On-Line Expansion of Output Space in Evolving Fuzzy Neural Networks

Akbar Ghobakhlou, Michael Watts, Nikola Kasabov

Department of Information Science
University of Otago, PO Box 56,
Dunedin, New Zealand
E-mail: Akbar,mjwatts,nkasabov@infoscience.otago.ac.nz
Web: http://kel.otago.ac.nz/

Abstract

The paper presents a methodology for expanding the number of classes an Evolving Fuzzy Neural Network(EFuNN) is able to classify. This is a useful trait for such applications as adaptive speech recognition systems, and strongly complements the ability of EFuNNs to adapt to new examples of known classes. Experiments with isolated word recognition demonstrate the efficacy of EFuNNs in this problem domain, while further experiments expand these networks with new words, demonstrating the methodology that is the crux of this work. The experimental results show that the suggested methodology is a promising approach to the problem of expanding adaptive connectionist classification systems to accommodate new classes.

1 Introduction

The central tenet of online, adaptive systems is that they are able to modify themselves to account for new data. Several paradigms have been suggested that are useful for this task, including Resource Allocation Networks (RAN) [8], Nearest-Neighbour MLPs (NN-MLP) [9], and Evolving Connectionist Systems (ECoS) [2]. While all of these are able to modify parts of their structures to accommodate new examples presented to them, none are able to adapt to the requirement to identify new classes when they appear on-line, during the system operation. For some applications, such as adaptive speech recognition systems, this is absolutely vital, as such systems are far less useful if they cannot learn to add and recognise new words during their operation.

Evolving Fuzzy Neural Networks (EFuNNs) [3] are a fusion of the ECoS principles and the principles of Fuzzy Neural Networks. Although they are quite capable of adapting to accommodate new examples, they are still unable to accommodate in their output space new classes when these are presented to them. While EFuNNs are useful for speech recognition systems, they still require the ability to adapt to new classes to maximise their usefulness in this application area.

This paper describes an attempt at fulfilling this requirement. It describes a methodology for adding new output nodes, corresponding to new data classes, to an EFuNN, allowing it to adapt to new classes while retaining all of its previously learned knowledge. The methodology is illustrated on the task of isolated spoken word recognition.

2 The Evolving Connectionist System (ECoS) Paradigm

The Evolving Connectionist System (ECoS) paradigm was suggested by Kasabov in [2]. It was developed in order to address several perceived problems with traditional, static connectionist paradigms, and is based upon the following principles:

- rapid, one pass training
- structural adaptation through environmental interaction
- resistance to catastrophic forgetting
- capability for both memorisation of data and generalisation
- discovery of spatial relationships
- knowledge is stored in exemplar nodes within the structure
- ECoS networks may be reduced in size without significant loss of knowledge, through aggregation of exemplar nodes.

ECoS are similar in concept to the Resource Allocation Network (RAN) suggested by Platt [8], and also to the Nearest-Neighbour MLP (NN-MLP) investigated by Zhao [9]. Where ECoS differ from the other paradigms is in the extreme flexibility of their structure, their ability to learn in both supervised and unsupervised modes, and in their ability to identify spatial similarities between examples.

3 Evolving Fuzzy Neural Networks

The Evolving Fuzzy Neural Network (EFuNN) [3] is a five neuron layer network with four layers of connections. It is derived from a fusion of the Fuzzy Neural Network FuNN [6] and the ECoS principles. The structure of an EFuNN is as follows: The first, input, neuron layer represents crisp input variables. The second, condition, layer represents the fuzzy membership functions attached to each input. Shouldered triangular membership functions are used, with the bounds of each MF being equal to the centres of the neighbouring MF. Thus, it is possible to represent each MF with a single parameter, represented in EFuNN by the connection weights in the input to condition connection layer. The third, or rule, layer represents the associations between the fuzzy inputs and outputs, i.e. it represents the system's fuzzy rules. The fourth, or action, layer represents the output membership functions while the final layer represents the crisp output value. The action to output connection layer is of the same form as the input to condition layer, i.e. connection weights within this layer represent the centres of the output MF. An idealised EFuNN, with three inputs and two outputs, is shown in Figure 1

The activation of an EFuNN rule node is determined by the normalised fuzzy difference between the condition node activations (the fuzzified input values) and the connection weights incoming to that node. Thus, activation of an EFuNN rule node is proportional to the distance between the fuzzified input values and the previously seen examples as represented by the connection weights. The activations of the rule layer nodes are propagated according to two different strategies: "One of N", and "Many of N". With "One of N" recall, only the most highly activated node has it's activation node propagated to the following layers. With "Many of N" recall, up to M of those nodes that are activated above a specified level (known as the Activation Threshold) have their activation values propagated onwards. The rule and action layer both have saturated linear activation functions, while the activation functions of the condition and output layers are constructed to appropriately handle the fuzzification and defuzzification respectively. EFuNNs allow for IF-Then rule extraction and there are special algorithms developed for this purpose. However, we do not use this option as the aim of this paper is to explore the concept of output expansion.

4 The EFuNN Training Algorithm

The EFuNN learning algorithm is not an evolutionary, but an evolving algorithm that is based on continues learning and structural adaptation. The EFuNN learning algorithm is as follows [4]:

- propagate the current input vector I through the network

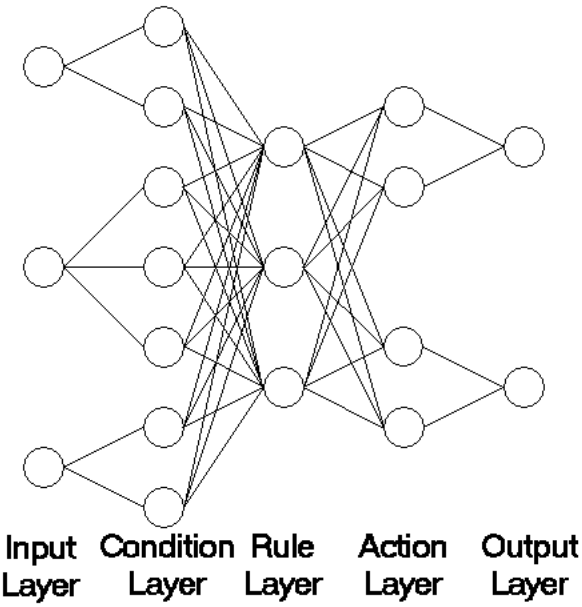

Figure 1: A simplified and exemplified (3 inputs, 2 output) diagram of an EFuNN.

- find the highest activated rule node (the winner)
- IF the maximum activation a_{max} is less than the sensitivity threshold S_{thr}
  - add a node
- ELSE
  - evaluate the error between the components of the calculated output vector O_c and the desired output vector O_d
  - IF the error over the desired output is greater than the error threshold E_{thr} OR the desired output node is not the most highly activated
    - add a node
  - ELSE
    - update the connections to the winning hidden node
- repeat for each training vector.

When a node is added, it's incoming connection weight vector is set to the fuzzified input vector I, and it's outgoing weight vector is set to the fuzzified desired output vector O_d.

The incoming weights to the winning node j are modified according to equation 1, where $W_{i,j}^t$ is the connection weight from input i to j at time t, $W_{i,j}^{t+1}$ is the connection weight from input i to j at time $t+1$, η_1 is the learning rate one parameter, and I_i is the ith component of the input vector I.

The objective of this function is to decrease the distance between I and W_j.

The outgoing weights from node j are modified according to equation 2, where $W^t_{j,o}$ is the connection weight from j to output o at time t, $W^{t+1}_{j,o}$ is the connection weight from j to o at time $t+1$, A_j is the activation of j, and E_o is the error at o. The objective of this function is to decrease the error over the outputs.

$$W^{t+1}_{i,j} = W^t_{i,j} + \eta_1(I_i - W^t_{i,j}) \quad (1)$$

$$W^{t+1}_{j,o} = W^t_{j,o} + \eta_2(A_j \times E_o) \quad (2)$$

4.1 The EFuNN Aggregation Algorithm

Aggregation of rule nodes [7] can be employed to control the size of the evolving layer (i.e. the rule layer) during the learning process. The principle of aggregation is to merge those nodes which are spatially close into one node. Aggregation can be applied for every (or after every n) training examples. It will generally improve the generalisation capability of EFuNN. The aggregation algorithm is as follows:

FOR each rule node r_j, $j = 1 : n$,

where n is the number of rule nodes and $W1$ is the connection weights matrix between the condition and rule layers and $W2$ is the connection weights matrix between the rule and action layers.

- find a subset R of rule nodes for which the normalized Euclidean distances $D(W1_{r_j}, W1_{r_a})$ and $D(W2_{r_j}, W2_{r_a})$ $r_j, r_a \in R$ are below the thresholds $w1Thr$ and $w2Thr$ respectively.

- merge all rule nodes from the subset R into a new rule node r_{new} and update $W1_{r_{new}}$ and $W2_{r_{new}}$ using the following formulae:

$$W1_{r_{new}} = \frac{\sum_{r_a \in R}(W1_{r_a})}{m}$$
$$W2_{r_{new}} = \frac{\sum_{r_a \in R}(W2_{r_a})}{m}$$

where m denotes the number of rule nodes in the subset R.

- delete the rule nodes $r_a \in R$

5 On-Line Expansion of the Output Space in EFuNN

It is highly desirable in some application areas, such as speech recognition systems to be able to expand the vocabulary of the recognition system without having to start again from scratch. EFuNN is suitable for this task because it uses local learning which tunes only the connection weights of the local node, so anything the rule nodes have captured as knowledge will be local and only covering a "patch" of the input-output space.Thus, adding new classes or new inputs does not require re-training of the whole system on the new and the old data as it is in the traditional NN.

The task is to introduce an idea for on-line expansion of the output layer in EFuNN in order to accommodate new classes. As described in section 3 the EFuNN is a five neuron layer network with four layers of connections. Each node in the output layer represents a particular class in the problem domain.

To add a new node to the output layer, the structure of the existing EFuNN first needs to be modified to encompass the new output node. This modification affects both the output and action layers and the connections between the action and rule layers. The graphical representation of this process is shown in Figure 2. The connection weights between the action and rule layer are initialised to the fuzzification of crisp output zero. In this manner the new output node is set by default to classify all previously seen classes as negative. Once the internal structure of the EFuNN is modified to accommodate the new output class, the EFuNN is further trained on both the new and old class data. As a result of the training process new rule nodes are created to represent the new class data.

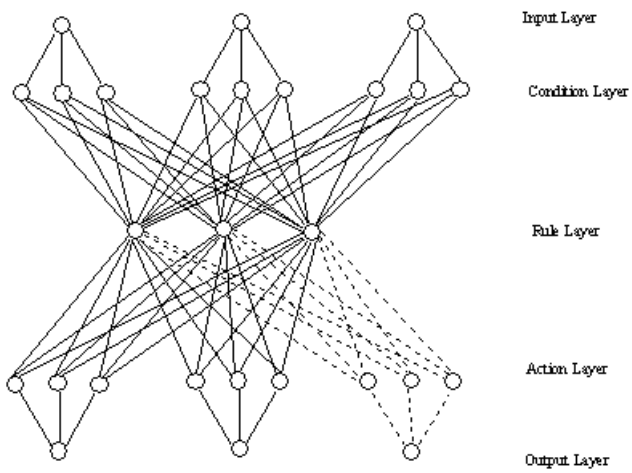

Figure 2: A simplified and exemplified diagram of an expanded EFuNN. The dotted lines represent the initial connections of the new class output node.

5.1 The EFuNN Output Expansion Algorithm

The process of adding new output nodes to EFuNN is carried out in a supervised manner. Thus, for a given input vector, a new output node will be added only if the user indicates that the given input vector is a new class. The output expansion algorithm is as follows:

FOR every new output class,

- insert a new node into the output layer.
- insert a set of fuzzy nodes according to number of MF into the action layer.

FOR every rule node r_i, $i = 1 : n$,

where n is the number of rule nodes.

- modify $W2$ the outgoing connection weights from rule nodes to action nodes by expanding $W2_{i_j}$ with fuzzified set of zero to reflect the zero output.

6 Experimental Procedure and Results

The motivation behind designing this experiment was to allow for the on-line vocabulary expansion of a small connectionist-based speech recognition system. This work is based on the ROKEL project[1]. ROKEL is an intelligent voice controlled system installed on a Pioneer DX mobile robot platform. An Evolving connectionist system is employed for spoken command recognition with an online speaker adaptation feature.

The experiments were carried out in two distinct phases. Two sets of data (one for training and one for validation) were prepared to conduct the experiments in each phase. In the first phase, speech data was prepared for the English digits from "zero" to "nine" as commands to adjust the setting of ROKEL's linear and rotational velocity. The data used in this phase was taken from 31 native speakers (22 male and 9 female) of New Zealand English. In the second phase, speech data was prepared for 5 additional commands (Go, Left, Return, Right, Stop) for controlling the basic movements of ROKEL. The data used in this phase was taken from 11 native speakers (8 male and 3 female) of New Zealand English.

Each speaker uttered each word three times. There were a total of 750 training and 240 testing utterances in the first phase and a total of 240 training and 40 testing utterances in the second phase. The testing sets were obtained from new utterances from the same speakers used for training the network. The speech was sampled at 22.05 kHz and quantised to a 16 bit signed number.

Spectral analysis of the speech signal was performed over 30 msec with Hamming window and 50% overlap, in order to extract Mel Scaled Cepstrum Coefficients (MSCC) as acoustic features [5]. A fixed-size input vector consisting of $20 \times d$ units was used, where d is the dimensionality of the feature space. Since nine MSCCs were taken as acoustic features, the Discrete Cosine Transformation (DCT) was applied over each acoustic observation and the first twenty coefficients were retained for each input vector.

An EFuNN was initialised with 180 nodes in the input layer and 10 nodes in the output layer (one for each word). Each input and output was fuzzified into 3 membership functions. These fuzzy values are the representation of the crisp values from the input and output layers in the condition and action layers respectively.

In phase one of the experiments the EFuNN was trained on the training data with the parameters shown in Table 1. The rule nodes were aggregated during the learning process.

Table 1: EFuNN's Training and Aggregation Parameters

Sensetivity Threshold	0.8
Error Threshold	0.1
Learning Rate One	0.2
Learning Rate Two	0.2
Aggregation $W1$ Threshold	0.18
Aggregation $W2$ Threshold	0.18
Number of Examples Before Aggregation	40

There were 214 rule nodes created during the training period. The trained EFuNN was then tested on both training and testing data sets. Table 2 shows the performance of the EFuNN on its training and testing sets.

Table 2: performance of EFuNN on NZ English digits

	Training Set		Testing Set	
Words	Positive Accuracy	Negative Accuracy	Positive Accuracy	Negative Accuracy
Zero	98.67	100.00	95.83	100.00
One	96.00	99.70	83.33	97.69
Two	100.00	99.41	100.00	93.06
Three	98.67	100.00	66.67	100.00
Four	98.67	99.26	91.67	97.69
Five	96.00	99.26	79.17	99.54
Six	100.00	100.00	100.00	99.07
Seven	97.33	100.00	79.17	99.07
Eight	93.33	99.85	87.50	99.54
Nine	98.67	100.00	83.33	99.54
Total	97.73	99.75	86.67	98.52

This network was then expanded to recognise 5 additional words as described in section 5. The EFuNN was trained on the new data with the same set of parameters as before. There were 114 additional rule nodes created during the learning period. The performance of the expanded EFuNN on both digits and the additional words is illustrated

in Table 3.

Table 3: performance of the expanded EFuNN on both digits and additional words

Words	Training Set		Testing Set	
	Positive Accuracy	Negative Accuracy	Positive Accuracy	Negative Accuracy
Zero	98.67	100.00	95.83	99.64
One	92.00	99.77	79.17	98.21
Two	100.00	99.20	100.00	95.00
Three	97.33	100.00	62.50	100.00
Four	98.67	99.66	91.67	98.93
Five	93.33	99.32	75.00	99.64
Six	100.00	100.00	100.00	99.29
Seven	97.33	100.00	79.17	99.29
Eight	86.67	99.89	87.50	99.64
Nine	97.33	99.89	87.50	99.64
Go	96.97	99.89	100.00	99.66
Left	98.11	98.67	100.00	98.60
Return	96.97	100.00	88.89	100.00
Right	94.00	99.78	94.44	100.00
Stop	100.00	100.00	100.00	99.66
Total	96.32	99.74	88.16	99.53

7 Discussion

The recognition rate in the first phase of the experiment are illustrated in the Table 2. An overall positive accuracy of 86.67% and overall negative accuracy of 98.52% on the testing set was achieved. Generalisation capability of the EFuNN is clearly illustrated on its unseen testing set. Further more, as shown in Table 3, this was also the case when the EFuNN was expanded to allow for recognition of an additional 5 words.

Table 3 also shows the performance of the expanded EFuNN on digits. As it can be seen from these results, the expanded EFuNN has successfully classified the additional words while maintaining the knowledge learnt during the first phase (i.e. English digits) of the experiment.

Although, the performance of the expanded EFuNN on its training set and the testing set was very good, its performance on the testing set, especially on the recognition of the word "three" and "five" , slightly degraded. This could be caused by the small size of the training set and unevenly distributed training and testing sets.

8 Conclusions and Future Research

This paper describes the structure of the EFuNN as an evolving system employing the ECoS paradigm. An application of EFuNN for isolated word recognition along with a novel idea for expanding its vocabulary, was successfully implemented. It was clearly demonstrated that EFuNN is capable of adding and identifying a new class output with training only on examples of the new class. It was also shown that EFuNN can learn and adapt to new classes while maintaining its previously learned knowledge.

Further investigation into optimisation of the aggregation and its relevance to the output expansion algorithm is currently being conducted. Further research is also planned on other applications of the presented herein methodology mainly in the area of adaptive speech recognition.

9 Acknowledgments

We would like to acknowledge the contribution to this paper of Dr Georgi Iliev for his assistance with the data processing algorithm. The ECoS methodology was developed under the FRST grants UOO808 and UOOxod6, University of Otago. Simulators can be found on the http://divcom.otago.ac.nz/infosci/kel/CBIIS.html.

References

[1] A. Ghobakhlou, Q. Song and N. Kasabov, "ROKEL: the interactively learning and navigating robot of the knowledge engineering laboratory at Otago," **ICONIP/ANZIIS/ANNES'99 Workshop, Dunedin, New Zealand, November 22-24, 57-59**, 1999.

[2] N. Kasabov, "The ECOS framework and the ECO learning method for evolving connectionist systems," **Journal of Advanced Computational Intelligence**, vol. 2, no. 6, pp. 195–202, 1998.

[3] N. Kasabov, "Evolving Fuzzy Neural Networks - Algorithms, Applications and Biological Motivation," in T. Yamakawa and G. Matsumoto (eds.), **Methodologies for the Conception, Design and Application of Soft Computing**, World Scientific Publishing Co, 1998, pp. 271–274.

[4] N. Kasabov, "On-Line Learning, Reasoning, Rule Extraction and Aggregation in Locally Optimized Evolving Fuzzy Neural Networks," **Neurocomputing, in press**, 2000.

[5] N. Kasabov and G. Iliev, "Evolving fuzzy neural networks and adaptive filtering for robust recognition of noisy speech," **IEEE Transactions on Pattern Analysis and Machine Intelligence, submitted**, 2000.

[6] N. Kasabov, J. Kim, M. Watts and A. Gray, "FuNN/2 - A Fuzzy Neural Network Architecture for Adaptive Learning and Knowledge Acquisition in Multi-modular Distributed Environments," **Information Sciences - Applications**, 1997.

[7] I. Koprinska and N. Kasabov, "An application of evolving fuzzy neural network for video parsing," in N. Kasabov and K. Ko (eds.), **ICONIP/ANZIIS/ANNES'99 Workshop, Dunedin, New Zealand, November 22-24**, 1999, pp. 96–102.

[8] J. C. Platt and N. P. Matic, "A Constructive RBF Network for Writer Adaptation," in **Advances in Neural Information Processing Systems, 9**, 1996.

[9] Q. Zhao and H. Tatsuo, "Evolutionary Learning of Nearest-Neighbour MLP," **IEEE Transactions on Neural Networks**, vol. 7, no. 3, pp. 762–767, 1996.