

Neural Networks in the Undergraduate Curriculum^{*1}

Ingrid F. Russell

*Department of Mathematics and Computer Science
University of Hartford, West Hartford, CT*

ABSTRACT:

Neural networks have been and will continue to be major research areas in artificial intelligence. Such models show promise in achieving human-like performance, particularly in areas such as speech and pattern recognition. Recently, neural networks have begun to find their way out of the research labs and into the realm of practical applications. Unfortunately, however, the study of such networks has been largely overlooked in the computer science undergraduate curriculum. Tomorrow's marketplace demands that computer science students be familiar with neural networks and with their problem solving abilities.

This paper presents a proposal for a neural network module to be integrated into an Introduction to Artificial Intelligence course. Such a module proved to be a very popular and successful part of such a course given by the author. Sample projects assigned in the course will be presented. In a number of these projects, students either used microcomputer application software which simulates several neural network models or programmed their own simulations on microcomputers.

INTRODUCTION

Despite the fact that today's computers are faster and more precise than human beings, humans are still far better at certain skills, for example learning. Some recent attempts have been made to narrow this gap and to create computing systems that capture essential elements of human intelligence and learning. This has given rise to new types of computing systems, called distributed processing models, also referred to as connectionist models, or neural networks.

¹The full version of this paper appeared in **Collegiate Microcomputer**, February 1991, Vol.9, No.1, pp 1-6.

* Copyright © 1991 by the Consortium for Computing in Small Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

It is believed by many researchers in the field that such systems could serve as good models for our cognitive processes. In addition, rapid advances in technology have created faster and less expensive parallel hardware systems. Given these and our belief in the potential of parallelism, it is desirable that our undergraduate computer science students be exposed to distributed systems and their problem solving capabilities. Such exposure could aid in their further understanding of distributed versus sequential processing.

The impact that neural networks are making on the field of artificial intelligence makes them an important topic to be discussed in an AI survey course. This paper outlines a neural network module that was used in a junior/senior university level Introduction to Artificial Intelligence course. It is not intended to be an in-depth discussion of these models, but attempts to give enough of an overview of some of the basic models to give students an appreciation of the field. Some suggested class projects are presented.

MODULE OUTLINE

Below is an outline of the neural network module we are proposing. The material presented was covered within a three week period.

- Brief history of neural networks
- What is a neural network ?
- Learning in a neural network
- The pattern associator
- The Hebb rule
- The delta rule
- The generalized delta rule

Our aim in this module is to provide the students with an appreciation of the capabilities of neural networks in general and help them develop a basic understanding of the properties of some of these simple models. Because of time considerations, the history of neural networks could be treated only very briefly; some discussion of this topic, though, is helpful in providing students with an understanding of the motivations for and the current state of this field. We have chosen the pattern associator as a topic of discussion for a number of reasons. First, pattern associators have demonstrated several interesting and important properties which have led to their wide use in distributed memory modeling. Such models are good at generalization. Pattern associators can be trained to act as content-addressable memories and they also degrade gracefully with inconsistent and noisy input (McClelland and Rumelhart, 1986). In addition to their significance and wide use, such models provide us with one of the most basic neural network models and thus are good to introduce in such a course.

Two learning rules are often used with pattern associators, the Hebb rule and the delta rule. Both are examples of supervised learning and have been included in the module. Other neural network models include the unsupervised competitive learning model, the adaptive resonance theory model and the Boltzmann machine. Due to time constraints, these models cannot be included in the course. However, as discussed later, they provide us with topics for possible class projects.

All books available for an introductory artificial intelligence course include at most a very short chapter on neural networks. I thus had to supplement the book with the necessary notes to cover the topics listed above. Students have used McClelland and Rumelhart (1986, 1988) as reference books. McClelland and Rumelhart (1988) includes a software package which consists of simulations of the various models presented in class. As discussed later, students used the package in some of their projects.

WHAT IS A NEURAL NETWORK ?

A neural network consists of four main parts:

- Processing units $\{u_j\}$, where each u_j has a certain activation level $a_j(t)$ at any point in time.
- Weighted interconnections between the various processing units which determine how the activation of one unit leads to input for another unit.
- An activation rule which acts on the set of input signals at a unit to produce a new output signal, or activation.
- Optionally, a learning rule that specifies how to adjust the weights for a given input/output pair.

A processing unit u_j , as shown in figure 1, takes a number of input signals, say $a_{1j}, a_{2j}, \dots, a_{nj}$ with corresponding weights $w_{1j}, w_{2j}, \dots, w_{nj}$, respectively. The net input to u_j given by:

$$\text{net}_j = \sum w_{ij} * a_{ij}$$

The new state of activation of u_j is given by:

$$a_j(t+1) = F(a_j(t), \text{net}_j),$$

where F is the activation rule and $a_j(t)$ is the activation of u_j at time t . The output signal o_j of unit u_j is a function of the new state of activation of u_j :

$$o_j(t+1) = f_j(a_j(t+1))$$

One of the most important features of a neural network is its ability to adapt to new environments. Hence learning algorithms are critical to the study of neural networks. Learning implies that a processing unit is capable of changing its input/output behavior as a result of changes in the environment. Since the activation rule is usually fixed when the network is constructed and since one cannot change the input/output vector, to change the input/output behavior one has to change the weights corresponding to that input vector. We thus need a method by which, at least during a training stage, weights can be modified in response to the input/output process sketched above. A number of such learning rules are available for neural network models.

THE PATTERN ASSOCIATOR

A pattern associator learns associations between input patterns and output patterns. One of the most interesting characteristics of such a network is the fact that it can generalize what it learns about one pattern to other similar input. Its architecture consists of two sets of units, the input units and the output units. Each input unit connects to each output unit via weighted connections. Connections are only allowed from input units to output units. Figure 2 depicts

an example of a pattern associator with five input units and three output units. The effect of a unit u_i in the input layer on a unit u_j in the output layer is determined by the product of the activation a_i of u_i and the weight of the connection from u_i to u_j . The activation of a unit u_j in the output layer is given by:

$$3 \quad w_{ij} * a_i.$$

One can adjust the connection weights in order to change the input/output behavior. However, one of the most interesting properties of these models is their ability to self-modify and learn. The learning rule is what specifies how a network changes its weights for a given input/output association. The most commonly used learning rules with pattern associators are the Hebb rule and the delta rule.

THE HEBB RULE

The Hebb rule determines the change in the weight connection from u_i to u_j by: $\Delta w_{ij} = r * a_i * a_j$, where r is the learning rate and a_i , a_j represent the activations of u_i and u_j respectively. Thus, if both u_i and u_j are activated the weight of the connection from u_i to u_j should be increased.

Examples can be given of input/output associations which can be learned by a two-layer Hebb rule pattern associator. In fact, it can be proved that if the set of input patterns used in training are mutually orthogonal, the association can be learned by a two layer pattern associator using the Hebbian learning. However, if the set of input patterns are not mutually orthogonal, interference may occur and the network may not be able to learn the associations, as is the case in the above example. This limitation of Hebbian learning can be overcome by using the delta rule. For a detailed discussion and mathematical derivation one may consult McClelland and Rumelhart (1986).

CLASS PROJECTS

Because the course this module has been integrated into is a survey course in artificial intelligence, students were expected to submit a final project chosen from a number of proposed ones that covered various areas of artificial intelligence. The projects relating to neural networks included papers which investigated certain neural network models that were not given in-depth coverage in class. Such models include competitive learning models (Rumelhart and Zisper, 1985), multilayer networks with backpropagation (Rumelhart, Hinton and Williams, 1988), Grossberg's Adaptive Resonance Theory model (Carpenter and Grossberg, 1988) and the Boltzmann machine (McClelland and Rumelhart, 1986). Another project consisted of writing a program to simulate one of the neural network models that were discussed in class. Such programs could be written on a personal computer.

Another project that was given involves the use of a neural network simulation software such as the package provided by McClelland and Rumelhart (1988) which runs on PC-compatible computers. The purpose of this project was to expose the students to the limitations of the Hebb rule and how such limitations can be overcome by using the delta rule. The project is presented in the next section.

A SAMPLE PROJECT

In this project students used the pattern associator simulation provided in the McClelland/Rumelhart (1988) simulation package to develop and test a neural network that uses the Hebb rule to learn a given input/output association. Students were given the set of input/output training patterns and were expected to train the network using a linear activation function. They were then asked to study the effects of factors such as: learning rate, straight versus permuted training (randomly selecting the order) and the number of training iterations. Students were then expected to come up with a combination that provides reasonable performance and use this combination for training the network.

After training the network, students were asked to test the network's performance using both the training patterns and some test patterns provided to them. These test patterns allowed the students to study the network's ability to generalize what it learned about the training patterns to other similar patterns. They then had to specify what was favorable about the network's performance and what the network's deficiencies were.

Finally, students had to make suggestions on ways to improve the network's performance. These could include using a different model architecture or a different learning rule with an explanation of their choices.

CONCLUSION

This paper presented a neural network module that the author has incorporated into an introductory artificial intelligence course. A sample project was included as well as suggestions for other projects.

This module was favorably received by the students as was evidenced by teaching evaluation forms. As a result of this course we had requests by several students for independent study in neural networks. We are currently considering offering a senior level course in neural networks.

REFERENCES

1. Carpenter, G.A. and Grossberg, S. 1988. The ART of Adaptive Pattern Recognition by a Self-Organizing Neural Network. *IEEE Computer*, 21.
2. Hopfield, J.J. 1982. Neural Networks and Physical Systems With Emergent Collective Computational Abilities. *Proceedings of the National Academy of Sciences*, 79.
3. McClelland, J. and Rumelhart, D. and the PDP Research Group. 1986. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*. MIT Press.
4. McClelland, J. and Rumelhart, D. 1988. *Explorations in Parallel Distributed Processing*. MIT Press.

5. McCulloch, W.S. and Pitts, W.H. 1943. A Logical Calculus of the Ideas Immanent in Nervous Activities. *Bulletin of the Mathematical Biophysics* 5.
6. Rumelhart, D., Hinton, G. and Williams, R. 1988. Learning Internal Representations by Error Propagation in *Neurocomputing*. Anderson, J. and Rosenfeld, E. (eds.). MIT Press.
7. Rumelhart, D. and Zisper, D. 1985. Feature Discovery by Competitive Learning. *Cognitive Science*, 9, pp. 75-112.