Sleep Learning and Max-Min Aggregation of Evolving Connectionist Systems

Michael J. Watts
Information Technology Programme
Auckland Institute of Studies
Auckland
New Zealand
Email: mjwatts@ieee.org

***Abstract*—This paper describes two new algorithms for optimising the structure of trained Evolving Connectionist System (ECoS) artificial neural networks (ANN). It also presents the results of preliminary empirical evaluations of the algorithms. While ECoS are fast and efficient constructive ANN algorithms they can lose efficiency if they are allowed to grow too large. The algorithms presented in this paper reduce the size of a trained ECoS while retaining the knowledge that the ECoS has learned. That is, they remove redundant elements of the ECoS structure in such a way that the performance of the network is not reduced. The experimental evaluations showed that each algorithm is capable of achieving this to a different degree over different data sets. Optimisation of the parameters of one of the algorithms using an evolutionary algorithm yielded better results. While the work reported in this paper is preliminary, the results are promising and the algorithms have the potential to enhance the usefulness of ECoS ANN.**

I. Introduction

Evolving Connectionist Systems (ECoS) are a family of constructive neural networks [1]. They are based on the following principles as stated in [2]:

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;

The simplest of the ECoS algorithms is the Simple Evolving Connectionist System (SECoS) [3], [4]. This consists of three layers of neurons: the input layer; the evolving layer; and the output layer. Neurons are added to the evolving layer during learning, and the activation of the evolving layer neurons is based on the distance between the current input vector and the evolving layer neuron's incoming weight vector. SECoS have been applied to a variety of problems, including phoneme recognition [5], [6], computer network security [7] and predicting outbreaks of insect pests [8], [9]. While SECoS are likely not capable of solving the kind of complex problems that deep-learning convolutional neural networks are, they do have some key advantages:

1) They are fast learning
2) They are computationally efficient in recall

These make SECoS useful for applications in resource-constrained environments. However, the advantage of being computationally efficient is lost if the evolving layer is allowed to grow too large. This can happen when a large number of training examples have been presented to the SECoS. Due to the local-learning employed by the training algorithm, it is possible that many of the evolving layer neurons are actually redundant [10]. That is, the information represented by the neuron is or can be represented by another neuron.

Previous work has described different approaches to dealing with this problem. One way is to optimise the training parameters via evolutionary algorithm [11]–[14]. This has the disadvantages of firstly requiring training and testing data sets with which to gauge the performance of the trained ECoS, and secondly of being computationally intensive due to the number of evaluations required by an EA. Another approach was via neuron aggregation [4]. This involves finding groups of evolving layer neurons that are close together, then combining them into a single neuron. This has the advantages of not requiring any external data sets and also being computationally efficient.

This paper introduces two alternative methods for optimising trained SECoS networks, one based on the principle of sleep learning and the other on data clustering of neurons. Both of these methods are computationally efficient, and do not require external data sets.

II. SECoS Learning

The ECoS learning algorithm is based on accommodating within the evolving layer new training examples, by either modifying the weight values of the connections attached to the evolving layer neurons, or by adding a new neuron to that layer. The algorithm employed is described in Figure 1.

When a neuron 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 $\mathbf{O}_d$.

for each input vector $\mathbf{I}$ and its associated desired output vector $\mathbf{O}_d$ **do**
  Propagate $\mathbf{I}$ through the network
  Find the most activated evolving layer neuron j and its activation A_j
  if $A_j < S_{thr}$ **then**
    Add a neuron
  else
    Find the errors between $\mathbf{O_d}$ and the output activations A_o
    if $|\mathbf{O}_d - A_o| > E_{thr}$ **then**
      Add a neuron
    else
      Update the connections to the winning evolving layer neuron j
    end if
  end if
end for

Fig. 1: ECoS learning algorithm

The weights of the connections from each input i to the winning neuron j are modified according to Equation 1.

$$\mathbf{W}_{i,j}(t+1) = \mathbf{W}_{i,j}(t) + \eta_1(\mathbf{I}_i - \mathbf{W}_{i,j}(t)) \quad (1)$$

where:
$\mathbf{W}_{i,j}(t)$ is the connection weight from input i to j at time t
η_1 is the learning rate one parameter
$\mathbf{I}_i$ is the ith component of the input vector $\mathbf{I}$

The weights from neuron j to output o are modified according to Equation 2.

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

where:
$\mathbf{W}_{j,o}(t)$ 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
E_o is the signed error at o, as measured according to Equation 3.

$$E_o = \mathbf{O}_d - A_o \quad (3)$$

where:
$\mathbf{O}_d$ is the desired activation value of o
A_o is the actual activation of o.

This is essentially the perceptron learning rule.

III. Sleep Learning

The neurons in the evolving layers of a SECoS represent prototypes that are acquired and modified during training. Novel training examples are added as prototypes, and they are modified to be representative of groups of examples. During training, modification of the connection weights of ECoS networks may move the evolving layer neurons to be very close together in the input space, making some of the neurons redundant. Sleep learning is a way of identifying and removing these redundant neurons. It is performed in an offline mode, that is, between training sesssions, using the exemplars stored within the ECoS network: the network is "asleep" to external stimuli.

The idea of sleep learning in ECoS networks has been suggested before [15] but the form of sleep learning discussed in the previous work is based on the idea of strengthening already learned concepts, rather than reducing the size of the network. As it is described in this section, it is based upon the Grow and Learn ANN (GAL) sleep learning algorithm [16], with modifications that account for the differences between GAL and ECoS. Thus, it is able to reduce the size of the ECoS network while also retaining the knowledge captured by the network. The algorithm for sleep learning of ECoS is as in Figure 2.

for each neuron n in the evolving layer **do**
  Extract the incoming and outgoing connection weights to use as an example.
  Because the outgoing weight values can exceed unity, process these weights to fall into the range of the output layer activation function.
end for
for each example x in the set of extracted examples **do**
  Remove the corresponding neuron n from the network.
  Propagate x through the network.
  if the maximum neuron activation in the evolving layer is less than S_{thr} OR the error is greater than E_{thr} **then**
    Re-insert n into the network
  else
    modify the incoming and outgoing connection weights of the winning neuron, according to Equations 1 and 2
  end if
end for

Fig. 2: Sleep learning algorithm

This is essentially the ECoS learning algorithm, applied one example at a time, where the example is a prototype extracted from the SECoS. All exemplars are extracted at the start of learning because the weight modification to neurons examined later in the sleep learning process can cause a disruption to the sleep learning process. This is because the purpose of sleep learning is to eliminate superfluous neurons that are not necessary to adequately represent all exemplars stored within the network at the start of sleep learning. If the exemplar for a neuron is extracted after learning has been applied to that neuron, then the exemplar will be different. In effect, this would cause the sleep learning algorithm to chase a moving target.

IV. Max-Min aggregation

Evolving layer neuron aggregation is the process of combining several adjacent neurons into one neuron 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 neurons is calculated. If the distances are below specified thresholds, the neurons are aggregated together. The rationale behind aggregation is to reduce the size of the evolving layer of the SECoS, while retaining the knowledge stored within the connections to each neuron.

Offline aggregation involves finding all of the neurons that are closer to each other than a certain threshold distance, then aggregating them together. Aggregation is achieved by finding the arithmetic mean of the neuron connection weights. Offline aggregation is performed when training is not being carried out, and usually has both an incoming connection weight distance threshold and an outgoing connection weight distance threshold. This approach does not allow for direct control of the final size of the SECoS network.

An alternative aggregation method as proposed here is the offline Max-Min aggregation algorithm. This is described in Figure 3.

- Extract the exemplars from the trained SECoS
- Calculate the distance between each pair of exemplars
- Select the k exemplars that are the furthest away from other exemplars
- Perform k-means clustering on the exemplars, using the k exemplars selected above as the initial cluster centres
- Replace the evolving layer neuron connection weights with the cluster centroids

Fig. 3: The Max-Min aggregation algorithm

It is called the "Max-Min" algorithm because it maximises the minimum distance between the initial cluster centroids, and therefore maximises the distance between the resulting aggregated neurons. The advantage of this algorithm is that the number of remaining neurons (k) is determined *a priori*, and that this number is the sole parameter used. A single parameter is, of course, easier to optimise compared to approaches using multiple parameters.

V. Experiments

A. Sleep Learning

As SECoS is primarily a classification algorithm, the preliminary experiments were carried out using the Iris, Mushroom and Wine classification data sets from the UCI Machine Learning Repository [17]. As the exact examples used and their order presented to a SECoS during training will affect its performance [10] , each data set was randomly partitioned into a training and testing data set one hundred times. For each partition, a new SECoS was created then trained on the training portion of the data set. The SECoS was then recalled over the training portion, to determine how well it had memorised the training data, and recalled again over the test portion to determine how well it generalised. The SECoS was then subjected to sleep learning optimisation. The optimised SECoS was then recalled over the training and test data portions. This showed how much the SECoS forgot during sleep learning, and how much generalisation deteriorated. The size of the SECoS, in terms of the number of neurons in the evolving layer, was recorded before and after sleep learning. The initial learning and sleep learning parameters were set as in Table I. The sleep learning sensitivity threshold (SensThr) parameter was set low to reduce the number of neurons, while the error threshold (ErrThr) parameter was lowered to maintain the accuracy of the network. One hundred iterations were carried out over each data set.

TABLE I: Training and sleep learning parameters

	SensThr	ErrThr	η_1	η_2
Initial	0.5	0.1	0.5	0.5
Sleep	0.2	0.01	0.5	0.5

B. Evolved-parameter Sleep Learning

In light of the results found for the mushroom data set, it became apparent that sleep learning is sensitive to parameters. Additional experiments were therefore carried out to investigate using a simple genetic algorithm (GA) to optimise the sleep learning parameters. While evolutionary optimisation of the learning parameters is computationally intensive and time-consuming, because sleep learning is a relatively light-weight optimisation algorithm, it is feasible to optimise the parameters using a GA.

The fitness function used in the GA was:

$$f_i = \frac{n_t - n_{t+1}}{n_t} + (1 - e_{t+1}) \tag{4}$$

where:
n_t is the number of neurons in the evolving layer before sleep learning, and
n_{t+1} is the number of neurons in the evolving layer after sleep learning.
e_{t+1} is the network error over the extracted exemplars after sleep learning. The error over the extracted exemplars before sleep learning is always zero, as every exemplar will perfectly match one evolving layer neuron.

The goal of this fitness function is to maximise the size reduction of the SECoS while minimising the forgetting. While a multi-objective GA would likely give slightly better results, for this preliminary work a simple GA with a simple fitness function was felt to be sufficient.

The experimental approach used was highly similar to the above. A new SECoS was created and trained on a partition of the data set, then tested on both the training and testing partitions. Then the SECoS was subjected to sleep learning with evolved parameters and the resulting optimised SECoS recalled again over the training and testing partitions. The GA had a population size of 50 individuals and was run for 50 generations. One hundred iterations were performed for each data set.

C. Max-Min Aggregation

The Max-Min aggregation algorithm was investigated in a slightly different way. The data sets were partitioned into a

training and testing partition as before, and a new SECoS was created and trained over the training partition. The initial training parameters were the same as in Table I. At the completion of training the SECoS was recalled over the training and testing partitions. The SECoS was then subjected to Max-Min aggregation for every possible value of k for that SECoS. That is, if the number of evolving layer neurons was n, then k ranged from $n-1$ down to unity. Max-Min aggregation was performed for each value of k and the training and testing accuracy evaluated for each result. One hundred iterations of this process were carried out.

VI. Results

The results for the experiments using sleep learning are presented in Table II. Accuracies were measured as Cohen's Kappa statistic rather than percentages because Kappa is unaffected by unbalanced class distributions, which could occur with the random partitioning used in these experiments.

TABLE II: Results of sleep learning as mean and standard deviation of Cohen's Kappa.

Set		Train	Test	Neurons
Iris	Before	0.95/0.03	0.89/0.05	22.36/3.29
	After	0.94/0.04	0.88/0.05	20.53/3.27
Mushroom	Before	0.99/0.01	0.99/0.01	352.75/12.48
	After	0.08/0.09	0.08/0.09	269.49/14.16
Wine	Before	0.99/0.01	0.94/0.04	66.71/5.34
	After	0.96/0.03	0.91/0.05	52.82/5.89

While the change in accuracy for the iris networks was small the reduction in size of the SECoS was also small. Conversely, the mushroom SECoS had a greater reduction in size but the post-sleep learning accuracy was poor. Only the experiments with the wine data set showed a reasonable decrease in size with 21 % of the initial evolving layer neurons being removed, with only a 3 % decrease in the Kappa for both the train and test partitions. This indicates that the sleep learning algorithm is sensitive to the parameters used.

The results of using a GA to select the sleep learning parameters are presented in Table III. These results show that the GA was able to find a set of parameters that reduced the size of the mushroom networks without decreasing accuracy as much as the manually selected parameters: the number of evolving layer neuron on average were reduced by 91 %, while accuracy decreased by only 4 %. Meanwhile, the iris and wine networks were also reduced in size to a greater extent than before, by 68 % and 86 % respectively. These networks however suffered a slightly greater decrease in accuracy than before. It appears that the GA favoured reduction in size over retaining accuracy.

The sleep learning parameters selected by the GA are presented in Table IV. These parameters show a relatively large standard deviation (up to 69 % of the mean for the η_1 parameter of the iris data set), which indicates that there are a large number of parameter combinations that will reduce the size of the network.

The results of the experiments with Max-Min aggregation are plotted in Figure 4. As the initial (pre-aggregation) size

TABLE III: Results of GA sleep learning as mean and standard deviation of Cohen's kappa statistic.

Set		Train	Test	Neurons
Iris	Before	0.96/0.03	0.89/0.05	23.06/2.71
	After	0.87/0.09	0.81/0.12	7.4/2.43
Mushroom	Before	0.99/0.01	0.99/0.01	356.57/11.8
	After	0.95/0.15	0.95/0.15	29.75/6.82
Wine	Before	0.99/0.01	0.94/0.04	66.1/5.67
	After	0.91/0.06	0.85/0.08	9.03/2.88

TABLE IV: Mean and standard deviation of sleep learning parameters as determined by GA.

Set	SensThr	ErrThr	η_1	η_2
Iris	0.34/0.23	0.68/0.22	0.37/0.25	0.7/0.37
Mushroom	0.78/0.11	0.33/0.21	0.65/0.23	0.94/0.1
Wine	0.37/0.22	0.67/0.2	0.36/0.25	0.85/0.19

of the trained SECoS varied between iterations, the x-axes of these plots are the k values as a percentage of the initial size of the SECoS. The plots show the median values of the training and testing set kappas. The plots for all three data sets level off at the 50 % mark for both training and testing. The median kappas for each data set, with k set to 25, 50 and 75 % of the initial evolving layer size, are presented in Table V. Even with only 50 % of the number of neurons, the performance approaches that of the original, unoptimised networks as presented in Table II.

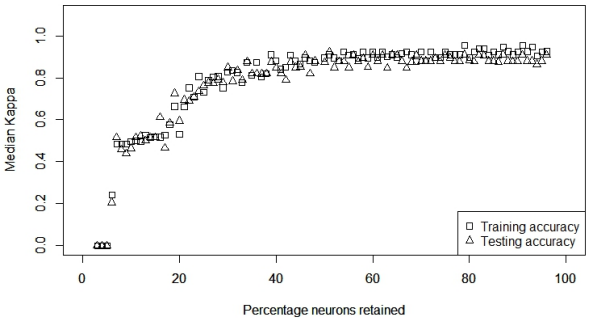
TABLE V: Median accuracies after Max-Min aggregation, when retaining 25, 50 and 75 % of the evolving layer neurons.

	25%		50%		75%	
Set	Train	Test	Train	Test	Train	Test
Iris	0.73	0.76	0.89	0.87	0.92	0.91
Mushroom	0.65	0.66	0.71	0.71	0.75	0.74
Wine	0.83	0.82	0.95	0.9	0.95	0.92

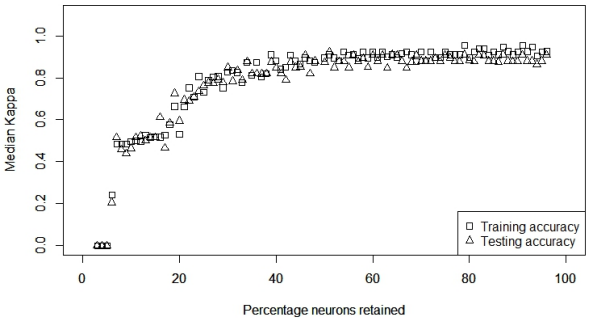
VII. Discussion

It is clear from the results that both sleep learning and Max-Min aggregation are able to reduce the size of a trained SECoS network, while retaining the classification accuracy of the network. For sleep learning, the performance is strongly determined by the four sleep learning parameters. Finding an optimal combination of parameter settings is not an easy task, although a simple genetic algorithm was able to yield good performance. It is apparent from the large variances of the sleep learning parameters found by the GA that there were many combinations of parameters. What is notable about these parameters is that the means are very different from the initial training parameters. The two learning rates were on average substantially higher, as were the error thresholds. It is apparent that the fitness function used for the GA allows too much weighting on size reduction and not enough on maintaining performance. An improved fitness function, or a multi-objective GA, would solve this problem.

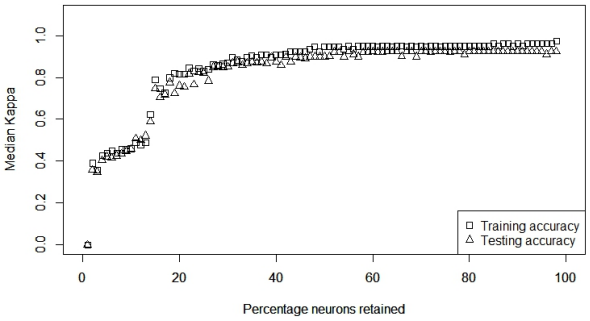
The Max-Min aggregation algorithm also yielded smaller networks with good classification performance. Depending on the number of neurons selected, the final accuracy for the iris



(a) Iris



(b) Mushroom



(c) Wine

Fig. 4: Max-Min aggregation results. Accuracies are Cohen's Kappa over the training and testing partitions of the Iris, Mushroom and Wine classification data sets.

and wine data sets exceeded the accuracy of the sleep-learning SECoS. Only the mushroom dataset was consistently better with sleep learning, but then only when a GA was used to optimise the parameters.

Optimisation of the Max-Min algorithm is possible and simpler as there is only one parameter, the value of k. By developing a suitable measure of optimisation, similar to the one used by the GA sleep training, it should be possible to efficiently find an optimal k value using an algorithm such as Golden Section search.

The algorithms presented in this paper both work, in that they achieve the desired results of reducing the number of neurons in the SECoS without degrading accuracy by an unacceptable amount. They are both relatively fast algorithms, with the Max-Min algorithm being slightly faster. The sleep learning algorithm is harder to use as it has four parameters compared to the one in Max-Min aggregation. However, optimising the four parameters in sleep learning using an evolutionary algorithm is less efficient than optimising the single parameter in Max-Min aggregation. In terms of ease-of-use, Max-Min aggregation is the only ECoS optimisation algorithm described so far that has only one parameter.

It is likely that which of the algorithms introduced in this paper will be useful for a particular problem, depends on the problem itself. Future work will examine more benchmark data sets, an improved evolutionary algorithm for selecting the sleep learning parameters, and optimisation of the k parameter for Max-Min aggregation. Most importantly, the algorithms must be evaluated over much larger data sets. The data sets in this paper, while well-known and adequate for the preliminary evaluation of the algorithms, were small and simplistic. They were also static data sets, which are not ideal for evaluating adaptive models such as ECoS.

REFERENCES

[1] M. Watts, "A decade of Kasabovs evolving connectionist systems: A review," *IEEE Transactions on Systems, Man and Cybernetics Part C - Applications and Reviews*, vol. 39, no. 3, pp. 253–269, 2009.

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

[3] M. Watts, "Evolving connectionist systems: characterisation, simplification, formalisation, explanation and optimisation," Ph.D. dissertation, University of Otago, 2004.

[4] M. Watts and N. Kasabov, "Simple evolving connectionist systems and experiments on isolated phoneme recognition," in *Proceedings of the first IEEE conference on evolutionary computation and neural networks, San Antonio, May 2000*. IEEE Press, 2000, pp. 232–239.

[5] A. Ghobakhlou and R. Seesink, "An interactive multi modal system for mobile robotic control," in *Proceedings of the Fifth Biannual Conference on Artificial Neural Networks and Expert Systems (ANNES2001)*. Citeseer, 2001, pp. 93–99.

[6] A. Ahmadi, F. Karray, and M. Kamel, "Modular-based classifier for phoneme recognition," in *Signal Processing and Information Technology, 2006 IEEE International Symposium on*. IEEE, 2006, pp. 583–588.

[7] H. Bensefia and N. Ghoualmi, "A new approach for adaptive intrusion detection," in *Computational Intelligence and Security (CIS), 2011 Seventh International Conference on*. IEEE, 2011, pp. 983–987.

[8] M. Watts and S. Worner, "Comparison of multi-layer perceptrons and simple evolving connectionist systems over the lincoln aphid data set," Bio-Protection and Ecology, Lincoln University, Tech. Rep., February 2007.

[9] M. J. Watts, "Evolving connectionist systems can predict outbreaks of the aphid rhopalosiphum padi," in *Neural Networks (IJCNN), 2014 International Joint Conference on*. IEEE, 2014, pp. 646–650.

[10] M. Watts, "Towards a formalisation of evolving connectionist systems," in *Artificial Neural Networks*. Novascience Press, 2010.

[11] M. Watts and N. Kasabov, "Evolutionary optimisation of evolving connectionist systems," in *Proceedings of the Congress on Evolutionary Computation (CEC) 2002*. IEEE, 2002, pp. 606–610.

[12] Z. Chan and N. Kasabov, "Evolutionary computation for on-line and off-line parameter tuning of evolving fuzzy neural networks," *International Journal of Computational Intelligence and Applications*, vol. 4, pp. 309–319, 2004.

[13] N. Kasabov and Q. Song, "GA-parameter optimisation of evolving connectionist systems for classification and a case study from bioinformatics," in *Proc. of ICONIP2002 - International Conference on Neuro-Information Processing, Singapore, November 2002*. IEEE Press, 2002, pp. 602–605.

[14] N. Kasabov, Q. Song, and I. Nishikawa, "Evolutionary computation for dynamic parameter optimisation of evolving connectionist systems for on-line prediction of time series with changing dynamics," in *Proceedings, IJCNN'2003, Portland, Oregon, July 2003*, 2003, pp. 438–443.

[15] N. K. Kasabov, "ECOS: Evolving connectionist systems and the ECO learning paradigm," in *ICONIP'98 Proceedings*, S. Usui and T. Omori, Eds., vol. 2, 1998, pp. 1232–1235.

[16] E. Alpaydin, "GAL: Networks that grow when they learn and shrink when they forget," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 8, no. 1, pp. 391–414, 1994.

[17] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml