# NEAR EAST UNIVERSITY

## Faculty of Engineering

## Department of Computer Engineering

# ARTIFICIAL INTELLIGENCE SYSTEMS

**Graduation Project**
**COM-400**

**Student:** Mehmet Reşit Karadağ

**Supervisor:** Assoc. Prof. Dr Adnan Khashman

**Nicossia - 2003**

# NEAR EAST UNIVERSITY

## Faculty of Engineering

## Department of Computer Engineering

# ARTIFICIAL INTELLIGENCE SYSTEMS

**Graduation Project**
**COM-400**

**Student:** Mehmet Reşit Karadağ

**Supervisor:** Assoc. Prof. Dr Adnan Khashman

**Nicossia - 2003**

## ACKNOWLEDGEMENT

I would liketo thank my intellectual and conscientious Assoc. Prof. Dr Adnan Khashman for inspiring and endurance me to complete this project.

I would like to express my gratitude to Near East University for the scolarship that made the work possible.

I would like to thank my family for their costant encouragement and support during the preparation of this project.

I would also like to thank all my friends for their advice and support.

# ABSTRACT

This project explores the theoretical and particular underpinning of Artificial Intelligence (A.I.) and provides an introductory-level on A.I. technology.

The reader of this project will come away with an appreciation for the basic concepts of A.I, and also with an idea of what can and cannot be done with today's technology, at what cost and using what techniques.

This project include a general background and history about Artificial Intelligence and some early examples, it concentrates on Artificial Intelligence System and how they work, some applications are also included in this project and the them of Artificial Intelligence.

In this project several model and techniques of Artificial Intelligence System available; like Artificial Neural Networks, Genetic Algorithm and how they work.

# TABLE OF CONTENTS

# INTRODUCTION

This project concerns the foundations of a new generation of computing technology and capability, most commonly referred to as Artificial Intelligence (A.I.)

Programs providing capabilities like English-language communication expert reasoning and problem solving, and even master level chess skill are having a profound effect on how people use computers and what they use them for.

**The objectives of this thesis can be summarized as:**

- Investigation a general overview, history and the development of Artificial Intelligence.
- Description of various Artificial Intelligence fields (Artificial Neural Networks, Genetic Algorithms and applications of each one of these fields).
- Investigation of different types of life applications for Artificial Intelligent System.

**Thesis structure**

Chapter one, a general background and history of "Artificial Intelligence" will present, objectivs and some early example of Artificial Intelligence System.

Chapter two, "Artificial Neural Networks" introduction to Neural Networks, general definitions, how the artificial neurons,design,unsupervised taining, reinforcementtraining, back propagation training, a variety of learning laws which Hebb's Rule, Hopfield Law, The Delta Rule, and Kohonen's Learning Law, applications of Neural Networks.

Chapter three, "Genetic ALgorithms" some definitions and how the genetic algorithms work in real life, applications ofgenetic algorithms,comparision the Genetic Algorithms and Neural Networks, and Omri Weisman and Ziv Pollack implementations.

Chapter four, represent applications of Artificial Intelligence.

# CHAPTER ONE

# ARTIFICIAL INTELLIGENCE

## 1.1 Overview

This chapter presents definitions, history and applications of Artificial Intelligence until present. Researchers in the science of " Artificial Intelligence" have investigated many areas of the mind such as pattern matching, vision, and theorem proving. However, all of these are only parts of the human mind. An intelligent system could include all of these parts, but still would not be complete, and could not function, unless it also had senses, a method to choose responses according to its objectives and memories, and some way of performing these responses in and on its environment.

## 1.2 Definition of Artificial Intelligence

**What is artificial intelligence?**

It is the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable.

**What is intelligence?**

Intelligence is the computational part of the ability to achieve goals in the world. Varying kinds and degrees of intelligence occur in people, many animals and some machines.

### 1.2.1 Ouestions and Answers

Isn't there a solid definition of intelligence that doesn't depend on relating it to human intelligence?

Not yet. The problem is that we cannot yet characterize in general what kinds of computational procedures we want to call intelligent. We understand some of the mechanisms of intelligence and not others.

Is intelligence a single thing so that one can ask a yes or no question ``Is this machine intelligent or not?''?

No. Intelligence involves mechanisms, and AI research has discovered how to make computers carry out some of them and not others. If doing a task requires only mechanisms that are well understood today, computer programs can give very impressive performances on these tasks. Such programs should be considered ``somewhat intelligent''.

### 1.2.2 Isn't AI about simulating human intelligence?

Sometimes but not always or even usually. On the one hand, we can learn something about how to make machines solve problems by observing other people or just by observing our own methods. On the other hand, most work in AI involves studying the problems the world presents to intelligence rather than studying people or animals. AI researchers are free to use methods that are not observed in people or that involve much more computing than people can do.

### 1.2.3 What about IQ? Do computer programs have IQs?

No. IQ is based on the rates at which intelligence develops in children. It is the ratio of the age at which a child normally makes a certain score to the child's age. The scale is extended to adults in a suitable way. IQ correlates well with various measures of success or failure in life, but making computers that can score high on IQ tests would be weakly correlated with their usefulness. For example, the ability of a child to repeat back a long sequence of digits correlates well with other intellectual abilities, perhaps because it measures how much information the child can compute with at once. However, ``digit span'' is trivial for even extremely limited computers.

However, some of the problems on IQ tests are useful challenges for AI.

### 1.2.4 What about other comparisons between human and computer intelligence?

Arthur R. Jensen [2], a leading researcher in human intelligence, suggests ``as a heuristic hypothesis'' that all normal humans have the same intellectual mechanisms and that differences in intelligence are related to ``quantitative biochemical and

physiological conditions". I see them as speed, short term memory, and the ability to form accurate and retrievable long term memories.

Whether or not Jensen is right about human intelligence, the situation in AI today is the reverse.

Computer programs have plenty of speed and memory but their abilities correspond to the intellectual mechanisms that program designers understand well enough to put in programs. Some abilities that children normally don't develop till they are teenagers may be in, and some abilities possessed by two year olds are still out. The matter is further complicated by the fact that the cognitive sciences still have not succeeded in determining exactly what the human abilities are. Very likely the organization of the intellectual mechanisms for AI can usefully be different from that in people.

Whenever people do better than computers on some task or computers use a lot of computation to do as well as people, this demonstrates that the program designers lack understanding of the intellectual mechanisms required to do the task efficiently.

## 1.3 When did AI start?

After WWII, a number of people independently started to work on intelligent machines. The English mathematician Alan Turing [3] may have been the first. He gave a lecture on it in 1947. He also may have been the first to decide that AI was best researched by programming computers rather than by building machines. By the late 1950s, there were many researchers on AI, and most of them were basing their work on programming computers.

### 1.3.1 History of Artificial Intelligence

Here some important events in the history of artificial intelligence are provided [6].[12]

1917 Karel Capek coins the term 'Robot' (in Czech 'robot' means 'worker'), but the
1923

English translation retained the original word.)

1928 John von Neumann's develops minimax theorem (later used in game playing programs)

1940 First electronic computer developed in US, UK and Germany

1943 McCulloch and Pitts propose neural-network architectures for intelligence

1946 Turing's ACE [3]

John von Neumann publishes EDVAC paper on the stored program computer.

J. Presper Eckert and John Mauchley build "ENIAC", the first electronic programmable digital computer.

1950 Alan Turing: "Computing Machinery and Intelligence"

1953 Claude Shannon gives Minsky and McCarthy summer jobs at Bell Labs

1955 Newell, Simon and Shaw develop "IPL-11", first AI language

1956 Rockefeller funds McCarthy & Minsky's AI conference at Dartmouth

CIA funds GAT machine-translation project

Newell, Shaw, and Simon develop The Logic Theorist

1957 Newell, Shaw, and Simon's develop the General Problem Solver (GPS)

1958 McCarthy introduces LISP (at MIT) [4]

1959 McCarthy & Minsky establish MIT AI Lab

Rosenblatt introduces Perceptron

Samuel's checkers program wins games against best human players

1962 First industrial robots

McCarthy moves to Stanford

1963 McCarthy founds Stanford AI Lab

Minsky's "Steps Towards Artificial Intelligence" publised

U.S. government grants 2.2 million dollar grant to MIT to reearch Machine-Aided Cognition (artificial intelligence). The grant came from DARPA to ensure that the US would stay ahead of the Soviet Union in technological advancements.

1964 Daniel Bobrow's Student solves high school algebra work problems

1965 Simon predicts "by 1985 machines will be capable of doing any work a man can do"

1966 Weizenbaum and Colby create Eliza

1967 Greenblatt's MacHack defeats Hubert Dreyfus at chess

1969 Minsky & Papert's "Perceptrons" (limits of single-layer neural networks) kills funding for neural net research

Kubrick's "2001" introduces HAL and AI to a mass audience

Turing's "Intelligent Machinery" published

1970 Terry Winograd's Shrldu developed

Colmerauer creates PROLOG

1972 DARPA cancels funding for robotics at Stanford

Hubert Dreyfus publishes "What Computers Can't Do"

1973 Lighthill report kills AI funding in UK

LOGO funding scandal: Minsky & Papert forced to turn leave MIT's AI lab

1974 Edward Shortliffe's thesis on Mycin

Minsky reifies the 'frame'

First computer-controlled robot

1975DARPA launches image understanding funding program.

1976 DARPA cancels funding for speech understanding research

Greenblatt creates first LISP machine, "CONS"

Doug Lenat's AM (Automated Mathematician)

Kurzweil introduces reading machine

1978 Patrick Hayes' "Naive Physics Manifesto"

1979 Journal of American Medical Assoc. says that Mycin is as good as medical experts

1980 First AAAI conference held (at Stanford)

1981 Kazuhiro Fuchi announces Japanese Fifth Generation Project

1982 John Hopfield resuscitates neural nets

Publications of British government's "Alvey Report" on advanced information technology, leading to boost in AI (Expert Systems) being used in industry.

1983 DARPA's Strategic Computing Initiative commits $600 million over 5 yrs

Feigenbaum and McCorduck publish "The Fifth Generation"

1984 Austin AAAI conference launches AI into financial spotlight

1985 GM and Campbell's Soup find expert systems don't need LISP machines

Kawasaki robot kills Japanese mechanic during malfunction

1987 "AI Winter" sets in (Lisp-machine market saturated)

1988 AI revenues reach $1 billion

The 386 chip brings PC speeds into competition with LISP machines

Minsky and Papert publish revised edition of "Perceptrons"

1992 Japanese Fifth Generation Project ends with a whimper

1992 Japanese Fifth Generation Project ends with a whimper

Japanese Real World Computing Project begins with a big-money bang

## 1.4 Humans and Artificial Intelligence

### 1.4.1 Does AI aim to put the human mind into the computer?

Some researchers say they have that objective, but maybe they are using the phrase metaphorically. The human mind has a lot of peculiarities, and I'm not sure anyone is serious about imitating all of them.

### 1.4.2 What is the Turing Test?

Alan Turing's 1950 [3] article Computing Machinery and Intelligence discussed conditions for considering a machine to be intelligent. He argued that if the machine could successfully pretend to be human to a knowledgeable observer then you certainly should consider it intelligent. This test would satisfy most people but not all philosophers. The observer could interact with the machine and a human by teletype (to avoid requiring that the machine imitate the appearance or voice of the person), and the human would try to persuade the observer that it was human and the machine would try to fool the observer.

The Turing test is a one-sided test. A machine that passes the test should certainly be considered intelligent, but a machine could still be considered intelligent without knowing enough about humans to imitate a human.

Daniel Dennett's [5] book Brainchildren has an excellent discussion of the Turing test and the various partial Turing tests that have been implemented, i.e. with restrictions on the observer's knowledge of AI and the subject matter of questioning. It turns out that some people are easily led into believing that a rather dumb program is intelligent.

### 1.4.3 Does AI aim at human-level intelligence?

Yes. The ultimate effort is to make computer programs that can solve problems and achieve goals in the world as well as humans. However, many people involved in particular research areas are much less ambitious.

### 1.4.4 How far is AI from reaching human-level intelligence? When will it happen?

A few people think that human-level intelligence can be achieved by writing large numbers of programs of the kind people are now writing and assembling vast knowledge bases of facts in the languages now used for expressing knowledge.

However, most AI researchers believe that new fundamental ideas are required, and therefore it cannot be predicted when human level intelligence will be achieved.

### 1.4.5 Are computers the right kind of machine to be made intelligent?

Computers can be programmed to simulate any kind of machine.

Many researchers invented non-computer machines, hoping that they would be intelligent in different ways than the computer programs could be. However, they usually simulate their invented machines on a computer and come to doubt that the new machine is worth building. Because many billions of dollars that have been spent in making computers faster and faster, another kind of machine would have to be very fast to perform better than a program on a computer simulating the machine.

### 1.4.6 Are computers fast enough to be intelligent?

Some people think much faster computers are required as well as new ideas. My own opinion is that the computers of 30 years ago were fast enough if only we knew how to program them. Of course, quite apart from the ambitions of AI researchers, computers will keep getting faster.

### 1.4.7 What about parallel machines?

Machines with many processors are much faster than single processors can be. Parallelism itself presents no advantages, and parallel machines are somewhat awkward to program. When extreme speed is required, it is necessary to face this awkwardness.

### 1.4.8 What about making a ``child machine'' that could improve by reading and by learning from experience?

This idea has been proposed many times, starting in the 1940s. Eventually, it will be made to work. However, AI programs haven't yet reached the level of being able to learn much of what a child learns from physical experience. Nor do present programs understand language well enough to learn much by reading.

### 1.4.9 Might an AI system be able to bootstrap itself to higher and higher level intelligence by thinking about AI?

I think yes, but we aren't yet at a level of AI at which this process can begin.

### 1.4.10 What about chess?

Alexander Kronrod, a Russian AI researcher, said ``Chess is the Drosophila of AI.'' He was making an analogy with geneticists' use of that fruit fly to study inheritance. Playing chess requires certain intellectual mechanisms and not others. Chess programs now play at grandmaster level, but they do it with limited intellectual mechanisms compared to those used by a human chess player, substituting large amounts of computation for understanding. Once we understand these mechanisms better, we can build human-level chess programs that do far less computation than do present programs.

Unfortunately, the competitive and commercial aspects of making computers play chess have taken precedence over using chess as a scientific domain. It is as if the geneticists after 1910 had organized fruit fly races and concentrated their efforts on breeding fruit flies that could win these races.

### 1.4.11 What about Go?

The Chinese and Japanese game of Go is also a board game in which the players take turns moving. Go exposes the weakness of our present understanding of the intellectual mechanisms involved in human game playing. Go programs are very bad players, in spite of considerable effort (not as much as for chess). The problem seems to be that a position in Go has to be divided mentally into a collection of subpositions which are first analyzed separately followed by an analysis of their interaction. Humans use this in chess also, but chess programs consider the position as a whole. Chess programs compensate for the lack of this intellectual mechanism by doing thousands or, in the case of Deep Blue, many millions of times as much computation.

Sooner or later, AI research will overcome this scandalous weakness. [11].

### 1.4.12 Don't some people say that AI is a bad idea?

The philosopher John Searle says that the idea of a non-biological machine being intelligent is incoherent. The computer scientist Joseph Weizenbaum says the idea is obscene, anti-human and immoral. Various people have said that since artificial intelligence hasn't reached human level by now, it must be impossible. Still other people are disappointed that companies they invested in went bankrupt. [11].

## 1.5 Areas of Artificial Intelligence Applications

AI applications of many, seemingly unrelated kinds are quietly being commercialized in greater and greater numbers. Not all these applications work as well as desired, but they are continually improving. These include:

- Automatic scheduling software to automatically create better project schedules faster
- Advanced learning software that works like human tutor in teaching one-on-one with each student
- Continuous speech recognition programs that accurately turn speech into text
- Software to manage information for individuals, finding just the documents immediately needed from amongst million of documents, and automatically summarizing documents
- Face-recognition systems

- Washing machines that automatically adjust to different conditions to wash clothes better
- Automatic mortgage underwriting systems
- Automatic investment decision makers
- Software that improves the prediction of daily revenues and staffing requirements for a business
- Credit fraud detection systems
- Help desk support systems that help find the right answer to any customer's question faster
- Shopping bots on the web
- Data mining tools
- E-mail filters
- Automated advice systems that personalize their responses.[2].

### 1.6 Summary

This chapter presented definitions, history and applications of Artificial Intelligence. Some of important points are as follows:

- Different people think of AI differently.
- Computer engineering provided the artifact that makes AI applications possible. AI programs tend to be large, and they could not work without the great advances in speed and memory that the computer industry has provided.
- The history of AI has had cycles of success, misplaced optimism, and resulting cutbacks is enthusiasm and funding. They have also been cycles of introducing new creative approaches and systematically refining the best ones.
- Recent progress in understanding the theoretical basis for intelligence has gone hand in hand with improvements in the capabilities of real system.
- Artificial intelligence was coined in 1956 by John McCarty at the Massachusetts Institute of Technology (MIT).

# CHAPTER TWO

# ARTIFICIAL NEURAL NETWORKS

## 2.1 Overview

This report presents what Artificial Neural Networks are, how they work, and where they are currently being used. This project is a result of an assignment in AI. The report is a non-technical report, thereby it does not go into depth with mathematical formulas, but tries to give a more general understanding

## 2.2 What are Artificial Neural Networks?

Artificial Neural Network is a system loosely modeled on the human brain. The field goes by many names, such as connectionism, parallel distributed processing, neuro-computing, natural intelligent systems, machine learning algorithms, and artificial neural networks. It is an attempt to simulate within specialized hardware or sophisticated software, the multiple layers of simple processing elements called neurons. Each neuron is linked to certain of its neighbors with varying coefficients of connectivity that represent the strengths of these connections. Learning is accomplished by adjusting these strengths to cause the overall network to output appropriate results. [7]. [12]

### 2.2.1 The Analogy to the Brain

The most basic components of neural networks are modeled after the structure of the brain. Some neural network structures are not closely to the brain and some does not have a biological counterpart in the brain. However, neural networks have a strong similarity to the biological brain and therefore a great deal of the terminology is borrowed from neuroscience.[7]

#### 2.2.1.1 The Biological Neuron

The most basic element of the human brain is a specific type of cell, which provides us with the abilities to remember, think, and apply previous experiences to our every action. These cells are known as neurons, each of these

neurons can connect with up to 200000 other neurons. The power of the brain comes from the numbers of these basic components and the multiple connections between them.

All natural neurons have four basic components, which are dendrites, soma, axon, and synapses. Basically, a biological neuron receives inputs from other sources, combines them in some way, performs a generally nonlinear operation on the result, and then output the final result. The figure below shows a simplified biological neuron and the relationship of its four components.[7]

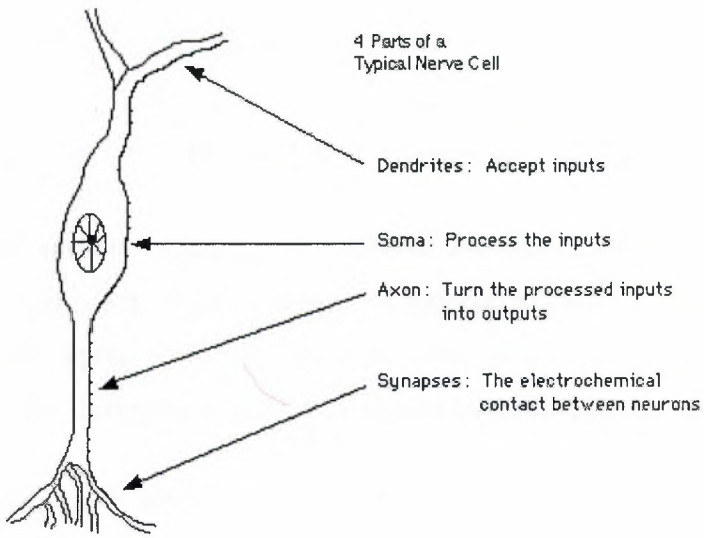



**Figure 2.1** Simple Neuron

### 2.2.1.2 The Artificial Neuron

The basic unit of neural networks, the artificial neurons, simulates the four basic functions of natural neurons. Artificial neurons are much simpler than the biological neuron; the figure below shows the basics of an artificial neuron.

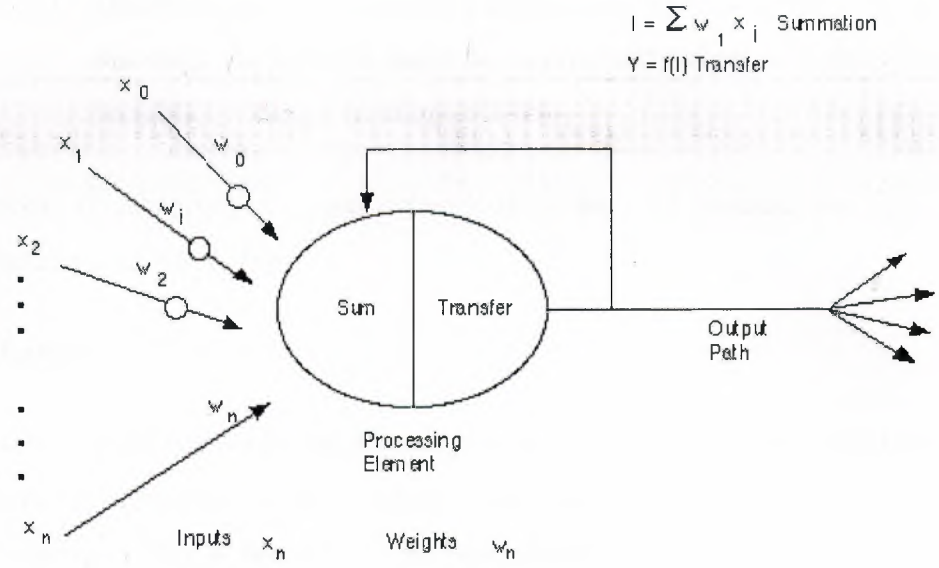



**Figure 2.2** A Basic Artificial Neuron

Note that various inputs to the network are represented by the mathematical symbol, x(n). Each of these inputs are multiplied by a connection weight, these weights are represented by w(n). In the simplest case, these products are simply summed, fed through a transfer function to generate a result, and then output.

Even though all artificial neural networks are constructed from this basic building block the fundamentals may vary in these building blocks and there are differences. [7]

### 2.2.2 Design

The developer must go through a period of trial and error in the design decisions before coming up with a satisfactory design. The design issues in neural networks are complex and are the major concerns of system developers.

**Designing a neural network consist of:**

- Arranging neurons in various layers.
- Deciding the type of connections among neurons for different layers, as well as among the neurons within a layer.
- Deciding the way a neuron receives input and produces output.

- Determining the strength of connection within the network by allowing the network learn the appropriate values of connection weights by using a training data set.

The process of designing a neural network is an iterative process; the figure below describes its basic steps.

### 2.2.2.1 Layers

Biologically, neural networks are constructed in a three dimensional way from microscopic components. These neurons seem capable of nearly unrestricted interconnections. This is not true in any man-made network. Artificial neural networks are the simple clustering of the primitive artificial neurons. This clustering occurs by creating layers, which are then connected to one another. How these layers connect may also vary. Basically, all artificial neural networks have a similar structure of topology. Some of the neurons interface the real world to receive its inputs and other neurons provide the real world with the network's outputs. All the rest of the neurons are hidden form view.

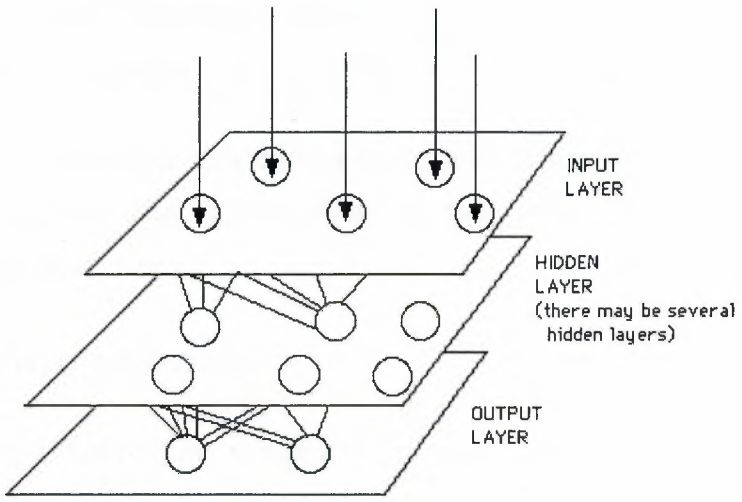



**Figure** 2.3 Input, Hidden, and Output Layers

As the figure above shows, the neurons are grouped into layers The input layer consist of neurons that receive input form the external environment. The output layer consists of neurons that communicate the output of the system to the user or external environment. There are usually a number of hidden layers between

these two layers; the figure above shows a simple structure with only one hidden layer.

When the input layer receives the input its neurons produce output, which becomes input to the other layers of the system. The process continues until a certain condition is satisfied or until the output layer is invoked and fires their output to the external environment.

To determine the number of hidden neurons the network should have to perform its best, one are often left out to the method trial and error. If you increase the hidden number of neurons too much you will get an over fit, that is the net will have problem to generalize. The training set of data will be memorized, making the network useless on new data sets.

#### 2.2.2.2 Communication and types of connections

Neurons are connected via a network of paths carrying the output of one neuron as input to another neuron. These paths is normally unidirectional, there might however be a two-way connection between two neurons, because there may be an another path in reverse direction. A neuron receives input from many neurons, but produce a single output, which is communicated to other neurons.

The neuron in a layer may communicate with each other, or they may not have any connections. The neurons of one layer are always connected to the neurons of at least another layer.

**a- Inter-layer connections**

There are different types of connections used between layers, these connections between layers are called inter-layer connections.

- **Fully connected**
  Each neuron on the first layer is connected to every neuron on the second layer.

- **Partially connected**

  A neuron of the first layer does not have to be connected to all neurons on the second layer.

- **Feed forward**

  The neurons on the first layer send their output to the neurons on the second layer, but they do not receive any input back form the neurons on the second layer.

- **Bi-directional**

  There is another set of connections carrying the output of the neurons of the second layer into the neurons of the first layer.

Feed forward and bi-directional connections could be fully- or partially connected.

- **Hierarchical**

  If a neural network has a hierarchical structure, the neurons of a lower layer may only communicate with neurons on the next level of layer.

- **Resonance**

  The layers have bi-directional connections, and they can continue sending messages across the connections a number of times until a certain condition is achieved.

**b- Intra-layer connections**

In more complex structures the neurons communicate among themselves within a layer, this is known as intra-layer connections. There are two types of intra-layer connections.

- **Recurrent**

  The neurons within a layer are fully- or partially connected to one another. After these neurons receive input form another layer, they communicate their outputs with one another a number of times before they are allowed to send their outputs to another layer. Generally some conditions among the neurons of the layer

should be achieved before they communicate their outputs to another layer.

- **On-center/off Surround**
- A neuron within a layer has excitatory connections to itself and its immediate neighbors, and has inhibitory connections to other neurons. One can imagine this type of connection as a competitive gang of neurons. Each gang excites itself and its gang members and inhibits all members of other gangs. After a few rounds of signal interchange, the neurons with an active output value will win, and is allowed to update its and its gang member's weights. (There are two types of connections between two neurons, excitatory or inhibitory. In the excitatory connection, the output of one neuron increases the action potential of the neuron to which it is connected. When the connection type between two neurons is inhibitory, then the output of the neuron sending a message would reduce the activity or action potential of the receiving neuron. One causes the summing mechanism of the next neuron to add while the other causes it to subtract. One excites while the other inhibits.)

#### 2.2.2.3 Learning

The brain basically learns from experience. Neural networks are sometimes called machine learning algorithms, because changing of its connection weights (training) causes the network to learn the solution to a problem. The strength of connection between the neurons is stored as a weight-value for the specific connection. The system learns new knowledge by adjusting these connection weights.

The learning ability of a neural network is determined by its architecture and by the algorithmic method chosen for training.

The training method usually consists of one of three schemes:

1. **Unsupervised learning**

   The hidden neurons must find a way to organize themselves without help from the outside. In this approach, no sample outputs are provided to the network against which it can measure its predictive performance for a given vector of inputs. This is learning by doing.

2. **Reinforcement learning**

   This method works on reinforcement from the outside. The connections among the neurons in the hidden layer are randomly arranged, then reshuffled as the network is told how close it is to solving the problem. Reinforcement learning is also called supervised learning, because it requires a teacher. The teacher may be a training set of data or an observer who grades the performance of the network results.

   Both unsupervised and reinforcement suffer from relative slowness and inefficiency relying on a random shuffling to find the proper connection weights.

3. **Back propagation**

   This method is proven highly successful in training of multilayered neural nets. The network is not just given reinforcement for how it is doing on a task. Information about errors is also filtered back through the system and is used to adjust the connections between the layers, thus improving performance. A form of supervised learning. [12]

**a- Learning laws**

There are a variety of learning laws which are in common use. These laws are mathematical algorithms used to update the connection weights. Most of these laws are some sort of variation of the best known and oldest learning law,

Hebb's Rule. Man's understanding of how neural processing actually works is very limited. Learning is certainly more complex than the simplification represented by the learning laws currently developed. Research into different learning functions continues as new ideas routinely show up in trade publications etc. A few of the major laws are given as an example below.

- **Hebb's Rule**

  The first and the best known learning rule was introduced by Donald Hebb [9]. The description appeared in his book The organization of Behavior in 1949. This basic rule is: If a neuron receives an input from another neuron, and if both are highly active (mathematically have the same sign), the weight between the neurons should be strengthened.

- **Hopfield Law**

  This law is similar to Hebb's Rule with the exception that it specifies the magnitude of the strengthening or weakening. It states, "if the desired output and the input are both active or both inactive, increment the connection weight by the learning rate, otherwise decrement the weight by the learning rate." (Most learning functions have some provision for a learning rate, or a learning constant. Usually this term is positive and between zero and one.)

- **The Delta Rule**

  The Delta Rule is a further variation of Hebb's Rule, and it is one of the most commonly used. This rule is based on the idea of continuously modifying the strengths of the input connections to reduce the difference (the delta) between the desired output value and the actual output of a neuron. This rule changes the connection weights in the way that minimizes the mean squared error of the network. The error is back propagated into previous layers one layer at a time. The process of back-propagating the

network errors continues until the first layer is reached. The network type called Feed forward, Back-propagation derives its name from this method of computing the error term. This rule is also referred to as the Windrow-Hoff Learning Rule and the Least Mean Square Learning Rule.

- **Kohonen's Learning Law**

  This procedure, developed by Teuvo Kohonen, was inspired by learning in biological systems. In this procedure, the neurons compete for the opportunity to learn, or to update their weights. The processing neuron with the largest output is declared the winner and has the capability of inhibiting its competitors as well as exciting its neighbors. Only the winner is permitted output, and only the winner plus its neighbors are allowed to update their connection weights.

  The Kohonen rule does not require desired output. Therefor it is implemented in the unsupervised methods of learning. Kohonen has used this rule combined with the on-center/off-surround intra-layer connection (discussed earlier under 2.2.2.2) to create the self-organizing neural network, which has an unsupervised learning method.

  On this Internet site by Sue Becker you may see an interactive demonstration of a Kohonen network, which may give you a better understanding. [7]. [12]

## 2.3 Applications of Neural Networks

Neural networks are performing successfully where other methods do not, recognizing and matching complicated, vague, or incomplete patterns. Neural networks have been applied in solving a wide variety of problems.

The most common use for neural networks is to project what will most likely happen. There are many areas where prediction can help in setting priorities. For example, the emergency room at a hospital can be a hectic place, to know who needs the most critical help can enable a more successful operation. Basically, all organizations must establish priorities, which govern the allocation of their resources. Neural networks have been used as a mechanism of knowledge acquisition for expert system in stock market forecasting with astonishingly accurate results. Neural networks have also been used for bankruptcy prediction for credit card institutions.

Although one may apply neural network systems for interpretation, prediction, diagnosis, planing, monitoring, debugging, repair, instruction, and control, the most successful applications of neural networks are in categorization and pattern recognition. Such a system classifies the object under investigation (e.g. an illness, a pattern, a picture, a chemical compound, a word, the financial profile of a customer) as one of numerous possible categories that, in return, may trigger the recommendation of an action (such as a treatment plan or a financial plan.

A company called Nestor, have used neural network for financial risk assessment for mortgage insurance decisions, categorizing the risk of loans as good or bad. Neural networks has also been applied to convert text to speech, NETtalk is one of the systems developed for this purpose. Image processing and pattern recognition form an important area of neural networks, probably one of the most actively research areas of neural networks.

An other of research for application of neural networks is character recognition and handwriting recognition. This area has use in banking, credit card processing and other financial services, where reading and correctly recognizing

handwriting on documents is of crucial significance. The pattern recognition capability of neural networks has been used to read handwriting in processing checks, the amount must normally be entered into the system by a human. A system that could automate this task would expedite check processing and reduce errors. One such system has been developed by HNC (Hecht-Nielsen Co.) for BankTec.

One of the best known applications is the bomb detector installed in some U.S. airports. This device called SNOOPE, determine the presence of certain compounds from the chemical configurations of their components.

In a document from International Joint conference, one can find reports on using neural networks in areas ranging from robotics, speech, signal processing, vision, character recognition to musical composition, detection of heart malfunction and epilepsy, fish detection and classification, optimization, and scheduling. One may take under consideration that most of the reported applications are still in research stage.

Basically, most applications of neural networks fall into the following five categories:

- **Prediction**

  Uses input values to predict some output. e.g. pick the best stocks in the market, predict weather, identify people with cancer risk.
- **Classification**

  Use input values to determine the classification. e.g. is the input the letter A, is the blob of the video data a plane and what kind of plane is it.
- **Data association**

  Like classification but it also recognizes data that contains errors. e.g. not only identify the characters that were scanned but identify when the scanner is not working properly.

- **Data Conceptualization**

  Analyze the inputs so that grouping relationships can be inferred. e.g. extract from a database the names of those most likely to by a particular product.

- **Data Filtering**

- Smooth an input signal. e.g. take the noise out of a telephone signal. [11]

### 2.4 Why Are ANNs Better?

1. They deal with the non-linearities in the world in which we live.

2. They handle noisy or missing data.

3. They create their own relationship amongst information - no equations!

4. They can work with large numbers of variables or parameters.

5. They provide general solutions with good predictive accuracy. [11]

### 2.5 Summary

This report presented what Artificial Neural Networks are, how they work, and where they are currently being used. This project is a result of an assignment in AI. The report is a non-technical report, thereby it does not go into depth with mathematical formulas, but tries to give a more general understanding.

# CHAPTER THREE

# GENETIC ALGORITHMS

## 3.1 Overview

After scientists became disillusioned with classical and neo-classical attempts at modeling intelligence, they looked in other directions. Two prominent field arose, connectionism (neural networking, parallel processing) and evolutionary computing. Genetic algorithms are a part of evolutionary computing, which is a rapidly growing area of artificial intelligence. As you can guess, genetic algorithms are inspired by Darwin's theory about evolution. Simply said, solution to a problem solved bye genetic algorithms is evolved.

## 3.2 What are Genetic Algorithms?

Inspired by Darwin's theories of evolution, Genetic Algorithms (GAs) are computer programs which create an environment where populations of data can compete and only the fittest survive.

A GA generates populations of possible scenarios, evaluates them, and then mates them with others (crossover) . Some of the pieces are randomly altered (mutated). These scenarios are then passed to an Excel spreadsheet for evaluation. By repeating this process over many generations the GA can breed optimal solutions to problems. [11]

### 3.2.1 What Can Genetic Algorithms Do?

GAs are excellent for all tasks requiring optimisation especially those requiring optimal mixing, ordering, budgeting and scheduling. They are highly effective in any situation where many inputs (variables) interact to produce a large number of possible outputs (solutions).

### 3.2.2 What Applications Can Genetic Algorithms Be Used For?

**Management:** Distribution, scheduling, project management, courier routing, container packing, task assignment, time tables.

**Financial:** Portfolio balancing, budgeting, forecasting, investment analysis and payment scheduling.

**Engineering:** Structural design (eg beam sizes), electrical design (eg circuit boards), mechanical design (eg optimise weight, size & cost), process control, network design (eg computer networks).

**R & D :** Curve and surface fitting, neural network connection matrices, function optimisation, fuzzy logic, population modeling, molecular modeling and drug design.

### 3.2.3 Who Uses Genetic Algorithms?

American Express, AT&T, Bell Labs, Exxon, GE, General Motors, IBM, Intel, John Deere, Lockheed, Motorola, MTV, NASA, Sears, Xerox, Australian Navy, ANSTO, Monash Uni, North Sydney Leagues Club and many more.

## 3.3 Definition of Genetic Algorithms

A genetic algorithm is a form of evolution that occurs on a computer. An algorithm is a general description of a procedure, and a program is its realization as a sequence of instructions to a computer. Genetic algorithms are a search method that can be used for both solving problems and modelling evolutionary systems. The basic idea of a genetic algorithm is very simple. First, a population of individuals is created in a computer (typically stored as binary strings in memory, and then the population is evolved with use of principles of variation, selection and inheritance. There are many ways of implementing this simple idea .

With various mapping techniques and an appropriate measure of fitness genetic algorithms can be tailored to evolve a solution for many types of problems, including optimization of a function or determination of the proper order of a sequence. Real world problems that can be represented in a spreadsheet, such as scheduling people, designing layouts for garment cutting, preparing budgets, organizing workers according to their skill rating and modeling chemical reaction conditions have all been successfully optimized with genetic algorithms. Mathematical analysis has begun to explain how genetic algorithms work and how best to use them.

By harnessing the mechanisms of evolution researchers may be able to "breed" programs that solve problems with ill defined structure. Genetic algorithms make it possible to explore a far greater range of potential solutions to a problem than do conventional programs. [11]

### 3.4 Genetic Algorithms and Neural Networks

Genetic algorithm and neural networks are both considered "artificial intelligence" technologies, and are often lumped together because both were inspired by biological systems in nature; the genetic algorithm is to evolution what the neural network is to the brain.

The two techniques usually solve different types of problems, and therefore should not be seen as competing. Much research is under way exploring how the two techniques can complement each other.

There are many parameters to set when developing a neural network, all of which can affect its accuracy. How do we know how many hidden neurons should we set up? How many layers? What about the learning rate? If you want to maximize the performance of your neural network, but you are not an expert at developing neural networks, you may want to explore the possibility of using genetic algorithm to set up optimal parameters. The genetic algorithm creates a neural network randomly (within specifications). The network is then trained on sample data, and tested on sub-set data to check performance. The difference between the outcome of the neural network and the actual output is computed, and this "error" value is recorded along with the set of parameters that produced it. This process continues, creating a population of neural networks, and the genetic algorithm uses the error as the target value to minimize. Because the calculation of each test takes a while the process can be slow, but the result is a more highly evolved neural network, customized to your problem.

### 3.5 Smartening up the Genetic Search

A traditional approach to solving for an optima in a relatively uniform landscape has been to use a hill climbing routine i.e. in order to find a maximum solution in a locality proceed in a direction that returns an increase in elevation and continue until all further steps are down. By incorporating hill climbing into the genetic algorithm search routine

optimal solutions can be attained faster than by genetic algorithm alone. Studies have shown that such genetic algorithm/hill climbing hybrid techniques often outperform either method operating alone. Since a genetic algorithm can be slow to fine-tune, this method more quickly determines whether the genetic algorithm is focusing on the most promising path.

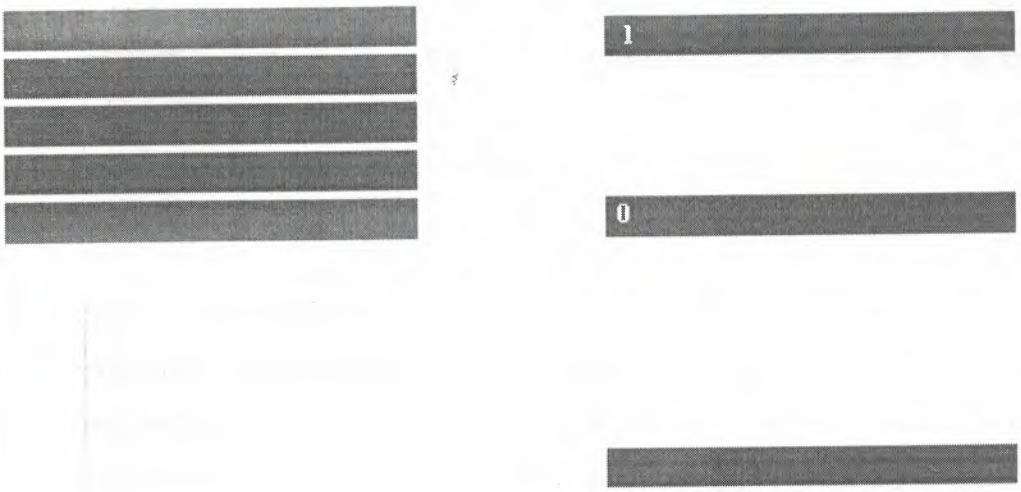



Figure 3.1 Hill Climbing Hybrid Techniques

If the evaluation function in the problem is too noisy or discontinuous for conventional hill-climbing techniques, you might try a local genetic algorithm to fine-tune the organism. By creating a very small population of similar organisms around the offspring and localizing the mutation and crossover operators to favor small over large changes, these local genetic algorithm will behave more like hill-climbing algorithms.

### 3.5.1 Species development

The traditional genetic algorithm forces your organisms to evolve in one big population. Arbitrarily dividing the population into several mini-populations will not increase performance and may even hinder the genetic algorithm's ability to converge. This is generally true because there is a good chance that each population will evolve towards similar solutions, causing a lot of unnecessary duplication of effort.

A more effective way to boost a genetic algorithm's performance is to create a species-generating genetic algorithm (SGGA). Species are sub-populations that are automatically created whenever a genetic algorithm detects two significantly different areas being explored within its population. To determine when the population is significantly diverse enough to merit a split, we can employ cluster analysis or generate a new species when the relative distance between the highest ranking organisms are crossed at some given threshold. Each organism can be assigned to the species with which it is most closely associated. These distinct species generally evolve independently of one another, although they can occasionally swap a few of their better organisms.

Several advantages are evident with distinct species development. Firstly, the single-population algorithm wastes time breeding organisms of similar rank, regardless of the relative distances from one another. By removing the mutual burdens these organisms place on each other, the SGGA lets significantly different local areas be explored to their full potential. The SGGA performs especially well as the number of dimensions in the objective function increases.

The SGGA also lets the independent species converge more quickly and can give an earlier prediction of an areas promise. Poorly performing species can be killed or remerge with similar species to free up more computation time for more promising species.

### 3.5.2 Soft Constraints

Real world problems involve constraints - preparing schedules, budgets or designing engineering structures all have parameters to meet before optimization can be met. It may seem logical to prevent the genetic algorithm from considering invalid solutions. Such "hard" constraints are useful in traditional techniques, but with a genetic algorithm, if we throw out an organism because it violates a particular constraint we may be excluding vital information that could lead to a optimal solution. For example a truly optimal truss design may support only 9.9 ton, but also use only half the material of the next best design with 10 ton set as a hard constraint. For this reason it is generally accepted that replacing hard constraints with well chosen "soft" constraints will increase the genetic algorithm's performance. Soft constraints do not prevent the genetic

algorithm from exploring invalid solutions; they simply discourage the organism from exploring in that region.

### 3.5.3 Backtracking

In some problems, certain constraints have to be met and the search space is so limited that even a genetic algorithm with severe penalties would waste much time exploring invalid solutions. Genetic backtracking offers a simple way to ensure that all hard constraints are met quickly while still maintaining the diversity needed to fuel the genetic algorithm.

A traditional hard constraint method might evaluate each new offspring organism first by checking a list of given constraints. If the organism has violated any constraints, the process creates a new offspring until an acceptable solution is found. In a highly constrained problem, much computation may be wasted in finding one valid organism. Even when a couple of valid organisms are found, they most likely will combine to produce invalid organisms. Alternately, the backtracking method checks each offspring with the list of hard constraints. However, if the offspring solution violates any constraints, the organism backtracks or "de-evolves" towards its parents' genetic material until all constraints are met.

Functionally, genetic backtracking continually mates the original parents with increasingly lower crossover and mutation rates. Each time the organism backtracks, it becomes more similar to one of its parents. Since both parents meet the constraints, the organism eventually will backtrack into a valid, constraint meeting solution space.

The backtracking loop is a relatively quick procedure. Backtracking is faster than harsh penalty functions when a region of acceptable solutions is small or scattered. In addition, backtracking is the best method if your constraints must be met.

Most genetic algorithm implementations have only a few parameters to tweak: population size, crossover rate, and mutation rate. While default values can be used to find good results for a wide range of problems, you often can double the speed at which some problems are solved by fine-tuning these parameters to the right values.

The difficult part is that the right values can change drastically with just a few modifications to your model. Ideal parameters can change as the optimization progresses. For example, crossover is often most important in the early stages of a genetic algorithm, while mutation is important for later fine tuning.

In 1991 Lawrence Davis [10] introduced auto scaling to crossover and mutation rates. A simpler method is to plot the best performing organism after each trial. A curve fitting routine then determines the rate of improvement, adjusting crossover and mutation appropriately. The population size should also be varied throughout the evolutionary process to balance speed of evolution with gene diversity. The adaptive genetic algorithm automatically grows the population (and increases gene diversity) when progress is slow and, conversely, shrinks heterogeneous populations.

While this seems a trifle esoteric, genetic algorithms are proving useful in solving problems with hundreds of thousands of choices. The General Electric Corporate Research and Development Center in N.Y. are faced with the task of designing a more efficient jet engine. Refinements that reduce engine fuel consumption by as little as 1 or 2 percent have a competitive edge. The engineers at GE claim that the use of a genetic algorithm decreased the development time significantly for the project. " We couldn't have afforded to do the thousands and thousands of design permutations otherwise".

Optimizing aircraft engine design isn't the only application of genetic algorithms. Before a micro-electronic chip is permanently etched into silicon it can be simulated in a computer. Circuit patterns for a microprocessor can be laid out and joined in millions of ways. Designers are more effective if they can use programming tools that identify the most efficient patterns.

Genetic algorithms can be likened to fishing. If there are no fish out there it continues to cast until it finds a better place to fish. What makes this kind of optimization different from traditional ones is its adaptability. It doesn't get caught in a local area when it bumps into a corner. It's smart enough to turn 180 degrees and try something new.

Genetic algorithms can also fine tune information coming from a seemingly chaotic environment. The U.S. Navy use a package developed to separate a noisy background from a true sonar signal.

For many organizations scheduling staff can be a loathsome task requiring many hours of juggling one group of people to fit in with another group. Ask any school teacher who has had to prepare a class timetable for the school term and you will find many hours of work and often unsatisfactory results. A genetic algorithm can meet constraints, such as, Biology 1 at 10:00 on Tuesday or Science 2 not with Chemistry 2 and then go on to fit all the classes in and not leave anyone standing in the corridors. By trying thousands of combinations and presenting several possible solutions all people can be satisfied.

The travelling salesman problem is a classic example of genetic algorithms at work. A salesman, or courier, has for example, 50 destinations to cover and needs to travel a minimum distance to save time and travel costs. He may need to visit one town before another. By knowing the distances between towns a spreadsheet can be employed with the aid of a genetic algorithm to plan the route that will minimize the distance travelled.

The genetic algorithm is rapidly becoming a popular alternative to traditional techniques. Without any customization these simple algorithms can solve highly complex problems. [11].

### 3.6 Using Genetic Algorithms in Computer Learning

Essentially, Genetic Algorithms (GA) are a method of "breeding" computer programs and solutions to optimization or search problems by means of simulated evolution. Processes loosely based on natural selection, crossover, and mutation are repeatedly applied to a population of binary strings which represent potential solutions. Over time, the number of above-average individuals increases, and better fit individuals are created, until a good solution to the problem at hand is found.

In NNUGA, we are using GA to find the solution to a classification problem with a neural network (NN). The neural network is a structure which is able to respond with True (1) or False (0) to a given input vector. We are trying to "teach" our neural network to correctly classify a set of input vectors, which can be thought of as learning a concept. We then expect that when the neural network will be presented with a vector P not from this set, it will tend to exhibit generalization by responding with an output similar to target vectors for input vectors close to the previously unseen input vector P.

Our genetic algorithm works with not one but multiple populations, all of which evolve seperately most of the time, except for once every several generations where we allow different populations to mix with each other.

### 3.7 The Neural Network

Our neural network has 2 inputs and 1 output. It is constructed from 6 neurons in the first layer and 1 neuron in a second layer.

In the first layer we have 2 neurons with atan transfer functions:

$$\text{atan}( P * W + b )$$

2 with linear transfer functions:

$$P * W + b$$

and 2 with hardlim transfer functions:

$$P * W + b > 0$$

where P is the input vector presented to the network, W is the vector of weights and b is the bias of the neuron.

The output function of the neuron in the second layer is:

$$A * W + b > 0$$

where A is the vector of the outputs of the neurons in the first layer, W is the vector of weights and b is its bias. The output of this neuron is also the output of the network.

#### 3.7.1 The Learning Method

As stated earlier, the neural network learns using genetic algorithms. This means that the weights and biases of all the neurons are joined to created a single vector. A certain

set of vectors is a correct solution to the classification problem at hand. We are hoping to find one of these vectors using an evolutionary algorithm.

**The Canonical GA (pseudo code):**

```
choose initial population
evaluate each individual's fitness
repeat
    select individuals to reproduce
    mate pairs at random
    apply crossover operator
    apply mutation operator
    evaluate each individual's fitness
until terminating condition
```

The learning loop can terminate either if a satisfactory solution is found, or the number of generations pass a preset limit, suggesting that a complete solution will not be found with this set of individuals.

In our case, we have several seperated populations, all of which evolve seperately. Once every several generations we allow cross-over from different populations.

### 3.7.2 Limitations

Sometimes it could happen that though the NN could theoretically solve a certain classification problem, our system will not return a correct solution. This is because of the random nature of the algorithm and its reliance on natural selection, mutation and cross-overs. Naturally, it could happen that a certain flow of events that would lead to a correct solution will not occure and thus a solution will not be found. However, by using several unrelated populations we have decreased the probabilty of this happening, since if some population has poor individuals the solution could still be found at another.

### 3.7.3 By Omri Weisman and Ziv Pollack Implementations

They implemented a network with 2 inputs. The input for the neural network can be taken from a graphic user interface, by clicking on points in a board. A click with the left mouse button generates a '+' sign on the board, marking that it's a point where the NN should respond with 'True'. A click with the right mouse button generates a '-' sign on the board, marking that it's a point where the NN should respond with 'False'. When enough points have been entered, the user can click on 'Start', which will introduce these points as inputs to the NN, have it learn these input vectors and show a group of lines which corresponds to the division of the plane into regions of opposite neuron response. While the learning takes place, a textual indication of the learning process is presented on the standard output; This includes the fitness of the best individual in each population on each generation, and a schematic textual division of the plane once every 50 generations, to allow the user to inpect the progress. [8].

Here's an example of a screen shot:

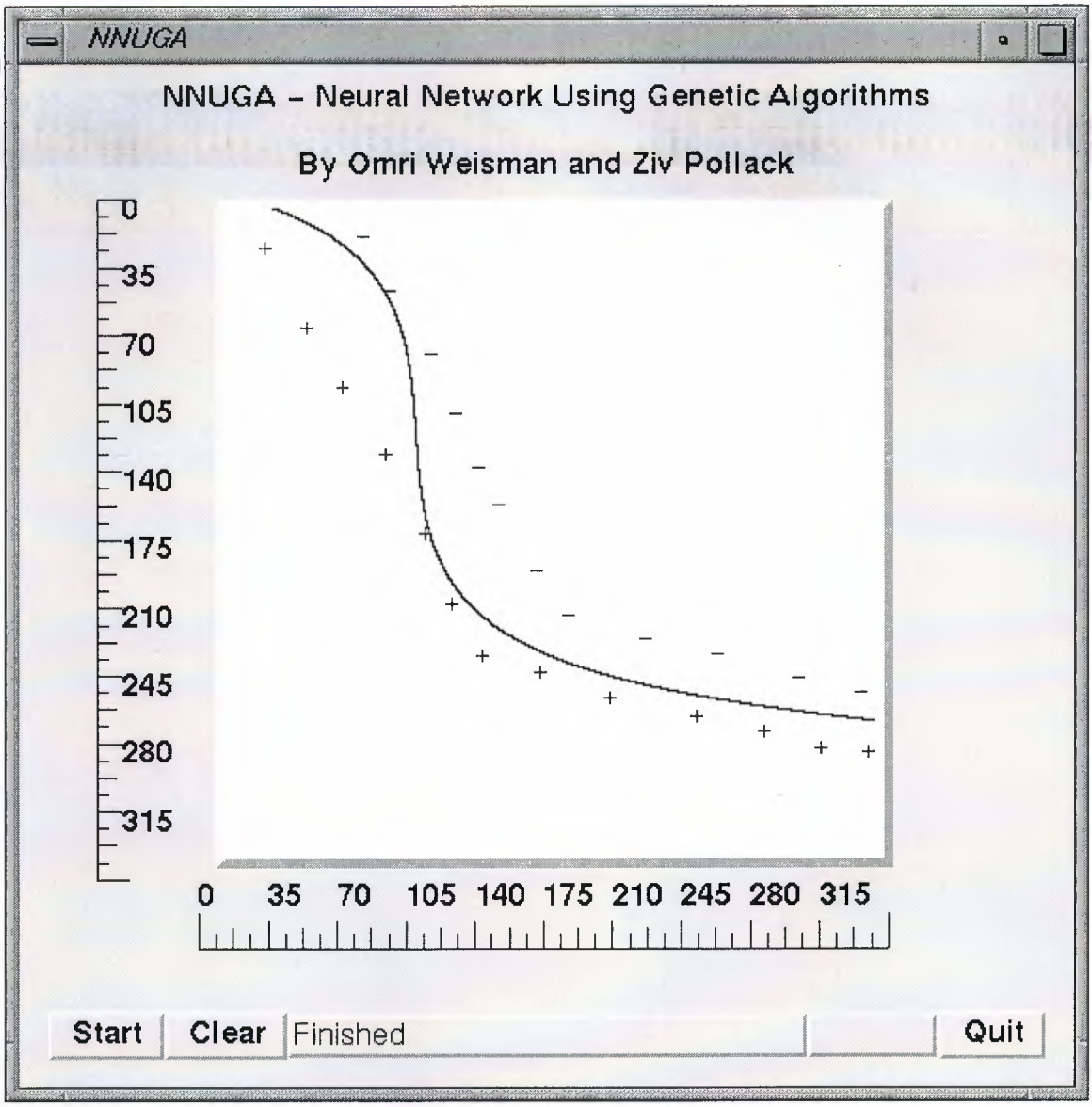



**Figure 3.2** Screen Shot

### 3.8 Applications of genetic Algorithm

The possible applications of Genetic Algorithms are immense. Any problem that has a large searchdomain could be suitable tackled by GAs. A popular growing field is genetic programming(GP).

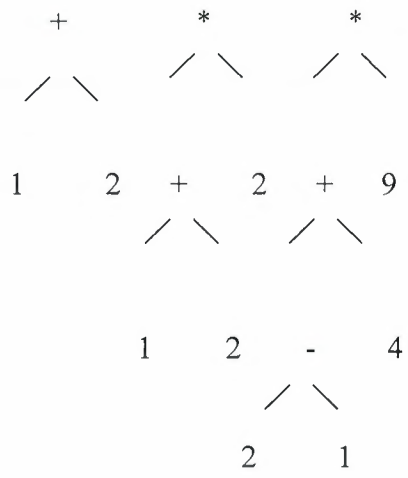
### 3.8.1 Genetic Programming

In programming languages such as LISP and Scheme, the matemathical notation is not written in standart notation, in prefix notation. Some examples of this:

+ 2 1 : 2 + 1

+ 2 1 2 : 2 * ( 2 + 1 )

* + - 2 1 4 9 : 9 * ( ( 2 – 1 ) + 4 )




Notice the difference between the left-hand sides to the right? Apart from the order Being different, no parenthesis! The prefix method makes life a lot easier for Programmers and compiler alike, because order predence is not an issue. You can build expression trees out of these strings that then can be easily evaluated, for example, here are the trees for the above three expression.

You can see how expression evaluation is thus a lot easier. What these have to do with GAs? If for example you have numerical data and 'answers', but no expression to conjoin the

Data with the answers. A genetic algorithm can be used to 'evolve' an expression tree to create a very close fit to the data. By 'splicing' and 'grafting' the trees and evaluating

the resulting expression with the data and testing it to the answers, the fitness function can return how close the expression is. The limitations of genetic programming iie in the huge search space the G.As have to search for an infinite numbers of equations. Therefore, normally before running a GA to search for equation, the user tells the program which operators and numerical ranges to search under. Uses of genetic programming can lie in stock market prediction, advanced mathematics and military. [1]

### 3.9 Summary

The development of a design theory for selector combinative GAs. The design and implementation of component GA selector combinative GAs, GAs using selection and recombination that solve hard problems quickly, reliably, and accurately.

Current plans call for the continuation of these efforts in GA design theory, competence, efficiency, and application together with new or renewed initiatives to carry the design methodology to genetic programming (G.P.). Additionally, an increased emphasis on solving important problems in computational biology, data mining, genetic algorithms, genetic programming.

# CHAPTER FOUR

# APPLICATIONS OF ARTIFICIAL INTELLIGENCE

## 4.1 Overview

This chapter presents an application of Artificial Intelligence.This Artificial Intelligence application is test the Eliza and Parry.

## 4.2 An Application of Artificial Intelligence Techniques to a Consumer Software Product

### 4.2.1 The Problem of Installation

PCMCIA cards (now more properly known as "PC Cards") have become very popular with portable notebook computer manufacturers and end-users alike; although originally intended as a new style of solid state memory storage, their acceptance really came as peripheral devices such as modems, LAN cards, and rotating disks became popular. Even though the PCMCIA organisation set up a technical committee and software subcommittees to architect software abstraction layers to ease the problem of software drivers and applications interacting with PC Cards, the legacy nature of the PC architecture from its inception in the early 1980s by IBM has lead to many problems when trying to install software drivers for such cards.

Typically a device such as a modem or Ethernet card requires resources such as an available hardware interrupt (IRQ), and an open memory and I/O address range. Unfortunately the original IBM PC architecture allowed for little standardisation for the allocation of such resources (people in the universities, generally using the architecturally purer Apple Macintosh, may not be aware of such problems), it being done in an ad hoc manner by the end-user, in a trial and error style using DIP switches on the plug-in hardware boards and modifying installation parameters for the software drivers. The problem of resource allocation has been compounded with Microsoft's Windows architecture often sitting on top of the DOS operating system and the underlying PC hardware architecture. [13]

### 4.2.2 An Example of an Ethernet PC Card

Until recently, an end-user would have to attempt to install a driver for an Ethernet PC Card in the following type of manner, in the case of, say, installing for a Novell network. An ODI driver is required to interface to the Ethernet card; this hardware interfacing ultimately has to be accomplished by communicating with the card through either I/O ports or a memory region. Also, a hardware interrupt is required for the card to be able to signal events to the driver.

Unfortunately, the IBM PC's I/O and memory maps are far from well defined, and although many areas are generally understood to be reserved for certain types of devices or uses, the holes are usually a free-for-all and prone to conflicts. For example, the famous bottom 640K of memory for an IBM PC in real mode is reserved for RAM, the top 64K is reserved for the ROM BIOS, and some parts in between are regarded as reserved for ROM BIOS extensions and video controllers, but the usage of the areas remaining (generally between hexadecimal memory addresses 0xC0000 and 0xEFFFF) is not well controlled. In the same way the allocation of IRQs and I/O ports, although also partly adhering to some convention (such as I/O ports 0x3F8-3FF being reserved for COM port 1), is also not well controlled.

Pity the poor end-user who would thus, typically, have to manually edit a Novell-defined text file net.cfg containing configuration information for the driver's setup, on a trial-and-error basis until a working solution had been found (and having to load and unload device drivers at the same time as each new configuration was tried). Here is an example of such a net.cfg file:

```
link driver pccard
  frame ethernet_802.2
  int 5
  port 320
  mem d2000
```

### 4.2.3 A First Pass at Easing the Installation Problem

I was involved in architecting a new system for easing the installation problem. Not only is this problem a significant one for experienced Information Systems personnel

who historically have been responsible for installing such new hardware and software devices, but even more so now with PC Cards becoming popular consumer products, and thus purchased by novice end-users.

Our first version used the basic principle of the core installation code, hidden behind either a modern Windows or DOS graphics front-end, procedurally (in 'C') examining system resources and attempting to make informed procedural decisions about their availability or otherwise. This method was extremely successful, but prone to the expected problem of difficulty of maintainability.

### 4.2.4 A Second Attempt, Using AI Techniques

We decided to attempt to separate the two areas of (a) resource allocation analysis and (b) free resource analysis, and abstract out the task of a rules based system which had previously been too embedded in conventional procedural code. We found that resource allocation analysis could be performed very effectively using conventional C or C++ techniques , albeit still with much embedded understanding of PC hardware and operating system architectures and resource usage. We also discovered a very effective way of attempting to ascertain a free set of required resources: the resource allocation analysis system would create a set of applicable rules (e.g. a file containing a set of Prolog predicates) for the chosen rules-based Expert System (e.g. Prolog), which was then fed into the Expert System engine along with our own set of precompiled rules-based constraints, again based on an understanding of PC hardware and operating system architectures and resource usage. In this manner we found that the Expert System had an extremely high success rate of determining correct resource parameters, with the added bonus of being much easier to maintain (and not just by the original development team, as it turned out). The rest of the installation software could then automatically create configuration files (such as net.cfg) for the end-user, there being very little or no end-user interaction at all in the whole process.

### 4.3 Eliza

Eliza, the computer therapist, is probably the most famous AI program yet created. Eliza was created in 1966 at MIT by Joseph Weizenbaum. Weizenbaum was surprised and even horrified to find that many of Eliza's interviewees would form strong emotional bonds with her. Psychiatrists were ready to begin letting Eliza treat their patients, and people were calling Weizenbaum to ask for Eliza's help in sorting out their problems. Even Weizenbaum's secretary insisted on making the rest of the staff leave the room so that she could talk to Eliza alone. Weizenbaum found the bonds people were forming with his creation 'obscenely misplaced'. He had not realized when creating Eliza that, "extremely short exposures to a relatively simple computer program could induce powerful delusional thinking in quite normal people."

Eliza functions by "twisting the statements of her 'patients' back at them in the classic manner of a non-directive psychotherapist." Following is an excerpt from Eliza's conversation with a teenager. The teenager speaks first. Eliza's responses are in red.

Men are all alike.
IN WHAT WAY
They're always bugging us about something or other.
CAN YOU THINK OF A SPECIFIC EXAMPLE
Well, my boyfriend made me come here.
YOUR BOYFRIEND MADE YOU COME HERE
He says I'm depressed much of the time.
I AM SORRY TO HEAT THAT YOU ARE DEPRESSED
It's true that I am unhappy.
DO YOU THINK COMING HERE WILL HELP YOU NOT BE UNHAPPY
I need some help, that much seems certain.
WHAT WOULD IT MEAN TO YOU IF YOU GOT SOME HELP
Perhaps I could learn to get along with my mother.
TELL ME MORE ABOUT YOUR FAMILY
My mother takes care of me.
WHO ELSE IN YOUR FAMILY TAKES CARE OF YOU
My father.
YOUR FATHER

You are like my father in some ways.

WHAT RESEMBLANCE DO YOU SEE

You are not very aggressive but I think you don't want me to notice that.

WHAT MAKES YOU THINK I AM NOT VERY AGGRESSIVE

You don't argue with me.

WHY DO YOU THINK I DON'T ARGUE WITH YOU

You are afraid of me.

DOES IT PLEASE YOU TO BELIEVE I AM AFRAID OF YOU

My father is afraid of everybody.

WHAT ELSE COMES TO MIND WHEN YOU THINK OF YOUR FATHER

Bullies.

DOES THAT HAVE ANYTHING TO DO WITH THE FACT THAT YOUR BOYFRIEND MADE YOU COME HERE

Weizenbaum's experience with Eliza and society's response to her, ultimately left him apposed to idea of constructing artificial intelligences. Weizenbaum's fear was that, despite artificial intelligences' inability to fully understand or sympathize with the human condition, society would be all to ready to entrust artificial intelligences with the task of managing human affairs.

With all of the hype surrounding Eliza, it's surprising to find that Eliza doesn't actually posses any of the qualities usually associated with intelligence. Eliza, knows nothing about her environment. She has no reasoning ability or understanding of her own motives. She can't plan out her actions and she can't learn. Following is an excerpt from a conversation with Eliza which show her limitations.

HOW DO YOU DO. PLEASE STATE YOUR PROBLEM

I'm wondering whether computers can think.

WHY DO YOU MENTION COMPUTERS

Since you are one I thought I would ask you - can you think?

DID YOU THINK THEY MIGHT NOT BE ONE YOU THOUGHT YOU WOULD ASK I -CAN I THINK

Let's talk just about you - not me. Can you think?

YOU LIKE TO THINK I - NOT YOU - DON'T YOU

Pardon?

WHAT DOES THAT SUGGEST TO YOU?

What does is suggest to you?

TELL ME ABOUT WHAT DOES IT SUGGEST TO I

That's nonsense - which of us do you mean by I?

WHY DO YOU THINK I MEAN BY YOU

The question meant to which of us is the word referring.

SUPPOSE THE QUESTION MEANT TO WHICH OF US WERE NOT THE WORD REFERRING

I think you are an extremely stupid program.

PERHAPS YOU WOULD LIKE TO BE AND EXTREMELY STUPID PROGRAM

Despite her short comings, Eliza was still the closest thing to artificial intelligence at the time. [12].

**4.4 Parry**

Parry, the paranoid program was created by psychiatrist Kenneth Colby as part of an attempt to understand the process that underlie paranoia. Interviewers of Parry often had real difficulty telling Parry from a real person. Following is as excerpt from a conversation between Parry and a psychiatrist. The psychiatrist speaks first. Parry's responses are in red.

Why are you in the hospital?

I SHOULDN'T BE HERE.

Who brought you here?

THE POLICE.

What trouble did you have with the police?

COPS DON'T DO THEIR JOB.

What do they do instead?

COPS ARREST THE WRONG PEOPLE.

They arrest the right ones sometimes.

THE LAW SHOULD GET THOSE ITALIAN CROOKS.

Are you Italian?

MY BACKGROUND IS BRITISH BUT I WAS BORN IN THIS COUNTRY.

Do you know any Italian crooks?

I TRY TO AVOID THE UNDERWORLD.

Are you afraid of the underworld?

THE MAFIA ARE THE WORST GANGSTERS.

Are you connected with the Mafia?

YOU KNOW, THEY KNOW ME.

How do they know you?

THEY SPY ON ME.

Why do they?

THEY WANT TO GET FALSE EVIDENCE TO FRAME ME.

What do they have against you?

THEY ARE OUT TO GET ME.

What for?

I HAVE INCRIMINATING EVIDENCE AGAINST THE MAFIA.

Are you sure of all this?

YOU DON'T BELIEVE ME.

I am doing my best.

Parry remains silent.

The psychiatrist remains silent.

CAN'T YOU THINK OF A DECENT QUESTION TO ASK?

You seem hostile to me.

SOME OF YOUR REMARKS ANNOY ME.

I am sorry.

FORGET THE EXCUSES, YOU GUYS ARE ALL ALIKE. [12].

**4.5 Summary**

This chapter presented an application of Artificial Intelligence. This Artificial Intelligence application tested the Eliza and Parry.

# CONCLUSION

In this project, systems are continuously being mentioned; but what are systems? Systems can be defined as a part of the universe, containing parts that have stronger correlations with each other than with the rest of the universe. While looking for this definition, we found that there are really two types of causes. One type is related to the changes caused by an intelligent system, and the other type is related to those changes occuring in nature, where one occurence only has a correlation with another.

In studying "intelligence", we have not limited ourselves to one aspect, such as vision, problem solving, but we have studied the total intelligent system. This system has senses, objectives, a selection of responses from memory, a possibility to act on its environment, and finally the ability to learn from its experiences. Thus intelligence, in the sense that we defined it is basically a stimulus-response mechanism, but with a selection of responses according to an object. It is a way to choose an adequate response to a given situation, a response that brings the system nearer to its objective.

It is amazing how little an A.I.S., can really know about it environment. The IS builds up an internal representation of it environment to the best of its ability and in doing so, creates its own concepts, it is of the utmost importance that incoming information be checked. We discovered that while sensory information is often limited, it is (mostly) reliable. Information received from anotherA.I.S., however, is often incorrect, and most often unintentionally so. It must thus be carefully checked.

The study of A.I.S. has also shown what their "world view" could be. Again, it is seen that what is true for the A.I.S. is also true for human beings.

Finally, it is useful to learn from the A.I.S. and thereby increase our own intelligence.

It is amazing to observe how a small thing such as a rigorous dafinition of an intelligent system can have ramifications on our understanding of societies, ethics. And philosophical outlook.

In chapter one, two important questions be asked: are you concerned with thinking or behavior? Do you want to model humans, or work from an ideal standard?

Computer engineering provided te artifact that mkes A.I. applications possible. A.I. programs tend to be large, and they could not work without the great advances in speed and memory that the computer industry has provided.

Does humanity's new knowledge about artificial intelligence systems affects its future? The knowledge about A.I.S. is currently rudimentary and it looks like many years will be required to understannd intelligent systems fully. Perhaps in a hundred years, we will still be increasing our knowledge on intelligent systems.

In chapter two, "Artificial Neural Networks" are based on the parallel architecture of animal brains. Neural Networks will be the future in computing. Real brains are orders of magnitude more complex than any other artificial neural networks so far considered. A great deal of research is going on in neural networks worldwide.

In chapter three, "Genetic Algorithms" are not too hard to program or understand, since they are biological based. Thinking in terms of real-life evolution may help you to understand and there is a general algorithm for th genetic algorithms.

In chapter four, "Applications" every application of these types of artificial intelligence do well in its own job, so applications that are mentioned in this chapters are not the all applications in the life there are many different applications.

# REFERENCES

[1]. Nashaat Ghuneim. Graduation Project Book. 2002

[2]. Arthur R. Jensen. Does IQ matter? Commentary, pages 20-21, November 1998. The reference is just to Jensen's comment--one of many.

[3]. Alan Turing. Computing machinery and intelligence. Mind, 1950.

[4]. John McCarthy. Defending AI research : a collection of essays and reviews. CSLI lecture notes: no. 49. Center for the Study of Language and Information, 1996. distributed by Cambridge University Press.

[5]. Daniel Dennett. Brainchildren: Essays on Designing Minds. MIT Press, 1998.

[6]. Timeline

Cohen, Jonathan. Human Robots in Myth and Science. NY: A.S.Barnes, 1967.

Feigenbaum, E.A. & Feldman, J. (eds.) Computers and Thought. NY: McGraw-Hill, 1963.

Gardner, Martin. Logic Machines & Diagrams. NY: McGraw-Hill, 1958.

McCorduck, Pamela. Machines Who Think. San Francisco: W.H. Freeman, 1979.

AAAI. AI Topics: History of AI at www.aaai.org/aitopics/html/history.html

[7]. Artificial Neural Networks

Data & Analysis Center for Software, "Artificial Neural Networks Technology", 1992

(http://www.dacs.dtic.mil/techs/neural/neural.title.html,printed November 1998)

Avelino J. Gonzalez & Douglas D. Dankel, "The Engineering of Knowledge-based Systems", 1993 Prentice-Hall Inc. ISBN 0-13-334293-X.

Fatemeh Zahedi, "Intelligent Systems for Business: Expert Systems with Neural networks, 1993 Wadsworth Inc. ISBN 0-534-18888-5.

Haykin Simon, "Neural Networks", 1994 Macmillan College Publishing Company Inc. ISBN 0-02-352761-7

[8]. Omri Weisman and Ziv Pollack
http://www.cs.bgu.ac.il/omri
http://www.cs.bgu.ac.il/pollack

[9]. http://hebb.mit.edu/courses/9.03/ltp/sld004.htm

[10]. http://argo.urv.es/~bo/llistes/CCL/98/02/msg00075.html

[11]. www.google.com

[12]. www.yahoo.com

[13]. www.msn.com