

**Masters in Electronic Engineering Thesis**

**Internet-based Computer-Aided  
Learning for Artificial Neural Networks**

**Author:** Paul J Keeling, BEng

**Date:** February 1998



**Dublin City University  
School of Electronic Engineering**

**Supervised by Dr. John Ringwood**

# Acknowledgements

I would like to thank Dr John Ringwood for his continual support and guidance throughout the term of the project

I also wish to thank the academic staff of DCU for their consistent help and interest

I wish to thank Gareth Galvin for his support and interest in the project throughout the year

Finally, I would like to acknowledge my family and friends for their persistant

## Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Masters in Electronic Engineering is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work

Signed: Paul Keeling

ID No.: 969703160

Data: 12-Feb-98

# Table of Contents

|   |             |
|---|-------------|
| <b>Acknowledgements</b>                             | <b>11</b>   |
| <b>Declaration</b>                                  | <b>11</b>   |
| <b>Table of Contents</b>                            | <b>111</b>  |
| <b>Abstract</b>                                     | <b>viii</b> |
| <b>Chapter 1: General Introduction</b>              | <b>1</b>    |
| 1 1 Introduction                                    | 1           |
| 1 2 Project Background                              | 2           |
| 1 3 Computer Aided Learning                         | 3           |
| 1 3 1 CAL via the Internet                          | 3           |
| 1 3 2 Artificial Neural Networks                    | 3           |
| 1 4 Thesis Structure                                | 4           |
| <b>Chapter 2: Educational Pedagogics</b>            | <b>7</b>    |
| 2 1 Introduction                                    | 7           |
| 2 2 The Computer Aided Learning Paradigm            | 7           |
| 2 2 1 Introduction to CAL                           | 7           |
| 2 2 2 Interactivity                                 | 8           |
| 2 2 3 The Environment                               | 9           |
| 2 2 4 Application of CAL                            | 10          |
| 2 3 CAL and the Internet                            | 11          |
| 2 3 1 Introduction                                  | 11          |
| 2 3 2 The Internet as a Learning Environment        | 11          |
| 2 3 3 Distance Learning and Classroom Teaching Aids | 12          |
| 2 3 4 Summary                                       | 13          |
| 2 4 Student Motivation                              | 14          |
| 2 4 1 Introduction                                  | 14          |
| 2 4 2 The Importance of Motivation for Learning     | 14          |
| 2 4 3 Motivational Approaches                       | 15          |
| 2 4 4 Adding Motivators to Learning                 | 17          |
| 2 4 5 Summary                                       | 19          |
| 2 5 Teaching Methodologies                          | 19          |
| 2 5 1 Introduction                                  | 19          |
| 2 5 2 A Proposed Structure                          | 19          |

|   |           |
|---|-----------|
| 2 5 3 Implications  | 21        |
| 2 5 4 Summary   | 22        |
| 2 6 Conclusions   | 23        |
| <b>Chapter 3: The CAL Environment</b>                                     | <b>24</b> |
| 3 1 Introduction  | 24        |
| 3 1 1 Requirements  | 24        |
| 3 1 2 Recommendations   | 25        |
| 3 2 The Internet  | 25        |
| 3 2 1 The Internet Platform   | 25        |
| 3 2 2 World Wide Web  | 26        |
| 3 2 2 1 Internet Browsers   | 27        |
| 3 2 2 2 The Hypertext Mark-up Language                                    | 28        |
| 3 3 The JAVA Programming Language   | 30        |
| 3 3 1 History of JAVA   | 30        |
| 3 3 2 JAVA Language Definition  | 31        |
| 3 3 2 1 Java is Simple  | 32        |
| 3 3 2 2 Java is Object-Oriented   | 32        |
| 3 3 2 3 Java is Distributed   | 32        |
| 3 3 2 4 is Interpreted (and Compiled)                                     | 32        |
| 3 3 2 5 Java is Robust  | 33        |
| 3 3 2 6 Java is Secure  | 33        |
| 3 3 2 7 Java is Architecture Neutral                                      | 34        |
| 3 3 2 8 Java is Portable  | 34        |
| 3 3 2 9 Java is High Performance  | 35        |
| 3 3 2 10 Java is Multithreaded  | 35        |
| 3 3 2 11 Java is Dynamic  | 36        |
| 3 3 3 The JAVA Environment  | 36        |
| 3 3 4 Internet Support  | 37        |
| 3 4 JavaScript  | 37        |
| 3 5 The Computational Engine  | 39        |
| 3 5 1 Introduction  | 39        |
| 3 5 2 MATLAB  | 39        |
| 3 5 2 1 MATLAB Communications   | 40        |
| 3 5 2 1 1 Success to Date   | 40        |
| 3 5 2 1 2 The VCLab Approach  | 41        |
| 3 5 2 1 3 Summary   | 42        |
| 3 5 3 JAVA as a Computational Engine                                      | 43        |
| 3 6 Conclusions   | 44        |
| <b>Chapter 4: A Prototype Course in Artificial Neural Networks (ANNs)</b> | <b>45</b> |
| 4 1 Introduction  | 45        |
| 4 2 The Course Structure  | 45        |

|  |           |
|--|-----------|
| 4 2 1 Lectures   | 45        |
| 4 2 2 The ANNs Web Site                                  | 46        |
| 4 2 3 MATLAB Examples and Exercises                      | 47        |
| 4 2 4 Java Demonstrations                                | 47        |
| 4 2 5 Java In-Class Demonstrations                       | 48        |
| 4 3 Artificial Neural Networks                           | 49        |
| 4 3 1 Introduction to ANNs                               | 49        |
| 4 3 2 Simple Neural Networks                             | 53        |
| 4 3 2 1 McCulloch-Pitts Neuron                           | 53        |
| 4 3 2 2 Hebbian Learning                                 | 53        |
| 4 3 2 3 Single Layer Perceptron Learning                 | 54        |
| 4 3 3 Multi-Layer Perceptron Networks                    | 56        |
| 4 3 3 1 Introduction                                     | 56        |
| 4 3 3 2 Functional Capabilities of the MLP Network       | 56        |
| 4 3 3 3 The Back-Propagation Training Algorithm          | 57        |
| 4 3 4 Radial Basis Function Networks                     | 59        |
| 4 3 4 1 Introduction                                     | 59        |
| 4 3 4 2 Functional Capabilities of the RBF Network       | 61        |
| 4 3 4 3 Training Algorithm for the Interpolation Problem | 62        |
| 4 3 5 Hopfield Networks                                  | 66        |
| 4 3 5 1 Introduction                                     | 66        |
| 4 3 5 2 Functional Capabilities of the Hopfield Network  | 68        |
| 4 3 5 3 Operational Features of the Hopfield Network     | 68        |
| 4 3 6 Kohonen Self-Organizing Feature Map (SOFM)         | 71        |
| 4 3 6 1 Introduction                                     | 71        |
| 4 3 6 2 The SOFM Algorithm                               | 72        |
| 4 3 6 3 Properties of the SOFM Algorithm                 | 75        |
| 4 4 Conclusions  | 77        |
| <b>Chapter 5: ANN Java Demonstrations Development</b>    | <b>78</b> |
| 5 1 Introduction   | 78        |
| 5 2 ANN Web Site Development                             | 78        |
| 5 2 1 Text Converters                                    | 78        |
| 5 2 2 HTML Editor  | 79        |
| 5 3 Generic Tools  | 79        |
| 5 3 1 The Matrix Class                                   | 79        |
| 5 3 1 1 Initialisation                                   | 80        |
| 5 3 1 2 Functionality                                    | 80        |
| 5 3 2 The Graph Class                                    | 80        |
| 5 3 2 1 Functionality                                    | 81        |
| 5 3 2 2 User Controls                                    | 83        |
| 5 3 3 The Neuron Class                                   | 83        |
| 5.3.3.1 Initialisation                                   | 84        |
| 5 3 3 2 Functionality                                    | 84        |

|  |            |
|--|------------|
| 5 4 The ANN Demonstrations                                     | 85         |
| 5 4 1 The Hebbian Learning Demonstration                       | 85         |
| 5 4 1 1 Objective  | 85         |
| 5 4 1 2 Hebbian Learning Demonstration Operation               | 86         |
| 5 4 2 The Perceptron Learning Demonstration                    | 86         |
| 5 4 2 1 Introduction   | 86         |
| 5 4 2 2 Perceptron Learning Demonstration Operation            | 87         |
| 5 4 3 The Multi-Layered Perceptron Demonstrations              | 87         |
| 5 4 3 1 Introduction   | 87         |
| 5 4 3 2 General Operation                                      | 88         |
| 5 4 3 3 Logic Function Approximation Demonstration             | 89         |
| 5 4 3 4 MLP Polynomial Function Approximation<br>Demonstration | 90         |
| 5 4 4 Radial Basis Function Network Demonstration              | 93         |
| 5 4 4 1 RBF Network Demonstration Operation                    | 94         |
| 5 4 5 Hopfield Network Demonstration                           | 98         |
| 5 4 5 1 Hopfield Network Demonstration Operation               | 98         |
| 5 4 5 1 1 Pattern Storage                                      | 99         |
| 5 4 5 1 2 Network Testing                                      | 99         |
| 5 4 6 Kohonen Self-Organizing Feature Map Demonstration        | 101        |
| 5 4 6 1 Kohonen SOFM Demonstration Operation                   | 102        |
| 5 5 Conclusions  | 103        |
| <b>Chapter 6: Evaluation of ANNs Courseware</b>                | <b>104</b> |
| 6 1 Introduction   | 104        |
| 6 2 Development of the Questionnaire                           | 104        |
| 6 2 1 The Purpose of a Questionnaire                           | 104        |
| 6 2 2 Questionnaire Structures                                 | 105        |
| 6 2 3 Types of Questions                                       | 106        |
| 6 3 Structure of the ANN Questionnaire                         | 108        |
| 6 3 1 ANN Questionnaire Introduction                           | 108        |
| 6 3 2 The Questions  | 108        |
| 6 4 Analysis of the ANN Questionnaire                          | 110        |
| 6 5 Conclusions  | 112        |
| <b>Chapter 7: Conclusions</b>                                  | <b>114</b> |
| 7 1 Introduction   | 114        |
| 7 2 The Internet CAL Paradigm                                  | 114        |
| 7 3 Artificial Neural Networks via Java Demonstrations         | 114        |
| 7 4 Future Work and Recommendations                            | 115        |
| <b>Bibliography</b>  | <b>116</b> |

|                   |            |
|-------------------|------------|
| <b>Appendix A</b> | <b>120</b> |
| <b>Appendix B</b> | <b>121</b> |
| <b>Appendix C</b> | <b>122</b> |
| <b>Appendix D</b> | <b>123</b> |

## **Abstract**

# **Internet-based Computer-Aided Learning for Artificial Neural Networks**

This thesis presents research performed to investigate the potential offered by the Internet for the implementation of an Engineering Computer-Aided Learning (CAL) environment. The research comprises two categories, a detailed literature survey of CAL and its application through the medium of the Internet environment.

As a direct result of the literature survey, the scope of CAL can be considered to comprise the use of text, graphics, animations and sound. It is through the use of the CAL media that the true power of computer-aided education can be realized. The research performed focuses on student motivation, with emphasis placed on the educational environment.

The Internet as a CAL environment was chosen for evaluation in this thesis due to its ability to convey information in a variety of contexts to any student with Internet access. Courseware was developed for the M Eng (Masters in Engineering) program, specifically in the field of Artificial Neural Networks (ANNs). ANNs lend themselves to CAL due to their mathematical, graphical and exploratory nature.

By considering the courseware developed the Internet is evaluated as an effective CAL medium through feedback from students taking the M Eng course and from other interested parties. The thesis then concludes with suggestions for further development of CAL courseware on the Internet.



# Chapter 1: General Introduction

## 1.1 Introduction

This project investigates the potential offered by the Internet for the implementation of an Engineering Computer-Aided Learning (CAL) environment. It is an example of how the Internet can be used as an interactive, multimedia aid for education. The courseware was developed for the MEng (Masters in Engineering) program, specifically in the field of Artificial Neural Networks (ANNs). However, it is the opinion of the author that the scope of project embodies all aspects of computer aided learning.

The research conducted presents some of the practical aspects of CAL education, but more precisely those concerned with Internet CAL. The Internet is then shown to be a well-suited environment in which to realize CAL courseware, especially with regard to the ANNs courseware. The practical application is hereby divided into four sections:

- the design of the course structure and the formulation of the ANNs courseware,
- the development of the ANNs World Wide Web site,
- a detailed discussion of the development of the ANNs course demonstrations, and
- the appraisal of the CAL paradigm with regard to the ANNs courseware produced.

## **1.2 Project Background**

This project is part of an Integrated BEng/MEng program that commenced during the Industrial Training (INTRA) six month period preceding fourth year. The program then continued as a fourth year BEng project. This MEng project is a natural progression from the work carried out during the BEng period.

The fourth year project investigated “A Computer Aided Learning Package to Teach System Dynamics and Control” [Keeling]. The package, SYSCAL, was developed upon the Microsoft Windows platform with the aid of the Borland C++ programming language. MATLAB was chosen as the appropriate computational engine to perform the complex mathematical computations and graphing procedures necessary to teach System Dynamics and Control [MATLAB]. However, considerable difficulties arose in obtaining a stable communication link between the SYSCAL package and MATLAB.

The research carried out during the BEng period found the Internet to be a suitable platform on which to develop a CAL package. However, the advantages discussed were nevertheless superseded by the problem of developing an effective communication link with the computational engine. As a result, the Borland C++ was adopted for the development of an entirely new CAL environment.

The environment incorporated an extensive system of typography, including symbolic characters and interactive hot spots. In addition, the environment was structured in a modular and extensible fashion thus facilitating the expeditious creation of courseware. Internet browsers are capable of achieving a high level of interactivity and can display a myriad of font styles, colours and sizes. However, at the time of the MEng Project's inception, it was not deemed possible to successfully incorporate a computational engine with an Internet browser.

## **1.3 Computer Aided Learning**

### **1.3.1 CAL via the Internet**

The Internet has generated a considerable amount of interest of late, it is a phenomenon with the potential to rival television as a learning medium. One contributing factor in this explosion of interest must be the development of the World Wide Web (WWW or simply the Web) and Web browsers such as Netscape [Netscape]. The Web is not the Internet, it is a structure on top that is part data and part navigational tool [Thackray].

The Web is a distributed, hypermedia information retrieval system with multimedia capabilities that include scalable fonts, graphics, audio, and video clips. The Web also supports the dissemination of Java Applets, which are high level programming language graphical inserts. The Web is therefore highly interactive and thus provides a self-learning environment that facilitates the effective learning of complex concepts. As a result, it may be concluded that the Internet is very exciting and potentially very powerful learning tool.

### **1.3.2 Artificial Neural Networks (ANNs)**

The year 1943 is often considered the initial year in the development of artificial neural systems. McCulloch and Pitts [McCulloch et al.] outlined the first formal model of an elementary computing neuron. The model included all necessary elements to perform logic operations, and thus it could function as an arithmetic-logic computing element. However, the implementation of its compact electronic model was not technologically feasible during the era of bulky vacuum tubes [Zurada].

Considerable developments in ANNs have been made since its inception. It is perhaps no coincidence that a renewed enthusiasm in the field of ANNs has occurred in an era of such explosive technological advances. The micro-processors of today's conventional programmable computers are capable of simulating neural networks at speeds comparable to that of biological systems [Patterson].

Artificial Neural Networks are simplified models of the central nervous system. They are networks of highly interconnected neural computing elements that have the ability to respond to input stimuli and to learn to adapt to the environment [Patterson]. ANNs have been shown to be effective as computational processors for various tasks including pattern recognition (e.g., speech and visual image recognition) [Tou et al.], classification, modeling and forecasting, combinatorial problem solving [Hush et al.], data compression [Kohonen], associative recall and multisensor data fusion and noise filtering [Hopfield].

As previously suggested, ANNs require a sizable level of computing power in order to perform the repetitive computations associated with their training and testing. In order to effectively convey the difficult concepts associated with the field of ANNs, it is necessary to supply a sufficient amount of graphical and exploratory learning material. Therefore, like many engineering disciplines, the teaching of ANNs is especially suited to CAL.

Due to the increasing use and versatility of ANNs, their incorporation into the Masters curriculum is considered an extremely worthwhile endeavor. The use of an Internet-based CAL educational tool provides easy access to a self-paced, tiered learning structure, which effectively delivers the ANNs courseware. The work described in this thesis is therefore of a practical nature and is considerably relevant to the fields of CAL and ANNs.

## **1.4 Thesis Structure**

This thesis consists of seven Chapters all of which, excluding this general introductory Chapter 1, are detailed in the following description:

- Chapter 2 - "Educational Pedagogics", demonstrates how the Internet could be employed as an effective CAL environment. This is achieved by first defining some of the essential components required by the CAL environment followed by a discussion of their implementation by the Internet. Internet CAL is then shown to be capable of increasing motivation levels and thereby improving the students'

learning A discussion of an effective structure through which to communicate the courseware is also provided

- Chapter 3 - “The CAL Environment”, provides a detailed description of the Internet CAL environment The Chapter begins by outlining the general components required by a proposed CAL environment as suggested by Chapter 2 This is followed by a discussion of the Internet with regard to the courseware development This discussion provides an insight into the technical details of coagulating all the facets of the CAL environment In addition, the reasoning behind the use of Java as a computational engine is provided The Chapter is concluded by a description of the proposed CAL environment with attention placed on its impending development procedure
- Chapter 4 - “A Prototype Course in Artificial Neural Networks”, conveys some of the practical considerations with regard to presenting the ANN courseware to an MEng student body An overview of the ANN course material is described in an attempt to clarify some of the development procedures of the demonstrations discussed within Chapter 5 This overview also justifies the need for demonstrations by showing that the course material requires graphical representations and repetitive calculations in order to enhance the students’ understanding and learning experience In addition, the Chapter is used to discuss some of the practical issues concerning the development of the ANN Web site
- Chapter 5 - “ANN Java Demonstrations Development”, discusses the practical considerations involved in the creation of the Java demonstrations or Applets The Chapter introduces some of the generic tools used throughout all the Applets, namely the Matrix Algebra, Single Neuron and the Graphing Tool The individual ANN demonstrations are as follows:

- Hebbian Learning Demonstration
- Perceptron Learning Demonstration
- Multi-Layered Perceptron Demonstrations
- Radial Basis Function Network Demonstration
- Hopfield Network Demonstration
- Kohonen Self Organizing Feature Map Demonstration

The objective and functionality of the individual ANN demonstrations are discussed followed by conclusions concerning the validity of the demonstrations

- Chapter 6 - “Evaluation of the ANNs Courseware”, presents the development of a questionnaire, which is used to evaluate the ANN courseware. The results obtained from the completed questionnaires are then used to analyse the effectiveness of the ANN courseware
- Chapter 7 - “Conclusions”, provides a set of recommendations concerning the future development of the courseware together with an analysis of Internet CAL on a whole

# Chapter 2: Educational Pedagogics

## 2.1 Introduction

The Engineering curriculum and the means for transmitting knowledge to students have not changed much during the past thirty years. Over the past three decades, the three primary modes of transmitting information to students are the textbook, the chalkboard (or the overhead projector), and the instructor's handouts [Lee et al]. It is clear there is a need to modernize expand upon the content and delivery of the engineering curriculum to prepare today's students for a more competitive working environment. Interactive multimedia presents a promising opportunity for enhancing the student's learning environment.

## 2.2 The Computer Aided Learning Paradigm

### 2.2.1 Introduction to CAL

Computer Aided Learning or CAL is defined by the use of computers to enhance the educational process. In the past, CAL consisted of little more than an electronic textbook. However, due to advances in modern technology, CAL has become synonymous with the integration of Multimedia, which can be defined as the combination of two or more media to present educational material. The media may consist of text, still or animated graphics, movie segments, sounds and music [Tolhurst].

*"From the Latin, 'multi', (many), and 'media', (ways of presenting information). The many ways you can present information on an audio-visual computer include text pictures (both photo and illustrations), sounds, music, animations (such as moving diagrams) and video clips. And that's it really. That's multimedia!" [Noonan]*

The motivation for CAL is to provide an educational environment that is conducive to improved learning. Such an environment seeks to increase students' attention and interest and thereby facilitate improved information retention. In addition, CAL attempts to encourage the student to learn at their own pace in a highly interactive, user-driven and exploratory environment [Harding et al.]

### **2.2.2 Interactivity**

The development of hypertext and hypermedia has been crucial in the move towards interactivity. Interactivity allows the user to navigate through information in many different ways. Using hypertext the user clicks on 'hot words' in the text with the mouse, and then related information such as an article or picture is shown on the screen [Stanley]. Hypertext allows the user to access additional information if required, thus reducing the drudgery of wading through previously encountered material. Hypertext has been defined by many people, however the following definition describes its functionality in its entirety

*"The term hypertext refers to computer-based texts that are read in a non-linear fashion and that are organised on multiple dimensions. The same material (which can be any kind of randomly accessible medium, e.g., text, video, audio) is capable of being explored in different ways, with different exploration paths producing what are essentially multiple texts for the same topic."* [Spiro]

Hypermedia is not strictly related to accessing information through textual 'hot spots', it is also defined in terms of graphical 'hot spots' such as pictures or graphs which when activated by the mouse can access any other kind of information medium or simply an altered representation of the current medium. The following definition describes the function of Hypermedia in the CAL environment



*"Hypermedia...shares many characteristics with other forms of computer based learning material, but has a node and link structure which enables the user to explore the content in a non-linear and interactive way, both quickly and easily." [Knussen]*

It is the interactivity provided by hypertext and hypermedia that makes CAL a well-suited educational medium because the student can easily explore their learning environment and navigate through information in a variety of ways. Students have the option to read the notes in the traditional linear fashion and then divert to relevant examples, exercises, etc. Alternatively, users may start with the exercise or examples and use the cross-referencing facilities of hypertext to look up any topics that need explaining.

### **2.2.3 The Environment**

An important aspect of producing a CAL package is to determine an appropriate environment through which the courseware may be effectively conveyed. As discussed in Section 2.2.1, the environment must be highly interactive, user-driven and exploratory. Ideally the environment should also be readily accessible by the students and independent of their computing platform. The aforementioned criteria may all be satisfied through the use of the Internet.

The Internet and the World Wide Web (WWW, W3 or the Web) provide a suitable visual environment on which to build a CAL package. It has the facilities to display text, graphics and Java language programs, in particular Applets as discussed in Section 3.2.2.4. In addition, the number of people with Internet access is increasing at a tremendous rate, especially within the college framework. The Internet is therefore promoting an increase in the number of distance learning courses.

In addition, it is more practicable to use the Internet platform on which to develop a CAL package, since the developing Web material is all supplied by a single Web server. As a result, the updating of CAL courseware requires changing only the host material.

### **2.2.4 Application of CAL**

CAL is especially suited to the field of Engineering, ranging from simulating dangerous environments, to replacing expensive laboratories and factories. The Engineering academic community currently has access to a wide range of material on the Internet, leading to an explosion of information that may be used by educational establishments for supporting their students, particularly with regard to current research, teaching methods and results. Engineering is a graphics-oriented discipline in which drawings, sketches and graphs have always played an important role, but required enormous manpower to produce. The text-only approach constrains the application domain, whereas application of graphical representations can enhance the understanding of complex concepts associated with many engineering disciplines [Lai]

Many calculations in engineering design are repetitive in nature. A set of principles or rules may need to be applied many times over the analysis of a problem [Lai]. The CAL environment can provide the computational means to eliminate the drudgery of tedious, repetitive calculations and thereby focus the students' attention on developing an insight into the educational material at hand. CAL is therefore encouraging an active investigatory approach to understanding the mathematical concepts and complements traditional teaching of how to calculate results.

A substantial number of engineering students have acquired a significant level of computer literacy and have access to computers, both within the college framework and at home. Therefore, it is quite advantageous to produce CAL courseware that contributes to the students' learning experience.

## **2.3 CAL and the Internet**

### **2.3.1 Introduction**

Through the use of Internet Web browsers, a plethora of information can be obtained that is highly interactive and multi-dimensional. The interactivity is supplied through various hypermedia and high level programming language inserts such as Java Applets. It is this interactivity that makes the Internet platform a good source of CAL material.

The Internet also encourages the exchange of information through the use of E-mail, newsgroups, bulletin boards, etc. The Internet is therefore supplying access to information, which is not readily accessible via traditional channels.

### **2.3.2 The Internet as a Learning Environment**

In the past two decades the Internet has set in place a novel paradigm which is now extending to methods of education. The learning environment provided by Internet browsers enhances the presentation of learning materials through the integration of multimedia and hypermedia. The additional media are used to increase students' interest and therefore motivation by providing them with an active role in their own education.

The Internet learning environment can be either supervised or unsupervised. The supervised approach to learning gives rise to the computer classroom, which is a natural evolution of the traditional classroom [Conway]. It consists of individual workstations for students, each viewing the required courseware on a local browser, and a projection of the lecturer's actions guiding their navigation through the instructional material. This teaching technique can involve several learning methods. The instructor may simply review parts of the chapter with the students following along, interacting by, e.g., varying parameters in the examples. The students can also work independently on the simpler exercises, with monitoring by the instructor [Harger]. Although this form of instruction takes advantage of advances in technology, it does however require a considerable hardware investment. The Internet is also oriented toward the unsupervised learning approach due to its support of hypermedia.

The integration of hypermedia into the learning environment allows learners to choose their own route through a course and adapt the material so that it uniquely fits their own learning styles

The Internet is inherently stand-alone in that all one needs to access the 'Information Super-Highway' is a computer with a network connection and a browser. No additional software need be obtained in order to take advantage of the abundance of Web pages, which are readily available. In addition, Internet access is becoming increasingly widespread, thus supplying the scope for reaching the student outside of the classroom. As a result, a practicable means of achieving effective distance learning programs exists.

### **2.3.3 Distance Learning and Classroom Teaching Aids**

Due to the nature of the Internet's accessibility, it lends itself to the development of distance educational programs. The Web may be used to distribute course material and assignments. The course material may be delivered in a multimedia environment with hypermedia providing a non-linear, self-paced structure. The student assignments (individual and group) and their respective grades may be communicated either through the Web or via E-mail facilities. The students may submit, along with standard text, coloured plots, software and audio files where appropriate [Orsak et al]. The Internet is therefore promoting an essentially paperless course for both students and faculty.

Traditional distance learning programs required a considerable amount of effort by the school in order to transport the learning material to the student. Many of the school's courses would be videotaped and mailed to the students. When the Web is used to promote course information and the students have Web access, the administrative overhead of teaching a video section is dramatically reduced [Flur et al]. Video-educated students, previously encumbered with large mailing delays, suddenly can become more active participants in the course.

Another way in which the administrative overhead of distance educators could be reduced is through cooperation. The Web and the Internet can provide an avenue for assistance and cooperation among universities, industry, and professional societies [Sears et al] One university could set up standard multimedia experiments for basic circuit analysis, which schools from across the world could use, and other universities could provide support for other courses Furthermore, the Web is frequently used to summarize current research projects, often including complete sets of related publications and abstracts [Flur et al] The Web is therefore paving the way for a global multimedia library

In addition to the educational benefits offered by the Internet, it also caters for more practical communications For instance, the Internet may be used to disseminate course descriptions, outlines, syllabi, grading scales, and even homework assignments The Internet is therefore reducing the level of outgoing mail and student inquiries

### **2.3.4 Summary**

The Internet offers exciting possibilities for learners, tutors or teachers, and learning providers The Internet's inherent benefits are a product of its expansive information system and its diverse communication protocols It is the Web protocol that offers the most obvious advantages in the educational field due to its ability to convey courseware via multimedia and hypermedia The Internet therefore offers potential help towards meeting some of the educational and resource challenges confronting us

## **2.4 Student Motivation**

### **2.4.1 Introduction**

Everyone knows intuitively that motivation is vital to learning, but few understand what it is or how to use it systematically [Reigeluth et al ] This is reflected in how little attention has been given to motivation in the educational technology literature In fact, motivation barely gets a mention in most books on learning theory, instructional design, or media [Spitzer (a)] It is the opinion of the author that motivation should be considered the central element in the instructional design procedure

One of the reasons why motivation has been the ‘neglected factor’ in instructional design is due to the fact that it is an internal state that cannot be readily observed or directly modified [Spitzer (a)] The degree of a student’s learning in any instructional program is solely dependent on their internal psychological state or motivation level When motivation is low, learning levels will be low That is why many of the most sophisticated and meticulously designed instructional systems often fail to achieve the desired results

### **2.4.2 The Importance of Motivation for Learning**

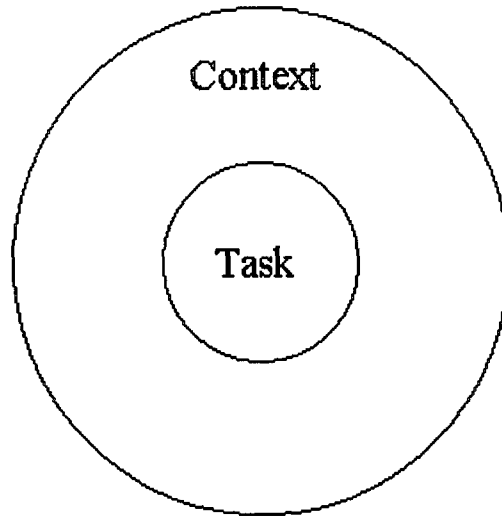
Traditional educational methods are not conducive to a self-motivating learning experience Conventional classroom instruction is synonymous with a passive assimilation of knowledge, the student has little or no control over the level of instruction or the rate at which it is conveyed. Therefore, a student who grasps a concept quickly is impeded, and conversely a slower student is forced to move on before properly understanding the information Moreover, an interested student is prevented from exploring a particular detail that stimulates their curiosity due to the linear, and perhaps limited structure of the courseware Although, the aforementioned is not always the case, it is however one of the inherent shortfalls of conventional instruction Motivation must therefore be considered an essential element in any effective instructional system

The key to motivating instruction resides in the courseware, rather than in the delivery system hardware [Spitzer (a)] It is generally accepted that if information is conveyed to students with a combination of text, colour, graphics, animation, sound, moving pictures, and a degree of interactivity, the attention of the student will be significantly increased, promoting learning interest and enhancing retention of knowledge [Lee et al ] Studies have shown that people retain 25% of what they hear, 45% of what they see and hear, and almost 70% when they actively participate in the process [Myers] It is therefore the content of the courseware and its delivery system that determines the effectiveness of the courseware in motivating a student

### **2.4.3 Motivational Approaches**

Motivation involves the release of human energy, and all human beings have enormous energy available The same people who may be lethargic in classrooms and workplaces are often dynamos of energy when they are doing the things they really want to do Motivating experiences cause stored energy to be released [Spitzer (a)]

In general, there are two kinds of motivation methods, intrinsic motivation approach and the extrinsic one [Matsubara et al.]. The intrinsic motivation method is focused on the student's interest and curiosity, and as a result, the student is inclined to study without external influence On the other hand, extrinsic motivation method is focused on the external target or outcome (e g , praise and reproach), and it is useful to motivate the student immediately The intrinsic motivation approach is difficult to utilize since it is dependent on the knowledge domain of the learner Therefore, the extrinsic motivation approach is often adopted, and thus provides the appropriate praise message or reproach message in the specific learning situation



**Figure 1.1: Importance of Context**

An alternative, 'SuperMotivation Approach' [Spitzer (b)] was developed by Dean Spitzer in an attempt to encapsulate the concepts of learning motivation. The research performed by Spitzer is not focused on the characteristics of people, but on the characteristics of highly motivating activities. He discovered that all activities are divided into two aspects: task and context. The task is the basic activity. The context is everything else [Spitzer (a)] (see Figure 1 )

It is interesting to note that most tasks are inherently boring, or will eventually become that way if they are repeated long enough. Think of even the activities that people enjoy the most. For example, how long could any person stay interested in basketball if all they did was dribble the ball, or just throw it back and forth? Unfortunately, most classroom instruction is conducted in a similarly repetitive and unmotivating manner. However, according to Spitzer, any activity can be made highly motivating if a motivating 'context' is added to the basic task [Spitzer (a)]. What makes an activity motivating is its context.



#### 2.4.4 Adding Motivators to Learning

As previously discussed, the context of a sport like basketball incorporates contextual characteristics, or motivators that release enormous positive energy in participants. The more motivators built into the context of an activity, the more interesting it will be. The basic thesis of SuperMotivation is that any activity can be made motivating if we build sufficient motivators into it [Spitzer (b)]. The motivators that transform a sport or game into an engaging and interesting activity can also transform the task of learning into a stimulating experience.

The following outlines the motivators that Spitzer suggested are inherent in any motivating experience,

- **Action:** Motivation is an active, not a passive state. Human beings are most highly motivated when they are actively engaged. Physical activity is one of the main reasons why participation in sports and games is so highly motivating. The most motivating learning experiences are almost always the most active ones. Therefore, the most effective instruction would actively involve the learner in the learning process, both physically and mentally [Spitzer (a)].
- **Fun:** Fun energizes people. It makes them more enthusiastic learners [Spitzer (a)].
- **Choice:** One of the most distinctive characteristics of human beings is their capacity and desire to make choices. To increase learner motivation the student should be able to choose the content, learning methods and instructional materials [Spitzer (a)].
- **Social Interaction:** Human beings have an intrinsic need for social interaction. A high level of interaction will increase motivation and also enhance learning. This is achieved via cooperative learning experiences, such as group discussions, peer tutoring, collaborative problem solving and decision making, etc [Spitzer (a)].

- **Error Tolerance:** Mistakes are an inevitable part of the learning process, and present great learning opportunities. Learners should feel that they can make errors without being unduly criticized and then proceed to strive energetically toward success [Spitzer (a)]
- **Challenge:** Challenges have the extraordinary ability to bring out the best in people. Almost everybody will respond enthusiastically to appropriate challenges [Spitzer (a)]
- **Measurement:** In sport, scoring is used positively to track progress, whereas all too often in learning measurement or grading is used to find fault. The key to creating motivating learning measurement is to focus on formative (improvement-oriented) evaluation, and encourage self-measurement [Spitzer (a)]
- **Feedback:** In sports, feedback tends to be immediate and predominantly encouraging, while learning feedback too often tends to be discouraging. Learning feedback can be dramatically improved by providing it continuously, reinforcing the positives, and focusing feedback on how performance can be improved in the future [Spitzer (a)]
- **Recognition:** Finally, highly motivated learners are a product of plenty of recognition for the progress they make. It is ongoing recognition that really motivates. This is achieved through pointing out as many positives as possible throughout the learning process [Spitzer (a)]

It is worth noting that all of the motivators are examples of intrinsic motivation approaches, with the exception of measurement, feedback and recognition. The three exceptions are associated with the extrinsic motivation approaches due to their ability to provide the learner with the suitable praise or reproach.

### **2.4.5 Summary**

Motivation is the key to effective instruction. It is responsible for increasing students' attention and enthusiasm, thereby enhancing the retention of knowledge. It is the motivators discussed in Section 2.4.4 that create a more motivating context for learning, regardless of subject matter or delivery system. Effective instructional design allows the classroom and other learning environments to release as much excitement, energy, and sense of achievement as are seen on basketball courts and playing fields around the world.

## **2.5 Teaching Methodologies**

### **2.5.1 Introduction**

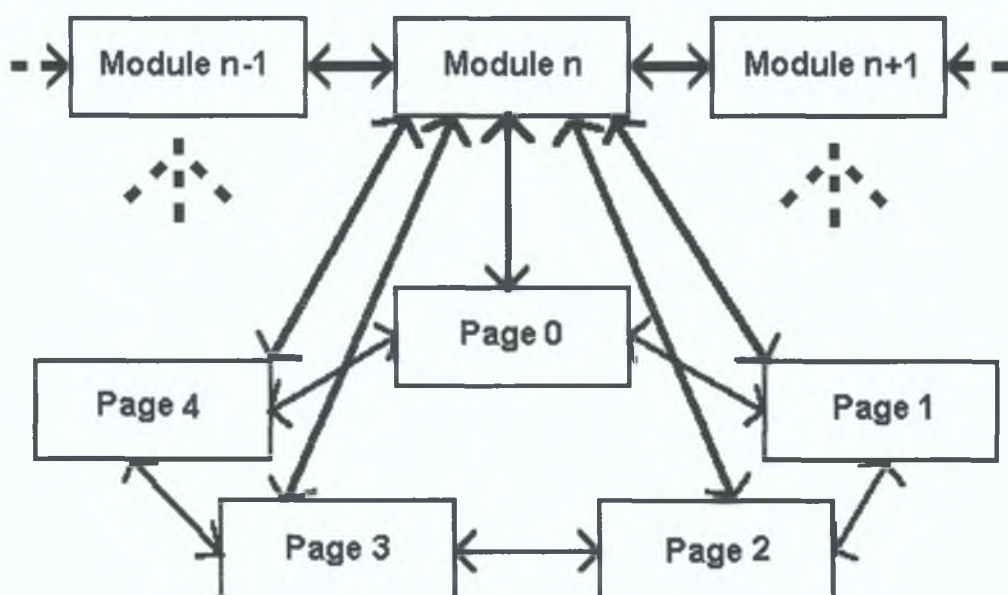
The structure of the courseware is vitally important to the success of an educational package. The course material and its structure must be clearly presented to the students on commencement of instruction. This ensures that students are fully aware of the course content and its objectives. In addition, if the courseware is developed with a modular structure the student can clearly see how topics interrelate.

### **2.5.2 A Proposed Structure**

The proposed structure for an effective CAL package incorporates Multimedia into a hierarchical and modular structure through the use of hypermedia. As discussed in Section 2.21, the incorporation of multimedia provides an interactive environment that engages the learner's attention, imagination and senses, thereby increasing knowledge retention. The employment of hypermedia to provide a hierarchical structure is based on Ausubel's theory of subsumption [cited in Merrill et al.].

Ausubel's theory of subsumption relies on two assumptions about the cognitive structure [Merrill et al.]: (1) knowledge is organized hierarchically with the more abstract components higher up the hierarchy (progressive differentiation), and (2) the information in a knowledge structure is interrelated [Hoffman]. Ausubel's first assumption implies that instructional design should ensure the more general

information be communicated higher up the courseware hierarchy, and conversely the more specific information should be communicated at the lower branches of the hierarchy. A defining characteristic of hypermedia is its concern with how aspects of subject-matter content are interrelated and the overall general context of the subject matter. This is certainly consistent with Ausubel's second assumption about cognitive structure. Figure 2 shows a model of the proposed hierarchical and modular structure, with inherent interlinkedness.



**Figure 1.2: Hierarchical and Modular structure**

The modular structure of a CAL package may be achieved through a standard and consistent educational environment through which all of the courseware is conveyed. The environment may incorporate various media and hypermedia, however its delivery system must always be consistent. In effect, there must be a standardized system of typography, universal graphical facilities, and operationally equivalent demonstrations.

The instructional design of a package should cater for students of varying abilities. The non-sequential approach to learning allows the explanation of supplemental material, without disturbing the flow of the core topic [Galvin]. The hyperlinked structure allows for a rich, multi-perspective view on a knowledge domain; learners are free to

explore the domain by browsing and navigating through it. Moreover, the richness, interlinkedness, and multi-perspectivedness may be argued to lead to a deeper understanding of the domain [Gros et al]. However, the hypertext nodes may be linked together in a hodgepodge network that leaves students frustrated and confused as to whether or not they made it through all of the material they should have and whether or not they learned the material. Rezabek and Ragan [Rezabek et al] suggest that the key to building a hypermedia knowledge environment is to find a balance between instruction and exploration. A framework must be created to guide and structure the learner's progress, but the learner must also be allowed to create associations and follow related pathways and ideas. Hypermedia-based instruction needs to provide both structure and freedom.

### **2.5.3 Implications**

The apparent implications of the proposed structure are increased levels of student interest, attention, and hence motivation. The hierarchical and modular structure is conducive to effective and efficient learning. The use of multimedia tends to add an element of the fun motivator to the learning experience. The integration of hypermedia into an instructional system allows students to choose their own learning path at the touch of a button. Hypermedia is therefore supplying both the action and choice motivators. As discussed in Section 2.4, the inclusion of motivators in an instructional system is inclined to enhance the educational process.

With regard to the courseware developer, the methodology is designed to be easy to use and to represent a complete modular design process. It enables designers to focus on building systems that will be expandable by adding more key concepts into the hyperweb, maintainable and amenable to modifications as curriculum needs change via key concept node alterations, and reliable through verifying links and instructional effectiveness [Mengel et al].

### **2.5.4 Summary**

Because of the modular and hierarchical structure of the courseware, students do not see topics as wholly independent, but rather they see them as connected applications based upon a common general established principal. They can go about the business of learning, at their own pace without having to worry about their whereabouts in the courseware. Hence, their interest is responsible for guiding their progression.

The hypermedia instructional system should facilitate the division of the material into hypertext nodes that are structured according to key concepts in the material. The methodology should ensure the hypertext linkages are tight within key concept boundaries, but loose between boundaries. This has the effect of promoting the desired modular structure.

## **2.6 Conclusions**

It is clear from the discussion provided in this chapter that an era of education reform is emerging. The advances in computing technology have created a world-wide classroom of learners through the expansive network of the Internet. The Web protocol of the Internet facilitates the dissemination of knowledge in an interactive-multimedia environment, which has been determined to be a valuable source of additional courseware.

The hypermedia environment provided by Web browsers makes it possible for educators to structure the courseware such that it accommodates diverse learning styles and varied learning rates. This type of a structure seeks to motivate learners, thereby augmenting their interest and attention levels. As a result, the students' learning experiences are enhanced, thus achieving improved knowledge retention.

The Internet is incontestably an excellent source of information that is bound to enhance education. It is the opinion of the author that the Internet provides a great source of additional material to a course, and may even lead to successful Internet-based distance learning programs.

# **Chapter 3: The CAL Environment**

## **3.1 Introduction**

The purpose of this chapter is to discuss some of the necessary components associated with the desired CAL environment. The components considered result from an analysis of the work carried out by the author prior to commencement of the current project (Section 1.2) combined with the research discussed in Chapter 2.

This Chapter comprises a detailed discussion of the Internet, comprising the World Wide Web and the Java programming language. The use of [MATLAB] as a computational engine is also discussed with regard to its communication with an Internet browser. In conclusion, the choice of an effective CAL environment is made.

### **3.1.1 Requirements**

As discussed in Section 2.5, an effective CAL environment would incorporate Multimedia into a hierarchical and modular structure via hypermedia. The computing technology of today is quite capable of relaying multimedia applications to students in the fast, seamless manner.

The development of such a multimedia environment should not be restricted to a single computing platform, such as Microsoft Windows '95. The CAL package must facilitate a modular and extensible construction mechanism, thus facilitating partial use of the developing product. In addition, the framework of the courseware's hypermedia should be clearly intelligible and cater for incremental development. The CAL courseware should also be readily available to learners.



### **3.1.2 Recommendations**

Due to the accessibility of the Internet and its support of the World Wide Web (WWW, W3 or the Web), the Internet may be considered a suitable platform over which to communicate CAL courseware. It is the evolution of the Web protocol that has transformed the Internet from a text haven to a multimedia and hypermedia archetype.

## **3.2 The Internet**

The Internet is a fairly recent phenomenon. Approximately 25 years old and with users numbered only in the thousands in the early 1980s, it now boasts 150,000 http (hypertext transport protocol) servers, 12 million host computers connecting with the Internet through other means and in excess of 30 million users. With current research suggesting online households world-wide will double to more than 66.6 million by the year 2000 [Thackray]. One definite factor in the explosion of interest in the Internet must be the development of the World Wide Web and Web browsers such as Netscape [Netscape].

The Internet is not about computing or calculating, it is about communication or, as has been argued recently, about "simulation, navigation and interaction" [Maul]. As a result, learning can take place wherever a connected computer is located -- college computer laboratory, organization learning center, desk or home, and even on the move as digital satellite technology advances [Thackray].

### **3.2.1 The Internet Platform**

The Internet is essentially a network of networks that links computers, allowing access to each other, in the form of text, graphics, sound, etc. The multimedia information supplied by the Internet may be read by internal converters and displayed by the converting application, or displayed with the aid of other helper applications [CTI Autumn 1995].

The Internet processes encrypted information, enclosed by identification markers. The information is split into packets of smaller information, including sequencing markers, and sends it into the Internet using the TCP/IP protocol [CTI Autumn 1995]. The Internet protocols choose any route available for sending each small packet. The packets may not all follow the same route. A computer at the receiving end waits until it receives all the relevant information packets, checks that all the expected ones have arrived by comparing the identifying and sequencing markers, reassembles them into the original continuous format, and presents the information to its processor and user. Note that all the methods described connect to the network or server only for the duration of the request.

The Internet is platform independent. This means that the information received from the Internet may be accessed and processed by the client machine, irrespective of its operating system. In effect, computer operating systems such as Dos, Unix, Microsoft Windows and Macintosh can all communicate using a standard Internet protocol. The TCP/IP protocol is the most common Internet protocol. It provides a convention that allows computers to communicate with each other. The Web is essentially a formatted text based information server.

### **3.2.2 World Wide Web**

The World Wide Web was originally conceived at Conseil Européen pour la Recherche Nucléaire (CERN) [CERN] in Switzerland in the late 1980's as a method of distributing hypertext documents internally among physicists at the Particle Physics Laboratory. The phenomenal growth in the WWW outside of CERN started when a few undergraduate students at the University of Illinois National Center for Supercomputing Applications [NCSA] created a Web browser called Mosaic [Mosaic]. Mosaic was designed to use the protocols and specifications developed at CERN to enable users on all platforms to exchange, access, view, and manipulate distinct types of information via a computer network using a simple point-and-click interface. Since Mosaic's first release in early 1993, both the number of WWW servers and the volume of WWW traffic on the Internet have grown exponentially.

The Web supports a myriad of protocols, including Usenet News, HTTP, Gopher, Gopher+, FTP, Telnet, TN3270 and mail. The aforementioned all add to the usability and versatility of the Web. The Web accesses HTML documents using addresses in the form of a URL (Universal Resource Locator), and takes the form “http://web-server-domain-name/directory/sub-directory/home-page”. An example of such a URL would be “http://eeng.dcu.ie/calcon/paul/index.html” which is the home page for the ANN courseware.

The Web is the universe of network-accessible information, the embodiment of human knowledge. The Web has a body of software, and a set of protocols and conventions. Through the use of hypermedia and multimedia techniques, the Web is easy for anyone to roam, browse, and make a contribution. The Web is not the Internet, it is a structure on top that is part data and part navigational tool. The Web uses hypertext links to tie together collections of documents dispersed across the Internet. Since such documents can contain text, pictures, audio and video, and since the technology can also allow for real-time tool sharing and interactive video, the potential to offer a rich multimedia learning experience is considerable [Thackray].

The Web material is generally formatted in a specific type of language called “html”--hyper text mark-up language, which is readily converted by Web client browsers for display. Other document formats require conversion by helper applications. HTML documents may be produced incrementally, thus facilitating the development of a library of information that is readily extensible. Moreover, due to the modular development of hyperlinked documents, it is possible to develop a framework of hypermedia documentation that is readily intelligible.

### **3.2.2.1 Internet Browsers**

Internet browsers are Web clients. Two of the more popular Web clients or browsers are Mosaic [Mosaic] and Netscape [Netscape]. Web browsers are readily available from the URL site of the respective browser, and usually free of charge. The appropriate browser application can be downloaded at the touch of a button. The browser application is however platform dependent, so the operating system of the browser application must be specified prior to downloading.

Web browsers are used to interpret HTML documents, which may in turn access and display pictures, audio and video clips. They may also invoke helper applications to perform operations that are not supported by the browser itself. They are also responsible for the functionality of hypermedia, linking documents in a non-linear fashion. Web browsers supply a high level of interactivity through their HTML support of hypermedia, JavaScript (see Section 3.4) and Java language inserts or applets (see Section 3.3).

### 3.2.2.2 The Hypertext Mark-up Language

HyperText Mark-up Language or HTML is composed of a set of elements that define a document and guide its display. The HTML system of typography allows complex hypermedia documents to be communicated by the Internet as simple text files. As a result, HTML documents require minimal storage space, thus solving the problem of memory constraints.

A HTML document is composed of a single element: `<html>...</html>`, that is, in turn, composed of head and body elements: `<head>...</head>`, and `<body>...</body>`. To allow older HTML documents to remain readable, `<html>`, `<head>`, and `<body>` are actually optional within HTML documents. The HTML tag names are used to instruct the browser on how a particular element of text, picture, etc. is to be displayed. A HTML element may include a name, some attributes and some text or hypertext, and will appear in an HTML document as:

```
<tag_name> text </tag_name>,  
<tag_name attribute_name=argument> text </tag_name>, or just  
<tag_name>.
```

For example the HTML code:

```
'Normal Sized Black Text & <FONT COLOR="#7F7F7F" SIZE=+3>Big Grey  
Text</FONT>'
```

would appear as:

Normal Sized Black Text & Big Grey Text

In addition to altering the physical appearance of text, HTML can also be used to show tables, script and style elements, symbols and glyphs used in mathematics, forms and international characters. Html tags are also used to embed hypertext links to other Web documents using the `<A>` and `</A>` tags as follows

```
<A HREF="LinkedDocument.html"> Click Here! </A>
```

The “LinkedDocument.html” file is the document that is loaded on clicking the mouse button over the ‘Click Here!’ text

## 3.3 The Java Programming Language

### 3.3.1 History of Java

The development of the Java programming language began in 1990 by Sun Microsystems. In 1990, a team of Sun researchers developed some concepts for a new direction in high-tech, consumer-driven technology. Even in 1990, it was obvious that computers were everywhere and were the driving force behind many of the products in the home: the VCR, the microwave oven, the security system, even the stereo system. Unfortunately, almost everyone regarded computers, especially the ones made by Sun Microsystems, as being too weird and too difficult to use. Sun, which felt that it had lost a major opportunity by not getting in on the ground floor of the PC explosion, launched a strange new project that would become Java: technology for everyday people [Newman et al.]

In 1990, a subsidiary of Sun Microsystems – Sun Laboratories established a team, code-named Green. James Gosling, a member of the Green Team had been developing the answer to a replacement language for C++ at home. Gosling realised that C++ was close, but its emphasis was speed, not reliability. In consumer electronics, reliability is more important than speed, and C++ programs had a tendency to crash at unpredictable times. In the meantime, a fellow Green Team member, Patrick Naughton had been developing the device's interface: a series of graphic animations. The two men brought the halves of the language together, and by August 1991, a new language called Oak (named for a tree outside Gosling's window) was born.

In 1994, Bill Joy saw an opportunity for Oak in the dramatic emergence of the World Wide Web. Joy's idea was to follow Netscape's [Netscape] Internet Play model and release Java for free over the Internet. Internet Play is the profitless approach to market share: by giving your product away for noncommercial use, you can make it the standard [Newman et al.]. In January 1995, Oak was renamed Java—which, incidentally, does not stand for just another vague acronym, it does not stand for anything.

In December 1995, Microsoft signed a letter of intent with Sun for a Java-technology source license. Additionally, Microsoft agreed in principle to give Sun two things: Microsoft's reference implementation of the Java virtual machine, and the applet application programming interface (AAPI) for Windows [Newman et al]. The deal was important. By integrating Java into its Explorer browser, Microsoft provided Java with a huge base of previously untapped Windows users. In addition, it was a major endorsement from the world's largest software firm that Sun's Internet technology is top-notch and goes a long way toward establishing Java as the *de facto* open standard for programming on the Internet.

With Java, Sun established the first programming language that was not tied to any particular operating system or microprocessor. Applications written in Java will run anywhere, eliminating one of the biggest headaches for computer users: incompatibility between operating systems and versions of operating systems.

### **3.3.2 Java Language Definition**

Java is a programming language used to compose a set of instructions. These groups of instructions are called programs, applications, executables, or, in the case of Java, applets. Java can be used to build stand-alone programs, called applications, just like any other programming language [Newman et al]. It is the applets that are the most innovative thing about Java. Java applets add life to the Web.

Java applets can provide animation, local data searches, and a wide variety of other functions and features that just were not possible without the Java environment. Java can be defined neatly in a set of buzzwords. Sun Microsystems' official definition of Java is

'Java: A simple, object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, high-performance, multithreaded, and dynamic language.'

### **3.3.2.1 Java is Simple**

Java was designed as close as possible to C++ in order to make the system more familiar, more comprehensible, and to shorten the time necessary to learn the new language. Java's features were chosen to ease development, implementation, and maintenance of software, while omitting things that would hinder the development process. This was achieved through the elimination of multiple class inheritance, operator overloading, memory de-allocation, and more.

### **3.3.2.2 Java is Object-Oriented**

Java is object-oriented (OO). This means that it's part of a family of languages that focuses on defining data as objects and the methods that may be applied to those objects. Such a structure is known as a class.

The benefits of object-oriented programming (OOP) are modularity, reusability via inheritance, and naturalness of OO analysis and modeling. The aforementioned make objects more abstract and stable, thus unifying the development approach to OOP and reducing the design life cycle.

### **3.3.2.3 Java is Distributed**

Java being distributed means that Java applets and applications can open and access objects across the Web via URLs as easily as they can access a local file system. The beauty of a distributed system is that multiple designers, at multiple remote locations, can collaborate on a single project. [Newman et al.]

### **3.3.2.4 Java is Interpreted (and Compiled)**

Java is interpreted. However, strictly speaking, Java is both interpreted and compiled. In fact, only 20 percent of the Java code is interpreted by the browser—but this is a crucial 20 percent. [Newman et al.] Both Java's security and its ability to run on multiple platforms stem from the fact that the final steps of compilation are handled locally. Java is first compiled into bytecode, using the Java compiler. This bytecode is binary and architecture-neutral. In the case of an applet, the bytecode is not runnable until it is interpreted by a Java run-time environment, such as a browser.



Since each Java run-time environment is platform-specific, the final product is going to work on that specific platform. This means that irrespective of the developing environment a Java applet may be viewed on several different platforms, providing a Java-capable browser is used.

### **3.3.2.5 Java is Robust**

Java is said to be robust because it is a very reliable language. The more robust a language is, the less likely that programs written in this language will crash and the more likely it is that they will be bug-free. A Java program can not cause a computer to crash because it does not have permission to access all of the computer's memory. Java programs can access only a restricted area of memory, so erroneous manipulation of memory does not occur.

As a final safety check, Java has the linker. The linker is part of the run-time environment. It understands the code format and during run-time repeats many of the type-checks done by the compiler to guard against version mismatch problems.

Java's robustness is one of its most important features. It allows programmers to focus on programming, rather than chasing bugs or memory leaks. Subsequently, more people will work with Java, creating the standard that Sun hopes it will be.

### **3.3.2.6 Java is Secure**

Java's security is still something that remains to be proven [Newman et al]. The security problem exists because nobody wants to unknowingly download a virus from the Internet onto their machine. The fact that Java is robust, interpreted and compiled, is an aid to its security. For example, the imposed memory restrictions mean that Java programs can be safely executed on the client machine.

Java programs are first compiled into bytecode instructions that are not platform-specific and contain extra type information. This type information can be used to verify the program's legality and to check for potential security violations. This verification process is used to ensure the bytecode has not been contaminated or altered and that it

conforms to Java language constraints. Moreover, since Java uses names instead of numerical addresses to access methods and variables, it is therefore easier to identify which methods and functions are actually being accessed. The aforementioned all imply safer, more secure program execution.

### **3.3.2.7 Java is Architecture Neutral**

The Green Team that developed Java did so with the intention of it supporting applications on networks, it would have to support a variety of systems with a variety of microprocessors and operating system architectures [Newman et al.]. This was achieved through the generation of platform-independent bytecode instructions that are easily translated into native machine code at runtime. The only prerequisite being that the run-time environment is a 32-bit operating system.

The advantage offered by this system of bytecodes, is that the same version of software runs on all platforms. Therefore, a program need only be compiled once. It does not have to be recompiled for each different platform. Apart from the obvious advantages for networks, the implications of this architectural neutrality for software developers are that multiple platform-specific versions of the same product need not be produced.

### **3.3.2.8 Java is Portable**

Java's portability stems from its architectural neutrality. The platform-independent bytecode makes the concept of porting from one platform to the next a little redundant. The Java system itself was built to be portable. Javac, the Java compiler, is written in Java, and the run-time environment is written in ANSI C to be portable. Even beyond the platform independence, Java's standard data types eliminates the 'implementation dependent' aspects of the specification that are found in C and C++ [Newman et al.]

For example, in Java the sizes of the primitive datatypes are specified, as is the behaviour of arithmetic on them. Two examples that Sun uses are that 'int' always means a signed two's complement, 32-bit integer, and 'float' always means a 32-bit IEEE 754 floating point number. Almost all interesting CPU's share these

characteristics Furthermore, the Java libraries define portable interfaces for the three most common platforms UNIX, Windows and Machintosh

### **3.3.2.9 Java is High Performance**

Java is not high performance in the sense of being faster than a comparable C++ routine In fact, in some cases, it is almost the same speed In benchmark tests run by Sun Microsystems on one of their SPARCStation 10 machines, the performance of bytecodes converted to machine code is almost indistinguishable from native C or C++ Of course, that does not take into account the time of run-time compilation [Newman et al ] However, execution of a Java application, upon the average personal computer or PC, is significantly slower than that of its C++ counterpart Java is almost 20 times slower than C++ That being said, developers are working on code generators to enable Java programs to run nearly as quickly as those written in C++

In most circumstances, the performance of the interpreted bytecode is quite good However, in certain instances interpreting bytecodes may not meet the needs of the application Fortunately, Java is capable of linking C and C++ methods with those of a Java application, and hence optimize performance

### **3.3.2.10 Java is Multithreaded**

Multithreading means that the application can perform multiple actions simultaneously What multithreading means, to Java users, is that they do not have to wait for the application to finish one task before beginning another For example, if you were viewing a computerised animation, one thread could be handling the mathematics, while another was taking care of the graphics

What this built-in multithreading means to programmers is that constructing multithreaded programs is a lot easier in Java [Newman et al ] The synchronisation features eliminated much of the difficulties of dealing with a multithreaded environment's unpredictable nature

### **3.3.2.11 Java is Dynamic**

Java is a more dynamic language than C or C++. Unlike these languages, Java was designed to adapt to an evolving environment. Java makes it easy to make use of the object-oriented paradigm.

### **3.3.3 The Java Environment**

The Java environment is command-line driven. In other words, the Java environment is not graphical, as are other popular languages like Visual C++ and Visual Basic, which use a GUI (Graphical User Interface) environment [Newman et al]. Command-line driven also means that you call JDK's (Java Development Kit's) tools, including the compiler, from the command prompt (DOS prompt for Windows NT and 95), and the text of the Java file (filename.java) is generated using an ASCII text editor such as the MS-DOS Editor or Windows 95's WordPad.

The JDK's environment consists of an Applet Viewer, a Java Compiler, a Java Interpreter, a Java Disassembler, a Java Documentation Generator and a Java Debugger. It is the Java compiler that generates the class file (filename.class), which contains the necessary bytecode to be interpreted at run-time. Although the JDK was designed for command-line instruction, there are however more graphical packages developed that enhance the development process. One such package is the Microsoft Visual J++.

Microsoft Visual J++ is an integrated Windows-hosted development tool for Java programming. Visual J++ is built around Developer Studio, Microsoft's common development environment. It consists of a set of tools that run under Microsoft Windows 95 or Microsoft Windows NT (versions 4.0 and later). Developer Studio provides a set of tools to complete, test, and refine a program. You can control the operation of all the tools from a single, integrated GUI environment without the need for time-consuming command-line instructions.

### 3.3.4 Internet Support

The Internet supports Java language inserts in the form of applets that are embedded into HTML documents using the tag names, as discussed in Section 3.3.2. These applets, are transparently downloaded into the browser along with the HTML pages in which they appear. With applets as the main focus, Java has demonstrated a new way to make use of the Internet to distribute software. This new paradigm goes beyond browsers. Java is an innovation with the potential to change the course of computing.

Java applets are embedded in a Web document using the `<Applet>` and `</Applet>` HTML tags as follows,

```
<APPLET CODE="filename.class" HEIGHT=200 WIDTH=400></APPLET>
```

The “filename.class” contains the pre-compiled code necessary to run the applet. The height and width tag names are used to inform the browser of the size of the applet.

## 3.4 JavaScript

In December 1995, Netscape [Netscape] Communications Corporation and Sun Microsystems announced a new scripting language called JavaScript. JavaScript is an open, cross-platform scripting language that complements the features of Java. JavaScript is easy to learn, even by users who have little or no programming background. JavaScript was first available in the beta versions of Netscape Navigator 2.0 [Netscape, Newman et al.]

Although browsers have many functions, they are essentially multimedia display tools—nothing more. JavaScript provides client-side capabilities that allow a retrieved Web document to modify itself or the retrieving document. This capability moves the power of server-side programs to the client, in essence empowering the browser and making it more intelligent.

As the name suggests, JavaScript is a script language. Scripts allow a Web browser to intelligently deal with situations that otherwise would require a program on the Web.

server. In addition, the user's perception is that the situation is handled much faster, because the browser does not need to send the request to the server and display the response. The distinction between Java and JavaScript is important. Java applets must be compiled and downloaded by the browser before they are executed, JavaScript scripts are simply downloaded and interpreted.

JavaScript code is embedded in a HTML document using HTML tags as follows,

```
<SCRIPT LANGUAGE="JavaScript">  
.  
  
// JavaScript Code  
  
</SCRIPT>
```

JavaScript is a complementary language that provides some of the features of Java without requiring the user to learn object-oriented programming. JavaScript can recognise and respond to various events that are generated by the browser software, including mouse clicks and movements, page loading, and form input. In fact, an excellent use of JavaScript is to perform error checking in online forms. In effect, it relinquishes the need of a server-side program to perform the error checking, thereby reducing the network traffic and the time needed for transmission of data.

## **3.5 The Computational Engine**

### **3.5.1 Introduction**

As discussed in Section 1.3.2, Artificial Neural Networks require considerable computational power in order to perform the numerous calculations associated with their training and testing. It is therefore necessary to choose a package or system, which can adequately meet these mathematical needs. In addition to the mathematics, such a system must possess the necessary graphing facilities required to properly convey a concept.

Since the courseware is developed for the Internet, it must also be possible for the computational engine to be accessed by the Internet. This contingency suggests that the Internet browser would either communicate with an external client-based computational engine, or that the engine itself would be incorporated into a Web-based application. The former requires the client machine possesses the engine prior to accessing the Web courseware. This is of course contrary to the whole concept of the Internet's stand-alone accessibility. On the other hand, the latter would overcome this predicament, and relieve the inconvenience and expense of obtaining an external application.

### **3.5.2 MATLAB**

MATLAB [MATLAB] is a technical computing environment for high performance numerical computation and visualisation. The environment contains a Command Window into which MATLAB instructions may be inputted. Its operation is oriented towards the processing of matrices, through its expansive range of internal functions. MATLAB also possesses excellent plotting facilities, comprising optional colours, two or three dimensions, log scales, labeling and zooming capabilities.

A vast range of functions are made available through MATLAB's Toolboxes. The Toolboxes range from elementary mathematical functions to specialized functions such as those contained in the Neural Network Toolbox. The Toolbox functions, which frequently consist of little more than the combination of some of the more elementary

functions, make it easy to perform complex mathematical and graphical operations. They are what make MATLAB such a useful computational tool. MATLAB also contains the tools necessary to build a GUI (Graphical User Interface) or set up a communications link with another application using DDE (Dynamic Data Exchange).

In addition to MATLAB's numerical and graphical abilities, its environment is also very familiar, especially to those connected with the field of engineering. As a result, the development of instructional examples and problems is made more effective and efficient. It is therefore quite suitable for CAL, particularly if used in conjunction with lecture material.

### **3.5.2.1 MATLAB Communications**

#### **3.5.2.1.1 Success to Date**

As discussed in Section 1.2, the fourth year project investigated "A Computer Aided Learning Package to Teach System Dynamics and Control" [Keeling]. The project, SYSCAL, was developed on the Microsoft Windows platform, using the Borland C++ programming language in conjunction with MATLAB, as the computational engine.

The success of the communications link between the SYSCAL application and MATLAB was limited to unidirectional DDE, with MATLAB as the server and SYSCAL as the client. In other words, it was possible to send data from MATLAB to the SYSCAL package, but not vice-versa. In addition, the synchronisation of the data transmission was somewhat erratic. These encumbrances were not overcome within the allotted time of the fourth year project.

Due to the difficulties that arose while attempting inter-application communications, DDE was deemed as being too difficult to realise. As a result, it was necessary to consider other communication options.



### **3.5.2.1.2 The VCLab Approach**

Virtual Control Laboratory or VCLab [VCLab] is a newly developed software tool which allows a communications channel to be set-up between Netscape [Netscape] and the Computational engine of MATLAB [MATLAB, Galvin] The VCLab application may be downloaded, in the form of a ZIP [WinZip] (compressed) file, from the Web site “<http://www.esr.ruhr-uni-bochum.de/VCLab/>” The package was developed in the University of Bochum, in Germany by Stefan Mueller and Christian Schmidt Used in conjunction with Java and HTML, values can be passed to and from the computational engine as required It consists of three software elements

**1) MATLAB Plugin;**

**2) EMF-plugin; and**

**3) Java DDE.**

These form the basis for the communications between the browser and the MATLAB engine Plugins are used to provide added functionality to a browser The MATLAB and EMF Plugins were specifically developed for Netscape [Netscape] using Microsoft Windows 95/NT's DDE system The MATLAB Plugin has methods associated with it that can be called from within Java Applets or from the JavaScript code [Galvin] These are used to first invoke the MATLAB engine and then to pass values back and forth between the browser and MATLAB In order to pass the plots generated in the computational engine, they are first sent to the Clipboard which is one of Windows systems of inter-application communication Netscape [Netscape] is able to access them via the EMF Plugin method, and can be displayed on the Web page

In order for the Plugin methods to be used, the browser must be informed of how to use the Plugin This is achieved in a HTML Web document, through the use of HTML tag-names For example

```
<EMBED TYPE="application/x-matlab" HIDDEN=true NAME="matlab">
```

The type, 'application/x-matlab' is the name of the Plugin to be used. The 'HIDDEN' tag provides the option of hiding the workspace of the application from the user, via the true or false values. The 'NAME' tag refers to the name of the application concerned, in this case the MATLAB computational engine. In order to insert or embed MATLAB plots into the Web document, the second Plugin, EMF, must be declared.

```
<EMBED TYPE="image/x-emf" WIDTH=400 HEIGHT=90 NAME="title">
```

The size of the image is provided so that the Web document can load the neighbouring information prior to receiving the MATLAB plot. The name tag is used as a reference tool by the JavaScript code.

### **3.5.2.1.3 Summary**

The VCLab approach offers a definite solution to the communications problem with the computational engine. It is purely Java and HTML driven, both being widely used for Internet applications. The package also makes extensive use of both the MATLAB and Windows DDE systems, making VCLab very efficient. The only problem with the VCLab approach is one of portability.

Portability is one of the Internet's greatest assets. It allows people to access a plethora of information, without the need to install expensive applications. The development of a CAL package which relies on the user downloading and/or purchasing software greatly diminishes its portability. In order to use the VCLab software, one must first download the Zipped (compressed) package onto the client machine, followed by un-zipping it, which may involve the obtainment of the Zipping application itself. In addition, the user must obtain a suitable version of MATLAB. The aforementioned all result in reduced portability, which may deter learners from using the Internet CAL courseware.

### **3.5.3 Java as a Computational Engine**

The advantage of using Java as a computational engine is that it is stand-alone. All one needs to view a Java applet is a Java-capable browser. Java applets are downloaded with its associated Web document onto the client machine, where it is interpreted and executed forthwith. This means that the server-side need not communicate with the running applet, thereby reducing the network traffic and the time needed for transmission of data.

In addition, since the applet is client-based, its speed of execution is dependent only on the host machine's microprocessor. As Java is a high level programming language which is comparable to C++, applets are capable of performing the complex mathematical calculations necessary for engineering applications. The Java language is also especially suited to the development of GUIs and graphical displays, which supply a superior level of interactivity to those generated within the MATLAB environment. It is Java's versatility that makes it such a well-suited language on which to develop effective CAL courseware.

With regard to Artificial Neural Networks (ANNs), Java's object-oriented development is comparable to that of ANNs architectures and training methods. An ANN neuron is essentially an object with associated data and methods. The data may consist of a bias, weights, etc., whereas the methods would perform input-output calculations. The neuron objects may be combined to form a network object. The data associated with this network object comprises the neuron objects and additional data concerning the type of network and its configuration. The methods associated with the network object would comprise the training and testing algorithms. It is clear from this analogy that ANNs are well-suited to Java's object-oriented development approach.

There are however disadvantages to using Java as a computational engine. Although Java is a high level programming language and is therefore fast and efficient, it is low level with regard to its support of matrix algebra and graphing facilities. Unlike MATLAB, Java does not have the generic tools available to perform the complex matrix operations and graphing procedures required by ANN demonstrations. In fact,

these facilities must be programmed from the bottom up. Therefore, using Java as a computational engine imposes a sizeable development time.

### **3.6 Conclusions**

The Internet is a highly versatile world-wide network, supporting a diverse range of information protocols. The Internet's Web protocol is what has catapulted the attributes of the Internet into the limelight. It is the Web's support of multimedia, hypermedia and Java applets that make it possible for the Internet to supply such a rich learning experience to students all over the world. The Web provides a highly interactive environment through its support of both hypermedia and Java applets.

HTML's interactive hypertext make it possible to link Web documents either in a non-linear fashion or in the required hierarchical and modular structure, as discussed in Section 2. In addition, due to its modular structure, the courseware's documentation may be developed incrementally as the lecture material itself is composed.

Although hypermedia supplies interactivity to a Web browser, it is Java applets that make the Web a highly interactive and versatile graphical computing environment. The advantages of Java include reliability, architectural neutrality, portability, high-performance and object-oriented development. Unlike VCLab, Java is highly portable in that all its execution requires is a Java-capable browser. It is this portability and its high performance graphical and computational capabilities that make it such a capable computational engine, especially with regard to ANN's.

In conclusion, the Internet and Web, through their incorporation of hypermedia, multimedia and Java, have the potential to provide an effective learning environment to a world-wide classroom.

# **Chapter 4: A Prototype Course in Artificial Neural Networks (ANNs)**

## **4.1 Introduction**

This chapter discusses the presentation of ANN lecture material. The Chapter also explains the theory concerning the ANN courseware for which the Java demonstrations discussed in Chapter 5 were developed to illustrate. The ANN theory is intended to convey a sufficient level of knowledge to the reader such that the concepts discussed in Chapter 5 will be understood.

## **4.2 The Course Structure**

The ANNs courseware was compiled by Dr John Ringwood at the School of Electronic Engineering, DCU (Dublin City University). In addition to the lecture material, the course comprises two journal papers [Lippmann, Hush et al], MATLAB [MATLAB] examples and assessments, and in and out of class Java demonstrations. While the format of the course still focuses on traditional lectures, the supplementary material provided by MATLAB and the Java demonstrations play an important role in the educational process.

### **4.2.1 Lectures**

The lecture material was developed for the MEng (Masters in Electronic Systems) program in DCU, specifically the elective Taught Masters program. The lectures were conducted by Dr Ringwood, for a period of two hours, over 12 weeks within Semester 2 of the academic year 1996 / 97.

The Students were provided with the lecture notes both from DCU's internal network and the Internet ANNs web site [ANNs Web Site]. This had the effect of allowing the students' attention to be focused on the lecture material itself, rather than the

transcription of notes. In addition, students' had the option of reviewing the notes prior to the lecture, thereby reinforcing their understanding.

#### 4.2.2 The ANN Web Site

The ANN web site [ANNs Web Site], set up by the author, comprises a reproduction of the lecture notes in a hypertext environment. The environment is hierarchically structured, as recommended in Section 2.5, such that a particular topic can be accessed from a cascade of hyperlinked menus (See Figure 4.1). The top and bottom of Figure 4.1 show additional navigation tools. The icons, like the hypertext, are activated with the aid of the mouse. The back (leftmost) icon is used either to back-step to the current notes' menu or to move to the previous page of notes. The forward (middle) icon is used either to advance to the next page of notes, if it exists. The menu (rightmost) icon accesses the current notes' menu.



## Self-Organizing Systems

- Introduction
- Self-Organizing Feature-Mapping Algorithm
- Self-Organizing Feature-Map Example
- Properties of the SOFM Algorithm



**Figure 4.1** Hierarchical Hyperlinked Menus

The fact that the courseware is available on the web is one way of facilitating increased student interaction [McClellan et al.]. The web may be used to view all the

lecture notes and Java demonstrations. It also contains the files associated with the MATLAB examples and course assessments.

### **4.2.3 MATLAB Examples and Exercises**

Throughout the course, the courseware emphasises and illustrates both the theoretical and practical dimensions of ANNs. In addition to the Java demonstrations, this was achieved through numerical and graphical MATLAB examples and analytically based homework assignments. The combined MATLAB examples and exercises tend to increase students' understanding, while developing a practical consideration of the process of training and testing neural networks.

Due to MATLAB's versatility, students can experiment with various neural network configurations and thereby develop a greater understanding of their properties. For example, in one exercise, students were asked to determine the dynamical modeling abilities of Radial Basis Function (RBF) networks. This involved experimenting with various RBF training procedures and determining their respective advantages through analysis of the training data, network configuration and test results. Students now submit, along with standard text, all requisite plots and software files, where appropriate.

### **4.2.4 Java Demonstrations**

The development of the Java demonstrations is the focus of the work described in this thesis. They provide a superior level of interaction and versatility to the Internet ANN courseware, without the need for employing additional software packages. As a result, students gain access to the ANNs courseware with a minimal encumbrance on their part.

The Java demonstrations facilitate students accessing and experimenting with the ANNs systems outside the classroom, and so gain a better, more practical understanding of a particular neural network. In addition, the Java demonstrations can be used as a lecture aid, as discussed in Section 4.2.5.

### 4.2.5 Java In-Class Demonstrations

An in-class demonstration involves the lecturer presenting one of the web-based Java demonstration discussed in Section 4.2.4. This may be achieved by bringing into the classroom, a computer connected to the Internet with a Java-capable browser and accessing the desired Java demonstration. Alternatively, if an Internet connection is not readily available in the classroom, the Java demonstration may be downloaded onto the host machine prior to its transportation. The demonstration may then be presented to the students with the aid of a large computer screen or an overhead projector.

The in-class demonstrations are an effective means of presenting an overview of a particular topic [Schodorf et al.]. For example, in the early stages of the Hopfield lecture notes, students are introduced to the idea that Hopfield networks can perform pattern recognition. To reinforce this idea, the equations that underlie the behaviour of the network are derived. This has the effect of developing the students' insight into the Hopfield network so that a better understanding of the theory may be obtained.

When these demonstrations are presented in class, they allow some student interaction. However, the student can only influence the demonstration through the lecturer. The demonstrations also tend to hide many implementation details. Glossing over the details is good when the student is first exposed to a new topic, because the goal of the demonstration is to motivate a connection between a fundamental concept and its physical manifestation—an idea and the concept that underlies it [Schodorf et al.].

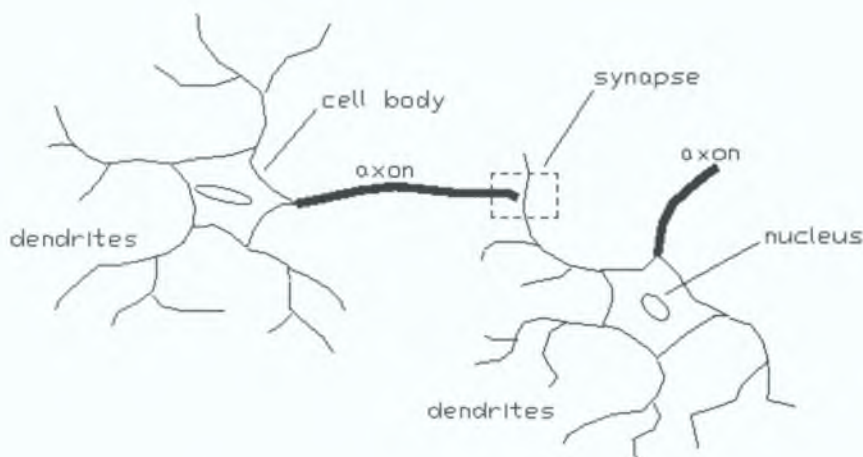


## 4.3 Artificial Neural Networks

### 4.3.1 Introduction to ANNs

Artificial Neural Networks (ANNs) are simplified models of the central nervous system. They are networks of highly interconnected neural computing elements that have the ability to respond to input stimuli and to learn to adapt to the environment [Patterson].

Figure 4.2 shows the biological inspiration for ANNs. In a standard picture of the neuron, dendrites receive inputs from other cells, the cell body processes and integrates the inputs, and information is transmitted along the axon to the synapses, whose outputs provide input to other neurons or to effector organs [Anderson]. Through electrochemical means, the synapses allow one cell to influence the activity of others. If the *sum* of the signals received by a neuron exceeds a threshold, the neuron is said to *fire* [Hebb].

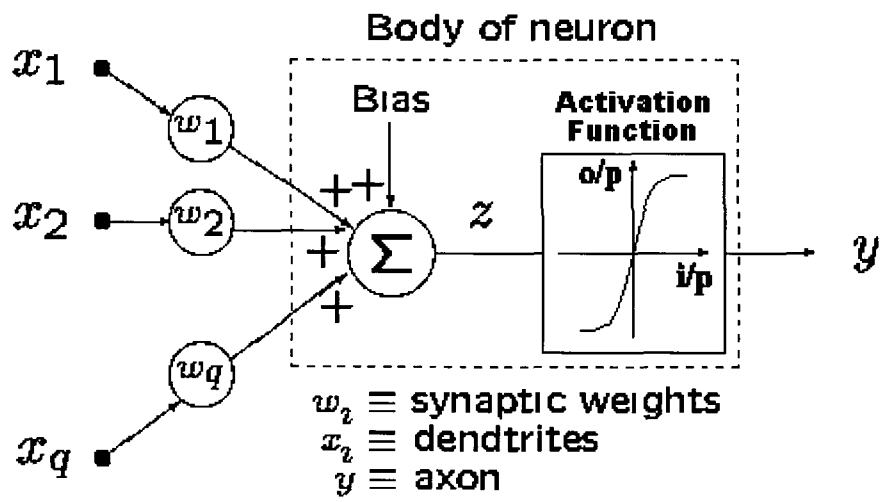


**Figure 4.2** Biological Inspiration

Figure 4.3 shows a model of the artificial neuron. The inputs,  $x_i$ 's are analogous to axons, whose activation is dependent on the physical size of its associative synaptic junction, or synaptic weight,  $w_i$ . The weighted inputs are analogous to the dendrites which are processed by the nucleus. In the artificial neuron the nucleus may be considered to be a combination of the *adder* which sums the weighted inputs with the built-in bias and the *activation function* or *squashing function*,  $\phi$  which is used to

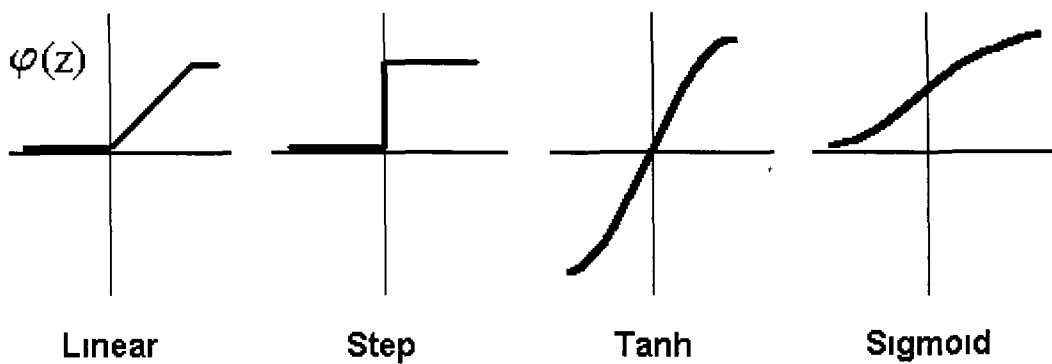
limit the amplitude of the neuron’s output. Typically, the normalised amplitude range of the output of an artificial neuron is within the closed unit interval  $[0,1]$  or alternatively  $[-1,1]$  [Haykin]. The activation function usually takes the form of those depicted in Figure 4.4.

The model of the artificial neuron, like its biological counterpart, consists of weighted inputs, a bias, and an activation function. ANN computational models consist of a number of neurons working in parallel. Different neural network *architectures* can be used to solve a given classification or prediction problem [Drossu et al.].



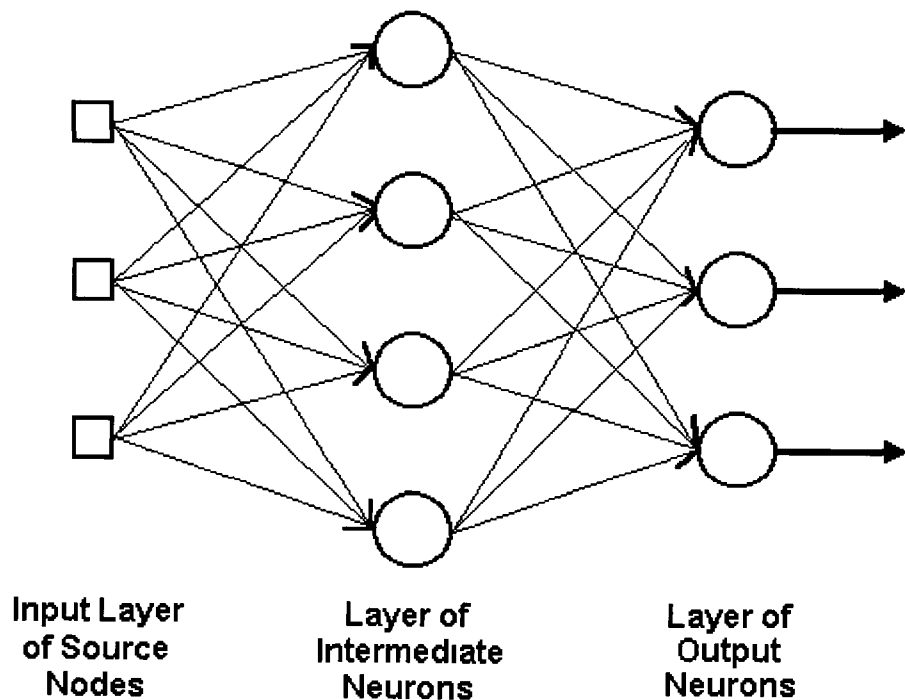
**Figure 4.3** Model of Biological Neuron

In a network, the neurons are interconnected through *synaptic links* and are grouped in *layers*, with synaptic links usually connecting neurons in adjacent layers. Typically three different layer types can be distinguished (see Figure 4.5): input (the layer that external stimuli are applied to), output (the layer that outputs results to the exterior world), and hidden (the intermediate computational layers between input and output layer) [Drossu et al.].



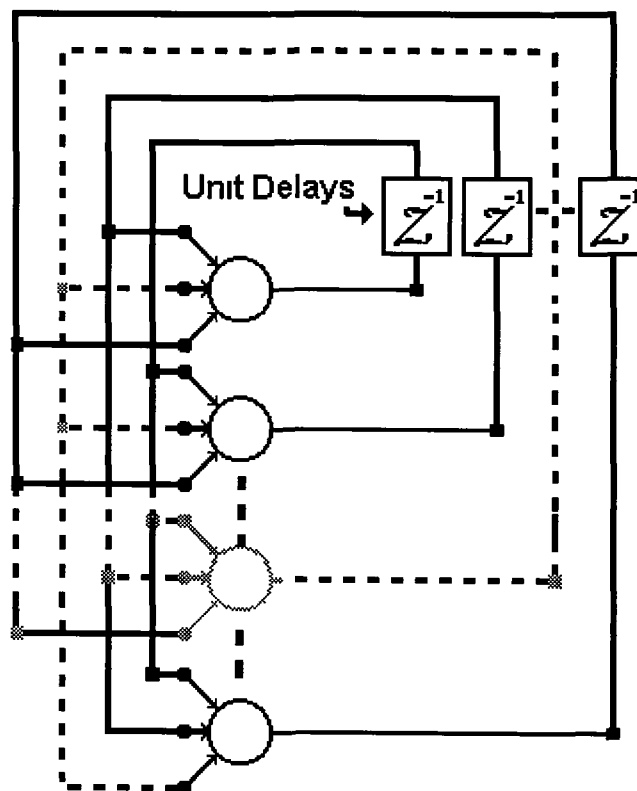
**Figure 4.4** Activation Functions

The neural network architectures can be, roughly speaking, divided in two categories *feedforward*, also shown in Figure 4 5, in which the signal flow between different layers is unidirectional (from input layer toward output layer), or *recurrent*, as shown in Figure 4 6, in which the flow between different layers is bi-directional (both from input layer toward output layer as well as the reverse) [Drossu et al ]



**Figure 4.5** Layers of a Feedforward Network

The most distinctive element in neural networks, as opposed to traditional computational structures, is denoted as *learning*. Learning represents the adaptation of some characteristic neural network parameters by using a set of examples (known outcomes of the problem for given conditions so that for given input patterns the outputs reflect the value of a specific function). The final data modelling goal is not only to memorise some known patterns, but to be also able to *generalise* from prior examples (to be able to estimate the outcomes of the problem under unknown conditions) [Drossu et al ]



**Figure 4.6** A Recurrent Network

### 4.3.2 Simple Neural Networks

#### 4.3.2.1 McCulloch-Pitts Neuron

The first formal definition of a synthetic neuron model based on the highly simplified considerations of the biological model described in the preceding section was formulated by McCulloch and Pitts [McCulloch et al ] The McCulloch-Pitts model of the neuron is shown in Figure 4.7 The inputs  $x_i$ , for  $i=1, 2, \dots, n$ , are 0 or 1, depending on the absence or presence of the input impulse respectively [Zurada]

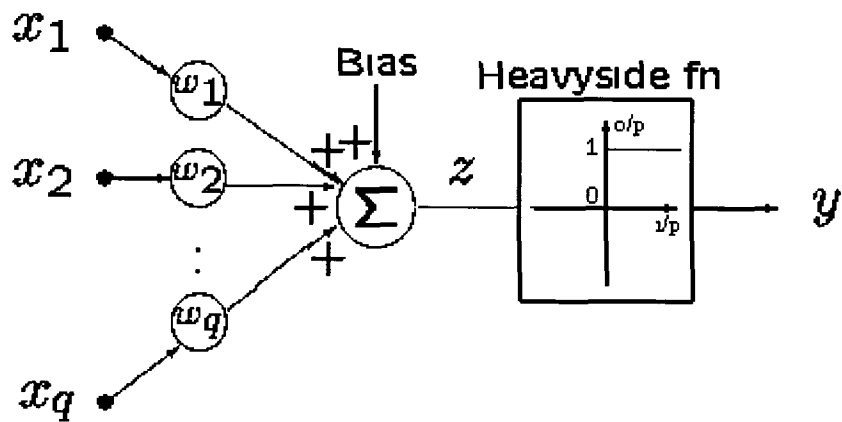


Figure 4.7 McCulloch-Pitts Neuron

The distinguishing feature of the McCulloch-Pitts neuron is its *hard limiting* activation function. The hard limiter forces the output of the neuron to either 0 or 1 depending on whether the input is less than zero or greater than or equal to zero respectively.

#### 4.3.2.2 Hebbian Learning

Donald Hebb [Hebb] first proposed a learning scheme for updating neuron's connections that is now referred to as the *Hebbian Learning Rule*. Hebb reasoned

“When an axon of cell A is near enough to excite cell B and repeatedly or persistently takes place in firing it, some growth process or metabolic change takes place in one or both cells, such that A's efficiency, as one of the cells firing B, is increased ” [Hebb]

He stated that the information can be stored in connections, and postulated the learning technique that had a profound impact on future developments in this field. Hebb's learning rule made primary contributions to neural networks theory [Zurada]

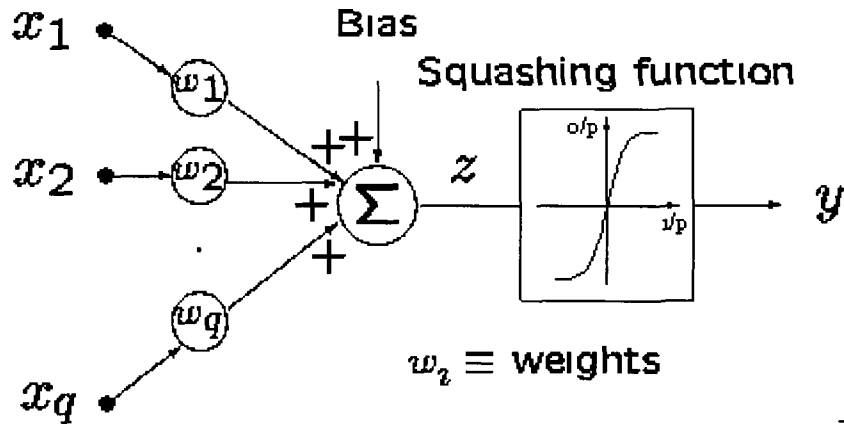
The rule states that if the crossproduct of output and input, or correlation term  $y_i x_j$  is positive, this results in an increase of weight  $w_{ij}$ , otherwise the weight decreases. As a result, the output is strengthened for each input presented. Therefore, frequent input patterns will have most influence at the neuron's weight vector and will eventually produce the largest output [Zurada]. Equations 1 and 2 describe the weight adjustments performed while learning, where  $\mu$  is the associated *learning rate*.

$$\Delta w = \mu x y \quad (1)$$

$$w_{k+1} = w_k + \Delta w \quad (2)$$

#### 4.3.2.3 Single Layer Perceptron Learning

The *perceptron* was invented by Frank Rosenblatt in 1958 [Rosenblatt]. The perceptron is the simplest form of a neural network used for the classification of a special type of patterns said to be *linearly separable* (patterns that lie on opposite sides of a hyperplane). Rosenblatt proved that if the patterns (vectors) used to train the perceptron are drawn from two linearly separable classes, then the perceptron algorithm converges and positions the decision surface in the form of a hyperplane between the two classes. The proof of convergence of the algorithm is known as the *perceptron convergence theorem* [Haykin].



**Figure 4.8** Model of the Perceptron

The perceptron model, shown in Figure 4.8, has essentially the same structure as that of the McCulloch-Pitts except for its activation function, which depends on the application. The activation function is usually in the continuous form of Equation 3

$$y = \frac{2}{1 + e^{-\beta z}} - 1 \quad (3)$$

As a result of equation 3, the output is restricted to the range  $-1 < y < 1$ , and the activation function is differentiable

Perceptron learning is *supervised*. This means that the desired signal response is required in order to train the neurons. In fact, the learning signal is proportional to the response error  $e$  (i.e., the difference between the desired response,  $d$  and the actual response,  $y$ ), described in Equation 4. Equations 5 and 6 describe the weights adjustment,  $\Delta w$  performed while training, where  $\mu$  is the learning rate usually in the range of  $0.01 \rightarrow 0.1$

$$e = d - y \quad (4)$$

$$\Delta w = \mu e x \quad (5)$$

$$w_{k+1} = w_k + \Delta w \quad (6)$$

## 4.3.3 Multi-Layer Perceptron Networks

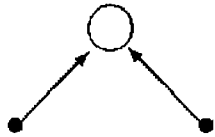
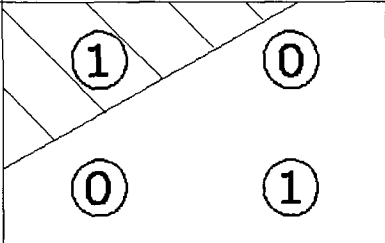
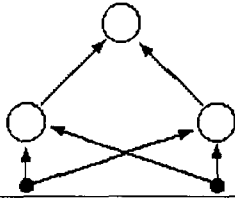
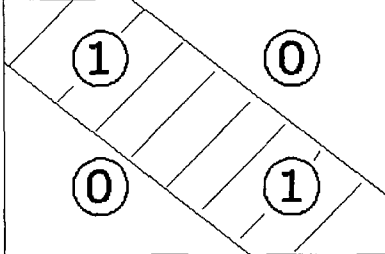
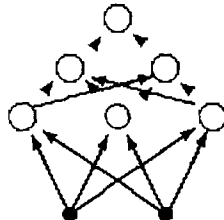
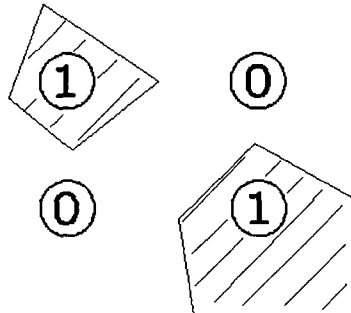
### 4.3.3.1 Introduction

Multi-layer perceptrons or MLPs consist of a set of sensory units or source nodes that constitute the *input layer*, one or more *hidden layers* of computation nodes, and an *output layer* of computation nodes (see Figure 4.5). Figure 4.5's architecture is referred to as a 2-layered network because it has one hidden layer and one output layer of neurons, the input vector is generally not referred to as a layer. MLPs represent a generalisation of the single-layer perceptron considered in Section 4.3.2.3. MLPs are feedforward networks of neurons with non-linear activation functions.

### 4.3.3.2 Functional Capabilities of the MLP Network

The capabilities of MLPs stem from the non-linearities used within the nodes. If nodes were linear elements, then a single-layer network with appropriately chosen weights could exactly duplicate those calculations performed by any multi-layer network. The capabilities of perceptrons with one, two, and three layers that use hard-limiting non-linearities are illustrated in Figure 4.9 [Lippmann]. The capabilities of the MLP can be viewed from three different perspectives. The first has to do with its ability to implement Boolean logic functions, the second with its ability to implement non-linear transformations for functional approximation problems, and the third with its ability to partition the pattern space for classification problems [Hush et al.]. The former two have been developed into the Java demonstrations discussed in Section 5.3.3.



| Structure  | Types of decision regions   | Exclusive OR problem   |
|--|---|--|
| <b>single-layer</b><br> | half-plane<br>bounded by<br>hyperplane                            |    |
| <b>two-layer</b><br>    | convex<br>open or<br>closed<br>regions                            |    |
| <b>three-layer</b><br> | arbitrary<br><br>complexity<br>limited by<br>number of<br>neurons |  |

**Figure 4.9** Capabilities of MLP Networks

#### 4.3.3.3 The Back-Propagation Training Algorithm

One of the limitations of Rosenblatt's original formulation of the MLP was the lack of an adequate learning algorithm. The most common approach is to use a gradient descent algorithm, but the key difficulty in deriving such an algorithm for the MLP was that the gradient is zero almost everywhere when the hard-limiting non-linearity is used. The solution is to use a non-linearity that is differentiable. The non-linearity most often used is the *sigmoid* function. With this non-linearity, it is possible to implement a gradient search of the weight space [Hush et al]. The algorithm developed is known as the *error back-propagation algorithm* or just the *back-propagation algorithm*.

Basically, the error back-propagation process consists of two passes through the different layers of the network: a forward pass and a backward pass. In the *forward pass*, an activity pattern (input vector) is applied to the sensory nodes of the network,

and its effect propagates through the network, layer by layer. Finally, a set of outputs is produced as the actual response of the network. During the forward pass the synaptic weights of the network are all fixed. During the *backward pass*, on the other hand, the synaptic weights are all adjusted in accordance with the error-correction rule. Specifically, the actual response of the network is subtracted from a desired (target) response to produce an *error signal*. This error signal is then propagated backward through the network, against the direction of synaptic connections—hence the name “error back-propagation”. The synaptic weights are adjusted so as to make the actual response of the network move closer to the desired response. Appendix A describes mathematically the back-propagation algorithm. See Section 5.3.3 for examples of the given training algorithm.

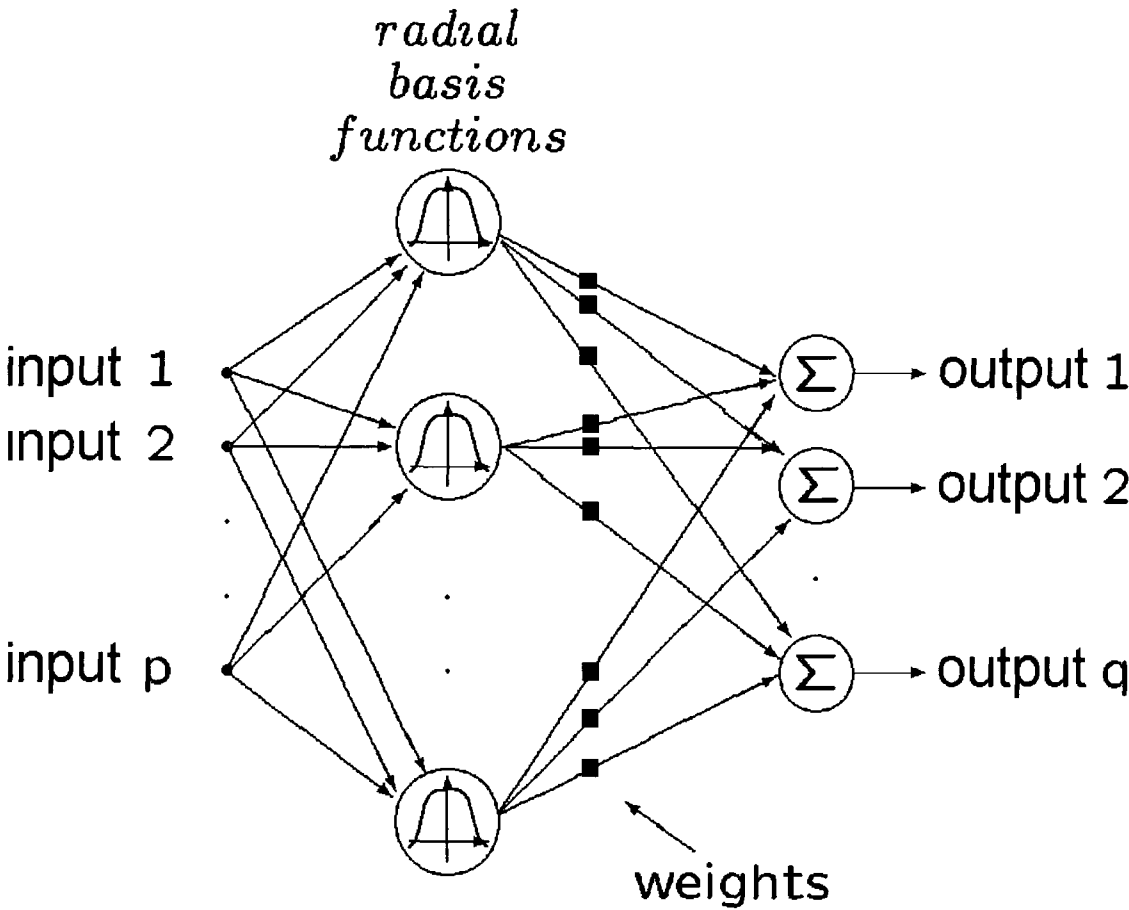
The momentum term  $\alpha$  described in Step 4 of Appendix A, can be adjusted in an attempt to increase the learning efficiency. The goal of the algorithm is decrease the *sum squared error* (SSE) of the outputs on consecutive iterations such that a minimal SSE is found as soon as possible. It is therefore desirable for the  $\Delta w_{ij}$  term to increase in magnitude, hence the inclusion of momentum, which tends to accelerate descent in steady downhill directions. Similarly, the learning rate  $\rho$  may be adjusted by an *increase learning rate* factor or *decrease learning rate* factor, depending on whether the SSE term has decreased or increased respectively. Usually, the *increase learning rate* factor is applied to the  $\rho$  when the SSE has changed by no more than a factor of what is called the *maximum error rate* (around 1.05). The aforementioned are frequently referred to as the *training settings*.

In general, the back-propagation algorithm does not converge at the global minimum on the SSE performance surface. An important consideration associated with the back-propagation algorithm is when to stop the algorithm. Two approaches may be adopted. The first approach is simply to stop the algorithm after a fixed *number of epochs*, which represent the number of training data cycles (i.e., a full batch of training data) that have been applied to the network. The second approach is concerned with stopping the algorithm once the sum squared error has reached a sufficiently small threshold, *SSE goal*. The *number of epochs* and *SSE goal* terms are also included in the *training settings* (see Section 5.4.3).

### 4.3.4 Radial Basis Function Networks

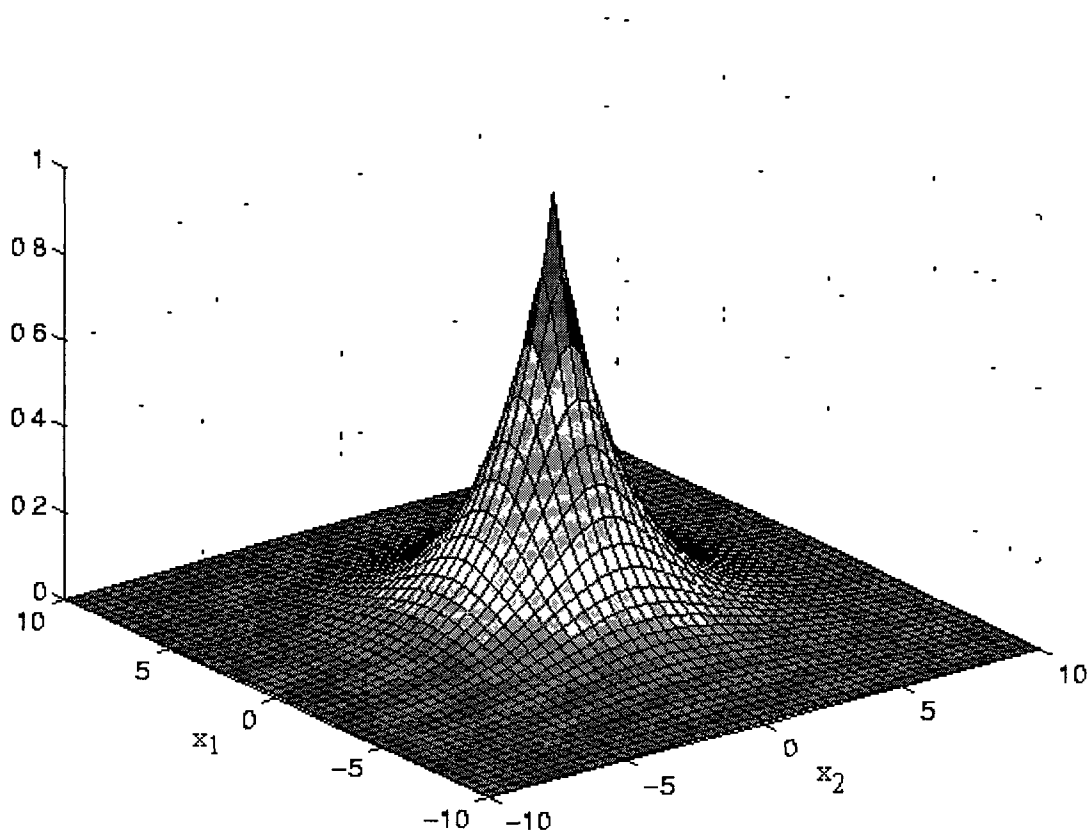
#### 4.3.4.1 Introduction

Radial-basis functions or RBFs were first introduced in the solution of the real multivariate interpolation problem. The early work on this subject is surveyed by Powell [Powell]. Broomhead and Lowe were the first to exploit the use of radial-basis functions in the design of neural networks [Broomhead et al., Haykin].



**Figure 4.10** RBF Network Architecture

The construction of a RBF network in its most basic form involves three entirely different layers (see Figure 4.10). The input layer is made up of source nodes (sensory units). The second layer is a hidden layer of high enough dimension, which serves a different purpose from that in a multi-layer perceptron. The output layer supplies the response of the network. The transformation from the input space to the hidden-unit space is *non-linear*, whereas the transformation from the hidden-unit space to the output space is *linear* [Haykin]. The justification for this architecture is provided by *Cover's theorem on the separability of patterns*, which states that a complex pattern-classification problem cast in a high-dimensional non-linear space is more likely to be linearly separable than in a low-dimensional space [Cover].



**Figure 4.11** Gaussian Function

Although implementations vary, the most common basis function is a *Gaussian kernel function*,  $\phi$  of the form given in Equation 7

$$\varphi(x, x_j) = \varphi(\|x - x_j\|) = \exp\left[-\frac{(x - x_j)^T(x - x_j)}{2\sigma_j^2}\right] \quad (7)$$

where  $\varphi$  is the output of the  $j$ th node in the first layer,  $x$  is the input pattern,  $x_j$  and  $\sigma_j$  are the *centre* and *spread factor* respectively for Gaussian node  $j$ . Figure 4.11 shows the response of a two-dimensional RBF or Gaussian function with its centre at (0,0) and a spread factor of 1.

#### 4.3.4.2 Functional Capabilities of the RBF Network

The RBF network can be used for both classification and functional approximation, just like the MLP. Both the MLP and RBF network are capable of forming an arbitrarily close approximation to any continuous mapping. The primary difference between the two is the nature of their basis functions. The hidden layer nodes in a MLP form sigmoidal basis functions, which are nonzero over an infinitely large region of the input space, while the basis functions in the RBF network cover only small localised regions.

While some problems can be solved more efficiently with sigmoidal basis functions, others are more amenable to localised basis functions [Hush et al.]. In the case of the classification problem where a decision boundary separates one class from another, the RBF network provides a more efficient solution than the MLP network. The efficiency of the RBF network becomes even more pronounced as this problem is extended to higher dimensions (in higher dimensions, one class forms a complete shell around the other) [Hush et al.].

Regardless of the dimension, the RBF network will require only a single hidden layer node to solve this problem, but the MLP network will require on the order of  $n$  (the input dimension) hidden layer nodes. The reverse is true for the function approximation problem, where the MLP network is more efficient [Hush et al.].

#### 4.3.4.3 Training Algorithm for the Interpolation Problem

There are a variety of approaches to learning in the RBF network. Most of them start by breaking the problem into two stages: learning in the hidden layer, followed by learning in the output layer. Learning in the hidden layer is typically performed using an unsupervised method, i.e., a clustering algorithm, while learning in the output layer is supervised [Hush et al.]

Consider the feedforward network with an input layer, a single hidden layer, and an output layer consisting of a single unit. The interpolation or function approximation problem may be defined as

Given a set of  $N$  different points  $\{x_i \in \mathbb{R}^p \mid i = 1, 2, \dots, N\}$  and a corresponding set of  $N$  real numbers  $\{d_i \in \mathbb{R}^1 \mid i = 1, 2, \dots, N\}$ , find a function  $F: \mathbb{R}^p \rightarrow \mathbb{R}^1$  that satisfies the condition

$$F(x_i) = d_i, \quad i = 1, 2, \dots, N \quad (8)$$

i.e., the interpolating surface is required to pass through all the training points

RBF networks specify an interpolating function of the form of Equation 9

$$F(x) = \sum_{i=1}^N w_i \phi(\|x - x_i\|) \quad (9)$$

The norm above  $\|\cdot\|$  is usually taken to be the Euclidean norm. The known data points,  $x_i \in \mathbb{R}^p$ ,  $i = 1, 2, \dots, N$  are the centres of the radial-basis functions.

Inserting the interpolating conditions of Equations 8 and 9, the following is obtained

$$\begin{bmatrix} \phi_{11} & \phi_{12} & K & \phi_{1N} \\ \phi_{21} & \phi_{22} & K & \phi_{2N} \\ M & M & & M \\ \phi_{N1} & \phi_{N2} & K & \phi_{NN} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ M \\ w_N \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ M \\ d_N \end{bmatrix} \quad (10)$$

where

$$\phi_{ji} = \phi(\|x_j - x_i\|), \quad j, i = 1, 2, \dots, N \quad (11)$$

Equation 10 may be rewritten as

$$\Phi w = d \quad (12)$$

It can be shown using *Light's theorem* [Light] that  $\Phi$  is positive definite, hence the solution given in Equation 13 is obtained

$$w = \Phi^{-1}d \quad (13)$$

Although in theory it is always possible to obtain a solution to the strict interpolation problem, in practice Equation 12 can not always be solved when matrix  $\Phi$  is close to singular. In addition, the problem may be over-determined, thereby generating poor interpolation between training data points. As a result, an alternative approach must be adopted that ensures a solution can always be obtained.

The new approach involves searching for a sub-optimal solution in a lower-dimensional space. The approximated solution  $F^*(x)$  is of the form shown in Equation 14

$$F^*(x) = \sum_{i=1}^M w_i \phi_i(x) \quad (14)$$

The new set of basis functions,  $\{\phi_i(x) | i = 1, 2, \dots, M\}$ , is assumed to be linearly independent. Typically, the number of basis functions is less than the number of data

points ( $1 \leq M \leq N$ ), and the  $w_i$  constitute a new set of weights. With radial-basis functions in mind,  $\varphi_i(x)$  is set as follows

$$\varphi_i(x) = G(\|x - x_i\|), \quad i = 1, 2, \dots, M \quad (15)$$

where the set of centres  $\{x_i | i = 1, 2, \dots, M\}$  is to be determined, and  $G$  represents Green's function as described in Equation 16

$$G(x, x_i) = \exp\left(-\frac{1}{2\sigma_i^2} \|x - x_i\|^2\right) = \exp\left(-\frac{1}{2\sigma_i^2} \sum_{q=1}^p (x_q - x_{iq})^2\right) \quad (16)$$

The generalised RBF network solution is obtained by Haykin to be the following

$$w = (G^T G + \rho G_0)^{-1} G^T d \quad (17)$$

where  $\rho$  is the *regularisation parameter* that specifies the desired trade-off between complexity and fitting of the training data,  $G$  and  $G_0$  are described by Equations 18 and 19 respectively

$$G = \begin{bmatrix} G(x_1, x_1) & G(x_1, x_2) & K & G(x_1, x_N) \\ G(x_2, x_1) & G(x_2, x_2) & K & G(x_2, x_N) \\ M & M & & M \\ G(x_N, x_1) & G(x_N, x_2) & K & G(x_N, x_N) \end{bmatrix} \quad (18)$$

$$G_0 = \begin{bmatrix} G(x_1, x_1) & G(x_1, x_2) & K & G(x_1, x_M) \\ G(x_2, x_1) & G(x_2, x_2) & K & G(x_2, x_M) \\ M & M & & M \\ G(x_N, x_1) & G(x_N, x_2) & K & G(x_N, x_M) \end{bmatrix} \quad (19)$$

Note that as the regularisation parameter  $\rho$  approaches zero, the weight vector  $w$  converges to the pseudo-inverse (minimum-norm) solution to the over-determined least-squares data-fitting problem (see Equations 20 and 21), as shown by Broomhead and Lowe [Haykin]



$$\underline{W} = G^+d, \quad \rho = 0 \quad (20)$$

where  $G^+$  is the pseudo-inverse of matrix  $G$ , that is,

$$G^+ = (G^T G)^{-1} G^T \quad (21)$$

The solution obtained in Equation 17 provides a solution to the weights connecting the hidden layer to the output layer. There is however still the important consideration of how to choose the position of the centres,  $\{x_i | i = 1, 2, \dots, M\}$ . There are a number of strategies, perhaps the simplest of which is use *fixed centres*, which are evenly placed over the input space. The spread factors are then chosen according to

$$\sigma = \frac{l}{\sqrt{2M}} \quad (22)$$

where  $l$  is the maximum distance between centres

## 4.3.5 Hopfield Networks

### 4.3.5.1 Introduction

The MLP and the RBF networks considered in the previous two sections represent important examples of a class of neural networks known as non-linear layered feedforward networks. In this section another important class of neural networks that have a *recurrent* structure such as that of Figure 4.6 is considered.

In 1982, John Hopfield presented a paper at the National Academy of Science [Hopfield] describing how an analysis of stable points could be performed for symmetrical recurrent networks [Patterson]. Hopfield's idea was to store each pattern at the bottom of a "valley" of an energy landscape, and then permitting a dynamical procedure to minimise the energy of the network in such a way that the valley becomes a basin of attraction [Haykin].

The Hopfield network may be viewed as a non-linear *associative memory* or *content-addressable memory*, the primary function of which is to retrieve a pattern (item) stored in memory in response to the presentations of an incomplete or noisy version of that pattern [Haykin]. Hopfield illustrated this meaning in the following quote from his 1982 paper:

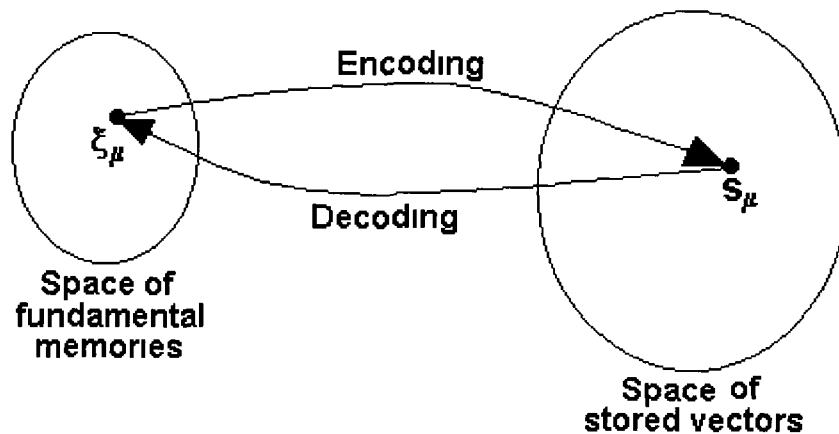
Suppose that an item stored in memory is "H. A. Kramers & G. H. Wannier, Phys. Rev. 60, 252 (1941)." A general content-addressable memory would be capable of retrieving this entire memory item on the basis of sufficient partial information. The input "& Wannier (1941)" might suffice. An ideal memory could deal with errors and retrieve this reference even from the input "Wannier, (1941)."

An important property of a content-addressable memory is therefore the ability to retrieve a stored pattern, given a reasonable subset of the information content of the pattern. Moreover, a content-addressable memory is *error-correcting* in the sense that it can override inconsistent information in the cues presented to it [Haykin].

The essence of a content-addressable memory (CAM) is to map a fundamental memory  $\xi_\mu$  onto a fixed (stable) point  $s_\mu$  of a dynamic system, as illustrated in Figure 4.12. Mathematically, the mapping can be expressed in the form

$$\xi_\mu \Leftrightarrow s_\mu$$

The arrow to the right describes an *encoding* operation, whereas the arrow to the left describes a *decoding* operation. The stable points of the phase space of the network are the *fundamental memories* or *prototype states* of the network. Suppose the network is presented a pattern containing partial but sufficient information about one of the fundamental memories. This particular pattern represents a starting point in the phase space. In principle, provided that the starting point is close to the stable point representing the memory being retrieved, the system should evolve with time and finally converge onto the memory state itself, at which point the entire memory is generated by the network. A Hopfield network may therefore be described as a *dynamic system whose phase space contains a set of fixed (stable) points representing the fundamental memories of the system* [Haykin]



**Figure 4.12** Coding-Decoding performed by a Hopfield Network

Hopfield networks can be either *discrete time* (using difference equations) or *continuous time* (using differential equations). The standard discrete-time version of the Hopfield network, considered in Section 4.3.5.3, uses the McCulloch-Pitts (Figure 4.7) model for its neurons. Retrieval of information stored in the network is

accomplished via a dynamical procedure of updating the state of a neuron being picked *randomly* and one at a time. This *asynchronous dynamical procedure* is repeated until there are no further state changes to report. The *synchronous dynamical procedure* involves updating all neurons simultaneously using the previous states of the network. Again, the procedure is repeated until the network stabilises.

#### 4.3.5.2 Functional Capabilities of the Hopfield Network

One of Hopfield's original applications was the associative memory. An associative memory is a device which accepts an input pattern and produces as an output the *stored pattern* which is most closely associated with the input. In the Hopfield associative memory, the input/output patterns are binary. For example, the input pattern may be a noisy version of one of the stored patterns. The function of the associative memory is to recall the corresponding stored pattern, producing a clean version of the pattern at the output [Hush et al.]

#### 4.3.5.3 Operational Features of the Hopfield Network

The algorithm for training a Hopfield network is described by Appendix B. Note that all of the test patterns are made up of a number of binary digits or bits which can be taken to be -1 or +1 at the output of each unit (McCulloch-Pitts neuron). Thus, the activation function is of the form of a hard-limiter.

With regard to the weight matrix,  $W$ , it is worth noting that the diagonal elements are all zero (i.e.,  $w_{ij} = 0$  for  $i = j$ ). In effect, each neuron has a feedback connection to every other neuron except for itself. This means that the next state of a particular neuron is dependent on the current state of every other neuron in the network. The rationale for the zero diagonal elements lies in the stability and convergence properties of the discrete-time model. Sufficient conditions for stability are that  $W$  be positive definite. Actually these conditions depend on the type of update that is used in the node equations. If *synchronous* updates are used, which implies that all nodes are updated simultaneously as suggested by Figure 4.6, then  $W$  must be positive definite, but when *asynchronous* updates are used (one node at a time) it is sufficient that the diagonal elements of  $W$  be nonnegative, that is  $w_{ii} \geq 0$  [Hush et al.]

In order to mathematically determine the resemblance of the current state of the network with a particular stored pattern, the *Hamming Distance* formula described by Equation 23, may be used

$$d_H(\xi, \zeta) = \frac{1}{2} \sum_i |\xi_i - \zeta_i| \quad (23)$$

where  $d_H(\xi, \zeta)$  is the Hamming distance,  $||$  obtains the absolute value,  $\xi_i$  denotes the stored pattern of neuron  $i$ , and  $\zeta_i$  denotes the current state of neuron  $i$ . The Hamming distance can be interpreted as the number of bits which are different in  $\xi$  and  $\zeta$ .

The storage and recall properties of Hopfield's discrete-time model are well known. It has been shown that if the input patterns are less than  $N/2$  away in Hamming distance from a stored pattern, and if the  $M$  stored patterns are chosen at random, the maximum asymptotic value of  $M$  for which all  $M$  of the patterns can be perfectly recalled is

$$M \leq \frac{N}{4 \ln N} \quad (24)$$

For example, if  $N$  is 120 then the storage capacity,  $M$  is 6, as ( $M \leq 6.266$ ). If  $M$  is significantly greater than 6 then the number of spurious attractors will increase. *Spurious states* are states which represent local minima in the 'Energy Landscape', but are not associated with stored patterns. They arise from a number of sources

- *Inverse States* represent the inverse of the stored patterns, with an inverse pattern for each stored pattern. They have the same domain of attraction as the stored state.
- *Mixture States* represent a linear combination of an odd number of stored patterns.

- *Spin Glass States* are so called because of a close correspondence with spin glass models in statistical mechanics. They are not correlated with any finite number of original patterns and are most significant in the case of a large number of stored patterns.

The *energy function* of the discrete-time Hopfield network is defined by (assuming that the externally applied threshold is zero for all neurons)

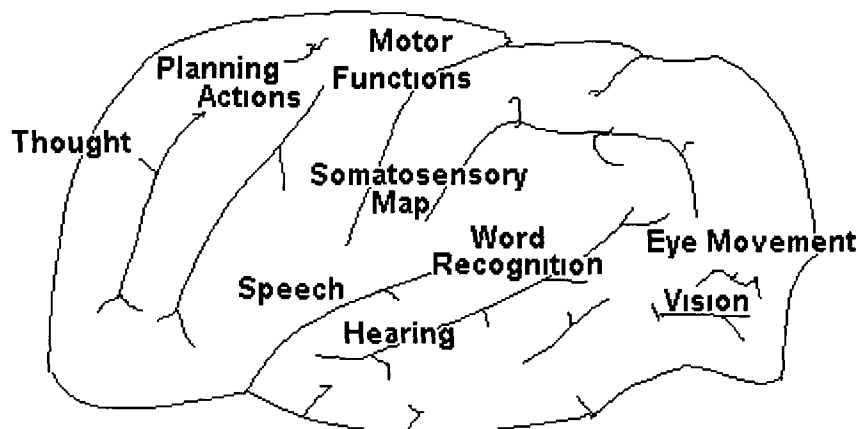
$$E = -\frac{1}{2} \sum_{i,j} w_{ij} \zeta_i \zeta_j \quad (25)$$

where  $\zeta_i$  represent all the possible states of the system

## 4.3.6 Kohonen Self-Organising Feature Map (SOFM)

### 4.3.6.1 Introduction

The cerebral cortex of the human brain is perhaps the most complex of all biological systems. On a micro level, it is organised into several groups of neurons of varying densities and types. On a macro level, it is organised into spatial regions by body function, as illustrated in Figure 4.13.



**Figure 4.13** Specialised Areas of the Cerebral Cortex

Each region consists of a large number of similar neurons that cooperate when carrying out the specific functions they have become specialised to handle. Each of the regions corresponds to a mapping from some functional group of sensory inputs such as the visual cortex, auditory or hearing receptors, motor functions, the somatosensory cortex (touch), thought, and so on. Groups of neuron cells within each region respond jointly to excitations from the sensory cells they service. For example, neurons in the visual cortex respond to certain light patterns falling on the retina. Somatosensory cortex region cells become excited by inputs from sensory cells beneath the skin and auditory or tonotopic map cells respond in localised groups to different sounds based on frequency or pitch. The receptive fields of these spatially organised neurons are associated directly with the sensory neurons. There is a mapping of the features from sensory neurons to the associated spatial regions of the cortex. This biological feature mapping of the brain has been modelled reasonably well with ANNs [Patterson].

The self-organising feature map ANN is a simplified model of the feature-to localised-region mapping of the brain from which it derives its name. It is a competitive, self-organising network which learns from the environment without supervision [Patterson]. In a *self-organising feature map* (SOFM), the neurons are placed at the nodes of a lattice that is usually one- or two-dimensional, higher-dimensional maps are also possible but not as common. The neurons become *selectively tuned* to various input patterns (vectors) or classes of input patterns in the course of a competitive learning process. The locations of the neurons so tuned (i.e., the winning neurons) tend to become ordered with respect to each other in such a way that a meaningful coordinate system for different input features is created over the lattice [Kohonen]. A self-organising feature map is therefore characterised by the formation of a topographic map of the input patterns, in which the *spacial locations (i.e., coordinated) of the neurons in the lattice correspond to intrinsic features of the input patterns*, hence the name "self-organising feature map" [Haykin].

SOFM networks are based on *competitive learning*, the output neurons of the network compete among themselves to be activated or fired, with the result that only one output neuron, or the neuron per group, is on at any one time. The output neurons that win the competition are called *winner-takes-all neurons*.

#### **4.3.6.2 The SOFM Algorithm**

The principal goal of the self-organising feature-mapping (SOFM) algorithm developed by Kohonen is to transform an incoming signal pattern of arbitrary dimension into a one- or two-dimensional discrete map, and to perform this transformation adaptively in a topologically ordered fashion. Many activation patterns are presented to the network, one at a time. Typically, each input presentation consists simply of a localised region, or 'spot', of activity against a quiet background. Each such presentation causes a corresponding localised group of neurons in the output layer of the network to be active.



The essential ingredients of the neural network embodied in such an algorithm are as follows

- A one- or two-dimensional lattice of neurons that computes simple discriminant functions of inputs received from an input of arbitrary dimension
- A mechanism that compares these discriminant functions and selects the neuron with the largest discriminant function value
- An interactive network that activates the selected neuron and its neighbours simultaneously
- An adaptive process that enables the activated neurons to increase their discriminant function values in relation to the input signals

The input vector of (sensory) signals to all the neurons in the lattice is represented by,

$$x = [x_1, x_2, \dots, x_p]^T \quad (26)$$

where  $p$  is the number of input terminals applied to each neuron of the network

The synaptic weight vector of neuron  $j$  is denoted by  $w_j$ .

$$w_j = [w_1, w_2, \dots, w_p]^T, \quad j = 1, 2, \dots, N \quad (27)$$

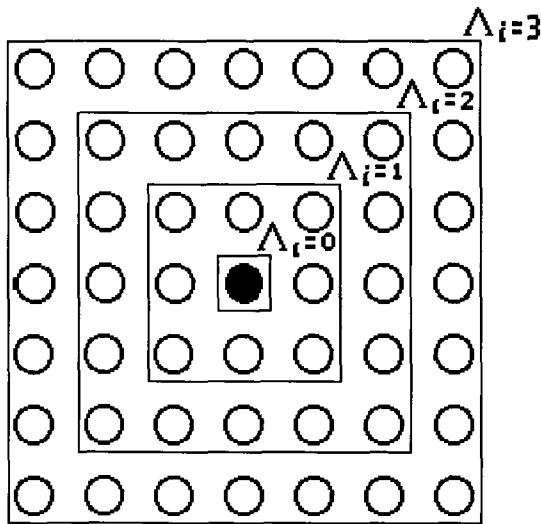
where  $N$  is the number of neurons in the network

To find the best match of the input vector  $x$  with the synaptic weight vector  $w_j$ , the *best-matching criterion* is used, which is described to be equivalent to the *minimum Euclidean distance* between the vectors. Specifically we use the index  $i(x)$  to identify the neuron that best matches the input vector  $x$

$$i(x) = \arg_j \min \|x - w_j\|, \quad j = 1, 2, \dots, N \quad (28)$$

The neuron  $i$  that satisfies this condition is called the *best-matching* or *winning neuron* for the input vector  $x$

The topology of iterations in the SOFM algorithm defines which neurons in the two-dimensional lattice are in fact neighbours. Let  $\Lambda_{i(x)}(n)$  denote the topological neighbourhood of winning neuron  $i(x)$ . Generally, the function  $\Lambda_{i(x)}(n)$  is taken to include neighbours in a square region around the winning neuron and the winning neuron itself, as shown in Figure 14



**Figure 4.14** Square Topological Neighbourhood, of Varying Size around Winning Neuron  $i$  (Black Circle)

The neighbourhood function  $\Lambda_{i(x)}(n)$ , is chosen to be a function of discrete time  $n$ . The neighbourhood function usually begins such that it includes all neurons in the network and then gradually shrinks with time. The shrinking of the neighbourhood function occurs during the *Ordering Phase* (first 1000 iterations) and is usually allowed to shrink to either the 0 or the 1 square region. The *Convergence Phase* (1000 to 10000 iterations) maintains the neighbourhood function in the 0 or the 1 square region.

The updating of the synaptic weights  $w_j$  is performed on all neighbours or the winning neuron  $i(x)$  and is proportional to the difference in the sensory input  $x$  and the weight  $w_j$  vectors, as described in Equations (29) and (30)

$$w_j(n+1) = w_j(n) + \alpha_{ij}(n) [x(n) - w_j(n)], \quad j \in \Lambda_{i(x)}(n) \quad (29)$$

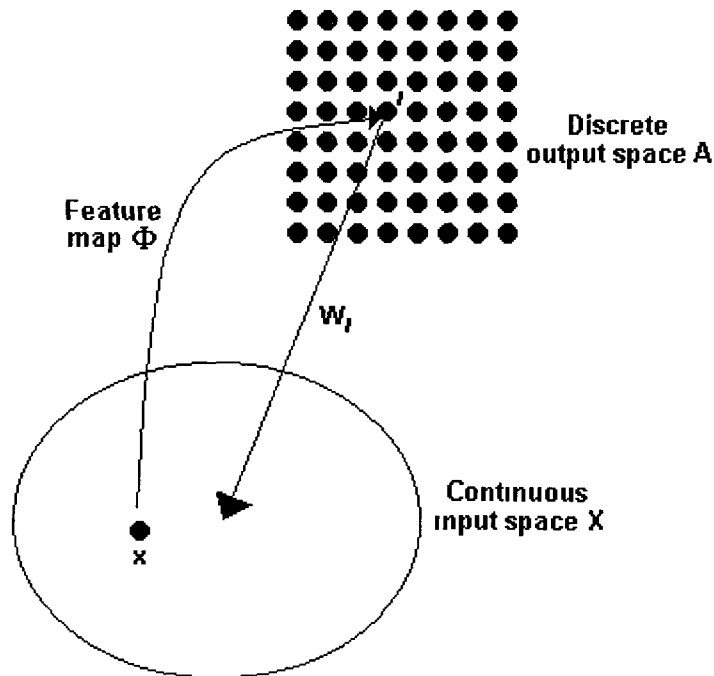
$$w_j(n+1) = w_j(n), \quad \text{otherwise} \quad (30)$$

Appendix C provides the actual algorithm used for the example described in Section 5.4.6

### 4.3.6.3 Properties of the SOFM Algorithm

#### Property 1: Approximation of the Input Space

The SOFM  $\Phi$ , represented by the set of synaptic weight vectors  $\{w_j \mid j = 1, 2, \dots, N\}$ , in the output space  $A$ , provides a good approximation to the input space  $X$ .



**Figure 4.15** Relationship Between the Feature Map  $\Phi$  and the Weight Vector  $w_i$  of Winning Neuron  $i$

The basic aim of the self-organising feature-mapping (SOFM) is to store a large set of input vectors  $x \in X$  by finding a smaller set of prototypes  $w_j \in A$ , so as to provide a ‘good’ approximation to the original input space  $X$ . The theoretical basis of the idea just described is rooted in *vector quantisation theory*, the motivation for which is dimensionality reduction or data compression. Figure 4.15 illustrates the relationship between the feature map  $\Phi$  and the weight vector  $w_i$  of winning neuron  $i$ .

### **Property 2: Topological Ordering**

**The feature map  $\Phi$ , computed by the SOFM algorithm is topologically ordered in the sense that the spatial location of a neuron in the lattice corresponds to a particular domain or feature of the input patterns.**

The topological ordering property is a direct consequence of the update equation that forces the synaptic weight vector  $w_i$  of the winning neuron  $i(x)$  to move toward the input vector  $x$ . It also has the effect of moving the synaptic weight vectors  $w_j$  of the closest neurons  $j$  along with the winning neuron  $i(x)$ . We may therefore visualise the feature map  $\Phi$  as an elastic net with the topology of a one- or two-dimensional lattice as prescribed in the output space  $A$ , and whose nodes have weights as coordinates in the input space  $X$ . The overall aim of the algorithm may thus be stated as follows:

*Approximate the input space  $X$  by pointers or prototypes in the form of synaptic weight vectors  $w_j$ , in such a way that the feature map  $\Phi$  provides a faithful representation of the important features that characterize the input vectors  $x \in X$ .*

### **Property 3: Density Matching**

**The feature map  $\Phi$  reflects variations in the statistics of the input distributions: regions in the input space  $X$  from which sample vectors  $x$  are drawn with a high probability of occurrence are mapped onto target domains of the output space  $A$ , and therefore with better resolution than regions in  $X$  from which sample vectors  $x$  are drawn with a low probability of occurrence.**

## 4.4 Conclusions

The rationale behind the choice of course material discussed in this chapter encompasses a number of relevant issues. Primarily, the ANN theory discussed has been incorporated into the lecture material developed by Dr John Ringwood [Ringwood]. Therefore, the Java demonstrations developed with the discussed ANN theory as their foundation, provide effective supplementary material for the MEng program.

Additional reasoning behind the choice of the discussed ANN theory lies in the nature of ANNs. ANN theory consistently incorporates conceptually challenging mathematical and spatial reasoning and consequently may be conveyed more intelligibly through additional graphs and interactive simulations. The diversity of the ANN architectures described in this chapter, can all be effectively described with the aid of versatile demonstrations that allow the student to experiment with the network's configuration, training data and parameters. In addition, the majority of the ANNs require computationally demanding training and testing procedures in order to effectively convey their functionality.

# Chapter 5: ANN Java Demonstrations

## Development

### 5.1 Introduction

This chapter is used to describe how the ANNs courseware is presented to students via the Web. This comprises the tools used to transform the lecture material into HTML Web documents and also the development procedure of the Java demonstrations. The *generic tools* are described in detail in order to convey the development process behind designing the ANN demonstrations. This chapter's main focus is in discussing the operation and presentation of the demonstrations.

### 5.2 ANN Web Site Development

#### 5.2.1 Text Converters

The lecture material developed by Dr. John Ringwood was created using the *Latex* word processor [Ringwood, Lamport]. In order to convert the Latex documents into HTML, it was necessary to use a special Latex to HTML converter, aptly named *Latex2HTML*. The Latex2HTML converter was quite successful in converting the textual code into HTML, however its shortfall was in converting Latex's built-in diagrams and equations into *gif* files (special graphic files). As a result, the *postscript* versions of the lecture material were used to obtain the correct diagrams and equations and convert them into gifs with the aid of *Microsoft Paint*. The gif files are then easily embedding into a HTML web document with the aid of a *HTML editor*.

#### 5.2.2 HTML Editor

The HTML editor used to create the Web courseware was *Netscape's Navigator Gold version 3.0*. The editor facilitated the expeditious generation of HTML documents. It has the facilities to alter text styles, embed pictures, tables, hypertext links, and even

*Java Applets* It therefore eliminates the need to learn a mark-up language and allows the developer to observe the document during its construction

## 5.3 Generic Tools

*Generic Tools* describes the Java classes that were re-used throughout all of the Java demonstrations These classes include the *Matrix class*, the *Graph class* and the *neuron class* As discussed in Section 3.3.2.2, Java is object-oriented and as a result the code is reusable This means that once a class has been developed and compiled it can then be used and re-used by any other class that requires its functionality

### 5.3.1 The Matrix Class

The *Matrix class* was developed in order to allow the Java demonstrations to make use of matrix algebra comparable to that of [MATLAB] Matrix algebra is seen throughout ANN theory as can be seen from Sections 4.3 and 4.4 Therefore, the development of a Matrix class was a natural starting point from which to develop ANN code

There are three data fields associated with the Matrix class the number of rows, the number of columns and the data array containing the matrix elements The Matrix class operation is determined by its constructors or initialisation methods and the methods which affect the data fields

#### 5.3.1.1 Initialisation

When a matrix object is created it is possible to initialise its data fields in a variety of ways, depending on its application The matrix object can be initialised as an exact copy of another matrix object or it can simply be initialised with the number of rows and columns, with all its elements set to zero as a default Alternatively, the matrix may be initialised with its rows, columns and data elements all set In addition, a particular type of matrix can be created, i.e., an identity matrix or a matrix of all zeros or ones

### 5.3.1.2 Functionality

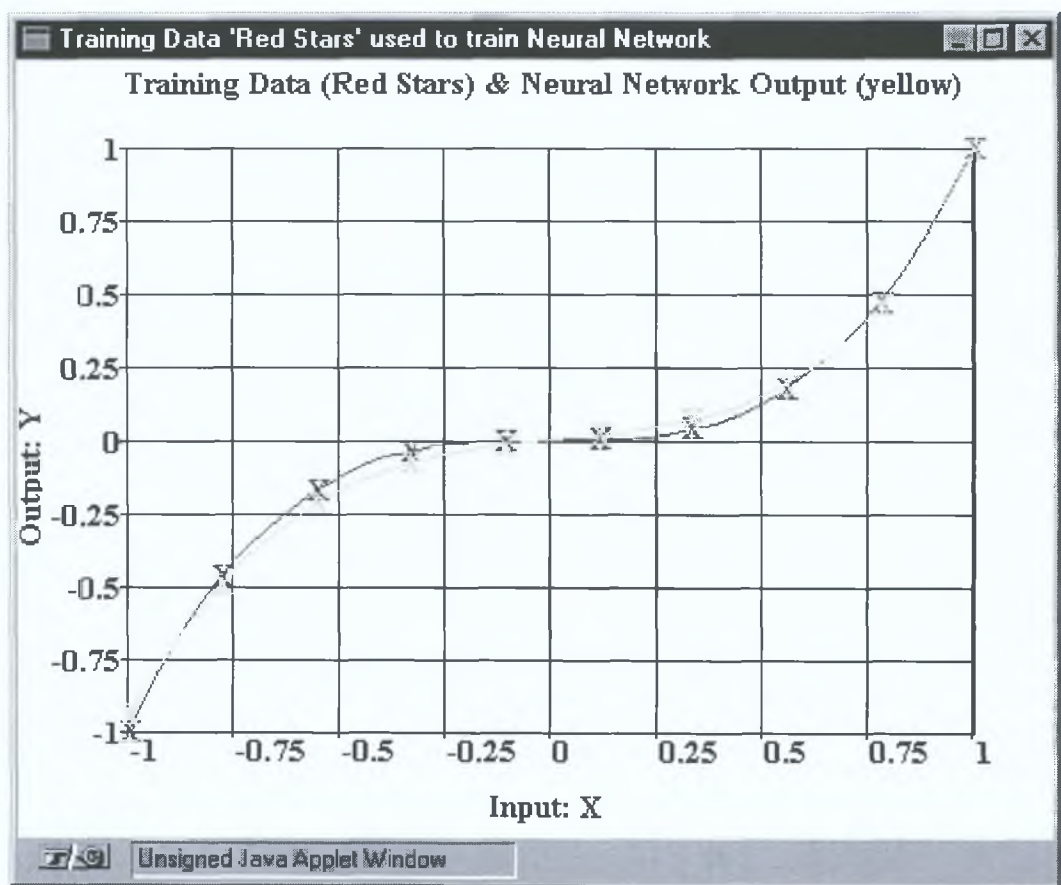
The *Matrix* class was developed to be no greater than two-dimensional. Some of the main unitary functions or methods associated with the *Matrix* class include obtaining the transpose, the minimum, maximum and absolute values of the matrix elements, the index of a particular element, performing row and column swapping, gaussian elimination, the inverse operation and retrieving a subsection of a matrix. The main binary methods include addition, subtraction, pre- and post-multiplication, dot product and augmenting one matrix with another. Of course, it is also possible to alter the values of matrix elements by simply providing the matrix with the new data elements.

### 5.3.2 The Graph Class

The *Graph* class was developed to display data in a two-dimensional graph window that floats above that of the application that uses the class. Figure 5.1 shows the appearance of a plot produced using the *Graph* class. As can be seen from Figure 5.1, the graph contains numerically labelled axes, X and Y Labels and a Title. A graph is usually initialised with only the name to appear across the title-bar. The default name is 'Graph Window'.

The data fields of the *Graph* class that can be set during an application's run-time comprise the *Grid* and *Auto-Scale* fields. The former determines whether or not a grid appears over the plot, as is the case in Figure 5.1. The latter determines whether or not the graphing methods should control the ranges of the X and Y axes when new data is submitted for plotting. They are both binary valued fields which means they have either a *true* or *false* value.





**Figure 5.1** Plot Generated using the Graph Class

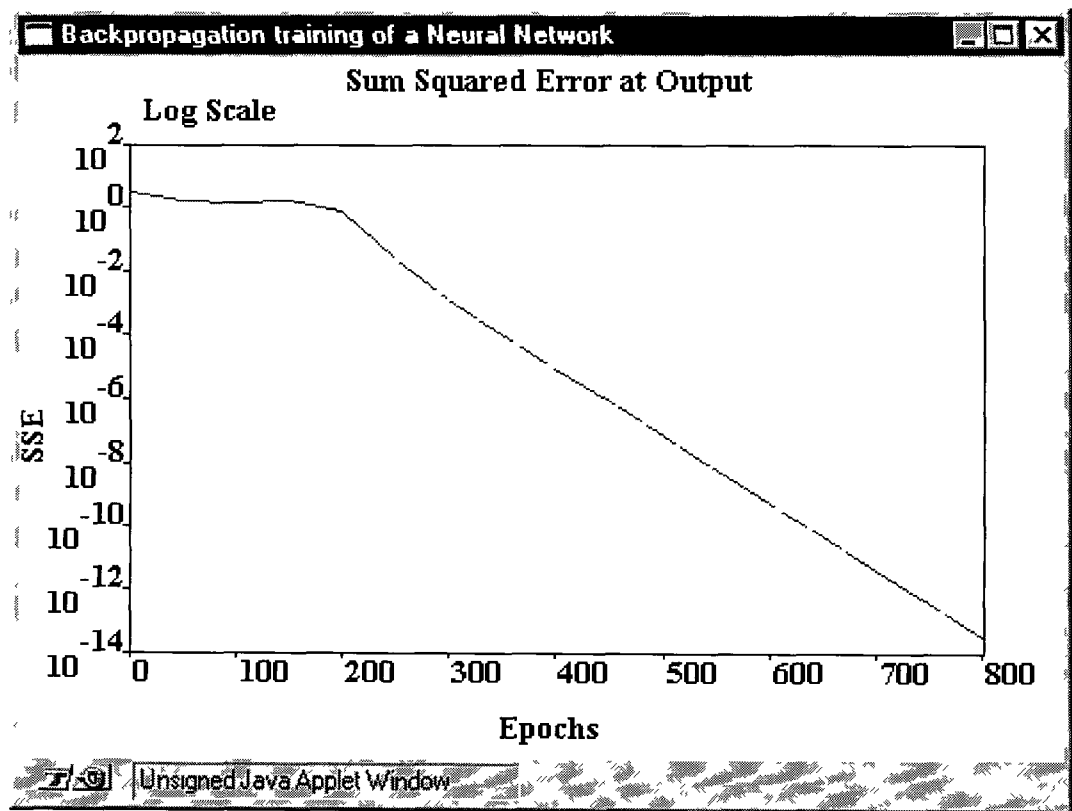
### 5.3.2.1 Functionality

Some of the more basic functions are used to set whether the plot is visible or not. This can be useful when a graph object is created but is not yet required to be visible by the user. Another function is used to set the background colour of the graph, which is purely aesthetic and is generally left at its default black setting. An important set of functions are used to set the *X* and *Y* labels, *title* and *name* of the graph. Note the title of the plot is that which appears directly above the graph itself, whereas the name refers to the text which appears in the bar across the top of the window.

When data is presented to the graph, there are *three* important functions that control the nature of its presentations. The first controls the *colour* and *type*. The range of colours is limited only by the host machine's display. The type refers to whether the data is to be plotted with connecting lines, dots or characters such as the X's shown in Figure 5.1. The second is the actual *plot* command through which data is passed to the

graph object. Data can be presented either as individual points or as arrays of data. Data can also be presented in the form of text, which can be used to identify particular points within a graph. The third set of functions determine the scaling of the X and Y axes, which can be set to either *log*, *decibels*, or the default *normal* scale. Figure 5.2 uses a log scale on the Y-axis to show sum-squared error (SSE) dropping off exponentially with time (epochs). The adjustable scale facility is extremely useful for presenting data which would otherwise be difficult to interpret.

Another important facility provided by the Graph class is being able to adjust the range of axes for which data is presented. In order to do this, the *Grid* field must first be toggled to false, then one of two functions may be applied. The *axes* function allows the application to determine exactly the range of the X and Y axes, i.e., 0→800 and  $10^{-14}$ → $10^2$  respectively in Figure 5.2. The *axesFit* function automatically sets the range so that all data is perfectly included in the plot, as can be seen in Figure 5.1.



**Figure 5.2** Sum Squared Error of the Back-Propagation Algorithm

### 5.3.2.2 User Controls

The *User Controls* refer to how a person experimenting with the Java demonstrations can alter their presentation. With regard to the graphing window, the user can *minimise*, *maximise* or *resize* the plot to suit their needs. In addition, the user can *zoom* in or out of a graph by drawing a box (i.e., by dragging mouse with the left button pressed) over the area of the graph for which closer investigation is required. The graph is then magnified in the selected graph region to show a more detailed representation of the data within. This can be performed multiple times, if so desired by the user. The original view of the plot is then recovered by simply clicking the mouse button once over the graph window.

### 5.3.3 The Neuron Class

The *neuron class* is used throughout the Java demonstrations. It is used as a building block for generating the diverse network architectures associated with the field of ANNs. Networks as structurally different as those of MLPs and SOFMs can all be constructed with a set of neuron objects specifically tailored for their particular network. The neuron class creates a neat package which controls the configuration and operation of any neuron within a network.

The neuron class is an extension of the *Squash* class which looks after all the methods associated with the neuron's *activation* or *squashing function*. The neuron class itself contains a number of data fields which are vital to a network's architecture. Each neuron has its own personal identification number or *ID*, which enables other neuron objects in the same network to identify a particular neuron and access its data. In addition, each neuron has a *type*, which classifies it as an input, output or hidden neuron. A neuron object also has a *bias*, and *weight matrix* which determines what proportion of the outputs of the connected neurons contributes to the *adder* or *linear combiner* shown in Figure 4.3. The weight matrix is an object of the *Matrix* class described in Section 5.3.1. A *connections array* field contains the *ID*'s of the neurons connected by the weight matrix. Finally, there is an *output* field which stores the final output of the neuron.

### **5.3.3.1 Initialisation**

By default a neuron object is initialised with a zero bias, no weight connections and a sigmoidal squashing function. A neuron can be initialised as a copy of another previously generated neuron object except for its ID which is always unique. Alternatively, a neuron can be created with all its data fields set by the application on construction.

### **5.3.3.2 Functionality**

There are a number of functionally basic methods which allow the data fields of the neuron to be accessed and set subsequent to initialisation. Apart from those methods, all that remains is the method which performs the operation of the adder and the method to obtain the output of the neuron using the squashing function method of the Squash class.

5.4 The ANN Demonstrations

5.4.1 The Hebbian Learning Demonstration

5.4.1.1 Introduction

The *Hebbian Learning Demonstrations* is located at the end of the Web document containing the Hebbian learning theory [ANNs Web Site] Figure 5 3 shows the Java demonstration as would be seen in Netscape’s browser [Netscape] The demonstration consists of a single neuron object generated from the *neuron class* that uses the Hebbian Learning rules described in Section 4 3 2 2

The objective of the Hebbian learning Demonstration is to interactively demonstrate the effect of Hebbian learning on the weights of a neuron It allows the students to alter the learning rate and the values of the neuron’s inputs, weights and squashing function such that they can see exactly how each affects the learning process

Hebbian Learning Demonstration

|                                       |           |                 |                 |                  |
|---------------------------------------|-----------|-----------------|-----------------|------------------|
| Input x0                              | Weight w0 |                 |                 |                  |
| 1 0                                   | 3 0       |                 |                 |                  |
| Input x1                              | Weight w1 |                 |                 |                  |
| 2 0                                   | 1 0       | Sum (z)         | Squash function | Output (y)       |
| Input x2                              | Weight w2 | 1 000           | Hard Limit      | 1 000            |
| 0 0                                   | 0 0       |                 |                 |                  |
| Input x3                              | Weight w3 |                 |                 |                  |
| 0 0                                   | 0 0       |                 |                 |                  |
| <div>Evaluate Neuron's Response</div> |           | Learning Rate = | 0 10            | <div>Learn</div> |

Figure 5.3 Hebbian Learning Demonstration

5.4.1.2 Hebbian Learning Demonstration Operation

The Inputs and Weights can all be adjusted by the user, along with a choice of squashing function and the *Learning Rate*. The student alters the values of the inputs, weights and learning rate by editing the values contained within the associated *text-field(s)*. The squashing function may be selected from the *list-box* labelled *Squash function*.

Once the desired adjustments have been made, the user then evaluates the response of the neuron by pressing the *Evaluate Neuron's Response* button. This has the effect of altering the *Sum (z)* and the *Output (y)* labels. The user can then force the neuron to learn by pressing the *Learn* button, which affects the weights according to Chapter 4's Equations (1) and (2).

5.4.2 The Perceptron Learning Demonstration

5.4.2.1 Introduction

The *Perceptron Learning Demonstrations* is located at the end of the Web document containing the Perceptron learning theory [ANNs Web Site]. Figure 5.4 shows the Java demonstration as would be seen in Netscape's browser [Netscape]. The demonstration consists of a single neuron again generated from the *neuron class*, that uses the Perceptron learning rules described in Section 4.3.2.3.

Perceptron Learning Demonstration

|          |           |      |            |       |                    |
|----------|-----------|------|------------|-------|--------------------|
| Input x0 | Weight w0 |      |            |       |                    |
| 1.0      | 3.2000    |      |            |       |                    |
| Input x1 | Weight w1 |      |            |       |                    |
| 2.0      | 1.6000    |      |            |       |                    |
| Input x2 | Weight w2 | 0.00 | Hard Limit | -1.00 | Desired Output (d) |
| 0.0      | 0.0000    |      |            |       | 1                  |
| Input x3 | Weight w3 |      |            |       |                    |
| 0.0      | 0.0000    |      |            |       |                    |

Evaluate Neuron's Response | Learning Rate = 0.10 | Learn

Figure 5.4 Perceptron Learning Demonstration

### 5.4.2.2 Perceptron Learning Demonstration Operation

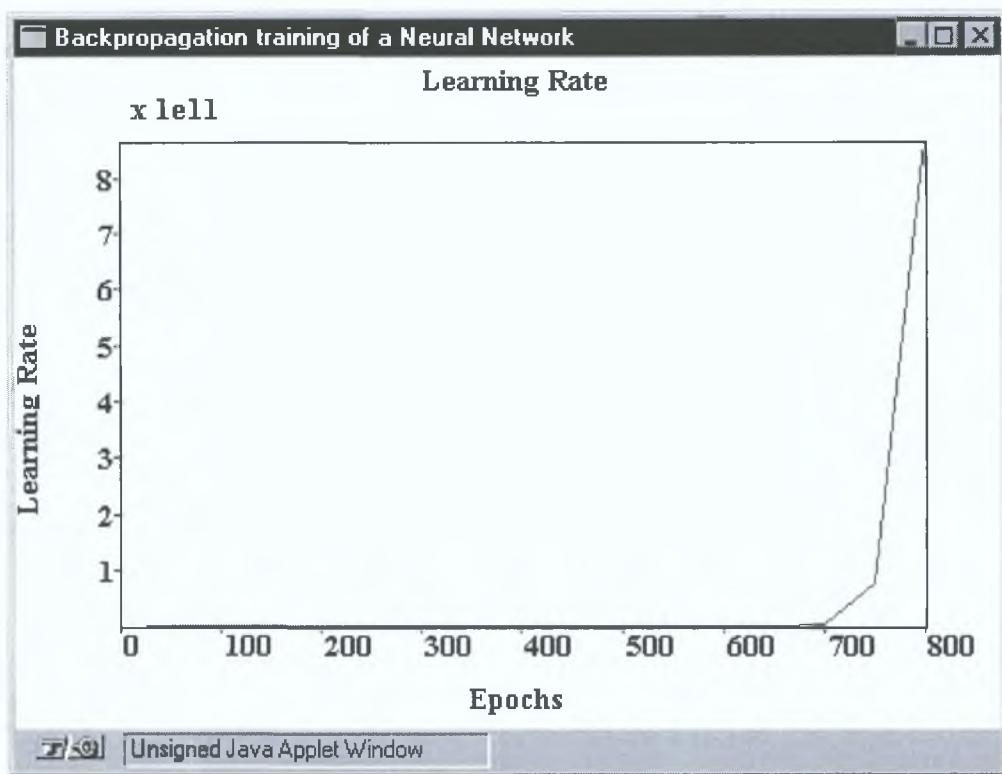
Consistent with the Hebbian learning demonstration, the *inputs* and *weights* can all be adjusted by the user, along with a choice of *squashing function* and the *learning rate*. The difference here is choice of the *desired output* which is a fundamental element in the perceptron learning rule. The user can choose a value of 1 or -1 from the *Desired Output (d)* list-box. Again, the user evaluates the response of the neuron by pressing the *Evaluate Neuron's Response* button. The user can then force the neuron to learn by pressing the *Learn* button, which affects the weights according to Equations (4) through (6).

## 5.4.3 The Multi-Layered Perceptron Demonstrations

### 5.4.3.1 Introduction

Both the *Logic* and the *Polynomial Function Approximation Demonstrations* provide examples of how the *Back-Propagation Algorithm*, described in Section 4.3.3, can be used to train a Multi-Layer Perceptron (MLP) network. The demonstrations have been designed such that it is possible for the student to manipulate as many parameters as possible that are relevant to the particular demonstration.

The objective of the demonstrations is to provide students with an insight into the performance of a MLP network and the success of its training procedure. The demonstrations also give students an idea of what kind of applications can be catered for by a MLP network.



**Figure 5.5** Learning Rate of the Back-Propagation Algorithm

#### 5.4.3.2 General Operation

In each of the demonstrations, the student has control over the MLP network or *Layer Settings* and the *Training Settings* that are used by the Back-Propagation Algorithm as described in Section 4.3.3.3. The Back-Propagation Algorithm described in Appendix A, is all performed by a *Back-Propagation class*, which was specifically designed for the MLP demonstrations. As the network is training, two graphs show the progress of the training algorithm in terms of the *sum-squared error (sse)* and the *learning rate*, as shown in Figures 5.2 and 5.5 respectively.

By changing the *Layer Settings* shown in Figures 5.6 and 5.7, the network's architecture can be adjusted to meet the students requirements. The first of the settings is the *Number of Inputs* associated with the MLP network. In the case of the logic function approximation demonstration, this is adjustable. In the case of the polynomial function approximation demonstration, it is fixed at one. The next setting is that of the *Number of Layers*, which includes both the hidden layers and the output layer. The student has the option of a choosing between one layer and three layers. If the number of layers is two or three then the *Number of Cells* associated with the respective one or two hidden layers must then be selected. In each of the hidden

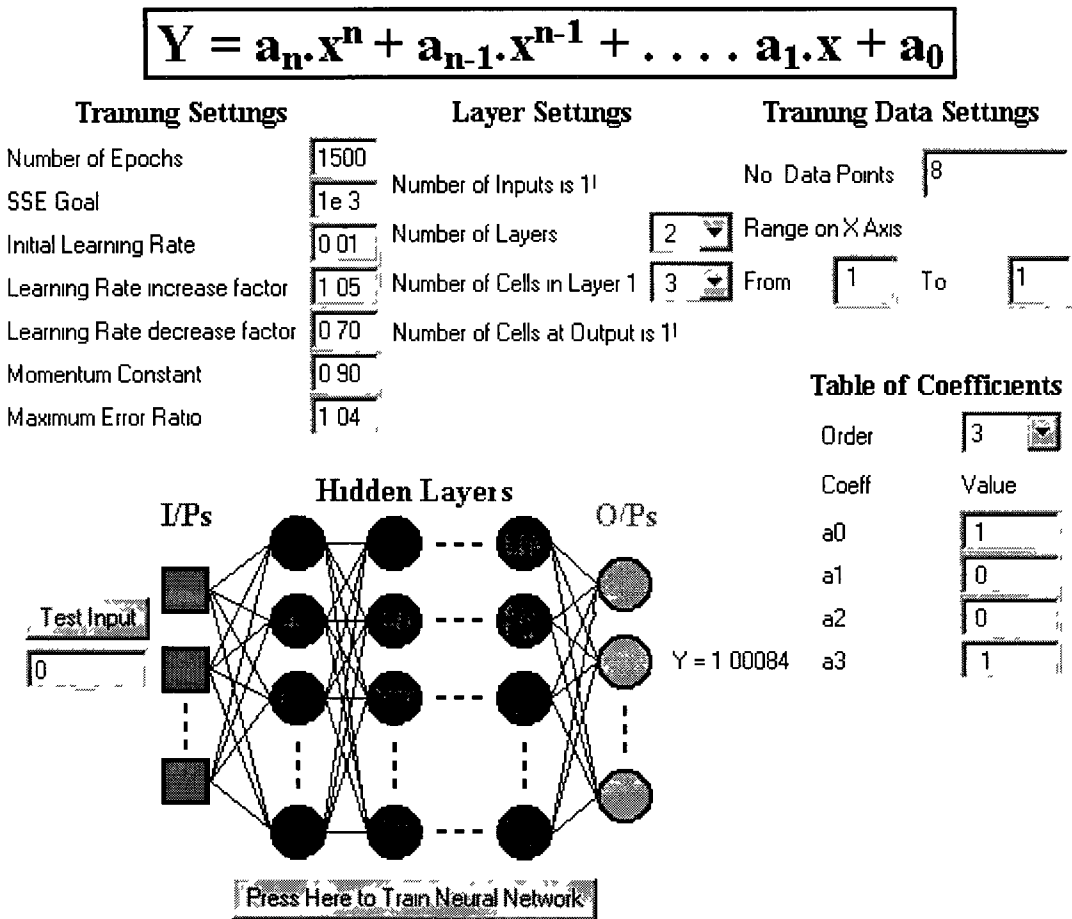


layers, a maximum of sixteen cells (neurons) can be selected. Finally, in both demonstrations the number of cells at the output of the networks is fixed at one, which is stated at the bottom of the *Layer Settings*

#### **5.4.3.3 MLP Logic Function Approximation Demonstration**

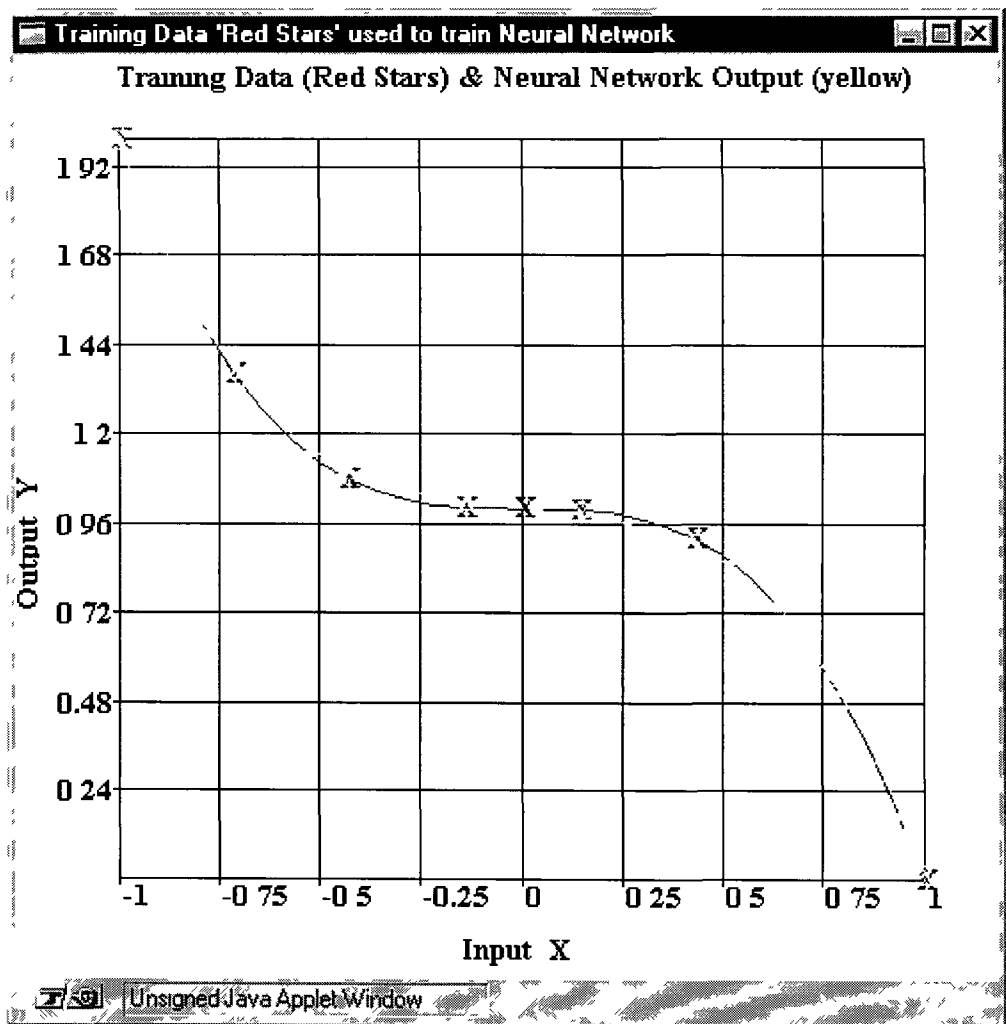
In this demonstration the MLP is being trained to implement a 1→3 input, 1 output logic function described by the *Truth Table* shown in Figure 5.6. The *Number of Inputs* parameter shown in the *Layer Settings*, can be changed between one and three depending on the desired logic function. The logic function is determined by the Truth Table's *check boxes*, which can be toggled by the user with the aid of the mouse. The next step for the student is to invoke the *Press Here to Train Neural Network* button. At this point, the student is presented with graphs of the SSE and the learning rate, as discussed in Section 5.4.3.2. The student can then experiment with other MLP architectures, training settings or logic functions.





**Figure 5.7** MLP Logic Function Approximation Demonstration

The next step for the student is to invoke the *Press Here to Train Neural Network* button. At which point, the student is presented with graphs of the SSE and the learning rate. After the network has been trained and tested to see how well it approximates the training data, also shown in Figure 5.8. The student can test the response of the network at particular points on the X-axis by inputting the X-coordinate into the text field below the *Test Input* button (see Figure 5.7) and pressing the button. The network's response is displayed both on the right hand side of the MLP diagram within Figure 5.7 (in this case  $Y=1.00084$ ) and also as an X at the centre of the graph shown in Figure 5.8.



**Figure 5.8** Graph of Polynomial Functions

5.4.4 Radial Basis Function Network Demonstration

The *Radial-Basis Function (RBF) Network Demonstration* shown in Figure 5 9, is also used as a single input, single output polynomial function approximator. The demonstration uses the formulae described in Section 4 3 4 3 to train the RBF network. As with the other demonstrations, the RBF demonstration has been designed to be as versatile as possible with regard to student interaction. The student interaction is apparent in five major steps of the demonstration's operation:

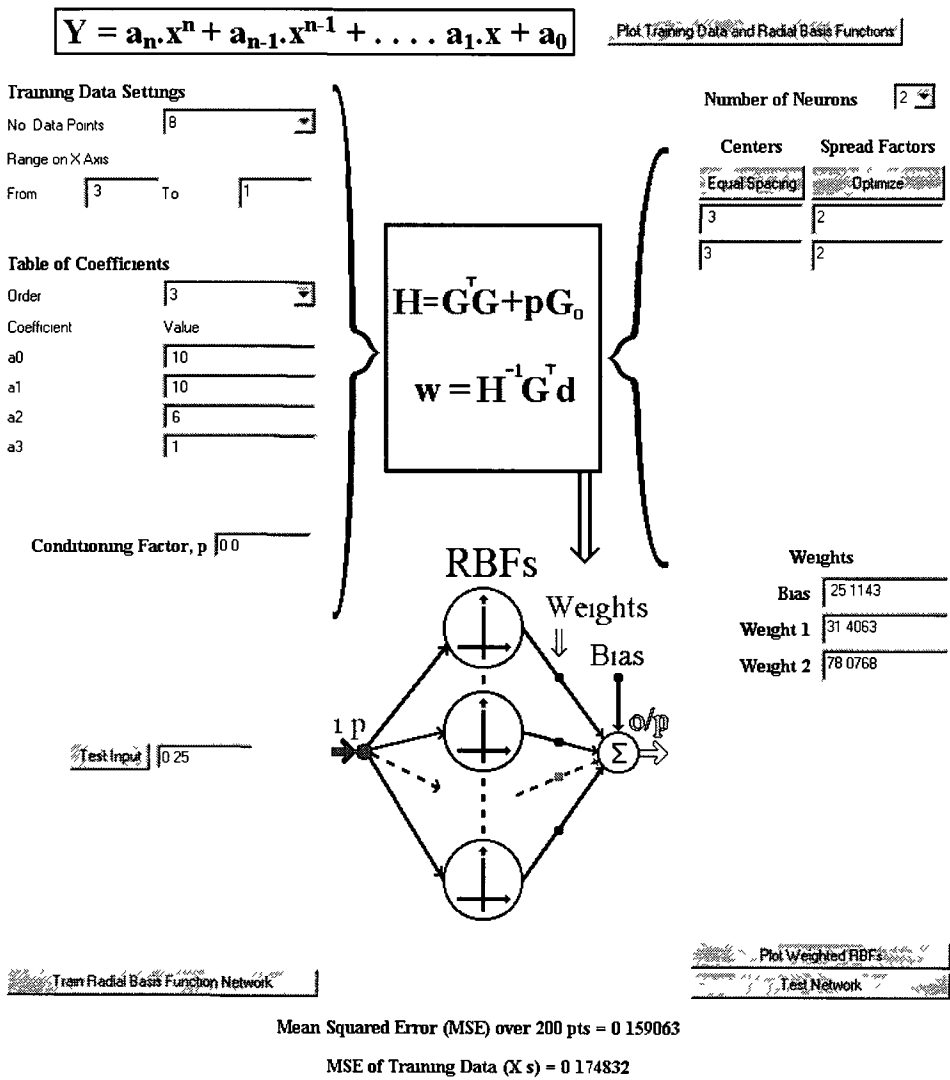
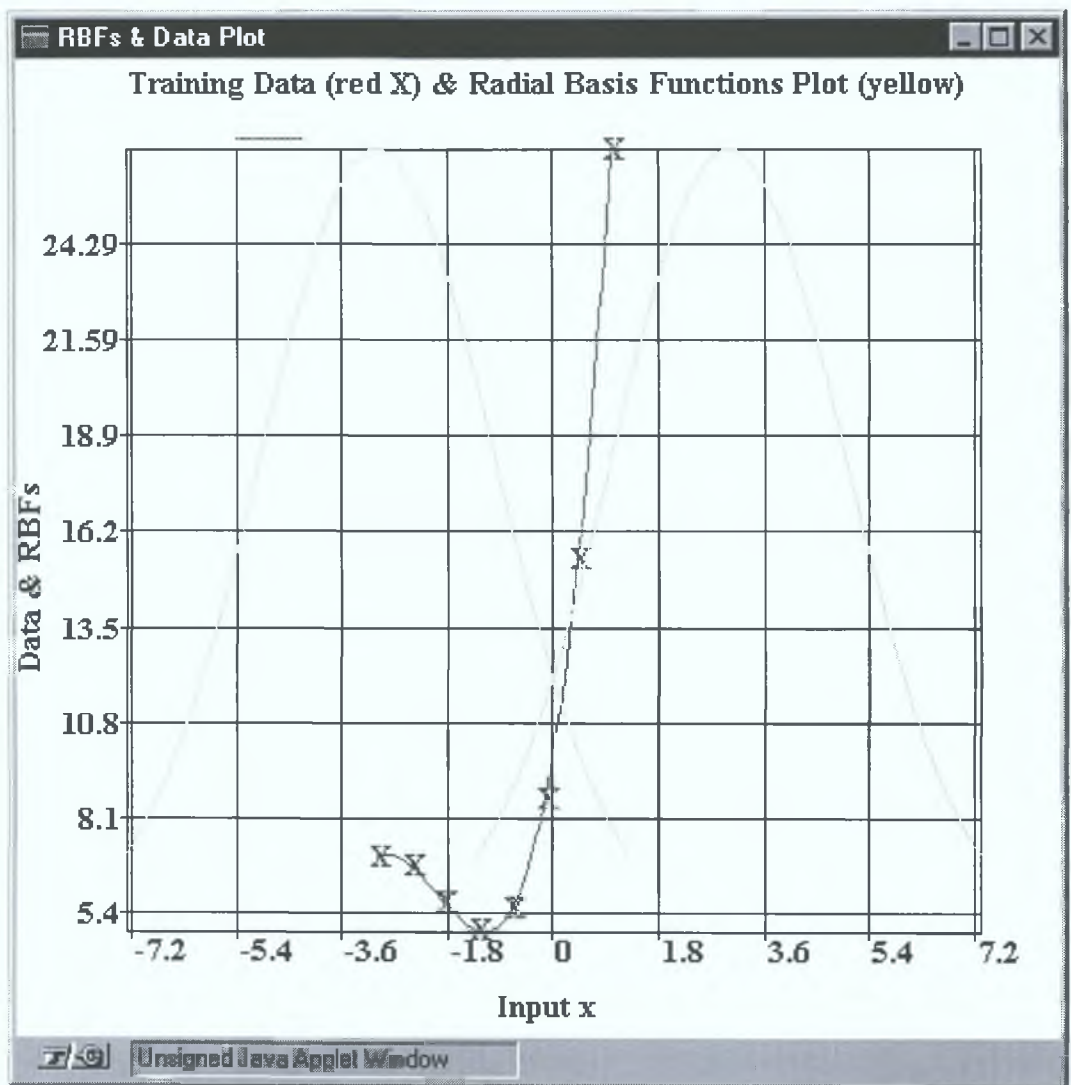


Figure 5.9 Graph of Polynomial Functions

#### 5.4.4.1 RBF Network Demonstration Operation

The first step in the procedure involves the *Training Data Settings* shown on the top left of Figure 5.9. This stage is similar to that of the MLP polynomial function approximator. The *Number of Data Points* ( $N$  from Section 4.3.4.3) determines the number of data points for which the RBF network will be trained. The *Range on the X-Axis* determines the domain of training data. The *Table of Coefficients* is then used to determine the polynomial as described in the formula at the top of Figure 5.9, from which the training data is taken.

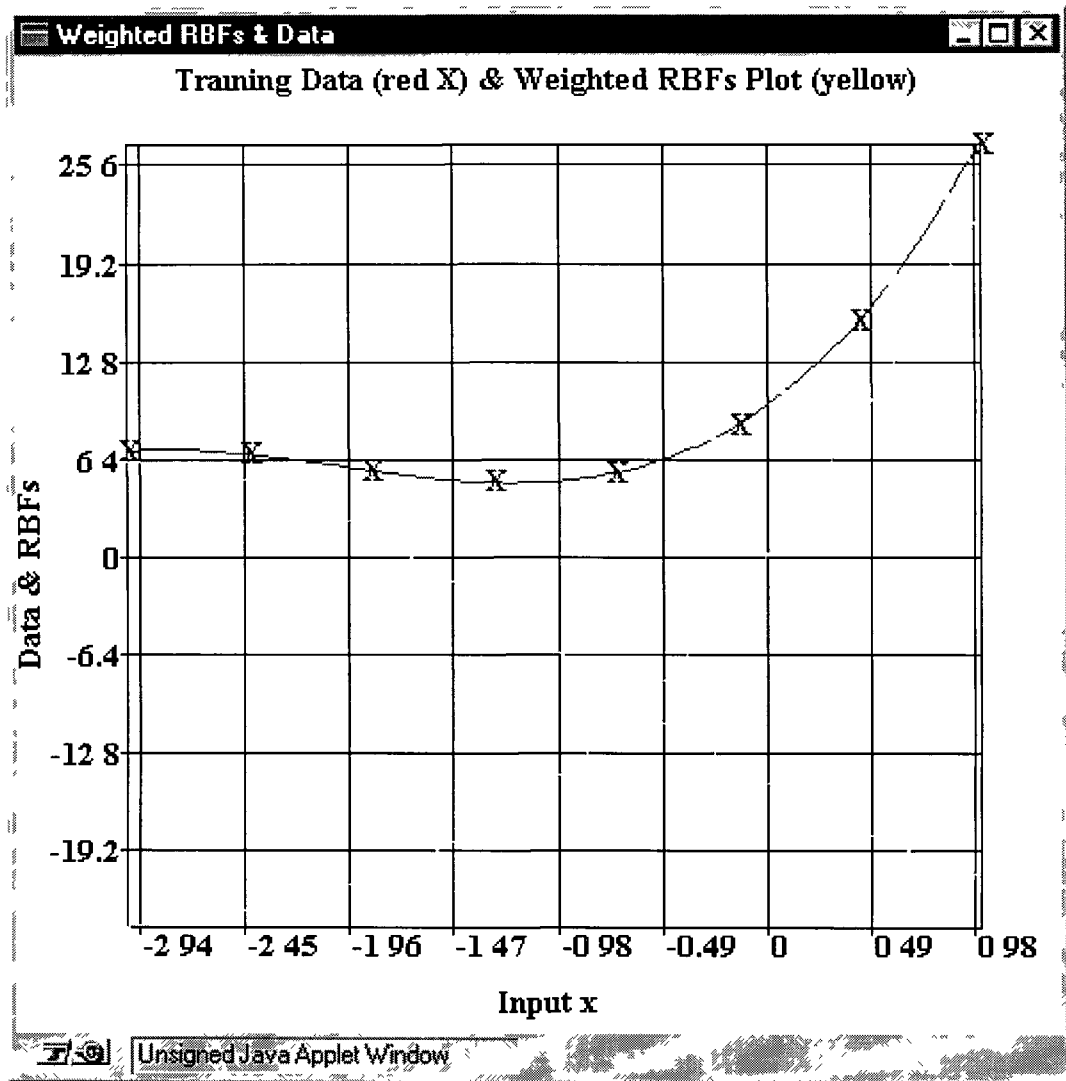
The second step requires the user to set the *Number of Neurons* or RBFs ( $M$ ) that will be used in the network. As described in Section 4.3.4.3, the number of neurons ( $M$ ) must be less than or equal to the number of data points ( $N$ ). Once  $M$  has been determined, the user must then select an appropriate *centre* and *spread factor* for each of the  $M$  neurons. The user has the option here to either manually set these parameters or use the *Equal Spacing* and/or *Optimise* buttons. The equal spacing button automatically spaces the centres between limits of the *Range on the X-Axis* variables. The optimise button uses Chapter 4's Equation (22) to set all of the spread factors.



**Figure 5.10** Training Data and RBFs Plot

Step three simple allows the user to view the graph of the Training Data and RBFs by pressing the *Plot Training Data and Radial-Basis Functions* button, located on the upper right hand side of Figure 5.9. Figure 5.10 shows an example of training data and RBFs that might be seen by the user. This graph can be accessed by the user at any time during the training procedure.

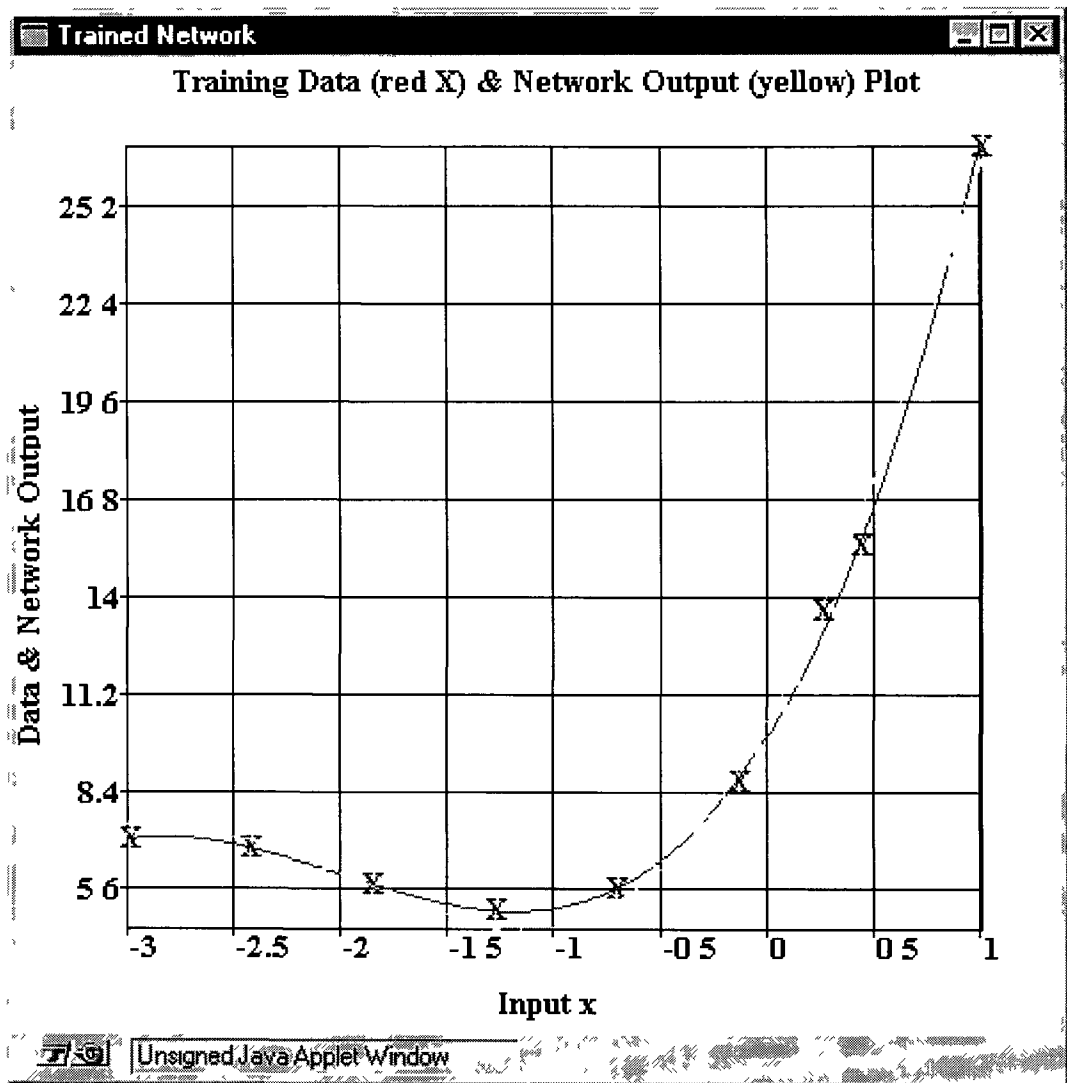
In step four of the training procedure, the user has two options: to allow the demonstration to automatically determine the *weight matrix* by using Chapter 4's Equation (17) (also shown in central diagram of Figure 5.9), or to manually set the weight matrix values using the text-fields provided on the lower right hand side of Figure 5.9. The former allows the user to determine how accurately Equation (17) configures the RBF network, whereas the latter encourages the user to experiment with the weight matrix and hopefully gain a greater insight into its operation.



**Figure 5.11** Training Data and Weighted RBFs Plot

Step five allows the user to view their weighted RBFs and test the network. Figure 5.11 shows an example of what might be seen if the *Plot Weighted RBFs* button was pressed. This graph allows users to conceptualise how the individual RBFs can be summed to provide a reasonable approximation for the training data. If so desired, the centres and the spreads could now be manually modified at this point. Figure 5.12 shows a plot generated after pressing the *Test Network* button. This graph shows how accurately the RBF network models the training data. Numerically, this may be determined by viewing the *Mean-Squared Error* (MSE) for the actual training data and for two hundred points even spaced between the training data (see bottom of Figure 5.9).





**Figure 5.12** Plot of Training Data and RBF Network Model

Note that if the computer generated weight matrix does not accurately model the training data, then the user can set the *Conditioning Factor*  $\rho$  as described in Chapter 4's Equation (17), to compensate form any ill-conditioning of the  $H$  matrix (central diagram of Figure 5 9)

### 5.4.5 Hopfield Network Demonstration

The *Hopfield Network Demonstration* shown in Figure 5 13, is used as a *content addressable memory* as described in Section 4 3 5 1 The algorithm provided by Appendix B is used in the training and testing of the network The Hopfield network used in this demonstration consists of  $N=120$  neurons It can be trained to retrieve any pattern that the user desires to store in its weights However, it has been shown that the best performance is achieved when the *predefined patterns* (described in section 5 4 5 1) are stored in its weights

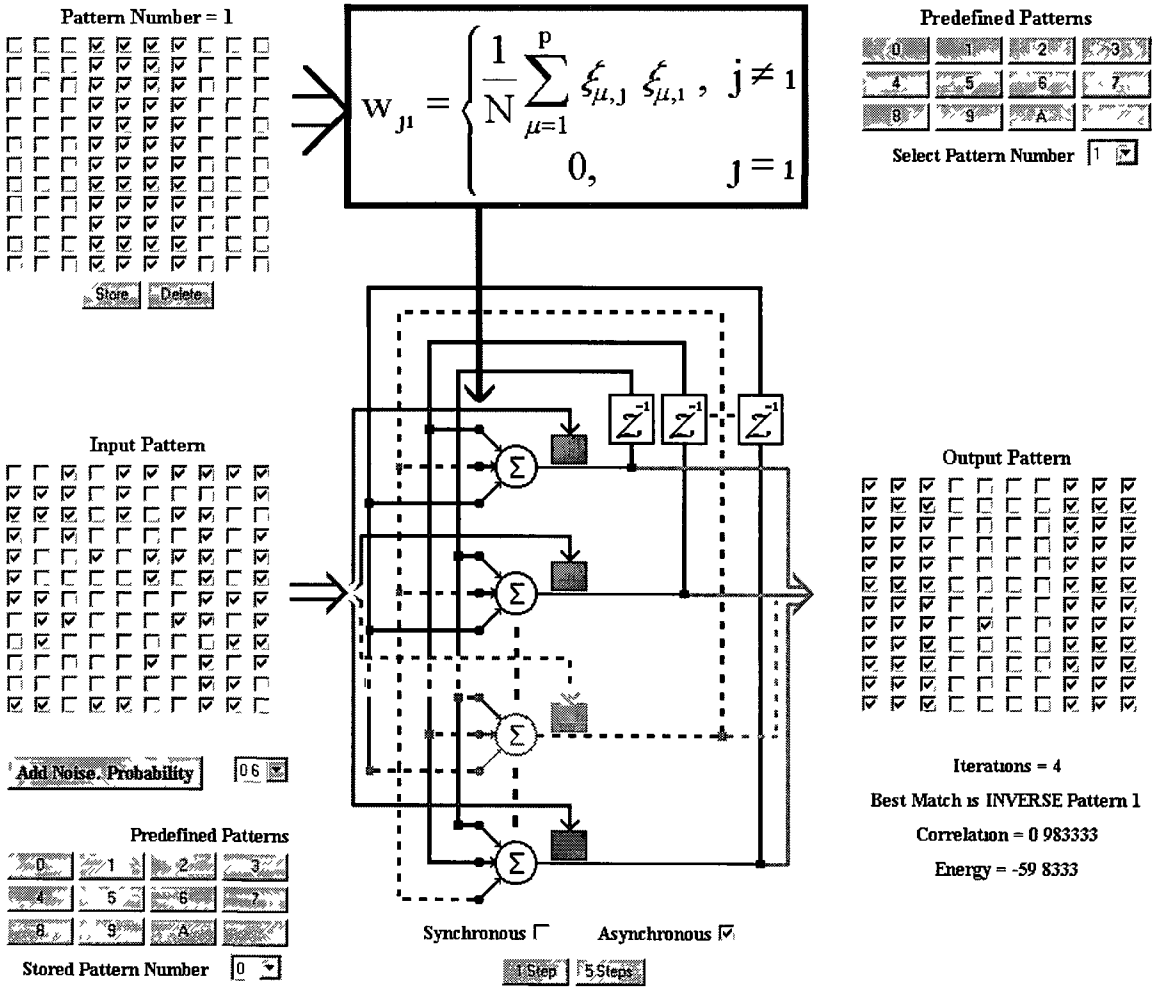


Figure 5.13 Hopfield Network Demonstration

#### 5.4.5.1 Hopfield Network Demonstration Operation

The Hopfield Network Demonstration is best described by dividing it into two distinct sections storing of patterns and testing the network The *Pattern Storage* (upper portion of Figure 5 13) is responsible for determining the weights as described by the diagram at the top of Figure 5 13, whereas the *Network Testing* (lower portion of

Figure 5 13) allows the user to evaluate the effectiveness of the network's pattern retrieval

#### 5.4.5.1.1 Pattern Storage

It is possible to store up to sixteen patterns in the network's weights, each having its own *Pattern Number* as shown above the pixels (check-boxes) at the top left hand side of Figure 5 13. Once the *Pattern Number* is selected from the *Select Pattern Number* list-box, the user can set the pattern to be stored in three ways: (1) by selecting a predefined pattern, (2) by using the mouse to toggle the pattern's pixels, (3) by using a combination of 1 and 2. Once the user has created the appropriate pattern, they can then use the *Store* button to have their pattern affect the network's weights as described by the equation shown within the diagram of Figure 5 13. Alternatively, if the user no longer requires a particular pattern to be stored in the network, they can select the appropriate *Pattern Number* and press the *Delete* button.

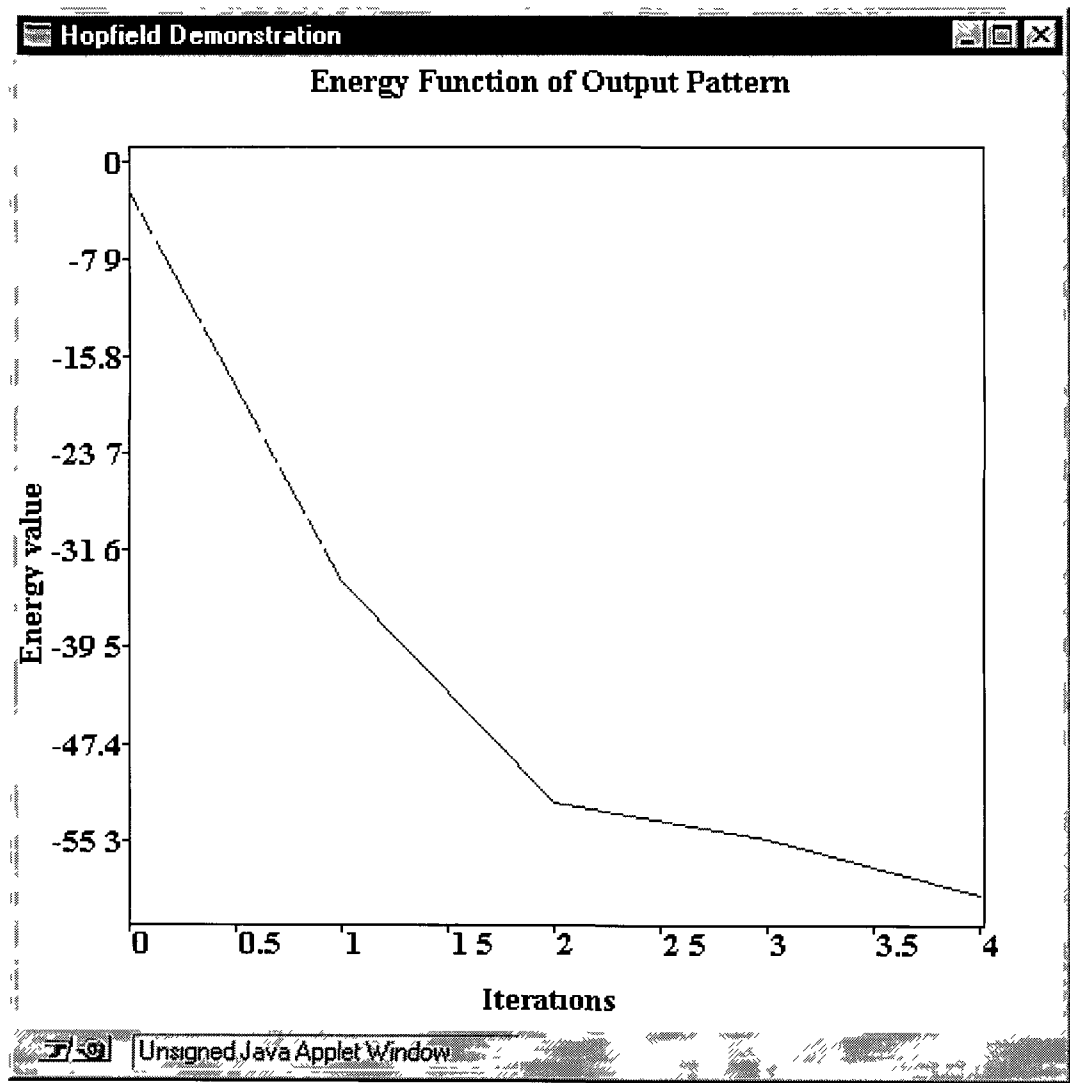
#### 5.4.5.1.2 Network Testing

After the network's weights have been set as described by Section 5 4 5 1 1, the user can then test the network's operation. This involves initialising the states of the network with the *Input Pattern* as shown in the diagram of Figure 5 13, and then iterating the network using the update rule described in Step 3 of Appendix B.

The *Input Pattern* can be set in a number of ways: (1) by selecting a predefined pattern from the *Predefined Pattern* buttons, (2) by selecting a stored pattern determined by Section 5 4 5 1 1, (3) by toggling the pixels of the *Input Pattern*, (4) by adding an amount of noise determined by the probability list-box, and evoked by the *Add Noise Probability* button, (5) by using a combination of 1 through 4.

Once the *Input Pattern* has been set, the next step is to iterate the network. But first the choice of iteration may be selected, i.e., synchronous (all at the same time) or asynchronous (randomly and one at a time). The user can then iterate the network using the *1 Step* or *5 Step* buttons and observe the network's progress as the *Output Pattern*.

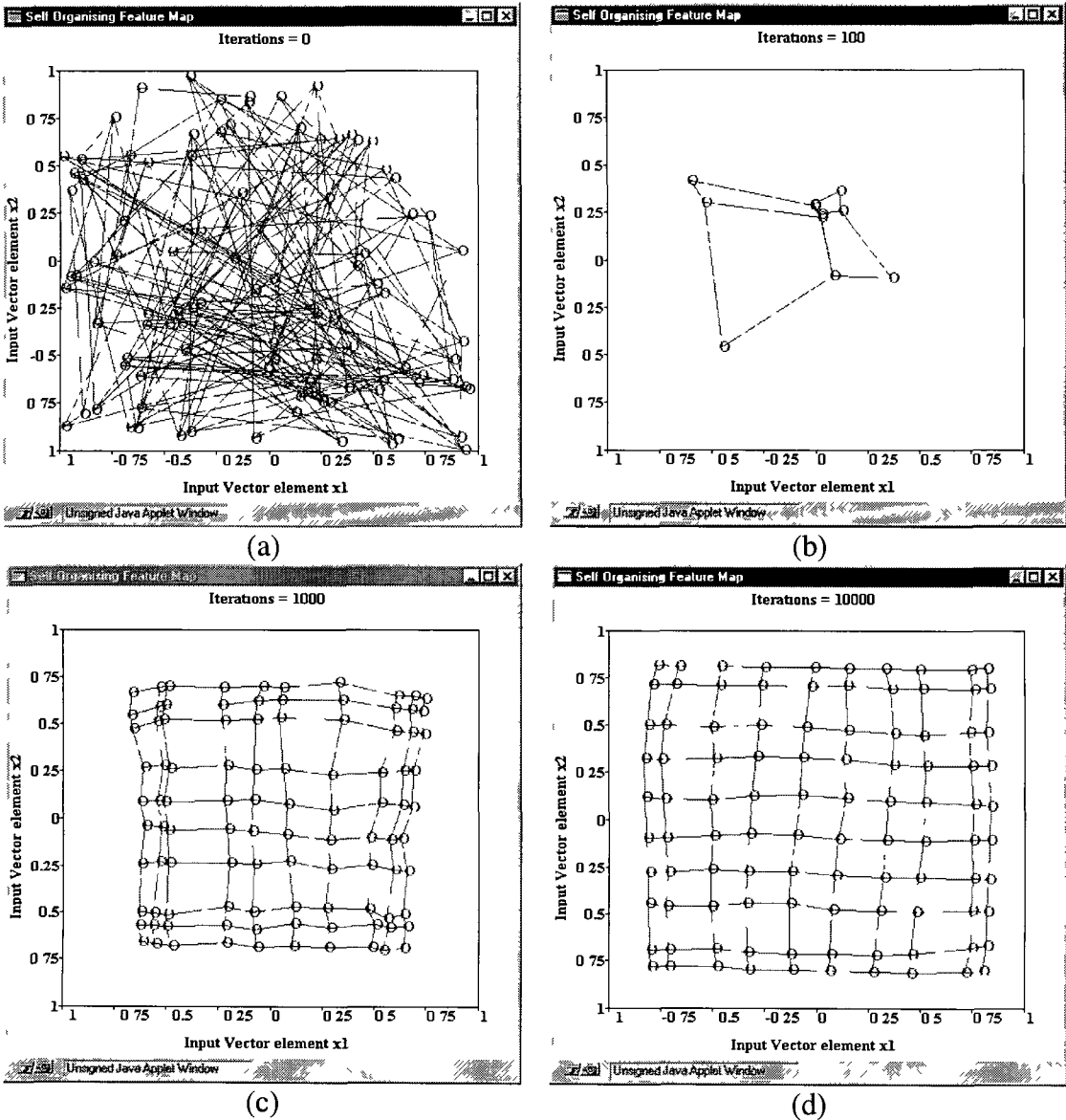
The details of the iterations and the networks performance are all shown below the *Output Pattern*. The user is presented with the *Number of Iterations*, the *Best Match Pattern* which refers to the pattern with the highest correlation (or lowest *Hamming Distance*) with the *Output Pattern*, the associated *Correlation* value and the *Energy* of current state of the network. The *Energy* is evaluated using Chapter 4's Equation (29) and is presented to the user both numerically as previously described and also in the form of the graph shown in Figure 5.14



**Figure 5.14** Energy Plot of Hopfield Network

### 5.4.6 Kohonen Self-Organising Feature Map Demonstration

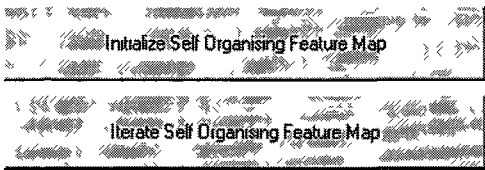
The *Kohonen Self-Organising Feature Map (SOFM) Demonstration* is used to show how an ANN can be competitive, self-organising and learn from its environment without supervision. The demonstration consists of 100 neurons, arranged in the form of a two-dimensional lattice with 10 rows and 10 columns. The network is trained with a two-dimensional input vector  $x$ , whose elements  $x_1$  and  $x_2$  are uniformly distributed in the region  $\{-1 < x_1 < 1, -1 < x_2 < 1\}$ . The network is initialised as shown in Graph (a) of Figure 5.15, with random synaptic weights.



**Figure 5.15** Progress of SOFM Algorithm (a) Initial Random Weights (b) After 100, (c) After 1000, (d) and After 10000 Iterations

### 5.4.6.1 Kohonen SOFM Demonstration Operation

The demonstration uses the algorithm provided in Appendix C. Unfortunately, in this demonstration the user has practically no interaction with the network's operation. The demonstration is presented to the user in a Web document in the form of two buttons as shown in Figure 5.16.



**Figure 5.16** SOFM Demonstration

The user must invoke the *Initialise Self-Organising Feature Map* button in order to view the initialised network shown in Figure 5.15. The next step is to invoke the *Iterate Self-Organising Feature Map* button and thereby commence the learning process. This consists of two phases: (1) the *Ordering Phase* which comprises the first 1000 iterations, presenting to the user in steps of 100, (2) the *Convergence Phase* which comprises the next 9000 iterations, presented in steps of 1000. The *Iterate Self-Organising Feature Map* button must be invoked after each 100 steps of the *Ordering Phase* or each 1000 steps of the *Convergence Phase*. Figure 5.15 shows the progress of the network at four different stages.

## 5.5 Conclusions

The Web environment supports the development of interactive graphical representations in the form of Java applets. The interactivity provided by the demonstrations described in this chapter allows students to play an active role in the investigation of ANN systems. The demonstration make extensive use of pictures, graphs and graphical user interfaces and text in order to facilitate the understanding of complex concepts associated with the field of ANNs.

Since Java is a high-level language, which supports object-oriented programming, it was possible for mathematical algorithms behind the ANNs to be executed in a fast and efficient manner. The training and testing of the ANNs described in this chapter, could therefore be performed automatically, and in a relatively seamless manner, depending on the dynamics of the application. The demonstrations successfully eliminate the drudgery of performing tedious, repetitive calculations and thereby focus the students' attention on developing an insight into the ANNs theory, as suggested by Section 2.2.4.

In addition, due to fact that the Java demonstrations offer a considerable amount of freedom and flexibility towards the investigation of ANNs, it would be advantageous to incorporate the demonstrations in assignments. A wide range of assignments could be developed around the demonstrations, which would highlight different characteristics of ANNs. As part of the assignments, students could submit both textual and graphical data obtained from the demonstrations as part of their findings. The Java demonstrations are therefore developing another channel through which knowledge can be obtained and assessed.

# **Chapter 6: Evaluation of the ANNs**

## **Courseware**

### **6.1 Introduction**

The purpose of this chapter is to discuss the process of evaluating the ANNs courseware. The evaluation consisted of developing a questionnaire, which would accurately convey the opinions of the students using the courseware. The results of the questionnaire are analysed, both quantitatively and qualitatively, thus providing the future work and recommendations discussed in Chapter 7.

### **6.2 Development of the Questionnaire**

#### **6.2.1 The Purpose of a Questionnaire**

The employment of questionnaires has long been the building block of market research. Questionnaires provide a means of determining the success of a product through obtaining feedback from consumers. The response of the consumer may then be analysed and used to improve the design of the product [Hague].

They are basically four purposes of questionnaires:

- The primary role of a questionnaire is to draw accurate information from the respondent. The researcher is trying to obtain as close a picture as possible of the general consensus. Accurate information is obtained by asking the right question of the right person [Hague].



- The questionnaire provides a structure to an interview so that it flows smoothly and orderly. It is important in any survey that all respondents are asked the same questions in exactly the same way. In addition, the questionnaire acts as a memory aid to the interviewer. For the respondent it gives a logical sequence to the questions, driving towards a point and moving smoothly on to the next subject [Hague]
- The third purpose of the questionnaire is to provide a standard format on which facts, comments and attitudes can be recorded. A record is essential, otherwise points could be forgotten or distorted [Hague]
- Finally, the questionnaire facilitates data processing. The efficiency of the questionnaire analysis depends on knowing the relevance and order of each response [Hague]

### 6.2.2 Questionnaire Structures

The effectiveness of the questionnaire is dependent on its structure. Researchers recognise three different types of interview situations, which in turn require three different types of questionnaires [Questionnaire Design]

- **Structured.** In structured interviews, the questionnaires set out precisely the wording of the questions and the order in which they will be asked. Most of the questions have predefined answers and there will be little latitude for a respondent to stray beyond them. Structured questionnaires and interviews are the foundation of large quantitative surveys.
- **Semi-structured.** This type of interview used questionnaires with a mixture of questions with predefined answers as well as those where the respondent is free to say whatever is liked. In each interview the questions are asked in the same way and there may be hundreds of interviews in the whole survey. The semi-structured questionnaire is a more flexible tool than its highly structured counterpart and there is likely to be more probing to find out the reasons for certain answers.

- **Unstructured.** This is an informal interview, the researcher uses a checklist of questions rather than a formal questionnaire on which the answers are written down. There is considerable latitude allowed on the part of the interviewer and different channels of questioning will be selected during the interview itself.

### 6.2.3 Types of Questions

Structured and semi-structured questionnaires are made up of three different types of questions depending on the type of information which is being collected [Questionnaire Design]

**1. Behavioural.** These types of questions seek factual information on what the respondent is, does or owns. The frequency with which these actions are performed is also relevant. Behavioural questions seek to determine the interest, usage and awareness associated with the product or service concerned.

Behavioural questions address the following

- Have you ever ?
- Do you ever ?
- Which do you do most often ?
- Do you have ?
- In the future will you ?

**2. Attitudinal.** Such questions determine people's thoughts on the product or service. They show their image and ratings of things, and why they do them. Attitudinal surveys provide an overall consensus of the public's thoughts about the product or service.

Attitudinal questions address the following

- What do you think of ?
- Why do you ?
- Do you agree or disagree ?
- How do you rate ?
- Which is best (or worst) for ?

The likelihood of the response is often gauged quantitatively. For example, responses could be collected on a five point scale running from strongly disagree with a value of 1, to strongly agree with a value of 5, i.e.

For the questions below circle the number which most closely represents your view. Strongly Disagree=1, Disagree=2, Uncertain=3, Agree=4, Strongly Agree=5

How would you rate the effectiveness of the Hopfield

Java demonstration in conveying the properties of Hopfield networks? 1 2 3 4 5

The results of quantitative responses can be easily analysed by calculating the mean or average value of each response. This can then be used to obtain the overall opinion or attitude of the respondents.

**3. Classification.** These questions seek information that can be used to group respondents to see how they differ from one another—such as age, gender, social class, location of household, type of house, family composition etc. Classification questions can be used in all types of surveys to provide a demographic picture of the respondents.

Classification questions are required to check that the correct quota of each type of person has been interviewed. They are often used to compare and contrast the different answers of one group of respondents with those of another. Usually the

information that is required for a classification question is behavioural (factual)  
[Questionnaire Design]

Classification questions are usually presented with the aid of check boxes, i.e.

Current Educational Status of Respondent is

- Undergraduate Student ☐
- Taught Masters Student ☐
- Research Masters Student ☐
- Higher Diploma Student ☐
- Other ☐

The results of these types of responses may be used to obtain a proportionate account of the classes of respondents

### 6.3 The Structure of the ANN Questionnaire

The *ANN Java Demonstrations Questionnaire* or *ANN Questionnaire* was developed such that the general consensus of people's opinions about the Java demonstrations could be ascertained. In order to allow respondents to review the material prior to filling in the questionnaire, it was decided that an interview was not required. In addition to allowing respondents to answer in their own time, this also has the effect of generating a more honest and accurate response. It was also decided that a *Semi-Structured* questionnaire would be more appropriate because in addition to having a set of predefined responses, there are plenty of opportunities for respondents to convey their thoughts. Appendix D shows the actual ANN Questionnaire.

#### 6.3.1 ANN Questionnaire Introduction

The questionnaire begins by providing a brief description of its purpose, i.e., instructing the respondent that the questionnaire concerns only the presentation of the Java demonstrations. The questionnaire then requests that the responses reflect an honest representation of the respondent's opinions.

### 6.3.2 The Questions

The questionnaire is structured such that five distinct sections may be observed. Section 1 consists of classification questions. The two questions are used to discover the educational status and the nature of the respondent's Internet access. In each case the questionnaire supplies a set of predefined descriptions along with their associated check boxes.

Section 2 of the questionnaire consists of both behavioural and attitudinal questions. The questions attempt to discover the amount of exposure that the respondent has had to computer-aided learning (CAL) and request their opinions on the matter.

Section 3 concerns the Java demonstrations. It consists of a number of quantitatively gauged statements concerning each of the Java demonstrations. The following are the statements made about each demonstration:

*The objectives of the demo were clearly stated.*

*The Instructions fully conveyed the demo's operation.*

*The demo offered a high degree of versatility.*

*The demo increased your understanding of the network.*

In each case the respondent would gauge the accuracy using the five-point scale described in Section 6.2.3. Similarly, the respondent's opinions concerning the graphing tool were ascertained using the following statements:

*The graphs clearly show the required plots.*

*The graphing tool offers a high degree of versatility.*

*The graphing tool compares well to other graphing facilities*

Section 4 of the questionnaire uses attitudinal questions in order to elicit a somewhat critical set of responses from the respondent. The first question asks how the ANN demonstrations could be improved upon. The second requests a comparison between the ANN demonstrations with other CAL packages. In both cases ample space is

provided in an attempt to encourage elaborate responses. Section 5 simply requests additional comments and suggestions.

## **6.4 Analysis of ANN Questionnaire**

The ANN Questionnaire proved to be quite effective at eliciting useful feedback from the respondents. Section 1 was used to classify the respondents on the basis of their Internet access and their current educational status. Of the fifteen respondents, two had Internet access from work, thirteen had access from college and two also had access from home. With regard to educational status, there were five taught masters students, six research students and four members of the academic staff, all members of Dublin City University.

Section 2 intended to determine the respondent's experience with other CAL packages. The first point of interest was that eight of the fifteen respondents had never used a CAL package before, and of those who had, only three had encountered an Internet-based CAL package. This is perhaps a reflection of the level of innovation associated with this project.

The following is an outline of the opinions stated upon inquiring, 'Did you find it more effective than traditional teaching methods?'. Although the question was intended to refer to previously encountered CAL packages, most respondents tended to offer their opinions toward the Java demonstrations. With regard to the Java demonstration replies, they were thought to be complementary to traditional teaching methods and can make the theory easier to understand and learn. A repeated response was that the demonstrations allowed the user to work at their own pace. The demonstrations' interactivity and their ability to show dynamic graphing procedures and perform complex computations were also well recognised. In terms of the general responses concerning CAL, the respondents communicated a preference towards instant feedback on their progress. In addition, CAL was thought to be more flexible than traditional teaching methods in that it allowed the user to interact with the theory.

The following summarises the responses of the question, 'What features did you like or dislike about it?'. The Java demonstrations based responses focused on the effectiveness of the demonstrations in conveying the ANN theory, both in terms of their ease of use and with regard to its facilities (such as the graphing tool). The ease

of navigation around the courseware was also acknowledged. The general CAL responses encompassed a number of advantageous features, including interactivity, simulations which allow repeated trial-and-errors and superior presentations especially in relation to the graphical-user interface (GUI) environments. It is clear from the positive responses obtained from Section 2 that the general consensus is that of appreciation towards the CAL philosophy.

Section 3 was used to obtain quantitative results concerning the Java demonstrations, using a five-point scale. Table 1 shows the mean value obtained by each of the demonstrations in relation to their four common questions.

**Table 1** Quantitative Results of ANN Questionnaire

| Question  | Hebb.<br>Demo | P'tron<br>Demo | Logic<br>MLP | Poly.<br>MLP | RBF<br>Demo | Hop-<br>field | SO<br>FM |
|---|---------------|----------------|--------------|--------------|-------------|---------------|----------|
| The objectives of the demo were clearly stated.       | 3.00          | 2.75           | 3.67         | 4.25         | 3.45        | 3.57          | 3.25     |
| The Instructions fully conveyed the demo's operation. | 2.70          | 2.50           | 3.67         | 4.00         | 3.91        | 4.14          | 3.88     |
| The demo offered a high degree of versatility.        | 4.30          | 4.13           | 4.33         | 4.25         | 4.27        | 4.21          | 3.38     |
| The demo increased your understanding of the network. | 3.67          | 3.75           | 4.00         | 4.13         | 4.09        | 4.08          | 3.75     |

It is clear from the results of Table 1 that the goals of the Java demonstrations were quite well recognised, although clearer objectives and more detailed instructions have been recommended by the respondents. This is especially true for the Hebbian learning and the Perceptron learning demonstrations, as can be seen from Table 1. Overall, the demonstrations were considered very useful and in general, enhanced the understanding of the ANNs.

Sections 4 and 5 were also extremely effective in generating useful feedback from the respondents. There were some criticisms posed in these sections regarding the presentation in terms of some buttons being only partially visible and one or two of the demonstrations being too large to fit, even on a high-resolution screen. The former can be resolved by ensuring that the Java code is sufficiently versatile to compensate for the peculiarities of browsers running on different operating systems. However, the latter represents a trade-off between completeness of the demonstration and its aesthetics. It is the opinion of the author that it is better to offer the students a

comprehensive simulation, even if it requires them scrolling the screen in order to see the whole demonstration

Some recommendations were made concerning improving the dynamics of the demonstrations there should be more example or default values within the text-fields, it should be possible to use the 'tab' button to move the cursor between values, the 'return' button should also signify the completion of a sub-section, as opposed to always having to press a particular button In addition, respondents would appreciate more step-by-step instructions regarding setting-up demonstrations and recommendations on how to choose initial values

With regard to the graphing tool, it has been stated that the graphing tool does not offer a substantial level of versatility in terms of the user being able manually configure the graph to suit their needs In addition, the zooming facility offered by the graphing tool, was not acknowledged by some of the respondents As a result, this and other facilities should be conveyed to the user through additional help tools

In relation to the instructions provided by some of the demonstrations, it was thought that rather than having the instructions after the demonstrations, it would be better to either have the instructions or a link to the instructions above the demonstration It was also recommended that there should exist more hypertext links for background theory and references In addition, there were some inconsistencies between the labelling of demonstration components within the instructions compared with those of the actual demonstrations

The remainder of Section 4 and 5's responses were positive In general, the comments suggested that the Java demonstrations performed well in relation to other CAL packages and were effective in conveying the operation of the ANNs The practical comments found the portability of the demonstrations to be a considerable asset, the content of the courseware to be very satisfactory and the speed of presentation to be more than ample, even when compared with other packages The fact that the graphs were presented in their own manipulatable window was also considered to be very useful The demonstrations were also said to be very easy to use, highly interactive and fun One respondent's comment stated 'the system shows great promise for



future development’ In conclusion, an obviously enthusiastic respondent made the following statement

‘A very impressive CAL package, which clearly conveys the theories involved Visually impressive and an effective means of teaching the topics ’

## **6.5 Conclusions**

The questionnaire has proved to be an effective and efficient means of obtaining the necessary feedback from the respondents The data obtained through the use of the questionnaire has made it possible to accurately assess the productiveness of the Java demonstrations It is clear from the results described in this chapter that the Java demonstrations were generally well received and were considered to be useful supplementary material for the MEng program The Java demonstrations have proved to be highly interactive, versatile, and from the limited evaluation received through the questionnaire, would appear to enhance a student’s understanding of artificial neural networks In addition, the employment of a questionnaire has made the author aware of some necessary amendments that should be made

# Chapter 7: Conclusions

## 7.1 Introduction

This chapter is used to present the findings of the work described in this thesis. Recommendations are made concerning the future of the Internet CAL courseware developed by the author and also the Internet CAL paradigm on a whole.

## 7.2 Artificial Neural Networks via Java Demonstrations

As previously described, there is a strong analogy between Java's object-oriented development and that of an artificial neural network. It is the discovery of this analogy which made it possible for the author to efficiently code diverse network architectures on the basis of one simple building block: the Neuron class.

The Java programming language has proved to be highly qualified to perform the necessary graphing procedures, interactive displays and mathematical calculations associated with the ANN demonstrations on a whole. The Java demonstrations have been described as highly interactive, easy to use, and as a visually stimulating means of supplementing traditional lecture material. It may therefore be concluded that Java has played an essential role in the Internet CAL paradigm.

## 7.3 The Internet CAL Paradigm

The Internet provides a medium over which to communicate courseware in the form of Web documents and their embedded applets. It is the support of the Web and its HTML structure that makes the Internet such an effective educational tool. The worldwide accessibility of the Web over the Internet has created a channel for distance learning that has previously been unimaginable. The Internet is therefore capable of supplying a rich learning experience to students all over the world. The work described in this thesis suggests that with a few embellishments, it may be possible for

the Internet ANNs courseware to be used as a wholly independent distance learning course

It is clear from the results described in Chapter 6 that, in general, students are extremely enthusiastic about the development of Internet CAL courseware. It is well known that if students are enthusiastic and thereby highly motivated, a much richer learning experience may be achieved. It is therefore the opinion of the author that CAL and Internet CAL will play an important role in the field of education.

## **7.4 Future Work and Recommendations**

In relation to the project discussed in this thesis, there are a number of amendments that are recommended. Firstly, the minor compatibility issues mentioned by the ANN Questionnaire respondents in Chapter 6, should be resolved as soon as possible. Moreover, some additional courseware developed for the MEng program requires conversion in order to be presented by the Web. Also, the existing Web courseware may be expanded upon through the incorporation of additional hypertext links, which would both interconnect the existing courseware and provide relevant supplementary material.

Due to the Web's modular and extensible hypertext document structure there is no reason why someone other than the author could not expand upon the work described in this thesis. In addition to the generation of additional Web courseware, the stand-alone and modular nature of Java applets means that a whole set of new Java demonstrations could be developed. As a result of Java being object-oriented and thereby inclined towards the re-usability of code, the generic classes such as the Graph class developed by the author, could all be incorporated into new Java demonstrations. Therefore, this project may be considered the foundation of an elaborate series of Internet-based instructional materials.

# Bibliography

Anderson, J A , “An Introduction to Neural Networks”, Massachusetts The MIT Press, p5, 1995

ANNs Web Site @ [http //eeng dcu ie/calcon/paul/](http://eeng.dcu.ie/calcon/paul/)

Broomhead, D S , Lowe, D , “Multivariate Functional Interpolation and Adaptive Networks”, Complex Systems, Vol 2, pp321-355, 1988

CERN @ [http //www cern ch/](http://www.cern.ch/)

Conway, K L , “Putting Technology in its Place The Classroom”, Institute for Academic Technology, p5, Spring 1991

Cover, T M , “Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition”, IEEE Transactions on Electronic Computers, Vol 14, pp326-334, 1965

CTI Autumn 1995, “The Internet - A Simple Technical Explanation”, The Newsletter from CTI Engineering Software for Engineering Education, Autumn 1995

Drossu, R , Obradovic, Z , Fletcher, J , “A Flexible Graphical User Interface for Embedding Heterogeneous Neural Network Simulators”, IEEE Transactions on Education, Vol 39 No 3, August 1996

Flur, P W , Lockhart, J B , Yalamanchili, S , “Integrating Academic Services in a Modern Networked Environment”, IEEE Transactions on Education, Vol 39, No 3, August 1996

Galvin, G , “Computer Aided Learning Tool”, BEng Report, School of Electronic Engineering, DCU, April 1997

Gros, B , Elen, J , Kerres, M , Merrienboer, J , Spector, M , “Instructional Design and the Authoring of Multimedia and Hypermedia Systems Does a Marriage Make Sense?”, Educational Technology, January-February 1997

Hague, P , The Market Research Series “Questionnaire Design”, Clays Ltd, St Ives Plc, England, 1993

Haykin, S , “Neural Networks A Comprehensive Foundation”, New York Macmillan College Publishing Company, p8, p106, 1994

Harding, R D , Lay, S W , Moule, H , Quinney, D A , “Multimedia Interactive Mathematics Courseware The Mathematics Experience within the Renaissance Project” Computers and Education, Vol 24, No 1, 1995.

Harger, Robert O , “Introducing DSP with an Electronic Book in a Computer Classroom”, IEEE Transactions on Education, Vol 39, No 2, May 1996

Hebb, D O , “The Organization of Behavior A Neuropsychological Theory”, New York Wiley, 1949

Hoffman, S , “Elaboration Theory and Hypermedia Is There a Link?”, Educational Technology, January-February 1997

Hopfield, J J , “Neural Networks and Physical Systems with Emergent Collective Computational Abilities”, Proceedings of the National Academy of Science, Vol 79, pp2554-8, 1982

Hush, D R , Horne, B G , “Progress m Supervised Neural Networks What’s New Since Lippmann?”, IEEE Signal Processing Magazine, January 1993

Keeling, P , “A Computer Aided Learning Package to Teach System Dynamics and Control”, BEng Report, School of Electronic Engineering, DCU, April 1996

Knussen, C , Tanner, R G , & Kibby, M R , “An Approach to the Evaluation of Hypermedia” Computer Education, 1992

Kohonen, T , “Self-Organizing and Associative Memory”, Berlin Springer-Verlag, 1984

Lai, L L , “Computer Assisted Learning in Power System Relaying” IEEE Transactions on Education, Vol 38, No 3, August 1995

Lee, P M , Sullivan, W G , “Developing and Implementing Interactive Multimedia in Education”, IEEE Transactions on Education, Vol 39, No 3, August 1996

Light, W A , “Some Aspects of Radial Basis Function Approximation” In Approximation Theory, Spline Functions and Applications (S P Singh, ed ), NATO ASI Series, Vol 256, p 163-190 Boston, Massachusetts Kluwer Academic Publishers, 1992

Lippmann, R P , “An Introduction to Computing with Neural Nets”, IEEE Acoustics, Speech and Signal Processing Magazine, Vol 4 No 2, April 1997

MATLAB @ [http //www mathworks com/](http://www.mathworks.com/)

Maul, R W , “The Network Classroom”, Interpersonal Computing and Technology An Electronic Journal for the 21<sup>st</sup> century, Vol 1, No 1, January 1993

Matsubara, Y , Nagamachi, M , “Motivation System and Human Model for Intelligent Tutoring”, Proceedings of the Third International Conference of Intelligent Tutoring Systems, Montréal, Canada, June 1996

McCulloch, W S , Pitts, W , “A Logical Calculus of the Ideas Immanent in Nervous Activity”, Bulletin of Mathematical Biophysics, Vol 5, pp 115-133, 1943

Mengel, S A , Adams, W J , “The Need for a Hypertext Instructional Design Methodology”, IEEE Transactions on Education, Vol 39, No 3, August 1996

Merrill, M D , Kelety, J C , Wilson, B , “Elaboration theory and cognitive psychology”, Instructional Science, Vol 10 No 3, pp217-235 1981

Mosaic @ [http //www ncsa uiuc edu/SDG/Software/Mosaic/NCSAMosaicHome.html](http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/NCSAMosaicHome.html)

Myers, D K , “Interactive Video A Chance to Plug the Literacy Leak”, Industry Week, No 239, pp 15-18, April 1990

NCSA @ [http //www ncsa uiuc edu/](http://www.ncsa.uiuc.edu/)

Netscape @ [http //cgi netscape com/](http://cgi.netscape.com/)

Newman, A , et al , “Special Edition Using Java”, Que Corporation, 1996

Noonan, D , “Multimedia” CD-ROM Today, Issue 1, April/May 1994

Orsak, G C , Etter D M , “Connecting the Engineer to the 21<sup>st</sup> Century Through Virtual Teaming”, IEEE Transactions on Education, Vol 39 No 2, May 1996

Patterson, Dan W , “Artificial Neural Networks Theory and Applications”, Singapore Prentice Hall, 1996

Powell, M J D , “Radial Basis Functions for Multivariate Interpolation A Review”, Technical Report DAMPT 1985/NA12, Department of Applied Mathematics and Theoretical Physics, Cambridge University, Cambridge, England, 1995

Reigeluth, C M , Curtis, R V , “Learning Situation and Instructional Models”, Instructional Technology, Hillsdale, New Jersey Erlbaum, 1987

Ringwood, J , “Artificial Neural Networks”, Lecture Material Presented to the Masters Students, DCU, 1997

Rosenblatt, F “The Perceptron A Probabilistic Model for Information Storage and Organization in the Brain”, Psychology Review, Vol 65, 1958

Rezabek, R H , Ragan, T J , “Elaborated Resources An Instructional Design Strategy for Hypermedia”, Paper presented at the annual meeting of the Association for Educational Communications and Technology, Dallas, Texas, 1989

Schodorf, J B , Yoder, M A , McClellan, J H , “Using Multimedia to Teach Theory of Digital Multimedia Signals”, IEEE Transactions on Education, Vol 39, No 3, August 1996

Sears, A L , Watkins, S E , “A Multimedia Manual on the World Wide Web for Telecommunications Equipment, IEEE Transactions on Education, Vol 39, No 3, August 1996

Spiro, R J , & Jehng, J , Cognitive flexibility and hypertext “Theory and Technology for the Nonlinear and Multidimensional Traversal of Complex Subject Matter” In D Nix & R Spiro (Eds ), Cognition, Education, and Multimedia “Exploring Ideas in High Technology”, pp 163-205, Hillsdale, New Jersey Lawrence Erlbaum Associates, 1990

Spitzer, D R (a), “Motivation The Neglected Factor in Instructional Design”, Educational Technology, May-June 1996

Spitzer, D R (b), “SuperMotivation”, New York AMACOM Books, 1995

Stanley, A , “An Introduction to Multimedia and Interactive Video in Higher Education” Computer Education 80, 1995

Thackray, J , “Internet Editorial”, Education & Training, Vol 39 No 2, 1997

Tolhurst, D , “Hypertext, Hypermedia, Multimedia Defined?” Educational Technology, March-April 1995

Tou, J T , Gonzalez, R C , “Pattern Recognition Principles”, Addison-Wesley, 1974

VCLab @ [http //www esr ruhr-um-bochum de/VCLab/](http://www.esr.ruhr-um-bochum.de/VCLab/)

WinZip @ [http //www winzip com/](http://www.winzip.com/)

Zurada, J M , “Introduction to Artificial Neural Systems”, St Paul, Minnesota West Publishing Company, pp18-30

# Appendix A

## The Back-Propagation Training Algorithm

The back-propagation training algorithm is an iterative gradient algorithm designed to minimise the mean square error between the actual output of a multi-layer feed-forward perceptron and the desired output. It requires continuously differentiable non-linearities. The algorithm assumes a sigmoid logistic non-linearity is used where the function  $f(z)$  is stated

**Step 1 Initialise Weights and Biases**

Set all weights and node biases to small random values

**Step 2 Present Input and Desired Outputs**

Present a continuous valued input vector  $x_0, x_1, \dots, x_{N-1}$  and specify the desired outputs  $d_0, d_1, \dots, d_{M-1}$ . The input could be new on each trial or samples from a training set could be presented cyclically until weights stabilise

**Step 3 Calculate Actual Outputs**

Calculate the outputs  $y_0, y_1, \dots, y_{M-1}$  by forward-propagating the response of each layer's nodes into the next

**Step 4 Adapt Weights**

Use a recursive algorithm, starting at the output nodes and working back to the first hidden layer. Adjust weights by

$$w_{ij}(t+1) = w_{ij}(t) + \rho \delta_j x_i$$

In this equation  $w_{ij}(t)$  is the weight from hidden node  $i$  or from an input to node  $j$  at time  $t$ ,  $x_i$  is either the output of node  $i$  or is an input,  $\rho$  is a gain term, and  $\delta_j$  is an error term for node  $j$ . If node  $j$  is an output node, then

$$\delta_j = y_j (1 - y_j) (d_j - y_j)$$

where  $d_j$  is the desired output of node  $j$  and  $y_j$  is the actual output

If node  $j$  is an internal hidden node, then,

$$\delta_j = x_i (1 - x_i) \sum_k \delta_k w_{jk}$$

where  $k$  is over all nodes in the layers above node  $j$ . Internal node thresholds are adapted in a similar manner by assuming they are connection weights on links from auxiliary constant-valued inputs. Convergence is sometimes faster if a momentum term is added and weight changes smoothed by

$$w_{ij}(t+1) = w_{ij}(t) + \rho \delta_j x_i + \alpha [w_{ij}(t) - w_{ij}(t-1)], \quad \text{where } 0 < \alpha < 1$$

**Step 5 Repeat by Going to Step 2**



# Appendix B

## Summary of Hopfield Model

The operational procedure for the Hopfield network may now be summarised as follows

### Step 1 Storage (Learning)

Let  $\xi_0, \xi_1, \dots, \xi_p$  denote a known set of  $N$ -dimensional memories. Construct the network by using the outer product rule to compute the synaptic weights of the network as

$$w_{ji} = \begin{cases} \frac{1}{N} \sum_{\mu=1}^p \xi_{\mu j} \xi_{\mu i}, & j \neq i \\ 0, & j = i \end{cases}$$

where  $w_{ji}$  is the synaptic weight from neuron  $i$  to neuron  $j$ . The elements of the vector  $\xi_{\mu}$  equal  $\pm 1$ . Once they are computed, the synaptic weights are kept fixed.

### Step 2 Initialisation

Let  $x$  denote an unknown  $N$ -dimensional input vector (probe) presented to the network. The algorithm is initialised by setting

$$s_j(0) = x_j, \quad j = 1, 2, \dots, N$$

where  $s_j(0)$  is the state of neuron  $j$  at time  $n = 0$ , and  $x_j$  is the  $j$ th element of the probe vector  $x$ .

### Step 3 Iterate until Convergence

Update the elements of state vector  $s(n)$  asynchronously (i.e. randomly and one at a time) according to the rule

$$s_j(n+1) = \text{sgn} \left[ \sum_{i=1}^N w_{ji} s_i(n) \right]$$

where  $\text{sgn}$  denotes the *signum function* (step function with output range  $[-1, 1]$ ).

Repeat the iteration until the state vector  $s$  remains unchanged.

Note: It is also possible to update the state vector  $s(n)$  synchronously (i.e. all at the same time).

### Step 4 Outputting

Let  $s_{\text{fixed}}$  denote the fixed point (stable state) computed at the end of Step 3. The resulting output vector  $y$  of the network is

$$y = s_{\text{fixed}}$$

# Appendix C

## SOFM Algorithm

The SOFM example provided consists of a two-dimensional lattice with 10 rows and 10 columns ( $N = 100$ ). The network is trained with a two-dimensional input vector ( $p = 2$ )  $x$ , whose elements  $x_1$  and  $x_2$  are uniformly distributed in the region  $\{-1 < x_1 < 1, -1 < x_2 < 1\}$

### Step 1 Initialisation

The initial weight vectors  $w_j(0)$  are chosen at random from the region  $\{-1 < x_1 < +1, -1 < x_2 < +1\}$  Where  $w_j(0)$  is different for all  $j = 1, 2, \dots, N$

### Step 2 Sampling

A sample input vector or sensory input  $x$  is drawn from the input distribution with a certain probability

### Step 3 Similarity Matching

The winning neuron  $i(x)$  at time  $n$  is found using the criterion,

$$i(x) = \arg \min_j \|x - w_j\|, j = 1, 2, \dots, N$$

### Step 4 Updating Neurons : Ordering Phase ( $n = 0 \rightarrow 1000$ )

All synaptic weight vectors of the neurons within the topological neighbourhood  $\Lambda_{i(x)}(n)$  are updated according to,

$$\begin{aligned} w_j(n+1) &= w_j(n) + \eta(n) \pi_{i,j}(n) [x(n) - w_j(n)], & j \in \Lambda_{i(x)}(n) \\ w_j(n+1) &= w_j(n), & \text{otherwise} \end{aligned}$$

where,

$$\begin{aligned} \eta(n) &= \eta_0 \exp(-n / t_1), & \eta_0 &= 1, t_1 = 432 \\ \pi_{i,j}(n) &= \exp(-d_{j,i}^2 / 2\sigma(n)), & d_{j,i} &= \text{lateral distance between } i \text{ \& } j \\ \sigma(n) &= \sigma_0 \exp(-n / t_2), & \sigma_0 &= 10, t_2 = 432 \\ \Lambda_{i(x)}(n) &= (\text{Integer}) \sigma(n) \end{aligned}$$

### Step 5 Update Neurons : Convergence Phase ( $n = 1000 \rightarrow 10000$ )

All synaptic weight vectors of the neurons within the topological neighbourhood are updated according to,

$$\begin{aligned} w_j(n+1) &= w_j(n) + \eta(n) [x(n) - w_j(n)], & j \in \Lambda_{i(x)}(n) \\ w_j(n+1) &= w_j(n), & \text{otherwise} \end{aligned}$$

where,

$$\begin{aligned} \eta(n) &= 0.01 \\ \Lambda_{i(x)}(n) &= \sigma(n) = 1 \end{aligned}$$

# **Appendix D**

## **ANN Java Demonstrations Questionnaire**

### **Introduction**

The purpose of this questionnaire is to evaluate the effectiveness of the Artificial Neural Network (ANN) Java Demonstrations in conveying the properties of the their respective network configurations. The ANNs web site is located at

[http //www.eeng.dcu.ie/~calcon/paul/](http://www.eeng.dcu.ie/~calcon/paul/)

The ANN Java demonstrations are as follows

- Hebbian Learning
- Perceptron Learning
- Multi-Layer Perceptron (MLP) Logic Function Approximation
- Multi-Layer Perceptron (MLP) Polynomial Function Approximation
- Radial Basis Function (RBF) Polynomial Function Approximation
- Pattern Storage using a Hopfield Network
- Kohonen Self-Organizing Feature Map (SOFM)

### **Instructions**

After investigating the Java Demonstrations, please answer the following questions. There are no right or wrong answers in this questionnaire. Please answer honestly and frankly. The data obtained from this questionnaire will be used for research purposes only and is completely confidential.

### **Questions**

#### **1.1 Where do you have Internet Access?**

Home  
College  
Work  
Other

#### **1.2 What is Your Current Educational Status?**

Undergraduate Student  
Taught Masters Student  
Research Student  
Academic Staff  
Other

## Appendix D

### 2. Have you ever used a computer-aided learning (CAL) package before?

If the answer is yes, please answer the following questions

2 1 Was it an Internet based CAL package? Yes / No

2 2 Did you find it more effective than traditional teaching methods? Please state

---

---

---

2 3 What features did you like or dislike about it? \_\_\_\_\_

---

---

---

### 3. For the following statements, circle the number which most accurately represents your view.

(1=Strongly Disagree, 2=Disagree, 3=Uncertain, 4=Agree, 5=Strongly Agree)

#### Hebbian Learning Demo

|  |   |   |   |   |   |
|--|---|---|---|---|---|
| The objectives of the demo were clearly stated       | 1 | 2 | 3 | 4 | 5 |
| The Instructions fully conveyed the demo's operation | 1 | 2 | 3 | 4 | 5 |
| The demo offered a high degree of versatility        | 1 | 2 | 3 | 4 | 5 |
| The demo increased your understanding of the network | 1 | 2 | 3 | 4 | 5 |

#### Perceptron Learning Demo

|  |   |   |   |   |   |
|--|---|---|---|---|---|
| The objectives of the demo were clearly stated       | 1 | 2 | 3 | 4 | 5 |
| The Instructions fully conveyed the demo's operation | 1 | 2 | 3 | 4 | 5 |
| The demo offered a high degree of versatility        | 1 | 2 | 3 | 4 | 5 |
| The demo increased your understanding of the network | 1 | 2 | 3 | 4 | 5 |

#### MLP Logic Function Approximation Demo

|  |   |   |   |   |   |
|--|---|---|---|---|---|
| The objectives of the demo were clearly stated       | 1 | 2 | 3 | 4 | 5 |
| The Instructions fully conveyed the demo's operation | 1 | 2 | 3 | 4 | 5 |
| The demo offered a high degree of versatility        | 1 | 2 | 3 | 4 | 5 |
| The demo increased your understanding of the network | 1 | 2 | 3 | 4 | 5 |

#### MLP Polynomial Function Approximation Demo

|  |   |   |   |   |   |
|--|---|---|---|---|---|
| The objectives of the demo were clearly stated       | 1 | 2 | 3 | 4 | 5 |
| The Instructions fully conveyed the demo's operation | 1 | 2 | 3 | 4 | 5 |
| The demo offered a high degree of versatility        | 1 | 2 | 3 | 4 | 5 |
| The demo increased your understanding of the network | 1 | 2 | 3 | 4 | 5 |

# Appendix D

## RBF Polynomial Function Approximation Demo

|  |   |   |   |   |   |
|--|---|---|---|---|---|
| The objectives of the demo were clearly stated       | 1 | 2 | 3 | 4 | 5 |
| The Instructions fully conveyed the demo's operation | 1 | 2 | 3 | 4 | 5 |
| The demo offered a high degree of versatility        | 1 | 2 | 3 | 4 | 5 |
| The demo increased your understanding of the network | 1 | 2 | 3 | 4 | 5 |

## Hopfield Network Pattern Storage Demo

|  |   |   |   |   |   |
|--|---|---|---|---|---|
| The objectives of the demo were clearly stated       | 1 | 2 | 3 | 4 | 5 |
| The Instructions fully conveyed the demo's operation | 1 | 2 | 3 | 4 | 5 |
| The demo offered a high degree of versatility        | 1 | 2 | 3 | 4 | 5 |
| The demo increased your understanding of the network | 1 | 2 | 3 | 4 | 5 |

## Kohonen Self-Organizing Feature Map Demo

|  |   |   |   |   |   |
|--|---|---|---|---|---|
| The objectives of the demo were clearly stated       | 1 | 2 | 3 | 4 | 5 |
| The Instructions fully conveyed the demo's operation | 1 | 2 | 3 | 4 | 5 |
| The demo offered a high degree of versatility        | 1 | 2 | 3 | 4 | 5 |
| The demo increased your understanding of the network | 1 | 2 | 3 | 4 | 5 |

4.1 Please state in what ways you would improve on the ANN demonstrations\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

4.2 Please state how the ANN demonstrations performed in relation to other CAL packages you have used\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

5. Additional Comments & Suggestions\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_