Evolving Fuzzy Neural Networks - Algorithms, Applications and Biological Motivation

Nikola Kasabov
Department of Information science, University of Otago,
P.O.Box 56, Dunedin, New Zealand, nkasabov@otago.ac.nz

Abstract
In the paper, the ECOS (Evolving Connectionist Systems) framework is used to develop a particular type of evolving neural networks - evolving fuzzy neural networks - EFuNNs. They can be trained in an on-line, incremental mode and are several order of magnitude faster than the ordinary neural networks trained with the backpropagation algorithm. This is illustrated on the task of adaptive phoneme recognition.

Keywords: evolving neural networks; fuzzy neural networks, adaptive phoneme recognition; eco training

1. Inroduction - The ECOS framework for evolving connectionist systems

In [5] the ECOS framework for evolving connectionist systems is presented and illustrated on two classification problems. An ECOS is a modular 'open' system that evolves over time. Initially it is a mesh of nodes (neurons) with very little connections between them, pre-defined through *prior* knowledge or 'genetic' information. These connections mainly connect modules of the initial connectionist structure. An initial set of rules can be inserted in this structure. Gradually, through self-organisation, the system becomes more and more 'wired'. The network stores different patterns (exemplars) from the training examples. A node is created and designated to represent an individual example if it is significantly different from the previous ones (with a level of differentiation set through dynamically changing parameters.

ECOS is in coherence with some theories about the functioning of the human brain [8,9,10].

2. EFuNNS - Evolving fuzzy neural networks and the EFuNN algorithm

2.1. The general FuNN architecture

Fuzzy neural networks are neural networks that realise a set of fuzzy rules and a fuzzy inference machine in a connectionist way [2,12,7]. The fuzzy neural network FuNN is a connectionist feed-forward architecture with five layers of neurons and four layers of connections [4]. The first layer of neurons receives input information. The second layer calculates the fuzzy membership degrees to which the input values belong to predefined fuzzy membership functions, e.g. small, medium, large. The third layer of neurons represents associations between the input and the output variables, fuzzy IF-THEN rules. The forth layer calculates the degrees to which output membership functions are matched by the input data, and the fifth layer does defuzzification and calculates values for the output variables. A FuNN has both the features of a neural network and a fuzzy inference machine. Several training algorithms have been developed for FuNN [4]: a modified backpropagation algorithm; a genetic algorithm; structural learning with forgetting; training and zeroing; combined modes. Several algorithms for rule extraction from FuNN have been also developed and applied. One of them (aggregated rule extraction method) represents each rule node of a trained FuNN as an IF-THEN fuzzy rule. FuNNs have several advantages when compared with the traditional NN or with the fuzzy systems: (a) They are both statistical and knowledge engineering tools; (b) They are robust to catastrophic forgetting, i.e. when a FuNN is initially trained with the structural learning with forgetting algorithm and then further trained only on new data, it keeps a reasonable memory of the old data; (c) They can work on both real input data and fuzzy input data represented as singletons (centres of the input membership functions),

2.2. EFuNNs - Evolving FuNNs

EFuNNs are FuNN structures that evolve according to the ECOS principles. All nodes in an EFuNN are created during learning. The nodes representing membership functions (fuzzy label neurons) can be modified during learning. As in FuNN, each input variable is represented here by a group of spatially arranged neurons to represent different fuzzy domain areas of this variable. For example three neurons can be used to represent "small", "medium" and "large" fuzzy values of a variable. Different membership functions can be attached to these neurons (triangular, Gaussian, etc.). New neurons evolve in this layer if for a given input vector the corresponding variable value does not belong to any of the existing membership functions to a membership degree greater than a membership threshold, e.g. 0.6. A new fuzzy label neuron or an input variable neuron can be created during the adaptation phase of an EFuNN.

2.3. The EFuNN algorithm

1. Initialise an EFuNN structure with a maximum number of neurons. Initial connections may be set through inserting fuzzy rules in a FuNN structure. FuNNs allow for insertion of fuzzy rules as an initialisation procedure thus allowing for prior information to be used prior to the evolving process. If there are no rule (case) nodes connected to the fuzzy input and fuzzy output neurons, then *create* the first node rn=1 to represent the first example EX=x_1 and set its input W1(rn) and output W2 (rn) connection weights as follows:.

<*Create 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 example EX.

2. WHILE <there are examples> DO

Enter the current, i_{th} example EX=x_i. 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.

3. Find the normalised fuzzy similarity between the new example EX (the vector of its membership degrees to the fuzzy input labels) and the already stored patterns in the case nodes j=1,2,…,rn:

Dj= sum (abs (EX - W1(j)))/ 2 / sum (W1(j))

4. Find the activation of the rule (case) nodes j, j=1:rn. Here radial basis activation function, or a linear one, can be used on the Dj input values (A1 (j) = radbas (Dj), or A1(j) = 1 - Dj)

5. Update the local parameters of the rule nodes: age, average activation.

6. Find all the case nodes j with an activation value A1(j) above a sensitivity threshold *Sthr.*

7. If there is no such case node, then <C*reate a new rule node*> using the procedure from point 1.

ELSE

8. Find the rule node *inda1* with maximum value ma*xa1.*

9. (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, zero all the activation values of case nodes that are below an activation threshold *Athr* and propagate the activation values A1 to the next neuronal layer.

10. Find the winning fuzzy output neuron *inda2* and its activation *maxa2.*

11. Find the desired winning fuzzy output neuron *indt2* and its value *maxt2.*

12. Calculate the fuzzy output error for each fuzzy output: Err=A2 - TE.

13. IF (*inda2* is different from *indt2)* or (Err (inda2) > *Errthr*) *<Create a new rule node>*

ELSE

14. Update the (a) input, and (b) the output connections of rule node k=inda1: (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

15. Prune rule nodes j and their connections that satisfy the following fuzzy pruning rule to a pre-defined level representing the current need of pruning:

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 associated with the corresponding "yes" class output nodes (for classification tasks only))

THEN the probability of pruning node (j) is HIGH

16. END of the while loop and the algorithm

17. Repeat steps 2-16 for a second presentation of the same input data or for eco training (see [5] for explanation).

3. EFuNNS for Adaptive Phoneme Recognition

Here the EFuNN algorithm is applied to the problem of phoneme recognition and phoneme adaptation.

3.1. The problem of adaptive speech recognition

Adaptive speech recognition is concerned with the development of speech recognition systems that can adapt to new speakers (of the same or a new accent); that can enlarge their database of words in an on-line mode; that can acquire new languages [1,3,6]. There are several methods that have been experimented for adaptive phoneme recognition. One of them [6] uses phoneme FuNN modules for each class phoneme trained with learning with forgetting algorithm. The adaptation to a new speaker is achieved through additional training of a phoneme FuNN on new speaker's data for a few epochs. This approach to adaptive speech recognition assumes that at the higher, word recognition level, a decision is made on which phoneme module should be adapted in order to accommodate the new speaker's data and to achieve a correct word recognition. The backpropagation (BP) algorithm was used. This method assumes a fixed number of rule nodes in the FuNNs. There are some difficulties when applying this method for on-line adaptation on continuous speech: (1) even few epochs of additional training with the use of the BP algorithm may not be fast enough; (2) in spite of the robustness of the FuNN architecture to catastrophic forgetting, a trained FuNN tends to forget old speech data if the new data differs significantly from the old one; (3) limited potential for accommodating new speech data because of the fixed size of the FuNN networks.

Here, the EFuNN algorithm are used for the purpose of phoneme adaptation of already trained EFuNNs on new accent data. In the experiments below, four EFuNNs are evolved to learn existing data on four NZ English phonemes. Recognition results are compared with the results when ordinary FuNNs are used and when GAFuNNs (FuNNs optimised by a genetic algorithm) are used [13]. Then one of the EFuNNs, the phoneme /I/ module, is further evolved to accommodate new accent data in the pronunciation of /I/ taken from a speaker of a different accent.

3.2. EFuNNS for phoneme recognition

The following phoneme data on four phonemes of New Zealand English spoken by two male and two female speakers are used in the experiment: /**I**/ (taken from the pronounced word "sit", 100 mel scale vectors, each of them consisting of 26 mel coefficients); /**e**/ (taken from "get", 170 mel scale vectors); /**ae**/ (taken from "cat", 170 vectors), and /**i**/ (taken from "see", 270 vectors). Three membership functions are used to represent "small", "medium" and "high" values for each mel-coefficient. The number of examples selected for each phoneme corresponds to the relative frequency of the appearance of this phoneme in the spoken NZ English. Phonemes /e/ and /i/ have similar average mel values which makes their differentiation more difficult.

Experiment 1. **EFuNNs trained on both positive and negative data**. Four EFuNNs are evolved from the 710 input vectors. The EFuNNs have the following characteristics: linear activation function for the case (rule) nodes; saturated linear functions for the fuzzy outputs and a linear function for the class output neurons; Sthr=0.9; Errthr=0.2; no pruning; lr=0; rn(phoneme /I/) = 361 (90 for the class phoneme - positive); rn(phoneme /e/) = 395 (90 positive); rn(phoneme /ae/) = 362 (110 positive); rn(phoneme /i/) = 393 (101 positive). The following mean sum-square error is evaluated for the four phoneme modules correspondingly: 0.0085; 0.055; 0.025; 0.145. The overall classification rate is: /I/ - 94 (94%); /e/ - 131 (77%); /ae/ - 152 (90%); /i/ - 167 (62%). The examples that have not been classified correctly have not been miss-classified either. They did not activate any of the four EFuNNs (for them all EFuNNs had zero output values). Of course this is a much better result than having a misclassification (false positive activation). Here the negative examples (that do not belong to a phoneme module) are rejected with 100% accuracy in all EFuNN modules.

Experiment 2. **Using positive phoneme data only.** The same experimental setting is used as in experiment 1, but four phoneme EFuNNs are evolved with positive data only. The EFuNNs have the following characteristics: rn(phoneme /I/) = 89; rn(phoneme /e/) = 89; rn(phoneme /ae/) = 108; rn(phoneme /i/) = 101. The overall classification rate is: /I/ - 94 (94%); /e/ - 149 (87.6%); /ae/ - 154 (90.6%); /i/ - 190 (70.3%). In contrast with experiment 1, the examples that have not been classified correctly have been miss-classified.

3.3. Comparative analysis of FuNNs, GAFuNNs and EFuNNs on the phoneme recognition task

Figures 1 and 2 show the results from the above experiments and also the results when: (1) four FuNNs, 'manually' designed and trained with a BP algorithm, are used for each of the phonemes, and (2) when four FuNNs are optimised with a GA algorithm and trained again with the BP[13].

For the FuNNs experiment four FuNNs were 'manually' created each having the following architecture: 78 inputs (3 time lags of 26 element mel vectors each), 234 condition nodes (three fuzzy membership functions per input), 10 rule nodes, two action nodes, and one output. This architecture is identical to that used for the speech recognition system described in [6]. Nine networks were created and trained for 1000 epochs for each phoneme. A bootstrap method is used for selecting statistically appropriate data sets at every 10 epochs of training the FuNNs. Each trained FuNN was recalled over the same data set, and the recall accuracy calculated. For these calculations an output activation of 0.8 or greater is taken to be a positive result, while an activation of less than 0.8 is negative. The mean classification accuracy of the manually designed FuNNs are presented in fig.1. The manually designed networks have great difficulty in correctly identifying the target phonemes, tending instead to classify all of the phonemes presented as negative examples (for the chosen classification threshold of 0.8).

For the GFuNNs experiment a population size of fifty FuNNs was used, with tournament selection, one point crossover, and a mutation rate of one in one thousand [13]. Each individual was trained with the BP algorithm for five epochs on the training data set with the learning rate and momentum set to 0.5 each. The GA was run for fifty generations, at the end of which the most fit individual was extracted and decoded. The resulting FuNN was then trained on the entire data set using the bootstrapped BP training algorithm. Each resultant network was trained for one thousand epochs, with the learning rate and momentum again set to 0.5 each, and the training data set being rebuilt every ten epochs. The GA was run nine times over each of the phonemes. The mean classification accuracy of the GA designed FuNNs is displayed in fig.1.

Overall, the best results have been obtained with the use of EFuNNs. The large number of rule nodes in the EFuNNs shows the variation between the different pronunciations of the same words by the four reference speakers. EFuNNs require four to six order of magnitude less time for training per example (see fig.2).

Experiment 3. **Sleep eco training.** The trained in experiment 2 EFuNNs on positive data, are further trained on negative data as stored in the other EFuNN modules (sleep eco training [5]). The same accuracy is achieved as in EFuNNp on positive data, but here 100% accuracy is achieved on the negative data.

	FuNN	GFuNN	EFuNN	EFuNNp
/I/	32%(98)	57% (97)	94% (100)	94% (98)
/e/	80%(94)	81% (95)	77% (100)	87% (87)
ae	52%(96)	72% (96)	90% (100)	90% (97)
/i/	5% (99)	18% (98)	62% (100)	70% (94)

Figure 1. True positive and true negative (in brackets) classification accuracy

6.4. On-line adaptation of phoneme EFuNNs on new accent data

Adaptation of a phoneme module to a new speaker's data takes place when this module is identified for on-line adaptation by the higher-level decision module according to the ECOS framework and the framework presented in [3]. Adaptation in EFuNNs is not different from its usual training (evolving) procedure. This is illustrated in the following experiment.

***Experiment 4*. Adaptation of the /I/ phoneme EFuNN**. The /I/ phoneme EFuNN that evolved in experiment 1, was tested on a new speaker's phoneme /I/ data taken from the pronounced by the new speaker word "sit". The /I/ EFuNN did not recognise any of the 10 new input vectors. The EFuNN was further evolved with the use of the 10 positive input vectors. After that, the EFuNN increased its rule nodes from 361 to 369 and recognised 9 out of 10 new input vectors.

4. Conclusions

The EFuNNs introduced in the paper have the following features: (1) Incremental, (possibly 'one shot') learning; (2) On-line adaptation; (3) 'Open' structure; (4) Learning is accomplished in the same ways for both supervised and unsupervised modes;(5) Multi-level, multi-modular, hierarchical organisation; (6) Allowing for time and space representation based on biological plausibility; (7) Rule extraction and rule insertion.

References

1. Cole, R., et al. The Challenge of Spoken Language Systems: Research Directions for the Nineties, IEEE Trans. on Speech and Audio Processing, vol.3, No.1, 1-21, 1995.
2. Jang, R. "ANFIS: adaptive network-based fuzzy inference system", IEEE Trans. on Syst.,Man, Cybernetics, 23, 665-685 (1993).
3. Kasabov, N. "A framework for intelligent conscious machines utilising fuzzy neural networks and spatial temporal maps and a case study of multilingual speech recognition", in: Amari, S. and Kasabov, N. (eds) *Brain-like computing and intelligent information systems*, Springer, 106-126 (1997)
4. Kasabov, N. *Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering*, The MIT Press, CA, MA (1996).
5. Kasabov, N. ECOS: A framework for evolving connectionist systems and the eco learning paradigm, Proc. of ICONIP'98, Kitakyushu, Oct. 1998
6. Kasabov, N., Kozma, R., Kilgour, R., Laws, M., Taylor, J., Watts, M. and Gray, A. "A Methodology for Speech Data Analysis and a Framework for Adaptive Speech Recognition Using Fuzzy Neural Networks". In: Proc. of ICONIP'97, Springer, Singapore (1997).
7. Lin, C.T. and C.S. G. Lee, "Neuro Fuzzy Systems", Prentice Hall (1996).
8. McClelland, J., B.L. McNaughton, and R.C. Reilly, "Why there are Complementary Learning Systems in the Hippocampus and Neocortx: Insights from the Successes and Failures of Connectionist Models of Learning and Memeory", CMU Technical Report PDP.CNS.94.1, March, 1994
9. Quartz, S.R., and Senowski, T.J. The neural basis of cognitive development: a constructivist manifesto, Behavioral and Brain Science, in print
10. Reed, R., "Pruning algorithms - a survey", IEEE Trans. Neural Networks, 4 (5) 740-747 (1993).
11. Sinclair, S., and Watson, C. "The Development of the Otago Speec h Database". In Kasabov, N. and Coghill, G. (Eds.), Proceedings of ANNES ’95, Los Alamitos, CA, IEEE Computer Society Press (1995).
12. Yamakawa, T., H. Kusanagi, E. Uchino and T.Miki, "A new Effective Algorithm for Neo Fuzzy Neuron Model", in: Proceedings of Fifth IFSA World Congress, 1017-1020 (1993)
13. Watts, M., and Kasabov, N. Genetic algorithms for the design of fuzzy neural networks, in Proc. of ICONIP'98, Kitakyushu, Oct. 1998

	FuNN	GAFuNN	EFuNN	EFuNNpos.examp.	EFuNNsleep-eco
/I/	2596/18.10^7	616/10.10^{10}	28960/60.10^3	7200/14.10^3	14000/30.10^3
/e/	2596/18.10^7	1045/14.10^{10}	31680/62.10^3	7200/14.10^3	14000/30.10^3
/ae/	2596/18.10^7	847/11.10^{10}	29040/61.10^3	8720/15.10^3	17000/35.10^3
/i/	2596/18.10^7	946/12.10^{10}	31520/62.10^3	8160/15.10^3	16000/34.10^3

Figure 2. The size of the NN in number of connections and / the approximate time for training per example (in relative units, representing the number and the complexity of the calculations)