WDN-RBF: Weighted Data Normalization for Radial Basic Function Type Neural Networks

Qun Song and Nikola Kasabov

Knowledge Engineering & Discovery Research Institute, Auckland University of Technology
Private Bag 92006, Auckland 1020, New Zealand
Email: qsong@aut.ac.nz, nkasabov@aut.ac.nz

Abstract – **This paper introduces an approach of Weighted Data Normalization (WDN) for Radial Basis Function (RBF) type of neural networks. It presents also applications for medical decision support systems. The WDN method optimizes the data normalization ranges for the input variables of the neural network. A steepest descent algorithm (BP) is used for the WDN-RBF learning. The derived weights have the meaning of feature importance and can be used to select a minimum set of variables (features) that can optimize the performance of the RBF network model. The WDN-RBF is illustrated on two case study prediction/identification problems. The first one is prediction of the Mackey-Glass time series and the second one is a real medical decision support problem of estimating the level of renal functions in patients. The method can be applied to other distance-based, prototype learning neural network models.**

Keywords: data normalisation; RBF; feature selection; time series prediction; renal function evaluation.

I. Introduction

In many neural network models and applications, raw (not normalized) data is used. This is appropriate when all the input variables are measured in the same units. Normalization, or standardization, is reasonable when the variables are in different units, or when the variance between them is substantial. However, a general normalization means that every variable is normalized in the same range, e.g. [0, 1] with the assumption that they all have same importance for the output of the system.

For many practical problems, variables have different importance and make different contribution to the output. Therefore, it is necessary to find an optimal normalization and assign proper importance factors to the variables. Such a method can also be used for feature selection, or for reducing the size of the input vectors through keeping the most important ones. This is especially applicable to a special class of neural networks (NN) – the clustering based NN (or also: distance-based; prototype-based) such as RBF [1, 2, 3, 4] and ECOS [5, 6]. In such systems, distance between neurons or nodes and input vectors are usually measured as *Euclidean distance*, so that variables with a wider range will have more influence on the learning process and vice versa.

The paper is organized as follows: Section II reviews the previous work about weighted data normalization. Section III presents the structure and algorithm of WDN-RBF NN. Sections IV and V illustrate the approach on two case study problems. Conclusions are drawn in section VI.

II. Review Of METHODS FOR Weighted Data Normalization

A. Using Correlation Coefficients as Weights (Importance) for Input Variables

A correlation coefficient indicates the strength of the association between any two metric variables. Its value is from –1 to +1: the sign (+ or -) indicates the direction of the relationship. The value can range from -1 to +1, with +1 indicating a perfect positive relationship, 0 indicating no relationship, and – 1 indicating a perfect negative or reverse relationship (as one variable grows larger, the other variable grows smaller) [7].

Some researches have used correlation coefficients to obtain the importance of each variable for feature selection, so that the more important variables can be selected [8]. However, the association between two independent variables that a correlation coefficient represents is a linear relationship and it is difficult to use correlation coefficients for weighting nonlinear data, as in most cases there is only a weak linear relationship between two variables or the variables are not independent. In Section IV and V, two experimental results with the use of the correlation coefficient method are shown.

B. GA for Weighted Data Normalization and Feature Selection in ECOS Systems

Evolving connectionist systems (ECOS) are multi-modular, connectionist architectures that facilitate modelling of evolving processes and knowledge discovery. An ECOS may consist of many evolving connectionist modules.

An ECOS is a NN that operates continuously in time and adapts its structure and functionality through a continuous interaction with the environment and with other systems according to: (i) a set of parameters that are subject to change during the system operation; (ii) an incoming continuous flow of information with unknown distribution; (iii) a goal (rationale) criteria (also subject to modification) that is applied to optimise the performance of the system over time.

One of the ECOS models is EFuNN [6] (evolving fuzzy neural network) that consists of five layers: input nodes, representing input variables; input fuzzy membership nodes, representing the membership degrees of the input values to each of the defined membership functions; rule nodes, representing cluster centers of samples in the problem space and their associated output function; output fuzzy membership nodes, representing the membership degrees to which the output values belong to defined membership functions; and output nodes, representing output variables. The EFuNN is usually used for solving on-line prediction problems.

The ECMC [9] (evolving clustering method for classification) is another ECOS models, which classifies a data set into a number of classes in the *n*-dimensional input space by evolving rule nodes. Each rule node is associated with a class through a constant. Its receptive field covers a part of the *n*-dimensional space around the rule node. Usually, such an influence field in the *n*-dimensional space is a hyper-sphere and a number of rule nodes may be associated to a same class.

A WDN method for ECOS is proposed in [9] that employs *genetic algorithms* (GA). A GA algorithm applies a multi-point, probabilistic search in the whole search space of all variable normalization ranges to discover a global optimum of normalization weights based on an objective (fitness) function that is the least classification (prediction) error. Using the GA-based WDN, the EFuNN obtained 11 % more accurate prediction than the EFuNN without WDN for the bench mark problem of Mackey Galss (used in this paper too), and the ECMC obtained 23% better results than the ECMC without WDN for a benchmark classification problem. Here, another algorithm for WDN based on gradient descent and particularly targeting the RBF class of NN is introduced.

III. Structure And Learning Algorithm Of WDN-RBF

The origin of the radial basis function (RBF) models is in the approximation theory and in the regularization theory [1]. They deal with the general approximation problem common to supervised learning neural networks. Standard RBF networks have three layers of neurons: input layer - that represents input variables; hidden layer – representing centers of clusters of data in the input space; and output layer – that represents the output variables. Although RBF networks usually need more neurons in their hidden layer than standard feed-forward back-propagation networks, such as MLP, they train much faster than MLP networks and can be used to extract meaningful rules based on clusters of data. RBF model work very well when there is enough training data.

A WDN-RBF network has one more layer – the WDN layer for weighted data normalization. *Gaussian* functions are used as activation functions for each neuron in the hidden layer. The WDN-RBF structure is shown in Fig.1.

Consider the system having *P* inputs, one output, and *M* neurons in the hidden layer, the output value of the system can be calculated on input vector $\boldsymbol{x}_i = [x_1, x_2, \ldots, x_P]$ as follows:

$$y = f(\boldsymbol{x}_i) = b_0 + b_1 R_1(\boldsymbol{x}_i) + b_2 R_2(\boldsymbol{x}_i), \ldots, + b_M R_M(\boldsymbol{x}_i) \qquad (1)$$

here, $$R_l(\boldsymbol{x}_i) = \prod_{j=1}^{P} \exp\left[-\frac{(w_j x_{ij} - m_{lj})^2}{2\sigma_{lj}^2}\right] \qquad (2)$$

m_{lj} and σ_{lj} are parameters of *Gaussian* functions, w_j are weights of input variables.

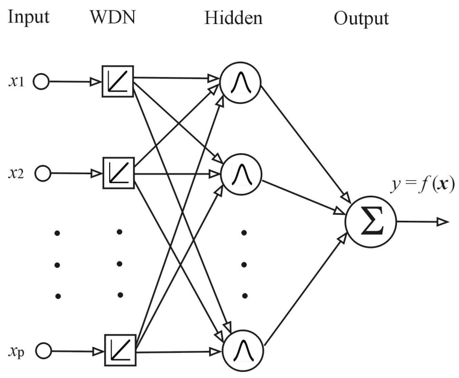

Fig.1. The Structure of the proposed WDN-RBF Network

In the WDN-RBF learning algorithm, the following indexes are used:

• Training Data pairs:	$i = 1, 2, \ldots, N$;
• Input Variables:	$j = 1, 2, \ldots, P$;
• Neurons in the hidden layer:	$l = 1, 2, \ldots, M$;
• Learning Iterations:	$k = 1, 2, \ldots$

The WDN-RBF learning algorithm minimizes the following objective function (an error function):

$$E = \frac{1}{2}\sum_{i=1}^{N} [\, f(\boldsymbol{x}_i) - d_i]^2 \qquad (3)$$

The steepest descent algorithm (back-propagation algorithm) is used on the training data pairs, $[\boldsymbol{x}_i, d_i]$, to obtain the recursions for updating the parameters $\boldsymbol{b}$, $\boldsymbol{m}$, $\boldsymbol{\sigma}$ and $\boldsymbol{w}$ such that *E* of Eq. 3 is minimized:

$$b_0(k+1) = b_0(k) - \eta_b \sum_{i=1}^{N} [\, f(\boldsymbol{x}_i) - d_i] \qquad (4)$$

$$b_l(k+1) = b_l(k) - \eta_b \sum_{i=1}^{N} \{R_l(\boldsymbol{x}_i)[f(\boldsymbol{x}_i) - d_i]\} \quad (5)$$

$$m_{lj}(k+1) = m_{lj}(k) - \eta_m \sum_{i=1}^{N} \left\{ R_l(\boldsymbol{x}_i) b_l [f(\boldsymbol{x}_i) - d_i] \frac{(w_j x_{ij} - m_{lj})}{\sigma_{lj}^2} \right\} \quad (6)$$

$$\sigma_{lj}(k+1) = \sigma_{lj}(k) - \eta_\sigma \sum_{i=1}^{N} \left\{ R_l(\boldsymbol{x}_i) b_l [f(\boldsymbol{x}_i) - d_i] \frac{(w_j x_{ij} - m_{lj})^2}{2\sigma_{lj}^3} \right\} \quad (7)$$

$$w_j(k+1) = w_j(k) - \eta_w \sum_{l=1}^{M} \sum_{i=1}^{N} \left\{ R_l(\boldsymbol{x}_i) b_l [f(\boldsymbol{x}_i) - d_i] \frac{x_{ij}(m_{lj} - w_j x_{ij})}{\sigma_{lj}^2} \right\} \quad (8)$$

here, η_b, η_m, η_σ and η_w are learning rates for updating the parameters $\boldsymbol{b}$, $\boldsymbol{m}$, $\boldsymbol{\sigma}$ and $\boldsymbol{w}$ respectively.

In the approach, a clustering method, ECM (evolving clustering method) [10], is used for clustering the training data to obtain the initial values of the network. Other clustering methods can also be used.

IV. Case Study Example Of Applying The WDN-RBF For A Time-series Prediction

In the present paper, the WDN-RBF network is applied to the time series prediction. The WDN-RBF learning is demonstrated on the Mackey-Glass (MG) time series prediction task [11]. The MG time series is generated with a time-delay differential equation as follows:

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1 + x^{10}(t-\tau)} \quad (9)$$

To obtain this time series values at integer points, the *fourth-order Runge-Kutta method* was used to find the numerical solution to the above MG equation. Here we use the following parameter values: a time step of 0.1, $x(0) = 1.2$, $\tau = 17$ and $x(t) = 0$ for $t < 0$. From the input vector $[x(t-18)\ x(t-12)\ x(t-6)\ x(t)]$, the task is to predict the value $x(t+6)$. In the experiments, 1000 data points, from t = 118 to 1117, were extracted for predicting the 6 steps ahead output value. The first half of the data set was taken as a training data, and another half as the testing data.

For comparison, the WDN-RBF implements three times with different data weights on the same data and the results, including the values of weights, training RMSE and testing RMSE, are shown in TABLE I.

- Experiment 1: Set all initial values of weights to '1'; the learning rate for the weights is set to '0'. It is the RBF learning without weighted data normalization.
- Experiment 2: Take the correlation coefficients between each input variable and output variable as the initial values of the weights and set the learning rate for weights to '0'.
- Experiment 3: Set the initial weights to 0.9 for each input variable and set the learning rate for weights to '0.002'.

TABLE I EXPERMENTAL RESULTS ON MG DATA

Experiment	Weights of the input variables x1,x2,x3,x4	Training RMSE	Testing RMSE
(1): 30 neurons	1, 1, 1, 1 (no WDN)	0.0064	0.0065
(2): 31 neurons (using correlation coefficients as weights in the RBF)	0.80, 0.57, 0.01, 0.67 (with WDN)	0.0072	0.0075
(3): 29 neurons (WDN-RBF)	1, 0.97, 0.62, 0,79 (with WDN)	0.0052	0.0052

With the use of the proposed method, the prediction results are significantly better then the results obtained with the same network without WDN as the weights represent the different contribution of each input to the output. When correlation coefficients are taken as the weights the accuracy is significantly less. This is because the relationship between the input variables and output variable is non-linear and the correlation coefficients can not indicate such relationship correctly.

V. Case Study Example Of Applying The WDN-RBF For A Medical Decision Support Problem

A real data set from a medical institution is used here for experimental analysis. The data set has 447 samples, collected at hospitals in New Zealand and Australia. Each of the records includes six variables (inputs): age, sex, gender, serum creatinine, serum albumin, and blood urea nitrogen concentrations, and one output - the Glomerular Filtration Rate (GFR) value. All experimental results reported here are based on 10-cross validation experiments with the same model and parameters and the results are averaged. In each experiment 70% of the whole data set is randomly selected as training data and another 30% as testing data.

For comparison, several well-known methods are applied on the same problem, such as the MDRD function [12], MLP neural network [13], adaptive neural fuzzy inference system (ANFIS) [14] and dynamic evolving neural fuzzy inference system (DENFIS) [15] along with the proposed WDN-RBF and results are given in TABLE II. The results include the number of fuzzy rules (for ANFIS and DENFIS), or neurons in the hidden layer (for RBF and MLP), the weights for input variables (for WDN-RBF), the testing RMSE (root mean square error), and the testing MAE (mean absolute error).

Four experiments with RBF were conducted. The first three are similar to the experiments 1, 2 and 3 as explained in the previous section. In the last experiment, only the top 4 variables out of 6 variables are used, as ranked in the previous experiment with the WDN-RBF method. The last experiment gives the result very close to the best one. This illustrates that the WDN –RBF model can be used to select the most important variables and to simplify the system.

TABLE II EXPERIMENTAL RESULTS WITH DIFFERENT MOFELS ON GFR DATA

Model	Weights for the Six Input Variables	RMSE	MAE
MDRD	No WDN	7.74	5.88
MLP 12 neurons	No WDN	8.44	5.75
ANFIS 36 rules	No WDN	7.49	5.48
DENFIS 27 rules	No WDN	7.29	5.29
Standard RBF 30 neurons	1, 1, 1, 1, 1, 1 (No WDN)	7.38	5.41
WDN-RBF 30 neurons (using correlation coefficients as weights in the RBF)	0.05, 0.12, 0.61, 0.5, 0.11, 0.14 (with WDN)	7.33	5.35
WDN-RBF 3 30neurons 6 variables (WDN-RBF)	0.79, 0.55, 1, 0.88, 0.23, 0.12	6.99	5.23
WDN-RBF 4 30 neurons 4 variables (WDN-RBF)	0.85, 0.42, 1, 0.94	7.2	5.25

VI. Conclusions

The proposed approach, weighted data normalization (WDN) for RBF network is a generic algorithm and can be applied successfully to other RBF type NN across all applications. The WDN-RBF is also efficient when used for feature selection.

Future development of the method includes using *non-Euclidean* distance and transductive WDN-RBF models for solving both prediction and classification problems.

Acknowledgements

The research presented in the paper is funded by the New Zealand Foundation for Research, Science and Technology under grant NERF/AUTX02-01.

Refreences

[1] Y. Lee, “Handwritten digit recognition using K nearest-neighbor, radial basis function, and back-propagation neural networks,” *Neural Computation*, vol.3 No.3, pp. 440 – 449, 1991.

[2] F. Poggio, “Regularization theory, radial basis functions and networks” In: *From Statistics to Neural Networks: Theory and Pattern Recognition Applications*. NATO ASI Series, No.136, pp. 83 – 104, 1994.

[3] T. Poggio and F. Girosi “Network for approximation and learning” In: *Proceedings of IEEE, 78(9) pp. 1481 – 1497, September, 1990.*

[4] B. Hammer and T. Villmann, “Generalized relevance learning vector quantization” *Neural Networks 15, pp. 1059 – 1068, 2002.*

[5] N. Kasabov, Evolving connectionist systems: Methods and Applications in Bioinformatics, Brain study and intelligent machines, Springer Verlag, London, New York, Heidelberg, 2002.

[6] N. Kasabov, “Evolving fuzzy neural networks for on-line supervised/unsupervised, knowledge–based learning,” *IEEE Trans. SMC* – part B, Cybernetics, vol.31, No.6, 902-918, December 2001.

[7] H. Anderson and T. Black, Multivariate Data Analysis, Prentice-Hall, London, 1998.

[8] L. Goh, Q. Song and N. Kasabov, “A Novel Feature Selection Method to Improve Classification of Gene Expression Data,” *Proceedings of Second Asia-Pacific Bioinformations Conference* (APBC2004), vol. 27, No.4, pp. 161 – 166, Dunedin, New Zealand, January, 2004.

[9] Q. Song and N. Kasabov, "Weighted Data Normalizations and feature Selection for Evolving Connectionist Systems Proceedings”, *Proc. of The Eighth Australian and New Zealand Intelligence Information Systems Conference* (ANZIIS2003), pp. 285 – 290, Sydney, Australia, December, 2003.

[10] Q. Song and N. Kasabov, "ECM - A Novel On-line, Evolving Clustering Method and Its Applications", *Proceedings of the Fifth Biannual Conference on Artificial Neural Networks and Expert Systems (ANNES2001)*, pp. 87 – 92, Dunedin, New Zealand, November, 2001.

[11] M. C. Mackey, and L. Glass, “Oscillation and Chaos in Physiological Control systems”, *Science*, vol. 197, pp. 287 – 289, 1977.

[12] A. S. Levey, J. P. Bosch, J. B. Lewis, T. Greene, N. Rogers, D. Roth, for the Modification of Diet in Renal Disease Study Group, “ A More Accurate Method To Estimate Glomerular Filtration Rate from Serum Creatinine: A New Prediction Equation”, *Annals of Internal Medicine*, vol. 130, pp. 461 – 470, 1999.

[13] Neural Network Toolbox user’s guide, The MATH WORKS Inc., 5: 33-34, 1996

[14] R. Jang, “ANFIS: adaptive network-based fuzzy inference system”, *IEEE Trans. on Syst.,Man, and Cybernetics*, vol. 23, No.3, pp 665 – 685, 1993.

[15] N. Kasabov and Q. Song, "DENFIS: Dynamic, evolving neural-fuzzy inference systems and its application for time-series prediction," *IEEE Trans. on Fuzzy Systems*, vol. 10, pp. 144 – 154, 2002.