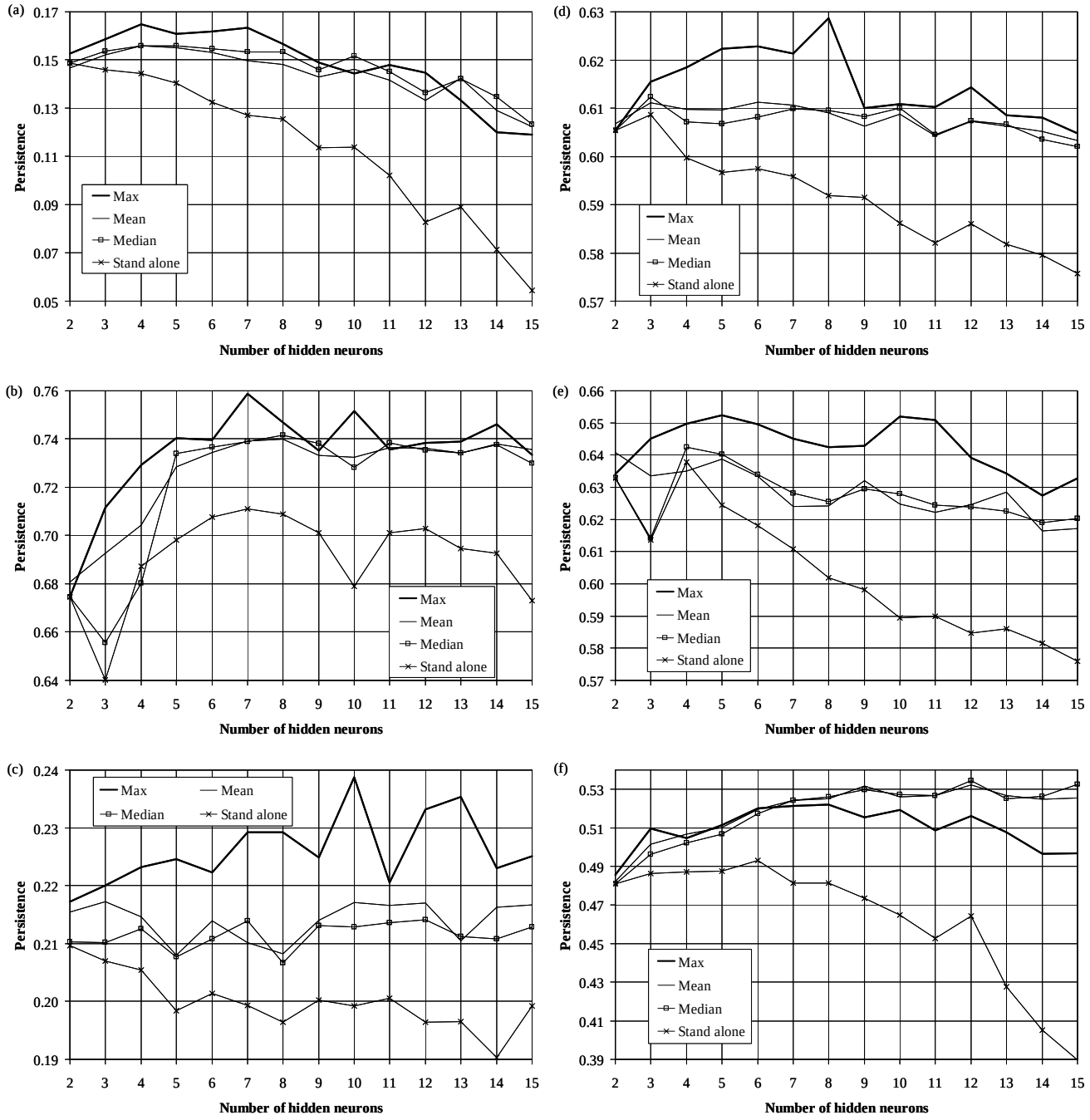

Fig. 3. *Persistence for the modelling configuration using stacking and the Levenberg-Marquardt training algorithm for the (a) Kavi, (b) Leaf, (c) Salt Fork, (d) San Juan, (e) Serein and (f) Volpajola rivers.*

VARIABILITY OF THE STREAMFLOW ESTIMATES AND OF MODEL PERFORMANCE

The results from Figs. 3 to 5 and Table 3 provide good trends about model performance, but do not provide much information on the variability of the performance and none about the variability of streamflow estimates. As a measure of performance variability, a ratio similar to the coefficient of variation is employed, in which the difference between the 75th and 25th persistence percentiles is divided by the median persistence (50th percentile). Percentiles are applied because they are usually less influenced by extreme values than the mean and the standard deviation. This ratio is the equivalent of the standardised measure of the range between confidence intervals, and is applied to all modelling

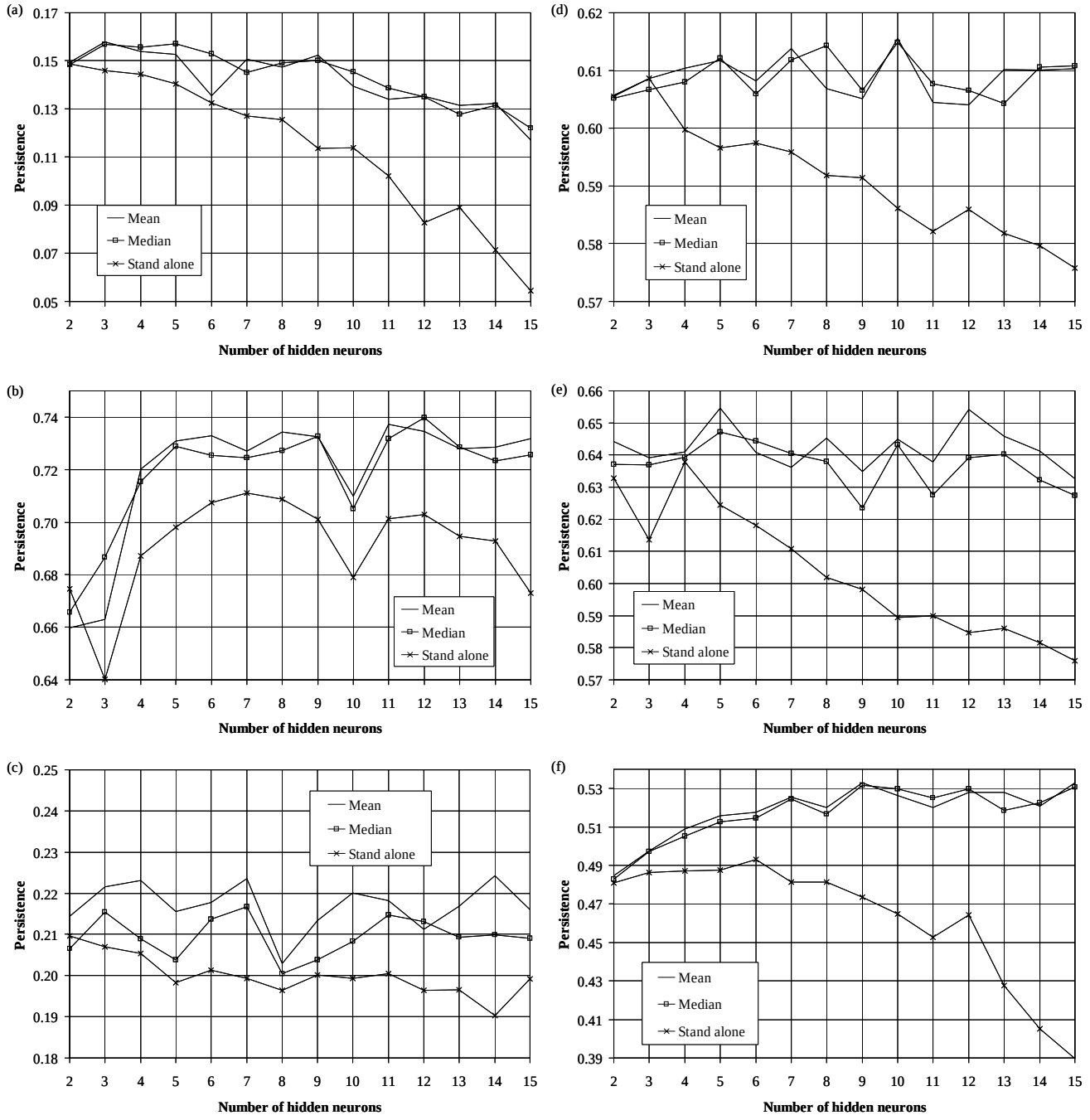

Fig. 4. *Persistence for the modelling configuration using bagging and the Levenberg-Marquardt training algorithm for the (a) Kavi, (b) Leaf, (c) Salt Fork, (d) San Juan, (e) Serein and (f) Volpajola rivers.*

configurations, with respect to the number of hidden neurons. By this ratio, the results of Fig. 2 can be summarised and compared with other modelling configuration, as in Fig. 6. The terms LM, LS and BR in the legend of Fig. 6 refer to the training procedures employed, respectively Levenberg-Marquardt alone, Levenberg-Marquardt with stop training and Levenberg-Marquardt with Bayesian regularisation. The terms 0, 1 and 2 in the legend refer to stacking (no sampling of the original training sets), bagging (purely random sampling) and boosting (stratified sampling).

The modelling configurations with the largest variability are those that use the Levenberg-Marquardt algorithm alone, the worst cases being consistently those that use a sampling

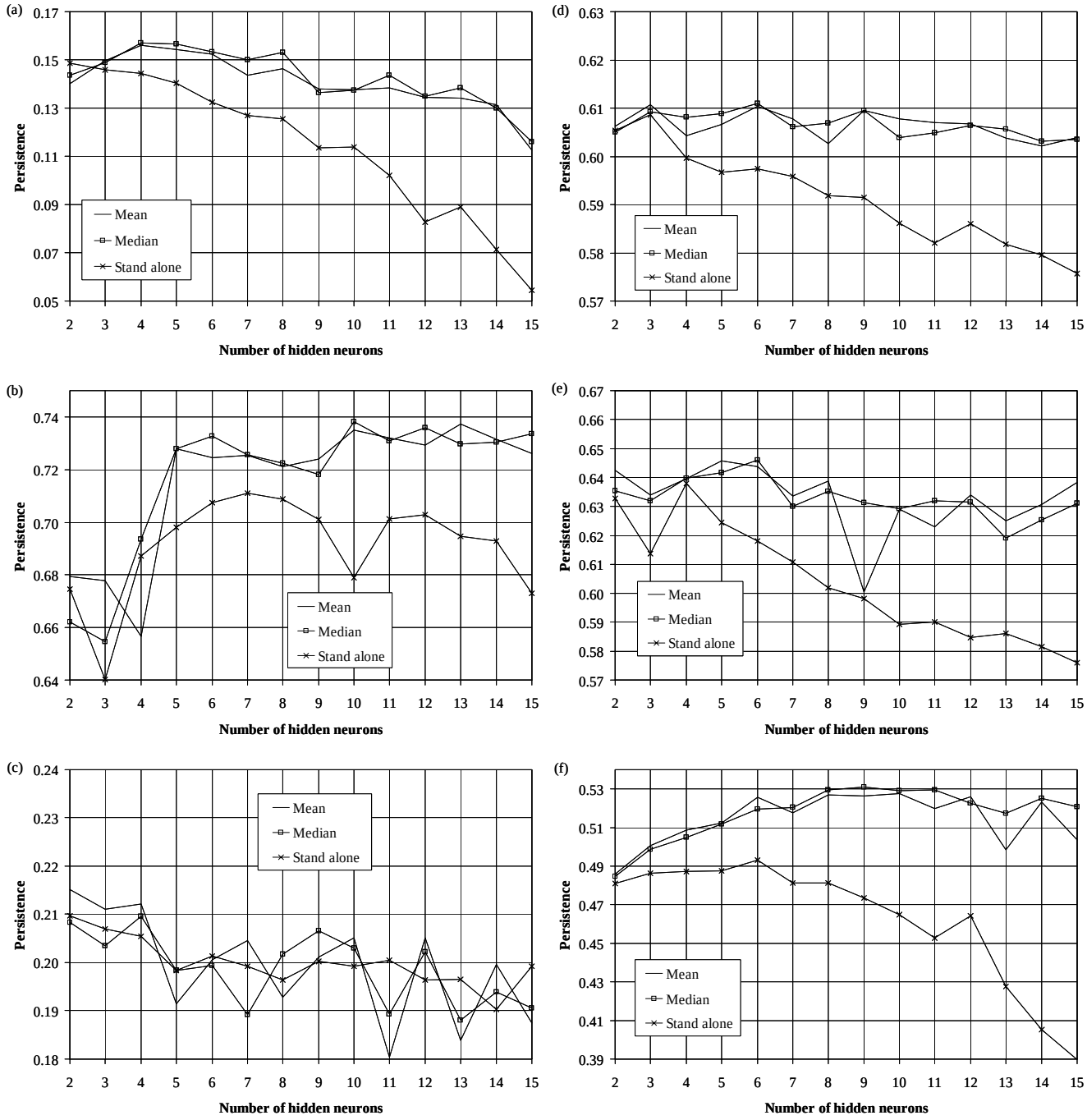

Fig. 5. *Persistence for the modelling configuration using boosting and the Levenberg-Marquardt training algorithm for the (a) Kavi, (b) Leaf, (c) Salt Fork, (d) San Juan, (e) Serein and (f) Volpajola rivers.*

process for the construction of the training sets (i.e. LM1 an LM2). Bagging and boosting can produce stand-alone models that perform somewhat poorly as a result of the sampling process, which may duplicate input vectors (i.e. it is a random selection with replacement) and consequently reduce the diversity of the training sets. This shortfall is, however, compensated afterwards by the aggregation of the stand-alone models, as demonstrated in Table 3. Performance variability is particularly large with Kavi, Salt Fork and Volpajola. In the case of Kavi, the ratio is even greater than 1 when the number of hidden neurons is large (see Fig. 6a). For the first two rivers, the poor performance of the model originating from the reduced explaining power of the inputs vector can be blamed for the resultant large

Table 3. Average persistence for all modeling configurations, estimators and rivers.

Rivers	*Estim.*	*Levenberg-Marquardt*			*Stop training*			*Bayesian regularisation*		
		Stack.	*Bag.*	*Boost.*	*Stack.*	*Bag.*	*Boost.*	*Stack.*	*Bag.*	*Boost.*
Kavi	Alone	0.114	——-	——-	0.115	——-	——-	0.137	——-	——-
	Max	0.148	——-	——-	0.143	——-	——-	0.154	——-	——-
	Mean	0.144	0.142	0.141	0.135	0.129	0.129	0.145	0.143	0.141
	Median	0.147	0.144	0.143	0.135	0.129	0.128	0.147	0.144	0.142
Leaf	Alone	0.691	——-	——-	0.700	——-	——-	0.692	——-	——-
	Max	0.734	——-	——-	0.735	——-	——-	0.741	——-	——-
	Mean	0.726	0.719	0.716	0.725	0.707	0.722	0.713	0.703	0.723
	Median	0.722	0.719	0.717	0.723	0.708	0.721	0.709	0.699	0.717
Salt Fork	Alone	0.200	——-	——-	0.195	——-	——-	0.207	——-	——-
	Max	0.226	——-	——-	0.213	——-	——-	0.222	——-	——-
	Mean	0.214	0.217	0.199	0.203	0.201	0.201	0.213	0.215	0.214
	Median	0.211	0.210	0.199	0.203	0.200	0.200	0.212	0.211	0.212
San Juan	Alone	0.591	——-	——-	0.601	——-	——-	0.604	——-	——-
	Max	0.614	——-	——-	0.617	——-	——-	0.617	——-	——-
	Mean	0.608	0.609	0.606	0.611	0.612	0.608	0.610	0.610	0.607
	Median	0.607	0.609	0.607	0.610	0.612	0.608	0.610	0.610	0.606
Serein	Alone	0.603	——-	——-	0.643	——-	——-	0.639	——-	——-
	Max	0.643	——-	——-	0.665	——-	——-	0.663	——-	——-
	Mean	0.628	0.642	0.633	0.665	0.663	0.665	0.654	0.659	0.650
	Median	0.627	0.637	0.633	0.665	0.663	0.665	0.651	0.657	0.648
Volpajola	Alone	0.463	——-	——-	0.485	——-	——-	0.504	——-	——-
	Max	0.510	——-	——-	0.510	——-	——-	0.520	——-	——-
	Mean	0.519	0.519	0.515	0.509	0.505	0.509	0.516	0.515	0.515
	Median	0.518	0.517	0.517	0.506	0.503	0.507	0.515	0.514	0.515

Note: Estim. for estimators, Stack. for stacking, Bag. for bagging, Boost. for boosting, and Alone for the median of stand-alone models.

variability. In the case of Volpajola, the short duration of the database can be a factor. Particularly when a sampling procedure is employed (i.e. LM1 and LM2), a short database increases the risk of duplication of input vectors, leading to less diverse training sets and larger variability in performance. .

Stop training (LS) and Bayesian regularisation (BR) lead to reduced performance variability for stand-alone models, even where a sampling procedure is in place. Not only can these generalisation approaches related to the training process produce stand-alone models that perform better than stand-alone models using only the Levenberg-Marquardt algorithm in the majority of the rivers, but they also lead to a more stable performance. Stop training or Bayesian regularisation with either bagging or boosting can, thus, produces better and more stable performances. Such a combination would be optimal with a small number of hidden neurons in the models, for variability in performance tends to increases as the number of hidden neurons increases (see Fig. 6).

The ratio used in Fig. 6 with the persistence index has also been applied to streamflow estimates for all the rivers. Indeed, the variability of the streamflow estimates can be as indicative as performance variability for checking the validity of stacking, bagging and boosting. The ratio is calculated from the daily estimates of all the 50 stand-alone models, per modelling configuration and with respect to the number of hidden neurons. The medians of the ratios obtained are illustrated in Fig. 7, the legend of which is the same as that of Fig. 6. Even though it is not consistent from one river to another, the cases that involve the use of stacking show the smallest variability of the streamflow estimates, followed by the cases of bagging, then boosting. As with performance variability, the sampling processes, with the possibility of duplications that reduce the diversity of the inputs vectors, affect the models. With boosting, duplication is more prevalent, particularly for extreme streamflow events, which is why the variability of streamflow estimates is often greatest with this generalisation approach.

Large variability in streamflow estimates is related to poor modelling performance. Figure 8 shows streamflow estimate variability with respect to persistence. The cases of Kavi

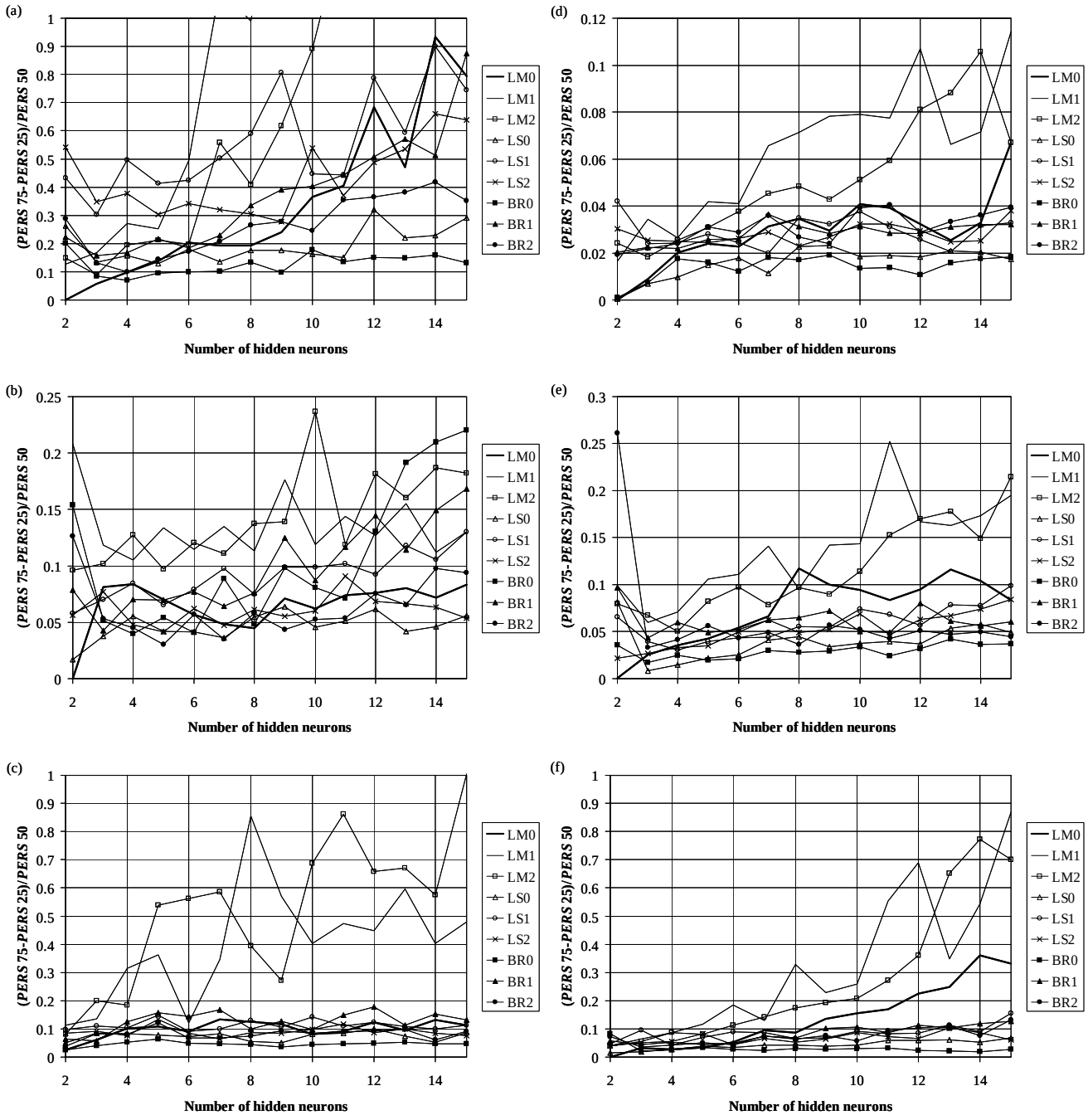

Fig. 6. *Variability of persistence for all modelling configurations for the (a) Kavi, (b) Leaf, (c) Salt Fork, (d) San Juan, (e) Serein and (f) Volpajola rivers.*

and Salt Fork are a combination of poor performance with large streamflow estimate variability, while the cases of Leaf, San Juan, Serein and Volpajola combine relatively good performances with smaller streamflow estimate variability. This indicates the limitation of generalisation approaches, which cannot perform fully if the input data to the models cannot well explain the outputs.

STOP-TRAINING

To complete the presentation of the results, Fig. 9 and Table 4 show the effect of stop-training on the number of epochs needed to perform the optimisation of the weights of the models. Figure 9 shows the number of epochs needed to complete the optimisation process for the modelling configuration where stacking is employed (i.e. always the

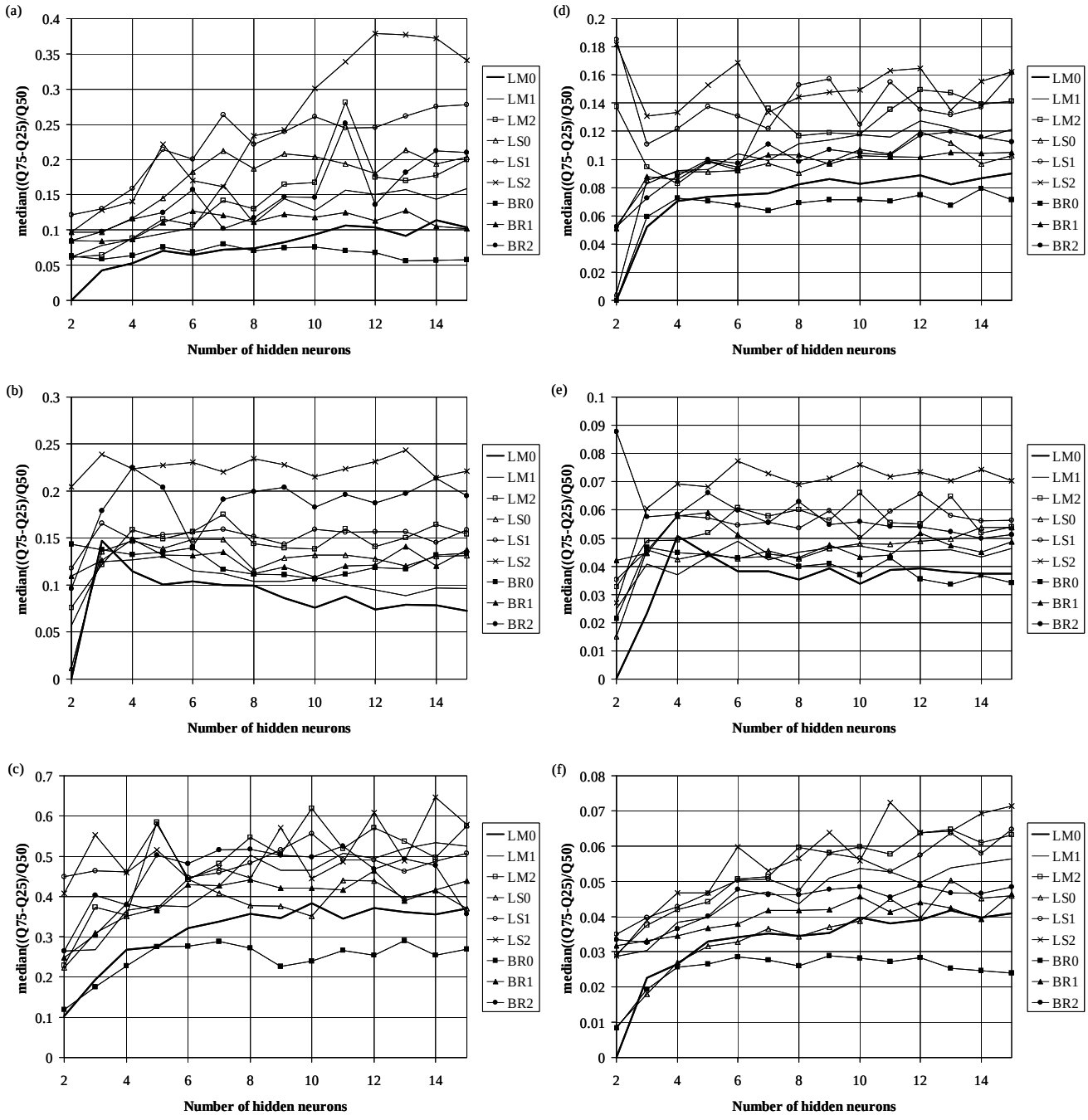

Fig. 7. *Variability of streamflow predictions for all modelling configurations for the (a) Kavi, (b) Leaf, (c) Salt Fork, (d) San Juan, (e) Serein and (f) Volpajola rivers.*

original training sets used). The maximum number of epochs has been set at one hundred and Fig. 9 shows that the limit is reached only on a reduced number of occasions. Table 4 gives the median of the number of epochs needed for the completion of the optimisation as well as the number of times the 100-limit is reached, for all rivers, with respect to the use of stacking, bagging and boosting. Sampling the training sets has little effect on the number of epochs needed, for the cases of bagging and boosting require as many epochs as the cases of stacking. There is a relation between modelling performance and the number of epoch needed in the optimisation process. Indeed, the poorer the modelling performance, the smaller is the number of epochs employed prior to stop-training, and the less frequent are the instances that reach the 100-limit.

Model complexity and the explaining power of the input

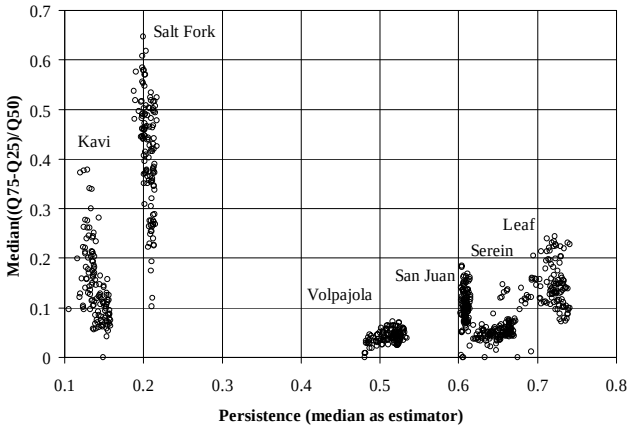

Fig. 8. *Variability of streamflow predictions as a function of persistence.*

Table 4. Number of epochs employed with stop training.

Case	*Kavi*	*Leaf*	*Salt Fork*	*San Juan*	*Serein*	*Volpajola*
MEDIAN NUMBER OF EPOCHS						
stacking	16	31	12	17	28	16
bagging	15	27	12	16	24	15
boosting	15	22	12	16	23	15
NUMBER OF TIMES THE 100-LIMIT IS REACHED						
stacking	7	14	3	1	42	26
bagging	2	21	3	2	34	6
boosting	2	7	5	4	22	13

vectors regulate when the optimisation of the weights halts with stop-training. A complex model (i.e. a large number of hidden neurons) with inputs vectors that have little explaining power lead to a rapid end of the optimisation process to ensure the generality of the model. Stop-training does not necessarily produce enhanced performance, as can be seen from Table 3 with the stand-alone models for Kavi and Salt Fork. Stop-training does improve modelling performance for all the other rivers, and this helps to legitimise this generalisation approach. Stop-training seems much less critical for the rivers where models perform well (e.g. Leaf and Serein), although it may simply mean that a larger number of epochs must be set as a limit.

Discussion and conclusion

Five generalisation approaches, stop-training, Bayesian regularisation, stacking, bagging and boosting have been tested with multi-layer perceptron neural networks for the prediction one-day ahead of streamflows on six hydrologically different catchments. Stop training and Bayesian regularisation affect the training procedure and ensure that the models do not become too specific for the training data. Stacking, bagging and boosting consist of the aggregation of the results of several models to obtain global estimates, and are distinguished from each other by the process employed to construct the training data set used in the calibration of these models.

While relatively extensive, this work does not yet provide strict guidelines on the use of generalisation approaches for neural network applications to streamflow modelling. The ultimate aim of this paper is to generate interest in generalisation approaches and encourage others to pursue their own comparisons so as to improve the knowledge base of these approaches. Admittedly, the efficacy of each of these approaches may vary from one application to another and may depend on which variant of the approach is employed. Nevertheless, the results obtained from the specific applications in this study lead to the following conclusions. All generalisation approaches improve modelling performance when compared with stand-alone models. Stacking, bagging and boosting provide the largest improvement from standard models, compared with stop-training and Bayesian regularisation. Stacking performs best although the benefit in performance is only slight compared with bagging and boosting, and is not consistent from one catchment to another.

Bagging and boosting have often been the main approaches selected in comparisons between generalisation approaches accomplished over the years (see for example Yang *et al.*, 1998; Bauer and Kohavi, 1999; Cunningham *et al.*, 2000; Zhou *et al.*, 2002; Sohn and Lee, 2003). Both approaches should be considered equivalent in terms of performance, as mentioned by Bauer and Kohavi (1999); the present work confirms this conclusion for bagging, boosting, and stacking. One must be careful with boosting, as this approach can be sensitive to aberrant inputs, such as outliers (Bauer and Kohavi, 1999; Zhou *et al.,* 2002), which may affect performance negatively. The particular form of boosting employed here (stratified sampling) is less affected by outliers, because the sampling process ensures an equal representation of all classes of inputs, thus reducing the effect of aberrant data. This might not necessarily occur with more traditional forms of boosting. Aberrant inputs can be very important in hydrology and water resources, where it is sometimes difficult to determine whether extremely large streamflow values are legitimate or erroneous.

Each type of generalisation approach must be analysed separately if one is concerned about performance variability or streamflow estimates. The improvement in performance with bagging and boosting comes from the aggregation of

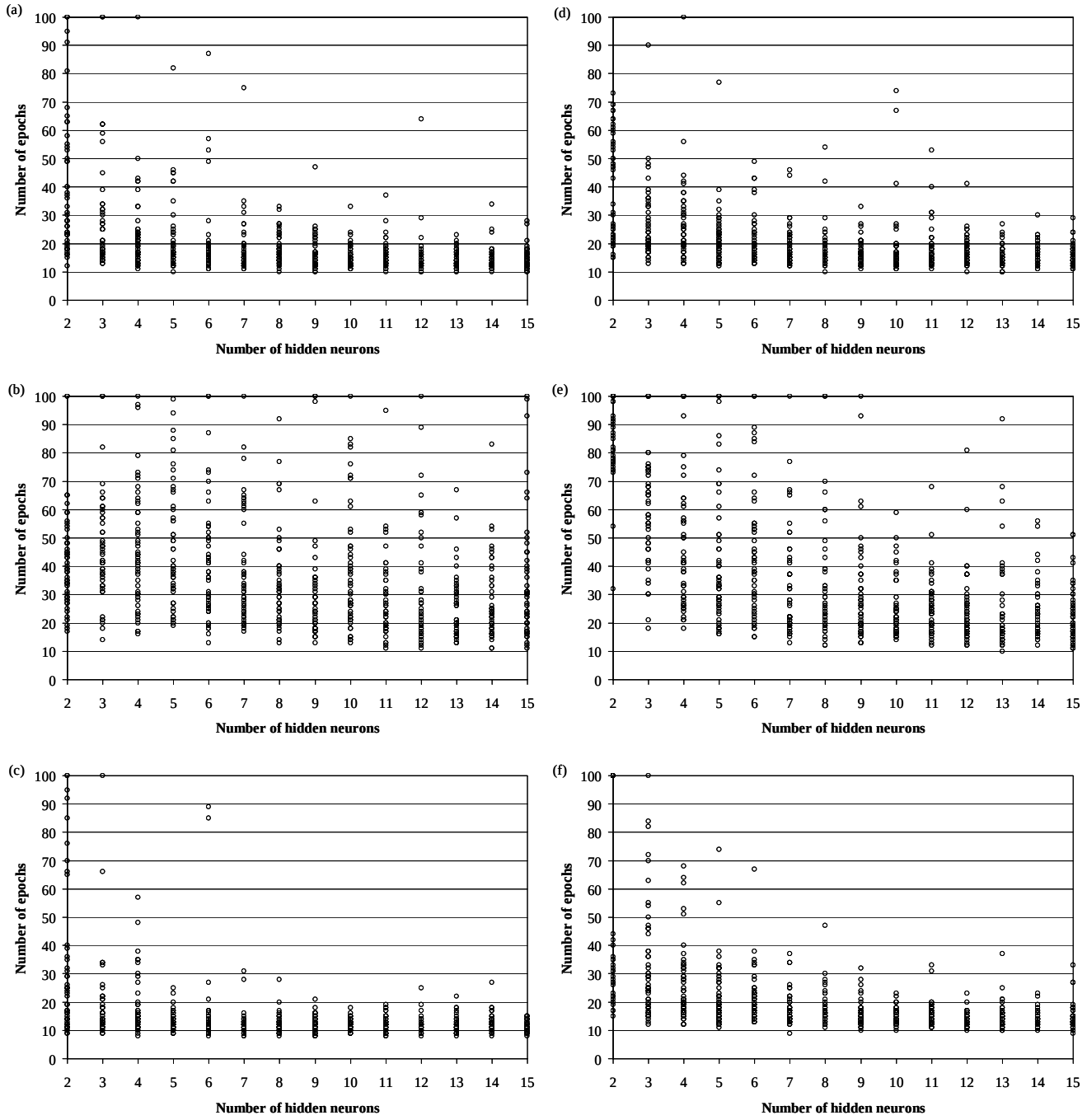

Fig. 9. *Number of epochs needed for stop training for the (a) Kavi, (b) Leaf, (c) Salt Fork, (d) San Juan, (e) Serein and (f) Volpajola rivers.*

the models. When considered alone, these models usually exhibit larger variability in performance and streamflow estimates than is the case with the models for stacking. This variability is due to the process employed in constructing the training sets. The original training set is employed for the calibration of all the models with stacking. Sampling with replacement from the original training set is used in constructing one training set for each model with bagging and boosting; allowing replacement duplicates input vectors, which consequently reduce the diversity of the training sets. Even though they do not perform as well as stacking, bagging and boosting, stop training and Bayesian regularisation reduce the variability of model performance and streamflow estimates. As a rule, variability increases with the number of hidden neurons for the models and this compels the use of simple models with a small number of

hidden neurons. Variability is also greater for cases where models are not performing well, as with the Kavi and Salt Fork rivers; this indicates the limitations of generalisation, which cannot help much if the data input to the models do not explain the outputs well..

Generalisation approaches are a necessity for modelling, including streamflow modelling. The benefit of generalisation in terms of modelling performance is demonstrated by the results of this study. The necessity for generalisation is also indicated by the additional results for stop training, that is, by how regularly training is stopped well before the limit is reached for the number of epochs, particularly for catchments where models do not perform well. Any generalisation approach tested in this work has improved performance. However, for a good combination of improvement and stability in modelling performance, the joint use of stop training or Bayesian regularisation with either bagging or boosting is recommended.

Acknowledgments

The authors acknowledge the Natural Science and Engineering Research Council of Canada, and the Fond québecois de la recherche sur la nature et les technologies, for providing funding for this work in the form of research grants. The authors are also grateful of the insights of the reviewers who evaluated this paper.

References

Abrahart, R.J. and See, L., 2000. Comparing neural network and autoregressive moving average techniques for the provision of continuous river flow forecasts in two contrasting catchments. *Hydrol. Process.*, **14**, 2157–2172.

Abrahart, R.J. and See, L., 2002. Multi-model data fusion for river flow forecasting: An evaluation of six alternative methods based on two contrasting catchments. *Hydrol. Earth Syst. Sci.,* **6**, 655–670.

Agrafiotis, D.K., Cedeno, W. and Lobanov, V., 2002. On the use of neural network ensembles in QSAR and QSPR. *J. Chem. Inform. Comput. Sci*., **42**, 903–911.

Amari, S.I., Murata, N., Muller, K.R., Finke, M. and Yang, H.H., 1997. Asymptotic statistical theory of over training and cross-validation. *IEEE Trans. Neural Networks*, **8**, 985–995.

Anctil, F., Perrin, P. and Andréassian, V., 2003. ANN output updating of lumped conceptual rainfall/runoff forecasting models. *J. Amer. Water Resour. Assoc*., **39**, 1269–1279.

Anctil, F., Perrin, P. and Andréassian, V., 2004. Impact of the length of observed records on the performance of ANN and of conceptual parsimonious rainfall-runoff forecasting models. *Environ. Model. Software*, **19**, 357–368.

ASCE, 2000a. Artificial neural networks in hydrology, I: Preliminary concepts. *J. Hydrol. Eng. - ASCE,* **5**, 115–123.

ASCE, 2000b. Artificial neural networks in hydrology, II: Hydrologic applications. *J. Hydrol. Eng. - ASCE,* **5**, 124–137.

Babovic, V., CaHisares, R., Jensen, H.R. and A. Klinting, 2001. Neural networks as routine for error updating of numerical models. *J. Hydraul. Eng. - ASCE,* **127**, 181–193.

Bauer, E. and Kohavi, R., 1999. An empirical comparison of voting classification algorithms: bagging, boosting, and variants. *Mach. Learn*., **36**, 105–139.

Bowden, G.J., Maier, H.R. and Dandy, G.C., 2002. Optimal division of data for neural network models in water resources applications. *Water Resour. Res.*, **38**, 2.1–2.11.

Breiman, L., 1996. Bagging predictors. *Mach. Learn.*, **24**, 123–140.

Breiman, L., 2000. Randomising outputs to increase prediction accuracy. *Mach. Learn.*, **40**, 229–242.

Breiman, L., 2001. Using iterated bagging to debias regressions. *Mach. Learn.,* **45**, 262–277.

Cannon, A.J. and Whitfield, P.H., 2002. Downscaling recent streamflow conditions in British Columbia, Canada using ensemble neural network models. *J. Hydrol*., **259**, 136–151.

Cavadias, G. and Morin, G., 1986. The combination of simulated discharges of hydrological models: Application to the WMO intercomparison of conceptual models of snowmelt runoff. *Nord. Hydrol*., **17**, 21–32.

Chan, S.S.H., Ngan, H.W. and Rad, A.B., 2003. Improving Bayesian regularisation of ANN via pre-training with early-stopping. *Neural Process. Lett*., **18**, 29–34.

Chawla, N.V., Moore, T.E., Hall, L.O., Bowyer, K.W., Kegelmeyer, W.P. and Springer, C., 2003. Distributed learning with bagging-like performance. *Pattern Recognition Lett*., **24**, 455–471.

Coulibaly, P., Anctil, F. and Bobée, B., 1999. Prévision hydrologique par réseaux de neurones artificiels: etats de l'art. *Can. J. Civil Eng.,* **26**, 293–304.

Coulibaly, P., Anctil, F. and Bobée, B., 2000. Daily reservoir inflow forecasting using artificial neural networks with stopped training approach. *J. Hydrol.*, **230**, 244–257.

Coulibaly, P., Anctil, F., Aravena, R. and Bobée, B, 2001. Artificial neural network modeling of water table depth fluctuations. *Water Resour. Res.*, **37**, 885–896.

Cunningham, P., Carney, J. and Jacob, S., 2000. Stability problems with artificial neural networks and the ensemble solution. *Artif. Intell. Med.*, **20**, 217–225.

Dawson, C.W. and Wilby, R.L., 2001. Hydrological modelling using artificial neural networks. *Progr. Phys. Geogr.*, **25**, 80–108.

Drucker, H., 2002. Effect of pruning and early stopping on performance of a boosting ensemble. *Comput. Statist. Data Analysis*, **38**, 393–406.

Foresee, F.D. and Hagan, M.T., 1997. *Gauss-Newton approximation to Bayesian learning*. Proceedings, 1997 IEEE International Conference on Neural Networks, Houston, TX, 3, 1930–1935.

Freund, Y. and Schapire, R.E., 1997. A decision-theoretic generalisation of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, **55**, 119–139.

Geman, S., Bienenstock, E. and Doursat, R., 1992. Neural networks and the bias/variance dilemma. *Neural Comput.*, **4**, 1–58.

Gençai, R. and Qi, M., 2001. Pricing and hedging derivative securities with neural networks: Bayesian regularisation, early stopping, and bagging. *IEEE Trans. Neural Networks*, **12**, 726–734.

Goh, W.Y., Lim, C.P. and Peh, K.K., 2003. Predicting drug dissolution profiles with and ensemble of boosted neural networks: a time series approach. *IEEE Trans. Neural Networks*, **14**, 459–463.

Golob, R., Stokelj, T. and Grgic, D., 1998. Neural-network-based water inflow forecasting. *Control engineering practice*, **6**, 593–600.

Gots, R., Steiner, B., Sievers, S., Friesel, P., Roch, K., Schworer, R. and Haag, F., 1998. Dioxin, Dioxin-like PCBS and organic compounds in the River Elbe and the Hamburg Harbour: Identification of sources. *Water Sci. Technol*., **37**, 207–215.

Hall, M.J. and Minns, A.W., 1999. The classification of hydrologically homogeneous regions. *Hydrolog. Sci. J.***, 44**, 693–704.

Hu, T.S., Lam, K.C. and Ng, S.T., 2001. River flow time series prediction with a range-dependent neural network. *Hydrolog. Sci. J*., **46**, 729–745.

Iyer, M.S. and Rhinehart, R.R., 1999. A method to determine the required number of neural-network training repetitions. *IEEE Trans. Neural Networks*, **10**, 427–432.

Kim, G. and Barros, A.P., 2001. Quantitative flood forecasting using multisensor data and neural networks. *J. Hydrol.*, **246**, 45–62.

Kitanidis, P.K. and Bras, R.L., 1980. Real-time forecasting with a conceptual hydrologic model: 2. Applications and results. *Water Resour. Res.*, **16**, 1034–1044.

Kohonen, T., 1990. The self-organising map. *Proc. IEEE*, **79,** 1464–1480.

Kohonen, T., 1997. *Self-Organising Maps, Second Edition*. Springer, Berlin, Germany.

Kuligowski, R.J. and Barros, A.P., 1998. Using artificial neural networks to estimate missing rainfall data. *J. Amer. Water Resour. Assoc.*, **34**, 1437–1447.

Lajbcygier, P.R. and Connor, J.T., 1997. Improved option pricing using artificial neural networks and bootstrap methods. *Int. J. Neural Systems***, 8**, 457–471.

Liong, S.Y., Lim, W.H., Kojiri, T. and Hori, T., 2000. Advance flood forecasting for flood stricken Bangladesh with a fussy reasoning method. *Hydrolog. Process.*, **14**, 431–448.

Lopes-Sabater, C.J., Renard, K.G. and Lopes, V.L., 2002. Neural-network-based algorithms of hydraulic roughness for overland flow. *Trans. Amer. Soc. Agr. Eng*., **45**, 661–667.

Luk, K.C., Ball, J.E. and Sharma, A., 2000. A study of optimal lag and spatial inputs to artificial neural network for rainfall forecasting. *J. Hydrol.*, **227,** 56–65.

MacKay, D.J.C., 1992. Bayesian interpolation. *Neural Comput.*, **4**, 415–447.

MacNamee, B., Cunningham, P., Byrne, S. and Corrigan, O.I., 2002. The problem of bias in training data in regression problems in medical decision support. *Artif. Intell. Med.,* **24,** 51–70.

Maier, H.R. and Dandy, G.C., 1997. Modelling cyanobacteria (blue-green algae) in the River Murray using artificial neural networks. *Math. Comput. Simulat.,* **43**, 377–386.

Maier, H.R. and Dandy, G.C., 2000. Neural networks for prediction and forecasting of water resources variables; Review of modelling issues and applications. *Environ. Model. Software,* **15**, 101–124.

Medeiros, M.C., Veiga, A. and Pedreira, C.E., 2001. Modeling exchange rates: Smooth transitions, neural networks, and linear models. *IEEE Trans. Neural Networks*, **12**, 755–764.

Morse, B., Hessami, M. and Bourel, C., 2003. Mapping environmental conditions in the St. Lawrence River onto ice parameters using artificial neural networks to predict ice jams. *Can. J. Civil Eng*., **30**, 758–765.

Pal, M. and Mather, P.M., 2003. An assessment of the effectiveness of decision tree methods for land cover classification. *Remote Sens. Environ*., **86**, 554–565.

Papadopoulos, G., Edwards, P.J. and Murray, A.F., 2000. Confidence estimation methods for neural networks: A practical comparison. *IEEE Trans.Neural Networks*, **12**, 1278–1287.

Raman, H. and Sunilkumar, N., 1995. Multivariate modelling of water resources time series using artificial neural networks. *Hydrolog. Sci. J*., **40,** 145–163.

Raviv, Y. and Intrator, N., 1996. Bootstrapping with noise: An effective regularisation technique. *Connection Sci.***, 8**, 355-372.

Schwenk, H. and Bengio, Y., 2000. Boosting neural networks. *Neural Comput.*, **12**, 1869–1887.

See, L. and Abrahart, R.J., 2001. Multi-model data fusion for hydrological forecasting. *Comput. Geosci.,* **27**, 987–994.

Shamseldin, A.Y., Nasr, A.E. and O'Connor, K.M., 2002. Comparison of different forms of the multi-layer feed-forward neural network method used for river flow forecasting. *Hydrol. Earth Syst. Sci.*, **6**, 671–884.

Shou, S.H., Wu, J. and Tang, W., 2002. Ensembling neural networks: Many could be better than all. *Artif. Intell*., **137,** 239–263.

Shu, C. and Burn, D.H., 2004. Artificial neural network ensembles and their application in pooled flood frequency analysis. *Water Resour. Res.,* **40**, W09301, doi: 10.1029/2003WR002816.

Sohn, S.Y. and Lee, S.H., 2003. Data fusion, ensemble and clustering to improve the classification accuracy for the severity of road traffic accidents in Korea. *Safety Sci*., **41**, 1–14.

Tan, Y. and Van Cauwenberghe, A., 1999. Neural-network-based d-step-ahead predictors for nonlinear systems with time delay. *Eng. Appl. Artif. Intell*., **12**, 21–35.

Vlassides, S., Ferrier, J.G. and Block, D.E., 2001. Using historical data for bioprocess optimisation: Modeling wine characteristics using artificial neural networks and archived process information. *Biotechnol. Bioeng*., **73**, 55–68.

WMO (World Meteorological Organisation), 1992. *Simulated real-time intercomparison of hydrological models.* Operational Hydrology report no 38, WMO report no 779. 241pp.

Wolpert, D.H., 1992. Stacked generalisation. *Neural Networks,* **5**, 241–259.

Wolpert, D.H. and Macready, W.G., 1999. An efficient method to estimate bagging's generalisation error. *Mach. Learn*., **35**, 41–55.

Xiao, R.R. and Chandrasekar, V., 1997. Development of a neural network based algorithm for rainfall estimation from radar observation. *IEEE Trans. Geosci. Remote Sens*., **35**, 160–171.

Yabunaka, K.I., Hosomi, M. and Murakami, A., 1997. Novel application of back-propagation artificial neural network model formulated to predict algal bloom. *Water Sci. Technol.,* **36**, 89–97.

Yang, H.H., Murata, N. and Amari, S.I., 1998. Statistical inference: Learning in artificial neural networks. *Trends Cognitive Sci.*, **2,** 4–10.