Dynamic Optimisation of Evolving Connectionist System Training Parameters by Pseudo-Evolution Strategy

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

Abstract- The paper presents a method based on evolution strategies that attempts to optimise the training parameters of a class of on-line, adaptive connectionist-based learning systems called evolving connectionist systems (ECoS). ECoS are systems that evolve their structure and functionality through on-line, adaptive learning from incoming data. The ECoS paradigm is combined here with the paradigm of evolutionary computation to attempt to solve a difficult task of on-line adaptive adjustment and optimisation of the parameter values of the evolving system. Although the method presented is unsuccessful, some useful information about the properties of the ECoS model is still derived from the work.

1 Introduction

There are several methods that have been introduced to optimise the parameters of neural networks (NN) such as MLP, RBF [5] fuzzy neural networks and others [7]. These NN have more or less a fixed structure (except the ones that have their number of hidden nodes optimised as a parameter), deal with off-line, batch mode learning, apply many iterations for learning and optimisation. These systems despite their advantages are not applicable to on-line, adaptive learning as described in [4].

On-line learning systems, that learn in an adaptive way from incoming data streams, are difficult to optimise in terms of choosing optimal values for their parameters especially in the case of input streams that change their dynamics and statistical parameters over time. This is one of the reasons why on-line learning is not so applicable at present.

The paper takes one of the recently introduced paradigms called evolving connectionist systems (ECoS) [3] and develops a method based on evolution strategies that attempts to optimise the learning parameter values of an evolving system as the system operates.

2 Evolving Connectionist Systems

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 [3].

ECoS are multi-level, multi-modular structures where many modules have inter-, and intra-module 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 [4]:

1. ECoS learn quickly 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 have an "open" structure where new features (relevant to the task) can be introduced at any stage of the systems operation, for example, the system creates "on the fly" new inputs, new outputs, new modules and connections
4. ECoS memorise data exemplars for a further refinement, or for information retrieval
5. ECoS learn and improve through active interaction with other systems and with the environment in a multi-modular, hierarchical fashion
6. ECoS adequately represent space and time in their different scales; have parameters that represent short-term and long-term memory, age, forgetting, etc.

3 Simple Evolving Connectionist Systems

3.1 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. 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.

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

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.

3.2 The SECoS Learning Algorithm

The learning algorithm is as follows.

1. propagate the input vector I through the network
2. IF the maximum activation a_{max} is less than the sensitivity threshold S_{thr}
   - add a node
3. 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 connections to the winning hidden node
4. 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 $W^t_{i,j}$ is the connection weight from input i to j at time t, η_1 is the learning rate one parameter, and I_i is the ith component of the input vector I.

$$W^{t+1}_{i,j} = W^t_{i,j} + \eta_1(I_i - W^t_{i,j}) \quad (3)$$

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

$$W^{t+1}_{j,o} = W^t_{j,o} + \eta_2(A_j \times E_o) \quad (4)$$

4 The Parameter Selection Problem and Evolutionary Computation

While ECoS is a flexible and powerful paradigm for on-line, knowledge-based learning, the performance of ECoS depend a great deal on their parameters, such as sensitivity and error thresholds, learning rate, pruning parameters and aggregation parameters [8]. Optimisation of these parameters is a difficult task that may be achieved using an evolutionary algorithm.

When applying Evolutionary Computation (EC) to the optimisation of ECoS there are two requirements that must be met:

- the evolving layer nodes in ECoS structures have the meaning of cluster centres in the input space and are not subject to optimisation with any of the EC algorithms
- the optimisation of the parameters has to be fast as the ECoS are learning on-line

While a model such as a Genetic Algorithm (GA) [2] would be useful, the requirement of a multiple-individual population causes some problems in this application area. Ideally, at every step of the training, the ECoS network must be available for use: this is a central tenet of the online learning paradigm. Optimising the parameters via GA would require that a population of individuals, representing the current set of candidate parameters, be kept at hand during the life of the network. Also, the time required to evaluate each individual in the population, for each generation run for each training step (training example) has the potential to significantly slow the operation of the system.

A two membered Evolution Strategy (ES) [1] avoids the first of these problems. The current training parameters can be used as the parent of the next generation, thus avoiding the need to maintain extraneous parameters. The second problem is also avoided, as only one individual at a time needs to be evaluated. By restricting the number of attempts at replacement to be made, the ES can be made much faster than a GA. While a GA based method is still feasible, and may be investigated at a later time, the method presented here will focus on ES.

5 Optimising ECoS Training Parameters with Evolution Strategies

One training cycle in the ECoS paradigm is defined as the presentation to the network, and the accommodation of, a single new training example. If we regard the change in network structure and connection weights from time t to time $t + 1$ as a state transition, then our goal is to optimise the fitness of this transition. Thus, the training parameters must be adjusted at each time step (that is, for each training example).

If the training parameters are encoded into a chromosome, then the chromosome can be mutated at each time step. By evaluating the fitness of the transition made using these mutated parameters, the parameters can be optimised for that time step. If the fitness of a new set of parameters is greater than that for the current parameters, (i.e. the new parameters produce a better transition than the current parameters), then the current parameters are replaced. Otherwise, the new parameters are discarded, and the mutation and testing is repeated. The mutating and testing cycle continues until the specified number of replacements have occurred, or the maximum number of replacement attempts has occurred, at which point the network is updated using the current training parameters. This algorithm is presented in pseudocode below, where f_c is the fitness of the current parameters p_c, f_m is the fitness of the mutated current parameters p_m, A is the number of attempts at replacement made, A_{max} is the maximum number of attempts allowed, R is the number of replacements and R_{max} is the maximum number of replacements allowed.

1. f_c=evaluate(p_c)
2. while $f_m < f_c$ and $A < A_{max}$ and $R < R_{max}$
   - p_m=mutate(p_c)
   - f_m=evaluate(p_m)
   - if $f_m > f_c$
     - $p_c = p_m$
     - $f_c = f_m$
     - $R = R + 1$
   - else
     - $A = A + 1$
3. train network using p_c

The fitness of the transition is measured according to Equation 5. This fitness measure is based on three components. The first is the change in size of the network. ECoS networks can rapidly increase in size if left unchecked, resulting in a large, inefficient network that lacks generalisation ability. The network size fitness component will reward the individual if the network does not increase in size. The purpose of this term is to discourage the network from growing too large too quickly. The second component is the local error of the network. At each time step, the error of the network over the current training example is evaluated. If the error increases then the individual will be punished. The purpose of this term is to ensure that the current training example is being adequately catered for by the network. The final component is the change in training data error. As nodes are added to the evolving layer of the network, the training examples that cause nodes to be added are copied into a "short term memory" (STM). Since these examples are sufficiently unique to have triggered the addition of a node, they are therefore representative of the entire training data set to date. By measuring the accuracy of the network over the STM, the training accuracy of the network can be measured. A positive change in this accuracy (i.e., the error at time $t+1$ is less than at time t) will cause the individual to be rewarded. A negative change will cause it to be punished. The purpose of this term is to limit forgetting of previous training examples.

$$f_t = s + e + g \tag{5}$$

where:
f_t is the fitness of the transition
s is the size change component of the evaluation, as in Equation 6
e is the local error change component of the evaluation, as in Equation 7
g is the global error change component of the evaluation, as in Equation 8

$$s = \begin{cases} 1 & \delta_n = 0 \\ -1 & \text{otherwise.} \end{cases} \tag{6}$$

where:
δ_n is the change in the number of nodes in the evolving layer.

$$e = \begin{cases} 1 + \delta_e & \delta_e \geq 0 \\ \delta_e & \text{otherwise.} \end{cases} \tag{7}$$

where:
δ_e is the change in local error for this timestep.

$$g = \begin{cases} 1 + \delta_g & \delta_g \geq 0 \\ \delta_g & \text{otherwise.} \end{cases} \tag{8}$$

where:
δ_g is the change in error over the STM.

6 Experiments

6.1 Isolated Phoneme Recognition

The suggested model is investigated 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 [6] (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.

6.2 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.

6.3 Experimental Design

For each 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. Experiments were carried out over three phonemes, as 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: Target Phonemes

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

6.3.1 Baseline Results

So that a comparison can be made between training with dynamically optimised parameters and training with static parameters, results from training with static parameters according to the experimental methodology above are presented. These results were first published in [9], and are reproduced here solely for comparison.

The training parameters used are listed in Table 2. 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 training on one data set.

6.3.2 Results

Table 2: Baseline 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

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

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

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

6.4 Experiment One

For the first experiment, the initial parameter values were set as in Table 2. The algorithm was then allowed to run, with one run of the evolution strategy / training algorithm being equivalent to one run of the training algorithm from subsection 6.3.1. There were no constraints applied to the parameters during the experiment.

6.4.1 Experiment One Results

The mean true positive percentage accuracies are presented in Tables 9, 12 and 15. The mean true negative accuracies are presented in Tables 10, 13 and 16. The mean final parameters selected are presented in Tables 11, 14 and 17.

The first thing that is immediately apparent in these results is the small number of hidden nodes created during training. This means that the parameters were driven towards values that would inhibit the addition of nodes. Inspection of the mean final values confirms this: for each phoneme, the error threshold parameter is very high, in every case over 0.9. This makes the network very tolerant of errors, which decreases the likelihood of nodes being added. The small number of nodes can lead to problems with discriminating between the two classes each network is learning. The accuracies for each phoneme show that this is indeed the case: while the phoneme /I/ network has a high true positive rate that increases with further training (demonstrating a resistance to forgetting of the target class) this is at the expense of the true negative rate. The networks for phoneme /e/ and /&/ both have poor initial recognition rates over both classes. While the true positive rates for these networks increase with further training, the discrimination ability decreases rapidly.

These problems are caused by the error threshold parameter being driven towards such a large value. The following experiment constrains the parameter values to more useful ranges.

Table 9: Mean true positive accuracies for phoneme /I/

	Recall Data			
Training Data	A	B	C	Hidden Nodes
A	88.7	87.71	92.17	4
B	96.93	96.86	99.81	5
C	96.93	96.86	100	5

Table 10: Mean true negative accuracies for phoneme /I/

	Recall Data			
Training Data	A	B	C	Hidden Nodes
A	43.8	44.05	39.73	4
B	38.25	38.46	34.5	5
C	38.24	38.45	34.49	5

Table 11: Mean final parameters for phoneme /I/

	Parameter			
Training Data	Sensitivity	Error	η_1	η_2
A	0.11	0.95	0.53	0.44
B	0.11	0.95	0.47	0.43
C	0.11	0.95	0.54	0.41

Table 12: Mean true positive accuracies for phoneme /e/

	Recall Data			
Training Data	A	B	C	Hidden Nodes
A	59.29	60.48	65.27	4
B	87.83	88.62	96.05	6
C	87.92	88.62	96.26	6

Table 13: Mean true negative accuracies for phoneme /e/

	Recall Data			
Training Data	A	B	C	Hidden Nodes
A	66.83	66.94	73.67	4
B	45.13	45.12	51.37	6
C	45.04	45.04	51.31	6

Table 14: Mean final parameters for phoneme /e/

	Parameter			
Training Data	Sensitivity	Error	η_1	η_2
A	0.11	0.91	0.34	0.46
B	0.09	0.93	0.41	0.4
C	0.09	0.93	0.41	0.4

6.5 Experiment Two

The second experiment was run much like the first, with the exception that the parameters were constrained to the ranges in Table 18.

Table 15: Mean true positive accuracies for phoneme /&/

	Recall Data			
Training Data	A	B	C	Hidden Nodes
A	50.59	64.98	62.16	4
B	90.51	84.62	78.95	6
C	89.55	85.91	84.68	8

Table 16: Mean true negative accuracies for phoneme /&/

	Recall Data			
Training Data	A	B	C	Hidden Nodes
A	67.37	67.75	72.56	4
B	43.72	43.8	45.81	6
C	43.35	43.89	46.03	8

Table 17: Mean final parameters for phoneme /&/

	Parameter			
Training Data	Sensitivity	Error	η_1	η_2
A	0.08	0.9	0.56	0.48
B	0.15	0.93	0.59	0.48
C	0.11	0.92	0.52	0.55

Table 18: Training Parameter Ranges

Parameter	Minimum	Maximum
Sensitivity Threshold	0.25	0.75
Error Threshold	0.1	0.25
Learning Rate One	0.25	0.75
Learning Rate Two	0	1

6.5.1 Experiment Two Results

The true positive percentage accuracies are presented in Tables 19, 22 and 25. The true negative accuracies are presented in Tables 20, 23 and 26.

High true negative accuracies are immediately apparent when examining these results. This high negative accuracy, however, has come at the cost of true positive accuracy. While constraining the values of the error threshold has allowed the networks to grow to a much larger size than in Experiment One, the final network sizes are much smaller than those of the baseline networks.

While it was initially thought that the low true positive accuracies were due to the implementation of the STM as a fixed length queue, later experiments have shown that this is not the case, as neither increasing or decreasing the size of the STM has been found to cause significant changes in accuracy.

6.6 Experiment Three

Although setting ranges of parameters, rather than exact parameter values, is an improvement over the previous state of affairs, it still falls short of the goal of a system that can control itself. Constraining the parameters is necessary because

Table 19: Mean true positive accuracies for phoneme /I/

	Recall Data			
Training Data	A	B	C	Hidden Nodes
A	34.67	29.41	8.41	194
B	68.2	82.75	4.35	330
C	68.39	81.76	90	445

Table 20: Mean true negative accuracies for phoneme /I/

	Recall Data			
Training Data	A	B	C	Hidden Nodes
A	100	99.71	97.86	194
B	98.85	99.92	97.02	330
C	98.97	99.75	99.91	445

Table 21: Mean final parameters for phoneme /I/

	Parameter			
Training Data	Sensitivity	Error	η_1	η_2
A	0.07	0.21	0.52	0.42
B	0.13	0.24	0.44	0.59
C	0.13	0.24	0.45	0.64

Table 22: Mean true positive accuracies for phoneme /e/

	Recall Data			
Training Data	A	B	C	Hidden Nodes
A	32.94	18.55	6.7	155
B	70.75	86.02	24.77	294
C	70.36	85.75	79.44	432

Table 23: Mean true negative accuracies for phoneme /e/

	Recall Data			
Training Data	A	B	C	Hidden Nodes
A	100	99.68	99.4	155
B	99.06	99.95	98.78	294
C	98.08	98.69	99.89	432

Table 24: Mean final parameters for phoneme /e/

	Parameter			
Training Data	Sensitivity	Error	η_1	η_2
A	0.13	0.17	0.55	0.67
B	0.09	0.18	0.61	0.79
C	0.26	0.23	0.57	0.43

the Error Threshold parameter, the parameter that controls the tolerance of the network to errors, will rapidly become extremely large if left alone. This is due to the component of the fitness function that rewards the network for maintaining its current size, and punishes it for adding nodes. Although some level of network accuracy is included in the fitness measure by evaluating the recall error over the STM, this has proven to be inadequate. The reason for this is that training examples

Table 25: Mean true positive accuracies for phoneme /&/

	Recall Data			
Training Data	A	B	C	Hidden Nodes
A	41.99	34.06	41.75	207
B	70.76	89.37	55.79	357
C	69.65	88.04	83.16	442

Table 26: Mean true negative accuracies for phoneme /&/

	Recall Data			
Training Data	A	B	C	Hidden Nodes
A	100	99.51	98.92	207
B	99.11	99.84	97.82	357
C	99.02	99.68	99.88	442

Table 27: Mean final parameters for phoneme /&/

	Parameter			
Training Data	Sensitivity	Error	η_1	η_2
A	0.07	0.25	0.54	0.52
B	0.07	0.25	0.54	0.52
C	0.1	0.25	0.61	0.44

are inserted into the STM only when a new node is added to the network. If few nodes are added to the network, there will be few examples in the STM, which makes it much easier for the network to maintain a low error across it. This problem may be addressed by altering the criteria under which examples are added to the STM. One possibility is to add examples randomly, with the probability of addition being proportional to the network error over that example. Thus, examples that do not trigger node addition will also be included in the STM, making it more difficult for the network to gain an inappropriately low error rate across it.

The algorithm was thus modified so that an example that does not trigger the addition of a node to the evolving layer will be added to the STM with a probability proportional to the current error threshold. This is shown in Equation 9, where p_a is the probability of adding the example to the STM, e_i is the error over example i and e_{thr} is the current error threshold.

$$p_a = \frac{e_i}{e_{thr}} \tag{9}$$

The true positive percentage accuracies are presented in Tables 28, 30 and 32. The true negative accuracies are presented in Tables 29, 31 and 33.

These results are, overall, worse than before. While the initial data set was learned very well, with a high true positive and true negative rate, the networks demonstrated severe difficulty in adapting to new data. The mean number of hidden nodes was larger than with experiment one, but smaller than experiment two.

The addition of more examples to the STM had the effect of pushing the network towards a greater number of hidden nodes, but the effect it had on the overall performance is unacceptable.

Table 28: Mean true positive accuracies for phoneme /I/

	Recall Data			
Training Data	A	B	C	Hidden Nodes
A	94.13	11.11	11.4	112
B	93.3	11.11	12.46	142
C	92.72	9.93	6.28	186

Table 29: Mean true negative accuracies for phoneme /I/

	Recall Data			
Training Data	A	B	C	Hidden Nodes
A	91.31	89.62	87.07	112
B	91.6	89.98	87.68	142
C	92.43	91	92.53	186

Table 30: Mean true positive accuracies for phoneme /e/

	Recall Data			
Training Data	A	B	C	Hidden Nodes
A	95.13	5.73	7.68	94
B	94.73	7.35	8.67	125
C	94.55	9.77	8.15	181

Table 31: Mean true negative accuracies for phoneme /e/

	Recall Data			
Training Data	A	B	C	Hidden Nodes
A	93.91	92.13	91.88	94
B	93.48	91.57	91.92	125
C	90.9	89.02	89.95	181

Table 32: Mean true positive accuracies for phoneme /&/

	Recall Data			
Training Data	A	B	C	Hidden Nodes
A	94.46	9.58	14	104
B	95	8.45	13.74	138
C	93.06	7.09	7.13	171

Table 33: Mean true negative accuracies for phoneme /&/

	Recall Data			
Training Data	A	B	C	Hidden Nodes
A	92.2	89.29	87.24	104
B	91.59	88.25	86.9	138
C	92.9	90	92.7	171

7 Discussion

The experiments presented in this paper demonstrate that the algorithm suggested is not appropriate for the task of optimising ECoS training parameters. However, analysis of the results does suggest several important points:

- the error threshold parameter is the most significant training parameter in an ECoS network. The fact that the ES consistently pushed the error threshold towards very high values, and that these high values resulted in very small networks, very strongly indicates that the error threshold is the defining parameter of ECoS training.
- the number of neurons in the hidden layer does not directly affect the accuracy of the network. There was a wide variety of both true negative and true positive accuracies displayed, with the accuracies seemingly unrelated to the accuracy of the network.
- the internal dynamics of ECoS style networks are more complex than anticipated. As indicated in the previous point, accuracy is not obviously linked to the number of neurons present. Thus, more rigorous investigation of the learning process of ECoS networks is required. Of particular interest is the question of what is actually stored in each added neuron, and how is that affected by the parameters?
- a purely local learning algorithm, combined with a purely local optimisation algorithm, does not guarantee good results. The need for optimisation at the global level is apparent. The STM was an attempt at including a global component in the optimisation procedure, but was inadequate.

Bearing in mind these points, it becomes clear that an evolution strategy based optimisation of parameters for every training example is not appropriate. While several points against the use of GA based optimisation algorithms were discussed in Section 4, the need for global optimisation may mitigate against the other disadvantages.

8 Conclusion

The paper presents an evolution strategy based algorithm for automatically and dynamically adapting the training parameters of a class of neural networks called Evolving Connectionist Systems (ECoS). The algorithm has been tested on the case study of isolated phoneme recognition, where it is shown to have poor performance. Analysis of the results, however, suggest several avenues for future research.

9 Acknowledgements

The authors would like to acknowledge the assistance of Richard Kilgour and Akbar Ghobakhlou with proof reading this paper.

Bibliography

[1] Thomas Bäck, Frank Hoffmeister, and Hans-Paul Schwefel. A survey of evolution strategies. In Richard K Belew and Lashon B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 2–9, 1991.

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

[3] 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.

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

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

[6] 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.

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

[8] Michael Watts. An investigation of the properties of evolving fuzzy neural networks. In *Proceedings of ICONIP'99, November 1999, Perth, Australia*, pages 217–221, 1999.

[9] Michael Watts and Nik Kasabov. Simple evolving connectionist systems and experiments on isolated phoneme recognition. In Xin Yao and David B. Fogel, editors, *2000 IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks*, pages 232–239, 2000.