THE LOGICAL GATES GROWING NETWORK

H. E. Ayestaran & R. W. Prager

CUED/F-INFENG/TR 137

July 1993

Cambridge University Engineering Department
Trumpington Street
Cambridge CB2 1PZ
England

Email: hea/rwp @eng.cam.ac.uk

THE LOGICAL GATES GROWING NETWORK

H. E. Ayestaran & R. W. Prager

CUED/F-INFENG/TR 137

July 1993

Cambridge University Engineering Department
Trumpington Street, Cambridge CB2 1PZ, England
Email: hea/rwp @eng.cam.ac.uk

Abstract

A modular three layer variable structure feedforward network capable of learning by stages is proposed. It consists of a first layer of threshold units, and two subsequent layers of logical gates. The threshold units have a vectorial threshold (instead of a simple scalar), which gives them a spatial reference within the input space. They are trained using the method of centroids, developed by the authors. The modularity of this arrangement allows progressive learning, and the extra units are added as needed, to match the complexity of the problem. The system was tested both with real data and with artificially generated data, to assess its potential. Finally, ways of expanding on the present model are discussed.

1 Introduction

One of the main shortcomings of conventional Artificial Neural Networks (ANNs) is their inflexibility: apart from a few exceptions, once the nets are trained it is difficult to train them on an extra unlearnt pattern without forgetting, to some extent, what they have previously learnt. This is not just merely related to the capacity of the nets, nor only to the efficiency with which the net stores information, the real problem is the training algorithm itself.

Most common supervised learning algorithms, and in particular the widely used Backpropagation algorithm, perform some sort of gradient descent on a multidimensional error surface. The error needs to be calculated after each training presentation (or cycle of presentations), and should one forget to train on any particular pattern that is substantially different from those in the training set, there is a large probability that the pattern will not be learnt. Thus the error surface is constantly changing, and in order to train on a new pattern without seriously distorting what was previously learnt, one has to train on the previously learnt patterns as well.

In order to cope with problems of ever-increasing complexity (as is the case when new patterns are introduced), one would need variable architecture nets, which could grow as more data is learnt (learning in stages, growing modularly as needed), trying not to interfere with previously learnt patterns. This approach has been taken by a few researchers, though sometimes with different aims.

A brief summary of some of the models regarded as having potential to learn in stages is presented in section 2, together with a discussion of the objectives of the proposed model. Section 3 looks at the Logical Gates Growing Network (LGGN) and the Method of Centroids which is used for the training. Section 4 details some of the experimental results, and the conclusions and areas of future research are presented in section 5.

2 Supervised Variable Structure Networks

The aim is to design a network capable being trained in stages, by modifying its architecture. Ideally the net should be able to stop and resume its training at any stage, so that once a pattern is learnt it does not need to be included in the learning set of subsequent stages. Note that although optimality and ease of implementation would be desirable qualities, more important however are learning speed, and generalisation properties.

A few variable structure networks are currently being researched. Among the most notable models are the Cascade-Correlation model ([FL90], and a variation [DYY93]), the Upstart Algorithm ([MF89]), and the Evolution-Oriented Algorithm for the Optimal Interpolative Net ([SF92]). All three models rely on expanding their hidden layers to cope with new patterns, but they do so in rather different ways. The Cascade-Correlation model minimizes the information loss from input to output layer by means of fully connected feed-forward triangular hierarchy. Nonetheless it is not capable of learning in stages.

The Upstart algorithm has a mathematical proof of convergence ([MF89]), and is guaranteed to solve a problem in a finite number of steps. The system is capable of learning by stages, though it can easily fall into the trap of over-generalization, making its performance poor when dealing with complex problems.

On the other hand, the Evolution-Oriented Algorithm for the OI net meets the objectives and is the most promising, but it could be computationally involved when dealing with very complex problems.

To overcome the limitations of these models, the authors researched another system which meets the objective, and has the added advantage of being modular in nature: the Logical Gates Growing Network.

3 The Logical Gates Growing Net

This net consists of three layers, using a combination of AND and OR gates after an initial layer of threshold units (see Figure 1). Only the first layer needs training, as the AND and OR gates have fixed weights. Furthermore, because the structure is modular, it would be simple to expand it, at a later stage, without modifying what was previously learnt. But before the algorithm can be explained, the method of centroids, which was used to train the units in the first layer, must be described.

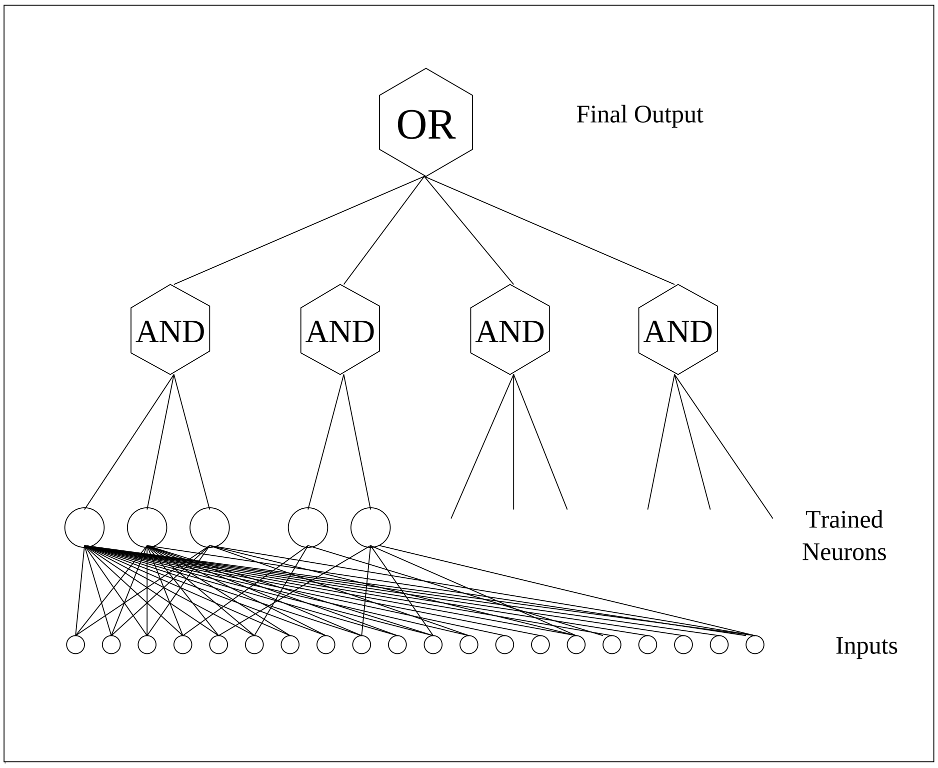


Figure 1: A Possible Arrangement for a Simple Growing Net

3.1 The Method of Centroids

The following method initially tried to place a single hyperplane between two clusters by using their statistical centroids. The centroids could be joined to make up the weight vector, and one would then need to determine a proper threshold (see Figure 2). Let the two centroids be $\vec{a}$ and $\vec{b}$, then if we let:

$$\vec{w} = \vec{a} - \vec{b} \tag{1}$$

and

$$\vec{\mu} = \gamma.\vec{a} + (1 - \gamma).\vec{b} \tag{2}$$

where $\vec{\mu}$ is our threshold vector, we have, for an input pattern $\vec{x}$ lying in the hyperplane perpendicular to $\vec{w}$ and passing through $\vec{\mu}$,

$$(\vec{x} - \vec{\mu}).\vec{w} = 0 \tag{3}$$

This can be rearranged as:

$$\vec{x}.\vec{w} = \vec{\mu}.\vec{w} = \theta \tag{4}$$

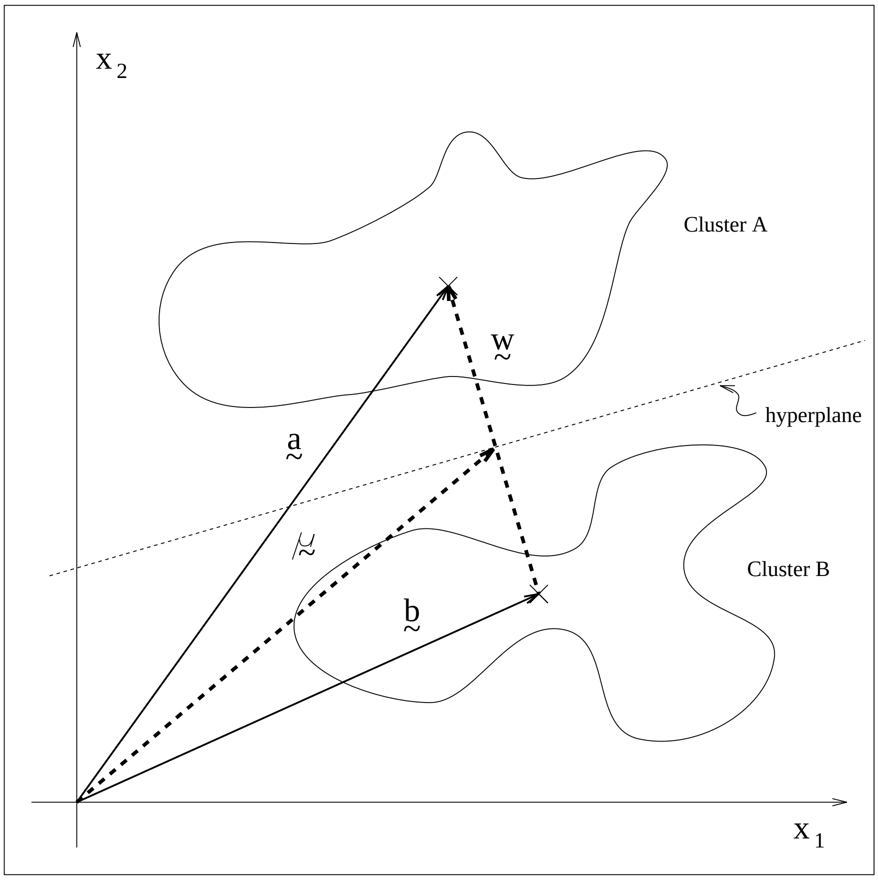
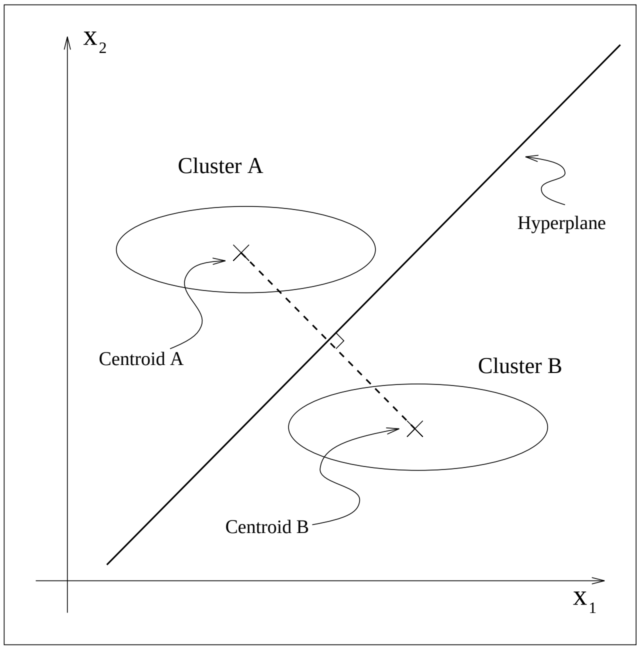


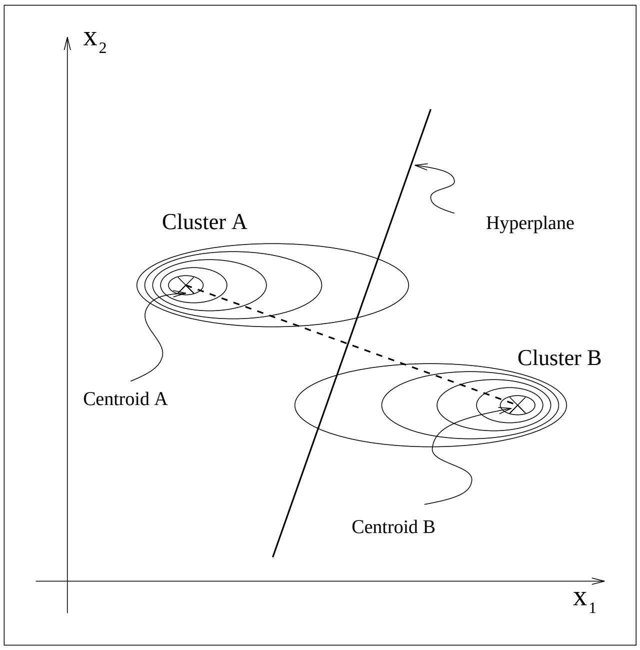
Figure 2: Equivalence of Centroid Method and Conventional Method

Hence note the equivalence of dealing with two vectors (one for each centroid), and dealing with the traditional weight vector and scalar threshold θ. However this method easily encounters simple situations which it cannot solve, as shown in Figure 3.

In order to cope with this, the weights were adapted iteratively, and only when an error occurred. Two more modifications were added: the threshold vector was assigned to be the midpoint between the two centroids (instead of calculating at each step the optimal threshold), and the norm of the weight vector was kept constant. The latter modification was introduced for stability



a) Simple joining of centroids, assuming ellyptical distributions.



b) A more realistic situation, where the method would always fail.

Figure 3: Simple Centroids Joining Algorithm

purposes ([RSS92]). The algorithm used to train a single unit was as follows:

If WRONGLY OFF,

$$a_{t+1} = \eta . a_t + (1 - \eta) . x_t \qquad and \qquad b_{t+1} = b_t \tag{5}$$

Else if WRONGLY ON,

$$b_{t+1} = \eta . b_t + (1 - \eta) . x_t \qquad and \qquad a_{t+1} = a_t \tag{6}$$

Else no change (already correct).

For best results a small initial value was used for η (zero) and this was increased as the algorithm progressed (always less than one).

The value of η that would fully correct the error was also calculated. Consider a 'WRONGLY OFF' error (error on set A), then Equation 5 applies, and from Equations 1 and 2 one gets:

$$\vec{w}_{t+1} = \eta . \vec{a}_t + (1 - \eta) . \vec{x}_t - \vec{b}_t \tag{7}$$

and with $\gamma = \frac{1}{2}$,

$$\vec{\mu}_{t+1} = \frac{\eta . \vec{a}_t + (1 - \eta) . \vec{x}_t + \vec{b}_t}{2} \tag{8}$$

and if after the update, we wish the error to be zero, ie: the point lies in the hyperplane, then from Equation 3:

$$(\vec{x}_t - \vec{\mu}_{t+1}) . \vec{w}_{t+1} = 0 \tag{9}$$

which after replacing gives the final formula:

$$\eta = \frac{||\vec{a}_t - \vec{x}_t||}{||\vec{b}_t - \vec{x}_t||} \tag{10}$$

For 'WRONGLY ON' errors, the result is just the inverse of Equation 10. However when applied to real problems this has many drawbacks, as illustrated by Figure 4. If error occurs near the hyperplane (the ridge in the figure), the value of η is very close to 1, so the system remains virtually unchanged, and is thus extremely slow to converge. The solution lies in tolerating small oscillations by making η slightly smaller than needed. This situation is analogous to that of critical damping in control theory.

Also, if one of the centroids were to react too strongly to a point that is relatively distant, then all that was just learnt by that hyperplane could be lost in a single correction. Because of this, the calculated value of η was multiplied by a function of the error made by the hyperplane. The larger the error, the larger the value of η, and the smaller the change. A threshold of correctness was also added, so that the net would not be satisfied with a hyperplane that just solves the problem, and instead would look for a margin of safety. The improved η function was as shown in Figure 5.

Furthermore, the algorithm described above can easily be expanded to train multiple units simultaneously. For example, consider a first layer using 'centroidal' units connected to a single AND unit. The training algorithm could be as follows:

If the output is correct, do not change.
Else if the output is incorrect :

- If target is 1, correct *least effort* unit, centroid a
- If target is 0, correct all wrong units, centroids b

The meaning of *least effort* is explained in the next subsection.

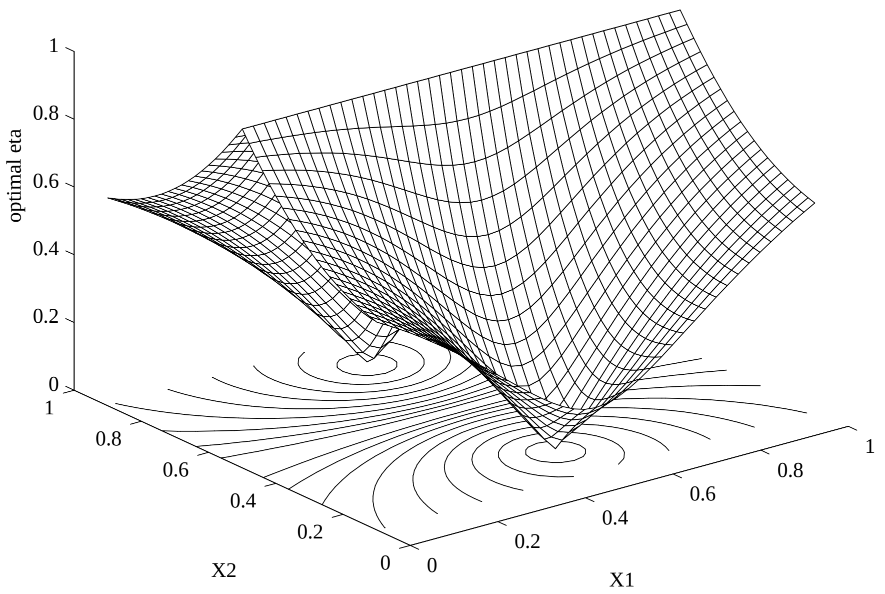


Figure 4: Full Error Correction Value of η

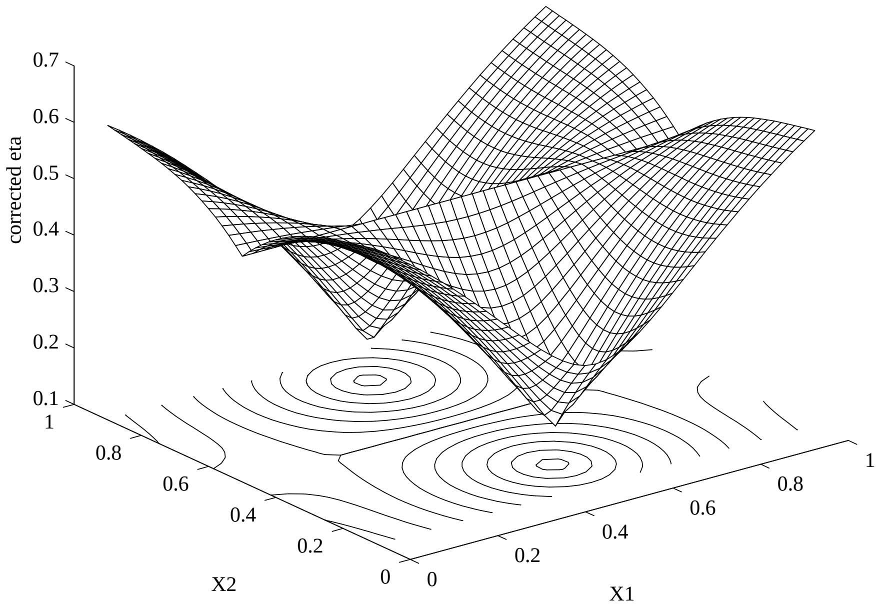


Figure 5: Improved Value of η

3.2 The Growing Model

If a unit is in a situation which it cannot resolve, its weights vector will start to swivel around its threshold vector. By measuring the average distance from this vector to the error, and classifying it according to the error type, one can decide whether an AND or an OR gate is needed (see Figure 6). A simplified algorithm is described below:

LGGN Algorithm

```
Initialize Model
Begin Execution Loop
    Alternate inputs from each set
    Calculate Outputs and Errors
    If Error Occurred:
        If target = 1,
            Find least effort AND unit
            Correct all errors for that AND unit
        If target = 0,
            Find all AND units with errors
            for each AND, correct least effort input
        If unit creation threshold exceeded
            Create appropriate unit and reset counters
Repeat Execution Loop if needed.
```

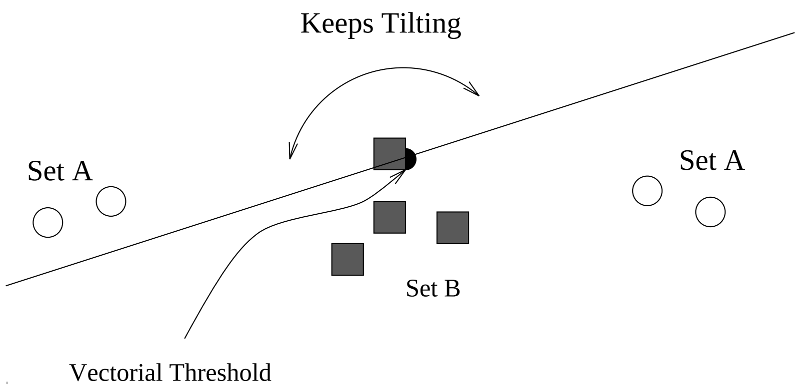


Figure 6: Determining Which Unit to Add (Here an OR unit)

Where the *least effort* unit is intended to be the unit which when modified would introduce the least number of errors into the system. The distance from the threshold (a vector) to the actual 'conflict' point, and the error made by each unit were measured for this purpose. Also of importance was how the choice of the unit to be duplicated was made. For this the modulus of the change at each correction was added, as well as the modulus of the total change since the structure was last modified. Then the ratio of net change to the sum of changes was used to see whether the corrections were oscillatory in nature (in which case the unit would need to be duplicated). A further enhancement being tried at the moment consists of keeping a measure of how crucial a unit is to the system, so as to avoid modifying vital units, and to remove useless ones.

Experimental data showed the model to work remarkably well, considering it is still largely under development.

4 Experimental Results

The algorithm is fast and simple, as very few units are trained at any time. It was tested both with Matlab and with real data: the wheat data ([KD86] and [PDK90]). This last test consisted in being able to distinguish one type of wheat, the mercia type, from the others (more than 22). The problem is outlined below:

The Wheat Problem

- The aim: To distinguish mercia wheat grain from all others. Mercia is good for making bread, others might cause the bakery to collapse in a gluey mess.
- Each grain described by vector of 33 real, normalized elements.
- Two files:

  Large File (L): 23006 input vectors

  Small File (S): 2000 input vectors
- It is a very difficult problem! Back Propagation could only manage 74.9% recognition (training on large file, testing on small one).
- Results for Growing Algorithm:

  Training on Small File : 100%

  Training on Large File : 96.85%

  Train on L, Test on S : 79.5%

  Train on S, Test on L : 75%

The architecture shown in Figure 1 was used to solve this. Individual groups of units, reporting to a single AND gate, were trained in stages to discriminate between the mercia type and one of the other types. And later the output of the AND gates was then fed to a single OR gate. However it was also found that the problem could be solved by using a single AND gate to isolate the mercia type from the rest. This yielded the result shown above for training on the large file (96.85%), after about 850 training cycles (an improvement of about 10% over other models [RWP92]). The drawback however is poor generalization properties, though this is mainly due to the lack of sufficient data. Because hyperplanes are used, sharp corners are introduced at their intersections, and this creates areas of poor generalization. To cope with this one needs a large number of hyperplanes, and representative data in the training set.

Finally, the version of the LGGN algorithm described in the previous section was tested on the concentric rings problem, and was found to work remarkably well (see Figure 7). The model sometimes struggled with complicated XOR problems, although it performed remarkably well for an unpolished and parameter robust algorithm.

5 Conclusions

The area of continuously learning networks seems to be quite fertile for research, and this paper presents various ideas that could be further explored. The Logical Gates Growing Network developed here is fast because of very few units are trained at any time. Furthermore its structure is modular, and new units are introduced by duplicating troublesome ones (hence only where needed). The model is as yet incapable of learning with a single pass through the data, but it can learn in stages.

It manages to solve problems which are complex in nature, obtaining as much as 10% better recognition than other methods when dealing with the Wheat data. Nonetheless it suffers from poor generalization properties, due to the existence of sharp corners at the intersections of the hyperplanes. One solution being considered is the replacement of all hyperplanes connected to the same AND gate, by a smooth interpolative function, such as a spline.

The model as a whole performed very well, though it still needs adjustments. Future research will be concentrating on practical implementation on a large number of inputs (Speech Data), multiple outputs, an adaptation to work with time-varying inputs, and possibly a semi-supervised

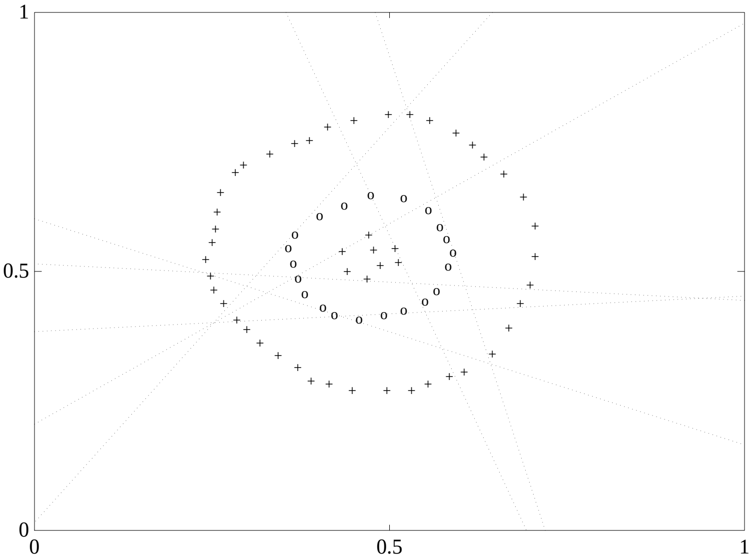


Figure 7: LGGN solving the Concentric Rings Problem

version as well. Also of interest would be the introduction of extra gates, such as NAND and NOR gates, and an increase in the number of layers.

References

[FL90] S.E.Fahlman, C.Lebiere, *The Cascade - Correlation Learning Architecture*, (1990), Technical Report #CMU-CS-90-100, School of Computer Science, Carnegie-Mellon University, Pittsburgh, PA.

[KD86] P.D. Keefe, S.R.Draper, *The Measurement of New Characters for Cultivar Identification in Wheat Using Machine Vision* (1986), Seed Sci. and Tech. Vol.14 pps 715-724.

[MF89] M.Frean, *The Upstart Algorithm: a method for constructing and training feed-forward neural networks*, Edinburgh Physics Department Preprint #89/469.

[PDK90] P.D.Keefe, *Observations Concerning Shape Variation in Wheat Grains* (1990), Seed Sci. and Tech. Vol.18 pps 629-640.

[RWP92] R.W.Prager, *Some Experiments With Fixed Non-Linear Mappings and Single Layer Networks*, Camb. Univ. Eng. Dept., Tech Report CUED/F-INFENG/TR.106, July 1992.

[RSS92] R.S.Shadafan, *A Research on Neural Networks: Architecture, Learning and Generalization*, Cambridge University Engineering Department, First Year Report, August 31, 1992.

[SF92] S-M. Sin, R.J.P. deFigueiredo, *An Evolution-Oriented Algorithm for the Optimal Interpolative Net*, IEEE Trans. Neur.Nets., Vol 3, No.2, March 1992.

[DYY93] D-Y. Yeung, *Constructive Neural Networks as Estimators of Bayesian Discriminant Functions*, Pattern Recognition, Vol 26, No.1, pp 189-204, 1993.