Ensembles of EFuNNs: An architecture for a multi module classifier

Brendon J.Woodford and Nikola K. Kasabov
Department of Information Science
University of Otago
PO Box 56, Dunedin, New Zealand
(E-mail : bjwoodford@infoscience.otago.ac.nz, nkasabov@infoscience.otago.ac.nz)

Abstract

This paper introduces an extension of the existing theory of the Evolving Fuzzy Neural Network (EFuNN) to also be a multi-module classifier. We call this proposed architecture multiEFuNN. The incorporation of the Evolving Clustering Method (ECM) is used to partition the input space of the dataset and also determine how many EFUNNs are to be used to classify it. The main advantages of this mult-module classifier is in the area of on-line learning and recall of data where there are a growing number of classes with more data coming. Preliminary results from experiments conducted using this architecture are compared to the existing single EFuNN classifier and reported.

1.Introduction

1.1.The deficiencies of an individual EFuNN for classification problems

Although the EFuNN works well in on-line learning and recall environments [7, 9], some disadvantages have been identified in its architecture and operation:

- When the number of classes are large and the vector sizes are large, the training time is dramatically increased apraching the time of an off-line learning system.
- The accuracy of the EFuNN is significantly reduced under validation of the training dataset and recall with the test dataset.
- The fuzzy rules that are extracted are much larger than and therefore it makes it difficult to interpret them. Although rule aggregation methods can be applied [11], whilst the size of the input vectors remains the same, the smaller the size of a rule is, the easier it is to interpret.

1.2.Neural Network Ensembles

When these problems have existed within the context of the traditional neural network models such as the Multi Layer Perceptron (MLP), several solutions have been proposed using neural network ensembles [15, 13, 1]. These solutions normally take the form of a collection of single neural networks connected in some fashion that work together to produce more robust classifiers. To generate a single output class, another layer of the classification system is added that finds a consensus between the individual neural networks.

However there are many different approaches that have been used to find consensus between these collective neural networks. They are normally related back to the distribution and nature of the dataset used to train and test the proposed architecture. Furthermore, selection of the number of individual neural networks to model the problem is based on either an arbitrary number or derived from a statistical analysis of the dataset.

The knowledge built up over time using the on-line learning method of the EFuNN is governed by what data has already been supplied to it from the environment. Therefore a statistical analysis of the dynamic data source to derive a number of EFuNNs to model the problem does not comfortably fit into the Evolving Connectionist System (ECOS) paradigm [4].

The paper suggests architecture for a multi-module classifier constructed from one or more EFuNNs that addresses these issues. The Evolving Clustering Method (ECM) is used to dynamically select the number of EFuNNs over time. These two methods are combined to form multi-module architecture that is dynamically constructed as data is fed into it. We illustrate this approach on two case studies using horticulture datasets.

2.Evolving Fuzzy Neural Networks EFuNNs

2.1.A general description

EFuNNs are structures that evolve according to the ECOS principles [3]. The architecture is depicted in Figure 1. EFuNNs adopt some known techniques from [2, 12, 14] and from other known NN techniques, but here all nodes in an EFuNN are created during (possibly one-pass) learning. The nodes representing MF (fuzzy label neurons) can be modified during learning. Each input variable is represented here by a group of spatially arranged neurons to represent a fuzzy quantisation of this variable. For example, two neurons can be used to represent "small" and "large" fuzzy values of a variable. Different membership functions (MF) can be attached to these neurons (triangular, Gaussian, etc.). New neurons can evolve in this layer if, for a given input vector, the corresponding variable value does not belong to any of the existing MF to a degree greater than a membership threshold. A new fuzzy input neuron, or an input neuron, can be created during the adaptation phase of an EFuNN.

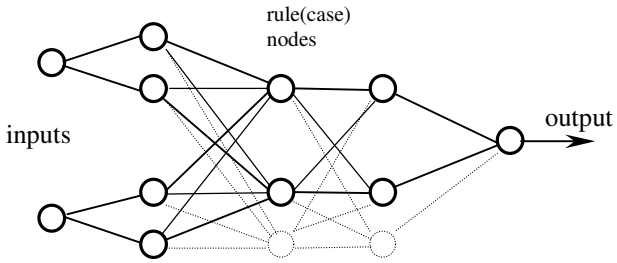

Figure 1: A EFuNN structure of 2 inputs (input variables), 2 fuzzy linguistic terms for each variable (2 membership functions). The number of the rule (case) nodes can vary. Two output membership functions are used for the output variable.

The EFuNN algorithm, for evolving EFuNNs, has been first presented in [4]. A new rule node rn is created and its input and output connection weights are set as follows: *W1(rn)=EX*; *W2(rn) = TE*, where *TE* is the fuzzy output vector for the current fuzzy input vector *EX*. In "one-of-n" EFuNNs, the maximum activation of a rule node is propagated to the next level. Saturated linear functions are used as activation functions of the fuzzy output neurons. In case of "many-of-n" mode, all the activation values of rule (case) nodes, that are above an activation threshold of *Athr*, are propagated further in the connectionist structure.

2.2.The EFuNN learning algorithm

The EFuNN evolving algorithm is given as a procedure of consecutive steps [3, 4, 5, 6, 8]:

1. Initialise an EFuNN structure with a maximum number of neurons and zero-value connections. Initial connections may be set through inserting fuzzy rules. If initially there are no rule (case) nodes connected to the fuzzy input and fuzzy output neurons with non-zero connections, then *connect* the first node *rn=1* to represent the first example *EX=x1* and set its input *W1(rn)* and output *W2(rn)* connection weights as follows: <Connect a new rule node *rn* to represent an example *EX* >: *W1(rn)=EX*;*W2(rn) = TE*, where *TE* is the fuzzy output vector for the (fuzzy) example it EX.

2. WHILE <*there are examples*> DO
   Enter the current example x_i , *EX* being the fuzzy input vector (the vector of the degrees to which the input values belong to the input membership functions). If there are new variables that appear in this example and have not been used in previous examples, create new input and/or output nodes with their corresponding membership functions as explained in [8]

3. Find the normalized fuzzy similarity between the new example *EX* (fuzzy input vector) and the already stored patterns in the case nodes *j=1,2..rn*: *Dj = sum (abs (EX - W1(j))/ sum (W1(j)) + sum (EX).*

4. Find the activation of the rule (case) nodes *j*, *j=1..rn*. Here radial basis activation function, or a saturated linear one, can be used on the *Dj* input values i.e. *A1(j) = radbas(Dj)*, or *A1(j) = satlin(1 - Dj)*. Previous activation at the same layer can be taken into account [8]

5. Update the local parameters defined for the rule nodes, e.g. Short-Term Memory (STM), age, average activation as pre-defined.

6. Find all case nodes *j* with an activation value *A1(j)* above a sensitivity threshold *Sthr*.
7. If there is no such case node, then <*Connect a new rule node*> using the procedure from step 1.
   ELSE
   Find the rule node *inda1* that has the maximum activation value. (*maxa1*).
   (a) in case of "one-of-n" EFuNNs, propagate the activation *maxa1* of the rule node inda1 to the fuzzy output neurons. Saturated linear functions are used as activation functions of the fuzzy output neurons: *A2 = satlin (A1(inda1) * W2)*.
   (b) in case of "many-of-n mode", only the activation values of case nodes that are above an activation threshold of $Athr$ are propagated to the next neuronal layer.
8. Find the winning fuzzy output neuron *inda2* and its activation *maxa2*.
9. Find the desired winning fuzzy output neuron *indt2* and its value *maxt2*.
10. Calculate the fuzzy output error vector: *Err = A2 - TE*.
11. IF (*inda2* is different from *indt2*) or *(abs(Err (inda2)) > Errthr*) then <*Connect/create a rule node*>
12. Update: (a) the input, and (b) the output connections of rule node *k=inda1* as follows:
    (a) *Dist = EX-W1(k)*; *W1(k) = W1(k) + lr1*Dist*, where *lr1* is the learning rate for the first layer;
    (b) *W2(k) = W2 (k) + lr2*Err*maxa1*, where *lr2* is the learning rate for the second layer. If STM is used update the feedback connections in the rule layer.
13. Prune rule nodes *j* and their connections that satisfy the following fuzzy pruning rule to a pre-defined level: *IF (node (j) is OLD) and (average activation A1av(j) is LOW) and (the density of the neighbouring area of neurons is HIGH or MODERATE) and (the sum of the incoming or outgoing connection weights is LOW) and (the neuron is NOT associated with the corresponding "yes" class output nodes (for classification tasks only) THEN the probability of pruning node (j) is HIGH.*

    The above pruning rule is fuzzy and it requires that all fuzzy concepts such as OLD, HIGH, etc., are defined in advance. As a partial case, a fixed value can be used, e.g. a node is old if it has existed during the evolving of a FuNN from more than 60 examples.
14. Aggregate rule nodes into layer clusters (prototype) nodes (see [12])
15. END of the WHILE loop and the algorithm
16. Repeat steps 2-15 for a second presentation of the same input data or for ECO training if needed.

2.3.The Evolving Clustering Method (ECM)

The ECM is an on-line, maximum distance-based clustering method that has been proposed to implement a scatter partitioning of the input space for the purpose of creating fuzzy inference rules. This method has successfully been applied as a component of the Dynamic Evolving Neural Fuzzy Inference System (DENFIS) model. This method is a fast, one-pass algorithm for a dynamic estimation of the number of clusters in a set of data and for finding their current centre in the input data space. With this method, cluster centres are represented by the evolved rule nodes. Refer to [17, 16] for a complete description of the algorithm

We can employ the ECM method not only select the number of clusters within our multi-module classifier but also seed each individual EFuNN with the rules derived using this method.

3.Creating a multi-module classification architecture from the EFuNNs

3.1.Training the multiEFuNN

1. For each input vector x:
2. Apply the input vector x to the ECM method to determine the cluster to which the vector belongs to.
3. For each cluster do:

- Create an EFuNN and add the cluster centre as the first rule node.
- Train incementally the EFuNN on the data from the input dataset that are associated with the cluster centre.

This procedure can be applied in an on-line or in an off-line mode.

3.2.Testing the multiEFuNN

1. For each instance of the test set do:
   - Present the instance to each individual EFuNN in the mutiEFuNN.
   - Select the output class that the majority of the EFuNNs had a consensus on. The current method used is to count the number of EFuNNs that selected a particular output class and select the output class that had the most number of EFuNNs select it.

4.Case studies using the mutltiEFuNN approach

The first case study is of a classification problem with image data. The second case study is of a classification problem illustrated on a dataset of Near Infrared Spectrometer readings of clones of tree of the variety *pinus radiata*.

4.1.Image data of damage to apples by pests in an orchard

96 full colour images of pest damage to apples were selected. Each image was processed using a wavelet transform as described in [18, 10] and a 768 wavelet data vector produced. There were three types of pests that could have caused damage to the apple in the image namely 1) Codling Moth, 2) Leafroller, and 3) Appleleaf Curling Midge. 70 images were used as the training data set and 26 images used as the test data set.

The ECM processing found three distinct clusters in the dataset . Training the multiEFuNN using rule aggregation, created a structure with a total of 52 rule nodes. On testing the multiEFuNN 20 examples were correctly classified with 6 incorrectly classified producing a 77% accuracy. Results of the confusion matrix are shown in in Table 1.

Testing Data										
12 EFuNNs: Parameters - 2 MFs, Err=0.1, No Pruning, Aggregation on, Radius=0.5, Normalisation and Fuzzification on.										
CM=Codling Moth, LR = Leafroller, ALM = Appleleaf Curling Midge										
	CM	**LR**	**ALM**		**Sum**	**Percent**				
CM	**0**	1	1		2	0				
LR	1	**9**	3		13	69				
ALM	0	0	**11**		11	100				
				20						
Sum	1	10	15		**26**					
Percent	0	90	73			**77**				

Table 1: multiEFuNN tested on image data

On testing the single EFuNN 10 images were correctly classified and 16 incorrectly classified producing an accuracy of 38% as shown in Table 2.

Testing Data										
Single EFuNN: Parameters - 2 MFs, Err=0.1, No Pruning, Aggregation on, Radius=0.5, Normalisation and Fuzzification on.										
CM=Codling Moth, LR = Leafroller, ALM = Appleleaf Curling Midge										
	CM	**LR**	**ALM**		**Sum**	**Percent**				
CM	**0**	1	1		2	0				
LR	4	**5**	3		13	38				
ALM	0	6	**5**		11	45				
				10						
Sum	4	12	9		**26**					
Percent	0	42	56			**38**				

Table 2: single EFuNN tested on image data

4.2.Pine wax data set

This dataset has been constructed from clones of *pinus radiata needles*. Initially waxes were solvent extracted from each sample and analysed in a Fourier Transform Infrared Spectrometer. Each spectra was applied to a wavelet transform producing a vector of 307 coefficients. There are twenty different clones with 50 samples of each clone combining to form a dataset of 1000 samples. 800 samples were used for training and 200 samples were used for testing.

The ECM method found thirteen distinct clusters. Training the multiEFuNN using rule aggregation created a structure of 336 rules nodes. One testing the multiEFuNN 113 examples were correctly classified with 87 examples incorrectly classified producing an accuracy of 56%. Again results of the confusion matrix are shown in Table 3. We compared this to a single EFuNN structure which created 42 rule nodes.

Testing Data																						
13 EFuNNs : Parameters - 2 MFs, Err=0.1, No Pruning, Aggregation on, Radius=0.5, Normalisation and Fuzzification on.																						
	C1	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**	**C9**	**C10**	**C11**	**C12**	**C13**	**C14**	**C15**	**C16**	**C17**	**C18**	**C19**	**C20**		**Sum**	**Percent**
C1	**5**		1		2																	8	63
C2		**6**				3																9	67
C3			**5**							1												6	83
C4		2		**4**			3															9	44
C5	2				**7**			2		2												13	54
C6						**5**			2													7	71
C7							**5**											1				6	83
C8	2			2				**5**			2											11	45
C9							2		**5**										1			8	63
C10										**7**			2	1				1		2		13	54
C11	1										**7**		1									9	78
C12			3					3				**7**		2								15	47
C13				2					2				**5**		2	3				1		15	33
C14					1	2								**6**				2	2			13	46
C15		1										2	2		**6**		2					13	46
C16																**6**			2			8	75
C17									1			1					**6**			2		10	60
C18				2											2			**6**				10	60
C19											1								**5**			6	83
C20		1	1											1		1	2			**5**		11	45
																					113		
Sum	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10		**200**	
Percent	50	60	50	40	70	50	50	50	50	70	70	70	50	60	60	60	60	60	50	50			**57**

Table 3: multiEFuNN tested on pine data

On testing the single EFuNN 23 samples were correctly classified and 177 incorrectly classified producing an accuracy of 11.5% as shown in Table 4.

Testing Data																						
Single EFuNN : Parameters - 2 MFs, Err=0.1, No Pruning, Aggregation on, Radius=0.5, Normalisation and Fuzzification on.																						
	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20		Sum	Percent
C1	**1**		1		2																	4	25
C2		**1**		1		3	1	1	1		1	1	1		1							12	8
C3	1		**1**	1						1				1			1	1				7	14
C4		2		**1**			3		1		1	1	1			1			1			12	8
C5	2				**1**			2		2		1		1	1		1	1				12	8
C6		1	1			**1**			2		1		1	1		1			1	1		11	9
C7				1	1		**1**		1	1				1		1		1				8	13
C8	2	1		2				**1**			2	1		1	1				1			12	8
C9		1	1		2	1	2		**1**	1	1					1	1	1	1	1		15	7
C10						1				**1**			2	1	1	1		1		2		10	10
C11	1		1				1	1			**1**	1	1									7	14
C12		1	3			1		3				**1**		2			1	1				13	8
C13	1	1		2					2	1	1		**1**		2	3				1		15	7
C14					1	1	1			1				**1**				2	2			9	11
C15		1	1		1			1				2	2		**1**		2			**1**		12	8
C16	1						1				1					**1**			2			6	17
C17						1		1	1	1		1	1		1		**1**			2		10	10
C18				2	1						1				2		1	**2**				9	22
C19	1					1			1										**2**			5	40
C20		1	1		1					1		1		1		1	2			**2**		11	18
																					23		
Sum	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10		**200**	
Percent	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	20	20	20			**12**

Table 4: single EFuNN tested on pine data

5.Conclusion

This paper suggests a novel architecture for an ensemble of EFuNNs consisting of a combination of the ECM algorithm and the EFuNN. This architecture has been developed to address identified deficiences in the EFuNN learning algorithm when it is attempting to classify datasets with a large number of classes. The results from the initial experiments are encouraging and warrant further investigation into this architecture.

6.Conclusion

This paper suggests a novel architecture for an ensemble of EFuNNs consisting of a combination of the ECM algorithm and the EFuNN. This architecture has been developed to address identified deficiences in the EFuNN learning algorithm when it is attempting to classify datasets with a large number of classes. The results from the initial experiments are encouraging and warrant further investigation into this architecture.

Acknowledgements

This research is part of a research programme funded by the New Zealand Foundation for Research Science and Technology, UOOX0016

References

[1] P. Hartono and S. Hashimoto. Effective Learning in Noisy Environment Using Neural Network Ensemble. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00)*, pages II–179–II–184, 2000.

[2] N. Kasabov. *Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering*. MIT Press, Cambridge: MA, first edition, 1996.

[3] N. Kasabov. ECOS: A Framework For Evolving Connectionist Systems And The eco Learning Paradigm. In *Proc. of ICONIP'98.,Kitakyushu, Oct 1998*, pages 1232–1236, 1998.

[4] N. Kasabov. Evolving Fuzzy Neural Networks - Algorithms, Applications and Biological Motivation. In *Proc. of Iizuka'98, Iizuka, Japan*, pages 271–274, 1998.

[5] N. Kasabov. Fuzzy Neural Networks, Rule Extraction and Fuzzy Synergistic Reasoning Systems. *Research and Information Systems*, 8:45–59, 1998.

[6] N. Kasabov. *Evolving Connectionist And Fuzzy Connectionist System For On-Line Decision Making And Control*, volume Soft Computing in Engineering Design and Manufacturing. Springer-Verlag, 1999.

[7] N. Kasabov. Evolving connectionist systems and applications for adaptive speech recognition. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'99)*, July 10–16, 1999.

[8] N. Kasabov. Learning, reasoning, and rule extraction in Evolving Fuzzy Neural Networks. *Submitted to Neurocomputing*, 1999.

[9] N. Kasabov. Evolving Connectionist Systems for On-line, Knowledge-based Learning. *IEEE Transactions on Man, Machine and Cybernetics*, 2000. in press.

[10] N. K. Kasabov, S. A. Israel, and B. J. Woodford. The Application of Hybrid Evolving Connectionist Systems to Image Classification. *Journal of Advanced Computational Intelligence*, 4(1):57–65, 2000.

[11] N. Kasbov and B. Woodford. Rule insertion and rule extraction from evolving fuzzy neural networks: algorithms and applications for building adaptive, intelligent expert systems. In *Proceedings of the 1999 IEEE Fuzzy Systems Conference*, volume 3, pages 1406–1411. The IEEE, Kyunghee Printing Co, August 22–25 1999.

[12] T. Kohonen. The Self-Organizing Map. *Proceedings of the IEEE*, 78(9):1464–1497, 1990.

[13] S. Lawrence, C. L. Giles, A. Tsoi, and A. Back. Face recognition: A convolutional neural network approach. *IEEE Transactions on Neural Networks*, 8(1):98–113, 1997.

[14] C. T. Lin and C. S. Lee. *Neuro Fuzzy Systems*. Prentice Hall, 1996.

[15] D. W. Opitz and J. W. Shavlik. Actively Searching for an Effective Neural-Network Ensemble. *Connection Science*, 8:337–353, 1996.

[16] Q. Song and N. Kasabov. DENFIS: Dynamic Evolving Neural-Fuzzy Inference System and Its Application for Time-series Prediction. *IEEE Transactions on Fuzzy Systems*, 2000. submitted.

[17] Q. Song and N. Kasabov. Dynamic Evolving Neural-Fuzzy Inference System (DENFIS): On-line Learning and Application for Time-series Prediction. In *Proceedings of the 6th International Conference on Soft Computing, Iizuka, Fukuoka, Japan*, pages 696–701, October 2000.

[18] B. J. Woodford, N. K. Kasabov, and C. H. Wearing. Fruit Image Analysis Using Wavelets. In *Proceedings of the ICONIP/ANZIIS/ANEES'99 International Workshop*, pages 88–91. University of Otago Press, November 22-23 1999.