Minimal and High Order Neural Network Topologies[1]

E. Fiesler

IDIAP
C. P. 609, CH-1920 Martigny, Switzerland

Abstract

One of the main goals of current neural network research consists in finding the optimal network topology for a given problem. Several methods have been proposed, but a mutual comparison of the resulting topologies remains a problem until proven minimal topologies are found. This paper describes a straightforward method for finding the minimal feed-forward topology for completely defined problems, based on an extensive search followed by mutual substitution. This method has been applied to finding the minimal topologies of the 'exclusive OR' problem for the most important interconnection schemes. Since the number of possible topologies grows very fast with the size of the network, it is only feasible to analytically determine the minimal topology for small networks.

In general, the smallest topologies can be obtained using high order neural networks. Small solutions to well known benchmark problems using high order neural networks are presented. It is shown that high order networks without a hidden layer can theoretically compute any Boolean or other bivalued function. More concrete, constructive algorithms are given for creating both fully connected and sparsely connected high order neural networks for the computation of an arbitrary Boolean function.

Keywords: (artificial) neural network, neural computing, neurocomputing, connectionism, neural network topology, minimal topology, high order neural network, sigma-pi neural network, partly connected neural network, partially connected neural network, neural network architecture, sparse neural network, feed-forward neural network

[1]This article will be published in: *Proceedings of the 1993 International Simulation Technology Conference (SimTec '93)*, Society for Computer Simulation (SCS), San Diego, 1993

1 Introduction

A key problem in current neural computing is the choice of the network topology. The optimum topology is almost always the smallest topology that allows the problem to be solved adequately (Occam's razor). Especially for ontogenic neural networks and other network construction schemes [Fiesler-93.2], like the ones based on genetic algorithms, it is important to know the minimal topology for a given problem. Although several approaches have been suggested to this neural network topology problem, optimal references for mutual comparison are lacking.

The best known and mostly used problems, which could be given the status of neural network benchmarks, are prime candidates for such an analysis. These benchmarks can be subdivided into those that are completely defined by their input-output pattern pairs, like for example the eXclusive OR (XOR) function, and those that are not, like many real-world problems. This publication focuses on fully defined Boolean benchmarks, since these are not subject to generalization issues.

Most neural networks are first order neural networks, where each connection is an information carrier from a source neuron to a destination neuron. If one considers higher order connections, where more than one neuron provides data for a connection, much smaller topologies are expected to be found as compared to first order neural networks. These networks, especially the two-layer ones (without hidden layers), are therefore of great interest with respect to minimal topologies. In this publication, as is most often the case, multiplication is used for combining the inputs for high order connections.

The computational complexity of the recall phase of a neural network[2] depends mainly on the computations at the neuron level (summations, multiplications, and the calculation of the non-linear function). The number of non-linear function evaluations is proportional to the number of neurons in the network, which can be considered proportional to the number of connections. The key variable in the topological complexity is therefore the number of multiplications ($\mathcal{M}$) and additions ($\mathcal{A}$) performed, which is determined by the number and type of connections. In general:

Definition: A *minimal neural network (topology)* for a given problem is a neural network topology with a minimal computational complexity that enables the problem to be solved adequately.

For all practical reasons, the topological complexity of (high order) neural networks can be adequately described by $\mathcal{M}$, optionally augmented by $\mathcal{A}$ for fine-tuning. Thus:

Definition: A *minimal high order neural network (topology)* for a given problem is a neural network topology which allows the problem to be solved and where $\mathcal{M}$ is minimal. In case of multiple solutions with equal $\mathcal{M}$, $\mathcal{A}$ can be used for fine-tuning.

For first order networks, this can be further simplified to:

Definition: A *minimal first order neural network (topology)* for a given problem is a neural network topology with a minimal number of connections that solves the problem adequately.

[2]Note that the computational complexity of the learning phase depends strongly on the chosen learning rule and is therefore unsuitable for universal comparisons.

Where appropriate, the two metrics $\mathcal{M}$ and $\mathcal{A}$ will be provided for the given network topologies in order to facilitate easy comparison of solutions. Note that for normal (McCulloch and Pitts) neurons:

- for each connection in a high order neural network: $\mathcal{M}_i = \Omega_i - 1$, where Ω_i is the order of the connection, plus at each neuron another d^{in} multiplications are performed between its incoming weights and its inputs, where d^{in} is the *fan-in* of this particular neuron, and
- for each neuron in a neural network $\mathcal{A} = d^{in}$, including the addition (or subtraction) of the neuron bias. (Note that the quantity d^{in} is also included in $\mathcal{M}$.)

2 Minimal Topologies

2.1 Finding Minimal Topologies

The method, which not only finds the minimal topology, but also all possible weight settings, consists of generating all possible topologies (that can yield a potential solution), ordered in increasing complexity. Next, for each of the possible topologies[3], starting with the one with the lowest complexity, the function performed by the network for each input pattern is written as an equation with non-linear function calls. For a first order three layer interlayer connected neural network with one output, this can be summarized as:

$$f(\mathbf{x}) = \phi(\mathbf{w_2}^T \phi(\mathbf{W_1 x})), \tag{1}$$

where $\mathbf{x}$ is the input vector $(x_1, \ldots, x_{N_1})$, N_1 the number of input neurons, $f(\mathbf{x})$ the desired function value, ϕ a non-linear function, $\mathbf{w_2}^T$ the transpose of the weight vector between the hidden neurons and the only output neuron (including the output bias), and $\mathbf{W_1}$ the weight matrix of weights between the first and second layer (including the hidden layer bias vector). For each topology, this is a system of P non-linear equations in $W{+}N$ unknowns, where P is the total number of different input patterns, W the number of weights, and N the number of neurons in the network. The exact nature of the non-linear function is not important, as long as it is monotonic and yields a positive value when an input greater than the bias is supplied and a negative value for inputs smaller than the bias[4]. Without loss of generality the non-linear function is in this publication replaced by the Heaviside function ($\mathcal{H}$), which is a hard-limiting threshold:

$$\mathcal{H}(x) = \begin{cases} 0 & \text{if} \quad x \leq 0 \\ 1 & \text{if} \quad x > 0 \end{cases} \tag{2}$$

The equations containing the non-linear functions can therefore be substituted by inequalities. This system of inequalities is now solved by mutual substitution. The first topology that has a solution (a viable non-empty set of values for all weights and biases), is the minimal topology for the given problem.

Note that output biases are optional and can be zero in this method.

[3]For convenience it is often easier to subdivide each candidate topology into subtopologies with one output each

[4]When non-monotonic activation functions are allowed, smaller solutions can often be obtained [Sharkey-92].

2.2 Minimal Topologies for the XOR problem

2.2.1 Exclusive OR

For one of the simplest functions of neural computing interest, the XOR function [Minsky-69], it is possible to determine the minimal topology exactly. The XOR function ($y = f(x_1, x_2)$) is fully determined by:

inputs $x_1\ x_2$	output $f(x_1, x_2)$
0 0	0
0 1	1
1 0	1
1 1	0

Although the minimal first order interlayer connected topology is known to be the one shown in figure 1, a formal proof was lacking. This has now been proved using the method described below. Analogously, the minimal topologies are determined for neural networks that have, besides interlayer, also supralayer connections (which 'skip' at least one layer). Finally, the smallest solution of all is found using high order neural networks.

2.2.2 First Order Minimal XOR Solutions

Due to space constraints, the only proof for minimality presented here is the one for the high order case (see section 2.2.3). The minimal first order solutions found by using the method described above, are shown by means of figures. The minimal first order interlayer connected topology for the XOR problem is shown in figure 1, which has a complexity: ($\mathcal{M} = 6$, $\mathcal{A} = 6$), including the output bias. The minimal first order topology for the XOR problem (using supralayer connections) is shown in figure 2, which has ($\mathcal{M} = 5$, $\mathcal{A} = 5$), including the output bias.

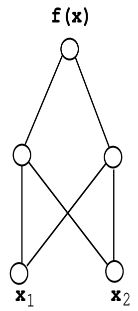

Figure 1: Minimal first order interlayer connected XOR topology.

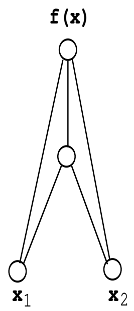

Figure 2: Minimal first order XOR topology.

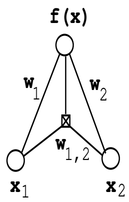

Figure 3: Minimal high order XOR topology.

2.2.3 High Order Minimal XOR Solution

Since the high order topology yields the smallest minimal topology, the equations are the simplest for this case. The smallest XOR topology with a high order connection is the two layer solution with one second order connection connecting both inputs (see figure 3).

It is easily shown that this is a viable solution. Since there are only two layers, only the input layer index is specified for the weights and w_0 stands for the output bias. The generic equation is:

$$f(\mathbf{x}) = \mathcal{H}(x_1 w_1 + x_2 w_2 + x_1 x_2 w_{1,2} - w_0). \tag{3}$$

which gives:

$$0 = \mathcal{H}(-w_0) \tag{4}$$

$$1 = \mathcal{H}(w_1 - w_0) \tag{5}$$

$$1 = \mathcal{H}(w_2 - w_0) \tag{6}$$

$$0 = \mathcal{H}(w_1 + w_2 + w_{1,2} - w_0) \tag{7}$$

The first three equations (4, 5, and 6) yield respectively: $w_0 \geq 0$, $w_1 > w_0$, and $w_2 > w_0$. Combining the last equation (7) and $w_1 > w_0$ yields: $w_1 + w_2 + w_{1,2} \leq w_0 < w_1$, which gives $w_{1,2} < -w_2$ and using $w_2 > w_0$ together with equation 7, one finds analogously $w_{1,2} < -w_1$. The solution can be summarized by:

$$w_1, w_2 > w_0 \geq 0 \quad \text{and} \quad w_{1,2} < -\max(w_1, w_2). \tag{8}$$

Any set of weights that obeys the above criteria gives a minimal high order network solution to the XOR problem. For a zero bias this gives: ($\mathcal{M} = 4$ and $\mathcal{A} = 2$).

There is however a very simple solution with only one second order connection, if one translates the binary XOR into a bipolar XOR problem, where the inputs and outputs are chosen from $\{-1, 1\}$ instead of $\{0, 1\}$. Any non-zero connection strength will do, but a weight value of one yields a solution which does not even need the output bias, so ($\mathcal{M} = 1$, $\mathcal{A} = 0$).

3 High Order Neural Network Solution to Benchmarks

3.1 The PDP Benchmarks

The best known collection of neural network benchmarks is the set of small problems described in chapter eight of the PDP book [Rumelhart-86]. In the PDP book, the problems are solved by multi-layer first order neural networks. For four of these problems (parity, symmetry, addition, and negation) small solutions using two layer high order neural networks are presented below.

3.2 The Parity Problems

The parity problems are generalizations of the XOR problem. They are known to be among the most difficult problems to be learnt by a neural network, since the closest input patterns, measured by their Hamming distance, have opposite function values. The straightforward two layer high order topology for any size parity problem is a fully interlayer connected one, whose weights of order Ω are set to: $(-2)^{\Omega-1}$, which means a total number of connections of: $2^{N_1} - 1$, and a complexity of

$$\mathcal{M} = \sum_{\Omega=1}^{N_1} \Omega \begin{pmatrix} N_1 \\ \Omega \end{pmatrix} \quad \text{and} \quad \mathcal{A} = 2^{N_1} - 2, \;\text{ where } \begin{pmatrix} n \\ k \end{pmatrix} \equiv \frac{n!}{(n-k)!k!}.$$

Analogous to the bipolar XOR solution, the parity problems also have bipolar solutions with one high order connection. Again, any non-zero connection strength will do, but a weight value of one yields a solution which does not even need the output bias, so $\mathcal{M} = N_1$, $\mathcal{A} = 0$.

3.3 The Symmetry Problems

Another set of XOR related problems are the symmetry problems (XOR is the inverse of the 2-symmetry problem, where a zero function value implies symmetry). For simplicity, the equivalent inverse symmetry problems are considered here. These problems can be expressed by small two layer high order neural networks by taking the XORs of all possible input pairs i and $N_1 - i$. Note that for uneven symmetry problems the value of the middle input is unimportant. Example topologies are those with all possible first order interlayer connections (except the middle one, for N_1 uneven) having weights equal to one and all second order connections, which pair inputs i and $N_1 - i$, with weights equal to minus two. This gives[5]: $\mathcal{M} = 4 \left\lfloor \frac{N_1}{2} \right\rfloor$ and $\mathcal{A} = 3 \left\lfloor \frac{N_1}{2} \right\rfloor - 1$. See figure 4 for a second order solution to the four input symmetry problem. Adopting a bipolar representation can also be beneficial for the symmetry problems. For bipolar inputs, there exist solutions with only $\left\lfloor \frac{N_1}{2} \right\rfloor$ second order connections: connect all input pairs i and $N_1 - i$ with weights equal to one; ($\mathcal{M} = 2 \left\lfloor \frac{N_1}{2} \right\rfloor$ and $\mathcal{A} = \left\lfloor \frac{N_1}{2} \right\rfloor$ - 1).

3.4 The Addition Problem

The two bit binary addition problem has three outputs and can therefore be split up into three subproblems. The subproblem for the third and least significant bit is simply a XOR problem between the second and the fourth input, which represent the two least significant bits of the two binary numbers to be added; ($\mathcal{M} = 4$, $\mathcal{A} = 2$).

The subproblem for the first and most significant bit can be solved with three connections: two first order connections from inputs one and three, and one second order connection combining the second and the fourth input, where the output bias plus all three weights are equal to one (1); ($\mathcal{M} = 4$, $\mathcal{A} = 3$). This network is shown in figure 5.

The subproblem for the middle bit can be expressed in a seven connection solution, as shown in figure 6; ($\mathcal{M} = 16$, $\mathcal{A} = 6$). This brings the total topological complexity for this solution of the addition problem to $\mathcal{M} = 24$, $\mathcal{A} = 11$.

[5] $\lfloor x \rfloor$ denotes the *floor function*, which yields the biggest integer smaller than or equal to x.

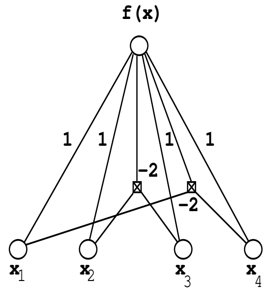

Figure 4: High order solution to the four input symmetry problem.

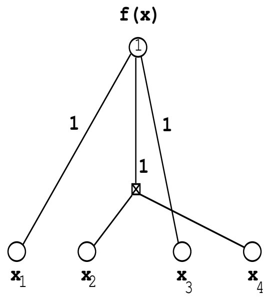

Figure 5: High order solution to the most significant output bit of the addition problem.

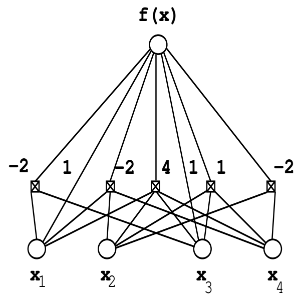

Figure 6: High order solution to the middle output bit of the addition problem.

3.5 The Negation Problem

Since the negation problem can be interpreted as three XOR problems between the first input (the negation bit) and each of the other inputs, a nine connection solution is apparent; ($\mathcal{M} = 12$, $\mathcal{A} = 6$).

4 Constructive Algorithms for High Order Neural Networks

4.1 The Computational Capabilities of High Order Neural Networks

In this section, without any loss of generality, only networks with one single output are considered (see section 1). Since any Boolean function can be written in *disjunctive normal form*, a two layer fully interlayer connected high order neural network can learn any Boolean function.

Even more general, a two layer fully interlayer connected high order neural network has 2^{N_1} degrees of freedom [Fiesler-93.1], and the total number of possible input patterns for any neural network is d^{N_1}, where d stands for the number of different values an input can assume. A fully interlayer connected high order neural network can therefore theoretically learn any function with bivalued inputs ($d = 2$), since for any N_1 points, there exists a (unique) interpolating polynomial of order $N_1 - 1$.

More concretely, it is possible to generate two layer high order neural networks for any given Boolean function. In the next two sections, a fully interlayer connected solution is presented, followed by a partially connected one that yields remarkably small topologies. For simplicity it is assumed that the function value for a zero input vector is zero ($f(\mathbf{0}) = 0$).

4.2 A Fully Interlayer Connected Solution

The following algorithm constructs a two layer fully interlayer connected high order neural network for an arbitrary Boolean function:

Algorithm 1

Construct a two layer fully interlayer connected high order topology and assign weights as follows:

FOR ALL $\mathbf{x} \neq 0$ DO
$W_\omega = (2\,\mathcal{H}(f(\mathbf{x})) - 1)\ \Lambda(\|\omega\|)$

$$\text{where } \Lambda(n) = \begin{cases} 1 + \sum_{k=1}^{n-1} \binom{n}{k} \Lambda(n-k) & \text{for } n \in \mathcal{Z}^+ \\ 0 & \text{for } n = 0 \end{cases},$$

W_ω is the weight of the connection which combines all inputs from the set of input neurons ω, $\|S\|$ denotes the cardinality or the number of elements of set S, and $\mathcal{Z}^+$ the set of positive integers. Note that $\|\omega\|$ equals the order of the connection.

Note that the function $\Lambda(n)$, also known as the *preferential arrangement* function[6], is the lowest order function possible; it can therefore be substituted by any faster increasing function, like for example: $2^{n^{n-1}}$. Nevertheless, even when using the preferential arrangement function, the weight values increase very rapidly with the order of the network.

4.3 A Partially Connected Solution

The previous algorithm has the drawbacks that both the number of connections and the values of the connections are exponential in the number of inputs. Both these disadvantages are relieved by the following algorithm, which enables the generation of two layer sparsely connected high order neural networks, also for arbitrary Boolean functions:

Algorithm 2

Assign interlayer connection weights for a two layer high order neural network as follows:

FOR ALL W_ω DO
  $W_\omega = 0$
FOR $n = 1$ TO $n = N_1$ DO
  FOR ALL $\omega : \|\omega\| = n$ DO
    FOR ALL $\omega' : \emptyset \subset \omega' \subset \omega$ DO
      $s = \sum_{\omega'} W_{\omega'}$
      IF $\mathcal{H}(f(\mathbf{x})) \neq \mathcal{H}(s)$ THEN
        $W_\mathbf{x} = 1 - s - \mathcal{H}(s)$

To construct the topology, for each non-zero weight create its associated interlayer connection.

Given the binary domain and an output bias of zero, this algorithm finds the same small networks as the ones shown in section 3. Note that it is possible to find smaller topologies when allowing a non-zero bias or non-binary input values.

[6]For a table of function vales of the preferential arrangement function, see for example [Gross-62].

5 Summary

A straightforward technique for finding minimal neural network topologies for very small problems is presented and an example is given for the minimal high order solution of the XOR problem. This exhaustive search method implies a combinatorial explosion of inequalities and is therefore not useful for larger networks, although a possible future generation of "smart" symbolic computational software that can handle inequalities might change this. For many of the often cited Boolean benchmarks described in [Rumelhart-86] very small solutions using high order neural networks are also presented. Finally, to illustrate the potential power of high order connections, two constructive algorithms for high order neural networks are given for arbitrary Boolean functions, one for fully connected networks and one for partially connected ones.

Acknowledgements

The author would like to thank Jehoshua Bruck for his contribution concerning the computational capabilities of high order neural networks and Eddy Mayoraz for pointing out the small bipolar solutions to the parity problems. Special thanks to Georg Thimm for his helpful comments and suggestions, and to Indu Saxena and Jeroen van Valburg for their proofreading.

References

[Fiesler-93.1] E. Fiesler, H. J. Caulfield, A. Choudry, and J. P. Ryan, **Maximum Interconnection Topologies for Neural Networks**, submitted to *IEEE Transactions on Neural networks.*

[Fiesler-93.2] E. Fiesler, **Topology Modifying Neural Networks**, submitted to *Expert Systems* Learned Information, Oxford, UK, ISSN: 0266-4720.

[Gross-62] O. A. Gross, **Preferential Arrangements**, *The American Mathematical Monthly*, volume 69, number 1, pages 4–8, January 1962.

[Minsky-69] Marvin L. Minsky and Seymour A. Papert. **Perceptrons**. MIT Press, Cambridge, Massachusetts, 1969.

[Rumelhart-86] David E. Rumelhart, James L. McClelland, and the PDP Research Group, **Parallel Distributed Processing, Explorations in the Microstructure of Cognition, Volume 1: Foundations**, The MIT Press, Cambridge, Massachusetts, 1986, ('A Bradford Book'), ISBN: 0-262-18120-7.

[Sharkey-92] Noel E. Sharkey, **Computing Parity on a Single Weight Layer: a Nonmonotonic Learning Function**, Connection Science Technical Report number TR 218, Department of Computer Science at Exeter, Exeter, Devon, UK, 1992,