Artificial Immune Systems for the Prediction of Corporate Failure and Classification of Corporate Bond Ratings

Master of Management Science

University College Dublin

Alice Delahunty B.Eng. Elec.
Denis O Callaghan B.Eng. Elec.

Supervisor: Anthony Brabazon
Department of Accountancy

August 2004

Chapter 1: Introduction

1.1 Introduction

The objective of this study is to determine whether an Artificial Immune System (AIS) algorithm, using information drawn from financial statements, is capable of accurately predicting corporate bond ratings and corporate failure.

When a company wants to issue debt (bonds) they must obtain a credit rating from a recognised agency. The credit rating represents the rating agency's opinion, at a specific date, of the creditworthiness of a borrower in general (an `issuer' credit rating), or in respect of a specific debt issue (a `bond' credit rating). Therefore, it serves as a surrogate measure of the risk of non-payment of interest or capital of a bond.

Corporate failure may be defined as the inability of a firm to meet its financial obligations resulting in liquidation of assets. The area of corporate failure prediction is closely related to the bond rating problem. Both areas have been investigated by many researchers and similar modeling techniques employed.

While the majority of research in this area has focused on statistical approaches, more recent studies have investigated artificial intelligence methods, mainly neural networks and case based reasoning. This study aims to tackle these well-established problems using the relatively new field of AIS.

Artificial immune systems (AIS) draw inspiration from the workings of the natural immune system (de Castro and Timmis, 2002). AIS extract ideas and metaphors derived from the workings of the immune system, which can be applied to help solve real-world problems. The key feature of the natural immune system is its ability to distinguish *non-self* (foreign substances) from *self* without prior knowledge of all possible non-self variants. This classification ability makes AIS highly suitable for application to the financial challenges outlined above.

1.2 Motivation and Contribution

An accurate bond rating prediction model would be of particular interest to several parties:

1. It would enable firms considering issuing debt to estimate what rate of return potential investors would require on bonds.
2. It would enable banks, and other firms, to estimate the creditworthiness of a previously unrated company.
3. It would allow investors seeking a trading edge to make early predictions of bond re-ratings
4. It would allow insurance providers to estimate the price of credit risk derivatives, which provide protection to investors against the downgrading of a bond.

Accurate corporate failure prediction would be of benefit to management, shareholders, employees, suppliers, customers, auditors and debt finance suppliers, providing them with early warning and enabling damage limitation measures to be taken. A corporate failure prediction model would also provide important information to potential investors.

Although several applications of AIS have been investigated since its emergence in the last decade, none of these have contributed to research in the financial area. This study therefore, not only contributes to the field of AIS but also brings it into the domain of management science by applying it to the important area of financial research. As shown above, accurate financial prediction models are highly desirable and while much research has been completed in this area this novel combination with AIS provides an important academic contribution.

This study considers two AIS algorithms, *negative selection* and *clonal expansion* in its attempt to develop acceptable prediction models. The main focus of the research is the negative selection algorithm and the implementation of recent proposed modifications. All the algorithms are used for both bond rating and corporate failure prediction with the testing limited to one dataset for each problem. Corporate failure prediction is implemented for one, two and three years prior to failure and provides

binary classification, i.e. fail or non-fail. The bond rating model is limited to binary classification between investment and non-investment grade bonds.

1.3 Structure of Study

This contribution is organised as follows. Chapter 2 provides a review of prior literature in the three areas relevant to this study: AIS, Bond Rating and Corporate Failure Prediction. Chapter 3 gives an outline of each of the algorithms considered, their implementation and observations made. Chapter 4 provides a detailed analysis of experimental results and interprets them in terms of previous research. The major conclusions and findings of the study are discussed in Chapter 5.

Chapter 2: Literature Review

This chapter gives an outline of the three main topics contained in this dissertation. The areas of Artificial Immune Systems (AIS), Bond Ratings and Corporate Failure are explained and followed by reviews of the prior literature on each.

2.1 Artificial Immune Systems

2.1.1 Introduction to Artificial Immune Systems

The Artificial Immune System (AIS) is a biological metaphor of the human immune system in the same way that Neural Networks are a *biological metaphor* of the workings of the human brain. AIS is a relatively new area of investigation falling in the Artificial Intelligence category with much of the research taking place in the last five years. For this reason, many papers on AIS focus on the workings of the natural immune system and the analogies that can be drawn from it. The following section gives an overview of the natural immune system and the key processes involved.

The study of Immunology began in 1796 with the discovery of the Smallpox vaccination by Edward Jenner. It is concerned with the defence mechanisms that confer resistance against diseases. The various cells and proteins that combine to provide this defence constitute the Immune System, and their collective and orchestrated response to the introduction of foreign substances (also called "non-self" substances) is the immune response. While historically immunology referred specifically to infectious diseases, developments in science discovered (de Castro, 2002) that the same processes protect against non-infectious foreign substances.

When an individual is first exposed to a cell or molecule, the immune system will determine if it is a "self" or "non-self" agent. Under normal conditions, if the substance/cell is the same as that found in the organism (i.e. self), the immune system will not react and is said to be tolerant to that particular agent. However, the immune response will trigger if the cell/substance is found to be "non-self". Foreign substances that trigger an immune response and react with the product of this response are generically termed *antigens*. In some cases the mechanisms that normally protect individuals from infections and eliminate foreign substances can themselves cause tissue damage and disease; examples of these are autoimmune

diseases such as diabetes where the body acts against its own pancreas or when a graft is rejected. Thus, Immunology deals with how the body distinguishes between "self" and "non-self" molecules; the remainder of this section deals with the technical and biological detail associated with this.

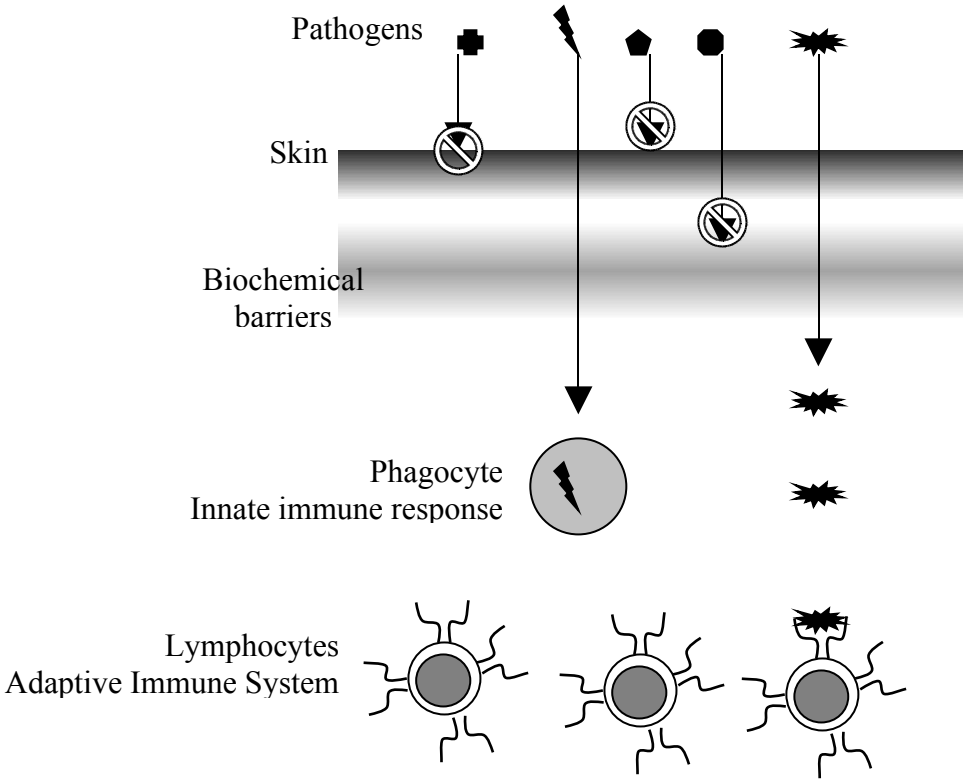


Fig 2.1 The Human Immune System Architecture

2.1.2 Innate and Acquired Immunity

Humans present two main types of immunity: *innate* (also known as natural immunity) and acquired or *adaptive* immunity. Innate immunity is the first line of defence against foreign cells or substances providing an immediate non-lasting response, which is non-specific. The cells and molecules that compose the innate immune system are present at birth and remain constant throughout an individual's life without improvement by repeated infection. Acquired immunity, on the other hand, is specific to the foreign molecule or cell, thus being an adaptive response to a given "non-self" substance and also presents memory (i.e. remembers encountered foreign molecules). The immunological memory is the basis of vaccination against infectious diseases. By injecting the body with a tiny amount of the infection it

develops a response that will be more effective, long lasting and stronger when faced with subsequent exposure to the specific disease.

Both the innate and acquired immune systems are comprised of a variety of molecules, cells and tissues. The most important cells are *leukocytes* (white blood cells), which fall into two major categories: *phagocytes* and *lymphocytes*, the first group belong to the innate immune system while the latter group mediate adaptive immunity.

The main soluble proteins responsible for the acquired immune system response are the *antibodies*. If the first innate defences are breeched, the specific immune mechanisms are activated and produce a specific reaction to each infectious agent and attempt to eradicate that agent.

1. Innate Immunity

Most infective agents encountered by the body are prevented from entering by a variety of physical and biochemical barriers such as skin, nasal hairs, and mucus. If an infectious agent penetrates these they meet a second set of barriers; the phagocytes and natural killer cells. Phagocytes can engulf many particles, bacteria and fungi destroying them in the process. The natural killer cells are able to recognise cell surface changes that happen in tumoral and virus infected cells and kill those altered cells.

2. Acquired Immunity

When an individual is exposed to a foreign antigen, two basic processes are stimulated. The first is mediated by antibodies which are present in the blood and biological fluids. This type of immunity is call humoral immunity. The second type is cell-mediated immunity (or cellular immunity), which is effected mainly by the T lymphocytes. Lymphocytes are grown, developed and deployed in the lympoid organs and consist principally of *B-cells* and *T-cells*, both of which specify activities in the system. While B-cells interact with antibodies in the humoral immunity processes and T-cells control the cellular immunity, most immune responses involve the activity and interplay of both, for example, T-cells can enhance or suppress the response of B-cells to a stimulus from an antigen.

Key Immune System Terms

Antigens	Foreign substances that trigger an immune response
Pathogens	Disease causing antigens
Leukocytes	White blood cells, include phagocytes and lymphocytes (B and T cells) for identifying and killing antigens
Antibodies	Proteins secreted into the blood in response to an antigenic stimulus that neutralizes the antigen by binding specifically to it

2.1.3 B-cells and T-cells

B-cells and T-cells have receptors on their surface that allow them to recognise and bind to antigens. B cell receptors (antibodies) are Y-shaped and facilitate the matching process. One antibody may recognise several antigens. The degree of matching determines the strength of the bond and hence the degree of stimulation of the lymphocyte. When a threshold of affinity has been reached, the B cell is activated, germinal centres are created in the lymph nodes and the B cell is transformed into a blast cell. The blast cell divides rapidly producing clones of itself. This B cell cloning process is termed *clonal expansion* and aims to produce a large number of antibody producing (B) cells which are specific to the offending antigen. B-cells are cloned in the lymph nodes at a rate proportional to their affinity with the antigen. Thus, antibodies that bind tightly are replicated at a higher rate and produce better variants. These clones then undergo an affinity maturation process to tune them more accurately to the antigen that initiated the response. Even when the antigen is previously unknown to the system, it is capable of producing a variety of closely matched receptors to detect antigens. *Somatic Hypermutation* is the name given to the process which mutates the newly generated lymphocytes, and the subsequent selection of the variants is referred to as *clonal selection*.

T-cells play a different role in the immune response. They are born in the bone marrow and matured in the thymus. There are various sub-populations of T-cells, including T helper cells, T memory cells and T suppressor cells. T-cells scan the body for fragments of antigens that have been presented on the surface of cells but their steps for defence are similar to those of B-cells.

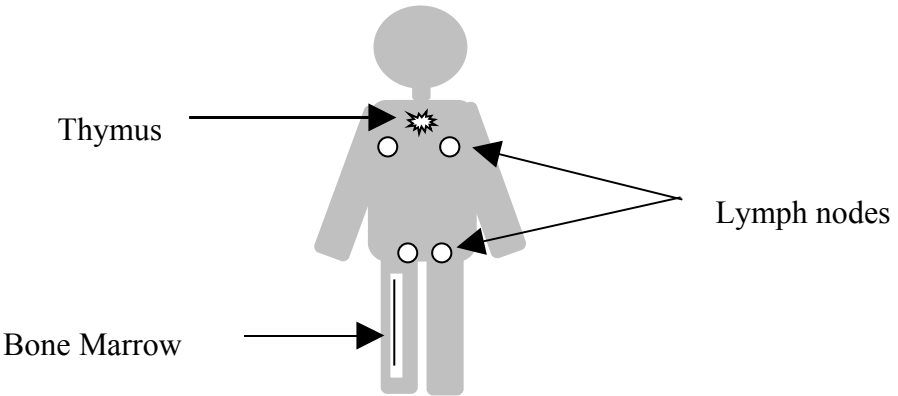


Figure 2.2 Anatomy of the immune system (lymphoid organs)

2.1.4 Self-tolerization

It is vital that the immune system only targets foreign (non-self) or misbehaving-self cells for destruction. In the normal creation of B-cells and T-cells, receptors are randomly generated and could potentially bind to either self or non-self. To prevent auto-immune reactions as discussed previously, the system must be self tolerized. During maturation the lymphocytes are exposed to a series of self molecules, any B-cells or T-cells that detect self at this stage are destroyed. The remaining detectors then circulate the system, this process is known as negative selection as only non-self reactive lymphocytes are permitted. Not all examples of self are necessarily presented during the tolerization stage and hence an additional security against automatic immune reactions is required (de Castro, 2002).

2.1.5 Immune System Memory

When the immune system encounters an antigen for the first time, a primary response as outlined above is provoked. After the infection is cleared, a memory of the successful receptors is maintained in the form of long-living lymphocytes (memory cells). Both B-cells and T-cells have "memory" variants. If the same or similar antigens are presented thereafter, a much quicker secondary response will result characterised by much more rapid and abundant production of the relevant antigen.

Steps in immune system response to foreign agent (antigen)

- Antigen enters the body
- B-cells stimulated
- With help of T-cells, B-cells undergo cloning and mutation
- Killer T-cells self-tolerized
- Antigens attacked by killer T-cells
- Memory maintained

As illustrated by the points above, the natural immune system has strong classification and pattern recognition abilities (i.e. self versus non-self) and has important learning and memory facilities. While a multitude of metaphors could be drawn from this system, the majority of research papers have focused on one of the following two areas:

1. Negative Selection
2. Clonal expansion

Both of these will be discussed in detail in Chapter 3.

2. 1.6 Previous Applications of AIS

Since its introduction in the mid 1980's, the field of artificial immune systems has been growing consistently and has been used in a variety of applications. A brief review of the literature of the last decade will be discussed under the next five headings.

1.Computational Security

The capability of the immune system to adapt to a disease previously unseen can be applied in the sphere of computing, in particular to the problem of recognising new viruses invading the system and detecting an intrusion into a system or network.

Forrest *et al.*(1994) proposed the original negative selection algorithm based on the generation of T-cells in the thymus. They described a method for detecting computer viruses by taking self as the normal state of the computer's system and training detectors on this normal (self) state to produce detectors that would recognise invaders or abnormalities (non-self) in the system. Detectors of self were rejected

and only those that did not match self were allowed into the valid detector set. Fig 2.1 illustrates this selection process.

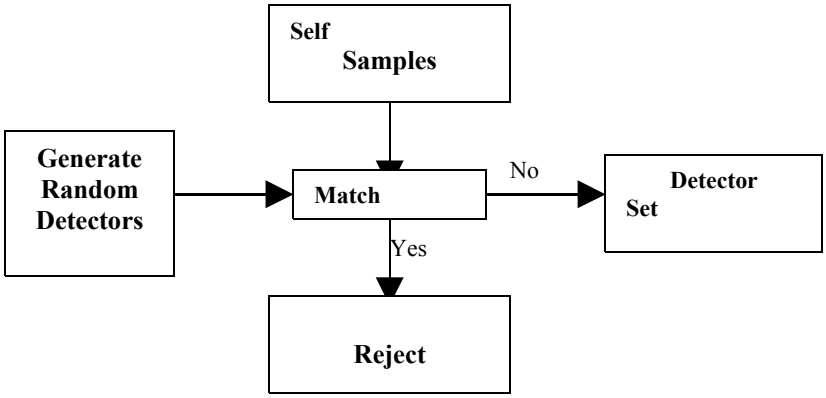


Fig 2.1 Generation of a valid detector set for computer virus detection

Kephart (1994) proposed an AIS method of generating computer antibodies to counteract formerly unknown viruses. The second time a virus attacks, the computer is equipped to respond rapidly to the danger using knowledge extracted from the first attack.

In the area of intrusion detection into a network of computers, Dasgupta (1999) proposed a system where agents roamed throughout the network examining the various elements for problems. The agents, in this case, being analogous to killer, suppressor and helper T-cells in the immune system.

2. Pattern Recognition

Systems that can recognise patterns in data are an important sector of AIS. Data such as measurements and observations contain patterns that can be identified by AI systems capable of doing so.

Dasgupta *et al*. (1999) used an AIS combined with a genetic algorithm in the field of spectra recognition. In a chemical reaction a set of reactants is mapped into a set of products and each one has a specific spectrum, which can be identified. The model used the following metaphors with the immune system:

Spectra Recognition	Immune System
Set of Reactants before chemical reactions	Self
Set of Products due to reaction	Nonself
Any of the Products	Antigen
Any evolved population that specifically recognises a single product	Antibody
Measures how well 2 spectra match	Affinity Function
Process of applying affinity function	Matching

3. Machine Learning

The immune system is capable of retaining knowledge of previous experiences and applying this knowledge to new problems. The area of machine learning attempts to create machines that are able to learn from experience and apply this acquired knowledge to a fresh problem.

Timmis and Neal's (2001) artificial immune system for unsupervised learning used the principles of clonal expansion, which will be described in Chapter 3. The detectors (B-cells) were reproduced proportional to the level of stimulation that they received from the environment. The Euclidean distances between the detector (B-cells), other cells and the antigen was used as the similarity measure. The stimulation level was a function of this similarity.

4. Fault Diagnosis

The immune system as a means of identifying faults in a system by recognising early warning signs has been researched. Several approaches to the problem have been made to develop fault diagnosis systems. Measures taken early to avoid the spread of a fault throughout a system can significantly limit the damage done to the system.

The negative-selection algorithm was used by Bradley and Tyrrell (2000) to design a hardware fault tolerant system. They used a finite state machine, a Field Programmable Gate Array, in their implementation of the system. The following were the analogies between the immune system and their hardware fault tolerance system:

Hardware Fault Tolerance	Immune System
Valid state/state transition	Self
Invalid state/state transition	Nonself
Error tolerance conditions	Antibody
Valid state/state transition tolerance conditions	Epitope
Invalid state/state transition tolerance conditions	Paratope
Tolerance condition variables	Genes
Recovery procedure activator	Helper T-cell

5. Robotics

A robot is a machine designed to perform specific tasks. Many robots are autonomous and do not have to be controlled by a human controller. They have their own artificial (machine) intelligence and have the capability to act alone to execute particular tasks. The field of robotics has evolved so that robots have characteristics such as self-replication, self-assembly and emergent behaviour. AIS principles are now being applied to the area.

Lee and Sim (1997) proposed a system for Distributed Autonomous Robotic Systems (DARS), which would allow a group of robots to behave in a certain way in a particular environment. Their model was based on the immune network model of Farmer *et al.* (1986) and the ideas of clonal selection, described further in Chapter 3. They used the following analogies between the model for DARS and the immune system:

DARS	***Immune System***
Environment	*Non-self (Antigen)*
Strategy of Action	*Antibody*
Robot	*B-cell*
Control Parameter	*T-cell*
Adequate Behaviour	*Stimulus*
Inadequate Behaviour	*Suppression*
Excellent Robot	*Plasma Cell*
Inferior Robot	*Inactivated Cell*

Table 4.3 Comparison between AIS for DARS and Immune System

This section has given an introduction to the AIS field, its current applications and prior literature in this area. The remainder of this chapter will concentrate on the financial issues relevant to this study and the methods applied to them to date.

2.2 Bond Rating

2.2.1 Introduction to Bond Rating

Most companies and governments require additional funds for everything from expanding into new markets to providing social services. The problem large organisations run into is that they typically need far more money than the average bank can provide. The solution is to raise money by issuing bonds to a public market. Thousands of investors then each lend a portion of the capital needed. Bond rating is a specification of the possibility of default by a bond issuer based on an analysis of the issuer's financial condition and profit potential. Bond rating services are provided by Standard & Poor's, Moody's Investors Service, and Fitch Investors Service. Bond ratings start at AAA (denoting the highest investment quality) and usually end at D (meaning payment is in default). In the US $1.7 trillion worth of bonds was issued in the first quarter of 2003, according to The Bond Market Association.

While “bond credit ratings” are more common, an agency may also provide an “issuer rating”, this is the agency’s opinion at a given date of the issuer’s ability and willingness to meet financial commitments on a timely basis.

Bond/credit ratings are of interest to many parties; they provide a means of determining risk premiums and marketability of bonds allowing firms issuing debt to estimate likely return investors require. Bankers and companies considering providing credit rely heavily on agency ratings to make important investment decisions, many regulatory requirements for financial decisions are based on credit ratings and some companies are restricted to investment grade bonds. Financial and accounting literature suggest both stock and bond markets react in a way that indicates ratings convey important information regarding value of a firm. Agencies invest a lot of time and human resources in deep analysis of the firms concerned and as a result company credit ratings are costly to obtain.

A model of the credit rating process would be of huge interest to the parties named above, it would enable them to rate firms that have not previously issued debt and hence do not have a published bond rating. As Rated Debt is publicly traded, accurate prediction could indicated re-rating before other investors see it, providing a

trading edge. It would also have implications for the pricing of credit risk derivatives, which provide insurance to investors against downgrade.

2.2.2 Bond Rating Classes

Various notations exist for rating classes, generally denoted by a letter. In the case of S&P there are 10 classes, from AAA to D. AAA to BBB considered investment grade, all others junk grade. C grade implies bankruptcy petition has been filed, D grade are currently in default on their obligations.

Bond Rating		Grade	Risk
Moody's	**S&P/ Fitch**		
Aaa	AAA	Investment	Highest Quality
Aa	AA	Investment	High Quality
A	A	Investment	Strong
Baa	BBB	Investment	Medium Grade
Ba, B	BB, B	Junk	Speculative
Caa/Ca/C	CCC/CC/C	Junk	Highly Speculative
C	D	Junk	In Default

Bonds may be down graded and are colourfully termed “fallen angels” this often results in a sell off by institutional investors who are governed by strict regulations. A lower rating means the perceived risk has increased and required interest yield must also increase. While a degree of consistency exists, agencies do not necessarily concur, this is clearly illustrated by a comparison in 1999 (Kish, Hogan, and Olson) of S&P and Moody of 1607 US bond ratings of investment grade, it was found they differed on 836 (52%) of the firms, and 111 cases differed by more than one grade.

2.2.3 The Bond Rating Process

Bond issuers pay credit rating agencies to assign them a credit rating; the agencies then take responsibility for that rating. In order to acquire a rating the firm submits documentation such as recent financial statements (annual reports, quarterly reports, income statements, balance sheets, statistical reports), prospectus for the debt issue and non-financial information. Meetings take place between firm management and rating agency staff, a report is prepared and considered by a committee. Agencies emphasise the consideration of non-financial information and industry and market factors, however they do not disclose the actual weights. Both quantitative and qualitative factors play an important role, with everything from analysis of strategic

competitiveness to operational level details being taken into account and a degree of subjective judgement employed. While agencies stress the importance of this, many studies have produced encouraging results using statistical and artificial intelligence methods. It is assumed that financial ratios and historical rating contain a lot of information relevant to the bond rating process. The objective of this paper is to build a model that extracts information from past observations and applies it to new firms. The following section reviews the main studies to date which attempt to model the process either by statistical or Artificial Intelligence methods.

2.2.4 Literature Review

1. Statistical Methods

Statistical methods were first explored in 1959, Fisher Utilized ordinary least squares (OLS) to explain the variance of a bond's risk premium. Several subsequent studies used similar methods and included linear regression (Horrigan, 1966) to predict bond ratings. Multiple discriminant analysis (MDA) was used by Pinches and Mingo (1973) to yield a linear discriminant function relating set of independent variables to dependant variable to better suit the ordinal nature of data and hence increase classification accuracy. Many researchers investigated Logistic regression analysis and probit analysis. A multi-nominal logit model was developed by Ederington, 1985 and ordered probit analysis by Gentry, Whitford and Newbold in 1988. The typical results of these studies were in the range of 50–60%. Different financial variables were used for different studies, however, most include a measure of size, financial leverage, long-term capital intensiveness, return on investment, short term capital intensiveness, earnings stability and debt coverage stability.

It was generally found that a small list of financial variables could classify about 2/3 of hold out sample bonds. Statistical methods are also succinct and easy to explain. The problem associated with these models however is that the multivariate normality assumptions for independent variables is often violated which means the methods being theoretically invalid for finite samples (Haung *et Al*., 2004).

2. Artificial Intelligence Methods

More recent studies have focused on Artificial Intelligence techniques, namely; rule-based expert systems, case-based reasoning systems and machine learning

techniques, specifically neural networks. Artificial intelligence (AI) models, copies or adapts systems from biology. The combination of human ingenuity and the explosion in computer power has created a host of creations that take as their starting point anything from human intelligence and emotions to genetic inheritance and evolution. A neural network is a computer system that can learn which combinations of inputs (representing companies financial data, say) lead to a particular output (such as a companies credit rating). Crucially, the network's inputs can be "weighted", so the influence of each can be increased or decreased depending on how significant its contribution is.

2a. Back Propagation Neural Networks

In bond rating studies Back Propagation Neural Networks was most frequently used. Dutta and Shekhar (1988) first investigated the applicability of Neural Networks in 1988, the reported a prediction accuracy of 83.3% for classifying “AA” and “non AA” rated bonds, this encouraged a significant amount of further research.

Sinlgeton and Surkan (1990) applied Back Propagation Neural Networks to 18 Bell Telephone companies; they experimented with one and two hidden layers. For the classification of a bond as either “Aaa” or one of “A1”,”A2” and “A3” by Moody’s they obtained a testing accuracy of 88%. They also demonstrated the superiority of Neural Networks over Multiple Discriminant Analysis. Kim (1993) compared Neural Networks with linear regression, discriminant analysis, logistic analysis and a rule-based system. Using Standard and Poor’s Compustat and Moody’s Annual Bond Record financial data sets they concluded Neural Networks consistently outperformed the other methods in terms of classification accuracy. As did Maher and Sen (1997) when they compared Neural Networks to Logistic regression, their best performance was 70%.

Later Moody and Utans used Neural Networks to predict 16 categories of S&P ratings, accuracy increased with decreasing number of categories and ranged from 36.2% for all 16 to 85.2% for 3-class prediction. Kwon et al. applied ordinance pairwise partitioning (OPP) approaches to Back Propagation Neural Networks. Utilising a Korean data they showed that OPP produced the best results (72%) followed by Neural Networks (66%) an multiple discriminant analysis (60%). Chsveesuk et al. Compared Back Propagation Neural Networks with radial basic

function, learning vector quantization and logistic regression. Neural Networks and Logistic Regression were the most accurate however they only achieved 51.9% and 53.3% respectively.

2b. Case-based Reasoning

Case-based Reasoning was also investigated by a number of researchers. The principle behind this is to match a new problem with the closest previous cases in the same way a human learns from experience to solve a problem. Sin & Han (2001) employed inductive learning for case indexing, and nearest neighbour matching algorithms to retrieve past cases. Using Korean bond-rating data and 5 catagories they reported accuracies of 75.5%, significantly higher than those of MDA and ID3.

2.2.5 Summary

The area of corporate failure prediction is closely related to the bond rating problem, it has also been investigated by many researchers and similar techniques employed, this will be discussed in more detail in the following section

The important prior studies exploiting AI have been summarised and the general conclusion is that Neural Networks out perform conventional statistical methods and other inductive learning techniques. The biggest drawback of Neural Networks is they are usual complicated and difficult to explain. Galindo and Tamayo differentiated statistical methods from machine learning methods by model size. For given training set there is an optimal model size. The models used in statistical methods tend to over simplify and hence under-fit the data, whereas AI models tend to over-fit, a trade off exists, therefore, between explanatory power and parsimony of a model.

This study aims to develop a model of the bond rating process using the relatively new AI technique of Artificial Immune Systems. In order to compare our findings to those of the prior literature the following must be taken into account; Results are highly dependant on the number of categories, the number of financial variable used in prior studies ranged from 7 to 87 and sample size from 47 – 3886, research to date has primarily focused on the United States and Korean Markets.

2.3 Corporate Failure

2.3.1 Introduction

There is no exact definition of corporate failure and any attempt to do so has proven problematic. The failure could be failure to operate effectively in the marketplace or in the legal sense it could be when the company goes bankrupt resulting in the liquidation of the firm's assets. Usually when a firm incurs liabilities that cannot be repaid from liquid financial resources, this leads to financial failure. However it may be the case that the company in question is just at the end of a period of financial decay and so, instead of failing, may recover. Another definition of failure was made by Beaver in (1966). He stated it as the inability of a firm to pay its financial obligations as they mature.

Corporate failure can result in considerable losses for many parties including shareholders, employees, suppliers, customers, auditors and debt finance suppliers. However, company failure can also have more widespread repercussions for industry in general and for the overall economy of the country in which the company operates. The worst hit by the failure of a company are naturally the management and the workers who can experience the effects of the decline of the company before the actual failure in the form of delayed payment of wages, poor housekeeping in the offices or even just a general deterioration in workplace morale.

2.3.2 Previous Prediction Methods

The accurate and early prediction of corporate failure is obviously desirable for many reasons. Early prediction could for example enable management to take steps to either prevent or at least reduce the impact of the failure. For the most part however, the literature and previous methods of prediction are aimed at the potential investor and warning him of future danger. Below are outlined a few of the previous methodologies developed in the past from the first discoveries that financial ratios might be used for prediction purposes through to the recent biologically based and regression type algorithms

Initial Discoveries

The first attempts at predicting business failures were made in the 1930s when the accounting profession was developing into its own distinct field. Studies made by Smith and Winakor (1935) and Merwin (1942) suggested for the first time that there was a considerable difference in the financial ratio patterns of failing and prosperous companies and so that they might be used as an early indicator of business failure.

The Univariate Approach

An important breakthrough was made in the prediction of corporate failure by Beaver (1966) by examining 30 financial ratios to determine which gave the greatest classification accuracy in separating failing and non-failing firms. His univariate analysis set the stage for the multivariate case and the studies that followed. He found that the ratio of cash flow to total debt had the most success in classifying failing and non-failing firms. This approach did demonstrate classification power but it suffers from the fact that a single weak financial ratio may be offset by the strength of other financial ratios. Also, Beaver equated bankruptcy and failure and as previously mentioned bankruptcy is a specific event whereas failure is harder to define. Predicting failure is more desirable since it gives the opportunity to take preventive measures when the first signs of failure are detected.

The Multivariate Approach

Multivariate analysis or multiple discriminant analysis (MDA) uses of a combination of ratios to differentiate between deteriorating companies and prosperous ones. The accuracy is increased over the univariate method since many ratios together are used to predict the health of a particular company as opposed to one single ratio. In these studies, the ratios that measured solvency, profitability and liquidity, were the ones which were found to give the greatest predictive accuracy. Altman (1968) developed his linear analysis Z-score model which uses five measures that are objectively weighted and summed to arrive at a classification rating. Altman's (1968) complex discriminant function had the following form:

$$Z = 0.012X_1 + 0.014X_2 + 0.033X_3 + 0.006X_4 + 0.999X_5$$

where Z indicated the overall health of the company and:

X_1 = *working capital to total assets*

X_2 = *retained earnings to total assets*

X_3 = *earnings before interest and taxes to total assets*

X_4 = *market value of equity to book value of total debt*

X_5 = *sales to total assets*

If Z was above 2.7 Altman found that the company was likely not to fail but if it was below 1.8 then failure was certain. Companies with Z in between these two figures were uncertainties and could go either way. One year prior to bankruptcy the prediction accuracy of Altman's model was 95%, with 72% for two years prior to bankruptcy and 48% for three years before.

Altman's original Z-Score model was modified by Altman, Hadelman and Naraynan (1977), using a larger dataset than Altman's original 33 failed and 33 non-failed companies. This ZETA model had the following seven variables:

X_1 = *return on assets (EBIT/Total Assets)*

X_2 = *stability of earnings*

X_3 = *debt service (EBIT / Total Interest)*

X_4 = *cumulative profitability (Retained Earnings / Total Assets)*

X_5 = *liquidity (Current Assets / Current Liabilities)*

X_6 = *capitalisation (Equity / Total Capital)*

X_7 = *firm size (Total Assets)*

The coefficients used in this study were not disclosed.

2.3.3 More Recent Studies

In more recent times a widely diverse set of methodologies have been used to try and predict corporate failure. From the regression models of the 1970s and 1980s (Edmister 1972; Ohlson 1980; Zmijewski 1984; Gentry, Newbold and Whitford 1985) to the more recent biologically-inspired methods such as artificial neural networks (Shah and Murtaza, 2000; Serrano-Cinca, 1996; Wilson, Chong, Peel, 1995; Tam, 1991), genetic algorithms (Varetto, 1998; Kumar, Krovi and Rajagopalan, 1997), grammatical evolution (Brabazon, O'Neill,Mathews and Ryan, 2002) and hybrid genetic algorithms (Brabazon and Keenan, 2003).

2.4 Conclusion

The areas described above in the three preceding sections have all been researched separately but, up till now, have never been combined. The uses of AIS in the financial classification and prediction domains were investigated in this study with a description of the concepts and implementations of the various algorithms, testing of the algorithms and the results obtained are given in the following two chapters.

Chapter 3: Research and Experimentation

3.1 Introduction

There were four main algorithms researched and implemented in the course of this study. The negative selection algorithm formed the basis of three of these while the other algorithm was based on the principle of clonal selection. In this chapter, an explanation of the concepts of each algorithm, their implementations, and the observations made on each is given. Several design issues had to be dealt with before implementing each algorithm and these are also covered in this chapter.

3.2 Explanations

Four different algorithms were implemented in order to optimally predict bond rating classifications and corporate failure. All of the algorithms are Real-Valued algorithms (Gonzalez and Dasgupta, 2002) as opposed to binary (i.e. those that run in binary time). Although they may not be as efficient, Real-Valued algorithms have the advantage that the companies and detectors can be seen as points represented by n-dimensional vectors. The self/non-self space corresponds to a subset of R^n, specifically in the case of these four algorithms, $[0,1]^n$. A brief summary of each of the four algorithms is given below with elaborations later in the chapter.

1. The Negative Selection Algorithm: The negative selection algorithm can be described as a mathematical representation of the maturation of T-cells in the thymus gland. It uses the principles of self/non-self discrimination to distinguish between two system states of normal and abnormal. The normal in this implementation are the healthy or prosperous companies while the abnormal are those that are on the decline. The unhealthy companies are identified using detectors trained on a sample set of healthy companies. This process is known as negative selection.

2. The Modified NS Algorithm: This algorithm is basically the same as the negative selection algorithm described above with the difference that

each detector is assigned an age or level of maturity. At each iteration, if a detector is still within the self-set, it is moved away from the self-set, and the age of the detector is increased. If the detector reaches a specified maturity and it still has not moved outside the self-set it is replaced by a new detector which then undergoes the same process.

3. The Clonal Expansion Algorithm: This algorithm is inspired by another process that takes place in the body's immune system. A large population of antibody-producing B-cells are generated which specifically react with a particular antigen of the evading pathogen. Clones of these initial cells are made, proportional to the degree of affinity or 'matching' they have with the antigen. In this manner the body 'tunes' itself to better detect the invader before repelling it. The algorithm trains itself on a sample of badly performing companies and clone detectors are generated in proportion to the initial detector's degree of similarity with the poorly performing companies. The similarity measure here is once again the Euclidean distance, but in this case it is between the detector and a non-self.

4. The Variable Detector Algorithm: The algorithm is another modification of the basic negative selection algorithm. In the basic NS algorithm, the detectors are hyper-sphere shaped and have a fixed radius of detection, the threshold Euclidean distance between them and the self samples. The detectors in the case of the variable detector algorithm have a variable property. The radius of detection in real-valued space of each detector is variable and so the self/non-self space can be covered with fewer detectors thus allowing the algorithm to be more efficient. Furthermore, while the basic NS algorithm might have 'holes' or 'gaps' in the area covered by its detectors the variable detector algorithm does not suffer from this drawback as smaller detectors may be generated to fill them.

3.3 General Design Issues

Five general design issues had to be dealt with before the implementation of each of the algorithms. They were:

1. Choice of Input Financial Ratios
2. The Source of Data Used
3. Choice of Design Language
4. The Preparation of the Data
5. The Cutting out of the Training Data Set

1. Ratio Choices

As detailed in the literature review, previous studies of both the bond rating and corporate failure problems have used a variety of financial ratios and the number of ratios used has ranged from 7 – 87 (Huang *et al.,* 2004*)*. In both cases we were guided by prior literature namely, Brabazon & Brennan (2003) and Brabazon & Keenan (2003).

Bond Rating Selection of Input Variables:

A total of eight financial variables were selected, for benchmark purposes we adhered to those used by Brabazon (2003) in a Neural Network study of the same data set. Prior literature (Altman, 1969) indicates that five groupings of explanatory variables are prime determinants of bond issue quality and default risk. These are:

1. Liquidity – Availability of cash resources to meet short-term cash requirements;
2. Debt – Measure of focus on the relative mix of funding provided by shareholders and lenders;
3. Profitability – Rate of return generated relative to size;
4. Activity/ Efficiency – Measure of operational efficiency, collecting cash, managing stock, controlling production etc;

and

5. Size – Measure of sales revenue and asset scale;

Each grouping has a range of individual financial ratios and the groupings themselves are interconnected, for example, a firm with high debt may have low profitability as a result.

A series of financial ratios were analysed. Their mean values were compared for investment grade and junk grade companies, and their correlation with each other examined, the objective being to minimise overlap of information while choosing ratios which vary between companies in different bond rating classes.

The following eight variables were chosen:

1. Current Ratio
2. Retained Earnings to Total Assets
3. Interest Coverage
4. Debt Ratio
5. Net Margin
6. Market to Book Value
7. Log (Total Assets)
8. Return on Total Assets

Corporate failure selection of Input Financial Ratios:

The data set for corporate failure contained 22 different financial ratios, each related to one of the five prime determinants previously mentioned. The ratios were chosen so as to give an even spread of the determinants. The following 8 ratios were chosen from the data set:

1. EBIT Margin
2. Return on Investment
3. Cash from Operations to Total Liabilities
4. Working Capital to Total Assets
5. Sales to Total Assets
6. Inventory to Working Capital
7. Total Liabilities to Total Assets
8. EBIT to Interest

2. Data Collection

Two principle data sets were used for the two different problems to be tackled. In the Bond rating case, 791 US companies, drawn from the Standard & Poor's (S&P) Compustat database, were used. Financial companies such as banks were excluded from the study on the basis that their financial ratios are incomparable with those of other firms in the sample. Companies without a credit rating from S&P on the required date and those that did not have a financial year end of 31 December. The required financial ratios for each company and their credit ratings were extracted from this dataset and an input file was better prepared. This will be discussed in more detail in the following section.

For the corporate failure prediction problem, three datasets were compiled, ata one, two and three years prior to failure. Each data set consisted of 180 companies, 90 matched pairs where one member of each pair failed while the other did not. The required ratios were extracted and input datasets created.

3. Language Choice

All algorithms were coded in Matlab as it is generally considered to be the leading language of technical computing; its vectorized base is ideally suited to the nature of the problem and largely eliminates the need for complex loops.

4. Data Preparation

As the financial ratios were real world figures, they took on a wide range of values that were not comparable as vector elements. To enable them to be compared, they were normalised to figures between 0 and 1 before being inputted into the algorithm. However, outlying data in the data sets caused an uneven distribution of the normalised data, which then, due to the high level of precision required, caused the program to be unable to distinguish between companies. The outlying data figures therefore were removed from the dataset and re-entered into the algorithm giving acceptable results.

5. The Cutting out of the Training Data Set

Once the data in the data set was normalised a percentage of the self samples was chosen at random and removed from the data set. This then became the

sample that the detectors were trained on. An example of a 25% cut is given in Table 3.1 below, with the shaded cells representing the selected companies. Five out of the 20 healthy companies are chosen. The figure in the final column indicates whether a company is investment grade, denoted by 1, or junk grade, denoted by 0. Several of these random cuts were taken at various percentages for both bond rating and corporate failure testing. A full description of the results obtained is given in chapter 4.

Table 3.1 Example of a 25% cut from a dataset

Company	Current Ratio	Ret Egs/TA	Interest Cover	Debt Ratio	Net Margin	Market Value	Total Assets	ROA	invest=1 junk = 0
1	0.094	0.8413	0.2201	0.161	0.9152	0.0286	0.0636	0.458	1
2	0.1104	0.8101	0.1853	0.1334	0.9163	0.0187	0.0345	0.374	1
3	0.0662	0.7957	0.1815	0.1886	0.9107	0.0069	0.1268	0.3817	1
4	0.239	0.8317	0.7805	0.0368	0.9386	0.9963	0.0241	0.6336	1
5	0.1257	0.8341	0.1685	0.184	0.9018	0.0067	0.0246	0.3588	1
6	0.0797	0.6947	0.1554	0.1564	0.894	0.014	0.2465	0.3359	1
7	0.1008	0.8221	0.2243	0.1564	0.9096	0.0137	0.0498	0.4351	1
8	0.118	0.8822	0.2856	0.0828	0.9107	0.1118	0.0094	0.4885	1
9	0.0374	0.7981	0.1764	0.23	0.9096	0.0058	0.2138	0.374	1
10	0.0518	0.7764	0.1601	0.2484	0.8783	0.0063	0.064	0.3511	1
11	0.0777	0.7812	0.1672	0.1978	0.8984	0.0051	0.1238	0.374	1
12	0.0912	0.8558	0.2162	0.1518	0.9062	0.0208	0.0171	0.4275	1
13	0.1161	0.8509	0.2288	0.253	0.9152	0.0165	0.0109	0.5114	1
14	0.0729	0.8437	0.1964	0.161	0.9096	0.013	0.0106	0.3969	1
15	0.0797	0.875	0.2246	0.1426	0.9118	0.0226	0.0149	0.4275	1
16	0.0893	0.774	0.1903	0.2254	0.9141	0.0074	0.0774	0.4198	1
17	0.1104	0.9327	0.0947	0.1334	0.8828	0.0198	0.0379	0.2595	1
18	0.0969	0.8798	0.3892	0.0598	0.9129	0.2299	0.0459	0.4504	1
19	0.1334	0.7716	0.1864	0.1702	0.9051	0.0097	0.0322	0.4046	1
20	0.3599	0.9591	0.2184	0.1794	0.9129	0.0126	0.0133	0.4046	1
21	0.1171	0.7476	0.1593	0.1932	0.8984	0.0036	0.0036	0.3511	0
22	0.2975	0.8413	0.1739	0.1656	0.904	0.0093	0.0042	0.3893	0
23	0.214	0.8053	0.2446	0.1104	0.9096	0.0299	0.0045	0.4427	0
24	0.1008	0.7548	0.1902	0.1288	0.9051	0.3475	0.0025	0.4122	0
25	0.3157	0.8437	0.1228	0.1196	0.8951	0.0447	0.0038	0.2977	0
26	0.0921	0.8029	0.1986	0.1978	0.9062	0.0061	0.0048	0.4351	0
27	0.2179	0.8341	0.2547	0.1564	0.9107	0.0207	0.0017	0.4733	0
28	0.167	0.8269	0.1838	0.2484	0.9062	0.0066	0.001	0.4504	0
29	0.2303	0.7957	0.1737	0.2898	0.9107	0.0063	0.0126	0.4198	0
30	0.0528	0.8365	0.169	0.3082	0.9029	0.0045	0.0039	0.374	0
31	0.0499	0.7139	0.1518	0.4232	0.8571	0.0038	0.001	0.313	0

3.4 The Algorithms

3.4.1 The Unmodified Negative Selection Algorithm

3.4.1.1 The Concept

The basis of the negative selection algorithm is the principle of *self/non-self* discrimination or, in other words, the ability of the body to differentiate between what is normal and what is abnormal. As pointed out in Chapter 2, the selection process that takes place inside the organ called the thymus gland is the inspiration for the algorithm. T-cells are generated in the thymus and undergo a censoring process whereby those that react with '*self*'-proteins are rejected and eliminated and only those that do not bind the 'self'-proteins are allowed to remain. This process ensures that the remaining T-cells recognise only foreign or anomalous molecules. These 'detector' T-cells are then circulated throughout the body, independently and freely, eliminating any material they can bind. Such material is regarded as *nonself.*

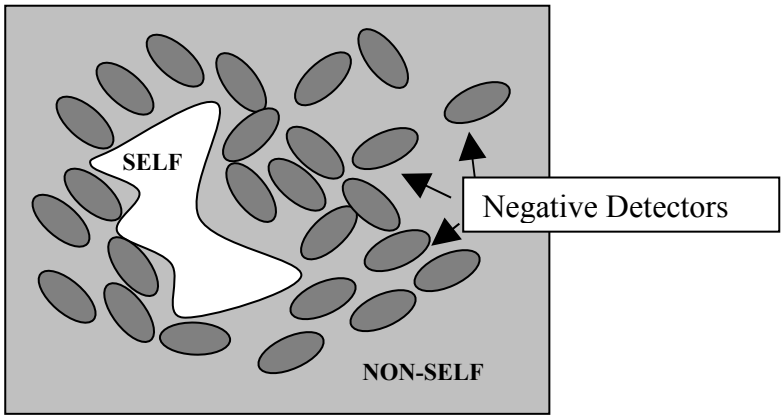


Fig 3.2 Self /nonself discrimination. A universe of data points is partitioned into two sets: self and nonself. Negative detectors cover subsets of nonself

The negative selection algorithm uses an analogous process in order to produce its negative detectors. Initially, a set of detectors is generated randomly. This randomness is comparable to the pseudo-random genetic rearrangement process that the body uses to create its T-cells. Each individual detector then goes through a 'training' period whereby any detectors that are found to match any elements of a set of 'self' samples are discarded and replaced by another randomly generated detector. This ensures that the set of detectors remains a constant size. These replacement detectors are also checked against the set of

'self' samples. A detector that goes through this process and is not eliminated becomes a mature detector. The procedure repeats until all of the detectors in the detector set are these mature, 'nonself' detectors.

When the set of mature detectors has been created, it is exposed to a new set of data. Each detector is compared with each element in the test set. If a detector is found to match any of the test set elements, it is reasonable to suppose that non-self has been detected. In this way, the negative-selection algorithm has the advantage of being able to identify elements not previously seen. In terms of the immune system, this technique enables hitherto unknown harmful pathogens to be both detected and dealt with. This useful aspect of the algorithm can be used to apply it to a wide range of areas where detection of variations from the normal behaviour or pattern is required.

3.4.1.2 Background

Since its introduction (Forrest, Perelson, Allen, & Cherukuri*, 1994)*, the realisation that the negative-selection algorithm could be used in many applications where anomaly pattern recognition is required has been increasing. So far it has been used in areas such as:

- Computer Virus Detection
- Tool Breakage Detection
- Intrusion Detection
- Fault Tolerant Hardware
- Industrial milling operations

The algorithm has not been applied in the area of financial prediction although its potential in differentiating between financially healthy and financially distressed companies has been noticed (Chen, 2002).

3.4.1.3 Algorithm and Implementation

The two system states of 'self' and 'nonself' are the healthy and distressed companies respectively. The algorithm is required to perform two separate classifications:

- Whether the bond ratings of 791 companies were investment grade or junk grade bonds;
- Whether or not a company would fail based on financial ratios for 180 companies over a period of 3 years;

The implementation is split up into two main sections:

- Training: Creating a set of detectors that will identify 'nonself', or unhealthy companies, by application of the negative-selection process;
- Testing: Using these detectors to distinguish between the prosperous and distressed companies.

The self for the algorithm is a set of vectors where each vector corresponds to one of the financially healthy companies. Each company has 8 financial ratios associated with it and so the 'self' is a set of vectors in the 8^{th}-dimension. The normalised financial data for the companies is then inputted into the program.

Training:

1) Firstly, a set of detectors is randomly generated where each detector corresponds to a vector with 8 elements. The seed for the random generation is set to the current clock time to ensure completely random generation. An example of 2 of these randomly generated detectors is shown in the following table:

0.12356	0.54322	0.98786	0.98766	0.32456	0.04532	0.34356	0.28492
0.29745	0.23567	0.23456	0.87623	0.01229	0.98562	0.54763	0.87763

A counter called 'numSelf' is initialised to 1 and a loop is entered which runs while the 'numSelf' counter is greater than 0.

2) Inside the loop, the 'numSelf' counter is set to 0 and each detector is checked against each entry (company) of the training sample set. In order to calculate the similarity between the detectors and the healthy companies, the Euclidean distances between each detector and the samples in the training set are found. The *k*-nearest neighbours in the self-set of each detector are found and the median Euclidean distance of these *k*-neighbours is calculated. If this median distance is less than *r,* the training threshold distance, the detector is discarded and replaced with a new randomly generated detector and the 'numSelf' counter is increased by one. The strategy of getting the median Euclidean distance of the *k*-nearest neighbours to each detector instead of just the distance of its nearest neighbour makes the algorithm more robust to outliers and noise in the data sets (Gonzalez and Dasgupta, 2002).

3) This loop repeats itself until the 'numSelf' counter is equal to zero, i.e. none of the detectors in the detector set match any of the healthy companies too closely. At this point, the while loop terminates. These detectors are now ready to be tested against the test set.

Testing:

4) Each of the matured or 'trained' detectors is now compared in turn with each of the vectors in the test set. As in the training, this is done by finding the Euclidean distances between them. Any of the companies in the test set that are less than the test threshold distance away from a detector are deemed to be 'nonself', or unhealthy, companies.

5) The percentages of correctly identified, incorrectly identified and non-identified companies are then returned.

A flowchart of the algorithm is given in Fig 3.3

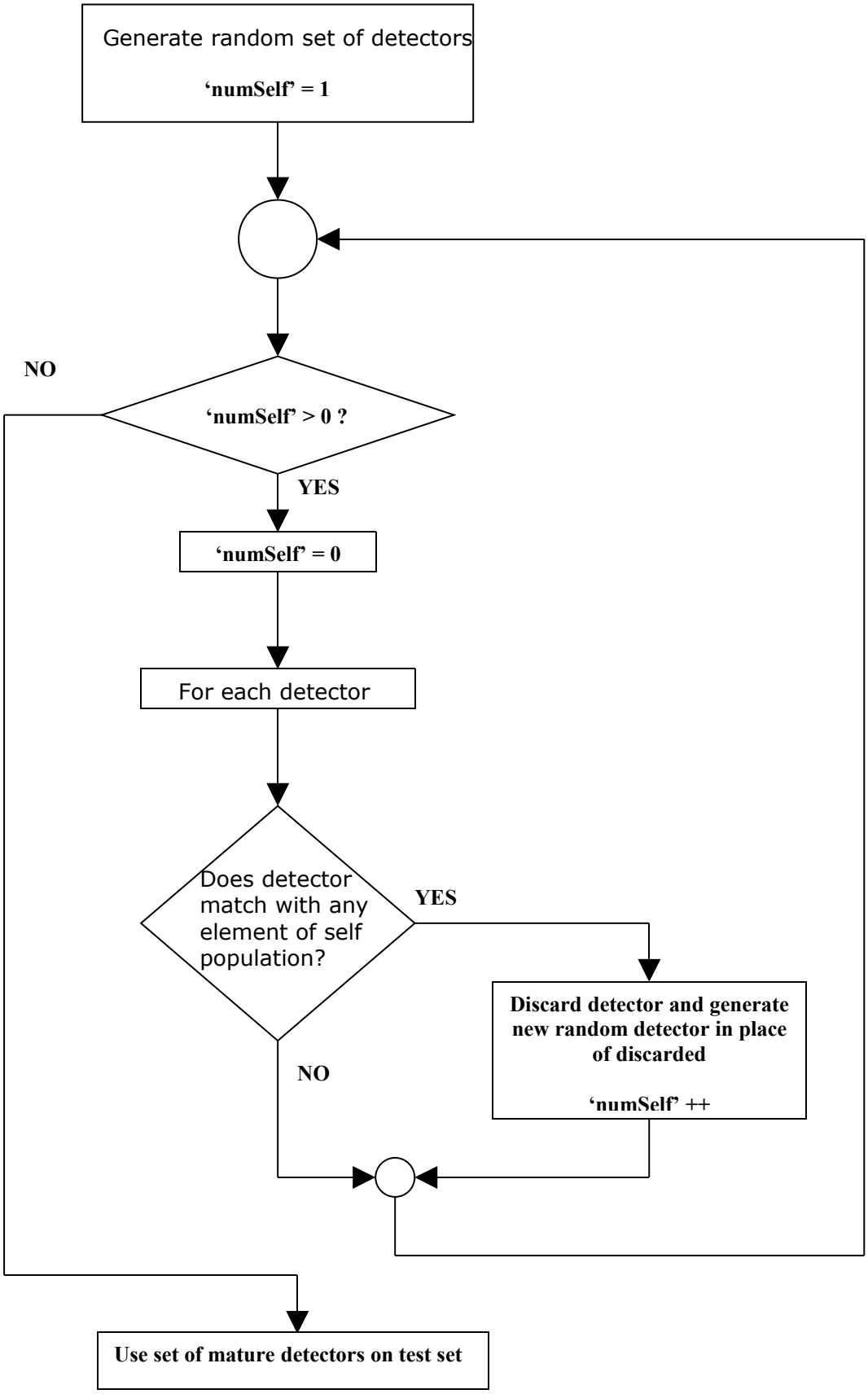


Fig 3.3 Flowchart of negative-selection algorithm

3.4.1.4 Observations

Bond Rating

After the program was written out in MATLAB, the next stage was finding the optimum size for the training sample set and the optimum tolerance levels. Several cuts were made at each of the percentages and the results obtained are discussed in further detail in the next chapter. A rough estimate of the tolerance levels were calculated by a process of trail and error and then a more exact figure by varying the tolerances around this initial rough estimate. Initially, the dataset was normalised to between 0 and 1 with outlying data included and the algorithm was run. When the outliers in the data set were removed and the dataset then normalised the results improved considerably. In order to increase the accuracy of the classification, the median Euclidean distance for the *k*-nearest neighbouring self-samples to the detector instead of the Euclidean distance to just the nearest neighbour was used in the training part of the algorithm. This increased accuracy due to the reduction of the effect of outliers and noise. The final optimum percentage of correctly identified companies in the dataset was found to be around 72%. This was obtained using a training sample that was 25% of the original, with a training threshold of 0.86 and a test threshold of 0.8.

Corporate Failure

The same steps were undertaken for the corporate failure datasets. For one year prior to failure, when a cut of 40% and training and test tolerances of 0.8 and 0.72 respectively were used, a prediction accuracy of 78% was obtained. For two years prior to failure, a 40% training cut with training tolerance of 0.8 and test tolerance of 0.74 gave a 66% accuracy was obtained. Finally, for three years prior to failure, a 20% cut and training tolerances of 0.81 and 0.79 achieved 57% accuracy of prediction.

Further detail of these results and how they compare to other methods of prediction are given in chapter 4.

3.4.2 The Modified Negative Selection Algorithm

3.4.2.1Concept

Gonzalez and Dasgupta (2002) proposed that a variation of the basic negative selection algorithm be used for the purposes of anomaly detection. The main difference between this algorithm and the basic NS algorithm is the method of detector training. In the basic algorithm, if a detector is found to be within the threshold Euclidean distance of a self-sample, it is discarded and replaced by a new randomly generated detector without any further processes being carried out on it. However, in the modified version, if a detector is found to match a self-sample, it is moved away from the self-sample instead of being replaced straight away. At each iteration, if the detector is still within the threshold, it is stepped further away from the self-sample. Every detector is assigned an age which is increased every time it is relocated. Its position continues to be updated in this way until the detector either moves outside the threshold distance or reaches a pre-specified maturity level.

3.4.2.2Algorithm and Implementation

Each detector has a radius of detection, *r,* associated with it. For a detector, *d*, to be valid the shortest Euclidean distance between it and any of the self samples can be no less than *r*. As in the case of the unmodified negative selection algorithm, the *k* nearest neighbours of *d* are calculated and then the median distance. The detector matches self (healthy company) if *r* is larger than this median distance. If the detector does match self it is discarded.

As in the case of the unmodified negative selection algorithm, non-self (unhealthy company) is detected by a detector *d* if the Euclidean distance between *d* and the unhealthy company is less than *r*.

The Modified Negative Selection algorithm is as follows:

Modified Negative Selection Algorithm

r: radius of detection

η: adaptation rate

t: once a detector reaches this age it will be considered to be mature

k: number of neighbours to take into account

1. Generate a random population of detectors
2. While stopping criterion is not satisfied
   3. For each detector *d* do
   4. *NearCells* = *k* nearest neighbours
   5. *NearCells* is ordered with respect to the distance to *d*
   6. *NearestSelf* = median of *NearCells*
   7. If distance (*d, NearestSelf)* < *r*
      8. Then:

$$dir = \frac{\sum_{c \in NearCells} (d - c)}{|NearCells|}$$

      9. If age of *d* > t → detector is old
         10. Then:
             Replace *d* by a new random detector
         11. Else:
             Increase age of *d*
             $d = d + \eta * dir$
      EndIf
   12. Else:
      age of *d* = 0

$$dir = \frac{\sum_{d' \in Detectors} \mu_d(d')(d - d')}{\sum_{d' \in Detectors} \mu_d(d')}$$

      $d = d + \eta * dir$
   EndIf
EndFor
EndWhile

As explained above, if the detector is within the self-set and has not reached maturation age, it is stepped away from the self-set. The size of the step that a

detector takes is determined by the parameter, η. However, this parameter must be reduced at each iteration in order to ensure that the algorithm converges to a stable state (Gonzalez and Dasgupta, 2002). The following updating rule is used:

$$\eta_i \leftarrow \eta_o e^{-\frac{i}{\tau}}$$

where η_o is the initial adaptation rate value and τ is its decay parameter.

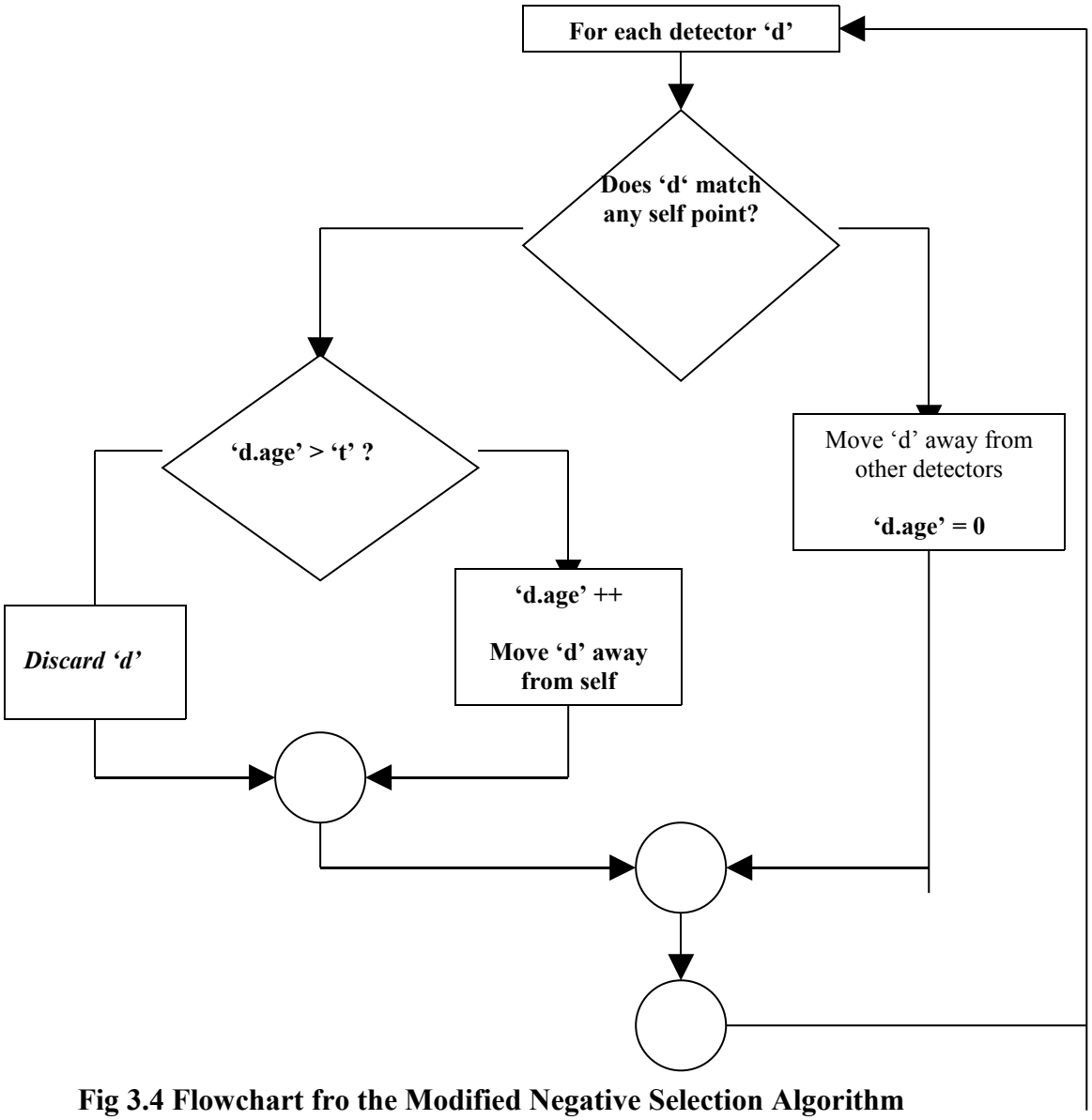


Fig 3.4 Flowchart fro the Modified Negative Selection Algorithm

3.4.2.3 Observations

For the bond rating dataset, the prediction accuracies obtained with the Modified Negative Selection Algorithm were lower than acceptable levels. The results gave no better prediction accuracies than random chance would give with no consistency achieved. Similarly, for all three of the corporate failure datasets, no consistency in results was attained. All of the parameters were varied in the following way with no noticeable change in the results:

1. The Number of Detectors was varied from 10 → 10,000
2. The size of *r* was varied from 0.1 → 1.5
3. The initial adaptation rate, η_o was varied : 0.1, 0.01, 0.0001, 0.001
4. The decay parameter, τ, was varied 0.01 → 100

3.4.3 The Clonal Selection Algorithm

3.4.3.1 Concept

Clonal Selection is the name given to the theory that explains how the adaptive immune system copes with pathogenic (disease causing) microorganisms. When a B cell receptor (antibody) recognises a non-self antigen with a certain affinity, it is selected to proliferate, and produces high volumes of the antibody. Reproduction of immune cells is asexual, the cells divide themselves and are subjected to a high rate of mutation followed by a selection process. This whole proliferation, mutation and selection process is known as affinity maturation of the immune response. The activated B cells with high antigenic affinities are also retained as long-living memory cells.

Important features of Clonal Selection from a computational point of view are:

- An antigen selects several immune cells to proliferate;
- The proliferation rate of each immune cell is directly proportional to its affinity with the specified antigen;

and

- The mutation inflicted upon each cell is inversely proportional to the affinity of the cell receptor with the antigen;

Thus, receptor cells, which match closely with the invading antigen, are replicated at a high rate but mutated at a low one and vice-versa.

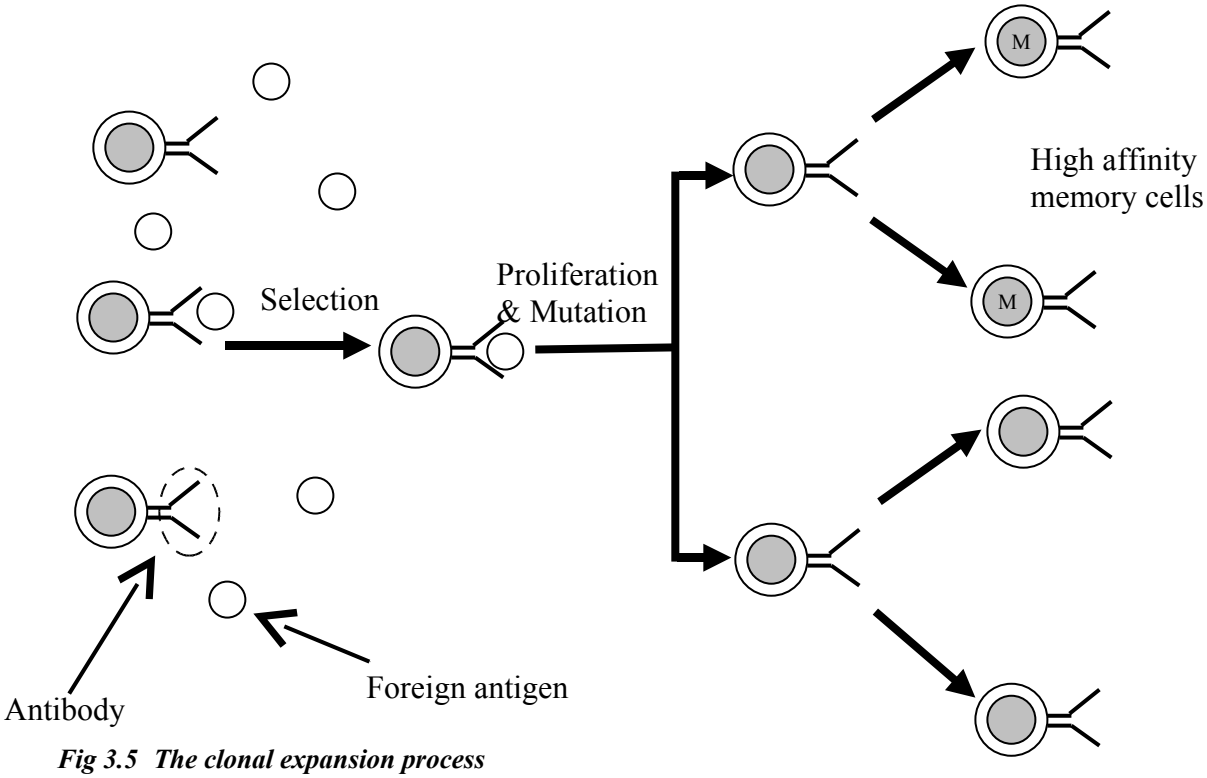


Fig 3.5 The clonal expansion process

3.4.3.2 Algorithm and Implementation

De Castro & Von Zuben (2000) proposed an algorithm CLONALG to represent this process. The algorithm is evolutionary in nature and was initially proposed to perform pattern recognition.

The basic steps of CLONALG as presented in "AIS a new computational Approach" (de Castro, 2002) are as follows:

1. Initialisation: Create an initial random population of individuals (P)
2. Antigenic presentation: for each antigenic pattern, do:
    a. Affinity evaluation: present it to the population P and determine its affinity with each element in the population P;
    b. Clonal Selection and expansion: select n_1 highest affinity elements of P and generate clones of these individuals proportionally to their affinity with the antigen: the higher the affinity, the higher the number of copies and vice-versa;
    c. Affinity maturation: mutate all these copies with a rate inversely proportional to their affinity with the input pattern: the higher the

affinity, the smaller the mutation rate and vice-versa. Add these mutated individuals to the population P and reselect the best individual to be kept as the memory m of the antigen presented;

d. Metadynamics: replace a number n_2 of individuals with low affinity by (randomly generated) new ones;

3. Cycle: repeat Step 2 until a certain stopping criterion is met.

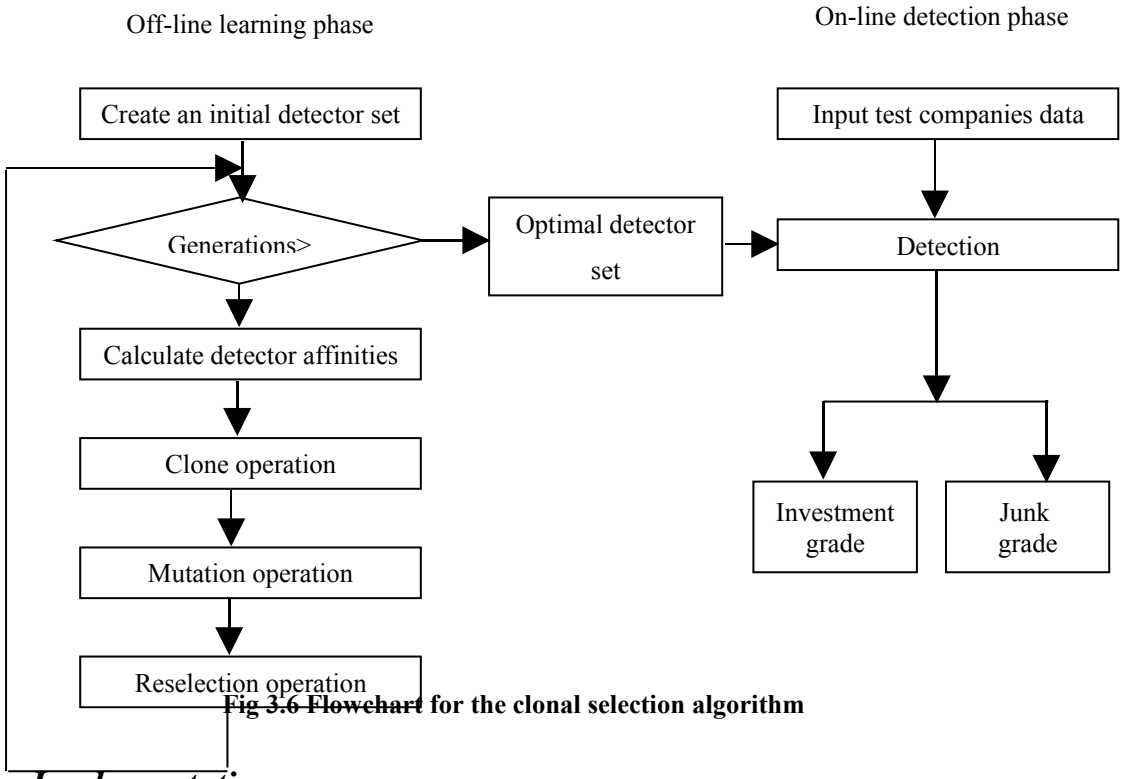


Fig 3.6 Flowchart for the clonal selection algorithm

Implementation

The above algorithm can be interpreted in terms of the financial datasets as follows:

Where the model is trained on a sample of non-self (unhealthy companies).

1. Initialisation: Create an initial random detector set D
2. Antigenic presentation: for each non-self sample, do:
    a. Affinity evaluation: Determine the distance between it and each detector in D;
    b. Clonal Selection and expansion: select n_1 nearest detectors and generate clones proportionally to their distance from the non-self sample element.

c. Affinity maturation: mutate all these copies with a rate inversely proportional to their distance from the non-self sample element. Add these mutated detectors to D and reselect the best to be kept as the memory m of the antigen presented;
d. Metadynamics: replace n_2 detectors with low affinity by (randomly generated) new ones;

3. Cycle: repeat Step 2 until a certain number of mutated detectors have been generated.

The above algorithm was coded in Matlab with the stopping criterion set at 500 detectors, a figure which had produced good results in the Negative Selection case.

The number of clones to be produced of each detector was calculated as follows:

$$n_i = round\left(N \times \frac{f_i}{\sum_{i=1}^{N} f_i} \right) i = 1, 2, ..., N$$

where N is the number of non-self samples and f_i is the affinity value (distance) of the ith non-self sample.

The mutation rate was calculated as follows:

$$p_i = \frac{f_{\max} - f_i}{f_{\max} - f_{\min}}, i = 1, 2, ..., N$$

Where f_{max} is the maximum affinity value and f_{min} is the minimum affinity value.

While many papers have investigated the mutation of binary strings, little research has been conducted into the mutation of real valued vectors. As the investigation of complex functions is beyond the scope of this study, detector vectors were mutated by adding random vectors of equal dimension. The vector elements were comparatively small positive or negative numbers.
In the reselection process, the detectors are sorted by descending affinity value and the lowest n_2 are replaced with randomly generated detectors.

3.4.3.3 Observations

For all test runs, the accuracies were found to be unacceptably low, with upwards of 90% of all investment grade companies being misidentified as junk grade. The following parameters were varied:

- The initial size of the detection set, $|D| = 5, 50, 100, 500$
- The number of detectors copied, $n_1 = 1, 5, 10, 50$
- The number of detectors reselected for memory, $n_2 = 1, 5, 10$
- The scaling factor for mutation, $x = 0.1, 0.01, 0.001, 0.0001$

The detectors generated, however, continued to span the entire hyperspace and result in all companies being detected and classified as non-self.
While this algorithm has been successful in other classification exercises we believe it is not suited to the given financial datasets and that the added complexity of multiple dimensions requires a more elaborate mutation function.

3.4.4 Variable Detector Negative Selection Algorithm

3.4.4.1 Concept

The negative selection algorithm, as described, generates a detector set by eliminating any detector candidates that match self-samples; the matching rule is generally based on Euclidean distance between the data to be tested and the detectors. Matching is usually defined as a distance that is within a certain threshold; the detectors are in fact hyper-sphere-shaped and the threshold is actually the radius of the detectors. Ji and Dasgupta (2004) suggested that allowing the detectors to have some variable properties enhances the negative selection algorithm from several aspects. In real valued application which uses the Euclidean distance matching rule varying the radius of the detectors is an obvious choice considering that the non-self regions to be covered by detectors are very likely to be in different scales. Variable detector radii permits a small number of detectors with large radii to cover large areas of non-self space which, in turn, makes the use of very small detectors to cover "holes" more feasible.

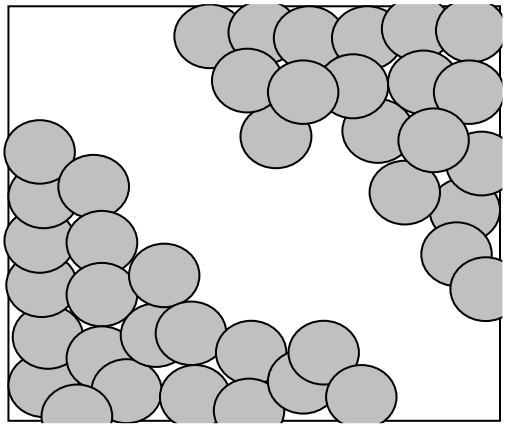


Fig 3.7.1 7.1

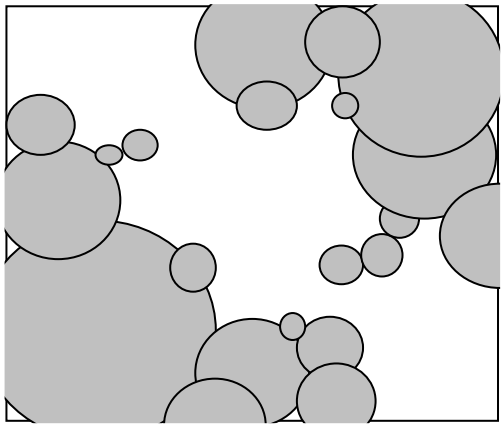


Fig 3.7.2

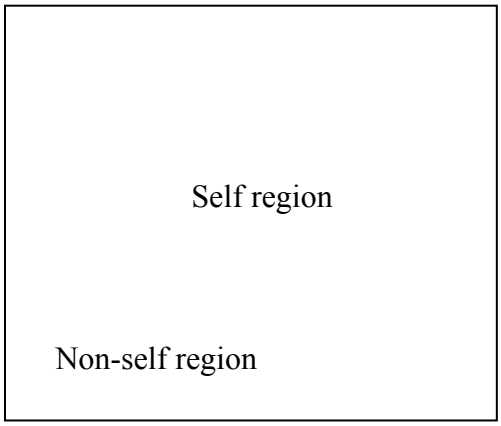


Fig 3.7.3

Figure 3.7.1 illustrates the unmodified negative selection case: many detectors of equal size are required to cover the entire non-self region, while small areas "holes" remain uncovered. Figure 3.7.2 demonstrates the advantage of detectors with variable radii sizes: fewer detectors are required to cover the large non-self regions on the top-right and bottom-left corners, while small detectors are available to cover the holes.

A key new feature of the variable-coverage method is the use of estimated coverage as a control parameter instead of the number of detectors, it estimates automatically when the detector set is generated. This will be discussed in more detail in the implementation section.

3.4.4.2 Algorithm and Source

In the negative selection algorithm with variable-coverage detectors each detector is assigned its own radius in addition to location, the radius is basically decided by the closest self sample. The control parameters of the algorithm are self radius $r_{s,}$ expected coverage c_0 and the maximum number of detectors T_{max} .

- Self radius is important in balancing between detection rate and false alarm rate, i.e. the sensitivity and accuracy of the system.
- Expected coverage is an attractive additional control factor. If we sample m points in a given space and only one point is not covered by a detector, the expected coverage is 1 – 1/m. Therefore, if m random tries are made without locating an uncovered point, we conclude the expected coverage is at least $\theta = I - 1/m$. This implies that the necessary number of tries to ensure expected coverage θ is $m = 1/(1-\theta)$.
    - Where expected coverage may be set to 90% for example, $\theta = 0.9$.
- Maximum number of detectors is preset to the largest acceptable number of detectors and will terminate the algorithm when reached if another stopping criterion has not already been called.

The V-Detector algorithm is given as follows:

V-Detector – Set(s, T_{max}, r_s, c_0)
S: set of self samples
T_{max} : maximum number of detectors
r_s: self radius
c_0: expected coverage
1: D¬
2: Repeat
3: t ¬0
4:T¬0
5: r¬infinity
6: x¬random sample from $[1,0]^n$
7: Repeat for every d_i in D={d_i, i=1,2,…}
8: d_d¬Euclidean distance between x(d_i) and x, where x(d_i) is the location of d_i
9: if d_d r(d_i), where r(d_i) is the radius of detector d_i
10: then t¬t+1
11:if t 1/(1-c_0) then return D
12: go to 4:
13: Repeat for every s_i in S
14: d¬Euclidean distance between s_i and x
15: if d - r_s r then r¬ d - r_s:
16: if r>r_s then D¬D{<x,r>}, where <x,r> is a detector with location x and radius r
17: else T¬T+1
18: if T > 1/(1-maximum self coverage) exit
19: Until |D|=T_{max}
20: return D

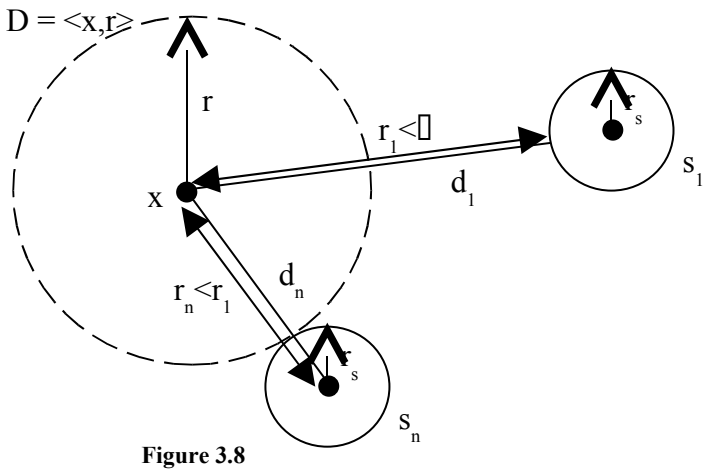


Figure 3.8

Figure 3.8 gives a graphic representation of steps 13 to 16, where the radius r associated with random detector x is established by comparison with each element of self s_n and their self radii r_s.

3.4.4.3 Implementation

The above algorithm was coded in Matlab using the following structure

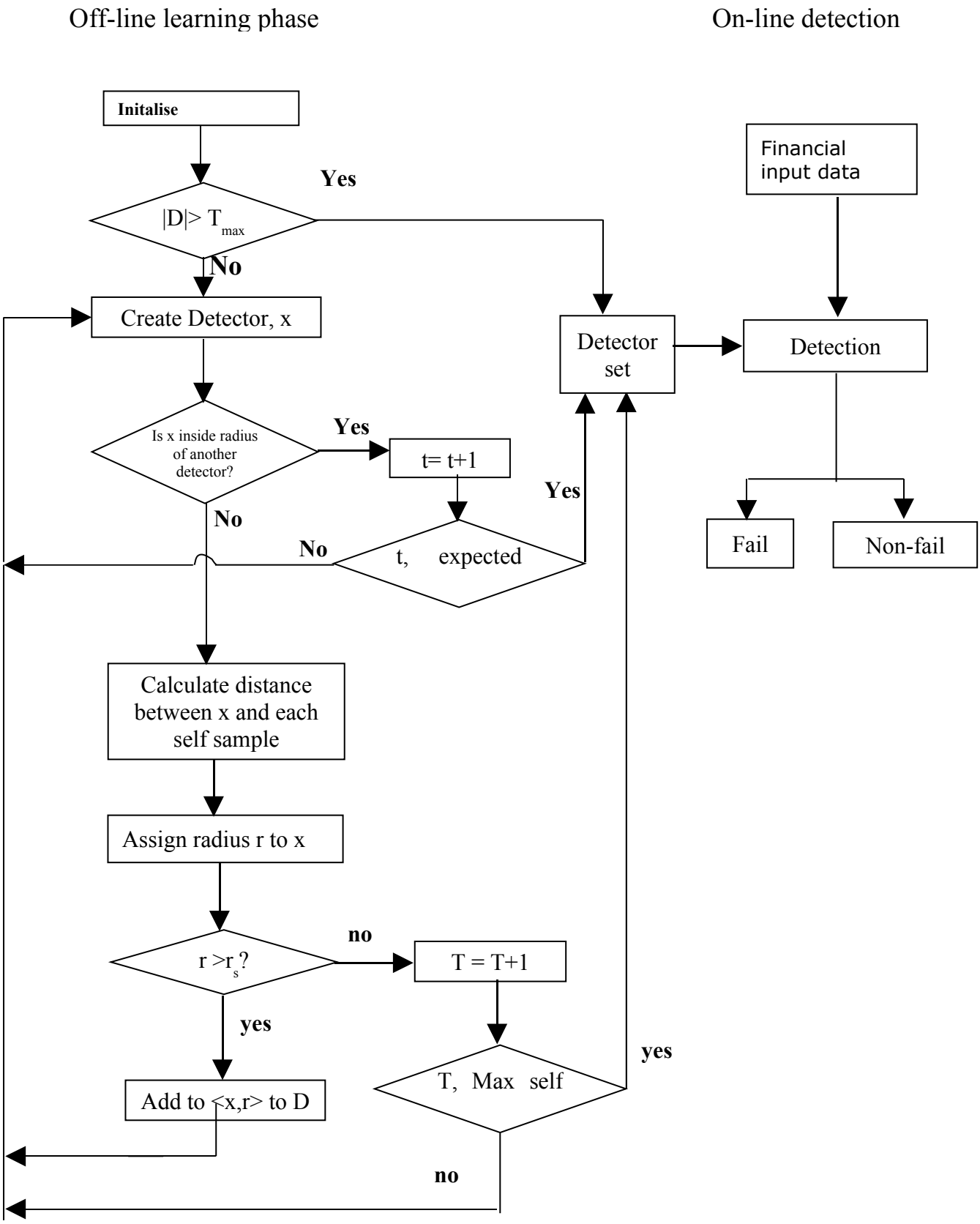


Fig 3.9 Flowchart of negative selection algorithm with variable-coverage detectors

Due to the structure of the algorithm several major loops were required, an outer *while loop* which exits when the maximum number of detectors have been generated. A *for loop* which loops through the current set of detectors of each randomly generated x and a *for loop* which loops through each member of the self sample for each randomly generated detector x. The input and output data structures and functions were retained from the unmodified negative selection algorithm.

3.4.4.4 Observations

Bond Rating

Initial tests of V-Detector on the bond rating data failed to detect any companies as junk grade. Following the rule of thumb "decreasing self radius results in increased detection rate and high false alarm" given by Dasgupata and Ji (2004) the self radius was reduced to r_s=0.01. The number of junk grade firms identified increased but not to a level comparable with that of the unmodified negative selection algorithm. The number of false alarms recorded also increased to an unacceptable level.

Decreasing the self radius has the effect of generating detectors with large associated radii. These large radii account for the high false alarm rate, however, the low detection rate is a result of there being too few detectors generated. The solution is to increase the number of detectors generated by altering the control variables. Increasing the expected coverage to 99% (θ=0.9) and the maximum number of detectors to 1000 should overcome this problem. However, the long runtime of the programme was prohibitive. The efficiency of the algorithm, its associated limitations and how they may be overcome will be discussed in more detail in the results and conclusions chapter.

Corporate Failure

The smaller size of the corporate failure dataset made it immediately more attractive for use with the V-Detector algorithm. With virtually no tuning, taking the recommended self radius r_s=0.01, max number of detectors = 500 and expected coverage = 99%, the average accuracy result for one year prior to failure was 80%, an increase of 6% on the unmodified algorithm. The accuracy

results were also significantly better for the data two and three years prior to failure, with no additional tuning required. Detailed results and comparisons are presented in the results and conclusions chapter.

The following confusion matrices provide a comparison of the unmodified and variable detector algorithms in terms of predicted versus actual bond ratings for all three years prior to failure. Where the format in terms of Type I and Type II errors is given in figure 3.10.

Predicted / **Actual**	Non-fail	Fail
Non-fail	Correct	Type 1
Fail	Type 2	Correct

Figure 3.10

Variable-Detector Algorithm

Predicted / **Actual**	Non-fail	Fail
Non-fail	77.78%	22.22%
Fail	15.05%	84.95%

Predicted / **Actual**	Non-fail	Fail
Non-fail	68.37%	31.63%
Fail	35.9%	64.1%

Predicted / **Actual**	Non-fail	Fail
Non-fail	46.32%	53.68%
Fail	53.77%	46.23%

Unmodified NS Algorithm

Predicted / **Actual**	Non-fail	Fail
Non-fail	69.78%	30.22%
Fail	18.28%	81.72

Predicted / **Actual**	Non-fail	Fail
Non-fail	60.15%	39.85%
Fail	36.55%	63.45%

Predicted / **Actual**	Non-fail	Fail
Non-fail	43.77%	56.23%
Fail	64.13%	35.87%

3.5 Summary

This chapter contains an overview of the algorithms investigated and implemented in the course of this study. Initially, the Unmodified Negative Selection Algorithm was tested extensively with both the bond rating dataset and the corporate failure dataset. The results achieved are displayed and discussed further in Chapter 4. The Modified NS Algorithm and the clonal selection algorithm were then implemented and tested but the results obtained did not compare favourably with those attained using the basic unmodified algorithm. The last section of this chapter deals with a recent extension made to the negative selection algorithm. The method of using variable detectors has, to date, been implemented on very few applications (Ji and Dasgupta, 2004). The results achieved for the implementation of the Negative Selection Algorithm with Variable Coverage Detectors are presented and examined in the last section of Chapter 4. All of the MATLAB code for each of the four algorithms can be found in Appendix A.

Chapter 4: Results and Discussion

4.1 Introduction

This chapter presents experimental results for both the unmodified negative selection algorithm and the variable detector algorithm. Figure 4.1 illustrates the testing strategy applied to the unmodified negative selection algorithm using the bond rating data set. As explained in the design issues section of chapter 3, three different percentage cut sizes (proportion of self used for training) were investigated and three different random re-cuts were made on each. The cut with the best average accuracy over ten runs of the program was selected for further testing.

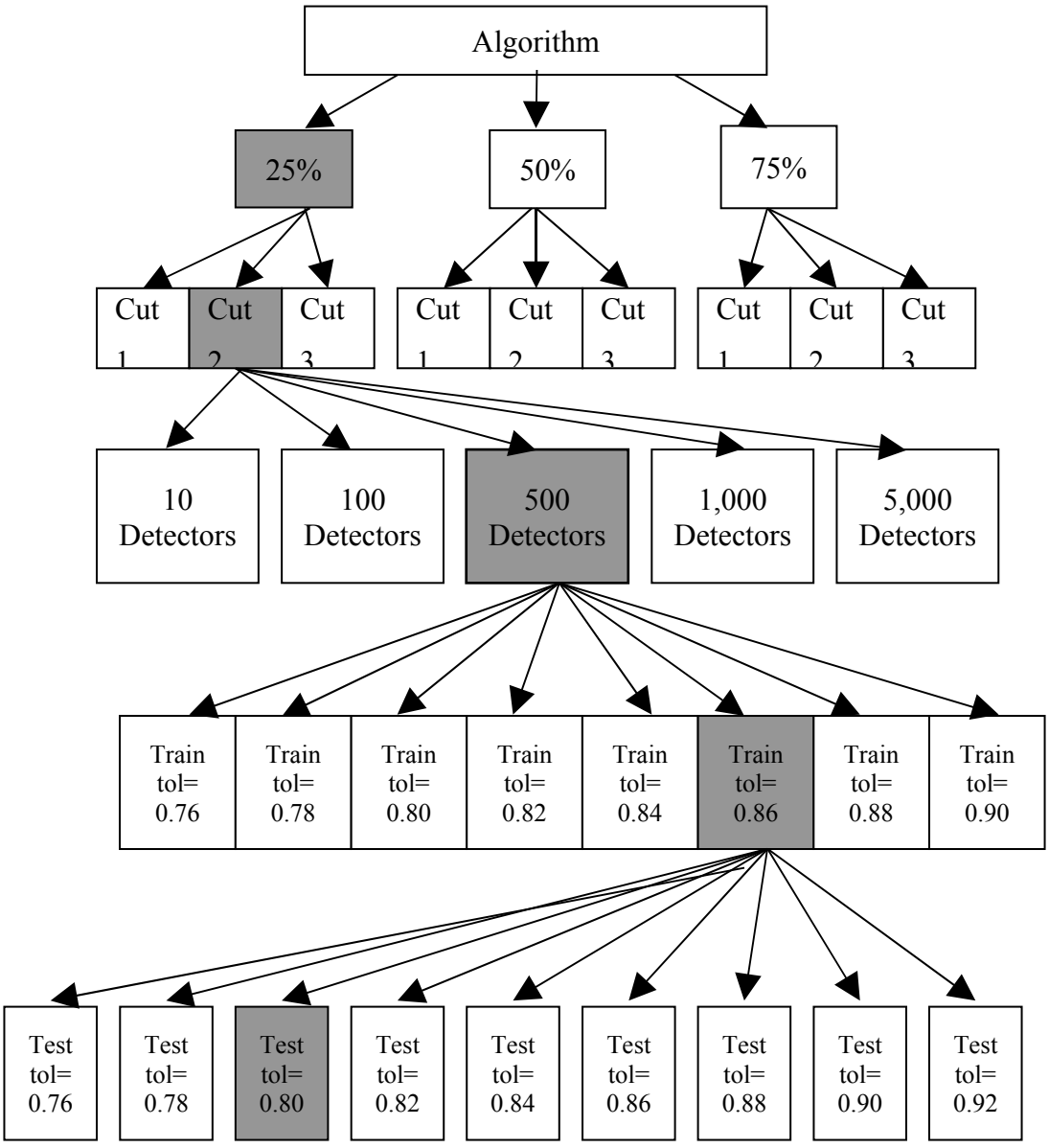


Fig 4.1 Test strategy for unmodified NS algorithm for Bond Ratings dataset

By trial and error approximate values were established for the three key variables:

- the number of detectors
- train-tolerance
- test-tolerance.

In Fig 4.1 above, the shaded options were found to be the optimum variable values. Each variable was tested over an appropriate range while the other two were fixed.

4.2 Bond Rating, Unmodified NS Algorithm Experimental Results

The following tables, 4.1, 4.2 and 4.3, give the results of each cut averaged over 10 runs of the algorithm for the 25%, 50% and 75% case. It can be clearly seen that using a large training set results in over-training of the model and consequently decreased accuracy. This is particularly apparent in the 75% case where over 50% of non-self are not detected.

Table 4.1 Results for 25% of self data selected for training

Cut Number	Overall Accuracy	% Investment grade identified as junk	% Junk grade not detected
1	70.57	22.75	35.99
2	69.53	40.41	20.65
3	72.47	44.91	10.36

Table 4.2 Results for 50% of self data selected for training

Cut Number	Overall Accuracy	% Investment grade identified as junk	% Junk grade not detected
1	61.87	28.69	44.34
2	65	28.74	39.09
3	65.24	45.29	24.35

Table 4.3 Results for 75% of self data selected for training

Cut Number	Overall Accuracy	% Investment grade identified as junk	% Junk grade not detected
1	58.47	27.66	46.08
2	57.24	18.47	50.71
3	56.91	21.71	50.09

The variation in results between cuts of the same size indicates that the model is to some degree reliant on the training set, and reinforces the importance of re-cuts during the training and testing stages.

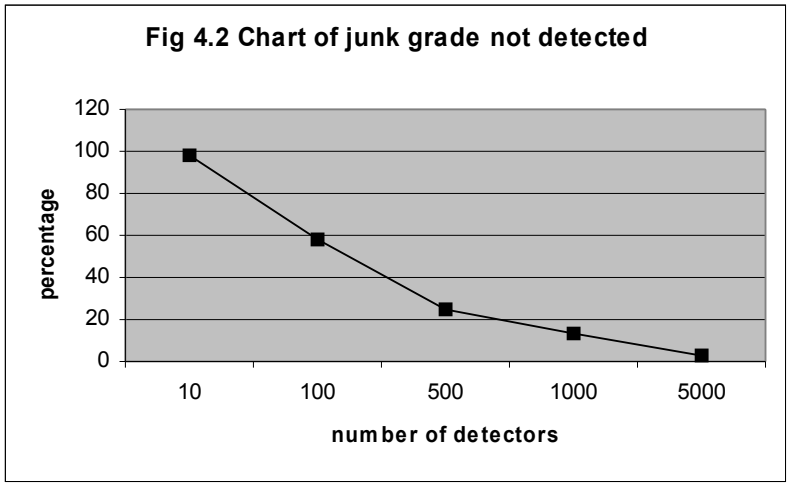
The re-cuts can also influence the balance between Type I and Type II errors. While an overall accuracy of 70.58% (table 4.1) appears better, almost 36% of junk grade companies were not identified as such (Type II error), whereas, for an overall

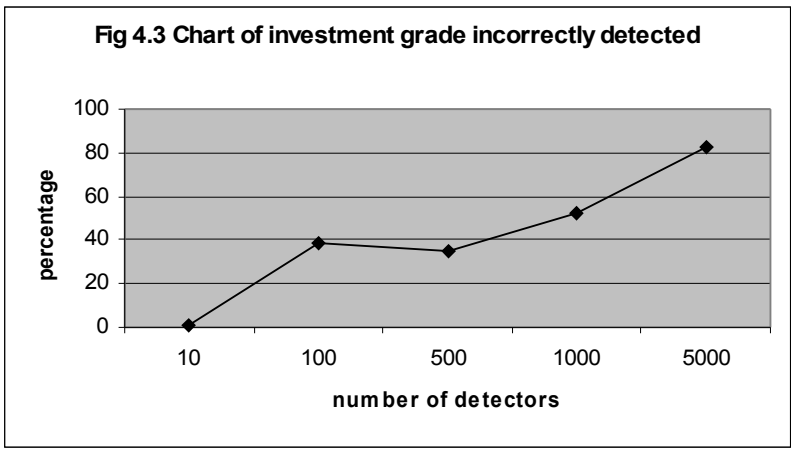
accuracy of 69.54%, only 20.65% of junk grade companies were not detected. For a danger recognition procedure such as this, it is generally desirable to err on the side of caution, i.e. increase detection rate at the expense of increased false alarms.

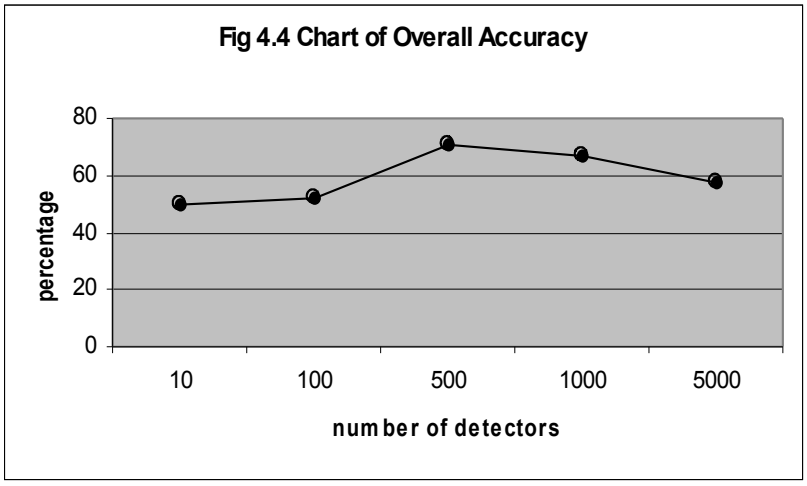
The effects of varying each of the three variables is discussed in the next three sections

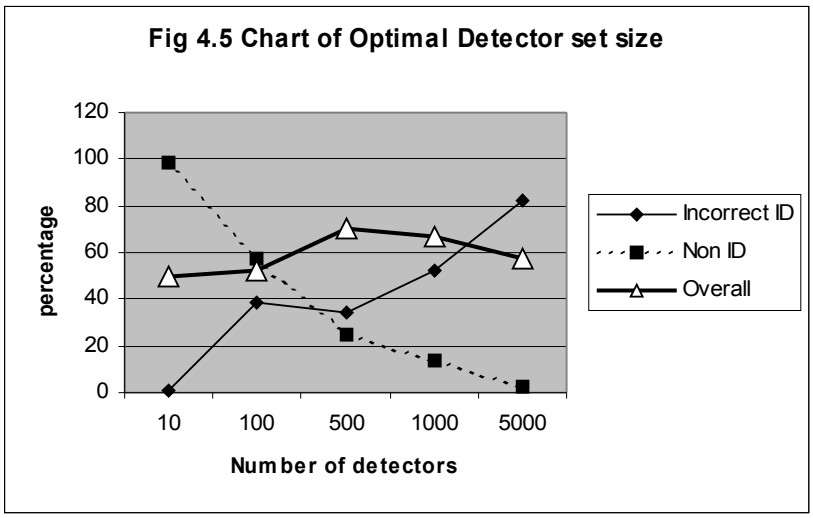
4.2.1 Number of detectors

The final parameter to impact on the results is the number of detectors generated, intuitively, too few will not cover the required space and will result in a large number of non-self vectors remaining undetected while too many may over train the model and cause an unacceptable amount of false alarms. In order to verify this and establish an optimum set size, the algorithm was run for 10, 100, 500, 1000 and 5000 detectors.



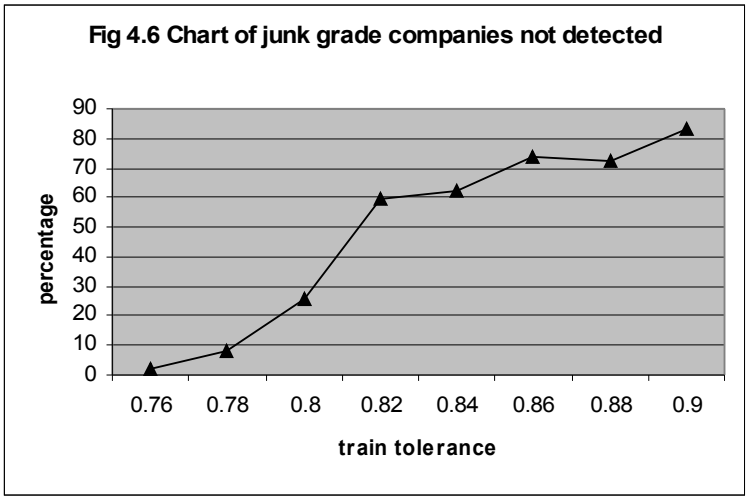






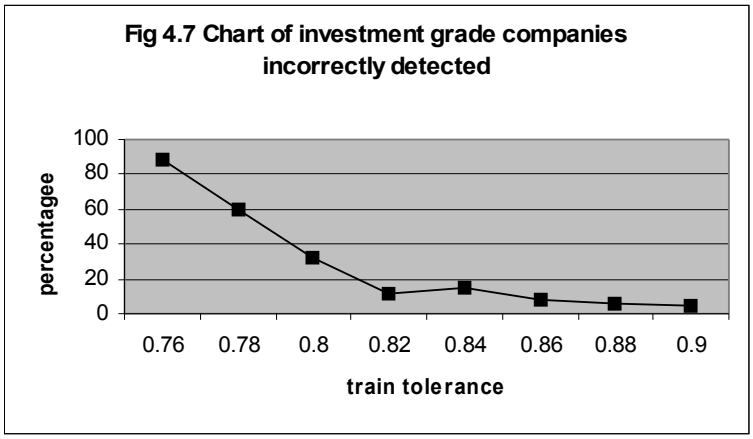
4.2.2 Effect of Train Tolerance

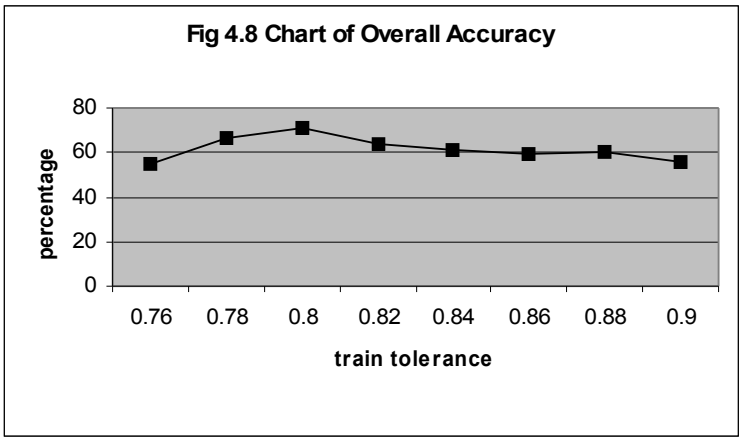
In order to investigate the effect of the train tolerance on the accuracy results, the number of detectors was set to 500 and the test tolerance to 0.86.Then the train tolerance was varied between 0.76 and 0.9 at intervals of 0.02.



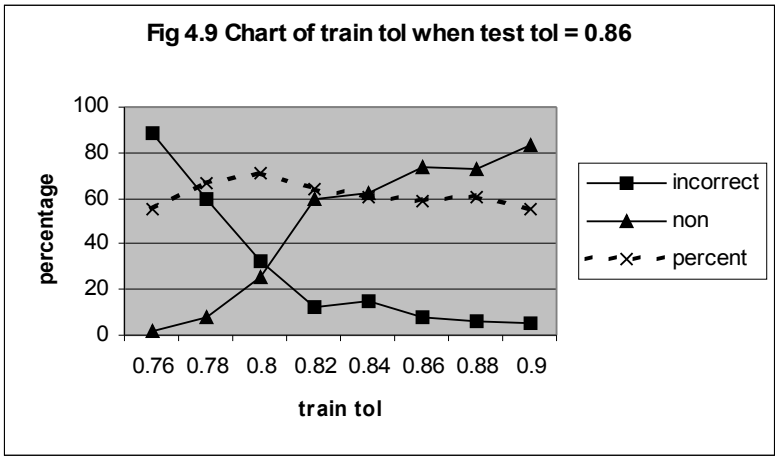
As illustrated above, increasing the train tolerance results in more non-investment (junk) grade companies being undetected. When the model is being trained, the large tolerance means that all the detectors are matching with self (investment grade companies) and are being destroyed. The remaining detectors have a very limited detection ability and fail to match any non-self for the given test tolerance

Likewise, the increasing training tolerance generates very few detectors which have an affinity with self, and the number of investment grade companies incorrectly identified as junk grade decreases rapidly, results which are the direct opposite of increasing the test tolerance. It is the relationship between the test and train tolerances that has the most profound effect on the accuracies, with the test tolerance being of the order of 0.05 greater than the train tolerance producing the best results.





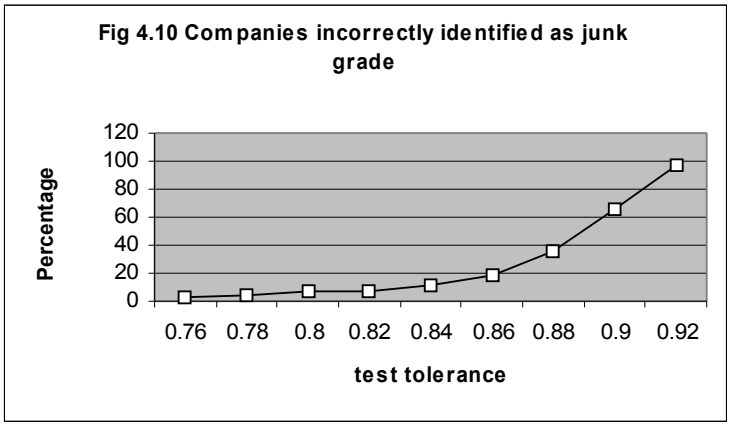
The overall accuracy can be seen to peak at around train tolerance = 0.8, at this point there is a balance between the detection and false alarm rates



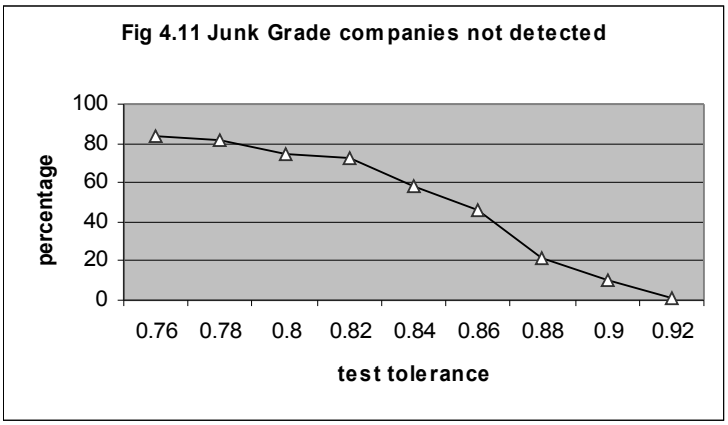
The above experiments show the importance of tuning the model to the given data. It was found to be relatively easy to tune, with good results being produced after a small number of guesses, and further systematic fine-tuning producing no significant improvement

4.2.3 Effect of Test Tolerance

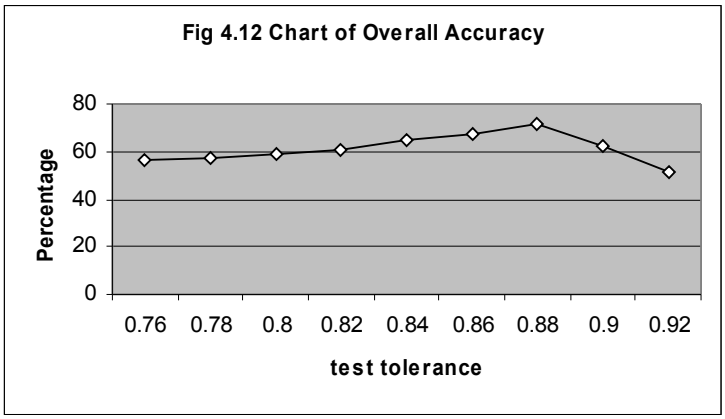
Similarly, to investigate the effect of the test tolerance on the accuracy results, the number of detectors was set to 500 and the train tolerance to 0.80. The test tolerance was varied from 0.76 to 0.92, in intervals of 0.02. The results are illustrated below.



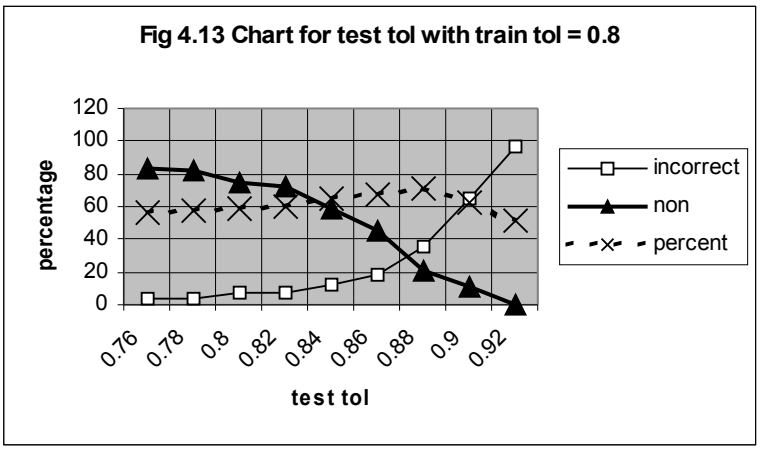
The percentage of investment grade companies incorrectly categorised as junk grade increases dramatically as test tolerance is increased. In reality, the detector radius size is being increase and hence more self-companies fall inside it.



The converse is true for the number of junk grade companies that are not identified as such. As the detector radii are increased the number of self and non-self being detected increase, resulting in a decline in the proportion of non-self that are not detected.



The overall accuracy is a measure of the proportion of all companies that are correctly categorised and thus requires a balance between detection rate and false alarm rate. It can be seen above that the accuracy value peaks at test tolerance = 0.88 for train tolerance of 0.8, while the following graph shows the relationship between the three accuracy measurements.



4.3 Corporate Failure, Unmodified NS Algorithm Experimental results

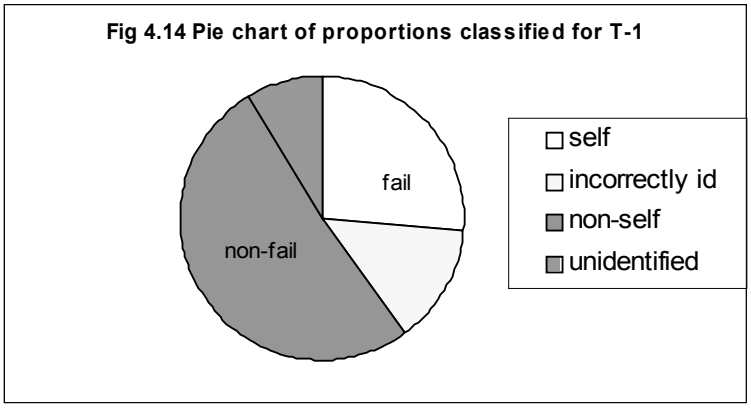
Similar tests were carried out on the unmodified algorithm for the corporate failure data. For each year prior to failure, T-1, T-2 and T-3 twelve cuts were examined, three in each of the following cut sizes; 20%, 40%, 60% and 80%. Observations were found to be consistent for the three years. The 40% cuts were the most accurate in each case, the 20% and 80% cuts tended to produce erratic results with accuracies differing by as much as 10% for the same cut. The results for each cut averaged over 10 runs of the algorithm are given in the tables in the following three sections.

4.3.1 One Year Prior to Failure, T-1

% of non-fail used for training	Cut Number	Overall Accuracy	% Non-fail incorrectly identified as fail	% Failure not detected
20	1	64.66666	44.66665	27.86666
20	2	63.25925	70.33333	9.86666
20	3	77.1111	34.16667	14.26666
40	1	78.1333	30.22223	18.28572
40	2	70.16667	20.66665	35.33333
40	3	74.08333	15.77777	32.00002
60	1	74.0001	11.33332	31.86666
60	2	70.38095	9.33333	37.73334
60	3	72.66667	17.33333	31.33332
80	1	64.3333	11.99999	40.40001
80	2	65.55557	13.3333	38.66666
80	3	63.66666	12.66664	41.06667

Table4.4 Results for one year prior to failure, T-1

The highest average accuracy achieved was 78.13%. 30.22% of the companies which did not fail were incorrectly classified as unhealthy while only 18.28% of the companies which failed in the following year were not identified as being in danger. The pie chart in Fig 4.14 illustrates the proportions of self and non-self data accurately classified

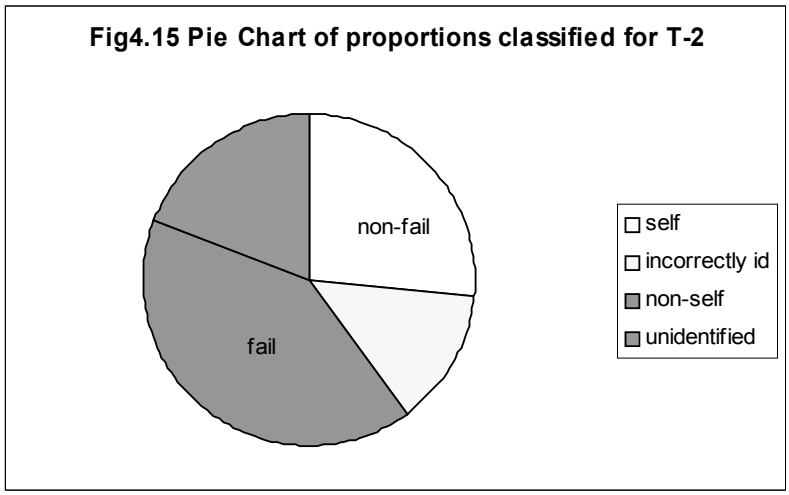


4.3.2 Two Years Prior to Failure, T-2

% of non-fail used for training	Cut Number	Overall Accuracy	% Non-fail incorrectly identified as fail	% Failure not detected
20	1	66.71877	54.73684	16.05634
20	2	61.32816	63.85966	18.45071
20	3	63.6719	52.45613	23.3803
40	1	64.73685	35.38557	37.1831
40	2	66.92984	33.48837	32.81691
40	3	64.64914	43.95349	30.14084
60	1	61.1	36.55171	39.85916
60	2	60.6	28.27587	43.94367
60	3	59.8	14.48277	50.70423
80	1	53.95352	20.66667	51.40846
80	2	47.6744	40.0021	54.9296
80	3	52.3256	20.4301	53.5211

Table 4.5 Results for two years prior to failure, T-2

For two years prior to failure, the best result was 66.92%. It can be seen from the pie chart below in Fig 4.15 that the proportion of misclassified data has increased significantly in comparison to the T-1 case.

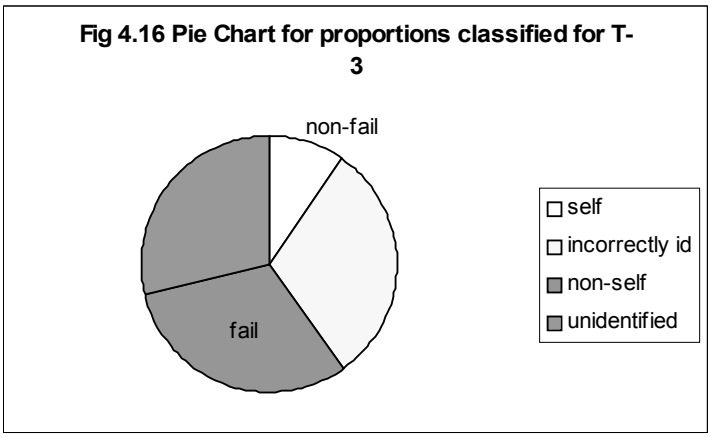


4.3.3 Three Years Prior to Failure,T-3

% of non-fail used for training	Cut Number	Overall Accuracy	% Non-fail incorrectly identified as fail	% Failure not detected
20	1	38.91304	76.39345	48.96103
20	2	57.10147	84.26229	10.12986
20	3	40.43478	76.22949	46.36362
40	1	38.29267	72.3913	55.32469
40	2	40.813	64.13043	56.23378
40	3	34.55284	62.39132	67.27276
60	1	36.2963	48.70968	69.74029
60	2	32.59259	50.64516	74.15587
60	3	37.59259	46.12904	68.96107
80	1	25.2174	319.9999	831.169
80	2	22.8261	26.6667	87.013
80	3	26.087	40	80.5195

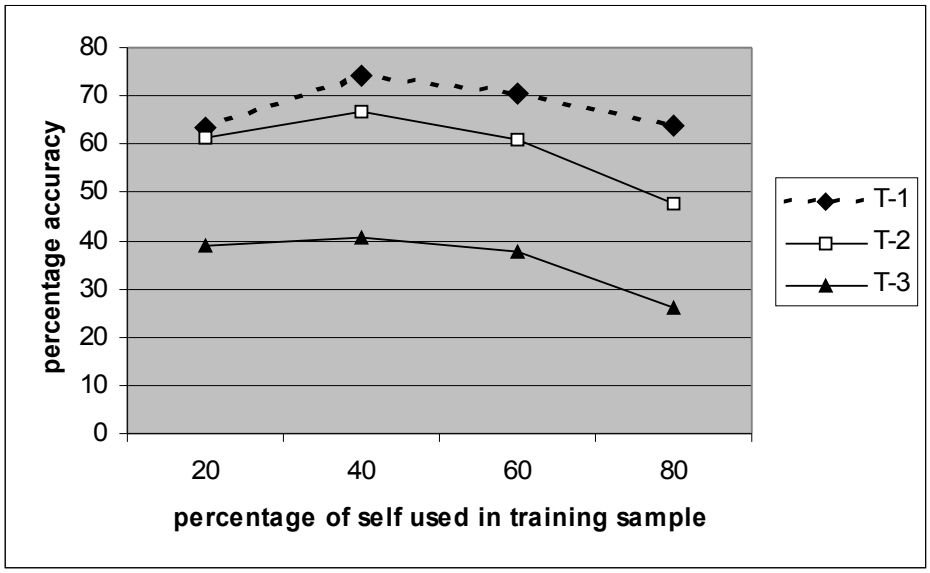
Table 4.6 Results for three years prior to failure, T-3

The experimental results observed for the data three years prior to failure are poor in virtually all cases, significantly lower than 50% (random chance) in many cases, with the exception of 20% cut 2, which correctly identified 89% of non-self firms. However, 84% of the self-companies were misidentified giving an overall accuracy of 57%. This further emphasises the point that the results must be interpreted not just in terms of overall accuracy but also in terms of the biasing of the results. The significance of the cut selection is also apparent.



The final graph illustrates the effect of the training sample size on the three data sets. The same trend is observed in all three cases, above 40% there is a steady decline in accuracy as a result of over training.

Fig 4.17 Chart of training sample size on three data sets



4.4 Variable Detector Algorithm

4.4.1 Corporate Failure Results

As outlined in section 4 of chapter 3, the variable detector negative selection algorithm was investigated for both the bond rating and corporate failure problems. However, due to the long run time associated with the larger bond rating data set, extensive testing and experimentation was confined to the corporate failure problem.

The algorithm was tested on the same cut sets as the unmodified version to enable direct comparisons. As the results for the 60% and 80% cut sets were considerably less accurate, only the 20% and 40% cases were considered. The results for each cut set averaged over 10 runs of the algorithm are given below. The balance between detection rate and false alarm rate is evident and the key to high overall accuracy.

T-1

Table 4.7 Variable Detector Results for one year prior to failure, T-1

% of self used for training	Cut Number	Overall Accuracy	% Non-fail incorrectly identified as fail	% Failure not detected
20	1	62.5185	67.66668	13.33332
20	2	62.2222	74.66668	8.26666
20	3	75.5556	38.3333	13.3333
40	**1**	**82.8**	**22.22222**	**15.04762**
40	2	70.33332	32	28.26668
40	3	74.50002	26.22222	25.06666

T-2

Table 4.8 Variable Detector Results for two years prior to failure, T-2

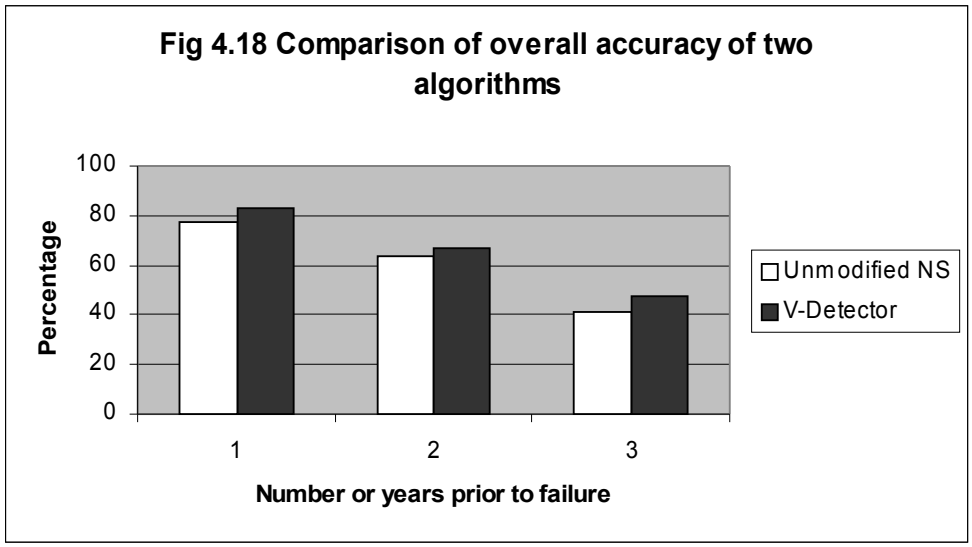
% of self used for training	Cut Number	Overall Accuracy	% Non-fail incorrectly identified as fail	% Failure not detected
20	1	66.56254	49.82456	20.28172
20	2	57.50004	58.59648	29.57748
20	3	61.09376	62.80702	19.71832
40	1	62.80702	29.30234	41.97182
40	2	63.85966	42.32558	32.39438
40	**3**	**65.98774**	**31.62792**	**35.9014**

T-3

Table 4.9 Variable Detector Results for three years prior to failure, T-3

% of self used for training	Cut Number	Overall Accuracy	% Non-fail incorrectly identified as fail	% Failure not detected
20	1	47.39132	53.77048	51.6883
20	2	68.8406	58.68852	9.35064
20	3	41.44928	56.06558	60.5195
40	1	38.86178	52.1739	66.49354
40	2	42.60162	43.47828	65.71432
40	3	36.74798	46.08698	73.50652

The variable detector method was found to be more accurate in all cases. For 40% cut-1 in T-1, the overall accuracy was 78.13% in the unmodified case versus 82.8% for V-Detector with the number of incorrectly identified self instances decreased from 30.2% to 22.2% and the number of non-self not detected down from 18.3% to 15.04%. Similarly, for T-2 40% cut-3, the overall accuracy increased to almost 66%. Fig 4.18 displays the average results for all three years using both algorithms. The V-Detector method is visibly more accurate in all instances.



4.4.2 Bond Rating, Variable Detector Algorithm

The best result recorded for the bond rating problem using the variable detector method was 69%, slightly less than the average result attained by the unmodified version. However, the increase in undetected non-self was significant.

Table 4.10 Classification accuracies for the Bond Rating dataset using the Variable Detector

	Predicted investment grade	Predicted junk grade
Actual investment grade	80.9%	19.1%
Actual junk grade	42.01%	75.99%

As mentioned in Chapter 3, this finding was due to not being able to strike the balance between accuracy tuning and acceptable run-time. Given greater computing power and/or coding in a lower level language (e.g. java), it is possible that high levels of accuracy could be obtained using this method. Careful consideration, however, must be given to the trade off between development time, run time and accuracy when compared with the unmodified version. A comparison and discussion of algorithm efficiencies is given in the next section.

4.5 Efficiency Analysis

Artificial Immune Systems in general compare favourably with other artificial intelligence techniques such as neural networks in terms of complexity and efficiency. However, it is clear from the experiments carried out that significant disparities exist between algorithm variants. As the detection phase is constant for all algorithms considered, analysis is confined to the training phase, i.e. the method by which the detector set is generated. The time complexity of the unmodified negative selection algorithm is $O(m|S|)$, where m is the preset number of detectors and |S| is the size of the training self sample set. The major computational expense is finding the distance between every detector and every self-sample, two *for* loops are required to cycle through both data sets. In the case of the variable-detector algorithm however, there are two main phases:

1. Finding the distance between the detector being considered and every self sample

2. Finding the distance between the detector being considered and every other detector.

This requires four *for* loops to cycle through the data. Thus, it is immediately clear that a longer run time will result for the same detector-set size m and training-set size |S|.

The difference in run-time observed suggests that the variable detector algorithm may not scale up to very large data types even given increased computing power.

4.6 Chapter Conclusion

Comparing the results of this study with those of prior research suggests artificial immune systems, in particular negative selection methods, using financial data as inputs, are capable of producing prediction accuracies for bond rating and corporate failure that are extremely competitive. In both cases, the average results are at least as good, while many specific cuts give superior accuracies. These observations are encouraging as the algorithm is extremely portable, with implementation requiring few, if any, modifications for adaptation to other data sets and classification problems. In comparison, applications of neural networks are not a trivial task, as many choices are open to the modeller, including the nature of the connection structure, number of layers of nodes, number of nodes in each layer etc. (Brabazon & Keenan, 2003). Thus, time spent on development and tuning in AIS is conserved, thus making it an attractive alternative to both neural networks and statistical methods for these important financial problems.

Chapter 5: Conclusion

The core research of this study was to investigate the suitability of Artificial Immune Systems to the financial problems of bond rating and corporate failure prediction. This chapter presents a summary of the principle findings, outlines their implications and gives suggestions for further studies.

Four key AIS algorithms were used to build models for the corporate failure and bond rating problems: Negative Selection (NS), Modified Negative Selection, Clonal Expansion and Variable-Detector Negative Selection. Of these, the focus was placed on the negative selection algorithm and the recently proposed variable detector modification. Encouraging results were obtained from initial tests on the unmodified negative selection algorithm, comparable with those of Neural Networks, making it the obvious choice for further research.

5.1 Comparison of Results

The most exciting recent development in NS is the variable detector (V-Detector) algorithm proposed by Ji and Dasgupta (June 2004). To date, V-Detector has not been implemented on real world data with testing confined to standard benchmark datasets for classification problems. V-detector proved to be a significant improvement on the unmodified NS algorithm in the corporate failure case with results for prediction one year prior to failure reaching 80%. Although this is equalled by some hybrid neural network models (Brabazon & Keenan 2003, Dutta & Shekhar 1988), the demand placed on development resources is significantly greater as well as requiring considerable modification for different datasets and problems. This study has shown that AIS is an attractive design alternative to existing methods such as neural networks, regression and case-base reasoning. It produces accuracy results that compare favourably with these methods while conserving the time and effort associated with development, tuning and modification.

5.2 Future Study

The promising results presented in this study suggest that there is a strong basis for further research. This study limits the bond rating case to binary classification (i.e. investment grade and junk grade). An obvious extension of this is to increase the

number of prediction classes to distinguish between the ten individual classes. With increasing number of classes, a decrease in accuracy should be expected, as is the case in neural network models (Huang *et Al*., 2004). The proposed algorithms could also be tested further on datasets of varying size, financial ratios and industry sectors with more rigorous testing of the effect of the key control variables and training sets. While further investigation into the functions required for mutation in the positive selection and clonal expansion algorithms, it can be assumed that their use in hybrid algorithms would improve classification accuracy. Unlike the natural immune system, examples exist of non-self (fail or junk grade) companies which could be used to train a hybrid model. As the concept of variable detectors is so new, there is a considerable amount of possibilities that have not yet been investigated. While this study concentrates on variable size detectors, the idea could be extended to include detectors of variable shapes and variable dimensions. The algorithm could also be combined with other AIS concepts such as clonal expansion. A more eloquent implementation of the code in a lower-level language (e.g. Java) could considerably enhance the run-time of the V-Detector algorithm. Further research to assess its applicability to large real world data sets would prove beneficial.

This study has shown that the AIS approach to the classification of bond ratings and prediction of corporate failure is not only computationally accurate but an attractive design method for management scientists.

Bibliography

Brabazon, A., O'Neill, M. (2003). Overview of Artificial Immune Systems.

Brabazon, A.(2003). Corporate Bond Rating using Neural Networks

Brabazon, A., Keenan, P.B.(2003). A Hybrid Genetic Model for the Prediction of Corporate Failure.

Bradley, D.W, Tyrell,A.M.(2000). Immunotronics: Hardware Fault Tolerance Inspired by the Immune System. Lecture Notes in Computer Science, 1801.

de Castro, L.N., Timmis, J.(2002). Artificial Immune Systems: A New Computational Intelligence Approach .London: Springer.

Dasgupta, D.(1997). Artificial Neural Networks and Artificial Immune Systems: Similarities and Differences. IEEE

Dasgupta, D.(1999). Immunity-Based Intrusion Detection System: A General Framework. Proc. of the 22nd NISSC.

Dasgupta, D., Coa,Y., Yang, C.(1999).An immunogenetic Approach to Spectra Recognition . Proc. of the Genetic and Evolutionary Computation Conference

Dasgupta, D.(2000). An agent based architecture for a computer virus immune system. Proc. GECCO Workshop Artificial Immune Syst.

Dutta, S., Shekhar S., (1988) Bond rating: a non-conservative application of neural networks, Proceedings of IEEE International Conference on Neural Networks, pp. II443– II450.

Ederington, H.L., (1985) Classification models and bond ratings, Financial Review 20 (4) (237–262.

Esponda, F., Forrest, S, Helman (2004). A Formal Framework for Positive and Negative Detection Schemes. IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, Vol. 34, No. 1

Forrest, S, Perelson, S.A., Allen,L., Cherukuri, R.,(1994). Self – Nonself Discrimination in a Computer. Proc. IEEE Symp. Research Security Privacy.

Gonzalez, F.A., Dasgupta, D. (2002). Anomaly Detection Using Real-Valued Negative Selection. Genetic Programming and Evolvable Machine.

Huang, Z., Chen, H., Hsu, C., Chen, W., Soushan, W.(2003). Credit rating analysis with support vector machines and neural networks: a market comparative study. Decision Support Systems, (Article in Press).

Ji, Z., Dasguptq, D.(2004). Real-Valued Negative Selection Algorithm with Variable-Sized Detectors.

Kephart, J.O(1994). A Biologically Inspired Immune System for Computers. Artificial Life IV Proc. of the Fourth International Workshop on the Synthesis and Simulation of Living System, MIT Press.

Kharbanda, O.P., Stallworthy, E.A.(1985). Corporate Failure: Prediction, Panacea and Prediction.London: McGraw-Hill

Kim, J.W., (1993) Expert systems for bond rating: a comparative analysis of statistical, rule-based and neural network systems, Expert Systems 10 167–171.

Lee, D-W., Sim,K-B(1997). Arificial Immune Network-Based Cooperative Control in Collective Autonomous Mobile Robots. Proc. of the IEEE Int. Workshop on Robotics and Human Communication.

Mahe,r J.J., Sen T.K., (1997) Predicting bond ratings using neural networks: a comparison with logistic regression, Intelligent Systems in Accounting, Finance and Management 6 59– 72.

Meshref, H., VanLangdingham, H. (2000). Artificial Immune Systems: Application to Autonomous Agents. IEEE

Moody, J., Utans J., (1995) Architecture selection strategies for neural networks application to corporate bond rating, in: A. Refenes (Ed.), Neural Networks in the Capital Markets, Wiley, Chichester, pp. 277–300.

Pinches, G.E., Mingo K.A., (1973) A multivariate analysis of industrial bond ratings, Journal of Finance 28 (1) 1 –18.

Shin K.S., Han I., (2001) A case-based approach using inductive indexing for corporate bond rating, Decision Support Systems 32 41–52.

Singleton, J.C., Surkan A.J., (1990) Neural networks for bond rating improved by multiple hidden layers, Proceedings of the IEEE International Conference on Neural Networks, pp. 163– 168.

Timmis, J., Neal, M. (2001). A Resource Limited Artificial Immune System. Knowledge Based Systems, 14.

Appendix A: MATLAB Code

1. The code for the Unmodified Negative Selection Algorithm:

```
%%%%%%%UNMODIFIED NEGATIVE SELECTION ALGORITHM%%%%%%%%%%%%%%%

load selfsampleT3_40__no_cut2.txt
load testsetT3_40__no_cut2.txt
rand('state',sum(100*clock))

%%%%%%%CHANGEABLE DEPENDING ON SAMPLE AND TEST SETS%%%%%%%%%%

selfsample = selfsampleT3_40__no_cut2; %NAMES OF TEXT FILES
testset = testsetT3_40__no_cut2;
numSelfRows = 46;  %NUMBER OF ROWS OF "SELF"
numDetcs = 500 %NUMBER OF DETECTORS
numMean = 5; %NUMBER OF NEAREST NEIGHBOURS TO BE COMPARED
AGAINST (+1)
trainTol = 0.81; %TRAINING TOLERENCE
testTol = 0.79; %TEST TOLERENCE

%%%%%%%%%%%%%%%%%%%%%%INITIALIZING%%%%%%%%%%%%%%%%%%%%%%%%%%

[m,n] = size(testset);
[l,p] = size(selfsample);
numRows = m;
numSamps = l;
numNonSR = numRows - numSelfRows;

%%%%%%%%%%%%%%%%%%%%%%TRAINING%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

detectors = rand(numDetcs,8);
numSelf = 1;
while numSelf > 0
    numSelf = 0;
    for y = 1:numDetcs % detectors rows
      sum_euclid = 0;
      for z = 1:numSamps % training rows
        total=0;
        difference = detectors(y,:)-selfsample(z,:);
        squared = difference.^2;
        total = sum(squared);
        euclid_dist(1,z) = sqrt(total);
      end%endfor z

    h = 1;
    while h < numMean;
        [shortest,I] = min(euclid_dist);
        euclid_updated(1,h) = shortest;
        euclid_dist(1,I) = 10;
        h = h + 1;
    end %endwhile h
```

```
    median_euclid = sum(euclid_updated)/(numMean-1);

    if median_euclid < trainTol;
        detectors(y,:) = rand(1,8);
        numSelf = numSelf+1;
    end %endif
    med_euclid(1,y) = median_euclid;
  end %endfor y
end %endwhile
%numSelf

%%%%%%%%%%%%%%%%%%%%%%%%TESTING%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for z = 1:numDetcs %detectors rows
    for c = 1:numRows %test rows
        total2=0; % initialise total to zero
       for a = 1:8 %columns
        difference2(1,a) = testset(c,a)-detectors(z,a);
       end %endfor a
      squared2 = difference2.^2;
      total2 = sum(squared2);
      euclid_dist2 = sqrt(total2);
       if euclid_dist2 < testTol %%!!!
        testset(c,10) = 0.0;
       end %endif
   end %endfor c
end %endfor z

%%%%%%%%%%%%%%%%%ACCURACY RESULTS%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

accuracy = (testset(:,10) - testset(:,9));
temp = sum(accuracy);
inaccs = sum(abs(accuracy));

incorrectly_id = (inaccs - temp)/2;
non_id = inaccs - incorrectly_id;
incorrectly_id_p = (incorrectly_id*100)/numSelfRows
non_id_p = (non_id*100)/numNonSR
percentage = 100 - (inaccs*(100/numRows))

%%%%%%%%%%%%%%%%%END OF PROGRAM%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

2. The code for the Modified Negative Selection Algorithm:

```
%%%%%%%%%%MODIFIED NEGATIVE SELECTION ALGORITHM%%%%%%%%%%

load selfsample3.txt
load testset3.txt

rand('state',sum(100*clock));

%%%%%%%%%%Changeable variable values for algorithm%%%%%%%%

no_of_detectors = 1000;
no_of_training_rows = 120;
no_of_test_rows = 784;
no_of_nearest_cells = 10;
adapt_rate = 0.01;
decay_parameter = 2;

detectors = rand(no_of_detectors,8);

%%%%%%%%%%%%%%%%%%The training algorithm%%%%%%%%%%

numSelf = 1;
trainTol = 0.7;

age(1,1000) = 0;

iterations = 0;
while  iterations < 5;
    for y = 1:no_of_detectors % detectors rows
      total_nearest_cells = 0;
      for z = 1:no_of_training_rows % training rows
        total=0;
        difference = detectors(y,:)-selfsample3(z,:);
        squared = difference.^2;
        total = sum(squared);

        euclid_dist(1,z) = sqrt(total);
      end

      h = 1;
      while h < no_of_nearest_cells + 1;
         [selDist, colI] = min(euclid_dist,[],2);
         [short, rowI] = min(selDist);
         shortest(h,:) = detectors(colI(rowI),:);
         euclid_dist(:,colI(rowI))= 10;
         h = h + 1;
      end
      topline_mu_detect = 0;
      % To calculate mu
      for k = 1:no_of_detectors
          detector_total = 0;
          detector_difference = detectors(y,:) -
detectors(k,:);
```

```
          detect_squared = detector_difference.^2;
          detector_total = sum(detect_squared);
          detect_euclid(1,k) = sqrt(detector_total);
          mu_detect(1,k) = exp((-((detect_euclid(1,k)))^2)/
(2*((trainTol)^2)));
          topline_mu_detect  = topline_mu_detect +
mu_detect(1,k)*(detector_difference);
      end


      total2 = 0;

      median_NearestSelf = shortest(5,:);

      difference_nearest_self = detectors(y,:) -
shortest(5,:);
      squared2 = difference_nearest_self.^2;
      total2 = sum(squared2);

      euclid_dist_nearest_self = sqrt(total2);

      total_nearest_cells = 0;
      sum_of_difference = 0;

      if euclid_dist_nearest_self < trainTol;
            for j = 1:no_of_nearest_cells
                  total_nearest_cells = total_nearest_cells
+ (shortest(j,:));
                  sum_of_difference = sum_of_difference +
(detectors(y,:) - shortest(j,:));
            end

            dir =  (sum_of_difference)/
(abs(total_nearest_cells));

            if age(1,y) > 5
               detectors(y,:) = rand(1,8);
               temp1 = y
            else
              age(1,y) = age(1,y) + 1;
              detectors(y,:) = detectors(y,:) +
(adapt_rate*(exp(-(iterations/decay_parameter))))*dir;
              temp2 = y
            end
      else
            age(1,y) = 0;
            dir = sum(topline_mu_detect)/sum(mu_detect);
            detectors(y,:) = detectors(y,:) +
(adapt_rate*(exp(-(iterations/decay_parameter))))*dir;
            temp3 = y
      end %endif

  end %endfor

  iterations = iterations + 1
end %endwhile
```

```
%%%%%%%%%%%%%%%%%%The Test Algorithm%%%%%%%%%%%%%%%%%%%%%%%%%%

testTol = 0.835;
numNonSelf=0;

for z = 1:no_of_detectors %detectors rows

    for c = 1:no_of_test_rows %test rows
       total2=0; % initialise total to zero

        for a = 1:8 %columns
            difference2(1,a) = testset3(c,a)-detectors(z,a);
        end

        squared2 = difference2.^2;
        total2 = sum(squared2);
        euclid_dist2 = sqrt(total2);


    if euclid_dist2 < testTol %%!!!
        testset3(c,10) = 0.0;
        numNonSelf = numNonSelf+1;
    end %endif
    end %endfor c
   %numNonSelf
end %endfor z
numNonSelf

%%%%%%%%%%%%%%%%%%Accuracy Calculation%%%%%%%%%%%%%%%%%%%%%%%%%%

accuracy = (testset3(:,10) - testset3(:,9));
temp = sum(accuracy);
inaccs = sum(abs(accuracy));

incorrectly_id = (inaccs - temp)/2
non_id = inaccs - incorrectly_id
percentage = 100 - (inaccs*(100/no_of_test_rows))
incorrectly_id_p = (((inaccs - temp)/2)*100)/445
non_id_p = ((inaccs - incorrectly_id)*100)/339

%%%%%%%%%%%%%%%%%%END OF PROGRAM%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

3. The code for the Clonal Selection Algorithm:

```
%%%%%%%%%%CLONAL SELECTION ALGORITHM%%%%%%%%%%

load nonselfsample.txt

numDetcs = 500
numSamp = 50;
testDetcs = rand(numDetcs,8);


    for y = 1:numSamp % sample rows
      sum_euclid = 0;

      for z = 1:numDetcs % detector rows
        total=0;
        difference = testDetcs(z,:)-nonselfsample(y,:);
        squared = difference.^2;
        total = sum(squared);
         euclid_dist(y,z) = sqrt(total);          %%1st sample
and every detector
      end
    end %endfor

    h = 1;
    while h < 51;
         [selDist, colI] = min(euclid_dist,[],2);
         [short, rowI] = min(selDist);
         shortest(h,:) = testDetcs(colI(rowI),:);
         euclid_dist(:,colI(rowI))= 10;
         h = h + 1;
    end%%endwhile

%%%%%%%%%%% Need to mutate 20 detectors %%%%%%%%%%%%%%%%

for x=1:50

    for v = 1:5
        finDetcs((v+(x-1)*5),:)=shortest(x,:)+rand(1,8)/10000;
    end
    finDetcs(250+x,:)=shortest(x,:);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

load testset4.txt
numFinDetcs = 300;
testTol = 0.6;
numNonSelf=0;

for z = 1:numFinDetcs %detectors rows

    for c = 1:732 %test rows
       total2=0; % initialise total to zero
```

```
        for a = 1:8 %columns
            difference2(1,a) = testset4(c,a)-finDetcs(z,a);
        end

        squared2 = difference2.^2;
        total2 = sum(squared2);
        euclid_dist2 = sqrt(total2);


    if euclid_dist2 < testTol %%!!!
        testset4(c,10) = 0.0;
        numNonSelf = numNonSelf+1;
    end %endif
    end %endfor c
   %numNonSelf
end %endfor z
numNonSelf

%%%%%%%%%%%%%%%%%%ACCURACY CALCULATION%%%%%%%%%%%%%%%%%%%

accuracy = (testset4(:,10) - testset4(:,9));
temp = sum(accuracy);
inaccs = sum(abs(accuracy));

incorrectly_id = (inaccs - temp)/2;
non_id = inaccs - incorrectly_id;
incorrectly_id_p = (((inaccs - temp)/2)*100)/393
non_id_p = ((inaccs - incorrectly_id)*100)/339
percentage = 100 - (inaccs*(100/732))

%%%%%%%%%%%%%%%%%%%%%END OF PROGRAM%%%%%%%%%%%%%%%%%%%%%
```

4. The MATLAB code for the Variable Detector Algorithm:

```
%%%%%%%%%%%%%%%%VARIABLE DETECTOR ALGORITHM%%%%%%%%%%%%%%%%%

load selfsampleT3_40__no_cut3.txt        %%load in sample set
load testsetT3_40__no_cut3.txt
rand('state',sum(100*clock))
selfsample = selfsampleT3_40__no_cut3;   %NAMES OF TEXT FILES
testset = testsetT3_40__no_cut3;
numSelfRows = 46;
[m,n] = size(testset);
[l,p] = size(selfsample);
numRows = m;
numSamps = l;
numNonSR = numRows - numSelfRows;


%%%%%%%%%%%%%%%% TRAINING%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% V-Detector %%

Tmax = 1000;                 %%max number of detectors
Rs = .09;                    %%self radius
Co = .999;                    %%expected coverage
SCmax = 0.99;                            %max self coverage
D = [];   %initialising detector set to null
numD = 0;
count1 = 0;
count2 = 0;
w = 1;                           %%for placing detectors

t = 0;
T = 0;


while numD < Tmax
x = rand(1,8);
r = inf;
[m,n] = size(D);
numD = m;
[l,p] = size(selfsample);
numSamps = l;
inside = 0;

for y = 1:numD      %%for every row in detectors
  %%%%getting dist from di to location of x
    sum_euclid = 0;
    total=0;
    for a = 1:8 %columns
        difference(1,a) = D(y,a)- x(1,a);
    end %endfor a
    squared = difference.^2;
    total = sum(squared);
    dd = sqrt(total);
    %%%%%%%%returning dist dd
```

```
    if dd <= D(y:9)
        t = t+1;
        inside = 1;
    end %%if
    if t >= 1/(1-Co)
        numD = Tmax;   %%return detectors continue to test
phase/// ensures exits while loop
        g=0;
    end %%if
    count1 = count1+1;
end%endfor y


for z = 1:numSamps %% for each s
    %%%%% get dist from s to x
    sum_euclid = 0;
    total=0;
    difference = selfsample(z,:)- x;
    squared = difference.^2;
    total = sum(squared);
    d = sqrt(total);
    %%%%%%%%%%% return dist d

    if d - Rs <=r
        r = d - Rs;
    end %%if

     count2 = count2 +1;
end %% for z

    if (r>Rs)&(inside<1)
        D(w,:) = [x,r];
        w = w+1; %%incremets position of while loop
    end
    if r<Rs
        T = T+1;
    end %%else/if

   if T > 1/(1 - SCmax)
       numD = Tmax; %%exits while loop
       f=0;
   end %%i

end %%while

count1;
count2;

%%%%%%%%%%%%%%%%%%%TESTING%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[m,n] = size(D);
numD = m;

for b = 1:numD %detectors rows
    for c = 1:numRows %test rows
```

```
        total2=0; % initialise total to zero
       for e = 1:8 %columns
        difference2(1,e) = testset(c,e)- D(b,e);
       end %endfor e
      squared2 = difference2.^2;
      total2 = sum(squared2);
      euclid_dist2 = sqrt(total2);
       if euclid_dist2 < D(b,9) %%!!!
        testset(c,10) = 0.0;
       end %endif
   end %endfor c
end %endfor b

%%%%%%%%%%%%%%%ACCURACY RESULTS%%%%%%%%%%%%%%%%%%%%%%%%%%%%

accuracy = (testset(:,10) - testset(:,9));
temp = sum(accuracy);
inaccs = sum(abs(accuracy));

incorrectly_id = (inaccs - temp)/2;
non_id = inaccs - incorrectly_id;
incorrectly_id_p = (((inaccs - temp)/2)*100)/numSelfRows
non_id_p = ((inaccs - incorrectly_id)*100)/numNonSR
percentage = 100 - (inaccs*(100/numRows))

%%%%%%%%%%%%%%%END OF PROGRAM%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```