Simple Evolving Connectionist Systems and Experiments on Isolated Phoneme Recognition

Michael Watts and Nik Kasabov
Department of Information Science
University of Otago
PO Box 56
Dunedin
New Zealand
EMail: mike@kel.otago.ac.nz, nkasabov@otago.ac.nz
Telephone: +64-03-479-5744, +64-03-479-8319, fax:+64-03-479-8311

Abstract- Evolving connectionist systems (ECoS) are systems that evolve their structure through on-line, adaptive learning from incoming data. This paradigm complements the paradigm of evolutionary computation based on population-based search and optimisation of individual systems through generations of populations. The paper presents the theory and architecture of a simple evolving system called SECoS that evolves through one pass learning from incoming data. A case study of multi-modular SECoS systems evolved from a database of New Zealand English phonemes is used as an illustration of the method.

1 Introduction: Evolving versus Evolutionary

Evolutionary computation (EC) is concerned with population-based search and optimisation of individual systems through generations of populations [3, 4]. EC is applied to the optimisation of different structures and processes, one of them being the connectionist structures and connectionist learning processes [1, 2, 14, 13]. EC, and the genetic algorithm (GA) in particular, include in principle a stage of development of the individual systems, so that a system develops, evolves through interaction with the environment that is also based on the genetic material embodied in the system. This process of development has been in many cases ignored or neglected as insignificant from the point of view of the long process of generating hundreds or thousands of generations each of them containing hundreds or thousands of individuals.

Evolving connectionist systems (ECoS) as described in [5] deal with the process of interactive, on-line adaptive learning of a single system that evolves from the incoming data. The system can either have its parameters (genes) pre-defined [5, 7, 8, 6], or self-optimised during the learning process starting from some initial values [9].

There are several ways in which EC and ECoS can be inter-linked. For example, it is possible to use EC to optimise the parameters of an ECoS at a certain time of their operation, or to use the methods of ECoS for the development of the individual systems (individuals) as part of the global EC process.

This paper presents a simplified model called SECoS derived from the general ECoS framework [5, 7, 6, 8, 9] and illustrates the model on the problem of classifying isolated phoneme data.

2 Evolving Connectionist Systems

2.1 General ECoS Principles

ECoS are systems that evolve in time through interaction with the environment, i.e. an ECoS adjusts its structure with a reference to the environment [5]. ECoS are multi-level, multi-modular structures where many modules have inter-connections, and intra-connections. The evolving connectionist system does not have a clear multi-layer structure. It has a modular open structure. The functioning of the ECoS is based on the following general principles:

1. ECoS learn fast from a large amount of data through one-pass training;
2. ECoS adapt in an on-line mode where new data is incrementally accommodated;
3. ECoS memorise data exemplars for a further refinement, or for information retrieval;
4. ECoS learn and improve through active interaction with other systems and with the environment in a multi-modular, hierarchical fashion;

One implementation of the ECoS paradigm is the Evolving Fuzzy Neural Network (EFuNN). This is a five neuron-layer network that repesents a fuzzy system that can be trained with the ECoS principles. For more on EFuNN, see [7, 8].

The next section introduces the Simple Evolving Connectionist System SECoS, while section 5 applies the model to isolated phoneme data classification similar to the approach in [10] with the use of the Otago Speech Corpus [12].

3 Comparison of ECoS with RAN

At first glance, the ECoS paradigm may seem very similar to the Resource-allocating Network (RAN) proposed by Platt [11]. Indeed, the two networks do have many features in common, as listed below:

- useful for online learning
- encapsulate in the network regions of input space
- scale sublinearly with the number of training examples
- add new units to represent novel examples
- use novelty of input vector as criteria for adding units
- use output error as criteria for adding units
- adjust network parameters when new units are not added
- use simple gradient descent to adjust network parameters
- units initially memorise examples and are later adjusted
- use multiply and sum operations on outputs

However, closer inspection reveals numerous differences between the two paradigms, These differences include:

- RAN uses gaussian functions to represent a region of input space, where the region is defined by parameters of gaussian functions
- ECoS defines a point in input space, where the point is defined by the parameters of a node in the evolving layer
- RAN is therefore a more complex system, as it has more parameters to adjust and requires more complex calculations
- RAN performs an exponential post-processing on the output values of the inputs
- RAN has a "bias" function attached to the output layer, which is adjusted to perform the function mapping
- RAN has a "resolution" parameter that determines how finely the RAN matches the function being learned. This parameter decays as learning progresses, which calls into question the use of RAN for continuous learning. Decay indicates that the training is going to stop at some point.
- ECoS is based upon continual learning, where training never stops
- RAN is not a one-pass learning algorithm

4 Simple Evolving Connectionist Systems

4.1 The Architecture of SECoS

The Simple Evolving Connectionist System, or SECoS, is a minimalist implementation of the ECoS principles. It consists of three layers of neurons. The first is the input layer. The second, hidden layer is the layer which evolves, and is the equivalent of the rule layer in EFuNNs. The activation of the nodes in this layer is determined as in Equation 1.

$$A_n = 1 - D \tag{1}$$

where A_n is the activation of the node n, and D is the distance between the incoming weight vector of n and the input vector I. Since D must be in the range $[0, 1]$, the distance measure used in SECoS is the normalised distance between two vectors, as calculated according to Equation 2. Here N is the number of input nodes and W is the input to hidden layer connection weight matrix.

$$D_{n,I} = \frac{\sum_{i}^{N} \mid W_{i,n} - I_i \mid}{\sum_{i}^{N} \mid W_{i,n} + I_i \mid} \tag{2}$$

The third layer of neurons is the output layer. Here the activation is a simple multiply and sum operation over the hidden layer activations and the hidden to output layer connection weights. Saturated linear activation functions are applied to the hidden and output layers, and input values are expected to be normalised between 0 and 1.

Propagation of hidden layer activations is done by two methods. In the first, known as One-of-N propagation, only the most highly activated hidden node has its activation value propagated to the output layer. In the second, known as Many-of-N propagation, only those nodes that are activated above the activation threshold A_{thr} have their activation values propagated to the output layer.

4.2 The SECoS Learning Algorithm

The learning algorithm is as follows.

- propagate the input vector I through the network
- IF the maximum activation a_{max} is less than the sensitivity threshold S_{thr}
  - add a node
- ELSE
  - evaluate the errors 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 connection to the winning hidden node
- repeat for each training vector

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

The incoming weights to the winning node j are modified according to Equation 3, 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 outgoing weights from node j are modified according to Equation 4, where $W_{j,o}^t$ is the connection weight from j to output o at time t, $W_{j,o}^{t+1}$ is the connection weight from j to o at time $t+1$, η_2 is the learning rate two parameter, A_j is the activation of j, and E_o is the error at o.

$$W_{i,j}^{t+1} = W_{i,j}^t + \eta_1(I_i - W_{i,j}^t) \tag{3}$$

$$W_{j,o}^{t+1} = W_{j,o}^t + \eta_2(A_j \times E_o) \tag{4}$$

4.3 Spatial Allocation of Exemplar Nodes

Allocating new exemplar nodes in the hidden layer to specific positions has two advantages: firstly, it allows the SECoS to act as a one dimensional vector quantiser, mapping spatially similar examples into spatially near groups in the hidden layer. Secondly, it allows the hidden layer nodes to be aggregated, decreasing the size of the hidden layer.

Strategies investigated are:

- linear allocation, where new nodes are added at the end of the hidden layer
- maximum activation clustering, where a new node is inserted adjacent to the winning hidden node,
- minimum distance, where the new node is allocated next to that hidden node whose output weight vector is spatially closest to the desired output vector.

Linear allocation has the disadvantage that no spatial meaning is assigned to the hidden nodes, and is used only when no such meaning is desired. Maximum activation clustering has the disadvantage that it groups nodes only according to the input spatial similarity: if the hidden nodes are aggregated then nodes that represent multiple classes may be combined, destroying the networks ability to differentiate between those classes. Minimum distance clustering ignores the input spatial similarity and groups according to output spatial similarity. This is far more useful for aggregation, because spatially nearby nodes, when aggregated, represent the same classes. Thus, aggregation is less likely to destroy the discriminatory ability of the network.

4.4 Hidden Layer Node Aggregation

Hidden layer node aggregation is the process of combining several adjacent nodes into one node that represents all of the previous exemplars for that spatial region. During the aggregation process, the distance between the incoming and outgoing weight vectors of two nodes is calculated. If the distances are below specified thresholds, the two nodes are either aggregated together, or added to a set of nodes that are all aggregated into one. The first methodology is called pair-wise aggregation, as it only aggregates nodes one pair at a time. The second is called group-wise aggregation, as it aggregates an entire group at a time. The rationale behind aggregation is to reduce the size of the hidden layer of the SECoS, while retaining the knowledge stored within the connections to each node. The incoming distance D^{in} between two nodes m and n is measured according to Equation 5, where I is the number of input nodes. The outgoing distance D^{out} between m and n is measured according to Equation 6, where O is the number of output nodes.

$$D_{m,n}^{in} = \frac{\sum_{i}^{I} \mid W_{i,m} - W_{i,n} \mid}{\sum_{i}^{I} \mid W_{i,m} + W_{i,n} \mid} \tag{5}$$

$$D_{m,n}^{out} = \frac{\sum_{o}^{O} \mid W_{m,o} - W_{n,o} \mid}{\sum_{o}^{O} \mid W_{m,o} + W_{n,o} \mid} \tag{6}$$

During the aggregation process, the incoming and outgoing weight vectors W_{in} and W_{out} are calculated according to Equations 7 and 8, where A_n is the set of nodes being aggregated.

$$W_{in} = \overline{(W_{in}(A_n))} \tag{7}$$

$$W_{out} = \overline{(W_{out}(A_n))} \tag{8}$$

5 Experiments

5.1 Isolated Phoneme Recognition

The learning and generalisation abilities of the SECoS model are demonstrated here on the problem of isolated phoneme recognition. This problem was chosen for several reasons: the problem is well characterised; Copious amounts of data exist and are readily available; The data is complex, yet well understood. The data used was taken from the Otago Speech

Corpus [12] (http://kel.otago.ac.nz/hyspeech/corpus/). This is a body of segmented words recorded from native speakers of New Zealand English, and covers all 45 phonemes present in the dialect. A subset of only four speakers was used here, two males and two females, and examples of 43 phonemes were used in the experimental data sets.

5.2 Experimental Data Sets

Three data sets were assembled, using data from four speakers. Sets A and B consist of data from the first two (one male, one female) speakers, and set C consists of data from the second two (also one male, one female). There were 10175 examples in set A, 4955 examples in set B, and 7058 examples in set C. Each row was a 78 element vector, consisting of a three time step mel-scale transformation of the raw speech signal. The values were linearly normalised to be between 0 and 1.

5.3 Experimental Design

For both of the experiments described here, the network had 78 input nodes (one for each feature) and a single output node. Each network was trained to recognise a single phoneme. For the first experiment, each network was first evolved over set A, then recalled on set A, B and C. The network was then further trained on set B, and again recalled over A, B and C. Finally, the network was further trained over set C and again recalled over all three sets. The purpose of this process is as follows: by training over set A and recalling with A, B and C, it is possible to determine how well the network memorises the training data as well as how well it generalises to new data. By further training on set B and testing over all three data sets, it becomes possible to see how well the network avoids the problem of catastrophic forgetting (by evaluating its accuracy over set A), how well it adapts to new data (by evaluating its accuracy over set B) and how much this affects its generalisation capability (by evaluating it over set C). Further training over set C allows investigation of how well the network adapts to new speakers and of how well it remembers the old. While experiments were carried out over all 43 phonemes, for brevity only three phonemes are presented here. These phonemes are presented in Table 1. Each of these phonemes are vowels, which are classically difficult to classify accurately. Also, they are quite long, which means there are more examples available for them than for the shorter phonemes.

Table 1: Exemplar phonemes

ASCII Character	Example Word
/I/	p**i**t
/e/	p**e**t
/&/	p**a**t

The second experiment was carried out in a similar fashion to the first. The difference is that after each training cycle, the network hidden layer was aggregated via the groupwise aggregation strategy with an incoming and outgoing distance threshold of 0.6. The aggregated network was recalled across each data set after each aggregation operation.

The training parameters used in both experiments are displayed in Table 2.

Table 2: Training parameters

Recall Method	One of N
Sensitivity Threshold	0.5
Error Threshold	0.1
Learning Rate One	0.9
Learning Rate Two	0.9

5.4 Experiment One Results

A plot of the size of the hidden layer against the training examples for phoneme /I/ is presented in Figure 1.

The true positive percentage accuracies (examples of the target phoneme successfully classified as such by the network) are presented in Tables 3, 5 and 7. The true negative percentage accuracies (examples that are not the target phoneme successfully classified as such by the network) are presented in Tables 4, 6 and 8. Each table row corresponds to the SECoS network after one training cycle.

After training on set A each network displays good positive classification generalisation over new examples from the same two speakers, with the generalisation over the new speakers for phoneme /&/ being quite good also. Rejection of non-target phonemes is also consistently high. Each network additionally displays excellent adaptation to positive examples after additional training, with only minor levels of forgetting in classification of both negative and positive examples. Of concern is the low true positive accuracy over data set C after training on A and B, for phoneme /I/ and /e/. Also of concern is the large size of the hidden layer of all networks at the conclusion of training, with a mean of one node every seventeen training examples. This, along with basic speaker variations, most probably accounts for the poor generalisation performance over set C. The large size of the hidden layer also causes efficiency problems, as there is obviously a huge number of calculations required for each recall operation. These concerns are the motivation for the second experiment, which features aggregation of the hidden layer.

Table 3: True positive accuracies for phoneme /I/

	Recall Data			
Training Data	A	B	C	Hidden Nodes
A	100	89.412	41.739	661
B	100	100	34.783	909
C	100	100	100	1091

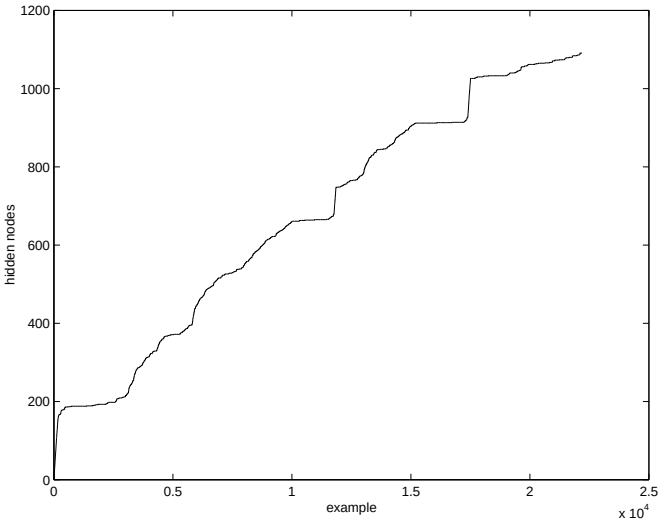

Figure 1: Plot of hidden layer size against training example for phoneme /I/

Table 4: True negative accuracies for phoneme /I/

	Recall Data			
Training Data	A	B	C	Hidden Nodes
A	96.3	96.181	91.891	661
B	96.61	97.495	93.605	909
C	95.25	95.4	92.93	1091

Table 5: True positive accuracies for phoneme /e/

	Recall Data			
Training Data	A	B	C	Hidden Nodes
A	99.21	78.226	35.981	633
B	99.605	99.194	42.523	872
C	99.605	99.19	100	1129

Table 6: True negative accuracies for phoneme /e/

	Recall Data			
Training Data	A	B	C	Hidden Nodes
A	97.42	97	97	633
B	97.21	97.87	96.74	872
C	94.99	95.05	95.47	1129

Table 7: True positive accuracies for phoneme /&/

	Recall Data			
Training Data	A	B	C	Hidden Nodes
A	99.65	92.75	71.05	729
B	99.3	100	67.89	1010
C	99.3	100	100	1204

5.5 Experiment Two Results

A plot of the size of the hidden layer against the training examples for phoneme `/I/` is presented in Figure 2. The three sharp drops in size correspond to the three aggregation operations. The true positive accuracies are presented in tables 9, 11 and 13. The true negative accuracies are presented in tables 10, 12 and 14. Each pair of rows represents the SECoS network after firstly a training operation, then an aggregation operation.

Table 8: True negative accuracies for phoneme /&/

	Recall Data			
Training Data	A	B	C	Hidden Nodes
A	96.9	95.95	92.87	729
B	96.48	97.05	94.73	1010
C	94.82	94.561	94.467	1204

It can be seen that aggregation does indeed increase generalisation accuracy over set C after training with set A across all the phonemes. After additional training on set B this increase is also apparent. Accuracy over previously seen examples decreases dramatically after aggregation for some phonemes. The widely disparate results between phonemes strongly suggests that the SECoS model is quite sensitive to both training and aggregation parameters.

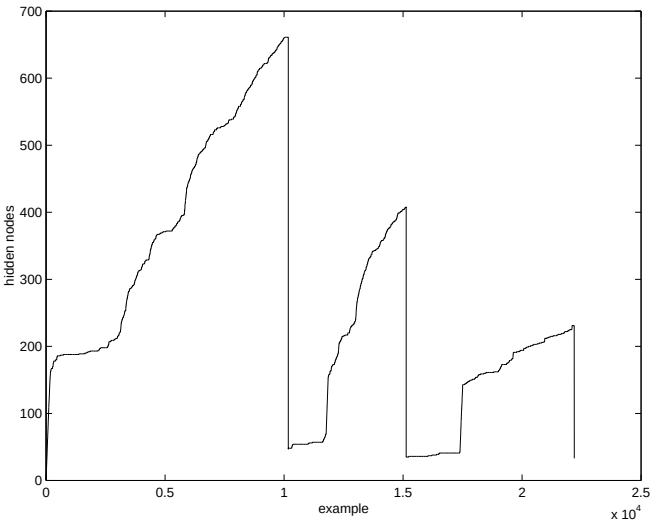

Figure 2: Plot of hidden layer size against training example for phoneme /I/

Table 9: True positive accuracies for phoneme /I/ pre and post aggregation

	Recall Data			
Training Data	A	B	C	Hidden Nodes
A	100	89.412	41.74	661
	61.494	63.53	79.13	47
B	83.91	100	26.09	408
	42.53	50.59	33.04	35
C	67.24	74.12	100	231
	21.84	25.88	80.87	33

Table 10: True negative accuracies for phoneme /I/ pre and post aggregation

Training Data	Recall Data			
	A	B	C	Hidden Nodes
A	96.3	96.18	91.89	661
	89.06	90.62	85.99	47
B	95.31	95.81	93.69	408
	96.4	96.69	94.92	35
C	78.58	80.78	80.54	231
	92.68	93.16	92.28	33

Table 11: True positive accuracies for phoneme /e/ pre and post aggregation

Training Data	Recall Data			
	A	B	C	Hidden Nodes
A	99.21	78.23	35.98	632
	63.64	52.42	20.09	43
B	79.45	98.39	40.19	370
	56.92	45.97	29.44	32
C	86.56	85.48	100	308
	93.28	89.52	79.44	26

Table 12: True negative accuracies for phoneme /e/ pre and post aggregation

Training Data	Recall Data			
	A	B	C	Hidden Nodes
A	97.42	97	97	632
	95.26	96.42	98.38	43
B	95.74	96.96	95.88	370
	95.15	96.27	98.47	32
C	77.62	79.13	89.68	308
	75.79	78.84	77.64	26

Table 13: True positive accuracies for phoneme /&/ pre and post aggregation

Training Data	Recall Data			
	A	B	C	Hidden Nodes
A	99.3	92.03	71.05	729
	62.46	55.8	75.26	63
B	86.32	100	84.21	447
	71.93	60.14	82.11	46
C	86.67	79.71	100	263
	63.86	51.45	76.32	21

5.6 Comparison with Multi-layer Perceptron

In order that a meaningful comparison can be made between the SECoS architecture and more traditional connectionist structures, multi-layer perceptrons were created, trained and tested on the same datasets as the SECoS. Each of the MLPs had ten hidden nodes, and was trained using the bootstrapped backpropagation algorithm. The ratio of positive and negative training examples was set at 1:3, with each non-target phoneme being represented equally in the negative training set. Each network was trained for a total of one thousand epochs, with the training set being rebuilt every ten epochs. The learning rate and momentum were both set to 0.5.

Table 14: True negative accuracies for phoneme /&/ pre and post aggregation

Training Data	Recall Data			
	A	B	C	Hidden Nodes
A	96.91	95.95	92.87	729
	97	97.36	95.08	63
B	95.06	95.2	94.63	447
	89.29	90.51	92.84	46
C	75.6	78.64	86.43	263
	85.08	87.61	92.94	21

The true positive accuracies are displayed in Tables 15, 17 and 19. The true negative accuracies are presented in Tables 16, 18 and 20. It can be seen from these results that while the MLPs are able to adapt to the new positive examples (at some cost of forgetting) they consistently lose the ability to reject non-target examples. Although the trained MLP are smaller, and therefore faster, than the equivalent SECoS, the loss of their ability to reject non-target phonemes severely limits their usefulness. This is especially evidenced by the performance of the `/e/` network in Table 18 after training on data set C. Here, all data sets had a true negative accuracy of zero: plainly, the MLP has completely lost the ability to discriminate between phonemes, and simply classifies every example as the target.

Although these results may be improved though careful selection of the network architecture, training examples and training parameters, this would be a very involved process: the degree of variation between phonemes means that selection of the optimal parameters and architectures for each network would be most problematic. Also, whenever further data needed to be accommodated by the network, the training parameters would need to be re-optimised.

Table 15: True positive accuracies for phoneme /I/

	Recall Data		
Training Data	A	B	C
A	100	82.35	66.09
B	89.66	100	51.3
C	90.81	90.59	100

Table 16: True negative accuracies for phoneme /I/

	Recall Data		
Training Data	A	B	C
A	98.31	97.97	95.62
B	99.09	99.49	99.38
C	54.25	52.36	42.27

Table 17: True positive accuracies for phoneme /e/

	Recall Data		
Training Data	A	B	C
A	99.61	94.35	55.61
B	90.51	99.19	33.18
C	100	100	100

Table 18: True negative accuracies for phoneme /e/

	Recall Data		
Training Data	A	B	C
A	96.55	96.42	97.15
B	97.87	98.8	99.11
C	0	0	0

Table 19: True positive accuracies for phoneme /&/

	Recall Data		
Training Data	A	B	C
A	100	95.65	99.47
B	87.02	100	59.47
C	92.98	99.28	100

Table 20: True negative accuracies for phoneme /&/

	Recall Data		
Training Data	A	B	C
A	97.01	96.06	93.49
B	98.83	99.13	99
C	49.42	48.33	39.97

6 Future Work

The experimental results indicate that the SECoS paradigm suffers from both a lack of generalisation and a sensitivity to the training parameters. Future development of the ECoS paradigm and SECoS architecture will attempt to remedy these shortcomings by focusing on several different areas.

The first of these is the addition of a fitness value to the nodes of the evolving layers. This will give several advantages, such as:

- more intelligent aggregation, by weighting the resulting, aggregated node towards the fitter node
- improved recall, by giving more trust to the activation of a more fit node
- allows for an intelligent implementation of "forgetting"

Forgetting in this case will consist of a gradual "drift" of nodes in space, moving progressively closer to those nodes that win the most. The rate of this drift will be related to the relative fitness of the nodes, so that a more fit node attracts a less fit node.

The second of these is the implementation of training parameters that are local to the nodes of the evolving layer. These parameters will be stored as genes, which will be crossed over and mutated as the network evolves. This will provide the ECoS with a self-adaptation capability for its parameters, which may help in reducing the wide disparity in results presented in this paper.

7 Conclusions

The paper presents a simplified version of an evolving connectionist system (ECoS) called SECoS where the second layer of neurons evolves through on-line, adaptive, incremental, one-pass learning from data. SECoS with aggregation are illustrated on phoneme data classification.

The results show that although the model is capable of good memorisation of data and adaptation to new data, this performance is at the expense of a large number of hidden nodes. Aggregation of the hidden layers has succeeded in significantly reducing the size of the hidden layer and somewhat improving the generalisation accuracy, but severely reduces the accuracy over previously seen data. Comparison of SECoS with bootstrapped backpropagation trained MLPs has shown that while SECoS are larger than MLPs, they are comparatively much more adaptive, capable of retaining their discriminatory capabilities even after further training on new examples.

The wide range of performance between phonemes strongly suggests that the SECoS model is sensitive to both training and aggregation parameters. The suggested future work therefore focuses on automatically and dynamically adapting both the training and aggregation parameters.

8 Acknowledgements

This work was done as part of the research project UOO808 funded by the Foundation of Research Science and Technology of New Zealand. We would also like to acknowledge the assistance of Richard Kilgour and Akbar Ghobakhlou with proof-reading this paper.

Bibliography

[1] H. DeGaris. Circuits of production rule gennets - the genetic programming of artificial nervous systems. In R. Albrecht, Reeves. C., and N. Steele, editors, *Artificial Neural Networks and Genetic Algorithms*. Springer Verlag, 1993.

[2] G. Edelman. *The theory of neuronal group selection*. Basic Books, 1992.

[3] D. Fogel. *Evolutionary Computation*. IEEE Press, 1996.

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

[5] N. Kasabov. ECOS: A framework for evolving connectionist systems and the eco learning paradigm. In *Pro. of ICONIP'98, Kitakyushu, Japan, Oct. 1998*, pages 1222–1235. IOS Press, 1998.

[6] N. Kasabov. The ECOS framework and the ECO learning method for evolving connectionist systems. *Journal of Advanced Computational Intelligence*, 2(6):195–202, 1998.

[7] N. Kasabov. Evolving fuzzy neural networks - algorithms, applications and biological motivation. In Yamakawa and Matsumoto, editors, *Methodologies for the Conception, Design and Application of Soft Computing*, pages 271–274. World Scientific, 1998.

[8] N. Kasabov. Evolving connectionist systems and evolving fuzzy neural networks - methods, tools, applications. In N. Kasabov and R. Kozma, editors, *Neuro-fuzzy techniques for intelligent information systems*. Springer Verlag, 1999.

[9] N. Kasabov. Evolving connectionnist systems for online, knowledge-based learning and self-optimisation. *IEEE Trans. on Man, Systems, and Cybernetics*, Submitted 1999.

[10] N. Kasabov, R. Kozma, R. Kilgour, M. Laws, J. Taylor, M. Watts, and A. Gray. A methodology for speech data analysis and a framework for adaptive speech recognition using fuzzy neural networks and self organising maps. In N. Kasabov and R. Kozma, editors, *Neuro-fuzzy techniques for intelligent information systems*. Physica Verlag (Springer Verlag), 1999.

[11] John Platt. A resource-allocating network for function interpolation. *Neural Computation*, 3(2):213–225, 1991.

[12] S. Sinclair and C. Watson. The development of the otago speech database. In N. Kasabov and G. Coghill, editors, *Proceedings of ANNES'95*. IEEE Computer Society Press, 1995.

[13] M. Watts and N. Kasabov. Genetic algorithms for the design of fuzzy neural networks. In *Proceedings of ICONIP'98, Kitakyushu, Japan, October 1998*, 1998.

[14] J. Yao. Evolving aritficial neural networks. *Proceedings of IEEE*, 87(9):1423–1447, September 1999.