Fractally Configured Neural Networks

by

John W. L. Merrill
and
Robert F. Port

Computer Science Department
Indiana University
Bloomington, Indiana 47405

TECHNICAL REPORT N0. 249

Fractally Configured Neural Networks

By

John W. L. Merrill and Robert F. Port

May, 1988

Research reported herein was supported in part by National Science Foundation grants DCR 85-18725 and DCR 85-05635.

Fractally Configured Neural Networks

John W. L. Merrill and Robert F. Port*
Department of Computer Science
Indiana University
Bloomington, Indiana 47405

Abstract

The effects of network structure on learning are investigated. We argue that there are problems for which specially tailored network structures are essential in order to achieve a desired result. We present a method to derive such network structures, and present the results of applying this algorithm to the problem of generalization in abstract neural networks. In order to derive these networks, it is essential that the system employ a flexible, yet efficient, representation of edge structure. The algorithm discussed here uses deterministic chaos to generate a fractal partition of the edge space, and uses that fractal partition to produce an edge structure. We discuss the results of applying this algorithm to a simple classification problem, and we compare the performance of the resulting network to the performance of standard feed-forward networks. Our results show that the specially constructed networks are better able to generalize than completely connected networks with the same number of nodes.

1 Introduction

All behavior is the result of the combined action of universal laws and specific constraints. The behavior of falling objects is governed by gravity under restrictions due to terrain. Animal life is primarily governed by the action of gravity and the constraints imposed by size and shape. Computers operate under the constraints of their programs through the action of the electromagnetic force.

Neural networks operate under the influence of two classes of laws: those which control the behavior of each node, and those which control the modification of the weights. The elucidation of constraints is far more challenging. These constraints can arise from a number of different sources,

*This research was supported by NSF grants DCR–85–18725 and DCR–85–05635

among them the environment in which the network operates, the nature of the interaction between the network and its environment, or the structure of the network itself. The mass of an animal's body constrains the ecological niche to which it fits. The nature of our ears restricts the sounds we can hear, and hence limits the languages we can learn. The details of the linkage between brain and limb affect the effects of patterns of firing in the motor cortex uponthat limb and hence constrain the motor map that the animal learns. These are examples of constraints embodied in the animal's environment and in the interaction between the animal's brain and it's environment.

We will be concerned with the effects of constraints within the network itself. We shall concern ourselves only with the effects of constraints which are established by the pattern of communication within the network, established by the selection of node inputs. We shall argue that the precise arrangement of the edges in an abstract neural network should constitute a constraint upon the final dependencies that occur among the nodes of a network. We shall employ a two-level search to find an optimal edge structure for an abstract neural network to learn a particular generalization task.

2 Edge structure affects learning

Consider a child learning language. He must realize that formant transitions are an important feature of the acoustic environment, and that the acoustically prominent sections of the signal, which are relatively slow-moving, are informationally impoverished. Then he must distinguish those transitions which are likely to be speech related from those transitions which are less likely to be speech related. How the child recognizes that apparently unimportant regions of the signal *are* important and then that only some of these matter is a major problem that any adequate theory of speech perception must address.

It has sometimes been suggested (eg. Eimas; 1971) that children are born with an innate set of detectors which provide clues to these facts. It is not hard to generate abstract architectures which could constitute a set of generic transition detectors. Smythe (1988) has constructed a hand-wired network using veto inhibition to detect transitions of different lengths and speeds. Anderson, *et al.* (1988) have used back-propagation to perform simple speech recognition; they find that their networks develop nodes which detect different transitions within the speech signal.

One possible neural circuit which could self-organize without external training to detect formant transitions would be a two-level continuous-time network employing some form of competitive learning. If the "lower" of the two levels attends to both the upper level and a single time-window of a Fourier transform of the acoustic signal, and if the time constant of the network is roughly comparable to the lengths of the acoustically relevant features of the input, then such a system will tend to self-organize to detect events of roughly the same length as those transitions. If such

a circuit were to be applied to the detection of the informationally important transitions of the speech signal, however, it might or might not capture them. If all of the edges between the two levels were present, then there would not be any particular reason to recognize transitions of any particular shape or duration. By contrast, if some edges between the levels were deleted, then the system would start out biased towards finding particular patterns within the signal. By simply forcing elements on the lower level to capture particular spectral patterns, and by allowing elements of the upper level to provide expectancy based upon both the actual signal encountered and an innate bias to react in certain ways, an appropriately chosen set of edges could provide enough information to bootstrap a system towards a particular class of perceptual solutions.

Dolan and Dyer (1987) appear to have been the first to observe that modifying the edge structure of a network could alter its performance after training was concluded. Using a competitive learning algorithm (Elman and Zipser (1986)), they trained networks consisting of cliques of elements. Within each clique, elements compete with one another through inhibitory connections. Between cliques, elements cooperate with one another through excitatory connections. Dolan and Dyer observed that by choosing which pairs of cliques communicate with one another, they could build systems which embodied desired semantic relationships in hardware, thus implementing a form of innate knowledge. Similarly, Lehar and Weaver (1987) present an algorithm for searching for a network which optimally learns according to a fixed algorithm. Unlike Dolan and Dyer, Lehar and Weaver allow several different kinds of connections from one node to the nodes in successive cliques.

Another recent paper which examines similar issues is that of Mjolsness, et. al. (1988), in which a method is discussed that allows the construction of small networks to solve problems in a manner that can be directly scaled to larger networks. Using an edge representation system based on recurrence relations, these authors train a small network to perform a coding problem, and then scale it up automatically to perform larger problems without retraining. Their work differs from ours, and from that described above, because they build not only the edge configuration, but also the weight arrangement, into this representation, and thus do not separate two very different kinds of networks change—weight modification and edge modification—from one another. In essence, they collapse the distinction between a predilection to learn a particular concept and true innate knowledge.

3 The generalization problem

The generalization problem is one of the fundamental problems of learning: how can a deterministic system infer a general rule from a limited sample of tokens? All learning systems appear to do this, whether in language acquisition or pattern recognition. One of the compelling features of abstract neural networks is the ease with which they solve this problem. Whether through the continuity

of the response functions, as in feed-forward networks, or by virtue of possessing open response basins, abstract neural networks tend to classify similar stimuli similarly.

One popular method of encouraging generalization in abstract neural networks is the introduction of a waist or constriction through which all information must pass (Cottrell, et. al., 1987; Elman and Zipser, 1988). This idea is most easily discussed in the context of a feed-forward network: If a multi-level feed-forward network contains one level that is much narrower than either its successor or predecessor levels, then that level extracts prominent features of the input signal. In theory, the resulting representation should tend to generalize better than one which employs more, less important, features.

This method suffers from an unfortunate limitation. The nodes in the waist of a completely connected feed-forward network develop a locally optimal representation of the input environment, in the sense that the representation explains as much of the variance of the training signal as can be extracted in a representation of that form. But there are psychologically important tasks, such as speech recognition, in which the cues that transmit information in a generalizable way explain very little of the variance in the signal. When unsupervised networks are trained to capture features in continuous speech, they tend to capturefeatures relevant to the portions of the signal during which change happen slowly or not at all. Even in a supervised condition, waist networks can fail by explaining the data upon which they are trained while not explaining other valid data (Anderson, et al., 1988).

In this paper, we investigate an alternative approach to the problem of encouraging generalization in connectionist networks. The fundamental insight upon which the waist network construction is based is that restricting the flow of information through the network enforces a stiffness constraint upon the response of the network. Waist networks are one method for introducing such a constriction into a network; in this paper, we will investigate the possibility of introducing such a constriction directly, by selecting which edges in the network are present (or absent) at the beginning of training of the weights.

We employ an "evolutionary search" running over the space of all possible edge configurations of connectionist networks. In order to perform this search, we select a novel representation for the edge configuration of the network. We have selected this representation because it combines three properties that are very desirable in configuration search: plasticity, stability, and biological plausibility. This representation, based upon the use of fractal subsets of the plane, is the reason that we refer to the networks in this paper as "fractally configured neural networks."

4 Implementing evolutionary search

The goal of evolutionary search is the selection of the edge configuration which best induces a network to learn some pattern. If evolutionary search is to be efficiently implementable, the outer search procedure, through which the edge configuration is modified, must be selected wisely. Derivatives of gradient descent will fail since the space being searched is not a manifold. Pure random search or simulated annealing could be applied to select the configuration directly, but this procedure will not scale to larger networks. These methods could also be applied to a coarse division of the network into sub-networks (as in Dolan and Dyer, 1988), but such a procedure limits the possible final edge configurations to either "all-present" or "all-absent". In this section, we discuss an alternative representation that is efficient enough to be employed in conjunction with simulated annealing and yet which is flexible enough to yield many final edge configurations.

Much of the inefficiency of a direct search could be avoided if the edge configuration of each network were represented at the node level with some node-by-node code, instead of directly with a simple edge-by-edge code. A node-by-node representation would need to be both combinatorially stable under small changes in the code for each node, so that small enough changes in the code for a node would yield small changes in it's edge structure, and yet would also need to allot a wide variety of different edge configurations for different codes, so that many different network architectures can arise from the search. The representation discussed here is based upon assigning to each node a single real number between -1 and 1, and basing all edge decisions upon that number.

Intuitively, each of the two nodes at the ends of a potential edge compete over whether or not that edge forms. The process by which these decisions are made is based upon the behavior of a certain deterministic dynamical system, which is sufficiently unstable in the interval of interest that small changes in the control parameters of the system can change the outcome of the competition. The result of this competition is a partition of the interval $[-1, 1]^2$ into two basins with a fractal boundary; if the initial conditions are such that the pair of codes corresponding to the two nodes is in one partition, then the edge is created; otherwise, it is not created.

In order to still further reduce the number of parameters in the search, the network was divided up into cliques, and the search was reformulated as a search for the optimal coefficients of the cliques. Elements of the cliques were indexed arbitrarily, since the order in which different nodes appear is unimportant (except for members of the input and output layers). Within each clique, the parameters of the individual nodes were computed by interpolating linearly between the first element of the clique and the last.

5 Methods

5.1 Details of two-level search

Structure of two-level search. Networks were trained by a two-level search. The inner level employed a standard gradient descent error minimization technique called second-order back-propagation. The outer level used a form of simulated annealing to randomly modify the edge structure of the network. An iteration of the outer search consisted of making a small random modification to an already-examined network structure, training the resulting network to perform a simple classification task, and then deciding whether to use the new network structure for the template or to retain the previous template. (See Figure 1 for a schematic of this search.)

Each network consisted entirely of standard semi-linear nodes, as described in, for instance, Rumelhart, et. al. (1987). Each node computes the dot product between its inputs and a set of weights upon them; a bias is then added. The sigmoid function

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

is applied to this sum. The result is broadcast to the network as the output of the node.

Training method. Training tokens were arranged in a circular linked list. The training device circulated through this list, always beginning at a fixed element. When a member of the list was presented, the weights in the network were modified to increase the response of the output node which corresponded to the type of token which had been presented, and to decrease the response of the output node which corresponded to the other type. The networks were trained using the second–order back–propagation algorithm (Parker, 1987) with learning rate $\alpha = 0.25$, inverse time coefficient $\mu = 0.25$, and time step $\Delta t = 0.10$. The trapezoidal rule was used to solve the iterative differential equation that arose from applying this algorithm.

Fractal edge decision procedure. Each node was assigned three real parameters which would control its behavior in a fractal edge decision procedure. These three parameters were an edge code, an input coefficient, and an output coefficient. Edge codes were selected from the interval $[-1, 1]$; input and output coefficients came from the interval $[2, \infty)$. If nodes n_1 and n_2 were in successive levels of the network, so that there was a potential edge between them, then the decision procedure was applied.

Let ec_1 and ec_2 denote the input and output codes for n_1 and n_2, respectively. Two temporary parameters were computed: $x_1 = (ec_1 + ec_2)/2$ and $x_2 = (ec_1 - ec_2)/2$. Letting c_1 and c_2 represent the input coefficients for n_1 and n_2, respectively, the dynamical system

$$x_i^{(t+1)} \leftarrow (c_i(1 - |x_i^{(t)}|)) - 1$$

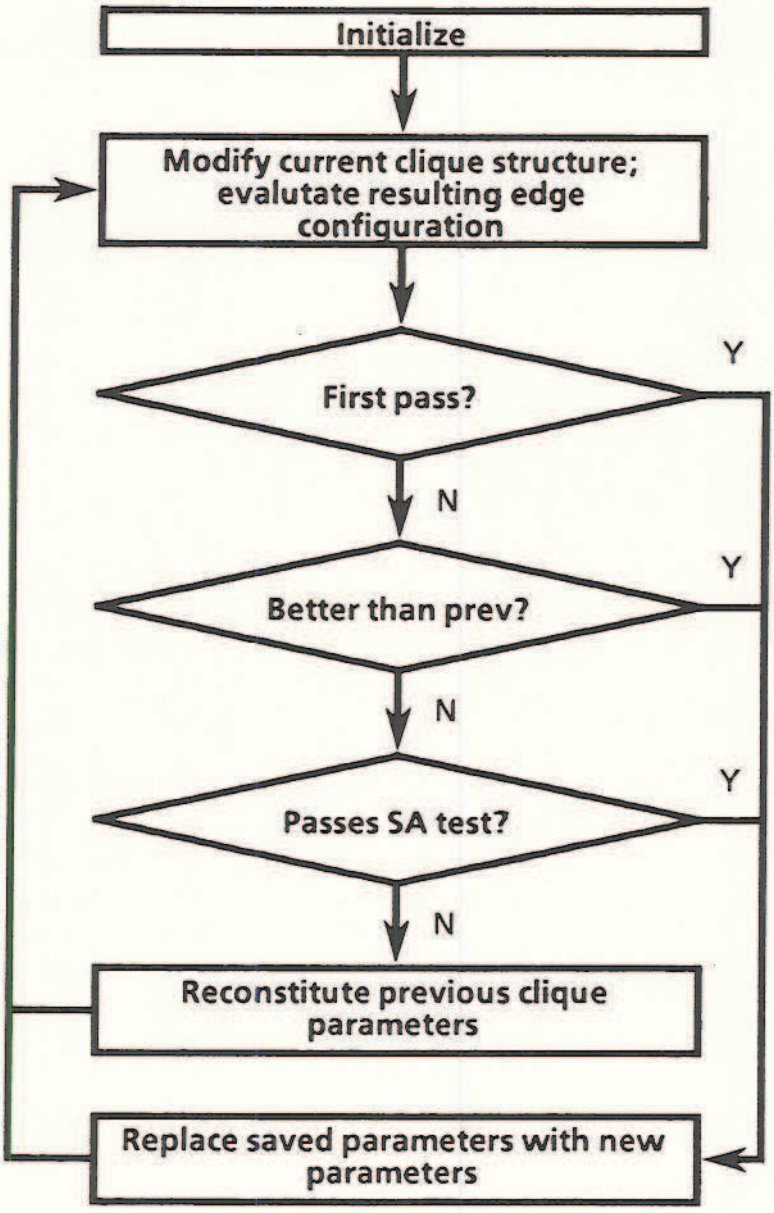


Figure 1 A schematic of the evolutionary search paradigm. The system gradually modifies the structure of a network to improve its performance.

was run with x_1 and x_2 as initial values. For most values of x_1 and x_2, there is some n for which the n-th iterate of at least one of x_1^n or x_2^n was not in the interval $[-1, 1]$. If x_1 left the interval before x_2 did, or if they both left at step n and $x_1^n > x_2^n$, then an edge was placed between the two nodes; otherwise, no edge was placed between them.

Clique structure. The network was partitioned into cliques and edge coefficients were distributed linearly through the clique. Elements of each clique were indexed arbitrarily, and edge coefficients were assigned to each element, say x_i, by

$$ec_{x_i} = ec_{x_0} + (i-1)\Delta_{ec}.$$

The input coefficients and output coefficients of elements within the same clique were constant across the clique.

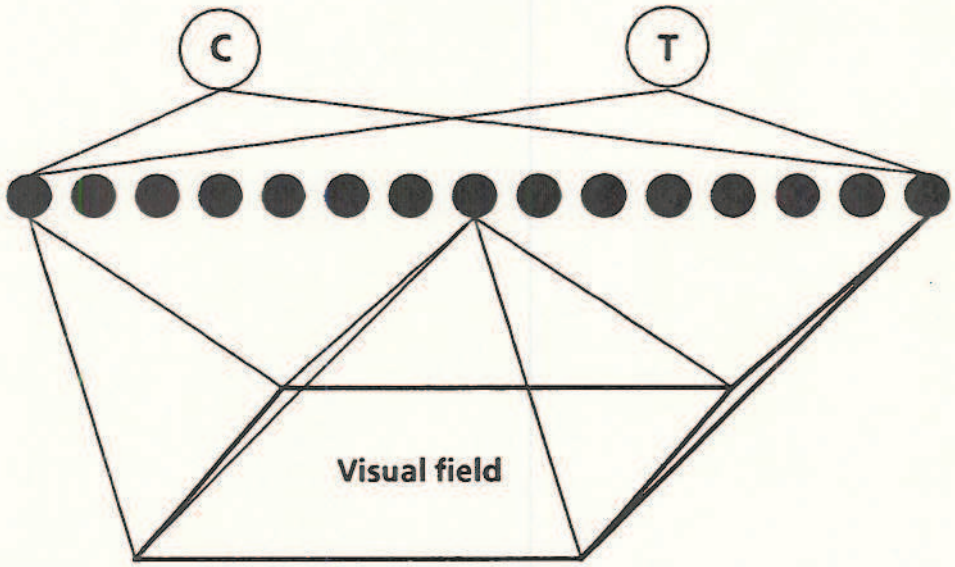
Network structure search. After each network was trained, it was evaluated by computing the number of tokens which it correctly classified, and this performance was optimized over time by applying fast simulated annealing (Szu; 1987) to modify the minimal and maximal values assigned to each clique. At stage k of the search, a parent structure, P is maintained; let p_P denote its performance on the desired task. A child structure, C, is created by modifying the parent structure. C is then evaluated, resulting in a score p_C. If $p_C \geq p_P$ or if $p_C < p_P$ and a certain random test is passed, then C becomes the parent for the next stage of the search; otherwise, P is retained.

As an efficiency device, whenever the edge configuration resulting from C was identical to that from P, C was immediately accepted and accorded the same performance score as P. In the example presented here, this yields a dramatic improvement in efficiency, since modifying the parameters of the top level in this search will almost never result in the addition or deletion of an edge between the hidden layer and the output layer.

Clique modification procedure. After each evaluation step, one of the cliques in the network was selected at random. The parameters corresponding to both the initial value of the control parameter of the clique and the increment of the clique were randomly modified, with the expected amount of the modification decreasing over time. Initially, the amount of the modification of each of these two parameters was taken to be on the order of 10^{-15}; this amount was decreased as the inverse of the number of annealing steps.

5.2 Simulations and results

Network task. Each network was presented with a 20×10 element raster field, upon which lay a 3-by-3 block C or T in one of the four possible orientations. (See figure 2 for some examples.)



(a)

(b)

Figure 2 A schematic of the *C-T* discrimination task which the networks in the first experimental condition were trained to perform. (a) A schematic of the network structure: a feed-forward network with three levels that "looks down" onto a 20×10 "visual field". (b) The two archetypical patterns, a block *C* and a block *T*. (c) A sample visual field at which a network might "look".