ORIGINAL RESEARCH PAPER

Volume-7 | Issue-8 | August-2018 | PRINT ISSN No 2277 - 8179

INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH

DESIGNING OF NEURAL NETWORK FOR SWITCHING NETWORK CONTROL

Computer Science

Santosh Kumar Asst. Prof., Dept. Of Computer science & Engg., women's Institute of Technology, Darbhanga-846004,

Dr. R. K. Singh Associate Professor, M.I.T, Muzaffarpur

ABSTRACT

A neural network is a highly interconnected set of simple processors. The many connections allow information to travel rapidly through the network, and due to their simplicity, many processors in one network are feasible. Together these properties imply that we can build efficient massively parallel machines using neural networks. The primary problem is how do we specify the interconnections in a neural network. The various approaches developed so far such as outer product, learning algorithm, or energy function suffer from the following deficiencies: long training/specification times; not guaranteed to work on all inputs; requires full connectivity. Alternatively we discuss methods of using the topology and constraints of the problems themselves to design the topology and connections of the neural solution. We define several useful circuits-generalizations of the Winner-Take-All circuit - that allows us to incorporate constraints using feedback in a controlled manner. These circuits are proven to be stable, and to only converge on valid states. We use the Hopfield electronic model since this is close to an actual implementation. We also discuss methods for incorporating these circuits into larger systems, neural and non-neural. By exploiting regularities in our definition, we can construct efficient networks.

KEYWORDS

Neural Network, Topology, Hopfield electronic model, Winner-Take-All circuit.

INTRODUCTION

Neural networks are a class of systems that have many simple processors -"neurons" that are highly interconnected. The function of each neuron is simple, and the behavior is determined predominantly by the set of interconnections. Thus, a neural network is a special form of parallel computer. Although a major impetus for using neural networks is that they may be able to "learn" the solution to the problem that they are to solve, we argue that another perhaps stronger impetus is that they provide a framework for designing massively parallel machines. The highly interconnected architecture of switching networks suggests similarities to neural networks, and indeed, we present three applications in switching in which neural networks can solve the problems efficiently. The first two problems come from circuit switching: finding routes through large multistage switches and calculating a rearrangement that allows a new call to be placed through a rearrangeable switch.

In this latter problem we show that a computational advantage can be gained by using non-uniform time delays in the network. The last application is to high-speed interconnection networks, of relevance to packet switching. Using the computational speed of many neural processors working in parallel, we are able to resolve contention for paths through the network in the necessary time. We analyze this problem in detail to show the applicability of neural systems to real applications.

NEURAL NETWORKS AND PARALLEL MACHINES

For applications requiring computational speed beyond what a single processor IS capable of, increasing the number of processors can decrease the computation time. Standard parallel computing models are all fundamentally equivalent to the Turing model of computation. While, in principle, the programming of the multiple nodes is a straightforward extension of the programming of a single node, unfortunately, complications arise since the processors must spend time communicating intermediate results and waiting for other processors to send needed data. The programming and even the way that the multiple processors must be connected so that the machine isn't bogged down in this interprocessor-communication and scheduling overhead is not so well understood. As a result, the increase in speed as a function of the number of processors is significantly sublinear. We illustrate this phenomenon using data. The time using one processor for a given task is less than N times the time spent with N processors. Putting this in a comparable form:

This was from a performance test that allowed the manufacturers to use the fastest possible algorithm that they could develop to solve a system of 1000 equations and 1000 unknowns. This comparison is interesting because these machines have a variable number of processors. By comparing only within a single architecture, we can control for the differences between machines. The loss in efficiency is significant for the architectures in the graph. For example, with just seven processors, the Alliant computer spends almost 25% of each processor's time on this communications overhead. Over the domain of the data given, the loss in efficiency grows linearly with the number of processors. The linear increase in the loss in efficiency implies a decreasing amount of speedup that, if extended, would ultimately lead to an absolute decrease in the computing speed. This does not bode well for systems with many processors.

DESIGNING NEURAL NETWORKS

This idea is analogous to the work of a digital designer. Given a problem, the designer doesn't look directly at desired input and output signals and then solder a circuit using transistors, resistors, and other components; rather they analyze what they know about the problem to formulate a solution using the already available AND gates, flip-flops, etc., leading to a circuit design. A good designer will often try to produce a solution to the general class of problems, not just the specific instance that is at hand. Our approach with neural networks will be similar. Given a problem, analysis can yield much information about the problem. We often know the constraints and the direct causes of particular elements in the response. It is therefore to our advantage to incorporate this knowledge into the neural solution. As research continues on neural networks, the number of classes of neural networks which are well understood increases. These can help us to incorporate the knowledge we have about the problem. The fact that a neural network can assume quite arbitrary topologies is extremely useful. The topology of the neural network can be matched to the topology of the problem. This obviates the correspondence between the problem and the neural solution. Finally, since we know the underlying structure which is producing the neural network, we can exploit this structure to simplify the construction. These principles will be very important in guiding us to our design solutions.

To introduce some of the basic ideas, consider the infamous (in neural circles) parity check problem: Given N inputs of "-1" or "+1," output a "+1" if the number of input "1 's" is odd, otherwise output a "-1." This is often quoted as an unnatural and difficult problem for neural networks to solve. General learning algorithms have great difficulty in finding a solution to the parity check problem. But it is known that there exists a straightforward neural network to solve this for any N; we present this network below.

CONCLUSIONS

We argue that neural networks are a significant break from conventional parallel computing. Whereas the fundamental problem with conventional parallelism is how to program the multiprocessors;

with neural networks it is how to determine the connections between processors. Most current methods assume that the problem is unstructured, unnecessarily ignoring available information. By designing solutions using available information, we can produce efficient general neural solutions to whole classes of problems.

REFERENCES:

[1] Miranker, W. L., Has Parallel Computing Failed Again? A Manifesto for Parallel Computation, EE Systems Seminar presented at Caltech, March 12, 2017.
[2] Dongara, J. J., Performance of Various Computers Using Standard Linear Equations Software in a Fortran Environment. Argonne Nat. Lab. Technical Memorandum No. 23, August 13, 1988.
[3] Hopfield, J. J., Neurons with Graded Response Have Collective Computational Properties Like Those of Two-State Neurons, Proceedings of the National Academy of Sciences USA, Vol. 81, May 1984, pp. 3088-3092.
[4] Marcus, C. P., Westervelt, R. M., Stability of Analog Neural Networks with Delay, Physical Review A, Vol. 39, 1989, pp. 34 7-359.
[5] Judo, J. S., Neural Network Design and the Complexity of Learning, MIT Press, Cambridge, MA, 2015.
[6] Blum, A., Rivest, R. L., Training a 3-Node Neural Network is NP-Complete, ed. Touretzky, D. S., Advances in Neural Information Processing Systems I, Morgan Kaufman Pub., San Mateo, CA, 2012, pp. 494-501.
[7] Hinton, G. E., Sejnowski, T. J., Learning and Relearning in Boltzmann Machines, ed. Rumelhart, D. E., McClelland, J. L., Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations, MIT Press, Cambridge, MA, 2002.