Spectral Technique for Hidden Layer Neural Network Training

Terry Windeatt, Robert Tebbs
Dept of Electronic Eng., University of Surrey, Guildford, Surrey, UK
Email: t.windeatt@surrey.ac.uk

Indexing terms: Multi-layer Perceptrons, Sequential Learning Systems, Rademacher-Walsh Spectrum, Spectrum Synthesis, k-Monotonic Check.

Abstract. We propose a new constructive algorithm for learning binary-to-binary mappings. Weight constraints derived from a spectral summation are used to check separability during the partitioning phase, and to limit hyperplane movement during training.

Corresponding Author: Terry Windeatt
Dept of Electronic Eng., University of Surrey, Guildford, Surrey, GU2 5XH, UK
Email: t.windeatt@surrey.ac.uk

For submission to Pattern Recognition Letters

1. Introduction

For solving classification tasks with Backpropagation (Backprop) the architecture is usually selected in advance as a compromise between supporting underlying decision boundaries and between fitting any idiosyncrasies of the training set. It is difficult to make an appropriate choice without running into well-known convergence problems. Constructive methods assume that architecture is not known in advance but rather grow the network a node at a time as needed. As reported by Muselli (1995), some of these growth algorithms are specifically devoted to solving binary-to-binary mappings thereby handling support of underlying decision functions in the Boolean domain. We are interested in constructive procedures that use Threshold Logic Units (TLU) and that are built around the concept of single perceptron training. Marchand and Golea (1993) characterise the partitioning step as finding a halfspace consistent with all the positive examples and a maximal subset of negative examples. Finding the maximum subset is a NP-hard problem (Marchand and Golea, 1993), and so an approximation algorithm is required. A common choice is the pocket variant of classical perceptron algorithm originated by Gallant (1990), which updates weights only after obtaining a longer run of correct classifications. Although guaranteed to converge on non-separable data, the pocket algorithm has no upper bound on number of iterations. Alternate methods of partitioning used in constructive algorithms include: (i) quality function to achieve faithful layers (Barkema et al., 1993), (ii) thermal perceptron that gradually reduces the size of the sensitive zone around a hyperplane (Frean, 1992), (iii) linear programming approach to include negative examples in the separable subset (Marchand and Golea, 1993), (iv) necessary checks for separability from threshold logic theory (Keibek, 1992) and (v) topological strategy for visiting vertices of the binary hypercube (Mascioli and Martinelli 1995).

We select the Sequential Learning (SL) model of recruiting neurons, described in Marchand et al. (1990), which builds hidden units by a partitioning method that sequentially excludes clusters of patterns of the same target. In our implementation of SL we incorporate a *k*-monotonic check into the partitioning step to find maximal separable subsets. The original goal of SL was to regularly partition the input hypercube so that (i) vertices not separated by a hyperplane are of the same classification, and (ii) each hyperplane separates vertices of different classification. We use the irregular partitioning version of SL that can choose between positive and negative subsets, thereby guaranteeing a single separable hidden-layer.

We represent the target mapping using the Rademacher-Walsh spectrum (described in Beauchamp, 1984), although any spectral ordering would be suitable. The Rademacher functions (e.g. see Hurst et al., 1985) are an incomplete set of orthogonal functions that can generate the complete set of Walsh functions, though not in original Walsh order. This alternative order, referred to as Rademacher-Walsh, is the one most frequently used for logic design. Previous attempts to use spectral techniques for logic network synthesis are reported by Falkowski and Perkowski (1992) and include disjoint spectral decomposition, spectral summation and spectral translation. The advantage of a spectral representation over vertices in the binary hypercube is that each individual spectral coefficient contains global information about the function.

In Section 2 we explain the background behind our representation of a Boolean function based on the concepts of binary transition propagation and sensitivity. We show in Section 3 how the representation facilitates a check for monotonicity using derived weight constraints for a TLU implementation. By formulating the constraints in terms of excitatory and inhibitory spectral contributions, we show in Section 4 how to incorporate a check for separability into the Sequential Learning model. The basic techniques described in this paper are not new. The novelty lies in combining them to produce a method that constrains network growth for learning noisy binary-to-binary mappings. In the final Section we present implementations of simple Boolean functions to demonstrate the approach, and describe a character recognition experiment which represents a first step towards applying the method to pattern classification.

2. Sensitivity and Binary Transition Propagation

Consider a feedforward network $\mathbf{N}$ represented as a graph with nodes as logic elements. If we assign binary values to inputs, conventional logic simulation allows us to propagate these values to all network lines. Now assume that we want to determine the effects of changing the binary value of a single line in the network. Following the approach of Roth (1966), we simulate injection and propagation of a $\bar{v}/v$ transition, $v \in \{0,1\}$, on line ℓ having logic value $\bar{v}$ by comparing operation in two networks. From $\mathbf{N}$ we derive a network that we label $\mathbf{N}_{\oplus}$ ($\oplus$ = logic exclusive-OR) by making a cut at ℓ and changing the value on ℓ from $\bar{v}$ *to* v. The point of cut is treated as a pseudo-input thus pruning the branch and ignoring the subtree that feeds the branch. The

changed value at ℓ in general leads to further logic changes between ℓ and network output. Thus network $\mathtt{N}_{\oplus}$ differs from $\mathtt{N}$ only in the assigned logic values, and binary transitions are traced by specifying a suitable formalism (such as D-notation of Roth, 1966) that compares logic values of corresponding lines in $\mathtt{N}$ and $\mathtt{N}_{\oplus}$. Propagating these binary transitions through networks is a complex process, about which there is an extensive literature in the context of finding test inputs for detecting logical faults in combinational circuits. As explained in (Fujiwara, 1985), various formalisms have been devised using this approach to efficiently compute the Boolean difference of function $f(x_1, x_2 \ldots x_n)$ with respect to input x_j given by

$$df(\mathbf{X})/dx_j = f(x_1,\ldots,x_{j-1},1,x_{j+1}, \ldots x_n) \oplus f(x_1,\ldots,x_{j-1},0,x_{j+1}, \ldots x_n)$$

From $df(X)/dx_j$ we can find, for example, all input patterns for which f changes value in response to binary transition on x_j, given by the union of two sets

$$\{\mathbf{X} : x_j . df(\mathbf{X})/dx_j = 1\} \cup \{\mathbf{X} : \bar{x}_j . df(\mathbf{X})/dx_j = 1\}$$

Furthermore this expression holds for hidden node h if x_j is replaced by $h(\mathbf{X})$ and $f(\mathbf{X})$ by $f(\mathbf{X},h)$.

We call the logic value $\bar{v}$ sensitive (insensitive) iff the injected $\bar{v}/v$ transition is (is not) propagated to network output. The representation for each logic value is enhanced by attaching another binary value that indicates sensitivity (σ), and for convenience we write the jth component as $x_j^{\sigma_j}$. To find $x_j^{\sigma_j}$ from a truth table, compare input patterns that are unit Hamming Distance (HD) apart and set $\sigma_j = 1$ if target response differs and $\sigma_j = 0$ otherwise. Note that σ_j, for all j and over all patterns, is available from the first stage of standard logic minimisation techniques such as Quine-McCluskey tabular method. As an example consider the 2-input NAND truth table with target $\sigma = 1$ (Table 1), noting that $\sigma_2 = 0$ for the two patterns with $x_1=0$, since the target value is independent of x_2. For an n-input NAND (NOR) gate exactly 1 or n inputs propagate binary transitions, but this restriction does not hold for TLUs which may be regarded as generalised logic gates (e.g. Muroga, 1971). Assignments of σ reflect functionality, and Akers and Krishnamurthy (1989) devised a constraint propagation system by tabulating integral inequalities between counts of $(0^0,0^1,1^0,1^1)$. In our system described in Section 4, we use a similar counting technique to characterise functionality of TLUs but we interpret it as a spectral summation.

Table 1: NAND truth table with sensitivity (σ) superscript

$x_1^{\sigma_1}$	$x_2^{\sigma_2}$	Target
0^0	0^0	1^1
0^1	1^0	1^1
1^0	0^1	1^1
1^1	1^1	0^1

When the output of one gate feeds the input of another and the logic assignments permit a transition to be propagated through both gates, we say that a sensitised path has been established. To implement an arbitrary Boolean function with a multi-layer network of TLUs, it is necessary to establish sensitised paths of two types, excitatory and inhibitory. We define the path as inhibitory if $x^{\sigma} = 0^1$ on line l for true pattern ***(X : f(X) = 1)*** or 1^1 for false pattern ***(X : f(X) = 0)***, and conversely for excitatory path. An example showing sensitised path polarities for XOR is given in Section 5.

3. k-monotonic check and Separability

The following definitions for n-dimensional Boolean functions *f(X)* and *g(X)* are adapted from Muroga (1971):

f implies *g*, $f \subseteq g$ if any *X* satisfying *f(X)=1* also satisfies *g(X)=1*, but not necessarily conversely.

f is *k*-comparable if either $f_A \subseteq f_{\bar{A}}$ or $f_A \supseteq f_{\bar{A}}$ holds for each *k*-assignment *A*, where *k*-assignment is an assignment of binary values to *k* out of *n* variables

f is *m*-monotonic if *f* is *k*-comparable for every *k* such that $1 \leq k \leq m$. If *m = 1*, *f* is unate.

If *m = n*, *f* is completely monotonic, a necessary but not sufficient (unless $n \leq 8$) condition for linear separability.

This classification shows that there are degrees of separability for which appropriate checks provide necessary and increasingly restrictive conditions for a data set to be separable. The ordering of the classes for all possible 2^{2^n} Boolean functions, as given by Hurst et. al (1985), is *1*-monotonic, *2*-monotonic, higher monotonic, completely monotonic and finally linearly separable which constitutes the most restrictive class.

We split the *k*-monotonic check into $k = 1$ and $k \geq 2$ checks:

1) If any two patterns p_1 and p_2 can be found such that for the jth component:

$$x_j^{(p1)} = \overline{x_j^{(p2)}},\ \sigma_j^{(p1)} = \sigma_j^{(p2)} = 1 \qquad (1)$$

the function is not *1*-monotonic since *1*-assignment $A = \{x_j \rightarrow 1\}$ shows it is not *1*-comparable. Failure of the *1*-monotonic check implies that it is not possible to implement with a single TLU since the requirement is for TLU orientation weight w_j to be both excitatory and inhibitory (positive and negative).

2) If a single pattern can be found such that for the *i*th and *j*th components:

$$x_i = x_j = f(\boldsymbol{X}),\ \sigma_i = 1,\ \sigma_j = 0 \qquad (2)$$

then $f_A \supset f_{\bar{A}}$ and the 2-assignment $A = \{x_i \rightarrow 1, x_j \rightarrow 0\}$ implies $w_i > w_j$. A similar check exists for $w_i < w_j$ if $x_i = x_j = \overline{f(\boldsymbol{X})}$. Since the weight constraint relationship is transitive, we can use weight ordering to check *k*-monotonicity, $k \geq 2$.

4. Spectral Summation and Constructive Algorithm

The transformation of binary data can be carried out using a variety of matrices that differ only in row ordering. For example, the Hadamard, Walsh and Rademacher-Walsh transform matrices use different orderings of the functions that collectively constitute the closed set. Our goal is to construct an approximate network that is capable of implementing incompletely specified and noisy functions so we are interested in information content rather than direct computation of the inverse transform. Therefore we can use any spectral ordering and the coding can be {0,1} or {+1,-1). Hurst et al. (1985) give the subscript notation and corresponding meaning for coefficients up to third order as follows:

s_0	correlation between f(**X**) and constant	
s_i $i=1...n$	correlation between f(**X**) and x_i	(3)
s_{ij} $i,j = 1...n, i \neq j$	correlation between f(**X**) and $x_i \oplus x_j$	
s_{ijk} $i,j,k = 1...n, i \neq j \neq k$	correlation between f(**X**) and $x_i \oplus x_j \oplus x_k$	
.....		

First order coefficients provide a unique identifier of a Boolean function if it is linearly separable. There is however no known mathematical relationship between these parameters and weight/threshold values

of a TLU implementation, although tables giving minimum integer threshold realisation exist for $n \leq 7$. We note that each element $x_j^{\sigma_j}$ of a pattern simply indicates by $\sigma_j = 1$ if the respective pair of patterns, unit HD apart in *j*th component, provides a net contribution to the spectrum. We define the contribution $x_j^{\sigma_j}$ for pattern $\boldsymbol{X}$ as excitatory if $x_j = f(\boldsymbol{X})$ and inhibitory if $x_j = \overline{f(\boldsymbol{X})}$, just as we defined sensitised path polarities in Section 2. The first order spectral coefficients come directly from a count of these contributions $(0^1,1^1)$ on each input. To characterise non-separable functions we need the higher order coefficients which can be calculated by a spectral summation, described in (Windeatt and Tebbs, 1996). The calculation is based on the excitatory and inhibitory first order contributions, ignoring any component with $\sigma_j = 0$ and with the sign of each contribution given by the correlation defined in (3).

We concentrate on first and second order coefficients for formulating weight constraints from the spectral equivalent of conditions (1) and (2). For false patterns, the pairwise constraint becomes:

- $w_i > w_j$ with w_i positive, holds between components *i* and *j* if a zero contribution to s_j from *j* is accompanied by an excitatory contribution to s_i from *i* and to s_{ij} from *i*
- $w_i < w_j$ with w_i negative, holds between components *i* and *j* if a zero contribution to s_j from *j* is accompanied by an inhibitory contribution to s_i from *i* and an excitatory contribution to s_{ij} from *i*

Our goal is to use these weight constraints to identify a minimal hidden layer of feature detectors that makes the problem separable. First note that it is always possible to find a single TLU feature detector that separates a pattern from its nearest (unit HD) neighbours. For example, if $x \in \{+1,-1\}$ choose each orientation weight as $cx_j\sigma_j$ where *c* is arbitrary positive scaling constant, and constrain bias weight w_B by

$$0 < w_B + \sum_{j=1}^{n} c\sigma_j < c \qquad \sigma \in \{0,1\} \tag{4}$$

In {0,1} coding, choose each orientation weight as $-c(-1)^{x_j}\sigma_j$ and constrain bias by

$$0 < w_B + \sum_{j=1}^{n} cx_j\sigma_j < c \qquad \sigma \in \{0,1\} \tag{5}$$

We show how to implement simple Boolean functions with c = 1 in Section 5. For large problems it is not computationally feasible to consider *k*-monotonic constraints with all combinations of individual features, so we employ SL with *k*-monotonic check. To construct a network capable of generalisation we introduce a spectral count threshold, with the threshold level specifying the spectral count below which evidence of non-monotonicity is ignored. The effect of the threshold is to allow some non-separable patterns into the separable set, and since we order the patterns by their contribution to non-separability we find the largest subset for a given threshold. The check for separability is recursive in that (*k-1*)-monotonicity is assumed before checking *k*-monotonicity, and we consider both cases of removing true and false patterns. The two main steps are as follows:

Step1: For all *j (j = 1...n)* use first order spectral coefficients, decomposed into excitatory and inhibitory contributions, to identify minimal fraction of false or true patterns that fail *1*-monotonic check

Step2: For all *ij (i,j = 1...n*), use first and second order spectral coefficients, decomposed into excitatory and inhibitory contributions, to compute pairwise weight constraints, and identify minimal fraction of false or true patterns that fail *k*-monotonic check, $k \geq 2$, for increasing values of *k*.

The procedure starts by removing patterns causing *1*-monotonic check to fail and continues recursively for increasing values of *k* until the set is completely monotonic. We recruit neurons one at a time, as with SL, and use the same fractional comparison for choosing between true and false subsets. Lack of guarantee of separability (Section 3) turns out not to be a practical limitation for large incompletely specified functions. However, if necessary we could incorporate the perceptron learning rule for a fixed number of steps; this is similar to the approach of (Keibek 1992) which limits the number of calls to the perceptron algorithm by first checking complete monotonicity.

5. Implementations

To see how the scheme works it is instructive to synthesise simple Boolean functions, as described in the first three examples below. The final example demonstrates its application to constraining hyperplane movement in a BackProp network for character recognition.

(a) **Separable function** $f(X) = x_1x_2 + x_1x_3 + x_2x_3x_4$ is implemented by a single TLU with weights $[w_1\ w_2\ w_3\ w_4{:}w_B] = [6\ 4\ 4\ 2{:}\ -9]$. The following three methods find the same set of weights for this function: (i) first order spectral contributions and table-look-up (Hurst, 1985), or (ii) solving weight constraints derived from (1) and (2) (Windeatt and Tebbs, 1995), or (iii) iteratively in a few cycles by constraining bias weight according to (5) with c=1, while orientation weights are adjusted by the unity increment perceptron rule (Windeatt and Tebbs, 1993).

(b) **XOR** (Table 2) fails *1*-monotonic check for both inputs, and removing a single pattern leads to two hidden nodes (Figure 1), one for excitatory and one for inhibitory paths.

Table 3: XOR table

$x_1^{\sigma_1}$	$x_2^{\sigma_2}$	Target
0^1	0^1	0^1
1^1	1^1	0^1

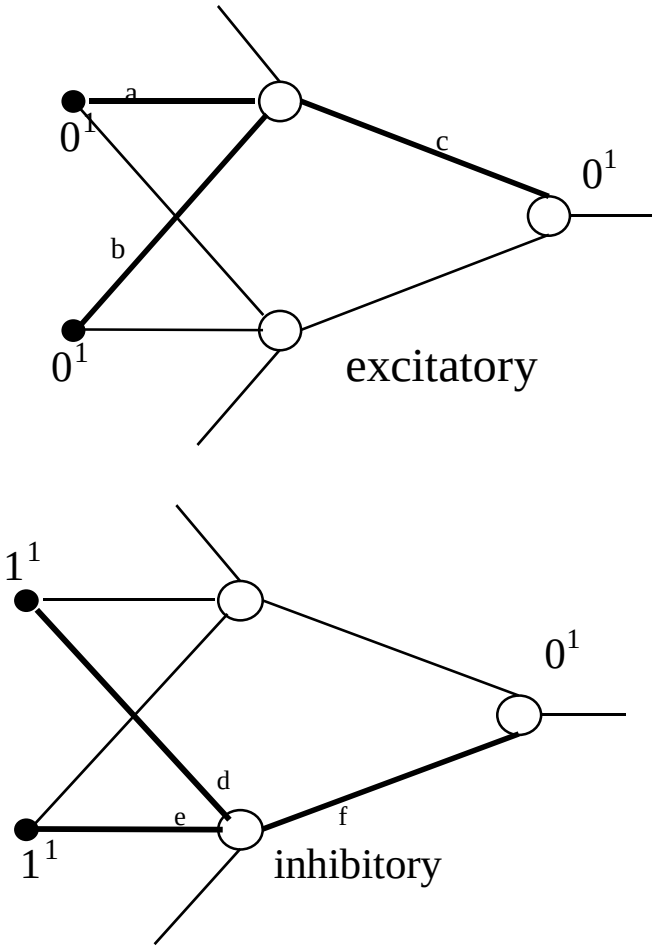

Figure 1: Architecture showing excitatory and inhibitory paths (Section 2): two possible polarity assignments are (a,b,c) = {(+,+,+),(–,–,–)} and (d,e,f) = {(+,+,–),(–,–,+)}

A hard-limiting activation function can be considered as an infinite-slope sigmoid, so to investigate the effect of constraining weight space search, we replaced TLU outputs with sigmoid activations. This enables calculation of standard Backprop derivatives and weight updates, but we constrained the bias weights

of hidden nodes according to (5). We repeated the experiment for varying gain and momentum, and produced the plot of convergence time versus gain versus momentum (Figure 2Figure). This 3D plot shows network convergence to be well-behaved with parameter choice not being critical even for gain values approaching 10. In contrast, a typical 3D plot for the equivalent experiment with conventional Backprop, shown in Figure 3, converges only for gains less than 1. Furthermore, a clearly defined minimum exists in gain range 0.2 to 0.6, with this minimum decreasing as momentum increases up to a value of 0.9.

(c) **Non-separable function** $f(X) = x_1x_2 + x_3x_4$ is *1*-monotonic, since no pair of patterns in its truth table can be found to satisfy (1). Pairwise constraints come from applying (2) and are given by:

$w_1>w_3$, $w_3>w_1$, $w_1>w_4$, $w_4>w_1$, $w_2>w_3$, $w_3>w_2$, $w_2>w_4$, $w_4>w_2$. Therefore the function fails *2*-monotonic check for all pairs of inputs except between x_1,x_2 and between x_3,x_4. To implement it we do not need both excitatory and inhibitory paths from any single input. A possible architecture is to restrict connectivity to hidden layer of two nodes, so that one node is used for x_1,x_2 and another for x_3,x_4.

(d) **Recognising printed characters** on a 16x16 binary grid independent of rotation and with added random noise provides data representing a complex Boolean function that is incompletely specified. We rotated characters through three hundred and sixty degrees, using data at twenty-degree intervals for training and testing each degree with increasing levels of random noise. This gave us 72 training samples with 1440 noise-free samples for testing. The goal of the experiment was to use the approximate network model and constraints derived from the method of Section 4 to enhance performance of a BackProp network. We modified the technique for obtaining σ by looking at the nearest neighbour only, and in our simple experiments we did not see significant improvement in generalisation by considering larger number of neighbours. Without any evidence to weight one component more than another, we assumed σ inversely proportional to Hamming Distance and apportioned the σ contributions equally (total contribution per pattern pair = HD * HD^{-1} =1). Using the *1*-monotonic check, we derived two hidden layer feature detectors for each character, with orientation weights set according to the first order spectral contributions. The output layer weights (separate output for each character) and hidden layer bias weights were tuned using gradient descent with sigmoid activations on all nodes. For comparison, we used the same architecture with conventional BackProp using different sets of random initial weights. We also tried increasing the number of hidden nodes

and used different learning parameters in an attempt to optimise performance. For classifying character rotation (training every 20 degrees, testing each degree), we consistently achieved better than 2 percent error rate with the constrained architecture, compared with an average result of 8 percent for the best of the runs using conventional Backprop. We also varied gain and momentum and found similar comparison of convergence surfaces as shown in Figure 2. A more complex experiment for classifying different fonts showed constrained Backprop achieving convergence easily on cases where conventional Backprop failed to converge for a variety of initial weights and learning parameters (Windeatt and Tebbs, 1995). A further advantage of our method is that the role of hidden layer feature detectors in the classification process is more transparent. We also found more consistent classification performance when random noise was increased up to the fifteen-percent level.

6. Discussion

By decomposing spectral contributions into excitatory and inhibitory components, we formulate a set of weight constraints that assist in choosing multi layer perceptron architecture and in constraining the weight space search. We do not yet have enough experience to comment in general on appropriate choice of k for checking k-monotonic subsets with incompletely specified functions. However in our character recognition experiments we have observed good generalisation if patterns are partitioned with $k=1$, and the resulting hidden node feature detectors used in a Backprop network. For realistic problems, a weakness of our approach is the heuristic determination of σ. The goal is to make the spectral technique of obtaining constraints more robust in the presence of noisy and possibly contradictory data. For example, an alternative for determining σ with incompletely specified functions would be to use symmetry check based on spectral coefficients (e.g. see Hurst et al., 1985).

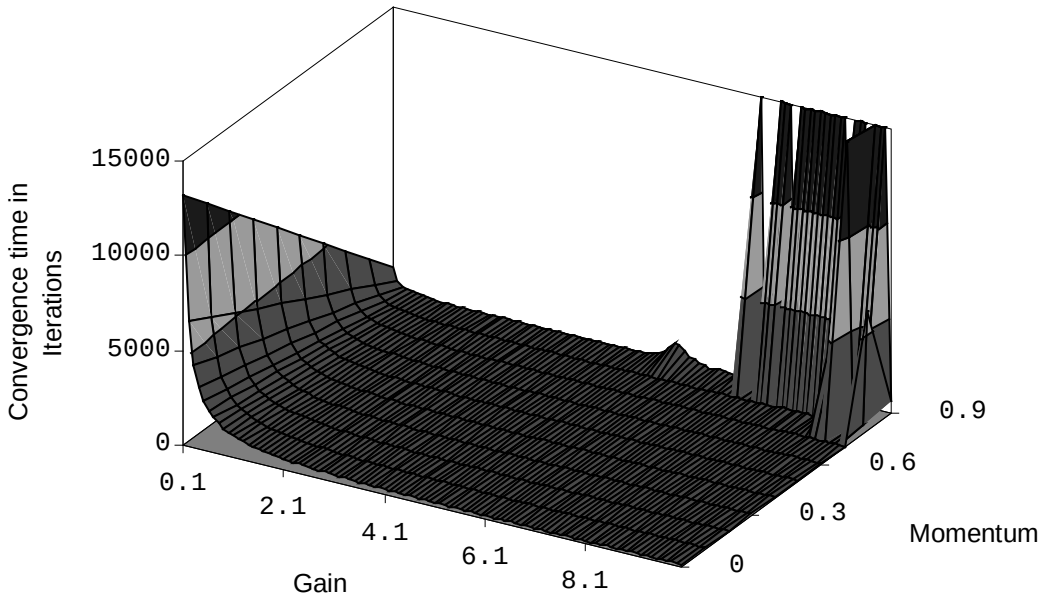

Figure 2: ***Convergence surface for constrained Backprop learning XOR***

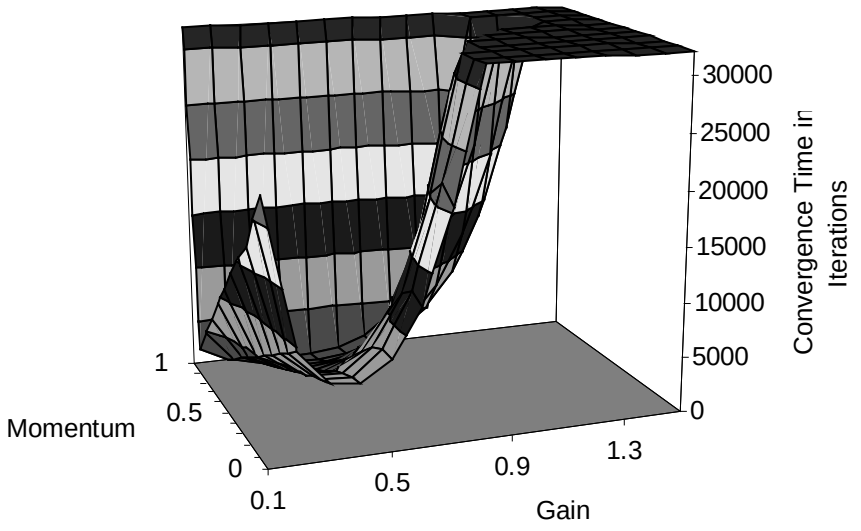

Figure 3: ***Convergence surface for conventional Backprop learning XOR***

References

Akers, S.B. and B. Krishnamurthy (1989). Test counting: A tool for VLSI testing, *IEEE Design and Test of Computers*, Oct. 1989, 58-77.

Barkema, G. T., H. Andree and A. Taal (1993). The patch algorithm: fast design of binary feedforward neural networks, *Network Computation in Neural Systems* 4 (3), 393-407.

Beauchamp, K. G. (1984). *Applications of Walsh and Related Functions,* Academic (New York).

Falkowski, B. J. and M. A. Perkowski (1992). Effective computer methods for the calculation of Rademacher-Walsh Spectrum for completely and incompletely specified Boolean Functions, *IEEE Trans. on Computer-Aided Design* 11(10), 1207-1226.

Frean, M. (1992). 'Thermal' perceptron learning rule, *Neural Computation* 4, 946-957.

Fujiwara, H. (1985).. *Logic Testing and Design for Testability*, MIT Press.

Gallant, S. I. (1990). Perceptron-based learning algorithms, *IEEE Trans. on Neural Networks* 1(2), 179-192.

Hurst, S.L., D.M. Miller and J.C. Muzio (1985) *Spectral Techniques in Digital Logic*, Academic Press.

Keibek, S.A.J., G.T. Barkema, H.M.A. Andreee, M.H.F. Savenlie and A. Taal (1992). A fast partitioning algorithm and a comparison of binary feedforward neural networks, *Europhys. Lett.*, 18 (6) 555-559.

Marchand, M., M. Golea and P. Rujan (1990). A convergence theorem for sequential learning in two-layer perceptrons, *Europhys. Lett*. 11 (6), 487-492.

Marchand, M. and M. Golea (1993). On learning simple neural concepts: from halfspace intersections to neural decision lists, *Network Computation in Neural Systems* 4 (1), 67-85.

Mascioli, F. M. and G. Martinelli (1995). A constructive algorithm for binary neural networks: The Oil-Spot Algorithm, *IEEE Trans. Neural Networks*. 6(3), 794-797.

Muroga, S. (1971). *Threshold Logic & its Applications*, Wiley.

Muselli, M. (1995). On sequential construction of binary neural networks, *IEEE Trans. Neural Networks* 6(3), 678-690.

Roth, J.P. (1966) Diagnosis of automata failure: a calculus and a method, *IBM J. Research and* Development 10, 278-291.

Windeatt, T. and R.G. Tebbs (1993). Functionality constraints in perceptron training. In: *Proc. Weightless Neural Network Workshop*, York, UK, April 1993, 1-8.

Windeatt, T. and R.G. Tebbs (1995). Perceptron training improvements using functionality constraints. In: *Proc. 12th European Conf. on Circuit Theory and Design*, Istanbul, Turkey, August 1995, 679-682.

Windeatt, T. and R.G. Tebbs (1996). Analytical feature extraction and spectral summation. In*: Proc. 13th IAPR Internat. Conf. Pattern Recognition*, Vienna, August 1996, IV 315-319.