Title: A VLSI OPTIMAL CONSTRUCTIVE ALGORITHM FOR CLASSIFICATION PROBLEMS

Author(s): Valeriu Beiu
           Sorin Draghici
           Ishwar K. Sethi

Submitted to: 7th International Conference on Artificial Neural Networks
October 7-10, 1997
Lausanne, Switzerland

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by the University of California for the U.S. Department of Energy under contract W-7405-ENG-36. By acceptance of this article, the publisher recognizes that the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. The Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy.
DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.
A VLSI optimal constructive algorithm for classification problems
Valeriu Beiu¹, Sorin Draghici², Ishwar K. Sethi²

¹ Los Alamos National Laboratory
Division NIS-1, MS D466
Los Alamos, New Mexico 87545, USA

² Vision & Neural Networks Laboratory
Dept. of Comp. Sci., Wayne State University, 431 State Hall, Detroit, MI, USA

Abstract - If neural networks are to be used on a large scale, they have to be implemented in hardware. However, the cost of the hardware implementation is critically sensitive to factors like the precision used for the weights, the total number of bits of information and the maximum fan-in used in the network. This paper presents a version of the Constraint Based Decomposition training algorithm which is able to produce networks using limited precision integer weights and units with limited fan-in. The algorithm is tested on the 2-spiral problem and the results are compared with other existing algorithms.

1. Introduction

It is widely accepted that the hardware implementation is the most cost effective solution for large scale use of computational paradigms. Artificial neural networks (NN's) are no exception. One particular difficulty regarding the neural network's migration towards hardware is that their present software simulations do not take into consideration issues which are very important from a VLSI point of view. For instance, most software simulations use floating point arithmetic and either double or simple precision weights. Any hardware implementation would be unreasonably expensive if it were to implement floating point operations and to store so many bits for each weight. Limited precision weight neural networks are better suited for such purposes because a limited precision requires fewer bits for storing the weights and also simpler computations. This brings a decrease in the size of the VLSI chip and therefore, a lower cost.

Another important issue for hardware implementations of neural networks is the complexity of the network. The complexity of a network can be described using different measures such as the number of neurons or the depth (i.e. the number of edges on the longest input/output path). For VLSI implementation purposes, the depth can be put into correspondence with the delay T and the number of neurons can be put into correspondence with the area A of a VLSI chip. However, these measures are not the best criteria because the area of a single neuron depends on the precision of its associated weights, as already mentioned. Better criteria are the total number of connections [Beiu, 1994] and references therein) the total number-of-bits needed to represent the weights [Bruck, 1990; Williamson, 1991, Denker, 1988] or the sum of all the weights and thresholds [Beiu, 1994]. In the following, the total number of bits will be adopted as one appropriate measure of network complexity.

A different important element is the maximum fan-in of the units. [Abu-Mostafa, 1988] and [Hammerstrom, 1988] are among the first papers to suggest that a limited maximum fan-in can be beneficial. Since then, some researchers have presented algorithms and architectures able to work with limited fan-in [Klaggers, 1993; Phatak,
1994). Recently, it has been shown that a small and constant fan-in (in the range \([6..9]\)) is indeed optimal from a VLSI point of view because it minimizes the \(AT^2\) measure of the chip [Beiu, 1995].

In conclusion, a VLSI friendly constructive algorithm would be able to:

- decide the appropriate architecture for the network
- construct a solution network (both architecture and weights)
- do so using limited precision integer weights in a range chosen by the user and
- construct solutions with a limited fan-in chosen by the user

This paper will present such an algorithm, some results of tests performed on the 2-spiral problem and a comparison with other related algorithms.

2. Theoretical Foundations

An important and interesting question is: how many bits of information do we need for a given problem or, alternatively, how complex a problem can we solve with a network of a given complexity? Recent work has established bounds which relate the number of bits (as defined in information theory) with the complexity of the problem described by the number of patterns \(m\) and the minimum distance \(d\) between the closest patterns from opposite classes. Recently, using an entropy based reasoning, Beiu has calculated first an upper bound of \(mn\left\lceil \log D/d \right\rceil + 5/2\) in [Beiu, 1996] and then improved it to \(mn\left\lceil \log D/d \right\rceil + 2\)/2 in [Beiu, 1997]. In these expressions, \(n\) is the number of dimensions, \(D\) is the radius of the smallest hypersphere including all patterns and the logarithm is in base 2. These results have been obtained considering a uniform quantization of the space which in turn, assumes that the separating hyperplanes can be placed in any necessary position. For integer weights, these assumptions do not hold anymore. The case of weights restricted to integer values in the range \([-p, p]\) is analyzed in [Draghici, 1997; 1997a and 1997b]. [Draghici, 1997] shows that neural networks using limited precision weights are indeed capable of solving any classification problem if the distance between the closest patterns of opposite classes is at least \(1/p\):

**Proposition 1** (from [Draghici, 1997]). Using integer weights in the range \([-p, p]\), one can correctly classify any set of patterns for which the minimum distance between two patterns of opposite classes is \(d_{\text{min}} = 1/p\).

Further work in this field has shown that the number of bits necessary for solving a given classification problem with limited precision weights can be bounded [Beiu, 1996, 1997; Draghici, 1997, 1997a, 1997b].

However, all these results show only that it is theoretically possible to implement a solution able to solve the given classification problem with limited precision integer weights.

Several algorithms which limit the precision of the weights used have already been presented. There are several different ways of dealing with the imposed limitations. For instance, [Hohfeld, 1992; Xie, 1991. Cogins 1994; Tang, 1993] use a dynamic rescaling of the weights and a corresponding adaptation of the gain of the activation function. [Hohfeld, 1992; Vincent 1992] rely on probabilistic rounding whereas
[Dundar, 1995; Khan 1994, Kwan 1992, 1993; Marchesi 1990; Simard 1994; Tang 1993] use weight values which are restricted to powers of two. Several other possibilities for improving the VLSI efficiency of neural network implementations (including a VLSI friendly constructive algorithm) are presented in [Beiu, 1994].

[Draghici, 1997c] presents a version of a constructive algorithm which produces networks with limited weights. Although VLSI friendly, due to the use of limited precision weights, this algorithm does not limit the fan-in of the units used. Thus, networks with an unreasonable maximum fan-in might result. This paper will present an enhanced version of this algorithm VCBD which produces a network which uses both limited weights and limited fan-in units. For a chosen fan-in in the range [6,9], the resulting networks will be VLSI optimal in the AT sense [Beiu, 1995].

3. The VLSI Friendly Constraint Based Decomposition Algorithm

The VLSI Friendly Constraint Based Decomposition algorithm presented here, VCBD (see Fig. 5) is based on the Constraint Based Decomposition (CBD) algorithm [Draghici, 1994; 1996a; 1996b] and ICBD [Draghici, 1997c]. As already mentioned, ICBD builds a network using limited precision integer weights but does not limit the fan-in of the units.

The CBD algorithm is based on the classical ‘divide and conquer’ strategy. It builds a 3 layer network with a first layer of units implementing hyperplanes, a second layer of AND units and a third layer of OR units (see also [Sethi, 1990; Beiu, 1994; Ayestaran, 1993, Bose, 1993]. This structure allows a simple approach to limiting the fan-in.

The solution can be expressed in DNF form. For instance, a solution with maximum fan-in 5 would be: \( C_1 = h_0 + \overline{h}_0 h_1 h_2 + \overline{h}_0 h_1 h_2 h_3 h_4 \) (the units on the first layer implement \( h_0 \) to \( h_4 \) in Fig. 1). The maximum fan-in can be reduced by inserting a new unit \( F \) and rewriting this solution as: \( C_1 = h_0 + \overline{h}_0 h_1 h_2 + (\overline{h}_0 h_1 h_2) h_3 h_4 = h_0 + \overline{h}_0 h_1 h_2 + F h_3 h_4 \). Fig. 2 presents the new architecture of the network (with max fan-in = 3). Unit 10 implements the same function as before but now has a fan-in of 3.

4. Experimental results

The algorithm was tested on several problems. Fig. 3 presents the XOR problem solved with integer weights in the range [-3, 3]. Note the fact that the solution uses only 2x3x3=18 bits for storing the weights and compare this with the 2x3x3x3=192 bits which is the minimum number of bits used by any algorithm employing floating point weights. Fig. 4 presents a solution for the 2 spiral problem. The solution uses 40 hyperplanes; each hyperplane uses 3 integer weights in the range [-5, 5] which can be stored on 4 bits. The total number of bits used for this solution is 40x3x4=480. Also, note that the 4 bits have not been exploited fully; they can implement weights in the range [-7, 7] not only in the range [-5, 5] which has been used. The maximum fan-in of this network is 7 which is within the optimal limits [6,9].

In comparison, the floating point version of the CBD algorithm [Draghici, 1994] managed to produce a solution with only 34 hyperplanes. However, each of these 34 hyperplanes used three floating point weights for a total of 34 x 3 x 32 = 3264 bits (for the particular compiler used). [Lang, 1988] solved the problem with 138
connections. If we assume these connections were floating point numbers we obtain \(138 \times 32 = 4416\) for the total number of bits used. It can be noted that, although the number of hyperplanes used by our algorithm is larger, the number of bits is actually greatly reduced.

**Fig. 1.** An architecture generated by the ICBD. Note that one unit has fan-in 5.

**Fig. 2.** An architecture generated by OCBD for the same problem. The square unit is an AND gate.

**Fig. 3.** The XOR problem solved with integer weights in the range \([-3, 3]\). The solution used 2 hyperplanes for a total number of \(2 \times 3 \times 3 = 18\) bits.

**Fig. 4.** The 2 spiral problem solved exclusively with integer weights in the range \([-4, 4]\). The solution used 40 hyperplanes for a total number of \(40 \times 3 \times 4 = 480\) bits.

### 5. Conclusions

The paper presented a constructive algorithm able to build solutions using arbitrary precision weights and limited fan-in. The algorithm does not require an a priori design of the network architecture and has a guaranteed convergence. It has been shown on a few examples that the algorithm is able to build solutions to complicated problems (like the 2 spirals) which use integer weights in a very limited range and an optimal fan-in. The number of bits necessary for storing the weights for the problems studied was approximately one order of magnitude less than the number of bits used in classical algorithms using floating point weights. At the same time, the number of bits used was approximately 40% less than the number of bits given by several limits (480 vs. 776).

Future work will include extensive testing of this algorithm on different real world problems. Also, we intend to test the CBD algorithm with different LPIW weight changing mechanisms which might bring improvements in the generalization performances of the solutions generated.
• separate (region, \(C_1\) set of patterns in class 1, \(C_2\) set of patterns in class 2, factor, range, fan-in)
  • if fan-in == max_fan-in then
    • create a new \(F\) unit which will implement the given factor
    • new_factor = \(F\)
    • Separate(region, \(C_1\), \(C_2\), new_factor, range, 1)
  • else
    • Build a subgoal \(S\) with patterns \(x_1^{C_1}\) and \(x_1^{C_2}\) taken at random from \(C_1\) and \(C_2\). Delete \(x_1^{C_1}\) and \(x_1^{C_2}\) from \(C_1\) and \(C_2\).
    • Choose \(h\), a canonical hyperplane (in range) which separates \(x_1^{C_1}\) and \(x_1^{C_2}\).
    • For each pattern \(p\) in \(C_1 \cup C_2\).
      • Add \(p\) to the current subgoal \(S\)
      • Save \(h\) in \(h\)-copy
      • Try to find another canonical hyperplane (in range) which separates the current subgoal \(S\)
    • if not success then
      • Restore \(h\) from \(h\)-copy and remove \(p\) from \(S\)
    • Let new_factor = factor and \((h, '+')\)
    • If the positive half-space determined by new_factor contains only patterns in the same class \(C_j\) then
      • Classify new_factor as \(C_j\) /* done */
      • else /* call the procedure recursively in the smaller region */
      • Delete from \(C_1\) and \(C_2\) all patterns which are not in \(h^+\). Store the result in new\(_C_1\) and new\(_C_2\).
      • Separate(\(h^+, new\_C_1, new\_C_2, new\_factor, range, fan-in+1\) )
    • Let new_factor = factor and \((h, '-'\) )
    • If the negative half-space determined by new_factor contains only patterns in the same class \(C_j\) then
      • Classify new_factor as \(C_j\) /* done */
      • else /* else call the procedure recursively in the smaller region */
      • Delete from \(C_1\) and \(C_2\) all patterns which are not in \(h^-\). Store the result in new\(_C_1\) and new\(_C_2\).
      • Separate(\(h^-, new\_C_1, new\_C_2, new\_factor, range, fan-in+1\) )

Fig. 5. The OCBD algorithm for limited precision integer weights and limited fan-in. The canonical hyperplanes depend on the given range (precision) \(p\).

6. Bibliography
