1991

Implementing a neural network in the Smalltalk graphical environment

John M. Damgaard
University of Northern Iowa

Implementing a Neural Network in the Smalltalk Graphical Environment

by

John M. Damgaard

University of Northern Iowa

Department of Mathematics and Computer Science

Spring 1991

Interest in artificial neural netwrorks has grown rapidly over the past few years. The technology is intriguing and the potential applications of the technology are exciting and diverse. Another area of growing interest in the computer science field is that of object-oriented programming. The object-oriented paradigm is a very powerful tool which can improve software quality and streamline the development process. The most common object-oriented language is Smalltalk. I feel that Smalltalk is an excellent platform on which to implement artificial neural networks.

Artificial Neural Networks

There are many reasons for interest in neural netwroks. Perhaps the most exciting characteristic of neural networks is their ability to learn. Thus, their development was fostered in the huge field of artificial intelligence.

Artificial neural networks are biologtically inspired. They consist of layers of neurons which loosely model their biological counterparts. Figure 1 shows a biological neuron.

The biological neuron consists of several key components:

axon - Fixed path by which neuron communicates to other neurons. It serves to conduct impulses away from the cell body.

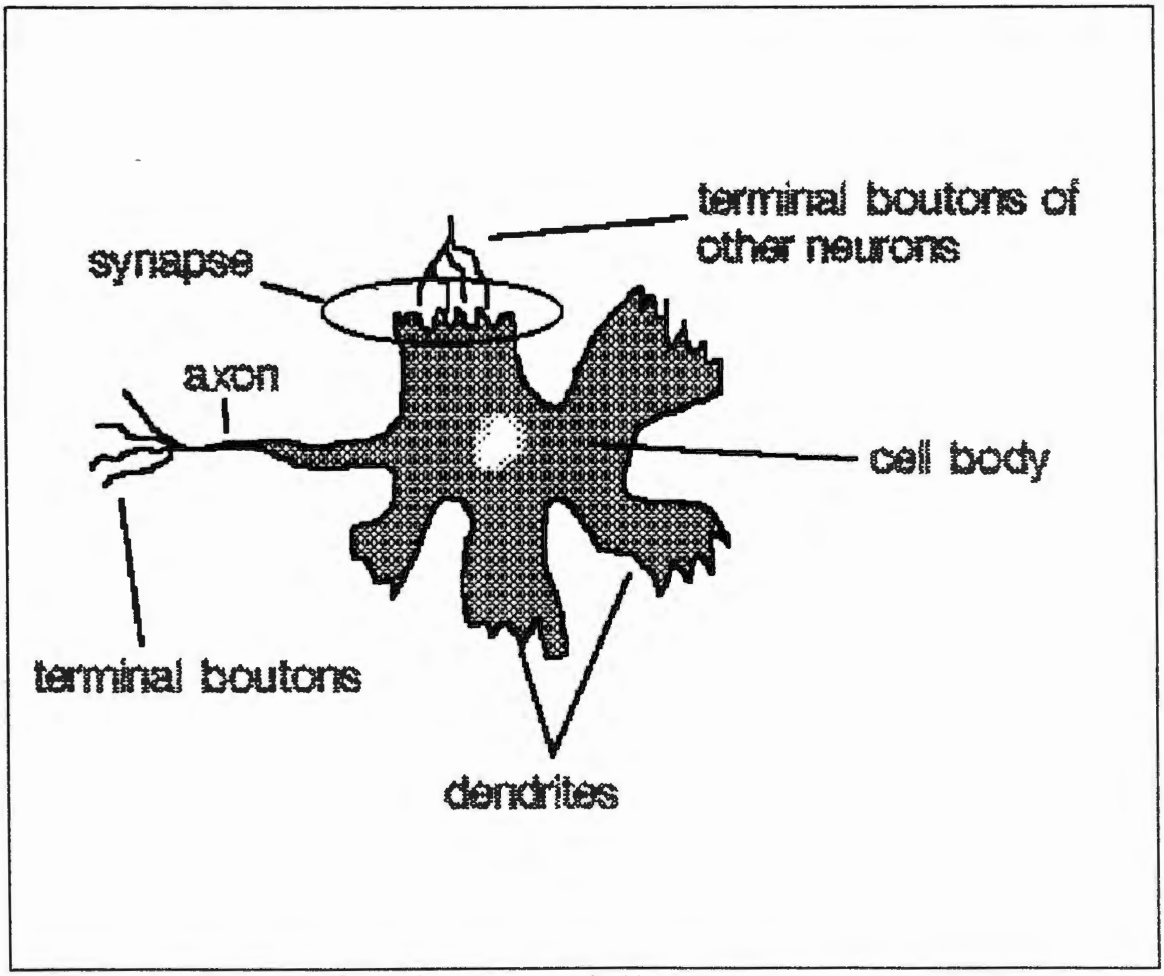

Figure 1. Biological Neuron

terminal bouton - A form of transmitting antenna which almost makes contact with the dendrites of other neurons at connection sights known as synapses.

dendrite - A form of receiving antenna which receives information from other neurons and conducts it toward the cell body.

Communication between neurons is performed when the terminal boutons release chemicals called neurotransmitters which pass through a synapse to another neuron's dendrite. If enough neurotransmitters are received into the cell body, the cell "fires". That is, an impulse is conducted down the axon and the terminal boutons release neurotransmitters.

It is believed that permanent memories are encoded by changes in the synaptic connections among neurons. During learning, synaptic connections change either by an increase in the release of neurotransmitters or an icrease in the sensitivity of the dendritic receptors.

An artificial neuron loosely models a biological neuron. It is a gross approximation at best. Howver, this crude representation has produced impressive results. Figure 2 shows an artificial neuron.

The artificial neuron consists of:

input vector(x[1]..x[n]) - These inputs model the synaptic connections of the biological neruon.

weight vector(w[1]..w[n]) - These weights model the "strength" of the synaptic connections.

activation function(F) - Also known as the "squashing" function, this function determines the neurons output based

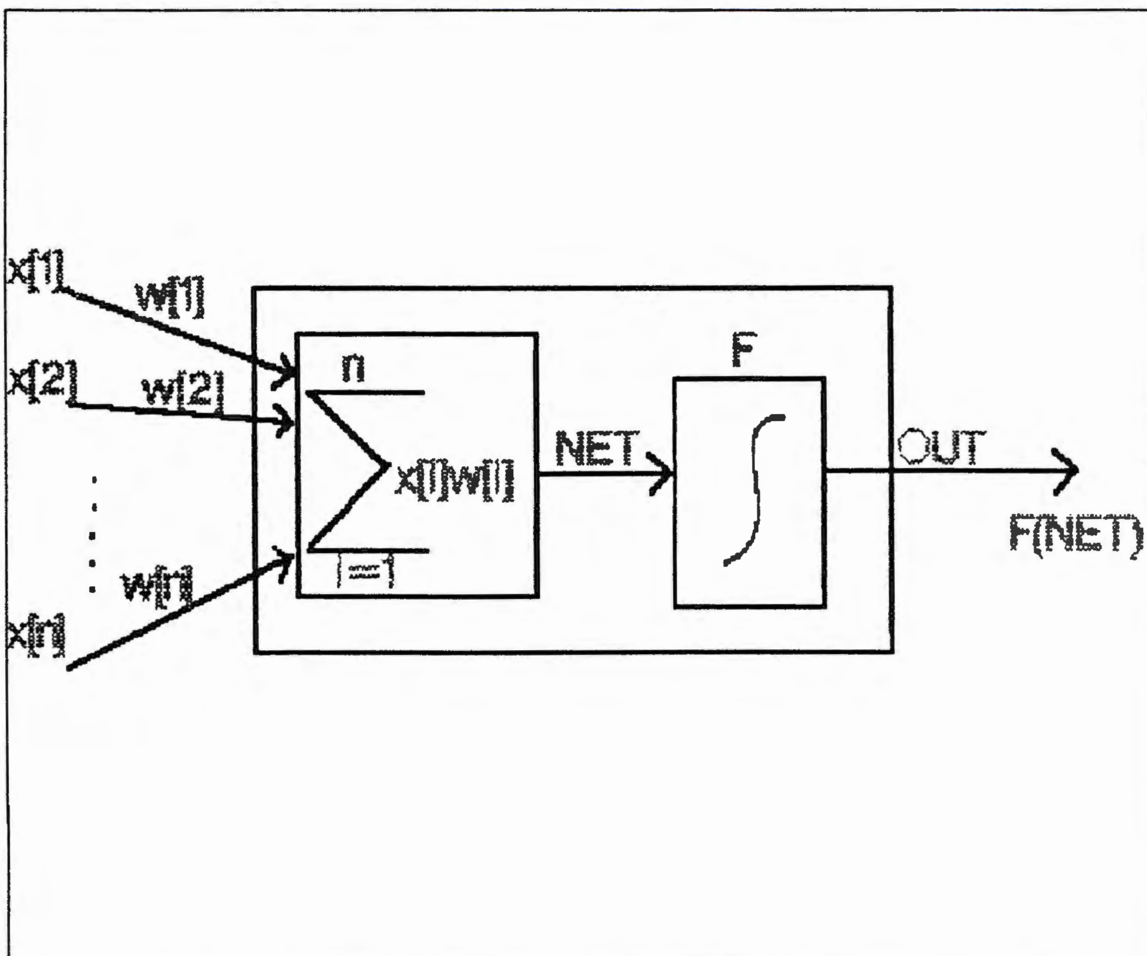

Figure 2. Biological Neuron

on the summation of the n input/weight products. The output is often either binary (0 or 1) or a small real value.

As with its biological counterpart, learning is achieved by changing the synaptic connections (i.e. weights) among neurons.

A single neuron by itself is of little use. The power of the neuron is tapped when multiple layers of neruron are connected to form a network. The number of layers in a network can vary, however networks with three or more layers of neurons are common because their classification capabilities are limited only by the number of neurons and weights in the layers. Figure 3 shows a simple three-layer network with two neurons in each layer. There are many different types of neural networks, each best suited to a specific kind of problem. Some of the common types are backpropagation (seen in Figure 3), counterpropagation, Hopfield, and Hamming.

Networks must be trained. Or in other words, the network's weights must be set. Training is accomplished by sequentially applying input vectors, while adjusting network weights according to a predetermined procedure. The backpropagation network uses one such procedure. This procedure assumes that each input vector is paired with a target vector representing the desired output; together these are called a training pair. A group of training pairs is known as a training set. Backpropagation requires the following steps:

1. Initialize all weights to small random numbers.

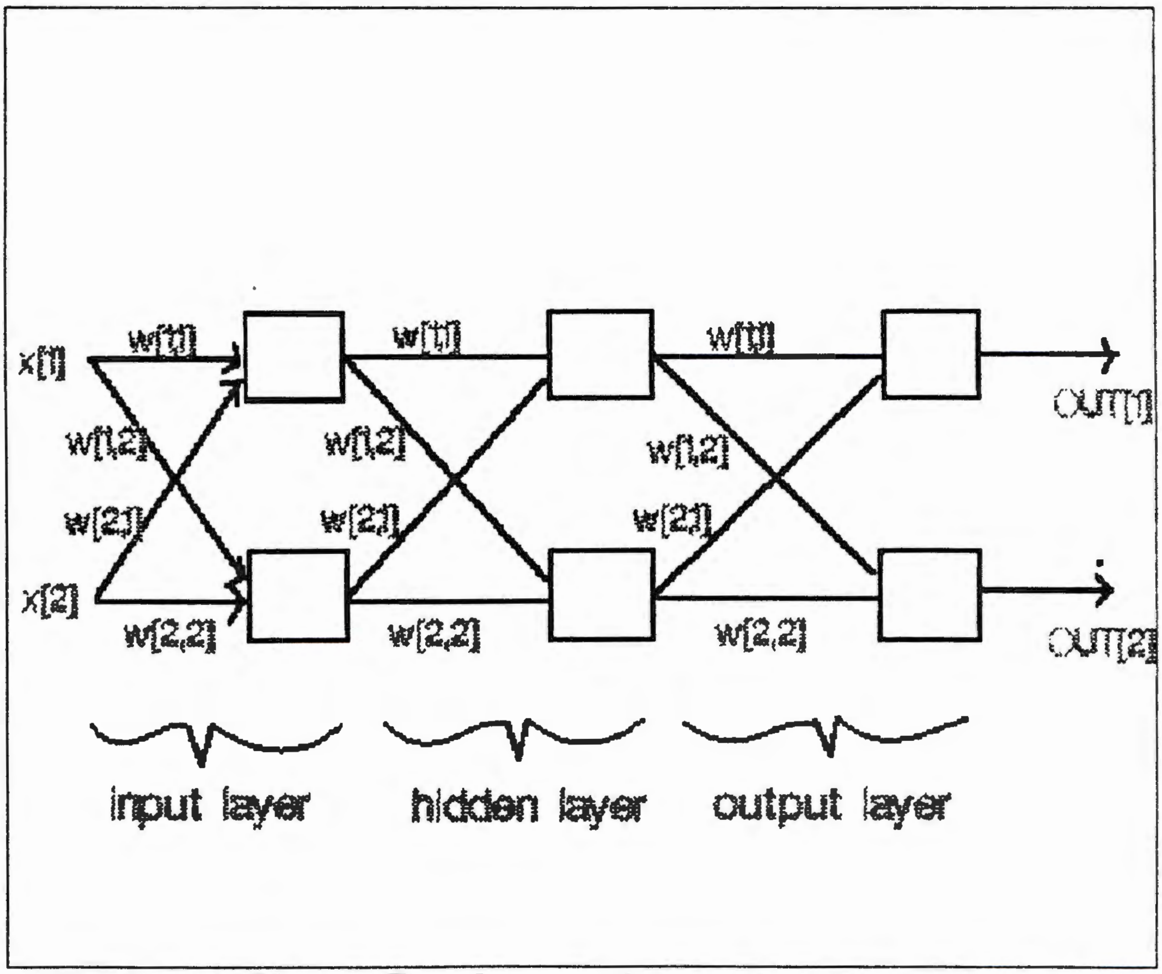

Figure 3. 3-Layer Backpropagation Network

2. Select the next training pair from the training set. Apply the input vector to the network input.

3. Calculate the output of the network.

4. Calculate the error between the network output and the desired output.

5. Adjust the weights of the network in a way that minimizes the error.

6. Repeat steps 2 through 5 for each vector in the training set until the error for the entire set is acceptably low.

Steps 2 and 3 constitute what is known as a "forward pass". That is, the signal propagates from the network input to its output. Steps 4 and 5 are known as the "reverse pass". That is, the error signal propagates backward through the network where it is used to adjust weights. The weights themselves are adjusted according a rule known as the Delta Rule. The Delta Rule is a function of the network error and a learning rate coefficient. Its specifics are beyond the scope of this paper.

Once a network is trained, the weights can be saved and the network will not have to be retrained. This is advantageous since the training process can be very lengthy depending on the size of the training pairs and the training set.

Applications of trained neural networks are numerous and diverse. Some of the more prominent ones are:

- handwritten character recognition

- conversion of printed text into speech

- data compression
- expert systems

I find the data compression application to be very impressive indeed. It has yielded data compression ratios of 10:1 to 100:1 with an acceptable level of data distortion. It is likely to be used in the voice/image telephones of the future.

Many problems unsolvable by other means can be solved by neural networks. One example is the infamous Traveling Salesman Problem for a large number of cities. As the power of neural networks continues to be revealed, they will be applied to more and more problems.

Object-Oriented Programming in Smalltalk

Smalltalk is the leading object-oriented language. However, Smalltalk is more than just a programming language; it is an integrated development environment.

Smalltalk employs a Graphical User Interface (GUI). It makes extensive use of windowing, dialogue boxes, and pop-up menus. Its primary input device is the mouse. Although it has been around for well over a decade, the rest of the computer world is just now beginning large scale adoption of its features, most notably its GUI.

Smalltalk is based on the object-oriented paradigm which allows you to model systems in terms that match human thinking and language, in terms of objects and actions on objects. This makes it a very powerful development environment indeed.

The methodology for using Smalltalk consists of:

- Identifying the objects appearing in the problem and its solution.

- Classifying the objects according to their similarities and differences.

- Designing messages which make up the language of interaction among the objects.

- Implementing methods which are the algorithms that carry out the interaction among objects.

Objects are protected data structures consisting of data and message definitions. The data stored inside of an object is accessible only through messages. This data encapsulation property is very desirable in the development of software.

Every object is an instance of some class. All objects which are instances of a class are similar because they have the same structure, the same messages to which they respond, and the same available methods. Classes form a hierarchy, consisting of a root class, Object, and many subclasses. Each class inherits the functionality of all its superclasses in the hierarchy.

All processing in a Smalltalk system involves sending messages to objects. Methods are the alorithms which are performed by an object in response to receiving a message. Methods represent the internal details of the implementation of an object.

Smalltalk's object-oriented nature makes it an exellent pltform on which to implement neural networks. Neurons can easily be modeled as objects and the entire network can also be considered as a single object for easy integration into large applications. The training set, too, can be considered as an object consisting of training pair objects.

The particular implementation of a network described here began as a public domain program for the Apple Macintosh Smalltalk environment. Many changes were necessary in porting the program to the IBM Smalltalk environment due to the program's many Macintosh-specific aspects.

The program is actually a graphical view on the training of a three-layer backpropagation network. It is invoked by creating a new instance of the DeltaANS class and sending it the message "open:" with the training set as an argument. This is achieved by evaluating the following Smalltalk expression:

```
(DeltaANS new) open: ((inputvector1 targetvector1)
                                          .
                                          .
                                          .
                      (inputvectorN targetvectorN))
```

It can also be invoked by sending the following messages which invoke built-in exemplars:

```
DeltaANS smallExample
DeltaANS largeExample
```

When the program is invoked, the following window is presented:

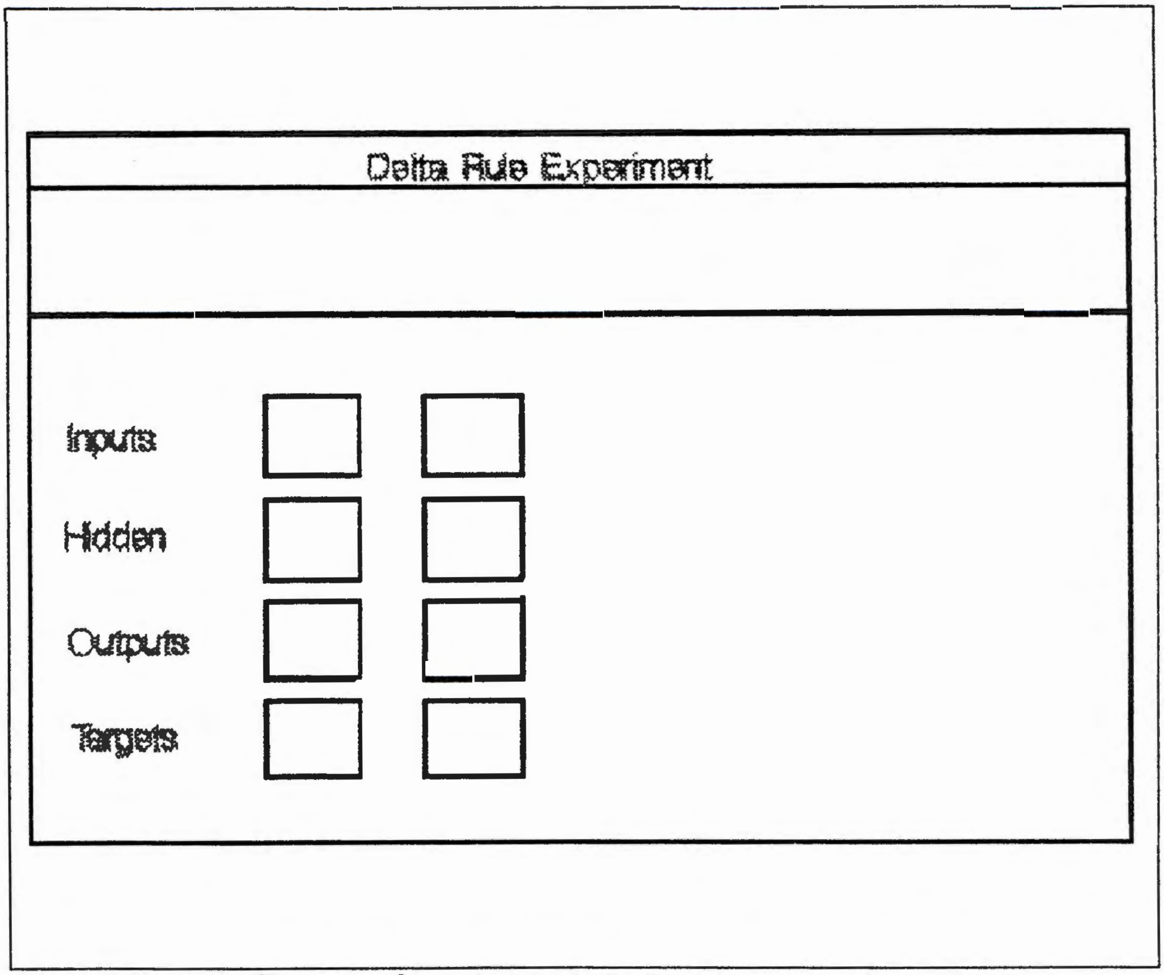

Figure 4. DeltaANS Program Window

The top pane of the window is simply a text pane for evaluating Smalltalk expressions. The bottom pane is a graphical pane, with four rows of blocks labeled Input, Hidden, Outputs, Targets. The number of blocks per row is determined by the size of the input vectors. The network is designed so that neuron output is in the interval [-1.0, 1.0].

if a neuron's output is in	its block will appear
[-1.0, -0.34]	white (not at all)
[-0.33, 0.33]	gray
[0.34, 1.0]	block

Clicking on the title bar with the right mouse button yeilds the following pop-up menu:

```
|learn
|learn 10 cycles
|learn 50 cycles
|learn 1000 cycles
|closeWindow
```

The menu options are selected by clicking on the desired option with the left mouse button. Selecting "learn" shows the effect of one forward/reverse pass cycle on the network. The other "learn" options simply repeat the learn cycle the specified number of times. "closeWindow" simply shuts down the program and removes its window from the display.

As the network becomes trained, the error between the network output and the target vector for each training pair will be reduced. Thus, the color of the blocks in the repective rows should become identical as the network converges.

Summary

Artificial neural networks and object-oriented programming are two of the hottest areas in the computer science field today. An object-oriented environment is an extremely powerful development tool and is arguably the best platform on which to tap the power of the neural network. I believe that as the two technologies mature, it will become more and more common for them to be jointly employed to solve complex problems.