Circuit Complexity for Neural Computation

Kai-Yeung Siu
Dept. of Electrical & Computer Engineering
University of California
Irvine, CA 92717

Vwani Roychowdhury
School of Electrical Engineering
Purdue University
West Lafayete, IN 47907

Thomas Kailath
Dept. of Electrical Engineering
Stanford University
Stanford, CA 94305

Abstract

One common model for ANNs is the threshold circuit - *multilayer feedforward network of linear threshold gates. We provide a rigorous analysis of this model via a circuit complexity theoretic approach by focusing on the basic computational properties of threshold circuits with binary inputs. The model of threshold circuits is shown to be computationally more powerful than the conventional model of AND-OR logic circuits. In particular, we present the best known results on the depth of threshold circuits in implementing common arithmetic functions such as multiplication, division, and sorting. Moreover, we investigate the issues of depth-size tradeoffs by demonstrating that a small increase in the depth can significantly decrease the size required in the threshold circuit for the class of symmetric functions.*

1 Introduction

Recent interest in the application of artificial neural networks [5, 3] has spurred research interest in the theoretical study of such networks. While neural networks have found wide application in many areas, the behavior and the limitation of these networks are far from being understood. One common model of a neural network is a *threshold circuit*. Incidentally, the study of threshold circuits, motivated by some other complexity theoretic issues, has also gained much interest in the area of computer science.

Threshold circuits are Boolean circuits in which each gate computes a linear threshold function, whereas in the conventional model of *unbounded fan-in* Boolean circuits only AND, OR, NOT gates are allowed. A Boolean circuit is usually arranged in layers such that all gates in the same layer are computed concurrently and the circuit is computed layer by layer in some increasing *depth* order. We define the *depth* as the number of layers in the circuit. Thus each layer represents a unit delay and the depth represents the overall delay in the computation of the circuit. In the rest of this paper, we shall use the terms "threshold circuits" and "neural networks" interchangeably.

Theoretical computer scientists have used *unbounded fan-in* Boolean circuits as a model to understand fundamental issues of parallel computation. To be more specific, this computational model should be referred to as *unbounded fan-in* parallelism, since the number of inputs to each gate in the Boolean circuit is not bounded by a constant. The theoretical study of unbounded fan-in parallelism may give us insights into devising faster algorithms for various computational problems than would be possible with bounded fan-in parallelism. In fact, any nondegenerate Boolean function of n variables requires at least $\Omega(\log n)$ depth to compute in a bounded fan-in circuit. On the other hand, in some practical situations, (for example large fan-in circuits such as programmable logic arrays (PLAs) or multiple processors simultaneously accessing a shared bus), unbounded fan-in parallelism seems to be a natural model. For example, a PLA can be considered as a depth-2 circuit of (arbitrary fan-in) AND/OR gates.

In the Boolean circuit model, the amount of resources is usually measured by the number of gates, and is considered to be 'reasonable' as long as it is bounded by a polynomial (as opposed to exponential) in the number of the inputs. For example, a Boolean circuit for computing the sum of two n-bit numbers with $O(n^3)$ gates is 'reasonable', though circuit designers might consider the size of the circuit impractical for moderately large n. One of the most important theoretical issues in parallel computation is the following: *Given that the number of gates in the Boolean circuit is bounded by a polynomial in the size of inputs, what is the minimum depth (*i.e. *number of layers) that is needed to compute certain functions?*

It was known that [13, 4] for many simple functions, such as the *parity* of n Boolean variables, or the multiplication of two n-bit numbers, any *fixed* depth

1058-6393/91 $01.00 © 1991 IEEE

(*i.e.* independent of n) conventional Boolean circuit of unbounded fan-in AND/OR gates (AND-OR circuit) computing these functions must have exponential (in n) number of gates. Another way of interpreting these results is that circuits of AND/OR gates computing these 'hard' functions which use polynomial amount of chip area must have *unbounded delay* (*i.e.* delay that increases with n). In fact, the lower bound results imply that the minimum possible delay for multipliers (with polynomial number of AND/OR gates) is $\Omega(\log n / \log\log n)$. These results also give theoretical justification why it is impossible for circuit designers to implement fast parity circuit or multiplier in small chip area using AND, OR gates as the basic building blocks.

The model of threshold circuits seems so much more powerful than the conventional model of AND-OR circuits. In fact, it was known [2, 1, 7] that many common arithmetic functions such as multiplication, division and sorting can be computed by polynomial-size threshold circuits of 'constant' depth. However, the issue of how small these 'constants' can be was not examined before. For example, it was not clear from previous known results that the 'constant' of a threshold circuit for division can be reduced to below 20. Because of the fundamental importance of these arithmetic operations, for both practical and theoretical reasons, it is interesting to determine the smallest possible depth of the circuits that compute these functions.

One of our main contributions is to demonstrate that small constant depth neural networks for multiplication, sorting, division and related arithmetic functions can be constructed. These results have the following implication on their practical significance: *Suppose we can use analog devices to build threshold gates with a cost (in terms of delay and chip area) that is comparable to that of AND, OR, logic gates, then we can compute many basic functions much faster than using conventional circuits.*

Another important issue in neural computation is the trade-off between the *size* and the *depth* of neural networks. This issue arises naturally in the implementation of neural networks. Sometimes when the time for the computation is not as crucial, one might want to have a smaller network at the expense of a slight increase in the time for computation. It is interesting to investigate if increasing the depth by a small additive constant can significantly reduce the size of the network. In fact, our most recent work [12] has given a positive answer to this question. The best known prior results [6] in computing any symmetric Boolean function of n variables with neural networks require $O(n)$ threshold gates and two layers. It has been an open problem for 30 years to determine if the number of threshold gates can be further reduced for an arbitrary symmetric Boolean function. We are able to show that by increasing the depth by one, i.e. by using a 3-layer neural network, one can reduce the number of threshold gates from $O(n)$ to $O(\sqrt{n})$. For functions with 'periodic structures' such as the parity, we have extended the depth-size tradeoffs results to higher depth: for every integer $d > 0$, one can construct neural networks of depth-$(d+1)$ and size $O(dn^{1/d})$ computing them. Moreover, using classical results from the theory of rational approximations and harmonic analysis, we obtained lower bound results by showing that our networks are almost optimal in the size ($\Omega(dn^{1/d-\epsilon})$ threshold gates are needed for any fixed $\epsilon > 0$).

In the following section, a summary of all these results will be presented. Because of space limitation, we refer the interested readers to our related journal papers [8, 9, 10, 12, 11] for the detailed descriptions and proofs of our results.

2 Summary of Main Results

Definition 1 A *linear threshold function* $f(X) : \{0,1\}^n \rightarrow \{0,1\}$ is a Boolean function such that

$$f(X) = sgn(F(X)) = \begin{cases} 1 & \text{if } F(X) \geq 0 \\ 0 & \text{if } F(X) < 0 \end{cases}$$

where

$$F(X) = \sum_{i=1}^{n} w_i \cdot x_i + w_0$$

$$X = (x_1, ..., x_n) \in \{0,1\}^n$$

Our model of neural network is simply a multilayer feed-forward network of linear threshold gates, *i.e.* a threshold circuit.

Note that the above definition does not impose any restriction on the *fan-in* and *fan-out* of each threshold gate in the network, *i.e.* each gate can have arbitrarily many inputs from previous layer and can feed its output to arbitrarily many gates in subsequent layers. Since neural networks are characterized by their massive parallelism, the unbounded fan-in parallelism of threshold circuits seems a natural assumption.

Given two input n-bit (unsigned) integers, X and Y, the product of X and Y (denoted by

$PROD(X,Y)$) will be a $2n$-bit integer, the (truncated) quotient of X divided by Y (denoted by $DIV_k(X,Y)$) will be an $(n+k)$-bit number (where the last k bits constitue the truncated fractional part of the quotient). Given n input n-bit positive integers z_i, $i = 1, ..., n$, their multiple product $\Pi_{i=1}^{n} z_i$ will be an n^2-bit integer denoted by $MPROD(z_1, ..., z_n)$, and the sorted list denoted by $SORTING((z_1, ..., z_n)$ is the same list of integers sorted in nondecreasing order. While the computation of these functions requires any fixed depth AND-OR circuit to have an exponential size, the following theorem states that these functions can all be computed by polynomial-size neural networks of small constant depth, independent of the input size n. These are the best known results on the depth of neural networks in computing these arithmetic functions.

Theorem 1

1. $PROD(X,Y)$ and $SORTING((z_1, ..., z_n)$ can be computed by polynomial-size neural networks of depth-4.
2. $DIV_k(X,Y)$ can be computed by polynomial-size neural networks of depth-5.
3. $MPROD(z_1, ..., z_n)$ can be computed by polynomial-size neural networks of depth-6.

Definition 2 A Boolean function $f : \{1,0\}^n \rightarrow \{1,0\}$ is said to be *symmetric* if

$$f(x_1, \ldots, x_n) = f(x_{(1)}, \ldots, x_{(n)})$$

for any permutation $(x_{(1)}, \ldots, x_{(n)})$ of $(x_1, \ldots, x_n)$, or equivalently, the value of f only depends on the number of 1's in the input variables.

An example of a symmetric function is the PARITY function, which is defined to be 1 if the number of 1's in the inputs is *odd* and is 0 otherwise. Symmetric functions constitute an important class of Boolean functions and come up very often in the design of logic circuits. As mentioned before, the best known prior result is that any symmetric function (of n variables) can be computed by a depth-2 neural network of size $O(n)$. The following results show that a significant reduction in the size can be achieved if the depth is increased by a small constant.

Theorem 2 Any symmetric function of n variables can be computed in a depth-3 neural network with $2\sqrt{n} + O(1)$ threshold gates.

Theorem 3 For any integer $d > 0$, the PARITY function of n variables can be computed in a depth-$(d+1)$ neural network with $O(dn^{1/d})$ threshold gates.

Using novel techniques from harmonic analysis and the theory of rational approximations, we are able to show that the sizes of our neural networks for symmetric functions are *almost optimal*. (For details of the proof, see [11])

Theorem 4 For any fixed $\epsilon > 0$ and any integer $d > 0$, almost all symmetric functions of n variables (including the PARITY function) require any depth-(d+1) neural network at least $\Omega(dn^{1/d-\epsilon})$ threshold gates to compute.

Acknowledgements

This work was done while the the first author was a research student associate at IBM Almaden Research Center, San Jose, CA and supported in part by the Joint Services Program at Stanford University (US Army, US Navy, US Air Force) under Contract DAAL03-88-C-0011, the SDIO/IST, managed by the Army Research Office under Contract DAAL03-87-0033, and the Department of the Navy, NASA Headquarters, Center for Aeronautics and Space Information Sciences under Grant NAGW-419-S6.

References

[1] P. W. Beame, S. A. Cook, and H. J. Hoover. Log depth circuits for division and related problems. *Siam J. Comput.*, 15:994–1003, 1986.

[2] A. K. Chandra, L. Stockmeyer, and U. Vishkin. Constant depth reducibility. *Siam J. Comput.*, 13:423–439, 1984.

[3] J. L. McClelland D. E. Rumelhart and the PDP Research Group. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1.* MIT Press, 1986.

[4] J. Hastad. Almost optimal lower bounds for small depth circuits. *ACM Symp. Theor. Computing*, 18:6–20, 1986.

[5] J. J. Hopfield. Neural Networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79:2554–2558, 1982.

[6] R. Minnick. Linear-Input Logic. *IEEE Trans. on Electronic Computers*, EC 10, 1961.

[7] J. Reif. On Threshold Circuits and Polynomial Computation. In *Structure in Complexity Theory Symp.*, pages 118–123, 1987.

[8] K. Y. Siu and J. Bruck. Neural Computation of Arithmetic Functions. Special Issue on Neural Networks, *Proc. IEEE*, 78, No.10:1669–1675, October 1990.

[9] K.-Y. Siu and J. Bruck. On the Power of Threshold Circuits with Small Weights. *SIAM J. Discrete Math.*, 4(3):423–435, August 1991.

[10] K.-Y. Siu, J. Bruck, and T. Kailath. Depth-Efficient Neural Networks for Division and Related Problems. *IEEE Trans. Information Theory.* in press.

[11] K. Y. Siu, V. P. Roychowdhury, and T. Kailath. Computing with Almost Optimal Size Threshold Circuits. submitted to JCSS, 1990.

[12] K. Y. Siu, V. P. Roychowdhury, and T. Kailath. Depth-Size Tradeoffs for Neural Computation. *IEEE Transactions on Computers* , Special Issue on Neural Networks, to appear December 1991.

[13] A. Yao. Separating the polynomial-time hierarchy by oracles. *IEEE Symp. Found. Comp. Sci.*, pages 1–10, 1985.