Weighted Data Normalization and Feature Selection for Evolving Connectionist Systems Proceedings

Qun Song
Knowledge Engineering &
Discovery Research Institute
Auckland University of Technology
Private Bag 92006, Auckland 1020
New Zealand
email qsong@aut.ac.nz

Nikola Kasabov
Knowledge Engineering &
Discovery Research Institute
Auckland University of Technology
Private Bag 92006, Auckland 1020
New Zealand
email nkasabov@aut.ac.nz

Abstract

This paper introduces a method for weighted data normalization (WDN) that optimizes the normalization ranges for the input variables of an evolving connectionist system (ECOS). ECOS perform incremental, local area learning based on clustering. A genetic algorithm is used as part of the WDN method. The derived weights have the meaning of feature importance and are used to select a minimum set of variables (features) that optimize the performance of the ECOS. The method is illustrated on two types of ECOS. The first one is for time series prediction and the second one is an original ECOS for classification, proposed in this paper.

1. Introduction

In some 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] and therefore has the 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 reducing the size of the input vectors through keeping the most important ones. This is especially applicable to a special class of neural networks, evolving connectionist systems (ECOS), where nodes and connections evolve from an input stream of data to capture clusters and to allocate a local output function for each cluster [1, 2, 3]. Distance between existing nodes and new input vectors are 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 two presents the main principles of ECOS, while section three introduces a new ECOS model for classification, called Evolving Clustering Method for Classification (ECMC). Section four introduces a method for weighted data normalization (WDN) based on genetic algorithms (GA). Section five illustrates the method on a known ECOS for prediction, as well as on the introduced ECMC method. Conclusions are drawn in section six.

2. Principles of evolving connectionist (ECOS)

Evolving connectionist systems (ECOS) are multi-modular, connectionist architectures that facilitate modelling of evolving processes and knowledge discovery [1, 2, 3]. An ECOS may consist of many evolving connectionist modules.

An ECOS is a neural network 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. The evolving connectionist systems have the following characteristics [1]:

1) They evolve in an open space, not necessarily of fixed dimensions.

2) They learn in on-line, incremental, fast learning - possibly through one pass of data propagation.

3) They learn in a life-long learning mode.

4) They learn as both individual systems, and as part of an evolutionary population of such systems.

5) They have evolving structures and use constructive learning.

6) They learn locally and locally partition the problem space, thus allowing for a fast adaptation and tracing processes over time.

7) They facilitate different kind of knowledge representation and extraction, mostly - memory based, statistical and symbolic knowledge.

ECOS are connectionist structures that evolve their nodes (neurons) and connections through supervised incremental learning from input-output data pairs. One of the ECOS models, evolving fuzzy neural network (EFuNN), is shown as a simple version in Figure1 [2]. It 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.

ECOS learn local models from data through clustering of the data and associating a local output function for each cluster. Rule nodes evolve from the input data stream to cluster the data, and the first layer W_1 connection weights of these nodes represent the co-ordinates of the nodes in the input space. The second layer W_2 represents the local models (functions) allocated to each of the clusters.

Clusters of data are created based on similarity between data samples either in the input space (this is the case in some of the ECOS models, e.g. the dynamic neuro-fuzzy inference system DENFIS [3], and the zero instruction set computer ZISC [4]), or in both the input space and the output space (this is the case in the EFuNN models [2]). Samples that have a distance to an existing cluster center (rule node) *N* of less than a threshold *Rmax* (for the EFuNN models it is also needed that the output vectors of these samples are different from the output value of this cluster center in not more than an error tolerance *E*) are allocated to the same cluster *Nc*. Samples that do not fit into existing clusters, form new clusters as they arrive in time. Cluster centers are continuously adjusted according to new data samples, and new clusters are created incrementally.

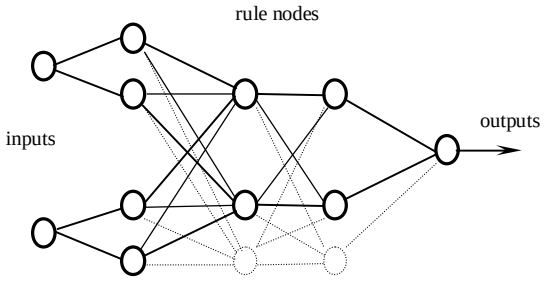

Figure 1: A simple version of an ECOS – EFuNN (from [2])

The similarity between a sample $S = (\boldsymbol{x}, \boldsymbol{y})$ and an existing rule node $N = (W_1, W_2)$ can be measured in different ways, the most popular of them being the *normalized Euclidean distance*:

$$d(S, N) = \frac{1}{n}\left[\sum_{i=1}^{n} \left| x_i - W_{1N} \right|^2\right]^{\frac{1}{2}} \qquad (1)$$

Where *n* is the number of the input variables.

ECOS learn from data and automatically create a local output function for each cluster, the function being represented in the W_2 connection weights, thus creating local models. Each model is represented as a local rule with an antecedent – the cluster area, and a consequent – the output function applied to data in this cluster, e.g.:

IF (data is in cluster *Nc*) THEN (the output is calculated with a function *Fc*) (2)

In case of DENFIS [3], first order local fuzzy rule models are derived incrementally from data, for example:

IF (the value of x_1 is in the area defined by a *Gaussian function* with a center at 0.7 and a standard deviation of 0.1) AND (the value of x_2 is in the area defined by a *Gaussian function* with a center at 0.5 and a deviation of 0.2) THEN (the output value *y* is calculated with the use of the formula $y = 3.7 + 0.5x_1 - 4.2\ x_2$) (3)

In case of EFuNNs [2], local simple fuzzy rule models are derived, for example:

IF [x_1 is High (0.7) and x_2 is *Low* (0.9)] THEN y is *High* (0.8) [radius of the input cluster 0.3, number of examples in the cluster 13] (4)

where: *High* and *Low* are fuzzy membership functions defined on the range of the variables for x_1, x_2, and y (see Figure1). The number and the type of the membership functions can either be deduced from the data through learning algorithms, or can be predefined based on human knowledge.

3. Evolving clustering method for classification (ECMC)

The ECMC method introduced in this section 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 [1, 2, 3].

There are two distinct phases of ECMC operation. During the first, learning phase, data vectors are dealt with one by one with their known classes. The learning sequence is described as the following steps:

1) If all learning data vector have been entered into the system, complete the learning phase; otherwise, enter a new input vector from the data set.
2) Find all existing rule nodes with the same class as the input vector's class.
3) If there is not such a rule node, a new one is created[*] and then go to step 1; otherwise:
4) For each of these rule nodes, if the input vector dose not lie within the associated field, increase this field if possible [**].
5) If the field increment is successful[***] or the input vector lies within an associated field, go to step 1; otherwise, a new node is created [*]. Go to step 1.
6) End of the learning procedure.

[*] Create a new rule node: the position of the new rule node is the same as the current input vector in the input space and the radius of its influence field is set to the Min-radius parameter.

[**] Suppose that the field has a radius of R^0 and the distance between the rule node and the input vector is d; the increased radius is $R^{new} = (R^0 + d) / 2$ and the rule node moves to the new position that is situated on the line connecting the input vector and the rule node before it changed its position and has the distance R^{new} to the input vector.

[***] If the new field does not include any input vectors from the data set, which belong to a different class, the increment is successful; the rule node changes its position and the field increases; otherwise, it is failed, both rule node and its field do not chang.

The learning procedure takes only one iteration (epoch) but all input vectors are retained in the system at their position in the input space.

The classification of new input vectors is performed in the following way:

1) The new input vector is entered and the distance between it and all rule nodes is calculated. If the new input vector lies within the field of one or more rule nodes associated with one class, the vector belongs to this class.
2) If the input vector does not lie within any field, the vector will belong to the class associated with the closest rule node.

To compare the performance of the ECMC method we conducted the following experiments. We used ECMC, ECF [13], ZISC [4], and a multiplayer perceptron MLP on the benchmark *Iris* data for classification. *Iris* data set has four input variables and each input vector belongs to one of three classes [6]. 50% of the whole data set is randomly selected for training data and another 50% for testing data. The parameters of each model are set as follows and the average test results of 50 experiments are shown in Table 1.

Parameters for ECMC: Min-radius = 0.02, one epoch. Parameters for ECF: Max-field = 1; Min-field = 0.2; MF = 2; MofN = 1; 5 epochs.

Parameters for ZISC: Max-field = 4096; Min-field = 1; 5 epochs.

Parameters for MLP: number of hidden layer 1; number of neurons in the hidden layer 12; learning epochs 100 with *Levenberg-Marquardt* learning algorithm [7].

Table 1. Comparison between ECMC and other models on the *Iris* classification data

	Rule nodes (or neurons)	Average number of test errors
ECMC	9.8	4
ECF	12.7	4
ZISC	14.4	6.1
MLP	12	4.6

The results from Table1 show that ECMC compares favorably with other classification methods. The ECMC method is distance-based. As it changes the influence field in an eccentric manner, ECMC will have less rule

nodes than the ECF or the ZISC models, which change their influence fields in a concentric manner.

4. Weighted data normalization (WND) and feature selection based on genetic algorithm

The WDN method described here makes use of *evolutionary computation* (EC) techniques [8, 9, 10] and of *genetic algorithms* (GA) in particular [8]. A GA algorithm applies a multi-point, probabilistic search in the whole search space to discover a global optimum subject to an objective (fitness) function. Methods based on EC techniques for optimizing parameters of MLP and other traditional neural network models are published in [11, 12]. In [13, 14], a GA is used to optimize some of the parameters of ECOS models.

The WDN method proposed here optimizes the normalization intervals (ranges) of the input variables and allocates weights to each of the variables from a data set. The method consists of the following steps:

1) The training data is preprocessed first by a general normalization method. There are several ways for this: a) normalizing a given data set so that they fall in a certain interval, e.g. [0, 1], [0, 255] or [-1, 1] etc [9]; b) normalizing the data set so that the inputs and targets will have means of zero and standard deviations of 1 [9]; c) normalizing the data set so that the deviation of each variable from its mean normalized by its standard deviation [10]. In the WDN, we normalize the data set in the interval [0, 1].
2) The weights of the input variables $x_1, x_2,\ldots, x_n$ represented respectively by $w_1, w_2,\ldots,w_{n,,}$ with initial values of 1,1,…1 , form a chromosome for a consecutive GA application. The weight w_i of the variable x_i defines its new normalization interval $[0, w_I]$.
3) A GA is run on a population of connectionist learning modules for different chromosome values, over several generations. As a fitness function, the root mean square error RMSE of a trained connectionist module on the training data or on a validation data is used, or alternatively – the number of the created rule nodes can be used as a fitness function that needs to be minimized. The GA runs over generations of populations and standard operations are applied such as binary encoding of the genes (weights); roulette wheel selection criterion; multi-point crossover operation for crossover.
4) The connectionist model with the least error is selected as the best one, and its chromosome – the vector of weights $[w_1, w_2,\ldots,w_n]$ defines the optimum normalization range for the input variables.
5) Variables with small weights are removed from the feature set and the steps from above are repeated again to find the optimum and the minimum set of variables for a particular problem and a particular connectionist model.

The above WDN method is illustrated in the next section on two case study ECOS and on two typical problems, namely EFuNN for a time series prediction, and ECMC for classification.

5. Case study examples ofpplying the WND method to ECOS

5.1 EFuNN with WDN for Prediction

In the present paper, EFuNN is applied to the time series prediction. Improved learning with the WDN method is demonstrated on the Mackey-Glass (MG) time series prediction task [6]. The MG time series is generated with a time-delay differential equation as follows:

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} \qquad (5)$$

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.

The following parameters are set in the experiments for the EFuNN model: *Rmax*=0.15; *E*=0.15; 3 membership functions. The following GA parameter values are used: for each input variable, the values from 0.16 to 1 are mapped onto 4 bit string; the number of individuals in a population is 12; mutation rate is 0.001; termination criterion (the maximum epochs of GA operation) is 100 generations; the *root-mean square error* RMSE on the training data is used as a fitness function. The resulted weight values, the number of the rule nodes created by EFuNN with such weights, the training RMSE and testing RMSE are shown in Table 2. For a comparison, EFuNN results with the same parameters, the same training data and testing data, but without WDN are also shown in Table 2.

Table 2: Comparison between EFuNN without WDN and EFuNN with WDN

	Normalization Weights	Number on Rule Nodes	Training RMSE	Testing RMSE
EFuNN without WDN	1, 1, 1, 1	87	0.053	0.035
EFuNN with WDN	0.4, 0.8 0.28 0.28	77	0.05	0.031

With the use of the WDN method, a better prediction results is obtained for a significantly less number of rule nodes (clusters) evolved in the EFuNN model. This is because of the better clustering achieved when different variables are normalized differently and the normalization reflects on their importance.

5.2 ECMC with WDN for Classification and Feature Extraction

In this section, the ECMC with WDN is applied on the *Iris* data for both classification and feature selection. The same as the experiments in the section 3, all experiments in this section are repeated 50 times with the same parameters and the results are averaged. 50% of the whole data set is randomly selected as training data and another 50% as testing data. The following parameters are set in the experiments for the ECMC model: *Min-radius* 0.02; each of the weights for the four normalized input variables is a value from 0.1 to 1, and is mapped into a 6-bit binary string.

The following GA parameters are used: number of individuals in a population 12; mutation rate 0.005; termination criterion (the maximum epochs of GA operation) 50; fitness function - the number of created rule nodes. The resulted weight values, the number of rule nodes created by ECMC with such weights and the number of errors on the testing data are shown in Table 3. For comparison, ECMC classification results with the same parameters, the same training data and testing data but without WDN are also shown in Table 3.

From the results, we can see that the weight of the first variable is much smaller than the weights of the other variables. The weights can be used as a measure of the importance of the variables and the least important variables can be removed from the input. Same experiment is repeated without the first input variable and the results have improved as shown in Table3. If another variable is removed, and the total number of input variables is 2, the test error increases, so it can be assumed that for the particular ECMC model the optimum number of input variables is 3.

Table 3: Comparison between ECMC without WDN and ECMC with WDN

	Normalization Weights	Number of rule nodes	Number of test errors
4 Inputs without WDN	1, 1, 1, 1	9.8	3.8
4 Inputs with WDN	0.25, 0.44 0.73 1	7.8	3.1
3 Inputs without WDN	1, 1, 1	8.8	3.7
3 Inputs with WDN	0.50, 0.92, 1	8.1	2.9
2 Inputs without WDN	1,1	7.7	3.2
2 Inputs with WDN	1, 0.97	7.4	3.1

6. Conclusions

The proposed method for weighted data normalization (WDN) is a generic one and can be applied to any connectionist models, but it is especially efficient when applied to *evolving connectionist systems* (ECOS) as the latter use local, clustering-based learning algorithms. The WDN method is also efficient when used for feature selection. Further development of the method includes using linear along with non-linear normalization, using different types of normalization across variables and across areas of the problem space for achieving a better performance of the system.

The proposed evolving clustering method for classification ECMC uses clusters of different shapes rather than hyper-spheres, as it is the case in other ECOS methods, and results in a less number of rule nodes and less error. New ECOS methods are being developed at present that will have all the parameters and the features optimised together.

7. Ackonwledgements

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

References

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

[2] Kasabov, N. "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.

[3] Kasabov, N. and Song, Q. "DENFIS: Dynamic, evolving neural-fuzzy inference systems and its application for time-series prediction," IEEE Trans. on Fuzzy Systems, vol.10, No.2, 144-154, April 2002.

[4] ZISC Manual, Silicon Recognition, www.silirec.com, 2001.

[5] Song, Q. Kasabov, N.,"ECM - A Novel On-line, Evolving Clustering Method and Its Applications", ANNES 2001, Dunedin, New Zealand, 22 - 24, November, 2001, 87 – 92.

[6] Blake, C. and Merz, C. (1998) Repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html Irvine, CA: University of California, Department of Information and Computer Science.

[7] Baeck, T. *Evolutionary algorithm in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*, Oxford University Press, New York (1995)

[8] Holland, J. H. *Adaptation in natural and artificial systems*, The University of Michigan Press, Ann Arbor, MI (1975)

[9] Koza, J. R. *Genetic Programming*, MIT Press (1992)

[10] Goldberg, D. E. *Genetic Algorithms in Search, Optimization and machine Learning*, Addison-Wesley, Reading, MA (1989).

[11] Fogel, D., Fogel, L. and Porto, V. (1990) Evolving neural networks, Biological Cybernetics, vol.63, 487-493

[12] Yao, X. (1993) Evolutionary artificial neural networks, Int. Journal of Neural Systems, vol.4, No.3, 203-222

[13] Kasabov, N. and Song, Q. "GA-Optimisation of evolving connectionist systems for classification with a case study from bioinformatics," Proc. of ICONIP'2002, Singapore, November, 2002, IEEE Press, 602-605.

[14] Kasabov, N., Song, Q. and Nishikawa, I, "Evolutionary computation for parameter optimisation of on-line evolving connectionist systems for prediction of time series with changing dynamics",. in Int. Joint Conf. on Neural Networks IJCNN'2003. 2003. USA.