Self Organizing Modular Neural Networks

Ira G. Smotroff, David H. Friedman, and Dennis Connolly
The MITRE Corporation
MS A-047
Bedford, Massachusetts 01730
email: ira@mitre.org

The Cascade Correlation algorithm introduced a sound basis for self-scaling learning of monolithic neural networks, but it suffers from a number of drawbacks. These include degradation of learning speed and quality with the size of the network and the development of deep networks with high fan-in rates to hidden units. The Iterative Atrophy algorithm preserves the good features of Cascade Correlation and eliminates its worst characteristics. In addition, we show that Iterative Atrophy extends naturally to the development of self-scaling modular networks which readily perform transfer of training among modules.

FIXED VS. SELF-SCALING LEARNING ALGORITHMS

Learning algorithms such as backpropagation (Rumelhart and McClelland 86), which begin with a predetermined network topology and attempt to fit it to the problem at hand, often exhibit slow learning performance. The major reason for this is that credit assignment for the observed error at the outputs must be assigned to all weights in the network at each epoch. The next reason is that all neuron unit weights in the network are initially randomly assigned and it takes a long time for the first layer(s) of hidden units to form reasonable values based on credit assignments derived from output layer errors that have been divided up among many neurons in middle layers. Yet another reason is that poor fits between the topology estimate of the researcher and an optimal topology causes either a scarcity of resources which leads to overgeneralization (and poor performance) or a surplus of resources which leads to an underconstrained situation with redundant neuron functionality, a resultant waste of computation effort and the potential to overfit the data.

An alternative to learning with a fixed topology has been observed in developmental neurobiology. In the development of binocularity in a kitten's optical cortex, the gross structure of connectivity is apparently genetically programmed, but the specific connectivity is determined by a competition for resources leading to specialization. For example, in a normal kitten's optical cortex, the distribution of cells is initially mixed between specific (e.g., dedicated to one eye), non-specific, and unresponsive cells (Fregnac and Imbert 78). As the normal kitten develops, competition for resources leads to the specialization of these cells for either eye. In contrast, a kitten reared in the dark develops an optical cortex in which the cells are all non-specific. A part of what is going on in developmental brains appears to be a process of search. The search is implemented by rich initial connectivity followed by a process of specialization through interaction with data implemented by competition and atrophy of less useful connections. Through this combination of operations, an efficient network topology can be learned. In the following, we describe a number of algorithms which mimic this process to train large neural networks with topologies specialized for the task. We also describe a new algorithm for the efficient learning of modularized neural networks whose expert subnetworks cooperate to form complex functions which are usable for real-world data sets. Finally, we demonstrate modular network learning on a complex task.

SELF SCALING NETWORK LEARNING

Cascade Correlation

Fahlman's Cascade Correlation algorithm ("Cascade") (Fahlman and Lebie're 90) avoids the problems of fixed topology learning algorithms in a number of ways. First, it is a bottom up algorithm which determines the number of hidden neuron units based on the requirements of the problem. At each epoch, it trains one hidden unit to maximally correlate its output with the residual error seen at the network's outputs. The trained hidden unit is then "frozen" and the output units are retrained to incorporate the new error predictor. This has a number of implications. Because the network structure being trained is never larger than one layer of

0-7803-0164-1/91/0000-0187$01.00©1991 IEEE

weights, training one neuron at a time avoids the computationally intensive credit assignment problem that backpropagation exhibits. As a direct result, both learning accuracy and speed are improved. A further advantage of training hidden units one at a time concerns learning computation efficiency. Once a hidden unit is trained, it is considered a static component of the network and its response to all training patterns may be recorded and placed in a cache. Thus, the complexity of Cascade is $O(n \log n)$, where n is the number of hidden units required to represent the function being trained, and it does not exhibit combinatoric explosion for networks of large size.

The second way in which Cascade avoids problems of fixed topology learning algorithms is in the determination of topology by a process which is analogous to blind search. As each new hidden unit is added to the network, its inputs are fully connected to the network inputs and to all the outputs of preceding hidden units. Thus, any extant information in the network that is useful to the current hidden unit being trained is available and utilized. This avoids redundant functionality of hidden units and thus speeds learning. In addition, there is never a case where the system is overconstrained, so overgeneralization is avoided. And, when a proper termination criterion is supplied, the algorithm does not overfit the data.

However, there are problems with Cascade, as Fahlman acknowledges (Fahlman and Lebie're 90). The technique of full interconnection for each hidden unit leads to deep networks which have many layers. In addition, the fan-in of hidden units rises linearly with the size of the network. Hence, learning speed and efficiency degrade as a network grows.

Our thesis is that what is missing from Cascade is the process of specialization implemented by competition and atrophy of less useful connections as observed in developmental brains. In the following section, we describe an algorithm which eliminates the problems via the addition of a specialization process.

Iterative Atrophy

The Iterative Atrophy algorithm adds specialization and focused search to Cascade for the training of each hidden unit. This results in a learning complexity of $O(n \log k)$, where k is a problem-specific constant which is smaller than n. If we consider a network layer to be a set of hidden units that share a set of inputs, Cascade produces deep networks in which every hidden unit is another layer. Iterative Atrophy avoids these deep networks, but the commonality of inputs that characterize the layers that humans construct in fixed topology schemes is not necessarily observed. Rather, inputs are limited for each hidden unit and connections which skip intermediate hidden units are typical.

Specialization. Specialization is added by introducing competition among connections followed by atrophy. As each hidden unit is trained, the saliency of its weights are analyzed relative to some criterion and the weights which are determined to be weak are eliminated. Le Cun, et al (Le Cun 90) have shown, in the context of backpropagation, that there are a number of benefits to approaches which minimize the connectivity of a network. If a connection is unrelated to the function that a hidden node is computing, the presence of unrelated information from the connection can reduce the node's performance. Thus, weight elimination can improve the performance of a hidden node. Another benefit is an improvement in the network's ultimate execution speed due to reduced computational demands. There are two equivalent methods for achieving connection minimization: weight pruning and weight decay.

In weight pruning methods, a hidden unit is fully trained, the weight saliencies are assessed, the losing weights are pruned from the network, and the hidden unit is then retrained to best utilize the remaining connections. To assess the saliency of a connection, we attempt to estimate the resultant increase in error when the weight is set to zero. One method of assessing connection saliency is a Brute Force method, which successively sets each weight to zero and measures the resulting increase in network error over the entire epoch. It is a good local estimate which ignores interactions among the weights, but it is computationally infeasible for runtime use. Le Cun, et al (Le Cun 90) have described computationally feasible methods for assessing weight saliency. Magnitude thresholding, in which small magnitude weights are considered less salient, has been used by many researchers.

We implemented and assessed a number of these methods for Iterative Atrophy. The results obtained with the Brute Force method were used as a control and compared to those obtained using Magnitude Thresholding. We found that with appropriate choices of thresholds, Magnitude Thresholding correlated well with the Brute Force analysis.

To include the interactions among the weights in our Brute Force method, the worst weight is determined using the normal Brute Force technique. We then *repeat* the process to find the next worst weight, measuring error performance resulting from a hidden unit with both the worst weight and the weight being tested removed. The process iterates until all weights are ranked. Learning curves for Iterative Atrophy on the spiral problem using the improved Brute Force method and Magnitude Thresholding are shown in figures 2-4. We see that Magnitude Thresholding does a respectable job of assessing weight saliency, especially if the weight thresholds are conservative. The spiral problem biases these results because the training of the last hidden units implements very specific patterns akin to a lookup table. Thus the spiral belongs to a class of problems which is more sensitive to aggressive weight elimination techniques.

In weight decay methods, the learning algorithm is modified to drive less salient weights towards zero (Weigand, et al 90). The zero valued connections can then be pruned, and no retraining is required. Because of the computational efficiencies obtained, we are implementing weight decay as the standard connection minimization approach for Iterative Atrophy. Lebie're (Lebie're 90) has considered the use of weight decay in the Cascade environment.

Focused Search. In Cascade, each new hidden unit is connected to all extant hidden units and input units. This is analogous to blind search, where all possibilities in a search space may eventually be investigated. Blind search is computationally prohibitive for all but the simplest of problems and should be avoided. In Cascade, it is the only aspect of the training which prohibits learning very large networks. We can make a loose analogy between network search and search in games, where searching forward in time is measured in plies. Focusing the search backward over the extant network structure is thus analogous to limiting the number of plies to be searched forward in a game program.

For Iterative Atrophy, we considered three methods of making the determination of connectivity a more focused process. The first method sets up a fixed-size moving window of connectivity. This method assumes that connections from inputs and early hiddens lose saliency as the learning progresses. The problem with this approach is that the estimation of window size is performed by humans and may be viewed as a fixed topology constraint. Like other fixed topology constraints, it introduces artificial limits on the learning that are unlikely to match the problem characteristics. For example, it is easy to imagine a set of inputs which may still hold valuable information far into the training, yet will be ignored by the fixed window scheme after the window moves from the vicinity of the inputs.

The second method is a modification of the first. In this method, a number of random interconnections are made for each hidden unit. While this method limits the potential fan-in for a hidden unit and avoids artificial limits on connectivity that might be introduced by the moving window technique, it may potentially reduce learning efficiency by causing less powerful hidden units to be developed if less salient connections are included that exclude more salient ones.

The *salient window* method incorporated into the algorithm may be seen as a synthesis of the first two which avoids the pitfalls of each. Salient window utilizes a problem-dependent window statistically determined from the results of previous weight saliency analyses. The network initially uses a full interconnection policy for Iterative Atrophy. As training proceeds, statistics are kept on the connections being eliminated. When a connection has been eliminated for r percent of the past l hidden units, the connection is removed from the set of connections considered for future hidden units. In this way, the search is directed away from connections that have been eliminated in the past.

Of course, there remains the possibility that connections determined to be less salient at a particular stage of training may be more salient later on in the training. To compensate for this possibility, we borrow from Turing's (Turing et al. 1953) game playing strategy of *fixed depth with extensions for quiescence.* In that scheme, the game playing program would normally search a fixed number of plies unless the limit left it in an unresolved stage of play. In that case, it would search a bit farther to see how the game state resolved itself. In the case of salient window search, if a hidden node's training performance is poor, the window of connectivity can be expanded and training repeated in an attempt to improve the performance. In this manner, connections found to be salient again are reinserted into the set of connections considered for future hiddens.

We have successfully used Iterative Atrophy on a range of problems including the classification of meteorological phenomena (Smotroff et al 1990).

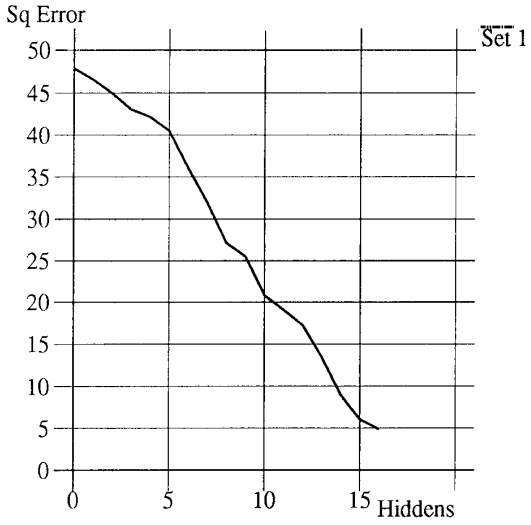

Figure 2.
Iterative Atrophy Learning using
the Brute Force saliency measure.

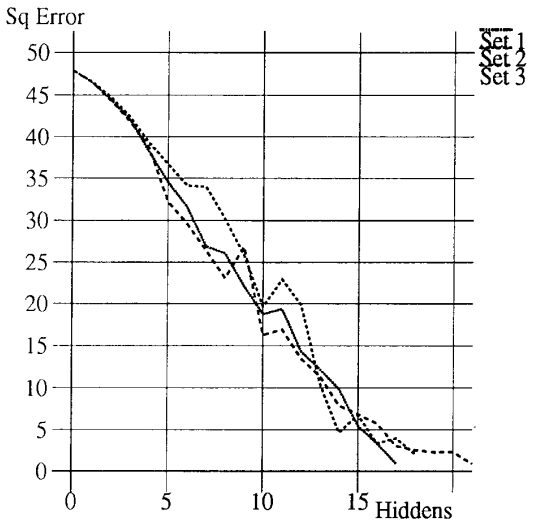

Figure 3.
Iterative Atrophy Learning using
Magnitude Pruning with
threshold = 0.8

MODULAR ITERATIVE ATROPHY

A recurring theme in the neural network literature is the notion that scaling up hill-climbing based learning algorithms requires the division of the problem into a set of simpler problems that can be solved by simple networks that cooperate (Carpenter and Grossberg 87; Hampshire and Waibel 90; Jacobs 90; Minsky 86). Although the size of Iterative Atrophy networks can be quite large, Jacobs (Jacobs 90) identifies two problems of monolithic networks which lead us to believe that modular systems are still necessary for training networks on complex problems. *Spatial Crosstalk* occurs when different groups of a network's output units serve different tasks and are supported by common hidden units. In this case, hidden units being trained to resolve residual error will receive conflicting information, leading to slower learning rates and larger networks. *Temporal Crosstalk* occurs when different portions of the training set contain data for separate and conflicting functions. In this case, the hidden unit being trained will suffer from overgeneralization, resultant slower learning rates, and potentially degraded network performance.

Separating complex tasks into subtasks which are handled by a group of cooperating expert subnetworks allows us to avoid Spatial and Temporal Crosstalk. In addition, modular networks enable positive *transfer of training* within modules by grouping similar functions on each module.

Modular Iterative Atrophy trains a number of expert subnetworks using the Iterative Atrophy algorithm. Modular Iterative Atrophy enables positive transfer of training among subnetworks as well as within subnetworks.. Expert subnetworks that compute a function useful to another subnetwork which is learning a related function can provide useful information which speeds learning. Transfer of training is easily implemented by allowing new hidden units to connect to extant hidden units *in other subnetworks*. Because Iterative Atrophy prunes away the less salient connections, the modules remain distinct entities. Spatial and Temporal crosstalk are avoided because each subnetwork is trained only on training data for its particular expert task. Modular Iterative Atrophy trains expert subnetworks in sequence, one at a time. Each subnetwork can transfer training from any of the extant subnetworks by connecting to their hidden units. When a subtask is very similar to one that is already training, the new subnetwork can consist of only a few new hidden units. Thus, training time and node redundancy are minimized.

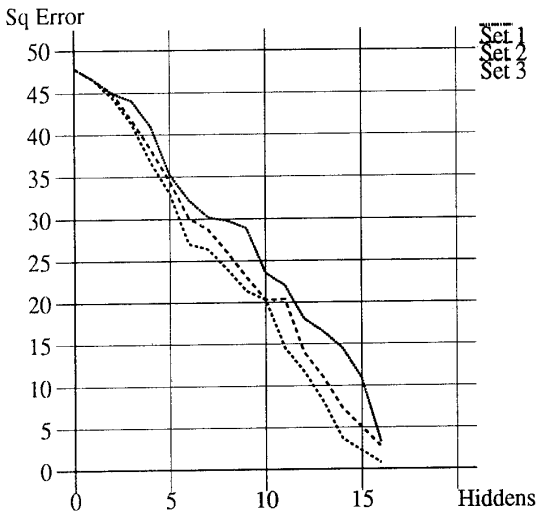

Figure 4.
Iterative Atrophy Learning using
Magnitude Pruning with
threshold = 0.4

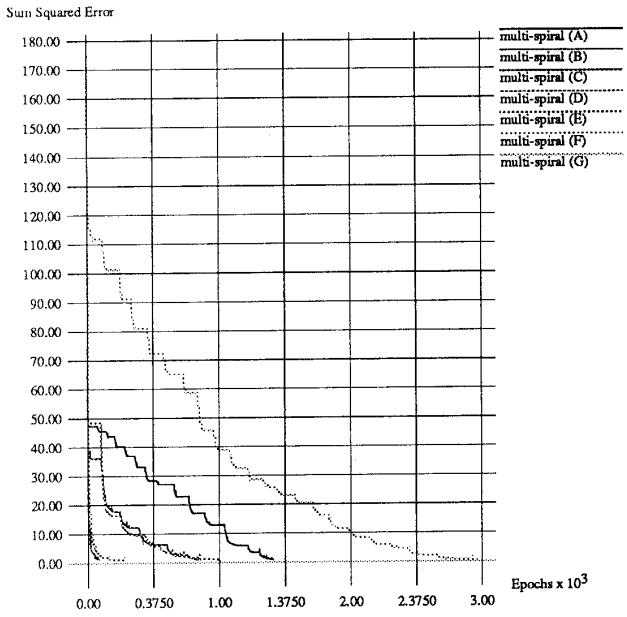

Figure 5.
Multi-Spiral Training Curves

An example of a complex training task which can benefit from modular training is the multi-spiral task of figure 6. In the simple spiral task, intertwined spirals are recognized based only on their X and Y coordinates. In the multi-spiral task, seven variations of the simple spiral task are presented to the Modular Iterative Atrophy algorithm which develops seven expert subnetworks. Spiral 6b is a 90-degree rotation of spiral 6a. Spiral 6c is the inverse of spiral 6a. Spiral 6d is a 90-degree rotation of spiral 6c. Spiral 6f is a 90 degree rotation of spiral 6e. Square spiral 6g is distinct. The serial mode learning curves for this task are illustrated in figure 5. It is clear that significant transfer of training is taking place. Spiral 6a takes 1400 epochs to learn. Spiral 6b takes 900 epochs to learn and is clearly benefitting from some of the abstract hidden units generated for spiral 6a. Inverse spirals 6c and 6d take only 100 epochs to train because they must only learn to invert the representations learned for spirals 6a and 6b. Spiral 6e takes 900 epochs and benefits from the same abstract hidden units that spiral 6b benefitted from. Spiral 6f, the inverse of 6e, learns in 260 epochs. Spiral 6g, the square spiral, learns in 2800 epochs because it cannot benefit from the representations developed by the other expert subnetworks.

SUMMARY

An argument has been made for the advantages of self-scaling neural network learning algorithms as opposed to fixed-topology algorithms such as backpropagation. Cascade Correlation has been shown to be a self-scaling learning algorithm of great promise that suffered from some bad characteristics. The Iterative Atrophy algorithm has been introduced as an enhancement of Cascade Correlation. It preserves the good features of Cascade Correlation while eliminating the worst characteristics. It has also been shown that the new algorithm extends nicely into the realm of modular learning algorithms, where it is named Modular Iterative Atrophy. We are currently extending these techniques to operate on monolithic data sets.

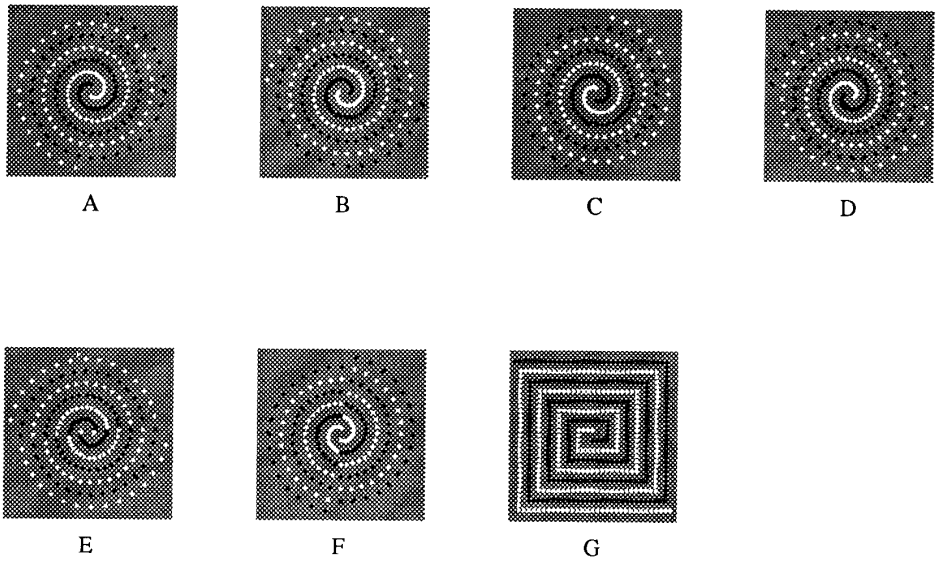

Figure 6. Multi-Spiral Task

REFERENCES

Carpenter, G.A. and Grossberg, S. (1987). ART2: self-organization of stable category recognition codes for analog input patterns. *Applied Optics* 26, no. 23: 4919-4930.

Fahlman, S. and Lebie're, C. (1990). The Cascade-Correlation Learning Architecture. *Neural Information Processing Systems 2*. 524-532. San Mateo, CA: Morgan Kaufman.

Fregnac and Imbert (1978). Early Development of Visual Cortical Cells in Normal and Dark-reared Kitten. *Journal of Physiology* 278: 27-44.

Hampshire, J. B. and Waibel., A. (1990). Connectionist Architectures for Multi-Speaker Phoneme Recognition. *Neural Information Processing Systems 2*. 203-210. San Mateo, CA: Morgan Kaufman.

Jacobs, R. A. (1990). Task Decomposition Through Competition in a Modular Connectionist Architecture. COINS Technical Report 90-44, University of Massachusetts at Amherst.

Lebie're, C. (1990). Personal Communication.

Le Cun, Y.; Denker, J.; Solla, S. (1990). Optimal Brain Damage. *Neural Information Processing Systems 2*: 598-605. San Mateo, CA: Morgan Kaufman.

Minsky, M. (1986). *The Society of Mind.* New York, NY: Simon & Schuster.

Rumelhart, D.; McClelland, J. (1986). *Parallel Distributed Processing*, 318-362 Cambridge, MA: MIT Press.

Smotroff, I. G.; Howells, T. P.; Lehar, S. (1990). Meteorological Classification Using Neural Network Data Fusion. In *Proceedings of the International Joint Conference on Neural Networks* (San Diego, CA, June 1990). IEEE, Piscataway, NJ, Vol. II, 23-29.

Turing, A. M. et al. (1953). Digital Computers Applied to Games. In *Faster Than Thought*, 286-310. London, UK: B.V. Bowden.

Weigand, A. S.; Rumelhart, D. E.; Huberman, B. A. (1990). Backpropagation, Weight Elimination and Time Series. *Connectionist Models. Proceedings of the 1990 Summer School.* 105-116. San Mateo, CA: Morgan Kaufman.