# GENETIC ALGORITHMS FOR THE DESIGN OF FUZZY NEURAL NETWORKS

*Michael Watts and Nikola Kasabov*

*Email: mjwatts@sol.otago.ac.nz, nkasabov@otago.ac.nz*
Department of Information science, University of Otago
P.O.Box 56, Dunedin, New Zealand

## ABSTRACT

The paper presents a methodology for designing the structure of a fuzzy neural network in a multi-modular connectionist system for classification purposes and illustrates the methodology on the task of phoneme recognition of the 43 phonemes in New Zealand English. The results show that by using this methodology the recognition rate can be improved significantly when compared to the recognition rate of the same modules designed manually.

## 1. INTRODUCTION

Choosing the optimum number of input nodes (features), connections and neurons is a major design issue when a neural network structure is created for a particular task [1], [2]. It is especially important to have optimally designed structures of individual neural network modules when a multi-modular approach is used and the system consists of many neural networks each of them performing one-class classification. This is the case of the phoneme based speech recognition task when each phoneme is recognised by a separate, dedicated module [3]. Optimal network modules have smaller number of parameters, work faster and are usually better when trained on new data (adaptation). Here a methodology for designing a fuzzy neural network FuNN [1] based on using genetic algorithms (GA) is presented and illustrated on the task of recognising the 43 phonemes in New Zealand English.

## 2. FUZZY NEURAL NETWORKS - FUNN

Fuzzy neural networks are neural networks that realise a set of fuzzy rules and a fuzzy inference machine in a connectionist manner. The fuzzy neural network FuNN is a connectionist feed-forward architecture with five la yers of neurons and four layers of connections (see fig.1). The first layer of neurons receives the 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, or large. The third layer of neurons represents associations between the input and the output variables, fuzzy rules. The fourth 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 m achine. Several training algorithms have been developed for FuNN:

(a) A modified back-propagation (BP) algorithm that does not change the input and the output connections representing the membe rship functions.
(b) A modified BP algorithm that utilises structural learning with forgetting, i.e. a small forgetting ingr edient, e.g. $10^{-5}$, is used when the connection weights are updated [12].
(c) A modified BP algorithm that updates both the i nner connection layers and the membership layers. This is possible when the derivatives are calculated sep arately for the two parts of the triangular membership functions which form a non-monotonic activation function of the neurons in the condition el ement layer.
(d) a GA algorithm;
(e) A combination of any of the methods above used in different time intervals as part of a single training procedure.

Several algorithms for rule extraction from FuNN have been developed and applied. One of them represents each rule node of a trained FuNN as an IF-THEN fuzzy rule.

FuNNs have several advantages when compared with the traditional connectionist sy stems or with the fuzzy systems:

(a) They are both statistical and knowledge enginee ring tools.
(b) They are robust to catastrophic forgetting, i.e. when further trained only on new data, they keep a reasonable memory of the old data.
(c) They can be used as replicators, where same input data is used as output data during training; in this case the rule nodes perform an optimal encoding of the i nput space.
(d) They can work on both real input data and fuzzy input data represented as singletons (centres of gravity of the input membership fun ctions))
(e) They are appropriate tools to build multi-modular IIS as explained in [1],[2],[3].

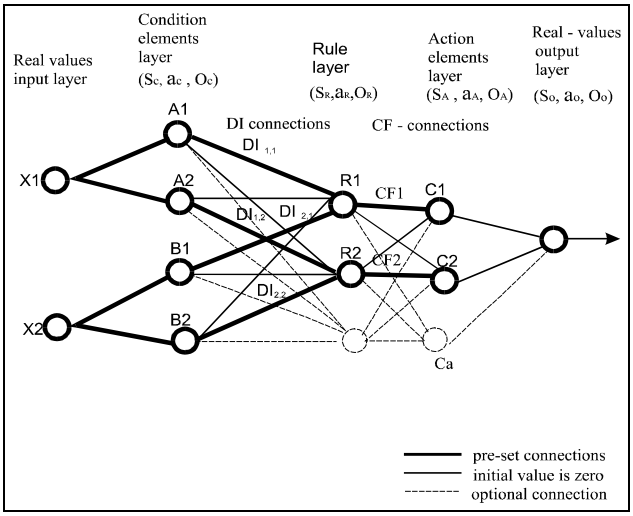


Figure 1 **:** A FuNN structure for two initial fuzzy rules: R1: IF x1 is A1 (DI1,1) and x2 is B1 (DI2,1) THEN y is C1 (CF1); R2: IF x1 is A2 (DI1,2) and x2 is B2 (DI2,2) THEN y is C2 (CF2), where DIs are degrees of importance attached to the condition elements and CFs are confidence factors attached to the consequent parts of the rules (adopted from [1]). The triplets (s,a,o) represent specific for the layer summation, activation, and output functions.

## 3. GENETIC ALGORITHMS

Genetic algorithms (GAs) are search and optimisation algorithms inspired by the mechanics of Darwinian selection and biological evolution [4],[5]. GAs have been applied to the optimisation of neural networks, e.g.: the topology of the network [6]; the network control parameters [7]; the connection weights [8]. The GAs used in this research are based upon the Goldberg's Simple Genetic Algorithm [4] and its software implementation [9]. The algorithm differs from Goldberg's in the following aspects:

- real valued genes of type boolean, integer or floating point are used
- crossover is always performed, with mutation being performed on the child
- mutation is implemented as a re-initialisation of the gene
- one child per crossover is generated

## 4. GA-DESIGN OF FUZZY NEURAL NETWORKS

The general form of the design algorithm used here was introduced in [10]. Briefly stated, it uses a GA to select the relevant features of the data set, while simultaneously configuring the network by selecting the number of membership functions to attach to each input and output node, and the number of rule nodes to include in the hidden layer. To evaluate each individual (a FuNN structure), a chromosome is decoded into FuNN and the network trained on, and recalled with the supplied sets of training and recall data. The fitness of the individual is calculated according to the following formula:

$$f = w_i \times \tanh\frac{1}{i} + w_t \times \tanh\frac{1}{e_t} + w_r \times \tanh\frac{1}{e_r}$$

where: $f$ is the fitness of the individual, $w_i$ is the weighting factor applied to the input component of the evaluation function, $i$ is the number of inputs in the network being evaluated, $w_t$ is the weighting factor applied to the training error component of the evaluation function, $e_t$ is the training error of the network, $w_r$ is the weighting factor applied to the recall error component of the evaluation function, and $e_r$ is the recall error of the network.

The *tanh* function is applied to each term of the fitness function to give each component the same significance in evaluation. The three weighting factors are adjustable to allow the user to direct the GA towards a particular fitness measure, e.g. to minimise the number of inputs in the network.

## 5. EXPERIMENTS ON PHONEME RECOGNITION DATA

The data used for the experiments was taken from the Otago Speech Corpus [11] and consisted of isolated phonemes spoken by one male and one female speaker of New Zealand English. For each phoneme, two subsets of data were extracted and used as training and recall data for the GA. For these experiments a population size of fifty FuNNs was used, with tournament selection, one point crossover, and a mutation rate of one in one thousand. 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. Bootstrapped training is useful when there is a large number of examples of some data classes but relatively few of others. With Bootstrapped training a new training set is built regularly after a certain number of training iterations with examples for each class taken from regular data intervals in a specified proportion. The network is then trained with the BP training algorithm on the generated training set. In this experiment two classes of data were used, one representing the target phoneme, the other all other phonemes. 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 43 phonemes available.

In order to compare the performance of the GA designed FuNNs with those of manually designed networks, FuNNs were created each having the following architecture: 78 inputs, 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 [3]. Nine networks were created and trained for each phoneme. The training parameters were identical to those used to train the GA designed networks. 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 results of these experiments are presented in tables 1 and 2.

The mean classification accuracies of the manually designed FuNNs are presented in table one. As may be seen from the table, 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). The mean classification accuracies of the GA designed FuNNs are displayed in table two. In addition to the classification accuracy, the table also presents the mean number of inputs selected automatically by the GA for each phoneme. Comparison of the two tables shows that even though the genetically designed FuNNs utilised a much smaller number of inputs, their mean ability to correctly classify the target phoneme exceeded that of the manually designed FuNNs forty one out of forty three times. Also, the mean ability of the GA designed FuNNs to reject phonemes that were not the target was only slightly inferior to that of the manually designed FuNNs.

## 6. CONCLUSION

The use of GA in the process of designing a neural network structure is applied here on fuzzy neural networks FuNN and illustrated on phoneme classification. The proposed method can be used for practical applications in pattern recognition and classification tasks, such as phoneme-based speech recognition and image recognition.

## REFERENCES


[1] Kasabov, N. *Foundations of Neural Networks, Fuzzy Systems and Knowledge Enginee ring*, The MIT Press, CA, MA (1996).

[2] Kasabov, N. "A framework for intell igent conscious machines utilising fuzzy neural ne tworks and spatial temporal maps and a case study of multilingual speech recognition", in: Amari, S. and K asabov, N. (eds) *Brain-like computing and intelligent information systems* , Springer, 106-126 (1997)

[3] 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: Kasabov, N. et al (eds.) Progress in Connectionist-Based Information Systems, (Proc. of ICONIP'97), Springer, Singapore (1997).

[4] Goldberg, D.E., *Genetic Algorithms in Search, Optimisation and Machine Learning* , Addison-Wesley (1989)

[5] Mitchell, Melanie, *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, Massachusetts (1996).

[6] Schiffman, W., Joost, M. and Werner, R. “Application of Genetic Algorithms to the Construction of Topologies for Multilayer Perceptrons”. In: Albrecht, R.F., Reeves, C.R., Steele, N.C, (Eds.), *Artificial Neural Nets and Genetic Algorithms* , Springer-Verlag Wien, New York (1993)

[7] Choi, B. and Bluff, K. “Genetic Optimisation of Control Parameter of a Neural Network”. In: Kasabov, N and Coghill, G. (Eds.), Proceedings of ANNES ’95, Los Alamitos, CA: IEEE Computer Society Press (1995).

[8] Fukuda, T., Komata, Y., and Arakawa, T. "Recurrent Neural Networks with Self-Adaptive GAs for Biped Locomotion Robot", In: *Proceedings of the International Conference on Neural Networks ICNN'97* , IEEE Press (1997)

[9] Ward, R., Purvis, M., Raykov, R., Zhang, F., and Watts, M. “An Architecture for Distributed Connectionist Computation” (in citation [3]),

[10] Kasabov, N., Watts, M. "Genetic algorithms for structural optimisation, dynamic adaptation and automated design of fuzzy neural networks". In: Proceedings of the International Conference on Neural Networks ICNN'97, IEEE Press, Houston (1997)

[11] Sinclair, S., and Watson, C. "The Development of the Otago Speech Database". In Kasabov, N. and Coghill, G. (Eds.), Proceedings of ANNES ’95, Los Alamitos, CA, IEEE Computer Society Press (1995).

[12] Ishikawa, M., "Structural Learning with Forgetting", Neural Networks 9, 501-521 (1996).

Table 1: Accuracy of manually designed FuNNs

| Phoneme (from word) | # | Mean Posit. Correct | Mean negative Correct |
|---|---|---|---|
| /p/ pin | 01 | 41.32 | 98.95 |
| /b/ bay | 02 | 8.64 | 99.08 |
| /t/ toy | 03 | 14.42 | 98.35 |
| /d/ die | 04 | 0 | 100 |
| /k/ key | 05 | 0 | 100 |
| /g/ get | 06 | 0 | 100 |
| /f/ five | 07 | 0 | 100 |
| /v/ van | 08 | 0 | 100 |
| /T/ thick | 09 | 0 | 100 |
| /D/ then | 10 | 0 | 100 |
| /s/ see | 11 | 43.75 | 98.84 |
| /z/ zink | 12 | 0 | 100 |
| /S/ ship | 13 | 92.2 | 97.08 |
| /Z/measure | 14 | 58.68 | 94.57 |
| /h/ he | 15 | 0 | 100 |
| /ch/ chin | 16 | 41.35 | 96.88 |
| /dj/ jam | 17 | 20.98 | 98.18 |
| /m/ me | 18 | 15.92 | 98.86 |
| /n/ not | 19 | 0 | 100 |
| /N/ sing | 20 | 0 | 100 |
| /l/ light | 21 | 0 | 100 |
| /r/ ring | 22 | 45.91 | 99.03 |
| /w/ win | 23 | 0 | 100 |
| /ie/ yes | 24 | 0 | 100 |
| /I/ sit | 25 | 32.15 | 98.24 |
| /e/ get | 26 | 80.46 | 94.19 |
| /&/ cat | 27 | 52.17 | 96.75 |
| /V/ hut | 28 | 41.25 | 95.54 |
| /A/ hot | 29 | 48.06 | 97.4 |
| /U/ put | 30 | 40.04 | 95.04 |
| /i/ see | 31 | 5.24 | 99.1 |
| /a/ father | 32 | 35.96 | 97 |
| /O/ sort | 33 | 53.82 | 96.05 |
| /3/ bird | 34 | 52.46 | 94.67 |
| /u/ too | 35 | 0.85 | 99.95 |
| /eI/ day | 36 | 5.72 | 99.31 |
| /aI/ fly | 37 | 22.38 | 97.01 |
| /Oi/ boy | 38 | 13.33 | 97.86 |
| /OU/ go | 39 | 2.53 | 99.71 |
| /aU/ cow | 40 | 0 | 100 |
| /i@/ ear | 41 | 0 | 100 |
| /U@/ tour | 42 | 0 | 100 |
| /e@/ air | 43 | 0 | 100 |

Table2: Accuracy of GA- designed FuNNs

| Phoneme # | Mean Posit. Correct | Mean Negat. Correct | # of Inputs |
|---|---|---|---|
| 01 | 61.01 | 97.05 | 21 |
| 02 | 23.45 | 96.58 | 14 |
| 03 | 58.37 | 97.98 | 23 |
| 04 | 26.49 | 98.34 | 21 |
| 05 | 38.88 | 98.34 | 20 |
| 06 | 31.62 | 98.75 | 22 |
| 07 | 69.99 | 95.6 | 17 |
| 08 | 27.93 | 97.27 | 17 |
| 09 | 67.2 | 95.99 | 16 |
| 10 | 54.97 | 95.27 | 18 |
| 11 | 89.01 | 97.89 | 28 |
| 12 | 74.26 | 97.51 | 32 |
| 13 | 91.04 | 97.32 | 28 |
| 14 | 44.44 | 97.55 | 28 |
| 15 | 69.88 | 95.92 | 18 |
| 16 | 84.15 | 95.94 | 28 |
| 17 | 73.45 | 95.78 | 26 |
| 18 | 61.29 | 96.26 | 22 |
| 19 | 57.28 | 96.66 | 20 |
| 20 | 26.32 | 97.92 | 26 |
| 21 | 39.89 | 96 | 23 |
| 22 | 66.87 | 97.47 | 19 |
| 23 | 20.13 | 98.29 | 16 |
| 24 | 6.49 | 97.81 | 11 |
| 25 | 57.38 | 97.15 | 18 |
| 26 | 81.89 | 95.19 | 31 |
| 27 | 72.94 | 96.78 | 25 |
| 28 | 52.22 | 95.94 | 24 |
| 29 | 80.62 | 96.39 | 32 |
| 30 | 43.28 | 97.42 | 24 |
| 31 | 18.37 | 98.4 | 16 |
| 32 | 73.02 | 95.68 | 34 |
| 33 | 78.04 | 94.96 | 20 |
| 34 | 52.84 | 96.25 | 19 |
| 35 | 30.42 | 97.86 | 15 |
| 36 | 22.12 | 97.36 | 19 |
| 37 | 40.76 | 86.26 | 21 |
| 38 | 28.2 | 96.64 | 16 |
| 39 | 13.03 | 98.58 | 13 |
| 40 | 18.58 | 97.47 | 19 |
| 41 | 15.31 | 98.25 | 15 |
| 42 | 12.99 | 98.14 | 15 |
| 43 | 14.52 | 98.34 | 15 |