AUSTRIAN JOURNAL OF STATISTICS
Volume 27 (1998), Numbers 1&2, 73–82

Modelling Sewerage Systems with an Artificial Neural Network

Stephen Langdell and John Mason
University of Huddersfield, U.K.
Roland Price
IHE Delft, The Netherlands

Abstract: A new approach to non-parametric modelling of simulated flows in an urban drainage system is presented. The new model uses the *storage* of the drainage system as an input variable, in addition to the rainfall input, to a radial basis function neural network. Previous attempts to model this problem, using only rainfall inputs, have been unsuccessful because of the highly non-linear relationship between a rainfall event and the one dimensional flow associated with it (Figure 2). Compared with the traditional parametric model, the NN reduces greatly the time taken to produce simulated flows, and is much less expensive to set up. Furthermore, with respect to traditional methods, the NN predicts flows without a significant loss in accuracy. A practical use of the method is highlighted which aids hydraulic engineers in the making of environmentally sensitive decisions.

Keywords: Radial Basis Functions, Urban Drainage Systems, Storage Capacity of Sewers.

1 Introduction

The relationship between the rainfall onto an urban catchment area and the one dimensional flow out of (or through) the corresponding drainage system may be described by a system of equations that includes a conceptual storage equation for the surface run-off and the Saint Venant equations for flow in a conduit. In this paper simulated flows are generated by Hydroworks^{TM} Wallingford Software (1994) which adopts the Preissmann four point approximation to the Saint Venant equations as demonstrated by Kellagher (1994). However, producing these simulated flows is computationally intensive and, if we replace them with a model whose form is determined by the data (and not by the physical system), then we reduce greatly the time taken to reproduce the simulated flows.

Neural networks are non-parametric statistical regressors that have been used to approximate one dimensional flows out of drainage systems. Minns and Hall (1996) have modelled flows due to recorded rainfall incident on laboratory drainage systems using *multi-layer perceptron* NNs. Typically, laboratory drainage systems have few conduits and contain no significant storage, e.g. via the use of tanks. However, many real drainage systems contain some discrete storage elements and, therefore, water may stay in these systems for several hours after the rainfall has ceased (e.g. Figure 2). Thus, the relationship between rainfall onto the catchment area and the flow in a real drainage system may be highly non-linear. In the present paper, this highly non-linear relationship is modelled successfully using a non-parametric model consisting of radial basis functions (RBFs) arranged in a NN architecture. The centres of the radial basis functions are found by

clustering the input space using the self organizing map (SOM) algorithm proposed by Kohonen (1989). The complexity of the NN model is controlled by using leave-one-out (LOO) cross validation (Stone, 1974).

2 Theory

2.1 Rule Extraction

Neural networks are *trained* to extract a rule from a data set consisting of independent (input) and dependent (output) variables. The training process involves the determination of the NN's free parameters so that the minimum of an error function is found. Formally, we may say that within the universal set of input-output pairs, **U**, there exists a set, **R**, that is consistent with the mapping:

$$y = f(\mathbf{x}) + \epsilon,$$

from which the modeller selects a set **S**, a subset of **R**, which is to become the training set. The goal is to model the underlying function $f(\mathbf{x})$, without modelling the noise, ϵ, associated with training set output, y. After the set $\mathbf{S}$ has been used to determine the free parameters in the NN, via the application of an optimisation algorithm, then, the performance of the neural network may be measured on a disjoint subset of **R**, say **T**. The set **T** is used to *test* the performance of the NN, using the mean sum of squares error function as a measurement of accuracy.

2.2 Radial Basis Functions

Radial basis functions, as advocated by Powell (1985), provide an excellent means of interpolation in high dimensional spaces. All RBFs are non-linear functions of the distance between an input vector, $\mathbf{x}_i$, and the centres of the RBFs, $\boldsymbol{\mu}_j$, i.e. of the form

$$\phi(\|\mathbf{x}_i - \boldsymbol{\mu}_j\|),$$

where the norm, $\|\cdot\|$, is taken to be Euclidean. The non-linearity $\phi(\cdot)$ may be any function that satisfies Light's theorem (1992). Broomhead and Lowe (1988) adopted RBFs to approximate functions and data, and showed that they correspond to the NN architecture shown in Figure 1.

The figure depicts a fully-connected NN, i.e. each element of the input vector is connected to each RBF in the hidden layer and each RBF is connected to each output. Connections between the input and hidden layers are weighted according to m $d-$dimensional vectors, $\boldsymbol{\mu}_j$. Connections between the hidden and output layers are weighted according to m scalars, w_j. Hence, the equation describing the output of the RBF neural network is:

$$s(\mathbf{x}_i) = \sum_{j=1}^{m} w_j h_j, \quad \begin{cases} i = 1, 2, \ldots, p \\ j = 1, 2, \ldots, m \\ h_j = \phi(\|\mathbf{x}_i - \boldsymbol{\mu}_j\|) \end{cases}, \tag{1}$$

for an input vector, $\mathbf{x}_i = \{x_i^1, x_i^2, \ldots, x_i^d\}$. It may be seen from Figure 1 that if centres of the RBFs are determined before the NN is implemented, then the problem of finding

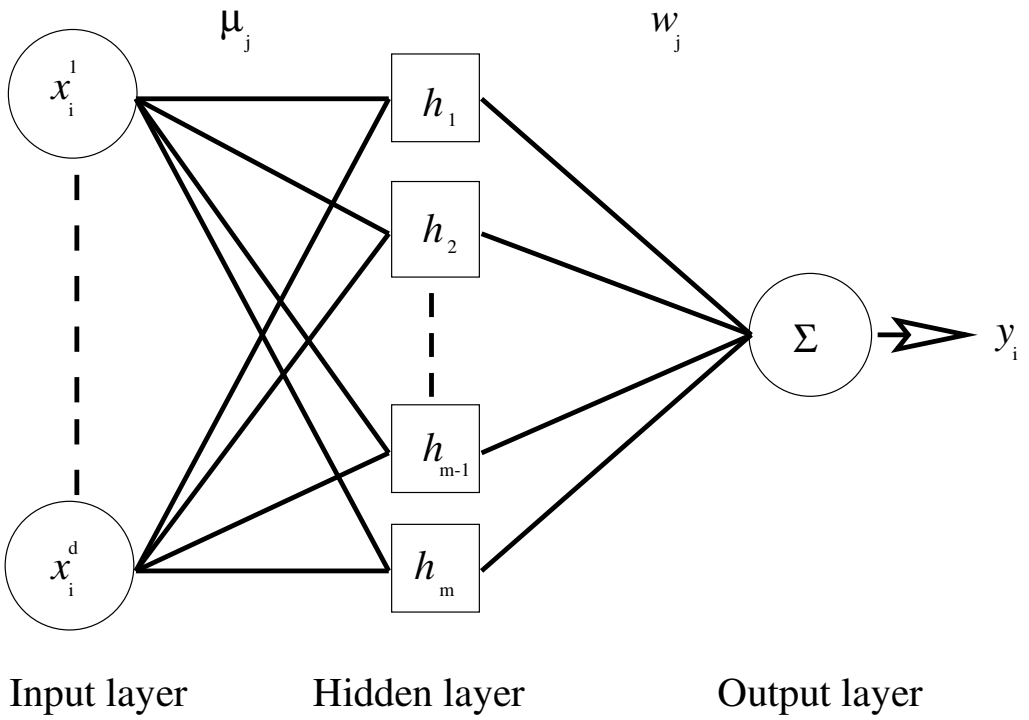

Figure 1: The architecture of a RBF neural network. The ith input pattern, $\mathbf{x}_i$ has components $\{x_i^1, x_i^2, \ldots, x_i^d\}$. The NN is fully connected, has m radial basis functions in the hidden layer and one output, s_i.

the weights between the hidden and output layers reduces to a linear problem – for which fast and efficient optimization techniques exist (Section 2.4). Furhermore a solution of 1 is a *best* solution (see Powell, 1985) that is, unlike multi-layer perceptron NNs, linear RBF networks do not get stuck in local minima during training. The centres, $\boldsymbol{\mu}_j$, may be taken to be a subset of the input data (Broomhead and Lowe, 1988), alternative and better results may be obtained by using a clustering algorithm, e.g. as described in Section 2.3.

2.3 A Clustering Algorithm to Determine the Centres of the RBFs

Self–organizing maps were introduced by Kohonen as a topology preserving clustering algorithm. In Kohonen's notation, m *codebook vectors* are sought that cluster a d-dimensional input space. These codebook vectors are the centres for the RBFs. In general, the algorithm compares an input vector $\mathbf{x} \in \mathbb{R}^n$ with each $\boldsymbol{\mu}_j$ and finds the codebook vector with the 'best match' . The measure of the best match is the smallest Euclidean distance between the input vector and the codebook vectors. Associating each codebook vector with a *node* in $\mathbb{R}^2$ enables a representation of the data $\{\mathbf{x}_i \in \mathbb{R}^d | i = 1, 2, \ldots, p\}$ to be examined on screen.

To determine the position of the $\boldsymbol{\mu}_j$'s in the input space:

1. Take random initial values for $\boldsymbol{\mu}_j, j = 1, 2, \ldots, m$

2. Choose, at random, an input vector $\mathbf{x}_i$ from the training set **S**

3. Let $\boldsymbol{\mu}_c$ be the codebook vector which minimizes: $\|\mathbf{x}_i - \boldsymbol{\mu}_j\|, \forall j$
4. The input vector $\mathbf{x}_i$ is then 'mapped' to the node associated with $\boldsymbol{\mu}_c$, i.e. the c^{th} node
5. Set: $\boldsymbol{\mu}_j(t+1) := \boldsymbol{\mu}_j(t) + g_{cj}(t)[\mathbf{x}(t) - \boldsymbol{\mu}_j(t)], \quad \forall\, j$
6. Repeat steps 2-5 until convergence

In step 5:

- t is a discrete time co-ordinate.
- g_{cj} is a neighbourhood kernel.

The neighbourhood kernel is taken as a function, over the two dimensional lattice, that has a maximum at g_{cc}. For example, the Gaussian function is often used. Hence, from step 5, it may be seen that $\boldsymbol{\mu}_c$ receives the greatest update. Conversely, as the distance from c on the lattice to the other nodes increases, so the update has less effect on the codebook vectors. Hence, the SOM preserves the topology of the mapping by encouraging similar responses (on the two dimensional lattice) for similar input vectors, $\mathbf{x}_i$. In practice, this algorithm is used to set the centres, $\boldsymbol{\mu}_j$, in Figure 1. However, it is also important for the modeller to visualize the data, so that decisions may be made governing the size, m, of the set of codebook vectors. Once the $\boldsymbol{\mu}$'s have been determined, it may be seen from Figure 1 that the problem of determining the second layer of weights, w_j reduces to solving a set of overdetermined linear equations. This may be achieved using the least squares method, as described in Section 2.4, and results in a best approximation in the Euclidean norm.

2.4 Least Squares Optimization

If the physical system to be modelled is given by the mapping:

$$y_i = f(\mathbf{x}_i), \quad i = 1, 2, \ldots, p$$

and the non-parametric model is described by (1), then the minimum of an error function, taken over the p vectors in the training set, is sought. For a NN model of m RBFs, the error function is:

$$E = \frac{1}{2}\sum_{i=1}^{p}(y_i - s_i)^2 + \lambda\frac{1}{2}\sum_{j=1}^{m} w_j^2, \tag{2}$$

where s_i is the NN output. This error function is used in ridge regression and includes an additional weight penalty term controlled by the value of the regularisation parameter, λ. In matrix form the solution of the normal equations is given by

$$\mathbf{w} = (\mathbf{H}^T\mathbf{H} + \mathbf{I}\lambda)^{-1}\mathbf{H}^T y, \tag{3}$$

where $\mathbf{I}$ is the unit matrix of m rows and m columns, $\mathbf{w}$ is a m by 1 vector of weights and $\mathbf{H}_{p \text{ x } m}$ is defined by

$$\mathbf{H} = \begin{bmatrix} h_1(\mathbf{x}_1) & h_2(\mathbf{x}_1) & \ldots & h_m(\mathbf{x}_1) \\ h_1(\mathbf{x}_2) & h_2(\mathbf{x}_2) & \ldots & h_m(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ h_1(\mathbf{x}_p) & h_2(\mathbf{x}_p) & \ldots & h_m(\mathbf{x}_p) \end{bmatrix},$$

with $h_j(\mathbf{x}_i) = \phi(\|\mathbf{x}_i - \boldsymbol{\mu}_j\|, \quad j = 1, 2, \ldots, m.$

2.5 Forward Subset Selection of RBFs

It is noted that, in order to find the minimum of (2) using (3), the appropriate number of RBFs, m, needs to be determined. Choosing the m that will yield the smallest error on unseen data, from the test set, is a difficult task. Using too few RBFs results in a NN that is too flexible and will perform poorly on a new set of data. Conversely, using too many RBFs results in a NN model which is inflexible and only produces accurate predictions if the test pattern is an exact copy of a training pattern. This situation is described by the trade-off between the bias and variance of the model, see Geman et al. (1992) for a discussion.

The purpose of the algorithm that follows is to select RBFs one at time until the minimum of a prediction heuristic of error on new data has been found (see Orr, 1995). Specifically,

1. Start with no radial basis functions, $M = 0$
2. Make an initial guess for the value of λ
3. From the set of m RBFs add the one that minimizes (2)
4. Set: $M = M + 1$
5. Calculate how the NN consisting of M RBFs may perform on a set of new data by using leave-one-out cross validationStone (1974) to estimate the sum of squares error
6. Update the value of λ based on the *evidence procedure* of Mackay (1992)
7. Stop if the heuristic has passed its minimum
8. Repeat steps (3)-(7)

It is assumed that the centres $\{\boldsymbol{\mu}_j\}_{j=1}^m$ of Figure1 have been determined by a clustering algorithm such as the one described in Section 2.3. Using an heuristic to predict the error on a new data set ensures that all the available (training) data may be used to determine the NN's parameters. The output of the NN is now given by

$$s(\mathbf{x}_i) = \sum_{j=1}^{M} w_j h_j, \quad \begin{cases} i = 1, 2, \ldots, p \\ j = 1, 2, \ldots, M \\ h_j = \phi(\|\mathbf{x}_i - \boldsymbol{\mu}_j\|) \end{cases} . \tag{4}$$

3 Method

3.1 Form of the Data

Given a set of recorded rainfall data, flows at the outfall in the drainage system of Bradford, England were produced for each rainfall event using Hydroworks^{TM}. The results

for a single rainfall event is termed a *hydrograph*; an example hydrograph (of flows) is shown in Figure 2.

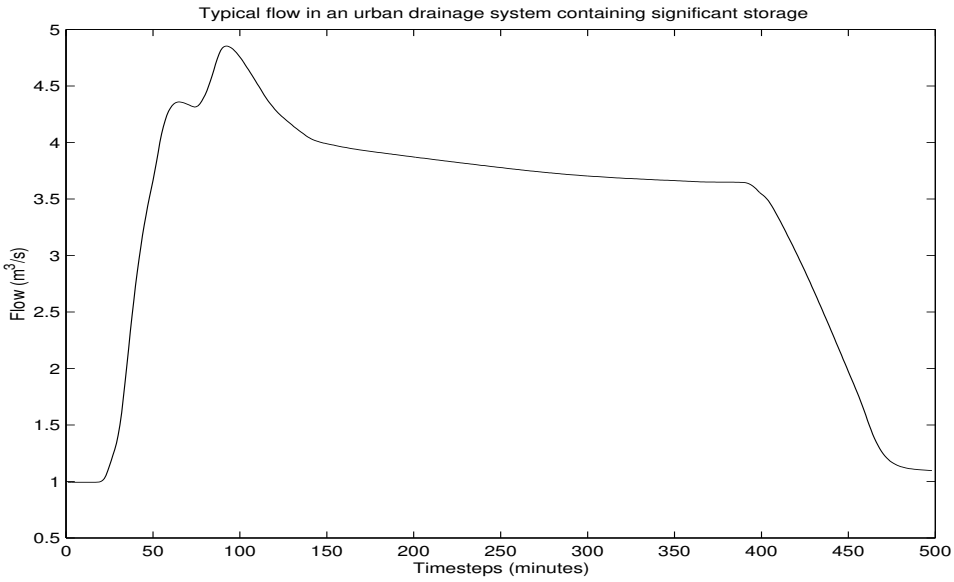

Figure 2: Figure showing the outfall hydrograph from a drainage network containing storage. The rainfall on the catchment area lasted 110 timesteps. Note that the hydrograph is superimposed on a *dry weather flow*, due to sewerage, which the Hydroworks^{TM} model assumes to be constant in this example.

The rainfalls were taken to be discrete, so that one rainfall event had one hydrograph associated with it. Furthermore, each rainfall event started under the same conditions (e.g. values of soil parameters), so that previous rainfalls had no effect on the flows generated by Hydroworks^{TM} due to the current rainfall.

Five inputs to the NN were derived from the rainfall data by taking windows of duration 7, 15, 30, 60, 120 timesteps over the rainfall event. The rainfall data was not used in its raw form because it was too erratic to produce the smooth outputs of Figure 2. Windows of small duration were used to capture the trend of a rainfall event, whilst longer windows were required to ensure the NN had rainfall-dependent inputs for some time after the rainfall ceased. Investigations have shown that this information is enough to produce satisfactory results for the modelling of the whole hydrograph apart from the recession limb.

In order to model the whole hydrograph the NN needs changing input data to correspond to changing output data. Hence, the concept of a drainage system's storage was introduced. The storage is defined by:

$$S = \text{constant.} \int_0^{t-\Delta t} I \, \mathrm{d}t - \int_0^{t-\Delta t} Q \, \mathrm{d}t, \tag{5}$$

where I is the instantaneous rainfall and Q is the measured flow. The constant in (5) is there to ensure that the storage, S, is zero after all water has flowed from the drainage network. The storage is used as a sixth input to the NN.

The rainfall data and associated hydrographs produced by Hydroworks^{TM}, were split into a training set and a test set. The training set was used to determine the free parameters of the NN. The type of non-linearity used for ϕ was the multi-quadratic function, so that:

$$h_j = \sqrt{(\mathbf{x} - \boldsymbol{\mu}_j)^2 + 1}.$$

In order to determine the NN's free parameters, the storage, S, had to be determined. This was done using the instantaneous rainfall, I, and the simulated outfall, Q, in accordance with (5). The constant in (5) was determined by equating the total rainfall to the total volume of outfall, so that the storage was zero when the flow at the outfall returned to the dry weather flow. With the storage determined for all rainfall events in the training set, inputs of the form of Figure 3 were produced by taking windows over the rainfall event. Each input variable was then scaled to have a mean of zero and unit variance. This scaled

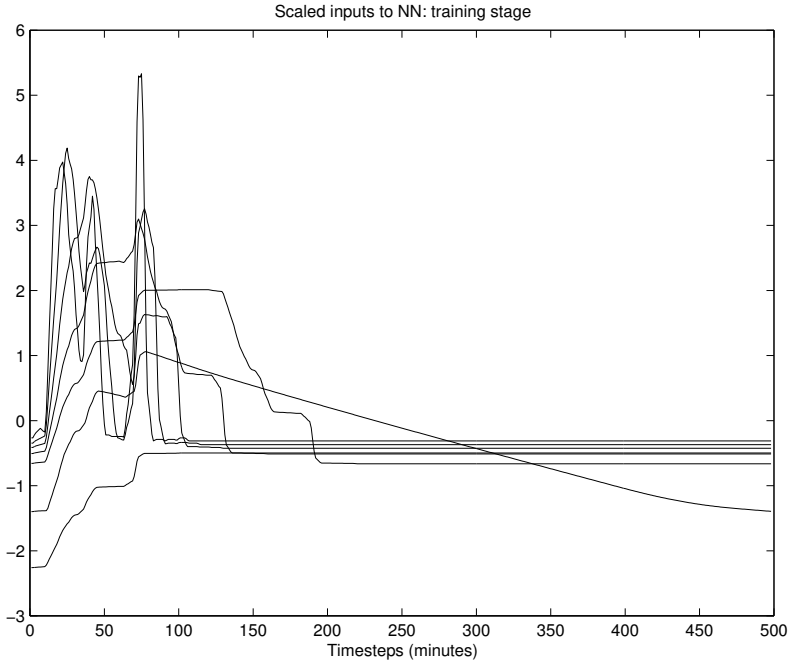

Figure 3: Figure showing an example of the type of inputs which the NN model received. The inputs were scaled to have zero mean and unit variance. The data was taken from the training stage so that the storage of the drainage network, (5), was derived from known hydrographs and not predicted values.

set of data was then clustered using the algorithm of Section 2.3, i.e. m $\boldsymbol{\mu}$'s were found. It was decided to cluster the input space with 49 codebook vectors and to use the algorithm of Section 2.5 to determine the number of RBFs in the NN model. After 14 iterations of the algorithm the error heuristic produced by LOO cross-validation had reached its minimum, the NN had been trained.

However, although the flows from the hydrographs were used in the training set, no simulated results were used in the testing stage of the experiment. This meant that the storage had to be calculated using information from the artificial neural network and that

NN predictions of flow had to be fed back to enable the calculation of storage, (5), to be performed.

4 Results and Conclusion

The training set consisted of seven rainfalls and their associated flows. The training set consisted of four rainfalls. For comparison with the NN's results, Hydroworks^{TM} computed flows for the test rainfalls also.

The mean sum-of squares error (MSSE) for the training set was 0.101 and the MSSE for the test set was 0.159. In addition, the simulated flows in the test set were produced in, on average, less than 20 seconds. This compares favourably to an average time of over 3 hours for the Hydroworks^{TM} simulator. Figure 4 shows a typical set of results.

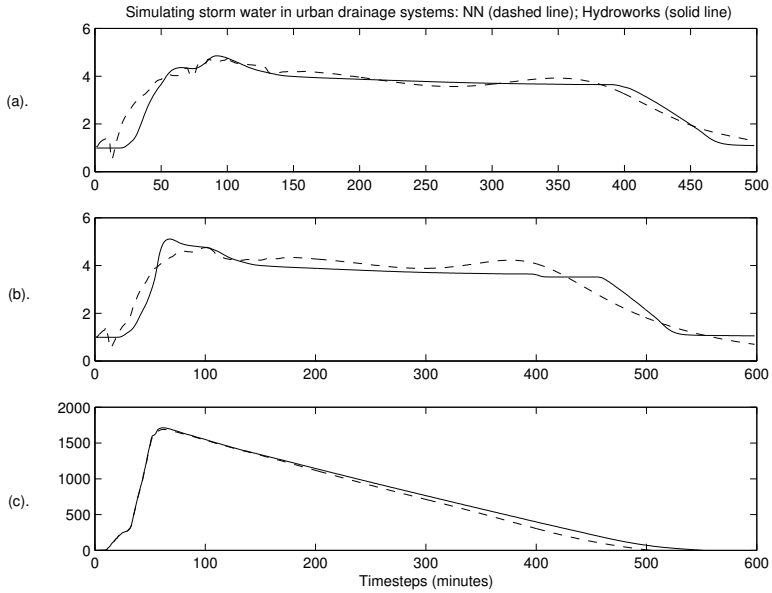

Figure 4: Example of results: Figure 4(a) shows the flows from the training set, where the storage of the drainage system was calculated with values from Hydroworks, using (5). Figure 4(b) shows the flows from the test set, where the storage of the drainage system was calculated with values of flows predicted by the NN. Figure 4(c) compares the storage, as calculated using NN predictions, with the storage found using Hydroworks^{TM} flows. This predicted storage was used to predict the hydrograph of Figure 4(b).

Given that this method is, on average, 500 times faster than the traditional method, its use is best suited to the area of on-line control. For example, given a storm rainfall, the flows up-stream of an hydraulic control (e.g. sluice gates, pumping stations, sewerage treatment works) may be quickly simulated using the model proposed in this paper. This allows engineers in the control stations to make real-time decisions based on the volume of storm water that they may expect.

It is concluded that the use of RBFs in an NN architecture improves greatly the speed with which simulated flows are generated. Furthermore, the speed up is achieved without

a significant loss of accuracy provided that the storage of the drainage system is included as an input variable and that this application has useful applications in the field of hydraulics.

References

D.S. Broomhead and D. Lowe. Multivariable function interpolation and adaptive networks. *Complex Systems*, pages 321–355, 1988.

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

R.B.B. Kellagher. The wallingford proceedure. Technical report, Wallingford Software, Howbery Park, Wallingford, Oxon., OX10 8BA, England, 1994.

T. Kohonen. *Self-Organisation and Associative Memory*. Springer-Verlag, third edition, 1989.

W. Light. *Some aspects of Radial Basis Function Approximation*, volume 256, pages 163–190. NATO ISI Series, 1992.

D.J.C. Mackay. The evidence framework applied to classification networks. *Neural Computation*, 4:720–736, 1992.

A.W. Minns and M.J. Hall. Artificial neural networks as rainfall run-off models. *Hydrological Sciences Journal*, 41 (3), 1996.

M.J.L. Orr. Regularisation in the selection of radial basis function centres. *Neural Computation*, 7 (3):606–623, 1995.

M.J.D. Powell. Radial basis functions for multivariable interpolation. In M.G. Cox and J.C. Mason, editors, *Algorithms fo Approximation*, 1985.

M. Stone. Cross-validatory choice and assesment of statistical predictions. *Journal of the Royal Statistical Society, B*, 36:111–147, 1974.

Wallingford Software. Hydroworks version 2.0. Technical report, Hydraulics Research Wallingford, Howbery Park, Wallingford, Oxon., OX10 8BA, England, 1994.

Main author's address:

S. J. Langdell
Department of Mathematics and Statistics
University of Huddersfield
Queensgate
Huddersfield, HD1 3DH
England

Tel. +44 (0)1484 472899
E-mail: s.j.langdell@hud.ac.uk