Towards a Formalisation of Evolving Connectionist Systems

Michael J. Watts*

Abstract

Evolving Connectionist Systems (ECoS) are a class of constructive artificial neural networks that grow their structure as they learn. They have been widely applied to many problems. Among their advantages are fast, efficient training and a resistance to catastrophic forgetting. An attempt has previously been made to formally describe their operation and behaviour. This formalisation has some objections associated with them. This chapter describes the basic algorithms behind ECoS networks, critiques the previous formalisation and presents a new theory of ECoS networks that overcomes the objections associated with the previous work. Finally, the formalisation is tested on two well-known benchmark data sets.

1 Introduction

Evolving Connectionist Systems (ECoS) [7, 10, 21] are a class of constructive artificial neural networks that are similar in the way in which neurons are added to their structures, and in the way in which their connection weights are modified. The Evolving Fuzzy Neural Network EFuNN [8] was the first ECoS network, from which a generalised constructive ANN architecture and training algorithm was derived. Other ECoS networks include the Simple Evolving Connectionist System SECoS [19, 22]. For a review of existing ECoS algorithms and applications, see [21].

While it seems that for many the term "evolving" evokes thoughts of evolutionary computation, ECoS are not evolutionary algorithms. ECoS networks do not use the mechanisms of evolutionary computation, such as fitness-based selection, reproduction and mutation. Instead, as far as ECoS networks are concerned, the word "evolving" has the much broader meaning of change through time. ECoS was designed around the following principles [7]:

1. The ECoS training algorithm is designed with the intention that all the algorithm can learn from the data is learned in the first training pass (one-pass learning). Additional exposure to the training data is not necessary.

*School of Earth and Environmental Sciences, University of Adelaide, SA 5005, Australia. e:mjwatts@ieee.org

2. ECoS are intended to be used in an on-line learning application. This means that new data will be constantly and continuously coming into the system, and that this data must be learned by the network without forgetting the old.

3. The general ECoS architecture and learning algorithm allows an ECoS network to accommodate new data without a catastrophic forgetting of the old.

4. The manner in which neurons are added to an ECoS means that some training examples are stored, initially, verbatim within the structure of the network. These examples are then either modified (refined) by exposure to further examples, or, depending upon the training parameters used, remain the same and can be later retrieved.

The advantages of ECoS are that they avoid the problems associated with traditional connectionist structures such as MLP [7, 10]: They are hard to overtrain, due to the constructive nature of their learning algorithm; They learn quickly, as the learning algorithm is a one-pass algorithm, that is, it requires only a single presentation of the data set; They are far more resistant to catastrophic forgetting than most other models, as new training data is represented by adding new neurons, rather than accommodating the additional data in the existing neurons.

ECoS networks also have several advantages over other constructive algorithms: Firstly, they are not limited to a particular application domain, they can be applied to both classification and function approximation; Secondly, they do not require multiple presentations of the training data set, as is the case with some of the constructive algorithms in existence, such as RAN [18] and GAL [1]; Finally, they are able to continue learning and are not restricted to learning a single training set as some other constructive algorithms such as RAN and Cascade Correlation [4] are.

Traditional ANN are supported by a large body of theory [12, 16, 3, 13]. This body of theory describes:

- How the ANN training algorithms behave, given the settings of their training parameters.

- How the training algorithms allow the network to capture knowledge.

- How this knowledge is represented by the ANN.

This theory assists the neural network practitioner in both applying these algorithms and in optimising and extending them. A theoretical basis is also useful in assisting the acceptance of a new algorithm: other researchers are more likely to utilise a new algorithm if its theoretical grounding is known.

It is for these reasons that a theoretical basis to ECoS is desirable. Any theory, or formalisation, that describes the ECoS algorithm must cover two distinct aspects.:

- The behaviour, or state, of the network at any time t.
- The way in which the state of the network changes as it trains, which includes the effect each training parameter has on the changes made to the ECoS by the training algorithm.

The chapter is arranged as follows: Section 2 describes ECoS algorithms. Section 3 describes the existing theory and describes objections to it. Section 4 introduces the foundations of the new ECoS formalisation. Section 5 builds on this foundation to describe the effect of the two training parameters that control the addition of neurons to ECoS, the sensitivity threshold and error threshold. Section 6 investigates the conjectures made about the sensitivity threshold parameters in Section 5, while Section 7 investigates the conjectures made about the error threshold parameter in Section 5. In Section 8 the learning rate one parameter is investigated, while the effect of the learning rate two parameter is investigated in Section 9. Differences in the behaviour of ECoS networks over classification and function approximation problems are discussed in Section 10. The predictions are tested in Section 11 across two well-known benchmark data sets. These results are discussed in Section 12. Finally, conclusions and future work are presented in Section 13.

2 ECoS Algorithms

An ECoS network is a multiple neuron layer, constructive artificial neural network. An ECoS network will always have at least one 'evolving' neuron layer. This is the constructive layer, the layer that will grow and adapt itself to the incoming data, and is the layer with which the learning algorithm is most concerned. The meaning of the connections leading into this layer, the activation of this layer's neurons and the forward propagation algorithms of the evolving layer all differ from those of classical connectionist systems such as MLP. For the purposes of this chapter, the term 'input layer' refers to the neuron layer immediately preceding the evolving layer, while the term 'output layer' means the neuron layer immediately following the evolving layer. This is irrespective of whether or not these layers are the actual input or output layers of the network proper. For example, in Figure 1, which shows a generic ECoS structure, the "input layer" could be the input layer proper of the network (as with SECoS networks) or it could be a neuron layer that processes the actual input values for presentation to the evolving layer (as with EFuNN networks). By the same token, the "output layer" could be the actual output layer of the network (as it is with SECoS) or a neuron layer that further processes the outputs of the evolving layer (as with EFuNN). The connection layers from the input neuron layer to the evolving neuron layer and from the evolving layer to the output neuron layer, are fully connected.

The activation A_n of an evolving layer neuron n is determined by Equation 1.

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

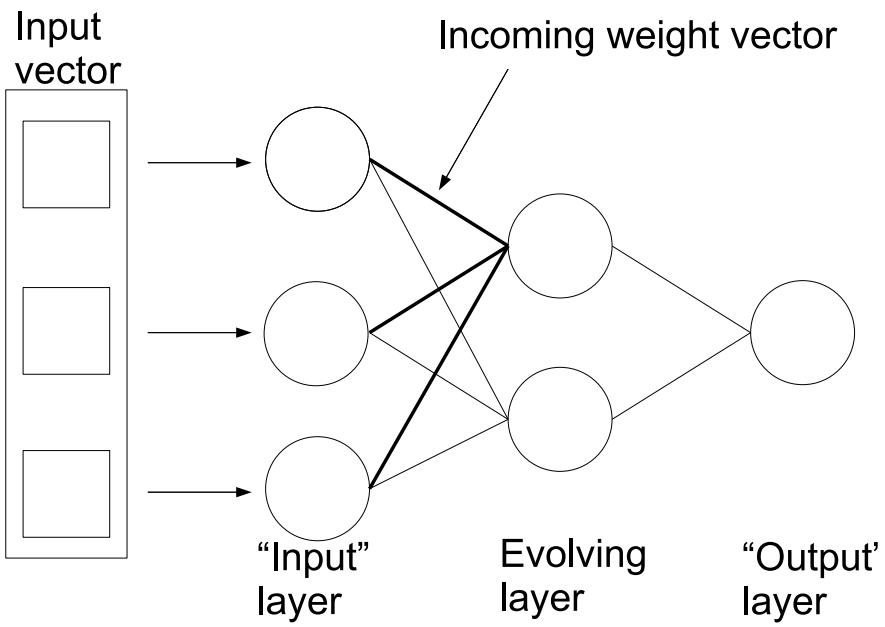

Figure 1: General ECoS architecture

where D_n is the distance between the input vector and the incoming weight vector for that neuron. Since ECoS networks are fully connected, it is possible to measure the distance between the current input vector and the incoming weight vector of each evolving-layer neuron. Although the distance can be measured in any way that is appropriate for the inputs, this distance function must return a value in the range of zero to unity. For this reason, most ECoS algorithms assume that the input data will be normalised, as it is far easier to formulate a distance function that produces output in the desired range if it is normalised to the range zero to unity.

Whereas most ANN propagate the activation of each neuron from one layer to the next, ECoS evolving layers propagate their activation by one of two alternative strategies. The first of these strategies, entitled *OneOfN* propagation, involves only propagating the activation of the most highly activated ("winning") neuron. The second strategy, *ManyOfN* propagates the activation values of those neurons with an activation value greater than the activation threshold A_{thr}.

The ECoS learning algorithm is based on accommodating new training examples within the evolving layer, 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 2. The addition of neurons to the evolving layer is driven by the novelty of the current train-

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

Figure 2: ECoS learning algorithm

ing example: if the current example is particularly novel (it is not adequately represented by the existing neurons) then a new neuron will be added. Four parameters are involved in this algorithm: the sensitivity threshold S_{thr}, the error threshold E_{thr}, and the two learning rates η_1 and η_2. The sensitivity threshold and error threshold both control the addition of neurons and when a neuron is added, its incoming connection weight vector is set to the input vector $\mathbf{I}$, and its outgoing weight vector is set to the desired output vector $\mathbf{O}_d$. The sensitivity and error thresholds are measures of the novelty of the current example. As can be seen in Figure 2, if the current example causes a low activation (that is, it is novel with respect to the existing neurons) then the sensitivity threshold will cause a neuron to be added that represents that example. If the example does not trigger the addition of a neuron via the sensitivity threshold, but the output generated by that example results in an output error that is greater than the error threshold (that is, it had a novel output), then a neuron will be added.

The weights of the connections from each input i to the winning neuron j are modified according to Equation 2.

$$W_{i,j}(t+1) = W_{i,j}(t) + \eta_1(I_i - W_{i,j}(t)) \tag{2}$$

where:
$W_{i,j}(t)$ is the connection weight from input i to j at time t
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 3.

$$W_{j,o}(t+1) = W_{j,o}(t) + \eta_2 A_j E_o \tag{3}$$

where:
$W_{j,o}(t)$ is the connection weight from j to output o at time t

A_j is the activation of j
E_o is the signed error at o, as measured according to Equation 4.

$$E_o = O_o - A_o \tag{4}$$

where:
O_o is the desired activation value of output o
A_o is the actual activation of o.

This is essentially the perceptron learning rule. From this it becomes apparent that in [8] and subsequent publications [11, 23] the terms O_d and A_o above were incorrectly reversed.

3 Previous Formalisation

The following theory was proposed by Kasabov in [7, 9, 10] to describe the state and training of an ECoS network (specifically an EFuNN). Note that the notation used for the training parameters has been altered to bring it into line with those used in this chapter:

> Each rule neuron, e.g. r_j, represents an association between a hyper-sphere from the fuzzy input space and a hyper-sphere from the fuzzy output space, the $W1(r_j)$ connection weights representing the co-ordinates of the center of the sphere in the fuzzy input space, and the $W2(r_j)$-the co-ordinates in the fuzzy output space. The radius of an input hyper-sphere of a rule neuron is defined as $(1 - S_{thr})$ ... For example, two pairs of fuzzy input-output data vectors $d1 = (Xd1, Yd1)$ and $d2 = (Xd2, Yd2)$ will be allocated to the first rule neuron r_1 if they fall into the r_1 input sphere and in the r_1 output sphere, i.e. the local normalised fuzzy distance between $Xd1$ and $Xd2$ is smaller than the radius r and the local normalised fuzzy difference between $Yd1$ and $Yd2$ is smaller than an error threshold E_{thr}. [9, pg 304]

On the topic of adaptation of the existing neurons, Kasabov goes on to say:

> Through the process of associating (learning) of new data point to a rule neuron, the centers of this neuron *(sic)* hyper-spheres adjust in the fuzzy input space depending on a learning rate η_1, and in the fuzzy output space depending on a learning rate η_2. [9, pg 304]

There are several objections to this theory. Firstly, the radius of the hyper-spheres is defined as $1 - S_{thr}$. The sensitivity threshold, however, is a property of the training algorithm, not of the evolving layer neurons. Although the example that caused the addition of that neuron will be within that radius, subsequent examples that cause the neuron to fire will not be. Also, if the radius of the hyper-spheres were dependent upon S_{thr}, then the radius of all hyper-spheres would be identical and only the learning rate parameters would

have any effect upon training. Experimental results (Section 11) show that this is not the case. Secondly, the suggestion that the region defined by a neuron is a hyper-sphere is not supported by the canonical ECoS algorithm. ECoS evolving layer neurons are unthresholded, thus a neuron will activate if it is the closest in the input space to the current example, *no matter how distant the example actually is*. As long as the neuron is the closest, it will activate for that example, even if the distance is greater than $1 - S_{thr}$. Which neuron is closest to the example, however, depends on the co-ordinates of the other neurons in the evolving layer. Since these may be distributed in any manner within the input space, the boundaries between the regions defined by each neuron are not regular, that is, the polygons defined by the points represented by each neuron in the evolving layer of an ECoS network are not regular polygons, nor are they necessarily of regular size. Plainly, then, neither the description of the regions defined by neurons as hyper-spheres, nor the definition of these hyper-spheres, is appropriate for this purpose.

Another problem with this theory is that it does not describe the effect of the training parameters. Although experimental results [20] show that the parameters have different effects upon the behaviour of an ECoS network, the theory above does not describe this in any way.

Finally, the theory is untestable: it makes no predictions about the behaviour of the network or training algorithm as parameters are altered.

Some elements of the theory are satisfactory, however: each neuron in the evolving layer does provide a mapping, or association, from a region of input space to a region of output space, and the coordinates of these regions in the input space are defined by the connection weights of the neuron. These elements will be retained in the improved theory proposed in this chapter.

4 A New Formalisation of ECoS

With the existing theory unsatisfactory, it becomes necessary to formulate a new theory that overcomes the shortcomings elucidated above. Three assumptions are made:

1. That the ECoS network has one layer of connections coming into the evolving neuron layer, and one layer of connections going out of the evolving layer.

2. That the distance between two vectors will be measured so that the distance is in the range of zero to unity.

3. That the evolving layer neurons are not thresholded.

Within this chapter, the term *region* is intended to mean a set of points in n-dimensional space that is defined by specific boundaries. The term *volume* means the amount of unit space occupied by a region.

This theory will be in two parts: the state of the network at a time t; and the behaviour of the network in relation to the training parameters, that is, the

way in which each training parameter effects the training process. The first set of theory here describes the state of the network.

4.1 Axioms of State

The following axioms are the basis of this theory. They are partially derived from an examination of the forward propagation algorithm of ECoS: some are also retained from the previous theoretical work described above. These axioms describe the way in which an ECoS network encapsulates what it has learned about the input space, that is, how an ECoS network represents knowledge.

Axiom 1 *each neuron n in the evolving layer of an ECoS network defines a single point in the input space.*

This is self evident from the ECoS algorithm. Since the activation of any neuron is based on the distance between the current input example and its incoming weights, each neuron therefore represents a single point in the input space.

Axiom 2 *the activation of a neuron n for an example* **I** *is proportional only to the distance of* **I** *from n.*

This also is self evident from the ECoS algorithm, specifically from Equation 1.

Axiom 3 *a neuron n will activate (win) iff its activation is greater than all other neurons in the evolving layer.*

Assuming the case of one-of-n activation, a neuron may propagate its activation to following layers only if it is the most highly activated. To be the most highly activated with a distance-based activation function, the point defined by the neuron in Axiom 1 must be the closest to **I**.

Axiom 4 *for every neuron j there is a region R_j with volume V_j in input space within which each point is closer to n than to any other neuron.*

This is self evident: in any plot with multiple points, there will be a region around each point that is closer to that point than any other. In effect, each neuron in the evolving layer corresponds to a Voronoi polygon in the input space [17]. This is consistent with the GAL algorithm [1], which is the constructive algorithm most similar to ECoS. In [1, pg 399], it is stated that:

> The input space is divided in the form of a Voronoi tessellation where exemplar units' domination regions are bounded by hyperplanes that pass through the medians of the two closest exemplar units.

Thus, an ECoS network with m neurons will partition the input space into m Voronoi regions, where a neuron will activate if an input example is at a point within the Voronoi region for that neuron. Figure 3 shows the Voronoi regions

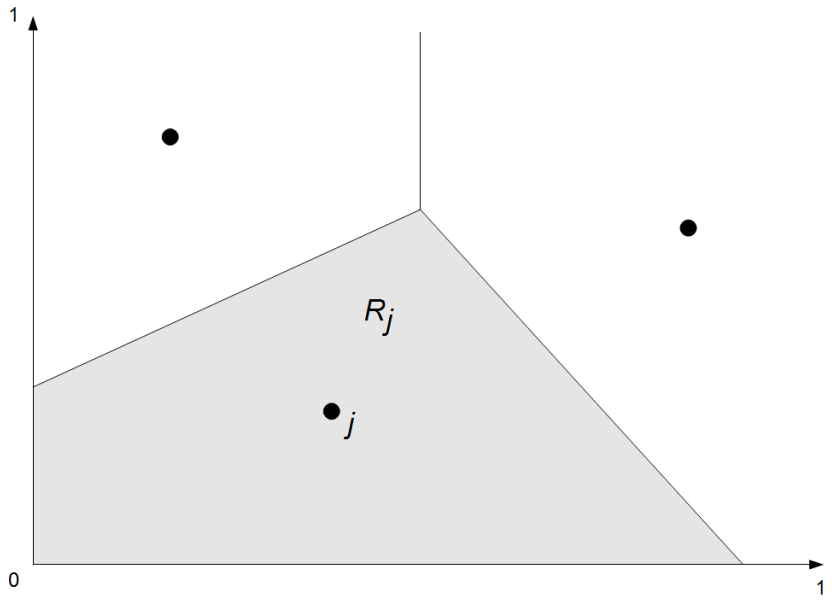

Figure 3: Voronoi regions defined by the evolving layer neurons of an ECoS network.

defined in two dimensional input space by a hypothetical ECoS network with three evolving layer neurons. The Voronoi region defined by the winning neuron j is shaded.

In Figure 4 the Voronoi regions of a SECoS network trained on the two spirals data set [14] is presented, where regions for neurons that represent the second of the two classes have been filled in. A basic spiral shape has become apparent, but is slightly coarser than for other constructive algorithms (see, for example, [6]).

From the above axioms and results, it is possible to infer some properties of ECoS networks. Assume that the task at hand is a classification problem of c classes, using an ECoS network with m neurons in the evolving layer, and the examples being presented to the network are uniformly distributed in input space.

The probability of an unknown example $\mathbf{I}$ of class C being correctly classified is determined by the probability of $\mathbf{I}$ falling in a region 'owned' by a neuron in the set of neurons M_C that represents C. In the general case this is determined by Equation 5.

$$P_t = \frac{\sum V_i}{\sum V_m} \tag{5}$$

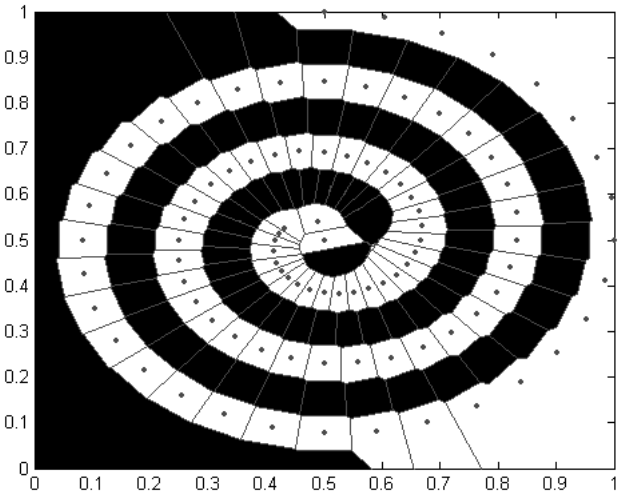

Figure 4: Voronoi regions of an ECoS network trained on the two spirals data set.

where:
P_t is the probability of correctly classifying **I** as being a member of class C
V_i is the volume of region i, where $i \in M_C$
V_m is the volume of region m, where m is the set of all neurons in the evolving layer of the network. Since the distance measures used in ECoS must return values between zero and unity, the sum of all neurons regions volumes must be unity. Thus, Equation 5 simplifies to:

$$P_t = \sum V_i \tag{6}$$

With uniformly distributed examples, the volume of each of the m regions will be equal. Thus P_t is proportional to the number of neurons allocated for each class. This implies that in the case of an unbalanced training set, that is, a training set with a large number of examples of one class and smaller numbers of other classes, a network will be produced that is less likely to generalise well to the under-represented class.

For non-uniformly distributed examples, multiple classes that are tightly clustered together will cause each neuron to have a very small region associated with it. This will require a larger number of neurons to model the problem. That is, more complex problems will require more neurons.

5 Influence of the Neuron Addition Parameters

Following this line of reasoning, it is apparent that there are two regions to be considered. The first is defined by the sensitivity threshold S_{thr} and will be denoted by R_s, which has the volume V_s. The second is defined by the error threshold E_{thr}, which will be denoted by R_e and has the volume V_e. The region R_a is therefore the intersection of the region defined by these two regions and region R_j, that is:

$$R_a = R_j \cap (R_s \cap R_e) \tag{7}$$

If it is assumed that the training examples are uniformly distributed through the input space, and that any training input vectors that lie outside of R_a will cause the addition of a neuron, then the probability of any training example within the neurons Voronoi region R_j causing the addition of a neuron, is equivalent to the ratio between the volume V_j of R_j and the volume V_a of R_a:

$$P_a = \frac{V_j - V_a}{V_j} \tag{8}$$

If the examples are non-uniformly distributed in input space, then the relationship between P_a and V_j will be non-linear. In either case, however, it is apparent that the following relationship is true [1]:

$$V_a \to 0, P_a \to 1$$

Since V_a is defined as the volume of R_a, and R_a is defined as the intersection of R_s and R_e, it can be deduced that:

$$\begin{aligned} V_s \to 0, V_a \to 0 \\ V_e \to 0, V_a \to 0 \\ \Rightarrow V_s \to 0, P_a \to 1 \\ \Rightarrow V_e \to 0, P_a \to 1 \end{aligned}$$

Since V_s and V_e are respectively determined by S_{thr} and E_{thr} it is conjectured that:

$$S_{thr} \to 1, P_a \to 1 \tag{9}$$

and:

$$E_{thr} \to 0, P_a \to 1 \tag{10}$$

No assumptions are made that these relations are linear. The relationship between each parameter and the rate at which neurons are added to the evolving layer will be influenced by the distribution of the training data in the input space, as well as the other training parameters.

[1] It can also be conjectured from this equation that an ECoS network will gain neurons rapidly during the early phases of training, but more slowly later on: as the network has few neurons initially, the volume assigned to each neuron is very large, thus the difference between V_j and V_a is very large and P_a is very high.

6 Influence of the Sensitivity Threshold Parameter

The hyper-sphere defined by S_{thr} is defined by a single distance, denoted here as D_s. The value of D_s can be derived simply from the activation equation of the neuron, $A_j = 1 - D_j$, where D_j is the distance between j and the example $\mathbf{I}$, by replacing D_j with D_s and rearranging to make D_s the dependent variable:

$$D_s = 1 - S_{thr}$$

Since V_a is a function of the distance D_s, $V_a = f(D_s)$, P_a is therefore also a function of D_s. Thus:

$$\begin{aligned} S_{thr} \rightarrow 1, D_s \rightarrow 0 \\ \Rightarrow S_{thr} \rightarrow 1, V_a \rightarrow 0 \\ \Rightarrow S_{thr} \rightarrow 1, P_a \rightarrow 1 \end{aligned}$$

In other words, as the sensitivity threshold increases, so too does the probability of a neuron being added to the network. This proves the conjecture in Equation 9 and is consistent with experimental results (Section 11).

The region R_s as defined by D_s around j is displayed as the shaded region in Figure 5. This figure shows the Voronoi regions defined in a two dimensional input space by a hypothetical ECoS network with three neurons in the evolving layer.

7 Influence of the Error Threshold Parameter

Before considering the influence of the error threshold parameter, it is necessary to consider the activation A_o of an output neuron o. There are three cases to consider for the activation of an output neuron:

$$A_o = \begin{cases} 0, & \mathbf{W}_{j,o} = 0 \\ \mathbf{W}_{j,o} A_j, & 0 < A_j \mathbf{W}_{j,o} < 1 \\ 1, & A_j \mathbf{W}_{j,o} \geq 1 \end{cases} \tag{11}$$

Proving the conjecture in Equation 10 therefore requires a proof for each of these three cases.

For the case of $\mathbf{W}_{j,o} = 0$, then A_o will be zero, no matter the activation A_j. Thus, the distance between $\mathbf{I}$ and j is indeterminable from this relation. Instead, P_a can be deduced directly. For an activation $A_o = 0$, the error threshold will trigger the addition of a neuron if the desired output $\mathbf{O}_d$ is greater than the error threshold. Therefore:

$$P_a = \begin{cases} 1, & \mathbf{O}_d > E_{thr} \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

This is consistent with the conjecture in Equation 10.

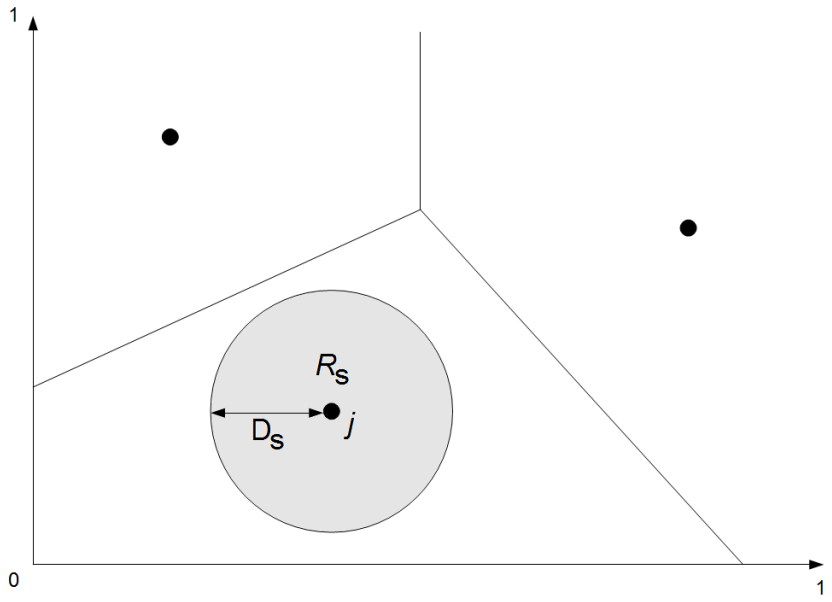

Figure 5: Distance and region R_s (shaded region) defined by the sensitivity threshold training parameter. Training examples that lie within R_s will not cause a neuron to be added.

For the case of $0 < A_j \mathbf{W}_{j,o} < 1$, the way in which the error is calculated must be examined. The error E_o across output o between the desired output value $\mathbf{O}_d$ and the actual output value A_o is defined as:

$$E_o = |\mathbf{O}_d - A_o| \tag{13}$$

which can be expanded as:

$$E_o = \begin{cases} \mathbf{O}_d - A_o, & \mathbf{O}_d > A_o \\ A_o - \mathbf{O}_d, & \mathbf{O}_d < A_o \end{cases} \tag{14}$$

Thus, there are two situations where E_o could exceed E_{thr}: when A_o is too low, that is, the example $\mathbf{I}$ is too far from n; or when A_o is too high, that is, $\mathbf{I}$ is too close to n. There are therefore two activations to consider: A_o^{max}, that is too high, and A_o^{min} that is too low. These activations can be determined by substituting E_{thr} for E_o and rearranging Equation 14 to make A_o the dependent variable, as follows:

$$\begin{aligned} A_o^{min} &= \mathbf{O}_d - E_{thr} \\ A_o^{max} &= \mathbf{O}_d + E_{thr} \end{aligned} \tag{15}$$

For the case that $0 < A_j \mathbf{W}_{j,o} < 1$, then the terms of Equation 15 above can be expanded as:

$$A_j^{min} \mathbf{W}_{j,o} = \mathbf{O}_d - E_{thr}$$
$$A_j^{max} \mathbf{W}_{j,o} = \mathbf{O}_d + E_{thr}$$

where A_j^{max} is the maximum activation of winning evolving layer neuron j that will cause the maximum output activation A_o^{max} and A_j^{min} is the minimum activation of j that will cause the minimum output A_o^{min}

These can be rearranged to make A_j^{max} and A_j^{min} the dependent variables, as follows:

$$A_j^{max} = \frac{\mathbf{O}_d + E_{thr}}{\mathbf{W}_{j,o}}$$
$$A_j^{min} = \frac{\mathbf{O}_d - E_{thr}}{\mathbf{W}_{j,o}}$$

Given that Equation 1 describes the activation of j with respect to the distance D, then expanding A_j and rearranging to make D^{min} and D^{max} the dependent variables yields:

$$D^{min} = 1 - \frac{\mathbf{O}_d + E_{thr}}{\mathbf{W}_{j,o}} \tag{16}$$

$$D^{max} = 1 - \frac{\mathbf{O}_d - E_{thr}}{\mathbf{W}_{j,o}} \tag{17}$$

Thus, points that lie at a distance between D^{min} and D^{max} will not cause a neuron to be added. Figure 6 shows this. This figure shows the same three Voronoi regions from Figure 5, where the shaded region is R_e.

While it may seem that the terms $\mathbf{O}_d + E_{thr}$ and $\mathbf{O}_d - E_{thr}$ in Equations 16 and 17 could yield distances of less than zero or greater than one, from Equation 14 it is apparent that, provided that $\mathbf{O}_d$ and E_{thr} are both constrained to the range zero to unity, this cannot happen. From these equations the following constraints can be derived:

$$\text{if } D = D^{min} \text{ then } \mathbf{O}_d \leq 1 - E_{thr}$$
$$\text{if } D = D^{max} \text{ then } \mathbf{O}_d \geq E_{thr}$$

It is apparent from Equations 16 and 17 that:

$$E_{thr} = 0, D^{min} = D^{max} = 1 - \frac{\mathbf{O}_d}{\mathbf{W}_{j,o}}$$
$$\Rightarrow E_{thr} \rightarrow 0, D_e^{min} \rightarrow D_e^{max}$$

Since the volume V_e of the region R_e is given by:

$$V_e = f(D_e^{max}) - f(D_e^{min})$$

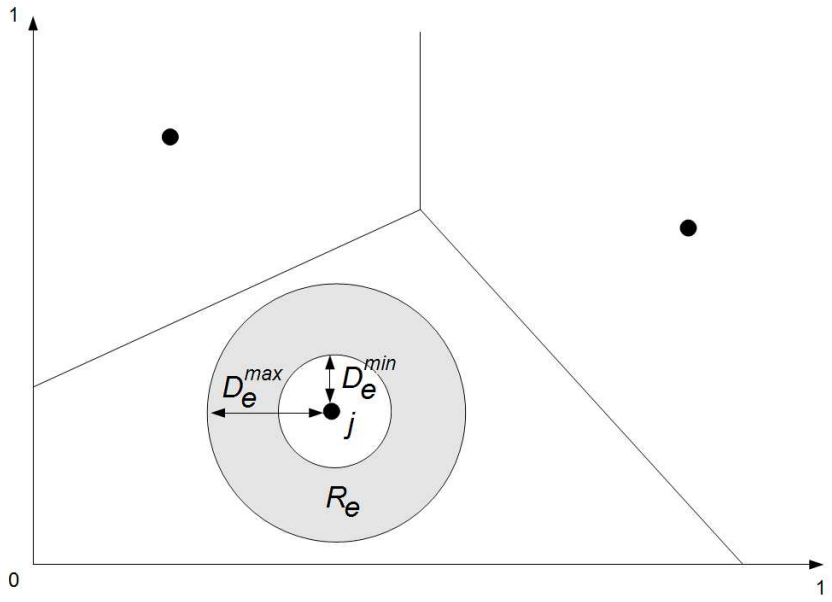

Figure 6: Distances and region R_e defined by error threshold parameter. Training examples that lie within R_e will not cause a neuron to be added.

where f is the volume function, then the following is implied:

$$\begin{aligned} E_{thr} \to 0, D_e^{min} \to D_e^{max} \\ \Rightarrow E_{thr} \to 0, V_e \to 0 \\ \Rightarrow E_{thr} \to 0, V_a \to 0 \\ \Rightarrow E_{thr} \to 0, P_a \to 1 \end{aligned}$$

In other words, as the error threshold decreases, the probability of a neuron being added increases. This proves the conjecture in Equation 10 for this case.

Additionally, if the distance D_s is less than the minimum distance D_e^{min}, then a neuron will be added for every training example. This is verified by Equations 7 and 8. If the intersection between the two regions is zero, then $P_a = 1$.

The final case is when $A_j w_{j,o} \geq 1$. Note that this requires $\mathbf{W}_{j,o} \geq 1$, although not every output will be unity whenever $\mathbf{W}_{j,o} \geq 1$. In this case, the error will exceed the error threshold only when $\mathbf{O}_d$ is less then $1 - E_{thr}$, thus:

$$P_a = \begin{cases} 1, & \mathbf{O}_d < 1 - E_{thr} \\ 0, & \text{otherwise} \end{cases} \tag{18}$$

The conjecture in Equation 10 again holds true.

The distance D_e can be derived by rearranging $A_j \mathbf{W}_{j,o} \geq 1$ to make A_j the dependent variable and substituting D_e, thus:

$$D_e \leq 1 - \frac{1}{\mathbf{W}_{j,o}} \tag{19}$$

Therefore, the maximum distance that an example can be at for this case is determined solely by the connection weight. If the following is the case

$$D_e > 1 - \frac{1}{\mathbf{W}_{j,o}}$$

then the neuron activation will not saturate and Equations 16 and 17 will apply.

8 Influence of the Learning Rate One Parameter

The learning rate parameters also have an effect upon training, although each parameter has a different effect. This is due to the different mechanisms by which they function. The weight update rule for the input to evolving layer of connections is intended to reduce the difference between the current weight vector and the current input vector. The weight update rule for the evolving to output layer is a variant of the perceptron learning rule, and is based on the idea of reducing errors for the outputs.

Intuitively, the higher the η_1 parameter is, the higher the activation of j will be the next time the current vector $\mathbf{I}$ is presented to it. The conjecture for this parameter is thus:

$$\begin{array}{l} \eta_1 \rightarrow 1, A_j^{t+1} \rightarrow 1 \\ \eta_1 \rightarrow 0, A_j^{t+1} \rightarrow A_j^t \end{array} \tag{20}$$

where A_j^{t+1} is the activation of neuron j at time $t+1$ for input vector $\mathbf{I}$. This conjecture can be proven as follows:

The weight update rule for the input to evolving layer weights can be expressed as:

$$\mathbf{W}_{i,j}(t+1) = \mathbf{W}_{i,j}(t) + \Delta \mathbf{W}_{i,j}$$

where:

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

This can be viewed as a change in distance between $\mathbf{W}$ and $\mathbf{I}$. The goal is thus to prove that when $\eta_1 = 1$ the change in distance between the two vectors $\mathbf{W}$ and $\mathbf{I}$ is such that the distance at time $t+1$, D_j^{t+1}, is zero.

Given the following:

$$A_j^t = 1 - D_j^t \tag{21}$$

$$A_j^{t+1} = 1 - D_j^{t+1} \tag{22}$$

$$D_j^{t+1} = D_j^t - \Delta D_j^t \tag{23}$$

From the weight update rule above, it can be seen that:

$$\Delta D_j^t = \eta_1 D_j^t \tag{24}$$

Rearranging Equation 21 to make D_j^t the dependent variable, and substituting for D_j^t in Equation 24 yields:

$$\Delta D_j^t = \eta_1 (1 - A_j^t) \tag{25}$$

while substituting for ΔD_j^t in Equation 23 gives the following:

$$D_j^{t+1} = D_j^t - \eta_1 (1 - A_j^t) \tag{26}$$

Rearranging Equation 22 to make D_j^{t+1} the dependent variable yields:

$$D_j^{t+1} = 1 - A_j^{t+1} \tag{27}$$

Replacing D_j^{t+1} in Equation 26 with Equation 27 yields:

$$1 - A_j^{t+1} = (1 - A_j^t) - \eta_1 (1 - A_j^t)$$

Finally, solving for A_j^{t+1} and simplifying yields:

$$A_j^{t+1} = A_j^t + \eta_1 (1 - A_j^t) \tag{28}$$

This holds true for any monotonic linear distance measure.

It can be seen that when $\eta_1 = 1$ Equation 28 becomes:

$$\begin{aligned} &A_j^{t+1} = A_j^t + (1 - A_j^t) \\ &\Rightarrow A_j^{t+1} = 1 \end{aligned}$$

When $\eta_1 = 0$, Equation 28 becomes:

$$\begin{aligned} &A_j^{t+1} = A_j^t + 0(1 - A_j^t) \\ &\Rightarrow A_j^{t+1} = A_j^t \end{aligned}$$

This is also true for non-monotonic distance measures, but is not proven. This objection aside, the conjecture in Equation 20 above is proven

From Equation 25 above, it is also possible to calculate the maximum distance that a neuron will move. A neuron moves the maximum distance when its activation is the minimum allowed. The minimum allowed activation is set by the sensitivity threshold parameter, S_{thr}. Thus, by substituting S_{thr} into Equation 25 above, we get:

$$\Delta D^{max} = \eta_1 (1 - S_{thr}) \tag{29}$$

Thus, the effect of the learning rate one parameter is determined by the sensitivity threshold parameter.

9 Influence of the Learning Rate Two Parameter

The change to the connection weight $\mathbf{W}_{j,o}$, $\Delta\mathbf{W}_{j,o}$, is determined by the following equation:

$$\Delta\mathbf{W}_{j,o} = \eta_2 E_o A_o \tag{30}$$

where:
η_2 is the learning rate two parameter,
E_o is the absolute error over output neuron o and
A_o is the activation of neuron o

From this equation, a relationship with the error threshold becomes immediately apparent: since the weights will only be updated if a neuron is *not* added to the network, then E_o in Equation 30 above will always be less than the error threshold E_{thr}. Thus, a limit on the value of $\Delta\mathbf{W}_{j,o}$, $\Delta\mathbf{W}_{j,o}^{max}$ can be calculated:

$$\Delta\mathbf{W}_{j,o}^{max} = \eta_2 E_{thr} A_o$$

This shows that the error threshold parameter and the performance of the η_2 parameters are closely coupled together. A high error threshold will reduce the number of neurons added by two different mechanisms: firstly, by reducing the sensitivity of the network to the error over the current example, and secondly, by reducing its sensitivity to error for later examples. Error threshold is thus a very significant training parameter.

The sensitivity threshold is also relevant. The activation of A_o is a function of A_j and $W_{j,o}$, and the minimum activation of j is determined by S_{thr}, the lower limit of $\Delta\mathbf{W}_{j,o}$, $\Delta\mathbf{W}_{j,o}^{min}$ can be calculated:

$$\Delta\mathbf{W}_{j,o}^{min} = \eta_2 E_o w_{j,o} S_{thr}$$

Since ECoS networks are intended to learn for the duration of their existence, it is entirely possible that the weights in this layer of connections will become very large: with an unending (or infinite) stream of training data, it is entirely possible that the weights themselves will approach infinity, especially with a low sensitivity threshold and a high error threshold.

An interesting implication of the unbounded growth of weights can be described using Equations 16, 17 and 19. As the weights continue to grow, the distance values in these equations tend ever closer to unity, that is:

$$\mathbf{W}_{j,o} \to \infty, D_e \to 1$$

Given the following:

$$D_e \to 1, P_a \to 0$$

Since the rate at which the weights grow is directly determined by the η_2 parameter, then it is clear that as the η_2 parameter increases, the probability of adding a neuron later decreases. Over a complete training set, then, an ECoS network will be expected to add fewer neurons during training with a high

learning rate. This has been experimentally verified (Section 11). Note that this applies to classification problems only. Function approximation problems behave differently, and are discussed in Section 10.

10 Function Approximation

The analysis of the effect of the sensitivity threshold and error threshold apply to all types of problems, no matter whether they are classification or function approximation problems. To explain the effect of the learning rate parameters for function approximation problems requires a slight reworking of parts of the formalisation.

Assume that an evolving layer neuron activates with a value of unity. The activation of the output neuron o will therefore be equal to the value of the connection weight $\mathbf{W}_{j,o}$. If $\mathbf{W}_{j,o}$ is greater than unity, then the output neuron o will saturate at unity. This will not cause a problem if this occurs for a classification problem, as the output values will be either zero or unity. If a neuron activates at less than unity, and the connection weight outgoing from j is equal to or greater than unity, then the error will still be low. Thus, a neuron that is not close to the current example can still correctly classify it, if its outgoing weight is sufficiently large. This means that for classification problems, a smaller number of neurons is possible, especially if E_{thr} and S_{thr} are low.

For function approximation problems, it is not possible for the evolving to output layer connection weights to exceed unity. This is because these weights represent the desired output values. Thus, an error that is less than the error threshold, will cause the weight to move towards the desired output value. This means that each outgoing connection weight has a region around it on the number line, where the bounds are defined by the error threshold. The neuron then represents a cluster of examples, with similar output values. Any input vector that causes that particular evolving layer neuron to activate, and that has a desired output value within the range $\mathbf{W}_{j,o} \pm E_{thr}$ will not cause a neuron to be added.

A high η_2, however, will cause the output values to move away from the centre of the cluster of output values. If the data is self-consistent, that is, the values are periodic, then additional neurons will be added, driven by the error threshold parameter. However, if η_2 is too low, then the neuron will not be able to find the centre of the cluster of output values. This will also cause more neurons to be added.

11 Experiments

The formalisations above made several predictions about the effect of the training parameters, insofar as they relate to the addition of neurons to the ECoS evolving layer. Firstly, that as the sensitivity threshold increases, the number

of neurons added increases; secondly, that as the error threshold decreases, the number of neurons added increases; thirdly, for classification problems, as the learning rate two parameter increases, the number of neurons added decreases, while for function approximation problems, as the learning rate two parameter increases, the number of neurons will first decrease, then increase. The ways in which these three parameters behave has been discussed from a theoretical viewpoint (Sections 6, 7, 9 and 10). The purpose of the work reported in this section was to experimentally investigate these predictions.

Two well-known benchmark data sets were used in these experiments. The first was Fisher's iris classification data set [5]. The second was the Gas Furnace function approximation problem [2]. There were 150 examples in the iris data set, and 289 examples in the gas furnace set.

Two techniques were used for investigating the effects of the parameters. The effects of the sensitivity threshold and error threshold were investigated using Latin hypercube sampling [15]. Parameters were randomly drawn from one thousand partitions, and a network (either EFuNN or SECoS) trained over the test data set. The number of neurons in the evolving layer was then determined. Since the effects of error threshold and sensitivity threshold tended to overwhelm the effects of the η_2 parameter, a slightly different method was used to investigate the parameter. For these experiments, the values of the sensitivity threshold, error threshold and η_1 were held constant at 0.5, 0.1 and 0.5, respectively. Only the η_2 parameter was randomly sampled across the range $[0, 1]$. Again, one thousand random samples were drawn.

The results are presented and discussed in the following subsections.

11.1 Sensitivity Threshold

The analysis in Section 6 predicted that the number of neurons added would increase as the sensitivity threshold approached unity. The results presented here show that this is the case for both SECoS and EfuNN across both data sets.

The results across the iris data set are presented in Figure 7. Here, there is a very strong trend apparent for both SECoS and EFuNN, with a minimal rise in the number of evolving layer neurons between zero and 0.9, and a sharp rise after 0.9, quickly approaching 150 neurons in total, or one for each example in the data set. There are a relatively small number of samples where the number of neurons is very much greater than most for that parameter value, that is, there are a number of points well above the rest. These are from runs where the error threshold was randomly set to such a low value that the effects overwhelmed the effects of the sensitivity threshold. The number of neurons added to the EFuNN tended to be slightly larger than the number added to the SECoS.

The predictions were again verified over the gas furnace data, with the upward swing in numbers occurring at almost the same point as the iris data set. Again, values of the sensitivity threshold near unity caused a neuron to be added for every example. The majority of the samples were lower in the plot than for the iris data set, with more of the samples tightly grouped together near the

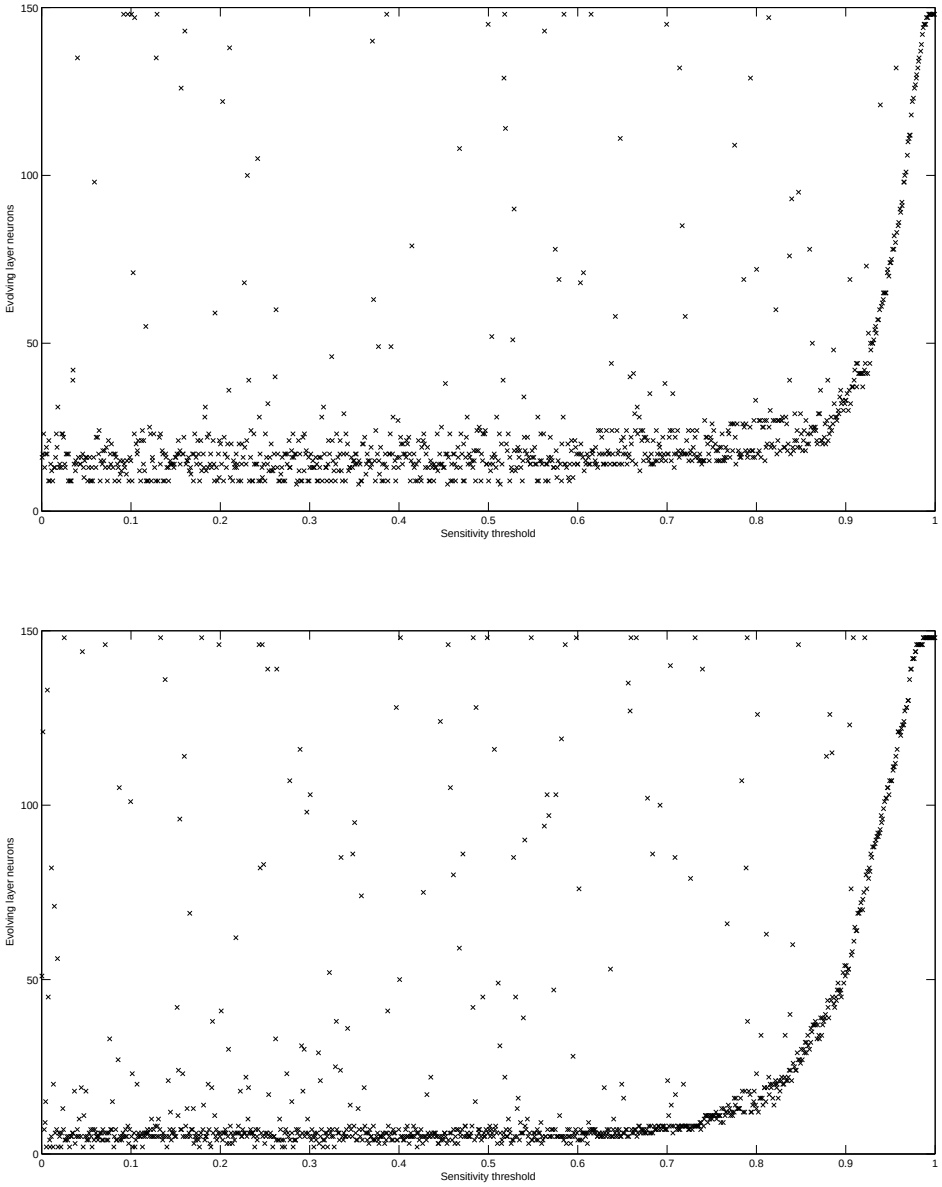

Figure 7: Number of evolving layer neurons versus sensitivity threshold parameter for SECoS (top) and EFuNN (bottom) networks trained on the iris classification data set.

x-axis. That is, there was much less variation in the size of the evolving layer than there was for the iris data set. Again, some samples were much higher than others about that parameter value, which was again due to the overwhelming effect of the error threshold for that sample. The differences in size between SECoS and EFuNN are more apparent here than with the iris data: once again, EFuNN had more neurons added during training.

11.2 Error Threshold

The analysis in Section 7 predicted that the number of neurons added to an ECoS network would increase as the error threshold approached zero. The results in this subsection show that this was the case for both SECoS and EFuNN across both data sets.

The results for the iris classification benchmark are presented in Figure 9. Near zero, a neuron was added for almost every example, and the numbers added dropped rapidly as the error threshold approached 0.1, then decreased much more slowly as the threshold approached unity. As before, there were a relatively small number of samples where the randomly-set value of the sensitivity threshold was large enough to overwhelm the effect of the error threshold. Again, EFuNN were slightly larger than SECoS.

The results across the gas furnace set are presented in Figure 10, where the networks again behaved as predicted. The curves appear to be much smoother than those for the iris classification problem. The plateau in the curve occurred slightly later than for iris classification, at an error threshold of approximately 0.2, as opposed to 0.1 for iris. As before, the number of evolving layer neurons were on the whole greater for EFuNN than for SECoS.

11.3 Learning Rate Two

Two predictions were made about the effect of the learning rate two parameter. The first was that for classification problems, the number of neurons added will decrease as the learning rate two parameter increases. For function approximation problems, the number of neurons will decrease up until a certain point, then start to increase again.

The results for the iris classification data set are presented in Figure 11. As predicted, there is a downward trend to the curve, as the number of neurons decreased with the increasing learning rate two parameter. The decrease for SECoS ended at a lower error threshold than for EFuNN, at approximately 0.4 as opposed to 0.5.

The results for the gas furnace data set are presented in Figure 12. For SECoS, a clear decrease in the number of neurons is visible from zero to 0.4, then an increase from 0.6 to unity. Although the corresponding plot for EFuNN is much less smooth than for SECoS, a definite decrease in the number of neurons is visible between 0.1 and 0.5, followed by an increase from 0.6 to unity.

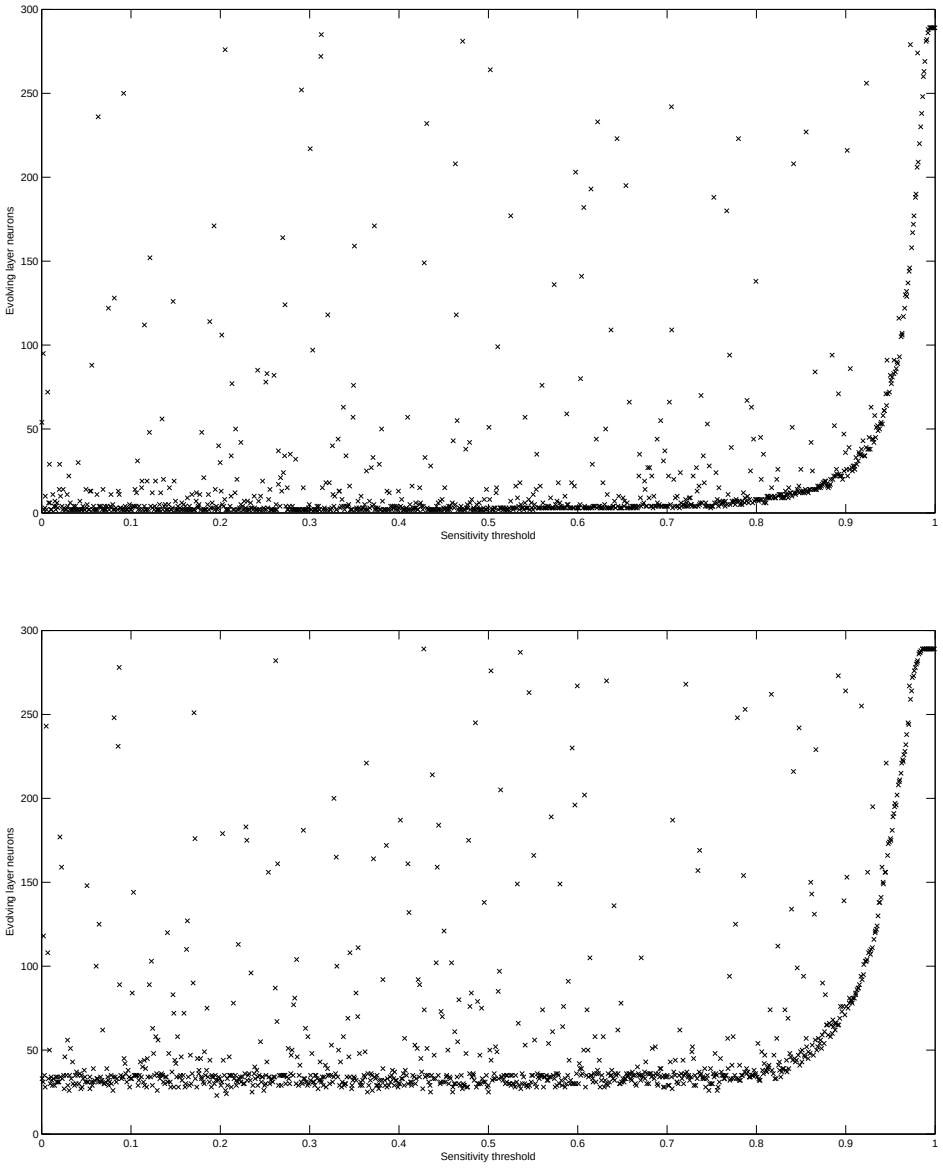

Figure 8: Number of evolving layer neurons versus sensitivity threshold parameter for SECoS (top) and EFuNN (bottom) networks trained on the gas furnace data set.

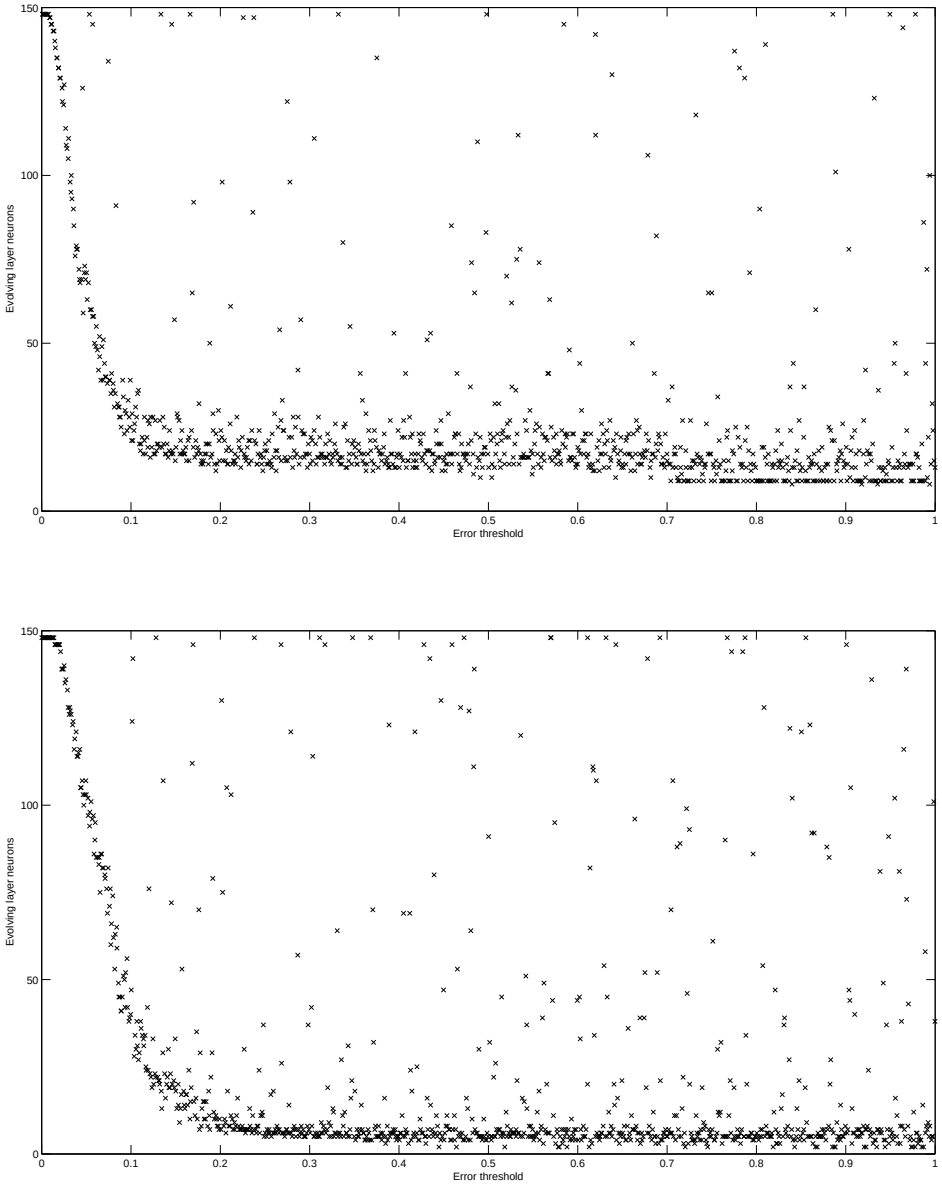

Figure 9: Number of evolving layer neurons versus error threshold parameter for SECoS (top) and EFuNN (bottom) networks trained on the iris classification data set.

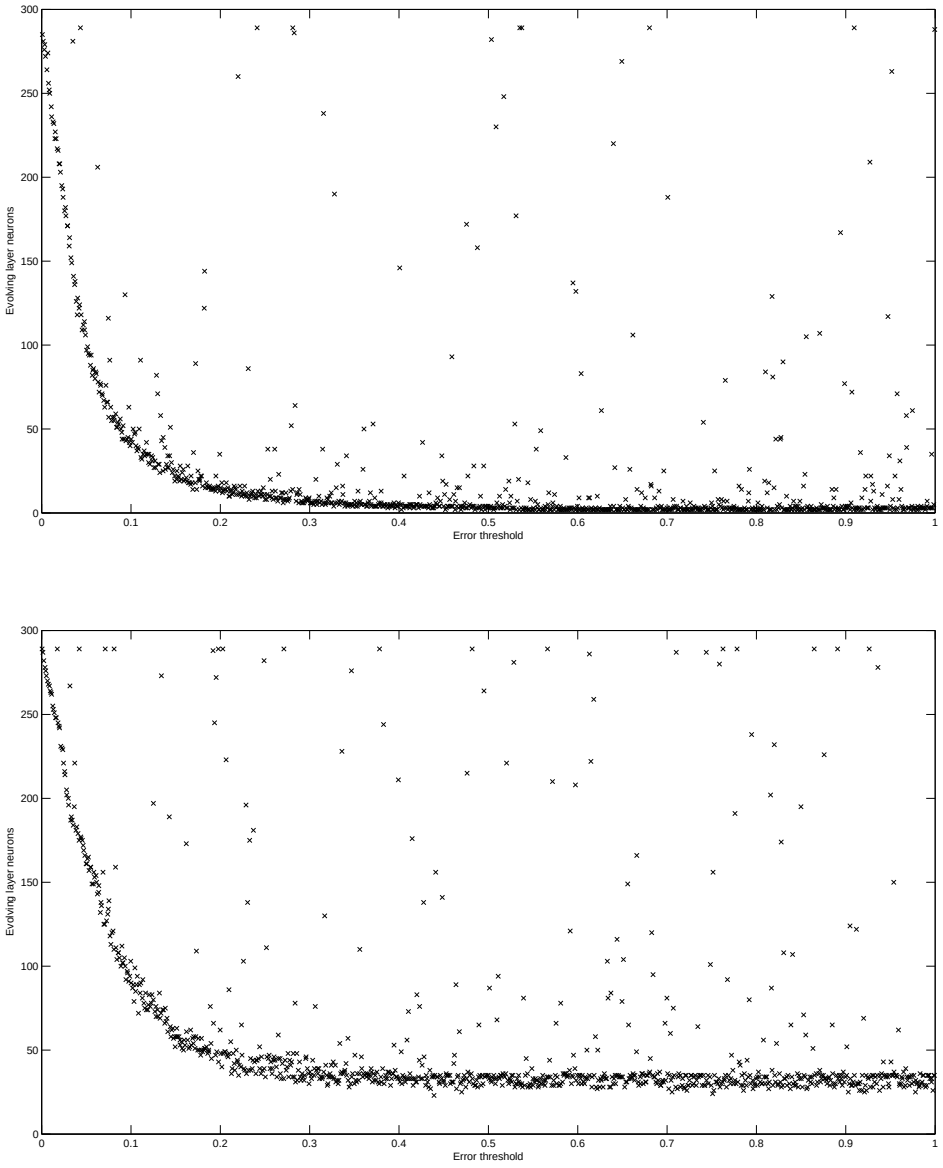

Figure 10: Number of evolving layer neurons versus error threshold parameter for SECoS (top) and EFuNN (bottom) networks trained on the Gas Furnace data set.

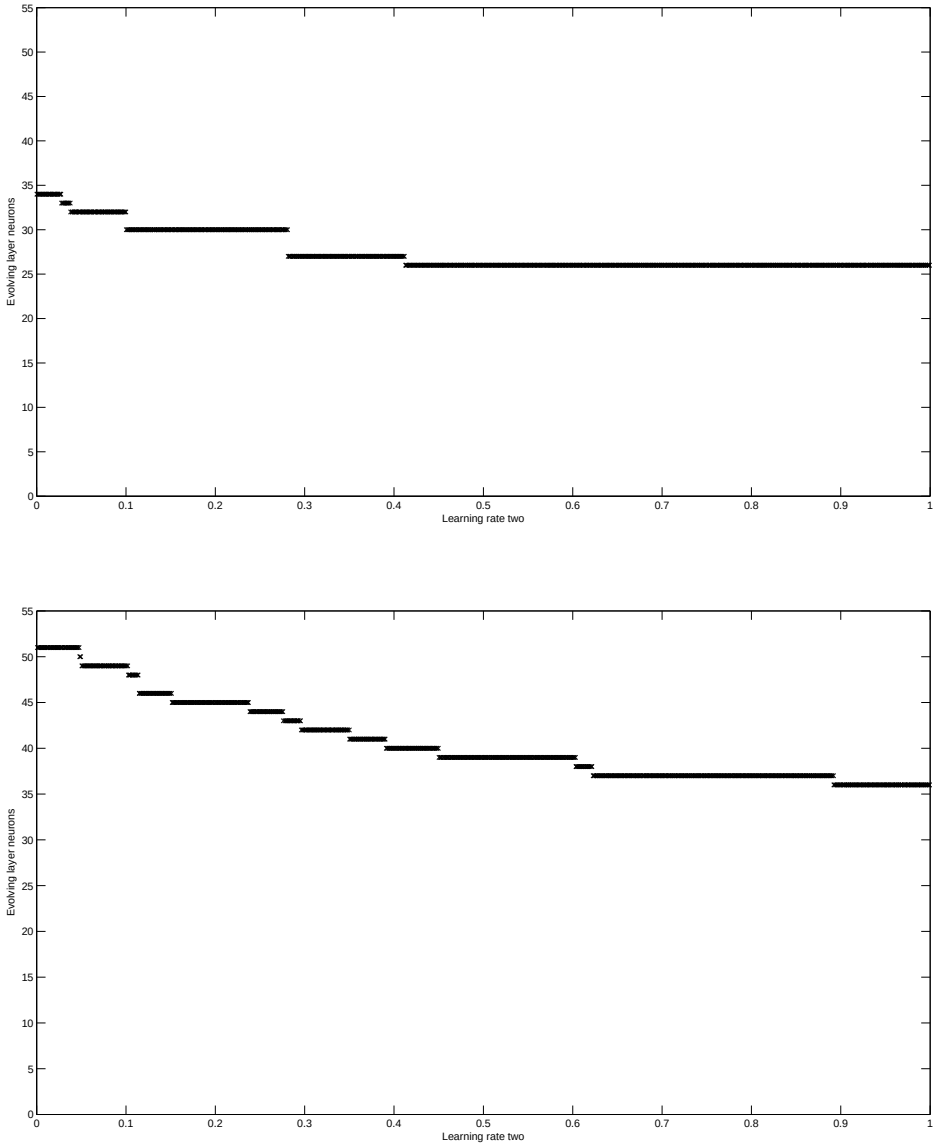

Figure 11: Number of evolving layer neurons versus learning rate two parameter for SECoS (top) and EFuNN (bottom) networks trained on the iris classification data set.

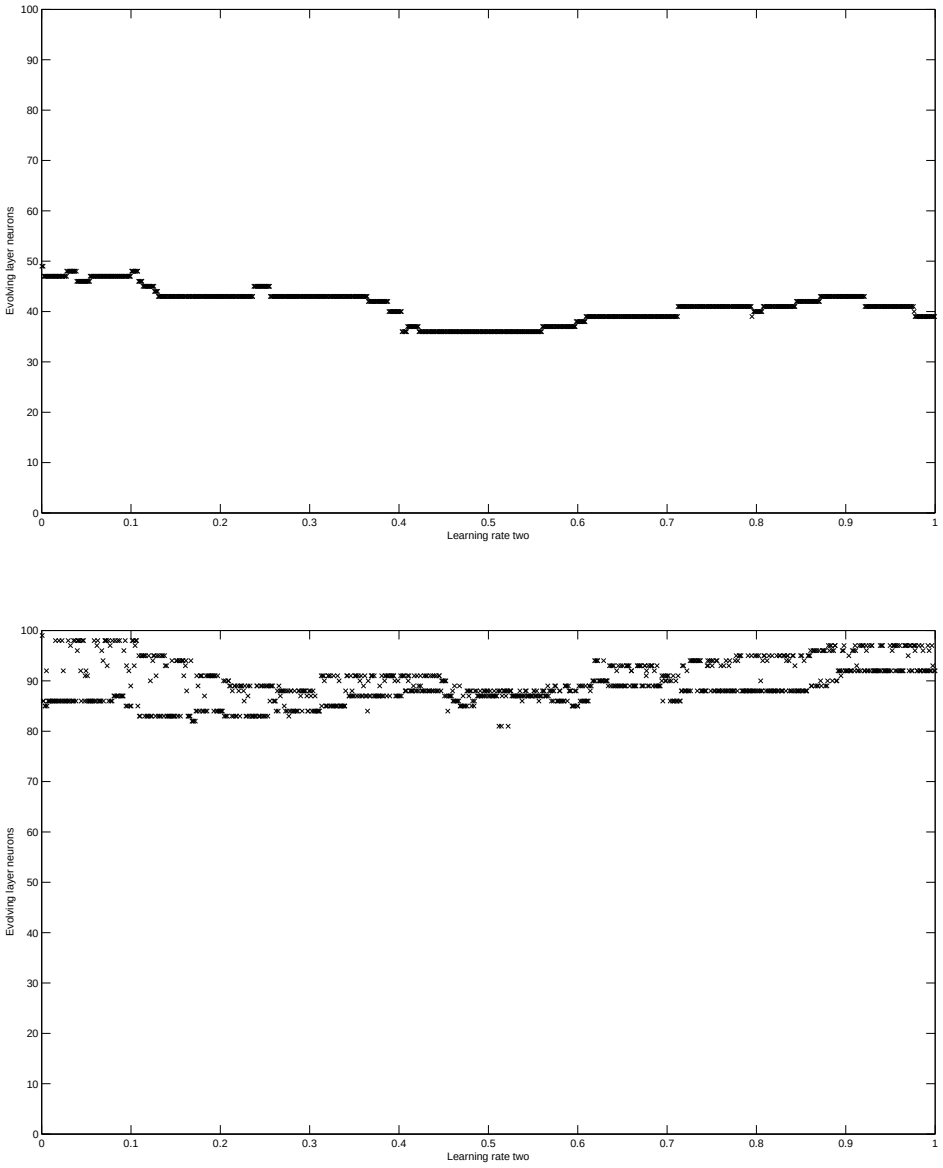

Figure 12: Number of evolving layer neurons versus learning rate two parameter for SECoS (top) and EFuNN (bottom) networks trained on the gas furnace data set.

12 Discussion

The analysis in Section 6 predicts that as the sensitivity threshold approaches unity, so does the probability of adding a neuron to the network. The results in section 11 show that this was the case across both of the benchmark data sets experimented with. The results all have a great deal of similarity about them, that is, the curves are highly similar to one another, despite the great differences in the data sets.

The points at which the curves start to increase were quite similar for both benchmark data sets. This indicates that the point at which the sensitivity threshold dominates (that is, the effect of the sensitivity threshold overcomes the effect of the other parameters) is in the region of $0.85 - 0.95$. Below that value, the curves were quite flat, which implies that the other parameters such as error threshold are responsible for adding most of the neurons.

The difference in size between SECoS and EFuNN was apparent throughout. EFuNN, as predicted, required more neurons than a SECoS to model the same data set.

The predictions made in Section 7 have been validated by these results. The prediction made was that as the error threshold approaches unity, the probability of adding a neuron to the network decreases. For both data sets, as the error threshold increased, the number of neurons added decreased. One difference observed between the classification and function approximation problems was that the decrease in network size was much smoother for the gas furnace function approximation data set. This was due to the greater range of error values that are possible during the training of function approximation networks, as discussed in Section 10.

The predictions made in Sections 9 and 10 have been validated by these results. For classification problems, the number of neurons added during training trend downwards as the learning rate two parameter increases. For function approximation problems, the number of neurons trend downwards to a point, then trend upwards again.

13 Conclusion

This chapter has presented a formalisation of the Evolving Connectionist System (ECoS) constructive ANN. The formalisation describes the internal workings of an ECoS network as a system of Voronoi regions in the input space that map to regions in output space. The formalisation also described the effect of each of the training parameters, that is, it predicted how an ECoS network will behave in relation to the training parameter settings. The formalisation has shown that the parameters are related to one another, that is, one parameter will affect another. This has implications for algorithms that attempt to optimise the training of ECoS.

The formalisation was investigated experimentally across two different data sets, and found to be supported.

It is possible that this formalisation could be used to develop methods of automatically optimising ECoS training parameters as training is underway. This will be the focus of future research.

References

[1] E. Alpaydin. GAL: Networks that grow when they learn and shrink when they forget. *International Journal of Pattern Recognition and Artificial Intelligence*, 8(1):391–414, 1994.

[2] G.E.P. Box and G.M. Jenkins. *Time Series Analysis forecasting and control*. Holden-Day, 1970.

[3] G. Cybenko. Approximation by superpositions of sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2:303–314, 1989.

[4] S.E. Fahlman and C. Lebiere. The cascade-correlation learning architecture. In David S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 524–532. Morgan Kaufman Publishers, 1990.

[5] R.A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

[6] B. Fritzke. Kohonen feature maps and growing cell structures - a performance comparison. In C.L. Giles, S.J. Hanson, and J.D. Cowan, editors, *Advances in Neural Information Processing Systems 5*. Morgan Kaufmann, 1993.

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

[8] N.K. Kasabov. Evolving fuzzy neural networks - algorithms, applications and biological motivation. In Takeshi Yamakawa and Gen Matsumoto, editors, *Methodologies for the Conception, Design and Application of Soft Computing*, volume 1, pages 271–274. World Scientific, 1998.

[9] N.K. Kasabov. Evolving connectionist systems: A theory and a case study on adaptive speech recognition. In *International Joint Conference on Neural Networks (IJCNN), July 10-16*, 1999.

[10] N.K. Kasabov. *Evolving Connectionist Systems: Methods and Applications in Bioinformatics, Brain Study and Intelligent Machines*. Springer, 2003.

[11] N.K. Kasabov and B.J. Woodford. Rule insertion and rule extraction from evolving fuzzy neural networks: Algorithms and applications for building adaptive, intelligent expert systems. In *IEEE International Fuzzy Systems Conference*, pages 1406–1411, 1999.

[12] A.N. Kolmogorov. On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. *Dokl. Akad. Nauk. USSR*, 114:953–956, 1957. (in Russian).

[13] B. Kosko. *Neural Networks and Expert Systems: A Dynamical Systems Approach to Machine Intelligence*. Prentice-Hall, Englewood Cliffs, New Jersey, 1992.

[14] K.J. Lang and M.J. Witbrock. Learning to tell two spirals apart. In David Touretzky, Geoffrey Hinton, and Terrence Sejnowski, editors, *Proceedings of the 1988 Connectionist Models Summer School*, pages 52–57, June 17-26 1988.

[15] M.D. McKay, R.J. Beckman, and W.J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, May 1979.

[16] M.L. Minsky and S.A. Papert. *Perceptrons*. MIT Press, 1969.

[17] A. Okabe, B. Boots, and K. Sugihara. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. John Wiley and Sons, Ltd, 1992.

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

[19] M.J. Watts. Evolving connectionist systems for biochemical applications. In N. Kasabov and K. Ko, editors, *Emerging Knowledge Engineering and Connectionist-based Systems (Proceedings of the ICONIP/ANZIIS/ANNES'99 Workshop "Future directions for intelligent systems and information sciences", Dunedin, 22-23 November 1999)*, pages 147–151. University of Otago Press, 1999.

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

[21] M.J. Watts. A decade of Kasabov's evolving connectionist systems: A review. *IEEE Transactions on Systems, Man and Cybernetics C: Applications and Reviews*, 39(3):253–269, 2009.

[22] M.J. Watts and N.K. 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*, pages 232–239. IEEE Press, 2000.

[23] B.J. Woodford and N.K. Kasabov. Ensembles of EFuNNs: An architecture for a multi module classifier. In *The proceedings of FUZZ-IEEE'2001. The 10th IEEE International Conference on Fuzzy Systems, December 2-5 2001, Melbourne, Australia*, pages 441–445, 2001.