Title:

NEURAL SYSTEMS DEVELOPMENT VIA CARELIA SIMULATOR

Authors:

PASI KOIKKALAINEN & ERKKI OJA

Published in:

COGNITIVA-90, MADRID Spain 20-23 Nov. 1990, accepted.

Publications from

THE
INFORMATION PROCESSING LABORATORY

Lappeenranta University of Technology

P.O.BOX 20, SF-53851 Lappeenranta
Finland

NEURAL SYSTEMS DEVELOPMENT VIA CARELIA SIMULATOR

PASI KOIKKALAINEN and ERKKI OJA

SUMMARY

In this paper the development strategies and the simulation environments for neural networks are studied. First a brief review on neural network simulators is given to introduce the readers to the state-of-the-art in the filed. The rest of the paper concentrates on the problematics of the simulation task. It is emphasized that any simulation experiment is not practical without a generic model architecture that allows the description of the neural network to be given in a structured environment. Such a framework also allows different neural paradigms to be implemented on various hardware platforms without extra work. As a result a suggestion for general purpose architecture is given and it's implementation, the Carelia simulator, is introduced. With this model a user can define a network-level unit, a layer, which is a set of neurons running a neural network algorithm. Multiple layers can be defined and they can communicate with each other. The system-level models can be built from such combinations of layers.

The Carelia simulator makes the design of algorithms and network structures possible at a higher level than in traditional programming languages. Users have tools for constructing the network architecture and monitoring the network behavior during a development phase. Interactions between real world data and neural network development are made easy.

Keywords: Neural modeling, Neural networks, Simulators, Programming environment

Authors are with the department of Information Technology, Lappeenranta University of Technology P.O.Box 20, 53851 Lappeenranta, Finland

This work was supported by Tekes Grant 4196/1989 under Finsoft project

1. The State-of-the-art in neurocomputer software

This paragraph is a summary of some of the best known neural programming environments, with selected features. More information on the subject can be found from references: Hecht-Nielsen (1988);Hanson, Crutz & Tam (1987); Koikkalainen & Oja (1990); Kraft,Frostrom, MacRitchie & Rogers (1989); Lee (1989). Paik, Gungner & Skrzypek, (1987); Rumelhart & McClelland, (1986). A closer look at these programming environments reveals similarities in their design. Many of them have some higher level language for structural design of neural networks. Algorithmic programming is done using very traditional programming languages, in many times C, with the help of a procedure or function library.

ANNE (Oregon graduate center)

- Neural simulator for Intel iPSC multiprocessor
- High level Network Description Language NDL
- Network operations are built-in C procedures

ROCHESTER CONNECTIONIST SIMULATOR (University of Rochester)

- Simulator for connectionist networks
- Network description language: C-programming language
- Network operations are built-in C procedures

IBM CONE (IBM Palo Alto)

- Neural programming environment for IBM PC/AT
- Network specification language GNL (made using YACC and LEX for C)
- Graphics display

UCLA SFINX (University of California at Los Angeles)

- High level network specification language
- Network operations are built-in C procedures
- Command interpreter for user commands

HNC ANZA (Hecht-Nielsen Neurocomputer)

- Neurocomputer co-board and development environment for SUN and PC
- Object-oriented specification language AXON
- C procedure library

SAIC DELTA (Scientific Applications International Corporation)

- Neurocomputer co-board and development environment for PC
- Object-oriented specification language ANSpec

PYGMALION (University College, London & Thomson-CSF)

- Neural C definition language
- Algorithm library
- Graphic monitor

Typically, a neurocomputer implementation is not built on special "neural" hardware. Simulations are run in normal sequential machines, the most popular ones being PC's and workstations. Neural co-processor boards have usually some popular processor like 68020 with FP unit for extra computing power and therefore they are not really "neural" computers. This is understandable since a more neural-like hardware does not allow easy algorithm development. It seems more realistic that neural algorithms are developed in traditional computers and then transferred onto special hardware.

2. Categories in neural modeling

It is useful to describe computational models via different levels of organization and abstraction. Kohonen (1988) has found the following categories of neural models useful:

1. Neuron-level; studying the properties of a neuron.
2. Network-level; the interconnected network of neurons.
3. Nervous-system-level; competing or co-operating networks of type 2.
4. Mental-operation-level; high level cognitive operations.

The lower the level categories are designed to match what is known about the brain; higher levels intend to imitate by man-made algorithms what has been observed in human information processing. Most of the presently popular neural network algorithms are network-level type. The research on nervous-system-level models has not yet caught up with the popularity of the other models, although it can be expected with good reason that system-level networks will and should play an important role for neural networks research in the future.

3. Stages in neural programming algorithms

Despite the differences in the computational algorithms, some common stages can be detected. We name these stages of computation by forward computing, matching, and updating.

3.1 Forward computing stage

This stage in the algorithm is where the neural outputs are computed from the given input pattern and the internal weight vector. All these operations are parallel in nature for a set of neurons and therefore require no knowledge about the state of the other neurons.

3.2 Matching stage

In general this stage requires knowledge about the state of all the neurons in the network. This knowledge is used for co-operation or competition between neurons in the network.

3.3 Updating stage

In this stage neural units adjust their internal weight values. Values are functions of the controls for the neurons that were defined in the matching stage. Since all the information is internal to a neuron, this stage can be computed in parallel for the neurons selected for updating.

4. The general computational model

The model is based on the network-level of thinking. First a very simple neuron-level approximation is given to form a basis for general network-level algorithms. The neuron-level computational unit is provided with operands and operations detected in most of the present computational models. These units are used in a network-level unit, a layer, which is a set of neurons provided with a superset of neuron-level operations. The realization of a layer in the simulator is usually a well known algorithm or a part of it. Multiple layers can be defined and they can communicate with each other. The nervous-system-level can be built from such combination of layers.

4.1 Neural operands

A computational model of a neuron is a unit that operates with a set of neural operands S_N. In our model there are 7 operands in the set $S_N : \{x, ex, m, o_i, t_i, c, a\}$.

4.2 Neural operations

The set of neural operations are neuron-specific and therefore do not access information from other neurons. A generic neural operation, $N(neuron, S_N)$, will update the set S_N for the *neuron*. These operations can only use the operands of a single neural unit.

4.3 Layer operands

Layer operands are the ones connecting neurons in the layer with the outside world. The basic ones are:

input vector	x (shared by all the neurons)
extra input vector	ex (shared by all the neurons)
output vector	o (outputs of all the neurons in the network)
control vector	t (controls of all the neurons in the network)
memory matrix	M (memory vectors m of all the neurons)

4.4 Layer operations

Layer operations are a superset of neural operations. In the simplest case a layer operation, $L(layer, S_L)$, performs a neuron operation in parallel with all the neurons. Layer operations are the primitives that a user can choose in neural network algorithm design. There are two groups of these operations, for computing and for communicating. The computing operations can furthermore be divided in five basic types: parallel, control, picking, targeting, and branching.

5. The user interface

The user interface (figure 1.) is designed to give the user many graphical aids when building his/her applications. In brief, the following tools are implemented using HyperNeWS programming environment for SUN workstations; (HyperNeWS is made by Artur van Hoff of the Turing Institute, UK). More detailed information about the simulator is available from authors.

- System editor; system-level design and parameters
- Program editor; for algorithm design
- Input editor; for making feed-forward connections
- Topology editor;for making lateral interconnections
- Visualization tool; for graphical monitoring

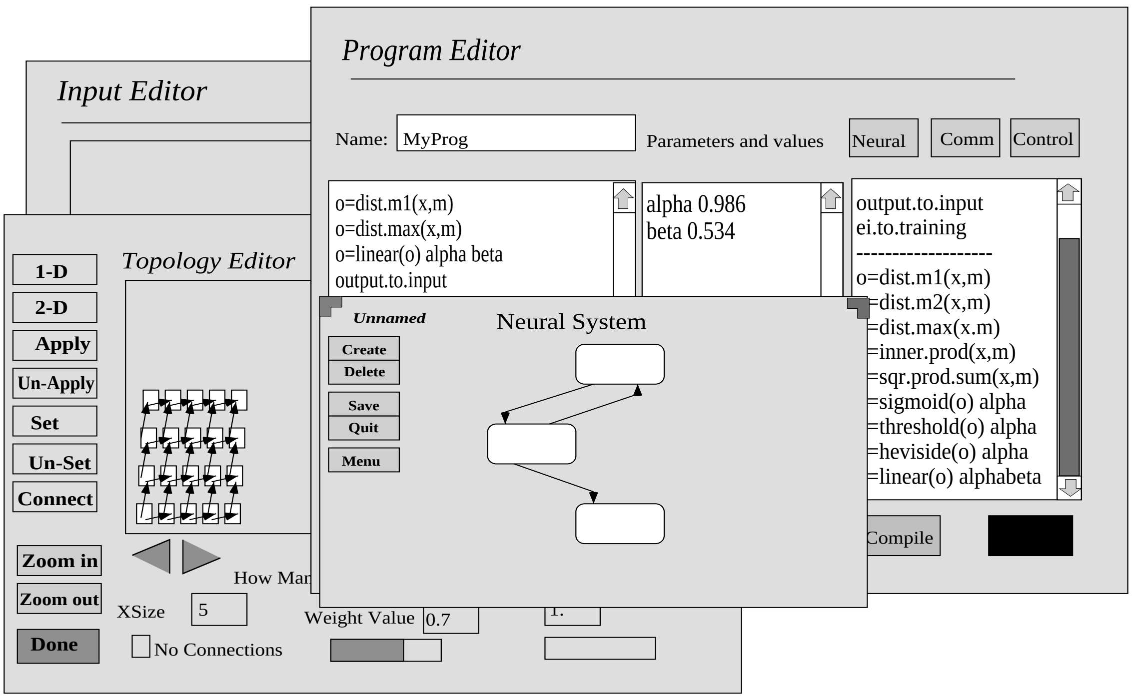

Figure 1. Neural programming interface for the Carelia simulator.

References

Hanson, W.A., Crutz, C.A., Tam, J.Y. (June 1987). CONE -Computational Network Environment, *Proceedings of the IEEE 1st International Conference on Neural Networks*, pp. III-531-III-545.

Hecht-Nielsen R. (March 1988). Neurocomputing: Picking the Human Brain, *IEEE Spectrum, 25*(3), pp. 36 - 41

Koikkalainen P. and Oja E.,The Carelia simulator: A Development and Specification Environment for Neural Networks, to be published in: Advances in Control Networks and Large Scale Parallel Distributed Processing Models, Ablex, NJ, 1990.

Kohonen T. (1988). *Self-organization and associative memory* (2nd ed.). Berlin: Springer Verlag.

Kraft, T., Frostrom, S. A., MacRitchie, B. & Rogers A. E. (1989). *The Specification of a Concurrent Backpropagation Network using Actors.* San Diego, USA: Science Applications International Corporation

Lee M.K.O (1989). PYGMALION: a European neural computing research project. *Proceedings of the Neural Computing*, pp. 179-186. London: IOP Publishing Ltd.

Paik E., Gungner, D. and Skrzypek, J. (1987). UCLA SFINX-A neural network simulation environment. *Proceedings of the IEEE 1st International Conference on Neural Networks*, pp. III-367-III-375.

Rumelhart D.E. & McClelland, J.L (Eds.) (1986) *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, Vol 1., Cambridge, MA: MIT press Lindsay, P.H. and Norman D.A., Human Information Processing, 2nd. Ed., Academic Press, 1977.