PRODUCT UNIT NEURAL NETWORKS FOR ESTIMATIONS OF LONGITUDINAL DISPERSION COEFFICIENTS IN RIVERS

Jaroslaw J. Napiorkowski[1], Adam P. Piotrowski[1], Pawel M. Rowinski[1], Steve G. Wallis[2]

[1]Institute of Geophysics, Polish Academy of Sciences, Poland, Ksiecia Janusza 64 st. 01-452 Warsaw
[2]School of the Built Environment, Heriot-Watt University, UK, Riccarton, Edinburgh, EH13 9QJ
E-mail: j.napiorkowski@igf.edu.pl

Abstract

The problem of estimating longitudinal dispersion coefficients in rivers, although studied for decades, is still a difficult task. A number of empirical equations have been proposed, many of them in a multiple power law regression form. Also, during the last ten years a number of data-driven techniques have been suggested to improve the results, including a few types of neural networks. However, Product-Unit neural networks (PUNNs), which should be well suited for dispersion prediction, have never been used for this task. Hence, in this paper PUNNs are applied to estimate longitudinal dispersion coefficients in rivers. As identifying the global optimum of PUNNs is much more difficult than for classical Multi-Layer Perceptron neural networks (MLPs), two different global optimization training algorithms are compared. In order to avoid the problem of overfitting of neural networks to training data the popular noise injection method is used. Based on 50 training runs, average objective function values from the PUNNs are generally not as good as those from the MLPs. However, if the best of the 50 runs are considered, the PUNNs allow for slightly better objective function values than MLPs. In general, noise injection makes a significant improvement to DEGL trained MLPs, but it appears to be much less beneficial for PUNNs.

Introduction

The simplest and most popular method of pollutant transport modeling is the one-dimensional advection-dispersion equation in the form of:

$$\frac{\partial C}{\partial t} + U\frac{\partial C}{\partial x} = \frac{1}{A}\frac{\partial}{\partial x}\left(AD\frac{\partial C}{\partial x}\right) \tag{1}$$

where C is the averaged pollutant concentration within the cross-section of area A, x is the longitudinal axis, t is the time, U is the cross-sectionally averaged velocity and D is the longitudinal dispersion coefficient. The most difficult parameter to obtain, even in such a simple model, is the longitudinal dispersion coefficient, which depends on the river reach and the flow conditions.

Although extensively studied during recent decades, the estimation of longitudinal dispersion coefficients in rivers still poses a practical difficulty for both scientists and engineers. To avoid the requirements of expensive and time consuming tracer tests, a number of empirical formulae have been proposed for predicting the longitudinal dispersion coefficient in rivers based on a few morphological and hydraulic characteristics, which are considered as relatively easy to obtain. The most widely used data-based methods applied in the field include empirical equations in power law regression-like form (Fischer, 1979), neural networks (Kashefipour et al. 2002) or genetic programming (Azamathulla and Ghani, 2011). The most widely used are power law regression-like equations (Wallis and Manson, 2004), for example proposed by Fischer (1979)

$$\frac{D}{HU*} = 0.011\left(\frac{U}{U*}\right)^2\left(\frac{W}{H}\right)^2 \tag{2}$$

Liu (1977)

$$\frac{D}{HU*} = 0.18\left(\frac{U}{U*}\right)^{0.5}\left(\frac{W}{H}\right)^2 \tag{3}$$

Seo and Cheong (1998)

$$\frac{D}{HU*} = 5.915\left(\frac{U}{U*}\right)^{1.428}\left(\frac{W}{H}\right)^{0.62} \tag{4}$$

and Deng et al (2001)

$$\frac{D}{HU*} = \frac{0.15}{8k_t}\left(\frac{U}{U*}\right)^2\left(\frac{W}{H}\right)^{5/3}$$
$$k_t = 0.145 + \frac{1}{3520}\left(\frac{U}{U*}\right)\left(\frac{W}{H}\right)^{1.38} \tag{5}$$

which are given in dimensionless form. In the above expressions U* is the shear velocity, W is the river width and H is the river depth, averaged over the cross-section.

Since Kashefipour et al.'s (2002) paper, a number of researchers have reported the possibility of estimating longitudinal dispersion coefficients by means of artificial neural networks (ANNs) with reasonable success

(Rowinski et al., 2005; Toprak and Cigizoglu, 2008; Noori et al., 2011). In most cases only the well known Multi-Layer Perceptron ANNs (MLP) were used (but there are exceptions – see Piotrowski et al., 2006 and Toprak and Cigizoglu, 2008). An obstacle in using other ANN types is the number of available data, which is very limited in the current subject. Unfortunately, most other ANN types require more parameters to be optimized to achieve comparable performance, and hence larger data samples are required.

However, a type of ANN exists that requires fewer optimized parameters compared to MLP models and which shows much similarity with power law regression-like equations, so frequently used in the literature. This method, called the Product-Unit neural network (PUNN), although proposed by Durbin and Rumelhart in 1989 and despite showing so much structural similarities to widely applied equations was somehow ignored by the hydrological and hydraulic community. In the present paper we explore its ability to predict longitudinal dispersion coefficients based on a well studied database (Deng et al. 2001), which comprises 81 tracer experiments performed in USA, Moldova and Poland (see Godfrey and Frederick, 1970; Nordin and Sabol, 1974; Sukhodolov et al., 1997; Deng et al., 2002).

Data

The longitudinal dispersion coefficient is estimated with the same independent variables as in the case of power law regression-like equations (*W,H,U,U**) and river sinuosity (*sin*), as proposed in Rowinski et al (2005). To allow a fair comparison with previously published results obtained by means of MLP neural networks and Deng et al.'s (2001) regression equation, in the present paper the same data, which includes 81 experiments, with identical division into training (50 cases) and testing (31 cases) data sets as in Piotrowski et al. (2012) are used. For details on the data division and the original data, consult Deng et al. (2002) and Piotrowski et al. (2012).

Product-Unit Neural Networks

MLP neural networks (probably the most popular type) have summation units in hidden and output layers. In practice, three layers are required to obtain good predictive performance (see Figure 1). Such an MLP may be defined as follows

$$y^P = v_0 + \sum_{j=1}^{J} v_j f\left(w_{j0} + \sum_{k=1}^{K} w_{jk} x_k \right) \qquad (6)$$

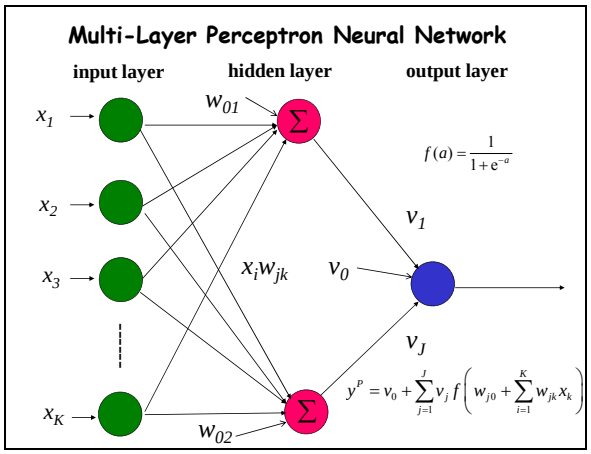

Figure 1: Scheme of multi-layer perceptron neural network.

Where y^P is the output, J is the number of hidden units, K is number of inputs x, w and v are ANN weights (parameters which need to be optimized) and $f(c)$ is a so-called activation function, which filters the results of nodal calculations before passing them to the next layer. Thus the argument, c, is the weighted sum of inputs to a node. Frequently and herein the logistic form of the activation function is used (Haykin, 1999)

$$f(c) = \frac{1}{1+e^{-c}} \qquad (7)$$

Such an MLP neural network requires optimization of $K*J + 2*J + 1$ parameters, if we assume that there is only one output from the network. Usually, MLP inputs and outputs are normalized to [0,1] interval (see Zhang et al., 1998).

Durbin and Rumelhart (1989) proposed a Product-Unit neural network, which in its simplest form (see also Martinez-Estudillo et al., 2006) may be defined as

$$y^p = v_0 + \sum_{j=1}^{J} v_j \prod_{k=1}^{K} x_k^{w_{jk}} \qquad (8)$$

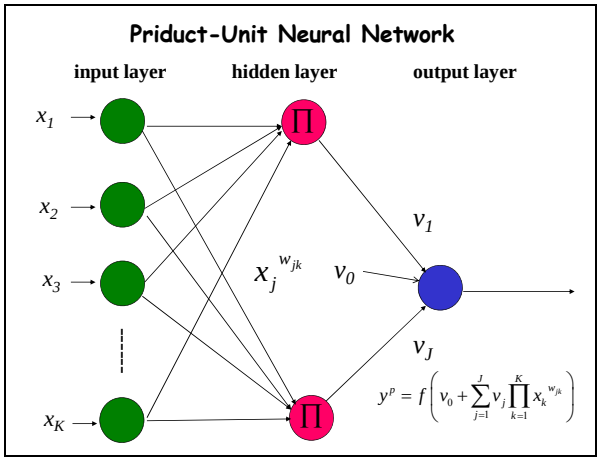

Figure 2: Scheme of product-unit neural network.

Such an ANN requires optimization of only of *K*J + J + 1* parameters. Moreover, according to Leerink et al. (1995), the information capacity of a PUNN increases to 3*K* (compared to 2*K* for a MLP), hence, at least theoretically, fewer hidden units are required to approximate a particular function with the same precision as by means of a MLP. Of course, input and output normalization range must exclude 0 (because of the product formulation). According to Martinez–Estudillo (2006), the [0.1,0.9] range is a reasonable choice. This is especially important as negative inputs, if present, would lead to complex numbers if raised to non-integer powers.

According to Janson and Frenzel (1993), PUNNs are difficult to train. This is a severe problem as the PUNN weights are unbounded, as are MLP ones. However, if the parameter space is reduced, for example by requiring that $w,v \in$ [-2,2], which we suggest in the present paper as a good choice at least for longitudinal dispersion estimation, the difficulty with PUNN training becomes mitigated, and the risk of overfitting is reduced (see Haykin (1999) and Liu et al. (2008) for details on overfitting in ANN training).

Following Kashefipour et al. (2002) and Piotrowski et al. (2012) we used

$$J\ (\mathbf{w},\mathbf{v}) = \min_{w,v} \frac{1}{N}\sum_{n=1}^{N}\left|y_n^P(\mathbf{w},\mathbf{v}) - \ln D_n\right| \qquad (9)$$

as the objective function for ANN training. Using logarithms of *D* avoids the situation where the model would fit to the large longitudinal dispersion coefficients only.

In Piotrowski et al. (2012) a number of Evolutionary Computation methods were used for MLP training with noise injection as a method to avoid ANN overfitting. Due to the non-differentiable nature of the above objective function two training algorithms of different nature, that do not require gradient computation are used for PUNN training here, namely: Differential Evolution-based (Storn and Price, 1995) DEGL algorithm (Das et al., 2009), which ranked among the best methods in ANN training in Piotrowski et al (2012), and the recently developed simplex-based (Nelder and Mead, 1965) SP-UCI algorithm (Chu et al., 2011).

ANNs require some method to avoid overfitting (Haykin, 1999). The most popular one is the so-called early stopping (Prechlet, 1998), which requires dividing the available data into 3 sets – training, validation and testing (the only independent one). However, this poses a problem when only a small number of data are available. Another well known option is using noise injection. In this case dividing the data into only two sets is required.

Noise injection to data during neural network training was experimentally found to improve their generalization capability (Holmstrom and Koistinnen, 1992), especially in the case of limited data (Hua et al. 2006). The similarities between noise injection and other methods designed to improve the generalization properties of neural networks, including weight decay, have been studied (Reed et al. 1995). Although different variants of noise injection may be found in the literature, in the present paper we use the same approach as in Piotrowski et al. (2012), which proved to have a good performance for longitudinal dispersion coefficient estimation by means of MLP neural networks.

In this, an *m*-th artificial case $(m \in [1, M])$ is generated for each *i*-th experiment $(i \in [1, 81])$ for 5 input and one output variables. Following Piotrowski et al. (2012) we set *M* to 100. For four of the input variables x_k, i.e. *W*, *H*, *U*, *sin*, and for the output an additional case is generated by adding Gaussian noise $\varepsilon = N(0,\sigma)$, specific to the particular variable, and to the measured value, e.g.

$$\begin{aligned} x_{ki}^m &= x_{ki} + \varepsilon_{ki}^m \\ D_i^m &= D_i + \varepsilon_{Di}^m \end{aligned} \qquad (10)$$

The uncertainty of *D*, *W*, *H*, *U* and *sin* depend mostly on the quality of on-site measurements, performed by different researchers in various conditions. Hence, the noise variance of these variables is defined as $\sigma_{ki} = 0.1\ x_{ki}$. The bulk shear velocity U^* is treated a little differently. Under steady and uniform flow conditions it may be approximated by

$$U^* = \sqrt{gHs} \qquad (11)$$

where *s* is the bed slope and *g* is acceleration due to Earth's gravity. We assume that Gaussian noise $N(0,\sigma_{si})$, with variance $\sigma_{si} = 0.1\ s_i$ is added to the bed slope $s_i^m = s_i + \varepsilon_i^m$ and then U^* is computed according to:

$$U^{*m}_{\ i} = \sqrt{gH_i^m s_i^m} \qquad (12)$$

Hence the total number of data include 101*50 training and 101*31 testing cases. Following Piotrowski et al. (2012), the case with only noise-free data was also considered.

Results and Discussion

In Piotrowski et al (2012) MLPs with noise injection were successfully used for dispersion estimation in rivers. A number of versions were studied – with 2 and 3 hidden nodes, with box constraints of [-1000,1000] and [-10,10], trained without noise injection and with noise injection, and using a number of Evolutionary Computation algorithms. Each MLP version was optimized 50 times. Due to the small number of testing data (31 experiments) the results were compared on noisy data (to be specific, 3100 noisy and 31 original experiments). It was shown that the MLP with 3 hidden nodes, box constraints set to [-1000, 1000]

and noise injection during training obtained the best results according to both the average and the single best objective function value obtained from the 50 optimizations.

In the present study, a PUNN with 3 hidden nodes was used for the same task. Two training algorithms, DEGL and SP-UCI, were used: each PUNN was optimized 50 times. Some objective function values evaluated using (9) from these two training methods, with and without noise injection, and the earlier MLP are presented in Table 1. Similarly to the previous study, the results are compared to noisy data.

Table 1 shows that the 50-optimization averaged objective function results obtained from the DEGL trained PUNN were much poorer than those obtained from the MLP. This is true for both the training data set and the testing data set. Moreover, in the case of noise free training the PUNN averaged objective function values show very large values, which are due to an occasional very poor prediction. More stable results were obtained when noise was injected during training, however they were still worse than those obtained by the MLP (again for both data sets).

However, a different view is possible if one concentrates on the single best objective function value obtained from the 50 optimizations using the DEGL algorithm (see Table 2). Here the lowest objective function from the testing data set was found for the PUNN trained without noise injection (value of 0.54) (although the statistic is derived from noisy data, as defined earlier). This suggests that the PUNN is difficult to train and in some optimization runs DEGL is unable to achieve a reasonable performance, leading to a large 50-optimization averaged value of objective function. However, some successful training does occur and in such cases the derived values of longitudinal dispersion coefficients are of very good quality, being slightly better than those from the MLP model.

Table 1: Averaged objective function values obtained for neural networks with 3-hidden nodes: NF – noise free; NI – noise injection.

Algorithm - data set - noise	MLP	PUNN
DEGL - training - NF	0.43	997.0
SP-UCI - training - NF	-	0.56
DEGL - testing - NF	0.65	77.6
SP-UCI - testing - NF	-	0.89
DEGL - training - NI	0.40	0.57
SP-UCI - training - NI	-	0.65
DEGL - testing - NI	0.62	0.89
SP-UCI - testing - NI	-	0.87

SP-UCI trained PUNNs didn't lead to any very poor 50-optimisation averaged results (see Table 1), however it also didn't beat the corresponding objective function values obtained from the MLP. Note that the single best objective function values obtained by SP-UCI (see Table 2) were always higher than the MLP values. Hence it may be concluded that SP-UCI trained PUNNs give more stable results than DEGL trained PUNNs, but the former are unable to reach occasionally as good solution as are the latter.

Table 2: Best objective function values obtained for neural networks with 3-hidden nodes.

Algorithm - data set - noise	MLP	PUNN
DEGL - training - NF	0.35	0.39
SP-UCI - training - NF	-	0.40
DEGL - testing - NF	0.58	0.54
SP-UCI - testing - NF	-	0.65
DEGL - training - NI	0.36	0.50
SP-UCI - training - NI	-	0.39
DEGL - testing - NI	0.55	0.76
SP-UCI - testing - NI	-	0.56

The graphical interpretation of calculated longitudinal dispersion coefficients from some of the cases shown in Table 2 are depicted in Figure 3 and 4. Figure 3 presents results for the training data set. The plot shows the measured data, results generated by means of the noise-injected MLP, with optimal weighting coefficients, results generated by means of the noise-injected PUNN, with optimal weighting coefficients and results generated by means of the noise-injected Deng et al (2001) formula. Note that all points are distributed along the 50 horizontal lines corresponding to 50 river reaches and that the plots are in full agreement with Table 2. The green clouds are more compact than the black ones.

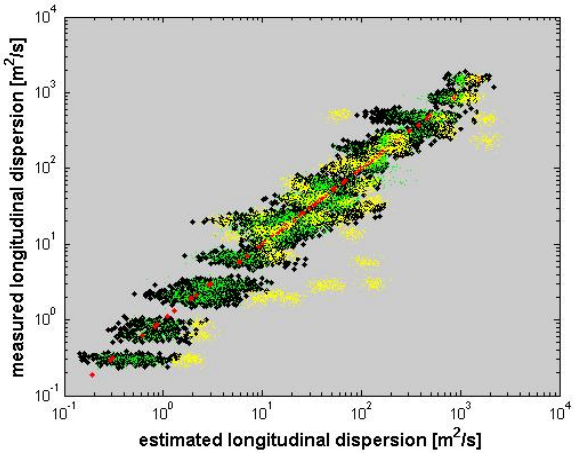

Figure 3: Comparison of longitudinal dispersion coefficients: measured (red squares), MLP (green dots), SP-UCI trained PUNN with noise injection (black squares) and Deng et al (2001) formula (yellow dots). Plot shows 100-times noise-injected and original data for 50 river reaches (training data).

Figure 4 presents a similar comparison of results for the testing data set. In this case all points are more or less scattered along the 31 horizontal lines corresponding to 31

river reaches. Again plots are in full agreement with Table 2 and the points forming the black clouds are more scattered then the green ones. The Deng et al (2001) formula in many cases produces results outside its range of applicability.

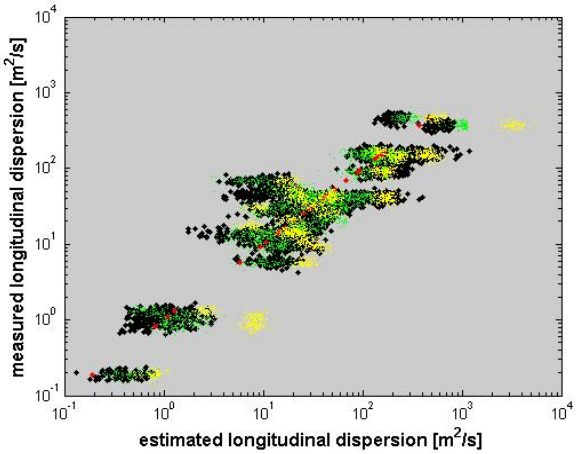

Figure 4: Comparison of longitudinal dispersion coefficients: measured (red squares), MLP (green dots), SP-UCI trained PUNN with noise injection (black squares) and Deng et al (2001) formula (yellow dots). Plot shows 100-times noise-injected and original data for 31 river reaches (testing data).

Conclusions

Comparing objective function values suggests that on average PUNNs do not perform better than MLPs for longitudinal dispersion coefficient estimation. In general, noise injection made a significant improvement to DEGL trained MLPs, but it appears to be less beneficial for PUNN training. This is probably the effect of the well known difficulty of training PUNNs. However, the single best result was obtained by means of a PUNN trained by DEGL, which shows the potential of this kind of neural network. The stochastic nature of MLPs and PUNNs means that any version of either of them has the potential for making good dispersion predictions.

References

Azamathulla, H. M. & Ghani, A. A. (2011) Genetic Programming for predicting longitudinal dispersion coefficients in streams. *Water Resources Management* 25(6), pp. 1537-1544.

Chu, W., Gao, X. & Sorooshian, S. (2011). A new evolutionary search strategy for global optimization of high-dimensional problems. *Information Sciences* 181, pp. 4909-4927.

Das, S., Abraham, A., Chakraboty, U. K. & Konar, A. (2009) Differential Evolution Using a Neighborhood-based Mutation Operator. *IEEE Transactions on Evolutionary Computation* 13(3), pp.526-553.

Deng, Z. Q., Singh, V. P. & Bengtsson, L. (2001) Longitudinal dispersion coefficient in straight rivers. *J. Hydrauic. Engng. ASCE* 127(11), 919-927.

Deng, Z. Q., Bengtsson, L. & Singh, V. P. (2002) Longitudinal dispersion coefficient in single-channel streams. *J. Hydraulic Engng. ASCE* 128(10), pp. 901–916.

Durbin, R. & Rumelhart, D. E. (1989) Product Units: a computationally powerful and biologically plausible extension to back propagation networks. *Neural Computation* 1, pp. 133-142.

Fischer, H. B., List, E. J., Koh, R. C. Y., Imberger, J., and Brooks, N. H. (1979) *Mixing in inland and coastal waters*, Academic. New York, 104–138.

Godfrey, R. G. & Frederick, B. J. (1970) Stream dispersion at selected sites. U.S.Geological Survey Professional Paper, 433–K.

Haykin, S. (1999). *Neural Networks, a Comprehensive Foundation*. Macmillan College Publishing Co., New York, USA.

Holmstrom, L. & Koistinen, P. (1992) Using additive noise in back-propagation training. *IEEE Transactions on Neural Networks* 3, pp. 24-38.

Hua, J.P., Lowey, J., Xiong, Z. & Dougherty, E. R. (2006) Noise-injected neural networks show promise for use on small-sample expression data. *BMS Bioinformatics* 7, art. no. 274.

Janson, D. J. & Frenzel, J. F. (1993) Training product unit neural networks with genetic algorithms. *IEEE Expert-Intelligent Systems & Their Applications* 8(5), pp. 26-33.

Kashefipour, S. M., Falconer, R. A. & Lin, B. (2002) *Modeling longitudinal dispersion in natural channel flows using ANNs.* In: River Flow 2002 (ed. By D. Bousmar & Y. Zech), 111–116. A.A. Balkema/Swets & Zeitlinger, Lisse, The Netherlands.

Leerink, L. R., Giles, C. L., Horne, B. G. & Jabri, M. A. (1995) Learning with product units. In : Advances in Neural Information Processing Systems 7, Cambridge, MA : MIT Press, pp. 537-544.

Liu H. (1977) Predicting dispersion coefficient of streams. *Journal of the Environmental Engineering Division*, American Society of Civil Engineers, 103, EE1, pp. 59–69.

Liu, Y., Starzyk, J. A. & Zhen, Z. (2008) Optimized approximation algorithm in neural networks without overfitting. *IEEE Transactions on Neural Networks* 19(6), pp. 983-995.

Martinez-Estudillo, A., Martinez-Estudillo, F., Hervias-Martinez, C., Garcia-Pedrajas, N. (2006) Evolutionary product unit based neural networks for regression. *Neural Networks* 19, pp. 477-486.

Nelder, J. A. & Mead, R. (1965) A simplex method for function minimization. *Computer Journal* 7, pp. 303-313.

Noori, R., Karbassi, A. R., Mehdizadeh, H., Vesali-Naseh, M.& Sabahi, M. S. (2011) A Framework Development for Predicting the Longitudinal Dispersion Coefficient in Natural Streams Using an Artificial Neural Network. *Environmental Progress & Sustainable Energy* 30(3), pp. 439-449.

Nordin, C. F. & Sabol, G. V. (1974) Empirical data on longitudinal dispersion. US Geol. Survey Water Resour. Investigations 20–74.

Piotrowski, A., Rowinski, P.M. & Napiorkowski, J.J. (2006). Assessment of longitudinal dispersion coefficient by means of different neural networks. In: 7th Int. Conf. on Hydroinformatics, HIC 2006, Nice, France, (ed by P. Gourbesville, J. Cunge, V. Guinot & S. Y. Liong), Research Publishing.

Piotrowski, A. P., Rowinski, P. M. & Napiorkowski, J. J. (2012) Comparison of evolutionary computation techniques for noise injected neural network training to estimate longitudinal dispersion coefficients in rivers. *Expert Systems with Application* 39, pp. 1354-1361.

Prechlet, L. (1998) Automatic early stopping using cross-validation: quantifying the criteria. *Neural Networks* 11(4), pp. 761-777.

Reed, R., Marks, R. J. & Oh, S. (1995) Similarities of Error Regularization, Sigmoid Gain Scaling, Target Smoothing and Training with Jitter. *IEEE Transactions on Neural Networks* 6(3), pp. 529-538.

Rowinski, P. M., Piotrowski, A. & Napiorkowski, J. J. (2005) Are artificial neural network techniques relevant for the estimation of longitudinal dispersion coefficient in rivers?. *Hydrological Sciences Journal* 50(1), pp. 175–187.

Seo I. W. & Cheong T. S. (1998) Predicting longitudinal dispersion coefficient in natural streams. *Journal of Hydraulic Engineering*, American Society of Civil Engineers, 124, No. 1, pp. 25–32.

Storn, R. & Price, K. V. (1995) Differential Evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces. Tech. Report TR-95-012, International Computer Sciences Institute, Berkeley, California, USA.

Sukhodolov, A. N., Nikora, V. I., Rowiński, P. M. & Czernuszenko, W. (1997) A case study of longitudinal dispersion in small lowland rivers. *Water Environment Research* 69(7), pp. 1246–1253.

Toprak, Z. F. & Cigizoglu, H. K. (2008) Predicting longitudinal dispersion coefficient in natural streams by artificial intelligence methods. *Hydrological Processes* 22, pp. 4106-4129.

Wallis, S. G. & Manson, J. R. (2004) Methods for predicting dispersion coefficients in rivers. *Proceedings of the Institution of Civil Engineers, Water Management* 157, pp. 131-141.

Zhang G., Patuwo, B. E., Hu, M. Y. (1998) Forecasting with artificial neural networks: the state of the art. *International Journal of Forecasting* 14, pp. 35-62.