12-14-2001

AIRS: a Resource Limited Artificial Immune Classifier

Andrew B. Watkins

AIRS: A RESOURCE LIMITED ARTIFICIAL IMMUNE CLASSIFIER

By

Andrew B. Watkins

A Thesis
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Computer Science
in the Department of Computer Science

Mississippi State, Mississippi

December 2001

AIRS: A RESOURCE LIMITED ARTIFICIAL IMMUNE CLASSIFIER

By

Andrew B. Watkins

Approved:

Lois C. Boggess Professor of Computer Science (Major Professor)	Susan M. Bridges Professor of Computer Science (Committee Member)
Donna S. Reese Associate Professor of Computer Science (Committee Memeber)	Rayford B. Vaughn Associate Professor of Computer Science (Committee Member)
Julian E. Boggess Associate Professor of Computer Science and Graduate Coordinator of the Department of Computer Science	A. Wayne Bennett Dean of the College of Engineering

Name: Andrew B. Watkins

Date of Degree: December 14, 2001

Institution: Mississippi State University

Major Field: Computer Science

Major Professor: Dr. Lois C. Boggess

Title of Study: AIRS: A RESOURCE LIMITED ARTIFICIAL IMMUNE CLASSIFIER

Pages in Study: 81

Candidate for Degree of Master of Science

The natural immune system embodies a wealth of information processing capabilities that can be exploited as a metaphor for the development of artificial immune systems. Chief among these features is the ability to recognize previously encountered substances and to generalize beyond recognition in order to provide appropriate responses to pathogens not seen before.

This thesis presents a new supervised learning paradigm, resource limited artificial immune classifiers, inspired by mechanisms exhibited in natural and artificial immune systems. The key abstractions gleaned from these immune systems include resource competition, clonal selection, affinity maturation, and memory cell retention. A discussion of the progenitors of this work is offered. This work provides a thorough explication of a resource limited artifical immune classification algorithm, named AIRS (Artificial Immune Recognition System). Experimental results on both simulated data sets and real world machine learning benchmarks demonstrate the effectiveness of the AIRS algorithm as a classification technique.

DEDICATION

For Molly.

ACKNOWLEDGMENTS

In producing this work, the author has been truly blessed to have the guidance of two stellar advisors. Dr. Lois Boggess has provided quality instruction both in the classroom and out of it, and it is through her encouragement and invaluable ideas that this work has come to fruition. Dr. Susan Bridges provided the freedom to explore new ideas and offered the right advice at the right time. The author would also like to thank Dr. Jon Timmis of the University of Kent at Canterbury for providing feedback and advice in the development of this work. Finally, thanks go to the student-colleagues of the Intelligent Systems Laboratory, particularly Janna Hamaker, for the numerous study breaks, and Brannon Smith, for his documentation support.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

TABLE Page

LIST OF FIGURES

FIGURE Page

CHAPTER I

INTRODUCTION

With the proliferation of data collection techniques and the conversion of numerous sets of data into digital form, the ability to automatically classify patterns of data into categories of interest has become not only a useful tool but a necessary one in order to utilize these large collections of data as sources of information rather than just repositories of data. One approach to this problem is to develop techniques which automatically discover similar patterns or classes in the data. This approach is known as unsupervised learning. Often, however, data has known classifications, but the task of classifying the large volumes of data by hand is onerous to say the least. Supervised learning involves the development of a predictive model based on input data and the known classes in the data set.

The past decade has seen a growing interest in the immune system as a source of inspiration for information processing capabilities. Like human nervous systems and neo-Darwinian evolutionary theory, which have provided fruitful ideas for models of computation, mammalian immune systems offer a number of attractive features from the standpoint of computation. Chief among these features are the abilities to remember, generalize, and classify encountered substances. The subject of this thesis is the introduction of a new supervised learning paradigm inspired by mechanisms found in natural immune systems.

1.1 Background

While still relatively young, with the first international workshop on "Immunity-based Systems" being held as recently as 1996 (Dasgupta 1998c), the field of Artificial Immune Systems has begun to gather many practitioners. The field encompasses a wide range of application domains and theoretical discussions aimed at building computational models based on the natural immune systems. Some of these are quite literally trying to simulate the immune system to further understand its workings, whereas others are attempting to use the immune system as a metaphor for information processing, much as the fields of artificial neural networks and genetic algorithms have been inspired by metaphors from nature. Application domains range from pattern recognition and data mining(Hunt and Cooke (1996); Timmis, Neal and Hunt (1999); Carter (2000); Timmis (2001)) to function optimization (de Castro and Von Zuben (2001b)), from anomaly detection and information security (Forrest et al. (1994); Hofmeyr and Forrest (1999)) to robotics (Jun, Lee and Sim (1999); Watanabe, Ishiguro and Uchikawa (1998)).

Artificial Immune Systems (AIS) draw inspiration from mammalian immune systems as a model for computation in much the same way that artificial neural networks look to the human brain and genetic algorithms look to evolution in nature. Natural immune systems are incredibly complex and for the most part poorly understood. However, one aspect that is understood is that natural immune systems consist of two different types of lymphocyte cells, T cells and B cells, which circulate through our body and interact with one another to provide our immune responses. There are numerous other types of cells in the immune system that aid in our response to attack, but in general it is the T and B cells that have most often been used as models for AIS. T cells are so named because they travel to the thymus

to undergo a process of maturation as opposed to B cells, which mature in the bone marrow. Those AIS practitioners who explore the use of AIS for machine learning tend to focus on B cells. B cells seem to be the location of much of the natural immune system's memory. A somewhat controversial view of the role of B cells which is often seized upon by the AIS machine learning community is the concept of an immune network as proposed in Jerne (1974) and expounded upon in Perelson (1989). The basic argument for the network theory of immunology is that the cascading response to certain pathogens in the body suggests that B cells are interconnected and that the stimulation level of a given B cell to the pathogen affects the stimulation level of all those B cells to which it is connected. Farmer, Packard and Perelson (1986) lay out the mathematical foundation for these interactions. The works explored in Hunt and Cooke (1996), Timmis, Neal and Hunt (1999), Timmis, Neal and Hunt (2000), Timmis and Neal (2000), and Timmis (2001) are direct descendants of this early work done in applications of immune network theory.

One of the most interesting ideas to develop from this Artificial Immune Network (AIN) work by Timmis and his colleagues is the concept of a resource limited approach to Artificial Immune Systems. Central to this concept is the use of an artificial recognition ball (ARB) which represents a collection of B cells which have the same feature representation. The basic concept is fairly simple: during each exposure of the immune cells to a given antigen (or antigen population) each cell attempts to acquire resources based on its stimulation level; however, there are only a limited number of system wide resources available. If more resources have been consumed than exist in the system, then resources are removed from cells, beginning with the least stimulated first, until the number of resources held by the cells equals the number of resources allowed in the system. This competition for

resources adds an appropriate amount of pressure to the system to ensure that there is clear motivation for only the most fit cells surviving.

In Timmis and Neal (2000) and Timmis (2001), the authors present an unsupervised machine learning algorithm based on the immune network theory of immune systems and embodying the concept of resource limitation. While these earlier approaches exhibited success in identifying known patterns in data sets, their mechanisms for evaluating the quality of their results relied solely on human evaluation of a visualization of the evolved immune network. They presented no automated way of assigning class labels to the data sets or of assessing the quality of the output apart from this subjective human interaction.

The thesis work discussed here gleans some features from this early work on AINs and expands upon it. In particular, the concept of artificial recognition balls as a means of immune cell representation as well as the idea of resource limitation is maintained. However, the representation of an idiotypic immune network is no longer used. While the network approach proved useful for data visualization, particularly when batch training was undertaken, for the one-shot, incremental learning algorithm presented here, this network representation proved less useful.

This thesis also extends the early work on AINs by providing a method of assigning class labels to the artificial recognition balls (ARBs), or cells, in the system. To date, very little has been investigated in the arena of adding a supervised learning component to artificial immune systems. A notable exception is the work presented by Carter (2000). However, his approach to AIS is significantly different than the current proposed approach and adds a greater level of complexity than the current approach.

1.2 Hypothesis and Main Goals

The hypothesis of this work is that a supervised learning paradigm can be developed using components inspired by natural immune systems and previous explorations of artificial immune systems. With this hypothesis in mind, the primary goal of this work is to provide a demonstration and exploration of the classification capabilities and other properties of the proposed algorithm through a detailed presentation of the algorithm and through an evaluation of the algorithm's performance both on simulated data sets and on real world data sets that have been used throughout the machine learning literature. In order to achieve this goal it is necessary not just to show that a classification algorithm based on immune system principles can be developed but also to show that this developed system provides accurate classifications. There exist many data sets that have proven to be challenging to established classifiers with regards to creating accurate predictive models. In this regard, one of the goals of this work is to develop a classifier that exhibits similar or better accuracy rates when compared to other, more established classifiers. The developed algorithm, Artificial Immune Recognition System (AIRS), exhibits several highly desirable characteristics for machine learning. These characteristics include one-shot learning, continuous learning, data reduction capabilities, and highly accurate predictions, among others. This thesis explores some of these capabilities while pointing to many areas that should prove fruitful for future investigation.

1.3 Organization

The remainder of this thesis is organized as follows:

- Chapter 2 presents a discussion of the key immunological components (both natural and artificial) that are essential to the development of the AIRS

algorithm. This chapter will provide appropriate pointers to the literature for these ideas.

- Chapter 3 provides an overview of the AIRS algorithm and discusses important factors in the algorithm's learning mechanisms.
- Chapter 4 investigates some of the basic learning capabilities of the AIRS algorithm by demonstrating its behavior on simulated data sets.
- Chapter 5 presents initial results of the AIRS algorithm on several standard benchmark data sets in the machine learning literature.
- Finally, Chapter 6 offers conclusions and points to areas of future investigation.

CHAPTER II

IMMUNE SYSTEMS: NATURAL AND ARTIFICIAL

This chapter discusses the fundamental components essential to the development of the AIRS algorithm. We examine some basic principles of natural immune systems and explore how these principles have been used in data analysis/machine learning artificial immune systems. The emphasis of this chapter is a presentation of the precursors, both biological and artificial, to the AIRS classification paradigm. As such, pointers to relevant references in the literature are provided where appropriate. It should be acknowledged here that the primary inspiration for the AIRS classifier is the adaptation of natural immune systems to artificial immune systems done by other researchers. As such, much of the discussion of the natural immune system comes through the filtering of other AIS practitioners rather than from primary biological sources.

2.1 Antibodies, Antigens, and B Cells

The most fundamental immunological principle embodied in the AIRS algorithm is the representation of the B cell. Natural immune systems contain a large population of B cells which provide the mechanism for adaptive immunological responses. This is accomplished through a process of recognition, stimulation, and clonal proliferation. On the surface of each B cell is a population of antibodies which provide the primary binding sites to foreign cells. The degree to which a B cell is said to match to a given invading cell depends on the degree of binding

between the antibodies' paratopes and epitopes of the invading cell's antigens. In a sense, then, it is this antibody-antigen interaction which provides the fundamental recognition capability of the B cell. However, the B cell, itself, plays an important role in the immune response to an antigen. Based on the degree of binding (affinity) between the antibodies and antigens, a B cell becomes stimulated. Stimulated B cells then go through a cloning and mutation process, rapidly producing offspring. These offspring are then used to attack and destroy the foreign presence. So, there is a close relationship between the recognition and response mechanisms present in the immune system.

In the world of artificial immune systems for pattern recognition, the B cell is the fundamental object for machine learning. In particular, the primary recognition mechanism is often modeled off of the antibody-antigen binding. In Timmis, Neal and Hunt (1999), Timmis, Neal and Hunt (2000), Timmis and Neal (2000), Timmis (2001), Knight and Timmis (2001) and de Castro and Von Zuben (2001a), the Euclidean distance between a simulated B Cell's feature vector and a training data item's feature vector is used to determine the degree of binding between the antibodies of the cell and the antigens of the invader. In Hunt and Cooke (1996), the antibodies and antigens are represented as binary strings, and the affinity between two cells is related to the number of matching bits. In de Castro and Von Zuben (2001b), Hamming distance is used as the affinity measurement. Finally, in Hunt et al. (1998), the authors employ multiple methods to determine the affinity between two cells.

Regardless of the binding mechanisms, all of these approaches are attempts to provide a computationally viable mechanism for simulating the interactions between the data representation that currently exists (akin to B cells or antibodies) in the

system and the incoming training data items (akin to antigens). And, with the notable exception of de Castro's work which focuses only on antibodies, all of these artificial immune systems use the concept of B cell stimulation to provide additional adaptive learning mechanisms.

2.2 Artificial Recognition Balls and Resource Competition

At this point in the discussion of the key mechanisms and components in the AIRS algorithm we deviate slightly from mechanisms found in nature. Given the previous discussion of B cells and antibodies, a B cell could be simplified in representation to having a single antibody. This antibody could be considered a string in the feature search space of the problem domain, where each element in the string is a different feature. Therefore, a B cell with its antibody represents one particular point in the overall feature space. With this in mind, the need to actually represent this exact same point in space more than once becomes rather redundant. It is this line of reasoning, coupled with some observations of natural immune systems, that leads to the concept of an artificial recognition ball (ARB) first proposed in Timmis and Neal (2000) and further developed in Timmis (2001) and Knight and Timmis (2001).

Essentially, an ARB can be thought of as a representation of numerous B cells, all of which have the same antibody. In Timmis and Neal (2000), the authors refer to the number of B cells a given ARB represents as resources. Continuing with this approach, an artificial immune system consists of numerous ARBs which provide the primary memory mechanism of the system. The number of resources available is limited to a finite number. Learning takes place through a competition for resources. As with the discussion of B cells, ARBs are stimulated through a response to an

invading antigen. After exposure to a given antigen (or antigen population) each ARB attempts to consume resources based on its stimulation level. However, since the number of resources are finite, only the most stimulated ARBs will actually consume resources. The remaining ARBs (*i.e.,* those without any resources) are removed from the system. This competition for resources applies a certain amount of evolutionary pressure to ensure that only the strongest ARBs (*i.e.,* those most adept at recognizing antigens) remain in the system.

2.3 Memory Cells, Mutation, and Clonal Selection

As we have discussed, when a population of B cells encounters an unknown antigen certain highly stimulated cells respond by producing clones and mutated offspring to combat the invader. This is considered the immune system's primary response. Through this primary response, certain cells which have proven particularly adept at counteracting the given antigen are allowed to remain in the system for possible future encounters with this or structurally similar antigens. These long-lasting cells, hereafter referred to as memory cells (de Castro and Von Zuben 2001b), provide the basis for the immune system's secondary response. When the immune system encounters previously seen antigens or antigens that are similar to those previously seen, the memory cells are rapidly stimulated and produce clones and mutated offspring at a much greater rate. These memory cells, then, are the centers for the immune system's ability to recognize and appropriately respond to known threats. Importantly, memory cells exhibit the ability to not only recognize the antigen they originally responded to but to generalize to structurally similar antigens.

At the heart of the AIRS algorithm is the development of objects modeled after the behavior of these memory cells. In de Castro and Von Zuben (2001b) and de Castro and Von Zuben (2001a), the authors discuss the role of the clonal selection principle and affinity maturation in the development of memory cells for AIS. Basically, these concepts are central to any evolutionary algorithm. In the terms of AIS, clonal selection means that those cells which are most stimulated or exhibit the greatest affinity for an antigen are selected to produce offspring. Since we are employing the ARB concept, producing offspring is in actuality producing only mutated offspring. In the AIRS algorithm, mutation is in the form of random feature mutation. That is, if a particular feature of an ARB's antibody is selected for mutation, it will randomly mutate to any value within a given range. Affinity maturation, in the context of AIRS, is the application of genetic pressure to the surviving cells to produce offspring that are more stimulated by the antigen. This genetic pressure is based on the competition for resources discussed in section 2.2. That is, only those ARBs which have successfully acquired resources are allowed to clone. Ultimately, through the process of clonal selection, feature mutation, and affinity maturation, a single ARB that is most stimulated by a particular antigen is chosen to remain in the system as a memory cell. Once exposure to the antigen population is complete, the evolved memory cell population forms the basis for the classification of new antigens.

2.4 Immune Networks and Cell-Cell Interactions

The final immune system principle that has played a formative role on the development of the AIRS algorithm is that of immune network theory. As mentioned in chapter 1, immune network theory, as presented in Jerne (1974) and further

developed in Perelson (1989), hypothesizes that immune responses are based not only on the interaction of B cells and pathogens but also on the interactions of B cells with other B cells. B cells provide both a stimulation and suppression effect on one another and it is partially through this interaction that the memory is retained in the immune system. This idea of an immune network has most prominently been translated into the AIS world through the work of (Timmis 2001). Timmis's extrapolation of immune network theory has resulted in a successful method of unsupervised learning, data analysis, and data visualization. One of the key points from this work is the idea of inter-cell affinity and link formation. In Timmis's work, a network of ARBs is formed based on the similarity or affinity exhibited among the ARBs. That is, if two ARBs show sufficient affinity for one another, then these two ARBs are linked together. These links provide a mechanism of reinforcing connected ARBs as they respond to incoming antigens. Clusters of closely related ARBs form and embody clusters in the data set.

While this network idea is not explicitly used in the AIRS algorithm, the concept of inter-cell affinity plays an important role. One of the principle features of the AIRS algorithm is the ability to perform data reduction. The trained AIRS classifier typically has far fewer memory cells than original training data items. This is achieved through incremental memory cell replacement, and the key criterion for memory cell replacement is based on inter-cell affinity between a newly evolved memory cell and an established memory cell. Chapter 3 provides more details concerning this mechanism.

2.5 Classification

Before leaving this chapter on the principle ideas that have provided the foundation for AIRS, a brief mention of AIS for classification or supervised learning should be made. To date, the author knows of only one attempt to use immune system principles to develop a supervised learning system. Carter (2000) presents a supervised classification method modeled on immune system principles. Unlike other AIS discussed in this chapter, Carter models both T cells and B cells, and the interactions between the two play an important role in the development of his classifier.

Rather than following this road, AIRS builds upon the methods that have been proven successful in the unsupervised learning realm as discussed in this chapter. Adding feedback to the evolving immune system based on an antigen's known classification proved initially more difficult than imagined. However, AIRS now incorporates this knowledge in several ways. First, the stimulation of an ARB is based not only on its affinity to an antigen but also on its class when compared to the class of an antigen. High affinity ARBs of the same class as the antigen are rewarded as are low affinity ARBs of a different class as the antigen. Second, allocation of resources to the ARBs also takes into account the ARBs' classifications when compared to the class of the antigen. Finally, memory cell hyper-mutation and replacement is based primarily on classification and secondarily on affinity.

CHAPTER III

OVERVIEW OF THE AIRS ALGORITHM

This chapter presents an overview of the AIRS algorithm. Section 3.1 defines some of the key terms and concepts important to the understanding of the algorithm. Section 3.2 provides a somewhat formal tour of the training routine of the algorithm. Section 3.3 presents a more conceptual overview of the algorithm.

3.1 Definitions

This section presents definitions of the key terms and concepts used throughout the rest of this thesis, particularly as they apply to the AIRS algorithm.

- *affinity*: a measure of "closeness" or similarity between two *antibodies* or *antigens*. In the current implementation, this value is guaranteed to be between 0 and 1 inclusively and is calculated simply as the Euclidean distance of the two objects' *feature vectors*. Thus, small affinity values indicate strong affinity.
- *affinity threshold (AT)*: the average *affinity* value among all of the *antigens* in the *training set* or among a selected subset of these training antigens.
- *affinity threshold scalar (ATS)*: a value between 0 and 1 that, when multiplied by the *affinity threshold*, provides a cut-off value for memory cell replacement in the *AIRS* training routine.
- *antibody*: a *feature vector* coupled with its associated *output* or *class*; the feature vector-output combination is referred to as an antibody when it is part of an *ARB* or *memory cell*.
- *antigen*: this is the same in representation as an *antibody*; however, the feature vector-class combination is referred to as an antigen when it is being presented to the *ARBS* for stimulation and/or response.
- *Artificial Immune Recognition System (AIRS)*: a classification algorithm inspired by natural immune systems.

- *Artificial Recognition Ball (ARB)*: also known as a *B-Cell*. It consists of an *antibody*, a count of the number of *resources* held by the cell, and the current *stimulation value* of the cell.

- *B Cell*: in this thesis, more commonly referred to as an *Artificial Recognition Ball*.

- *candidate Memory Cell*: the *antibody* of an *ARB*, of the same *class* as the training *antigen*, which was the most stimulated after exposure to the given antigen.

- *class*: the category of a given *feature vector*. This is also referred to as the *output* of a cell.

- *clonal rate*: an integer value used to determine the number of mutated clones a given *ARB* is allowed to attempt to produce. In the current implementation, a selected ARB is allowed to produce up to (*clonal rate* * *stimulation value*) mutated clones after responding to a given *antigen*. This product is also used in assigning *resources* to an ARB. Therefore, the clonal rate serves a dual-role as resource allocation factor and clonal mutation factor for the cell population.

- *established Memory Cell*: the *antibody* of an *ARB* which has survived competition for resources and was the most stimulated to a given training *antigen* and has been added to the evolving set of *memory cells*.

- *feature vector*: one instance of data represented as a sequence of values. Each position in the sequence represents a different feature associated with the data, and each feature has its own range of legitimate values.

- *hyper-mutation rate*: an integer value used to determine the number of mutated clones a given *memory cell* is allowed to inject into the cell population. In the current implementation, the selected memory cell injects at least (*hyper-mutation rate* * *clonal rate* * *stimulation value*) mutated clones into the cell population at the time of *antigen* introduction.

- *k nearest neighbor (KNN)*: a classification scheme in which the response of the classifier to a previously unseen item is determined by a majority vote among the k closest data points. For the *AIRS* algorithm, the k closest data points are in actuality the k most stimulated *memory cells* to a given test *antigen*.

- *k value*: the parameter which indicates how many *memory cells* should be used to determine the classification of a given test item. (see *k nearest neighbor* for more details)

- *memory cell (mc)*: the *antibody* of an *ARB* which was the most stimulated by a given training *antigen* at the end of exposure to that antigen. It is used for hyper-mutation in response to incoming training antigens (see *hyper-mutation rate*). An mc can be replaced, however. This occurs only when a *candidate mc* is more stimulated to a given training antigen than the most stimulated *established mc* and the affinity between the established mc and the candidate mc is less than the product of the *Affinity Threshold* and the *Affinity Threshold Scalar*.

- *mutation rate*: a parameter between 0 and 1 that indicates the probability that any given feature (or the output) of an *ARB* will be mutated.

- *output*: the classification category associated with a cell. Same as the *class* of the feature vector corresponding to the cell.

- *resources*: a parameter which limits the number of *ARB*s allowed in the system. Each ARB is allocated a number of resources based on its *stimulation value* and the *clonal rate*. The total number of system wide resources is set to a certain limit. If more resources are consumed than are allowed to exist in the system, then resources are removed from the least stimulated ARBs until the number of resources in the system returns to the number allowed. If all of a given ARB's resources are removed, then that ARB is removed from the cell population.

- *seed cell*: an *antibody*, drawn from the *training set*, used to initialize *Memory Cell* and *ARB* populations at the beginning of training.

- *stimulation function*: a function used to measure the response of an *ARB* to an *antigen* or to another ARB. In the current formulation of the *AIRS* classifier, this function should return a value between 0 and 1 inclusively. For the implementation of *AIRS* presented in this study, the stimulation function is inversely proportional to the Euclidean distance between the *feature vector*s of the *ARB* and the *antigen*.

- *stimulation value*: the value returned by the *stimulation function*.

- *stimulation threshold*: a parameter between 0 and 1 used as a stopping criterion for the training on a specific *antigen*. For the current implementation, only when the average *stimulation value* of the *ARB*s of each class is above the stimulation threshold does training in reaction to the particular antigen stop.

- *test set*: the collection of *antigens* used to evaluate the classification performance of the trained *AIRS* classifier.

- *training set*: the collection of *antigens* used to train the *AIRS* classifier.

3.2 Tour of the Algorithm

This section presents a tour of the AIRS algorithm. In particular, this section presents an overview of the primary routines, methods, and equations used in the training and building of an immune-system based classifier. This somewhat formal view of the algorithm is offered as a companion to the more conceptual discussion and explanation of these mechanisms presented in Section 3.3. There are four primary stages involved in the AIRS algorithm. The first stage is data normalization and initialization. The second stage is memory cell identification and ARB generation. The third stage is competition for resources in the development of a candidate memory cell. The final stage of the training algorithm is the potential introduction of the candidate memory cell into the set of established memory cells.

For this discussion, let us establish the following notational conventions:

- Let MC represent the set of memory cells and mc represent an individual member of this set.
- Let $ag.c$ represent the class of a given antigen, ag, where $ag.c \in C = \{1, 2, \ldots nc\}$ and nc is the number of classes in the data set.
- Let $mc.c$ represent the class of a given memory cell, mc, where $mc.c \in C = \{1, 2, \ldots nc\}$.
- Define $MC_c \subseteq MC = \{MC_1 \cup MC_2 \cup \ldots MC_{nc}\}$ and $mc \in MC_c$ iff $mc.c \equiv c$.
- Let $ag.f$ and $mc.f$ represent the feature vector of a given antigen and memory cell, ag and mc, respectively. Let $ag.f_i$ represent the value of the ith feature in $ag.f$ and $mc.f_i$ the value of the ith value of $mc.f$.
- Let AB represent the set of ARBs, or the population of existing cells, and MU represent a set of mutated clones of ARBs. Furthermore, let ab represent a single ARB where $ab \in AB$.
- Let $ab.c$ represent the class of a given ARB, ab, where $ab.c \in C = \{1, 2, \ldots nc\}$.
- Define $AB_c \subseteq AB = \{AB_1 \cup AB_2 \cup \ldots AB_{nc}\}$, and $ab \in AB_c$ iff $ab.c \equiv c$.
- Let $ab.stim$ represent the stimulation level of the ARB ab.

- Let $ab.resources$ represent the number of resources held by the ARB ab.
- Let $TotalNumResources$ represent the total number of system wide resources allowed.

3.2.1 Initialization

The first stage of the algorithm, initialization, can primarily be thought of as a data pre-processing stage combined with a parameter discovery stage. During initialization, first all items in the data set are normalized such that the Euclidean distance between the feature vectors of any two items is in the range of [0,1]. This can be performed through a variety of methods and could also be performed as a true pre-processing stage before the algorithm begins. It is important to note that, while for the current investigation Euclidean distance is the primary metric of both affinity and stimulation, other functions could be employed as well. What is important about this normalization is only that the range of possible reactions from cell-to-cell interaction remains within the range of [0,1]. After normalization, the affinity threshold is calculated. For established training sets (*i.e.,* those not being generated on the fly), the affinity threshold is the average affinity value over all training data items' feature vectors. For training data supplied from a user-defined data generation function, a finite number (arbitrarily chosen at fifty for the current work) of data items are generated to be used in this calculation. The affinity threshold is calculated as described in equation (3.1) below:

$$\text{affinity threshold} = \frac{\sum_{i=1}^{n} \sum_{j=i+1}^{n} \text{affinity}(ag_i, ag_j)}{\frac{n(n-1)}{2}} \tag{3.1}$$

where n is the number of training data items (antigens) in question, ag_i and ag_j are the ith and jth training antigen (or generated data item) in the antigen training

vector (or generated initialization vector), and affinity(x,y) returns the Euclidean distance between the two antigens' feature vectors.

The final step in initialization is the seeding of the memory cells and initial ARB population. This is done by randomly choosing 0 or more antigens from the set of training vector to be added to the set of memory cells and to the set of ARBs, or, in the case of the use of a data generation function, generating 0 or more antigens to be added to the set of memory cells and to the set of ARBs.

3.2.2 Memory Cell Identification and ARB Generation

Once initialization is complete, training proceeds as a one-shot incremental algorithm. The first step of this stage of the algorithm is memory cell identification and ARB generation. Given a specific training antigen, ag, find the memory cell, mc_{match}, that has the following property: $mc_{match} = \arg max_{mc \in MC_{ag.c}} stimulation(ag, mc)$, where $stimulation(x, y)$ is defined as in equation (3.2) below:

$$stimulation(x, y) = 1 - \text{Euclidean\_distance}(x.f, y.f) \tag{3.2}$$

If $MC_{ag.c} \equiv \oslash$, then $mc_{match} \leftarrow ag$ and $MC_{ag.c} \leftarrow MC_{ag.c} \cup ag$. That is, if the set of memory cells of the same classification as the antigen is empty, then add the antigen to the set of memory cells and denote this newly added memory cell as the match memory cell, mc_{match}. It should be noted here that while the stimulation function for the current work relies solely on Euclidean distance, this need not necessarily be the case. One possible future investigation for this algorithm would be the use of a variety of different stimulation functions.

Once mc_{match} has been identified, this memory cell is used to generate new ARBs to be placed into the population of (possibly) pre-existing ARBs (*i.e.,* those ARBs left in the system from exposure to previous antigens). This is done through the method shown in Figure 3.1, where the function $makeARB(x)$ returns an ARB with x as the antibody of this ARB and where $mutate(x, b)$ is defined in Figure 3.2.

$MU \leftarrow \oslash$
$MU \leftarrow MU \cup makeARB(mc_{match})$
$stim \leftarrow stimulation(ag, mc_{match})$
$NumClones \leftarrow hyper\_clonal\_rate * clonal\_rate * stim$
while ($\mid MU \mid < NumClones$)
do
  $mut \leftarrow false$
  $mc_{clone} \leftarrow mc_{match}$
  $mc_{clone} \leftarrow mutate(mc_{clone}, mut)$
  if($mut \equiv true$)
    $MU \leftarrow MU \cup makeARB(mc_{clone})$
done
$AB \leftarrow AB \cup MU$

Figure 3.1: Hyper-Mutation for ARB Generation

In Figure 3.2, the function $drandom()$ returns a random value in the range [0,1] and ($lrandom()$ mod nc) returns a random value in the range {0,nc}.

3.2.3 Competition for Resources and Development of a Candidate Memory Cell

At this point a set of ARBs (AB) exists which includes mc_{match}, mutations from mc_{match}, and (possibly) remnant ARBs from responses to previously encountered antigens. Recall that the AIRS algorithm is a one-shot algorithm, so while the discussion has been divided into separate stages, only one antigen goes through this entire process at time (with the obvious exception being the initialization stage

```
mutate(x, b)
{
  foreach(x.f_i in x.f)
  do
    change ← drandom()
    change_to ← drandom()
    if(change < mutation_rate)
      x.f_i ← change_to * normalization_value
      b ← true
  done
  if(b ≡ true)
    change ← drandom()
    change_to ← (lrandom() mod nc)
    if(change < mutation_rate)
      x.c ← change_to
  return x
}
```

Figure 3.2: Mutation Routine

which takes place over the entire data set before training begins). The goal of the next portion of the algorithm is to develop a candidate memory cell which is most successful in correctly classifying a given antigen, ag. This is done primarily through three mechanisms. The first mechanism is through the competition for system wide resources. Following the methods first outlined by Timmis and Neal (2000) and more fully realized by Knight and Timmis (2001), resources are allocated to a given ARB based on its normalized stimulation value, which is used as an indication of its fitness as a recognizer of ag. The second mechanism is through the use of mutation for diversification and shape-space exploration. The third mechanism is the use of an average stimulation threshold as a criterion for determining when to stop training on ag.

Similar to principles involved in genetic algorithms, the AIRS algorithm employs a concept of fitness for survival of individuals within the ARB population. Survival of a given ARB is determined in a two-fold, interrelated manner. First, each ARB in the population AB is presented with the antigen ag to determine the ARB's stimulation level. This stimulation is then normalized across the ARB population based on both the raw stimulation level and the class of the given ARB ($ab.c$). Based on this normalized stimulation value, each $ab \in AB$ is allocated a finite number of resources. If this allocation of resources would result in more resources being allocated across the population than allowed, then resources are removed from the weakest (least stimulated) ARBs until the total number of resources in the system returns to the number of resources allowed. Those ARBs which have zero resources are removed from the ARB population. This process is formalized in Figure 3.3.

While Section 3.3 will discuss this in more detail, it is important to note here two key aspects of this resource allocation routine. First, the stimulation value of an ARB is not only determined by the stimulation function in equation 3.2 but is also based on the class of the ARB. The stimulation calculation method outlined in Figure 3.3 provides reinforcement both for those ARBs of the same class as ag that are highly stimulated by ag and for those ARBs that are of a different class as ag that do not exhibit a strong positive reaction to ag. Second, the distribution of resources is also based on the class of the ARB. This is done to provide additional reinforcement for those ARBs of the same class as ag without losing the potentially positive qualities of the remaining ARBs for reaction to future antigens.

At this point in the algorithm, the ARB population AB consists of only those ARBs that were most stimulated by the given antigen, ag, or more specifically, AB now consists of those ARBs that were able to successfully compete for resources. The

$minStim \leftarrow 2.0$
$maxStim \leftarrow 0.0$
foreach($ab \in AB$)
do
  $stim \leftarrow stimulation(ag, ab)$
  if ($stim < minStim$)
    $minStim \leftarrow stim$
  if ($stim > maxStim$)
    $maxStim \leftarrow stim$
  $ab.stim \leftarrow stim$
done
foreach($ab \in AB$)
do
  if($ab.c \equiv ag.c$)
    $ab.stim \leftarrow \frac{ab.stim - minStim}{maxStim - minStim}$
  else
    $ab.stim \leftarrow 1 - \frac{ab.stim - minStim}{maxStim - minStim}$
  $ab.resources \leftarrow ab.stim * clonal\_rate$
done
$i \leftarrow 1$
while($i \leq nc$)
do
  $resAlloc \leftarrow \sum_{j=1}^{|AB_i|} ab_j.resources, ab_j \in AB_i$
  if($i \equiv ag.c$)
    $NumResAllowed \leftarrow \frac{TotalNumResources}{2}$
  else
    $NumResAllowed \leftarrow \frac{TotalNumResources}{2*(nc-1)}$
  while($resAlloc > NumResAllowed$)
  do
    $NumResRemove \leftarrow resAlloc - NumResAllowed$
    $ab_{remove} \leftarrow argmin_{ab \in AB_i}(ab.stim)$
    if($ab_{remove}.resources \leq NumResRemove$)
      $AB_i \leftarrow AB_i - ab_{remove}$
      $resAlloc \leftarrow resAlloc - ab_{remove}.resources$
    else
      $ab_{remove}.resources \leftarrow ab_{remove}.resources - NumResRemove$
      $resAlloc \leftarrow resAlloc - NumResRemove$
  done
  $i \leftarrow i + 1$
done

Figure 3.3: Stimulation, Resource Allocation, and ARB Removal

algorithm continues first by determining if the ARBs in AB were stimulated enough by ag to stop training on this item. This is done by defining a vector $\vec{s}$ that is nc in length to contain the average stimulation value for each class subset of AB. That is:

$$s_i \leftarrow \frac{\sum_{j=1}^{|AB_i|} ab_j.stim}{\mid AB_i \mid}, ab_j \in AB_i$$

The stopping criterion is reached iff $s_i \geq stimulation\_threshold$ for all elements in $\vec{s} = \{s_1, s_2, \ldots s_{nc}\}$.

Regardless of whether the stopping criterion is met or not, the algorithm proceeds by allowing each ARB in AB the opportunity to produce mutated offspring. While this adding of mutated offspring is similar to the method outlined in Figure 3.1, there are a few differences. This modified mutation generation routine is presented in Figure 3.4.

After allowing each surviving ARB the opportunity to produce mutated offspring, the stopping criterion is examined. If the stopping criterion is met, then training on this one antigen stops. If the stopping criterion has not been met, then this entire process, beginning with the method outlined in Figure 3.3, is repeated until the stopping criterion is met. The only exception to this repetition is that on every pass through this portion of the algorithm, except the first pass already discussed, if the stopping criterion is met after the stimulation and resource allocation phase, then the production of mutated offspring is not performed. Once the stopping criterion has been met, then the candidate memory cell is chosen. The candidate memory cell, $mc_{candidate}$, is the feature vector and class of the ARB that existed in the system before mutation that was the most stimulated ARB of the same class as the training antigen ag.

```
MU ← ∅
foreach(ab ∈ AB)
do
   rd ← drandom()
   if(ab.stim > rd)
      NumClones ← ab.stim * clonal_rate
      i ← 1
      while(i ≤ NumClones)
      do
         mut ← false
         ab_clone ← ab
         ab_clone ← mutate(ab_clone, mut)
         if(mut ≡ true)
            MU ← MU ∪ ab_clone
         i ← i + 1
      done
done
AB ← AB ∪ MU
```

Figure 3.4: Mutation of Surviving ARB

3.2.4 Memory Cell Introduction

The final stage in the training routine is the potential introduction of the just-developed candidate memory cell, $mc_{candidate}$, into the set of existing memory cells MC. It is during this stage that the affinity threshold calculated during initialization becomes critical as it dictates whether the $mc_{candidate}$ replaces mc_{match} that was previously identified. The candidate memory cell is added to the set of memory cells only if it is more stimulated by the training antigen, ag, than mc_{match}, where stimulation is defined as in equation (3.2). If this test is passed, then if the affinity between $mc_{candidate}$ and mc_{match} is less than the product of the affinity threshold and the affinity threshold scalar, then $mc_{candidate}$ replaces mc_{match} in the set of memory cells. This memory cell introduction method is presented in figure 3.5.

$CandStim \leftarrow stimulation(ag, mc_{candidate})$
$MatchStim \leftarrow stimulation(ag, mc_{match})$
$CellAff \leftarrow affinity(mc_{candidate}, mc_{match})$
if($CandStim > MatchStim$)
  if($CellAff < AT * ATS$)
    $MC \leftarrow MC - mc_{match}$
  $MC \leftarrow MC \cup mc_{candidate}$

Figure 3.5: Memory Cell Introduction

Once the candidate memory cell has been evaluated for addition into the set of established memory cells, training on this one antigen is complete. The next antigen in the training set is then selected (or the next antigen is generated using a data generation function), and the training process proceeds with memory cell identification and ARB generation. This process continues until all antigens have been presented to the system.

3.2.5 Classification

After training has completed, the evolved memory cells are available for use for classification. The classification is performed in a k-nearest neighbor approach. Each memory cell is presented with a data item for stimulation. The system's classification of a data item is determined by using a majority vote of the outputs of the k most stimulated memory cells.

3.3 Discussion of the AIRS Algorithm

While Section 3.2 presents a fairly formal overview of the AIRS algorithm, this section provides a conceptual discussion of some of the key elements of this classification system. The goal of the algorithm is the development of a set of memory cells that can be used to classify data. These artificial memory cells embody several characteristics seen in natural immune systems. Primarily, the memory cells are based on memory B Cells that have undergone a maturation process in the body. In mammalian immune systems, these memory B Cells are easily stimulated by invading antigens and undergo a process of hyper-somatic mutation as a response to recognized invading pathogens. The embodiment of this concept is seen in the function of memory cells in the AIRS algorithm. The artificial memory cells can also be said to take on the role of T Cells and Antigen Presenting Cells to some degree. In natural immune systems T Cells tend to be associated with a specific population of B Cells. When a T Cell recognizes an antigen, it then presents this antigen to the B Cells associated with it for further recognition. In the AIRS algorithm, this can be seen in the initial stages of identifying the matching memory cell which in turn develops a population of ARBs closely related to the matching memory cell.

The heart of the AIRS algorithm is the process of evolving memory cells from a population of ARBs. This evolutionary process has several key concepts which warrant mention here. The primary mechanism for providing evolutionary pressure to the population of ARBs in the development of memory cells is the competition for system wide resources. This concept, inspired by Timmis and Neal (2000) and Knight and Timmis (2001), is the means by which cell survival is determined and reinforcement for quality classification is provided. Like genetic algorithms, the goal of resource competition is the development of the fittest individuals. In the AIRS algorithm, fitness is initially determined by stimulation response of an individual ARB to an antigen. In the unsupervised learning work of Jon Timmis and his colleagues, the stimulation value of a given cell was sufficient for resource allocation. However, in the current work, it is necessary to not only examine the stimulation response of a cell to an antigen but to also take into account the cell's class when compared to the antigen's class. For this reason, in the AIRS algorithm, cells with high stimulation responses that are of the same class as the antigen and cells with low stimulation responses that are not of the same class as the antigen are rewarded most heavily. The reward comes in the form of being allocated more system wide resources. On the other hand, those cells with low stimulation values but the same class or high stimulation values but a different class as the antigen are seen as not just poor classifier cells, but potentially detrimental classifier cells and they are thus rewarded less. Since those ARBs with the least ability to acquire resources are purged from the system, there is great pressure to evolve toward a place in the search space that will provide the most reward. However, as we wish to maintain the generalizing capabilities of the system, the exact "amount" of error is not provided as feedback to an individual cell, rather a cell is merely provided reinforcement for higher quality

representations. This reinforcement toward correct classification is further enhanced through the use of non-uniform resource distribution. In the current implementation, those ARBs of the same class as the antigen are provided with more resources to compete over than those of different classes. This is primarily done to allow greater potential diversity when exploring the space around the training antigen. One future extension of this concept would be to divide the resources among the ARBs in a manner reflecting the class proportions in the data set.

ARBs which survive the competition for resources are further rewarded by being given the opportunity to produce mutated offspring. Again, as is the case with genetic algorithms, this competition for survival can also be seen in a truly evolutionary sense. That is, while the fittest individuals in a given round of antigen exposure might not survive to actually become a memory cell, their offspring might. Thus it is survival of a "species" of cell that this algorithm promotes. This is accomplished through the use of feature mutation in the training routine. The introduction of mutated offspring into the ARB population provides for a more thorough exploration of the search space. This exploration is further enhanced by the use of a stimulation threshold that must be met before the ARB population can be said to have "learned" to recognize a given antigen. This increases the evolutionary pressure to develop cells across the population which exhibit qualities desirable for memory cell inclusion.

After an ARB has successfully competed for resources among the general ARB population, if it was the most stimulated in response to the training antigen and was of the same class as the antigen, then it has the opportunity to be added to the pool of memory cells. When an antigen is first introduced to the system, the memory cell which is most stimulated by and of the same class as the antigen is allowed to inject mutated offspring into the ARB general population. However, at

the end of the evolutionary process of this ARB population, this original memory cell can potentially be replaced by the evolved memory cell. This occurs only when the evolved memory cell is closer in the search space to the training antigen than the originally located memory cell and when these two memory cells are "close enough" (as determined by a user-parameter) to each other as well. From a machine learning point of view, this process is what provides for the data reduction capabilities of the AIRS algorithm. By allowing better classifying memory cells that occupy near-by locations in the shape space to replace existing memory cells, the AIRS algorithm reduces the number of cells needed to represent the problem domain. The algorithm also generalizes since the evolved memory cells in the system are not necessarily identical to any training instances.

CHAPTER IV

INVESTIGATION OF AIRS: SIMULATED DATA SETS

This chapter presents an investigation of the behavior of the AIRS algorithm on two simulated data sets. There are several goals to this portion of the investigation. First and foremost is a demonstration of the ability of the AIRS algorithm to actually discern known classes based on exposure to training data. Second is an examination of the memory cells developed in relation to the training data. Finally, we look at the classification accuracy of the evolved classifier on previously unseen data items.

4.1 Data Sets

To demonstrate and investigate the behavior and principles of the current implementation of the AIRS algorithm, two somewhat basic simulated data sets are employed. Both data sets represent two-dimensional two-class problems consisting of points in a 10x10 space. The first data space is a simple linearly separable data space where class membership is defined by:

$$\text{class}(f) = \begin{cases} 1 & \text{iff} \quad f_0 < f_1 \\ 0 & \text{otherwise} \end{cases}$$

And the second data space is slightly more complex with class membership being defined as:

$$\text{class}(f) = \begin{cases} 1 \quad \text{if} & 1 \leq f_0 \leq 6, 3 \leq f_1 \leq 8 \\ 1 \quad \text{if} & 7 \leq f_0, 5 \leq f_1 \\ 1 \quad \text{if} & f_0 \leq 5, f_1 \leq 2 \\ 0 \quad \text{otherwise} & \end{cases}$$

For both of these membership functions, f is a feature vector representing a point in the 10x10 space with f_0 indicating the first feature in the vector (essentially the x coordinate) and f_1 representing the second feature in the vector (essentially the y coordinate). Figures 4.1 and 4.2 provide a graphical view of these class membership functions.

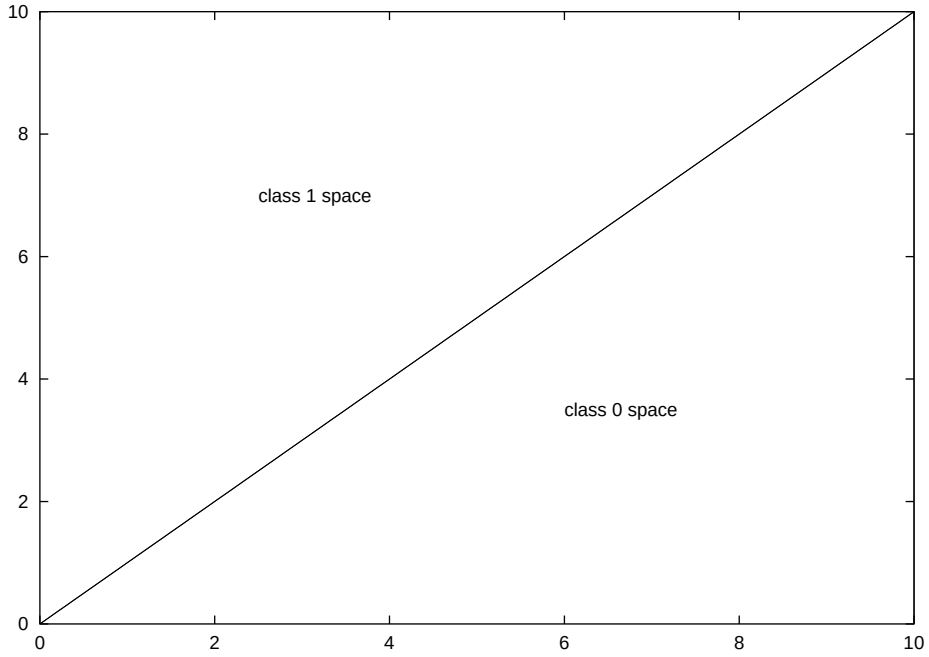

Figure 4.1: Linearly Separable Data Space

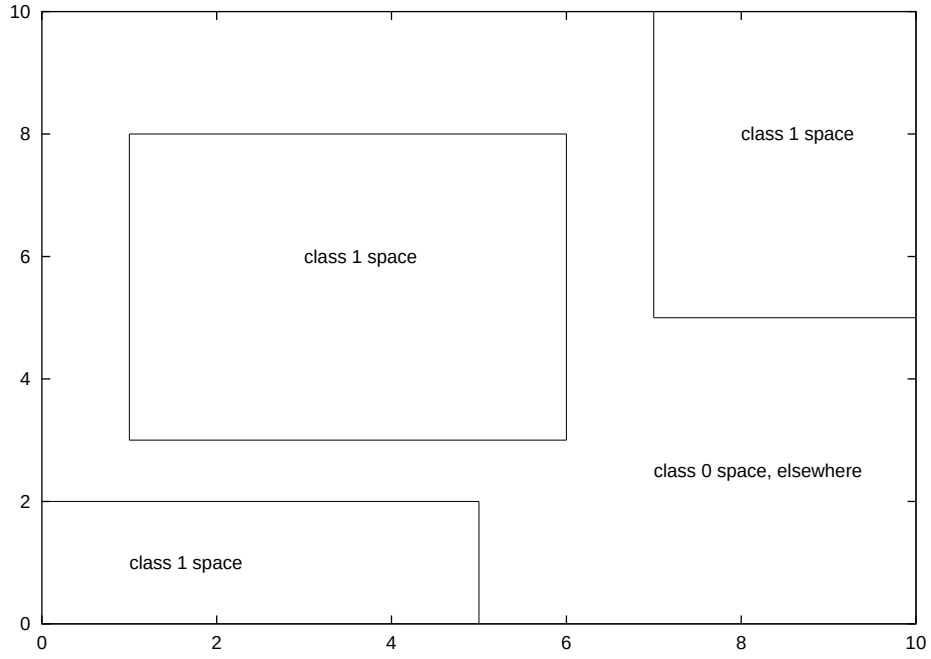

Figure 4.2: Non-Linearly Separable Data Space

The training and test data sets for these experiments are generated by randomly choosing points in this 10x10 space. For the training set, the data generation is done during run time through the use of a data generation function called from within the training routine. For the test set, fifty data points were randomly generated for use in assessing the algorithm's ability to classify previously unseen data.

4.2 Experimental Design

As previously mentioned, the primary purpose of this set of experiments was a demonstration and investigation of some of the basic principles of the AIRS algorithm. To this effect, these experiments were kept relatively simple. That is, only two-dimensional, two-class data sets were used, only one run of the algorithm on each data set is examined, and only one set of parameter settings is explored. The general purpose of the experimental design was two-fold:

1. explore, in an easily visualized manner, the memory cells produced through the exposure of the algorithm to training items, and
2. investigate the changes in classification capability of the algorithm as it was incrementally exposed to these training items.

To accomplish this, each training data item was randomly generated at run time and exposed to the evolving system. After each exposure, the performance of the system on the previously generated test set was evaluated. It should be stressed here that this performance assessment in no way influenced the behavior of the system or the training on subsequent data items. For both data sets, the following learning parameters were used:

- The number of seed cells was 0.
- The clonal rate was set to 10.
- The mutation rate was set to 0.1.
- The hyper mutation rate was set to 2.0.
- The number of resources allowed in the system was 500.
- The stimulation threshold was set to 0.8.
- The affinity threshold scalar (ATS) was set to 0.1.
- The k values used for the testing portion were 1 and 3.

An investigation and discussion of these various learning parameters is provided in Chapter 5, and we refer the reader to this chapter for an explanation of these parameters. Finally, in order to calculate the affinity threshold (AT) for these simulated data sets, fifty data items were randomly generated. These fifty data items were subsequently discarded after being used for the affinity threshold calculation.

4.3 Results

With any classification algorithm, there are two primary questions which must be addressed. First, can the algorithm develop a representation of the classes in

the data set from exposure to training items? Second, can the algorithm, using this representation, generalize in order to accurately classify previously unseen data items? To address this first concern, in Figures 4.3 and 4.4 we present a visualization of the evolved memory cells after exposure to the 250 training antigens. The

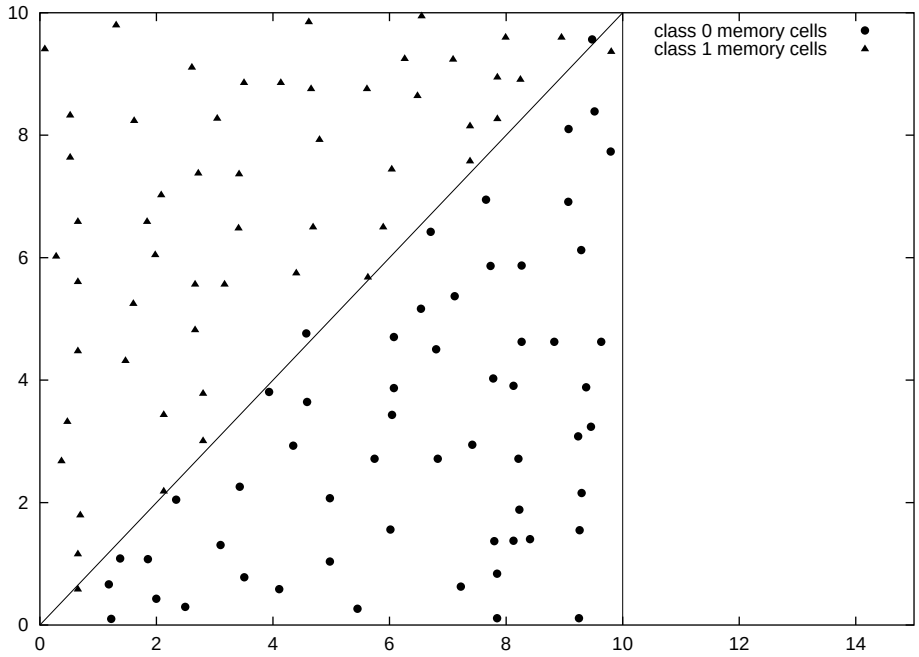

Figure 4.3: Memory Cells After 250 Antigens (Linearly Separable Data Set)

lines in these figures represent the class boundaries as depicted in Figures 4.1 and 4.2. As can be seen in these two figures, the memory cells evolved by the AIRS algorithm have developed a credible representation of the classes in the data set. Close examination of these figures reveals that this representation is not wholly perfect. This is particularly true in areas close to the class boundaries, as would be expected. One item of interest in Figure 4.4 on page 36 is that none of the class 1 memory cells are outside of the original class 1 boundaries, whereas there are a handful of class 0 memory cells that have bled over into class 1 territory. One

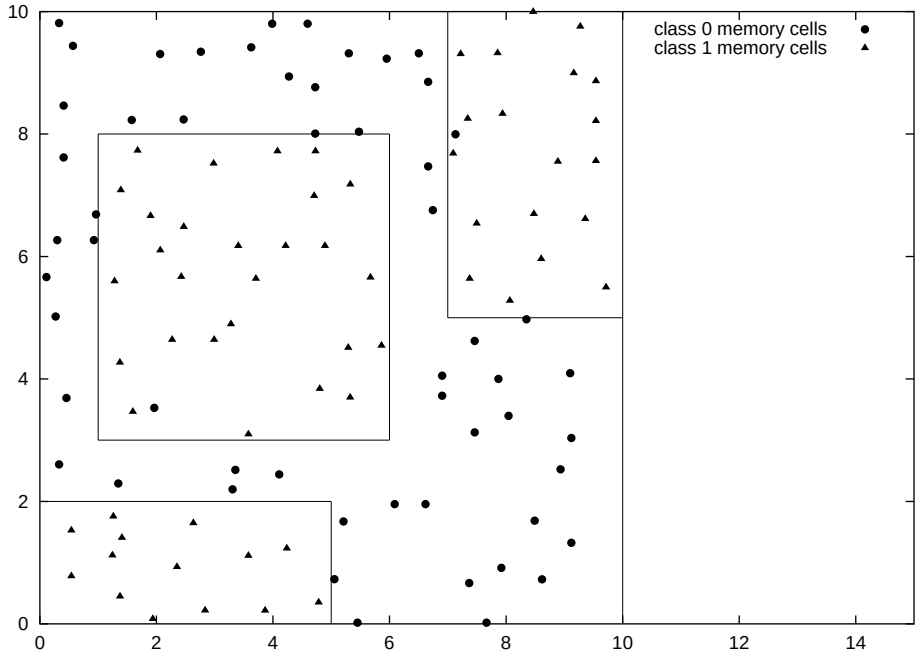

Figure 4.4: Memory Cells After 250 Antigens

possible explanation for this is that class 0 is less distinctly defined in space than class 1, which provides some ambiguity in the characteristics of a class 0 data item.

While perhaps not readily apparent from examination of Figures 4.3 and 4.4, there are far less than 250 memory cells in the final classifier. The data reduction and generalization capabilities of the algorithm are more easily discerned when the memory cells and training data items are presented together as seen in Figures 4.5 and 4.6. What is intriguing about these two figures is not just that there are far fewer memory cells than training items, but also that it is fairly easy to discern, in places, which memory cells are representing which groups of training items.

Another item of interest in these two figures is the outlier memory cells. While an initial supposition might be that these outliers developed in reaction to training items fairly close to the class boundary lines, examining these two figures reveals that some of the outlier memory cells are no closer to the training items than memory cells

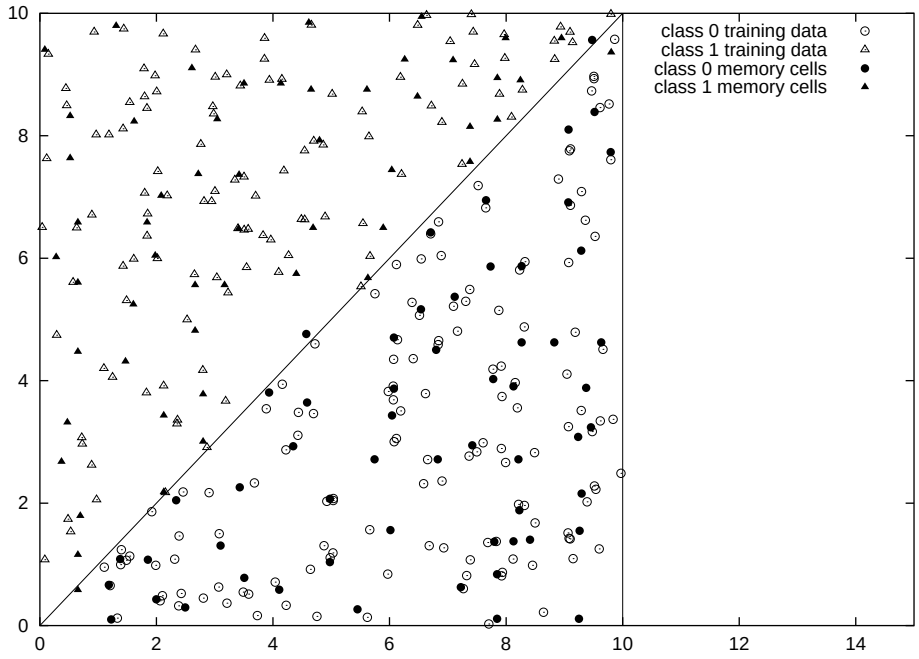

Figure 4.5: Memory Cells and Training Antigens After 250 Antigens (Linearly Separable Data Set)

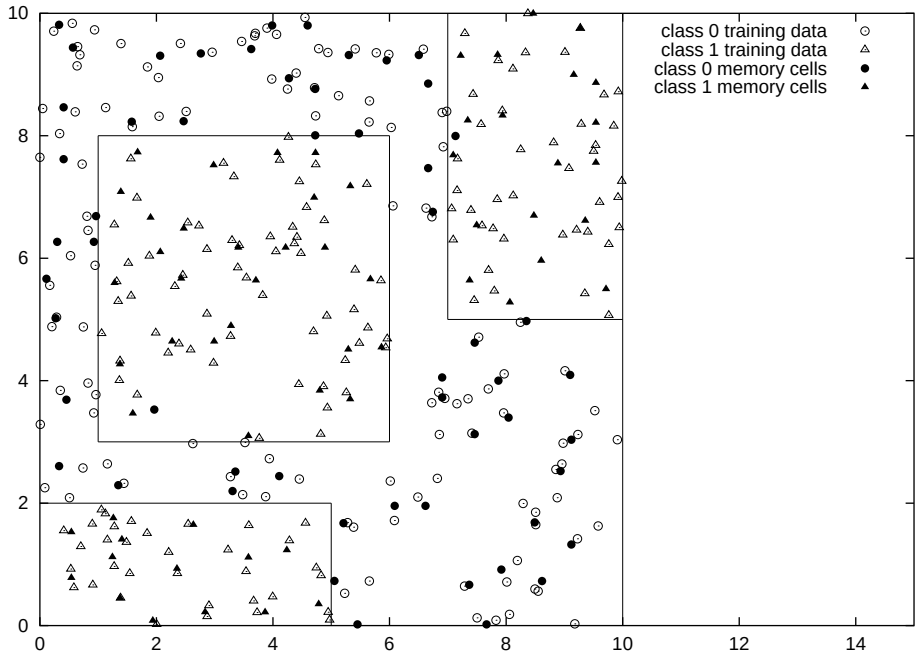

Figure 4.6: Memory Cells and Training Antigens After 250 Antigens

within the class boundaries. One possible explanation for this is the somewhat naïve method for memory cell replacement in the AIRS algorithm. Recall from Chapter 3 that memory cell replacement only occurs when a candidate memory cell is closer to the antigen than the established match memory cell and when the candidate memory cell is within a user defined distance of the match memory cell. While this mechanism is extremely useful in removing redundant memory cells from the set of memory cells, it does not necessarily ensure that only the best classifying memory cells are retained. For the outlier cells in Figures 4.5 and 4.6 that appear to be further away from the training data items than some of the memory cells that are within the class boundaries, it is possible that the outlier cells developed first and that, while subsequently evolved memory cells were more stimulated by training antigens than these outlier cells, none of these later evolved memory cells were close enough to the outlier cells to warrant the removal of the outlier cells.

Now that we have provided a demonstration of the AIRS algorithm's ability to develop a representation of the classes present in these two simple data sets, we next turn to the question of classification of previously unseen data items using this representation. For these experiments the quality of classification is only assessed through overall accuracy. That is:

$$\text{accuracy}(T) = \frac{\sum_{i=1}^{|T|} \text{assess}(t_i)}{\mid T \mid}, t_i \in T \tag{4.1}$$

$$\text{assess}(t) = \begin{cases} 1 & \text{iff} \quad \text{classify}(t) \equiv t.c \\ 0 & \text{otherwise} \end{cases}$$

where T is the set of data items to be classified (the test set in this case), $t \in T$, $t.c$ is the class of item t, and classify(t) returns the classification of t by the developed classifier. Recall from 3.2.5 that classification is performed in a k-nearest neighbor

approach with the classification of an item being based on the majority vote among the k nearest memory cells to the item. For the current experiments k values of 1 and 3 were chosen. Figures 4.7 and 4.9 depict the classification accuracy on the test set as antigens are introduced. At the end of the introduction of 250 antigens, the system is able to accurately classify 94% and 98% (47/50 and 49/50) for $k = 1$ and $k = 3$, respectively, of the linearly separable test set. And the system is able to accurately classify 86% and 94% (43/50 and 47/50) for $k = 1$ and $k = 3$, respectively, of the non-linearly separable test set. As might be expected, the more antigens that have been introduced to the system the better the accuracy of the classification, since more antigens imply more memory cells. Figures 4.8 and 4.10 are presented in juxtaposition to the two accuracy figures to provide a sense of the number of memory cells that existed in the system at the time of classification. What is interesting to note from this comparison is that for the linearly separable test set the accuracy for $k = 1$ reaches its peak fairly early indicating that this classification task requires fewer training items and fewer memory cells for accurate classification. Not surprisingly, the accuracy on the non-linearly separable data set is less than the classification accuracy on the linearly separable data set. This is to be expected for a more complex domain. Nevertheless, Figure 4.9 does show that even for more complex data sets the AIRS algorithm is able to perform fairly well as a classifier.

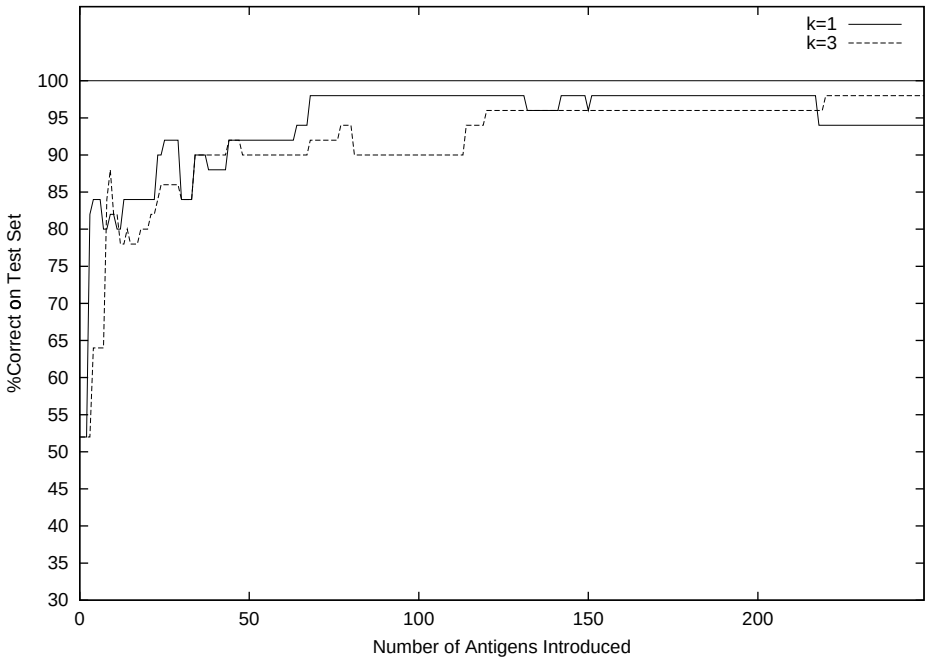

Figure 4.7: Classification Improvement Over Time (Linearly Separable Data Set)

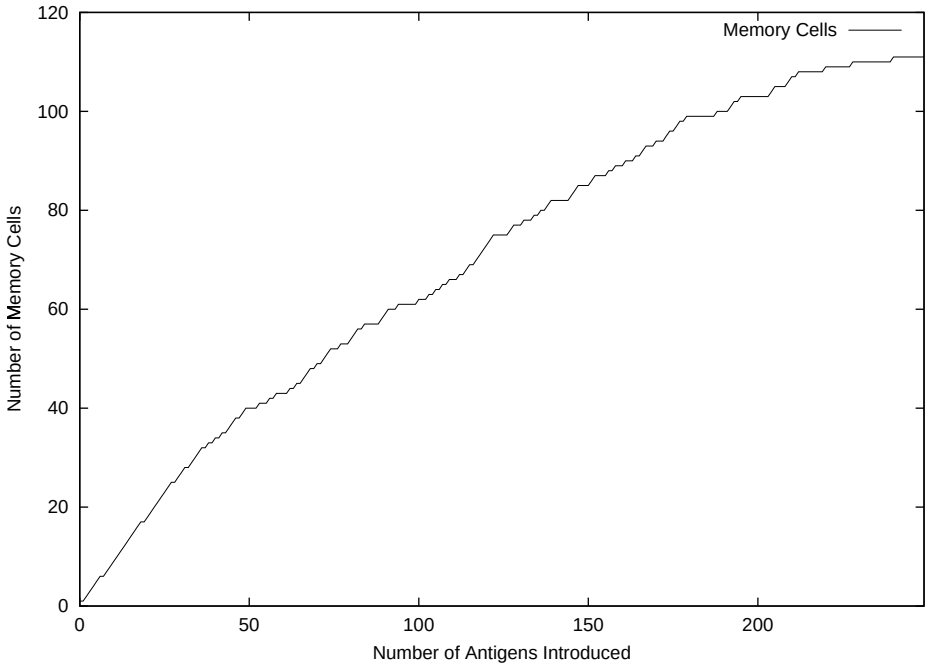

Figure 4.8: Memory Cells vs. Number of Antigens (Linearly Separable Data Set)

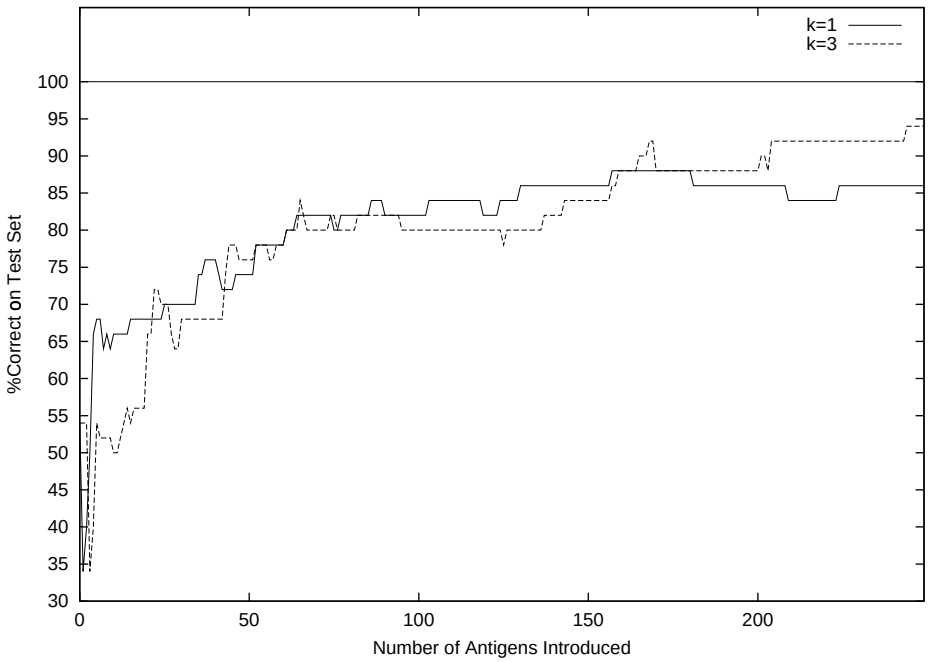

Figure 4.9: Classification Improvement Over Time

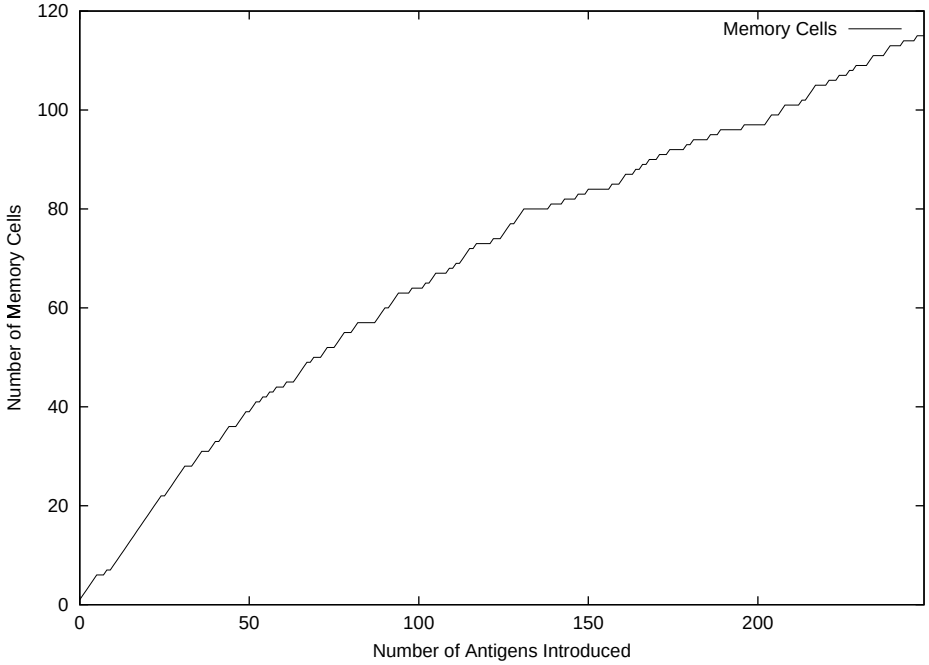

Figure 4.10: Memory Cells vs. Number of Antigens

CHAPTER V

INVESTIGATION OF AIRS: MACHINE LEARNING BENCHMARKS

To compliment chapter 4, this chapter explores the behavior of the current implementation of the AIRS algorithm on standard machine learning data sets. The data sets used in this chapter have been employed to investigate the performance of many different classification systems. In brief, the two primary purposes of this chapter are as follows:

1. to present an investigation of the effect of the various learning parameters on the classification performance of the algorithm, and
2. to present a comparison of the accuracy of the current implementation to the accuracy of other classifiers based on standard machine learning data sets.

This chapter is organized as follows. Section 5.1 discusses the four data sets used in this investigation and presents the experimental design employed when exploring each of these data sets. Sections 5.2 through 5.7 provide a discussion of the six learning parameters under investigation and the results obtained when varying each of these parameters. The parameters discussed are as follows:

- the number of seed cells used in initialization,
- the number of resources allowed in the system,
- the stimulation threshold used as a training stopping criterion,
- the mutation rate for feature mutation,
- the affinity threshold scalar employed for memory cell replacement, and
- the k value used for classification.

5.1 Data Sets and Experimental Design

Before discussing the individual data sets, let us first examine the characteristics the four data sets used for this chapter have in common. First, all four data sets have continuous-valued data for their features. While eventually use of binary and discrete valued data should be explored, the current implementation is suited best for continuous value data. Second, all four data sets have been widely used for evaluation of fledgling classifiers. The experimental design is such that the results from this study will be easily comparable to results reported elsewhere. Finally, the set of possible classes is small for all four data sets. In fact, three of the four data sets are two-class problems and the fourth data set is a three-class problem. All the data sets used for this chapter were obtained from the University of California, Irvine Machine Learning Repository (Blake and Merz 1998).

5.1.1 Fisher's Iris Data Set

The Fisher iris data is one of the most well-known data sets in the machine learning literature. It is a simple data set consisting of four features and three classes. There are a total of 150 data items in the set with 50 from each of the classes (iris setosa, iris versicolour, and iris virginica). The four features represent attribute measurements of the different iris plants. Specifically, the four features are sepal length, sepal width, petal length, and petal width. One of the classes is linearly separable from the other two which are not linearly separable from each other.

For the current experiments, the data set was randomly divided into five disjoint sets. These sets were used for five-fold cross validation experiments. That is, a concatenation of four of the sets were used as the training set and the fifth set was used as the test set. In this way, we had five training-test set combinations with each

set being the test set once. For any given run, the results are an average of three runs on each of the five training-test set combinations.

The default parameter settings for these runs were as follows:

- the number of seed cells was 1,
- the number of resources allowed in the system was 200,
- the stimulation threshold was set to 0.9,
- the mutation rate was set to 0.1,
- the affinity threshold scalar (ATS) was set to 0.2, and
- the k values used for classification were 1 and 7.

5.1.2 Ionosphere Data Set

The ionosphere data set is a two-class problem with 34 numeric features. The classification task is to determine "good" and "bad" radar returns from the atmosphere, where "good" returns are those that indicate structure in the ionosphere and "bad" ones do not.

The usages of this data set in the literature are fairly rigid in their data division. The first 200 items are chosen for the training set and the remaining 151 items are chosen for the test set. The current experiments conform to this data division scheme. All results reported are an average of three runs with the parameter settings in questions.

The default parameter settings from these runs were as follows:

- the number of seed cells was 1,
- the number of resources allowed in the system was 500,
- the stimulation threshold was set to 0.8,
- the mutation rate was set to 0.1,
- the affinity threshold scalar (ATS) was set to 0.2, and
- the k values used for classification were 1 and 3.

5.1.3 Pima Indians Diabetes Data Set

The Pima Indians diabetes data set represents a two-class problem with 8 numeric features. The classification task is to determine if the patient tested positive for diabetes (class 1) or not (class 0).

There are 768 instances in this data set with 500 instances (65%) from class 1 and the remaining 268 instances from class 0. For the current experiments this data set was randomly divided into 10 disjoint sets to perform 10-fold cross validation in much the same way that the iris data set was divided for 5-fold cross-validation. All results reported are an average of three runs on each of the 10 training-test set combinations.

The default parameters settings for these experiments were as follows:

- the number of seed cells was 1,
- the number of resources allowed in the system was 200,
- the stimulation threshold was set to 0.9,
- the mutation rate was set to 0.1,
- the affinity threshold scalar (ATS) was set to 0.2, and
- the k values used for classification were 1 and 3.

5.1.4 Sonar Data Set

The sonar data set is a two-class problem with 60 numeric features. The classification task is to determine whether a sonar signal bounced back from a metal or rock object, with the ultimate goal being the development of a system that can recognize the difference between a rock and a mine.

The experiments presented here follow the same data division as discussed in Blake and Merz (1998) for the aspect-angle independent experiments. Namely, the data set was randomly divided into 13 disjoint sets with 16 instances in each set.

The 13-fold cross validation was performed so that each item appeared in a test set once. All results reported are an average of ten runs on each of the 13 training-test set combinations.

The default parameter settings for these experiments were as follows:

- the number of seed cells was 1,
- the number of resources allowed in the system was 200,
- the stimulation threshold was set to 0.9,
- the mutation rate was set to 0.1,
- the affinity threshold scalar (ATS) was set to 0.2, and
- the k values used for classification were 1 and 3.

5.2 Seed Cells

This section examines the effect of varying the number of seed cells used in system initialization on classification accuracy. Recall from chapter 3 that during system initialization a random number of training items can be chosen to provide the initial memory cells and ARBs in the system. These seed cells are randomly chosen with replacement which implies that those data items chosen for seeding the system will remain in the training set. In general, what is expected from increasing the number of seed cells is that, at a minimum, the classification accuracy on the training set will increase. This expectation is due to the fact that, as training data items are presented to the system, some of the match memory cells will be exact matches if the given training item was one of those randomly chosen to seed the system. However, increasing the number of seed cells could potentially lead to an over-specific memory cell set which has been trained to recognize the training set quite well but has lost the ability to generalize beyond this.

Figures 5.1 through 5.4 present the results obtained on the four data sets by varying the number of seed cells. The range of seed cells explored was $\{1, 5, 10, 20, 30 \ldots 100\}$. Examining these results we do indeed see that in general the accuracy on the training set (at least for $k = 1$) increased. In fact, for all four data sets there is a general tendency toward increased accuracy as the number of seed cells increased. This general trend is most visibly evident in the results for the sonar data set (figure 5.4). Interestingly, we do not really see a decrease in the accuracy on the test set as the accuracy on the training set increases.

However, it should be observed that for all the test sets (except the sonar data set) the peak accuracy is achieved at some point other than the maximum number of seed cells evaluated. This is encouraging from the point of the AIRS algorithm design. One of the more interesting features of the AIRS algorithm is the ability to evolve a classification model with limited initialization. This ability is reinforced by peak test set classification being achieved on a smaller number of seed cells.

One final result to point out from this set of experiments is the behavior of the evolved classifiers on the ionosphere data set. Examining figure 5.2 we see that there are certain ranges of seed cells where the accuracy on the test set is greater than the classification accuracy on the training set. This is somewhat contradictory for most machine learning paradigms when it is almost a given that training set accuracy will be greater than test set accuracy. However, we will observe this trend in the ionosphere data set for all of the parameters discussed. One possible explanation for this is that, compared to the other data sets, the balance between the number of items in the training set and test set is more even. That is, for the ionosphere data set there are 200 training items and 151 test items, nearly a 1:1 ratio, contrasted with the 5:1, 10:1, and 13:1 ratio of the other data sets.

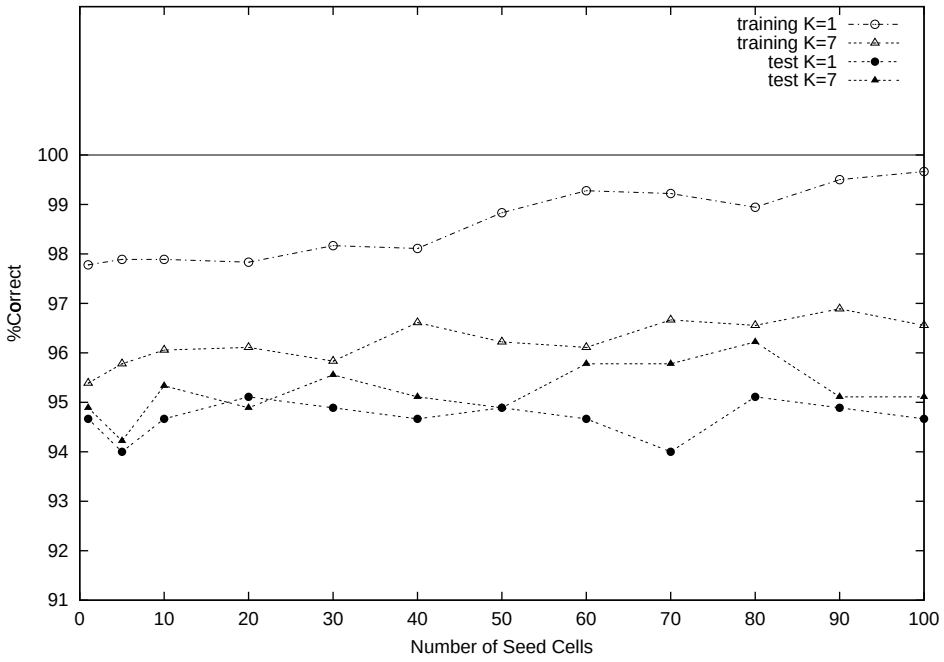

Figure 5.1: Seed Cell Variation (iris)

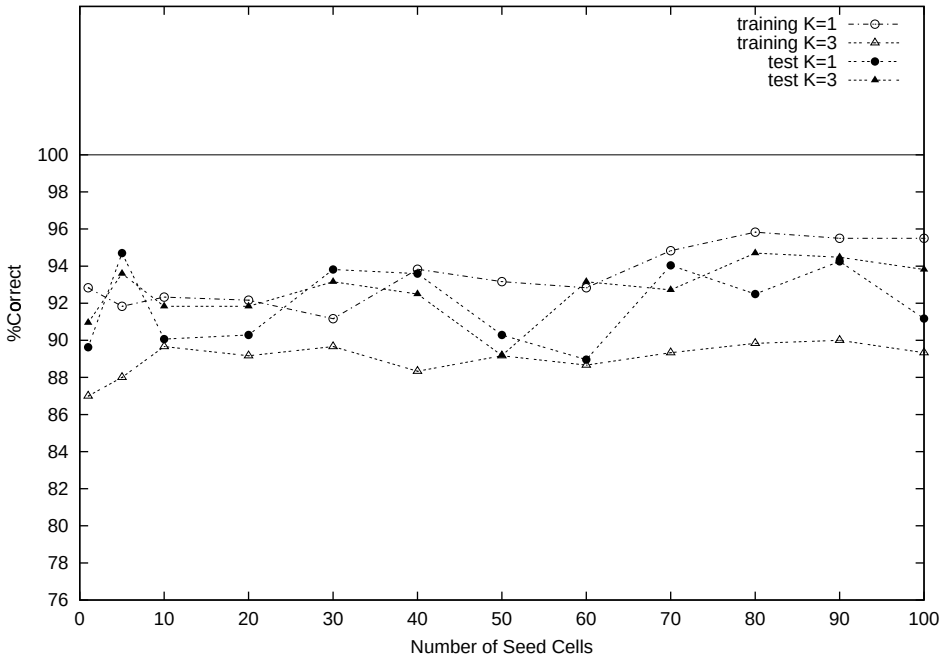

Figure 5.2: Seed Cell Variation (ionosphere)

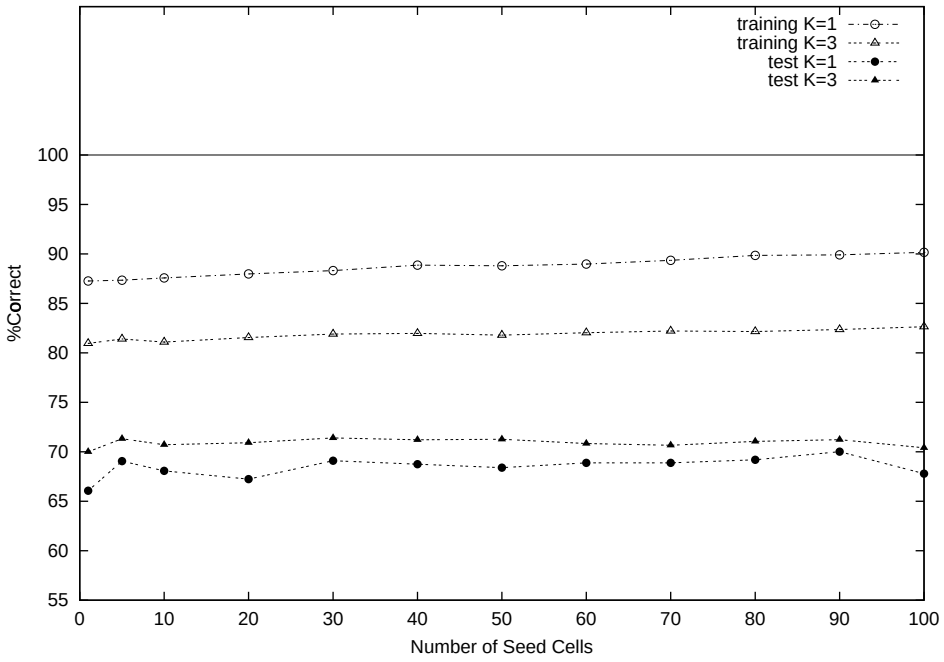

Figure 5.3: Seed Cell Variation (diabetes)

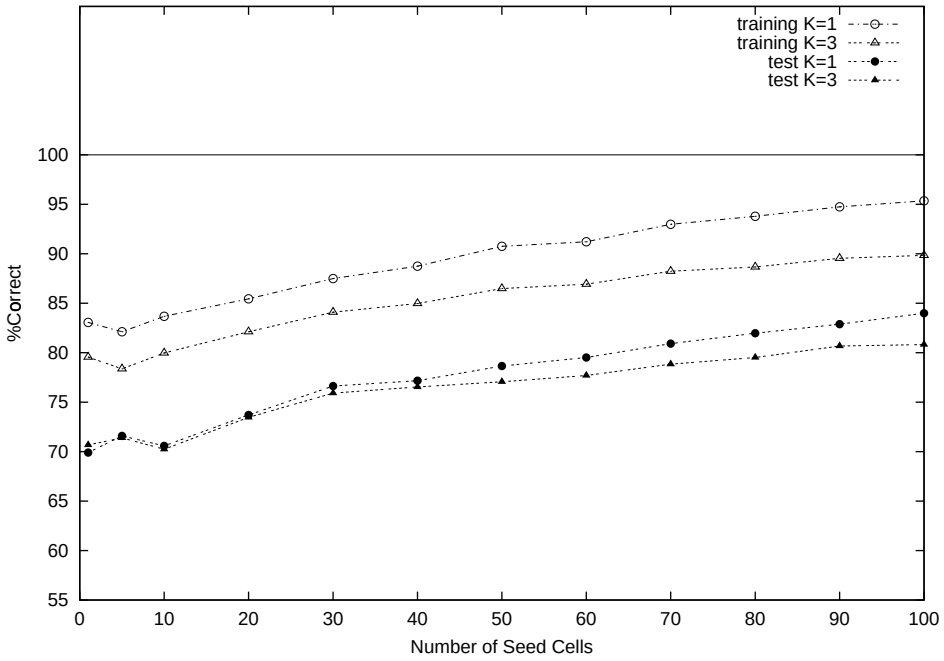

Figure 5.4: Seed Cell Variation (sonar)

5.3 Resources

The next learning parameter we investigate is the number of resources in the system. The number of resources has a direct effect on the number of ARBs present in the system as ARBs with zero resources are removed from the system after each training iteration. With this in mind, increasing the number of resources should increase the diversity of the ARB pool from which memory cells are evolved, as more ARBs will be allowed to remain alive. While there is no guarantee that this increased diversity will result in increased accuracy, in general this is what is expected.

Figures 5.5 through 5.8 present the results obtained on the four data sets when varying the number of resources in the system. The range of resources investigated was $\{50, 100, 150, 200, \ldots 500\}$. Examining the results presented in these figures, we do not find the clear trends that were observed for seed cell increases. That is, increasing the number of resources does not seem to necessarily lead to increased accuracy. In fact, for the most part we see that certain numbers of resources allow for better performance than others, but these numbers appear to be data set specific.

As previously mentioned, it was hypothesized that an increase in the number of resources would lead to an increase in classification accuracy. While this trend was not exhibited for the test set accuracy, it is somewhat noticeable for the training set accuracy. One possible explanation for this is only readily apparent when the effect of the number of resources is taken in conjunction with the concept of the stimulation threshold. Recall that training on a given antigen only ceases when the average stimulation value of the each class-specific subset of the ARBs rises above the stimulation threshold. So while an increase in the number of resources potentially results in an increase in the number of ARBs, all of these ARBs must be highly stimulated ones. Thus, this increased population is in actuality an increased

population highly attuned to the training set. Therefore, we do see some trend toward classification improvement on the training set, but this does not seem to carry over to increased classification accuracy (or ability to generalize) to the previously unseen test sets. Again, it should be observed that for the ionosphere data set, the test set accuracy is almost always greater than the accuracy on the training set.

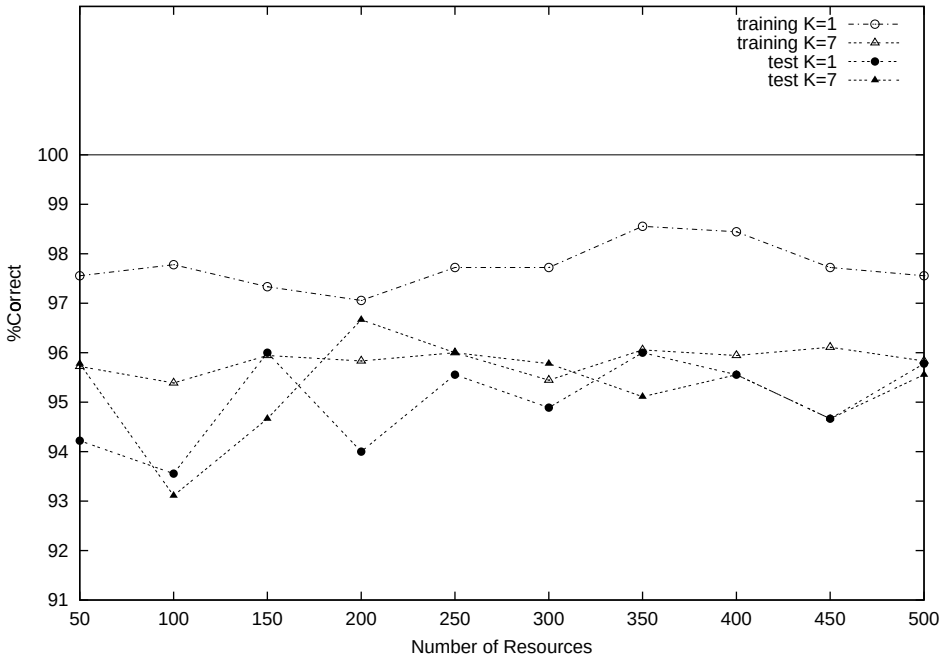

Figure 5.5: Number of Resources Variation (iris)

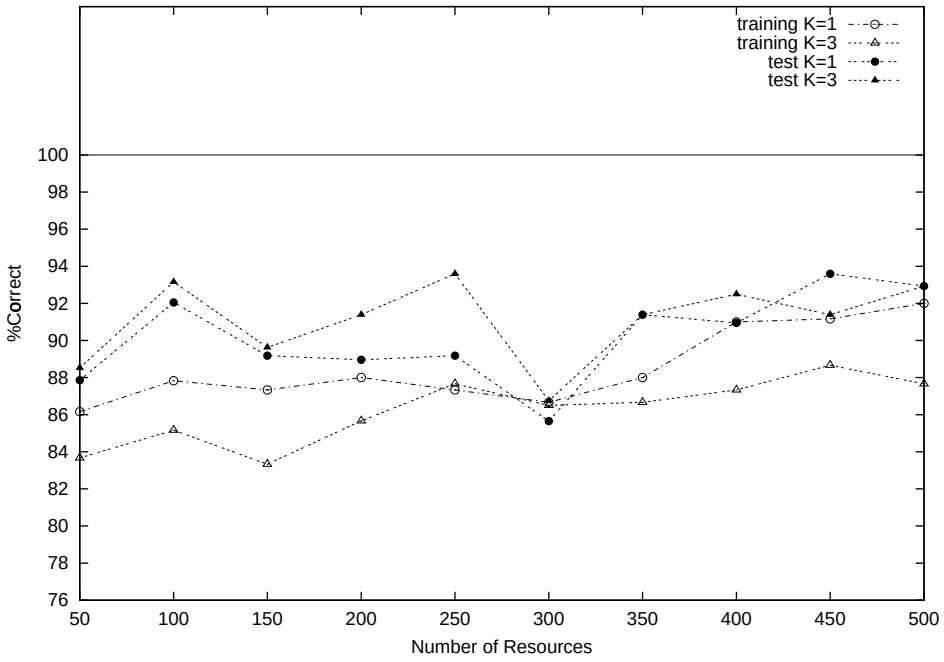

Figure 5.6: Number of Resources Variation (ionosphere)

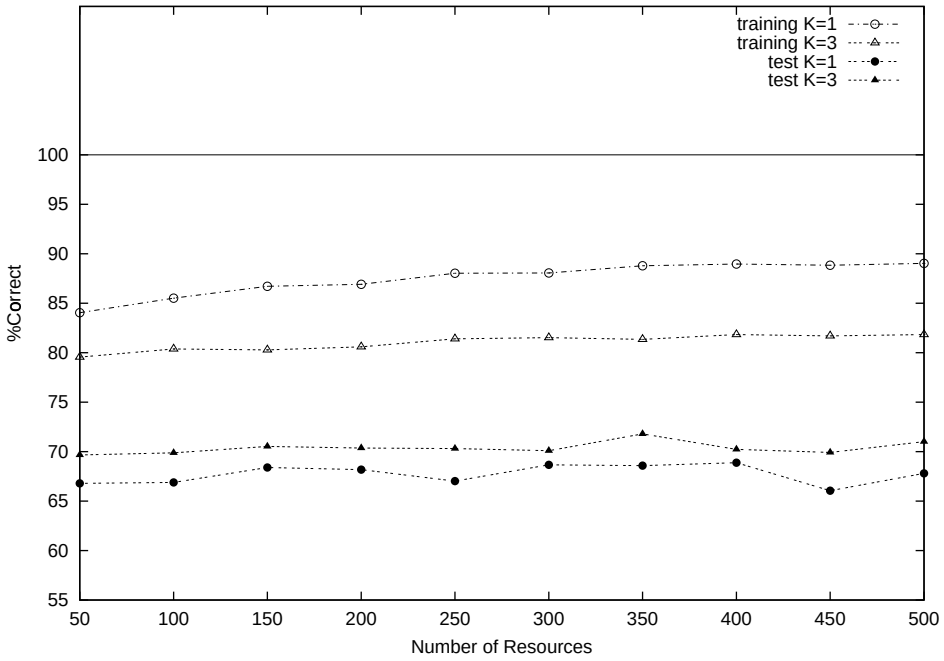

Figure 5.7: Number of Resources Variation (diabetes)

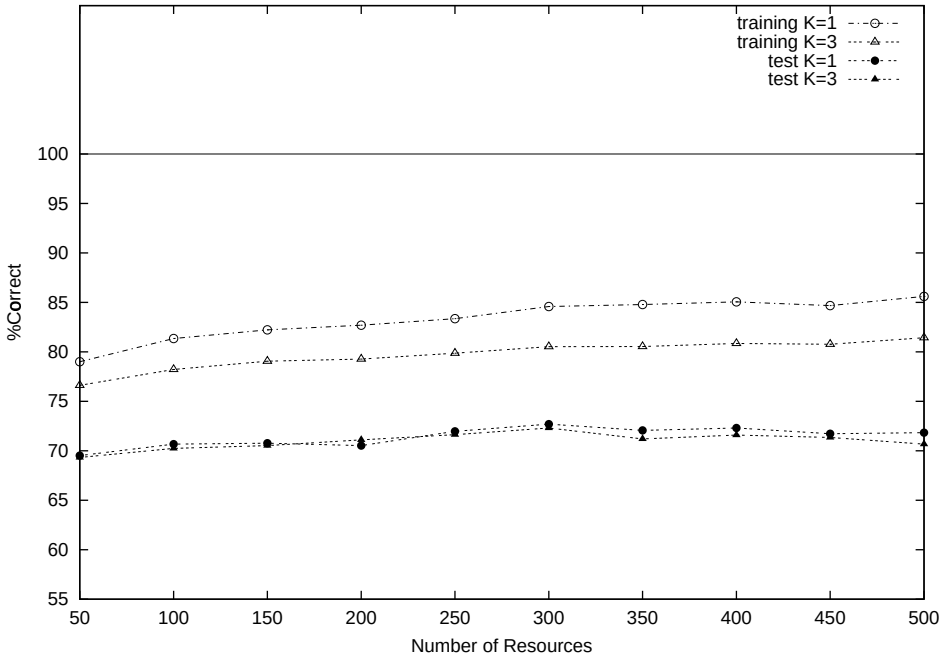

Figure 5.8: Number of Resources Variation (sonar)

5.4 Stimulation Threshold

Next, we investigate the effect of the stimulation threshold on the classification accuracy of the system. As mentioned in the discussion of resource variation in section 5.3, the stimulation threshold is used as a stopping criterion for training the ARB population on a specific antigen. For clarity, we repeat the use of the stimulation threshold provided in section 3.2 here. Define a vector $\vec{s}$ that is number of classes, nc, in length to contain the average stimulation value for each class subset of the ARB population AB. That is:

$$s_i \leftarrow \frac{\sum_{j=1}^{|AB_i|} ab_j.stim}{\mid AB_i \mid}, ab_j \in AB_i$$

The stopping criterion is reached iff $s_i \geq stimulation\_threshold$ for all elements in $\vec{s} = \{s_1, s_2, \ldots s_{nc}\}$. So, the stimulation threshold dictates how stimulated the ARB population must be before it is said to have learned a particular antigen. Recall, again from section 3.2, that the stimulation value of an ARB of the same class as the training antigen is directly proportional to the Euclidean distance of the antigen and the ARB, whereas the stimulation value of an ARB of a different class as the training antigen is proportional to 1 - the Euclidean distance of the antigen and the ARB. With this in mind, it is expected that as the stimulation threshold increases the classification accuracy will also increase or at least the training set classification accuracy will increase. Again, there is the possibility that an increase of a stimulation threshold will result in evolved memory cells that are too specific to the training set and not able to generalize to previously unseen data items well.

Figures 5.9 through 5.12 present the results of varying the stimulation threshold on the four data sets. The range of values investigated was $\{0.1, 0.2, 0.3, \ldots 0.9, 0.95\}$. Based on these figures, it appears that the stimulation threshold value has very little effect on the iris, diabetes, and sonar data sets training set classification accuracy, with the exception that we do see a slight rise in accuracy between the 0.9 and 0.95 settings for the latter two data sets. The stimulation threshold appears to have a fairly erratic effect on the ionosphere data set with the general trend being increased accuracy on both the training and test sets. Again we notice that the accuracy of the classifier on the ionosphere test set is generally greater than the accuracy on the ionosphere training set.

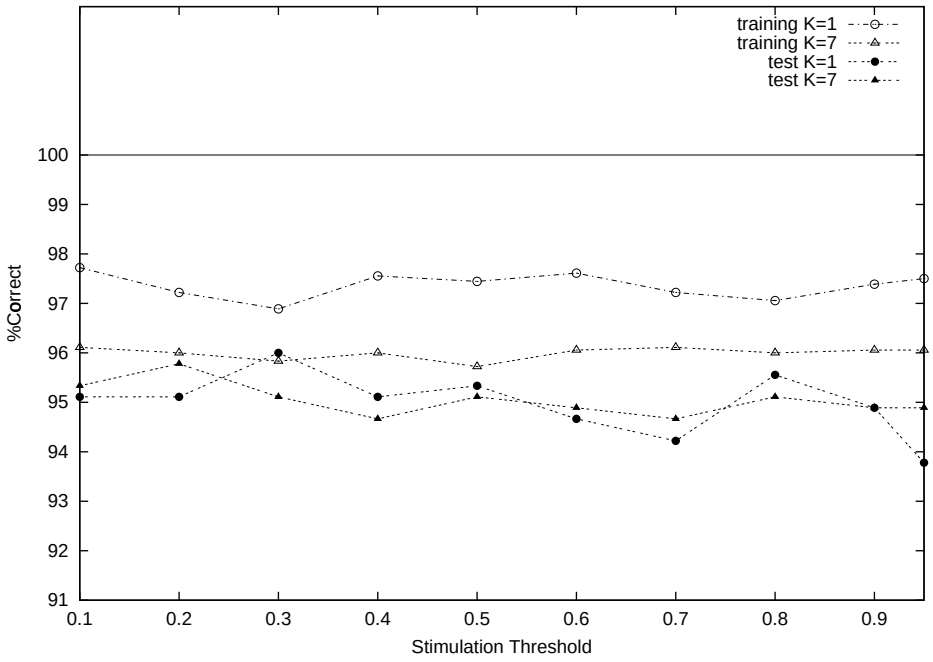

Figure 5.9: Stimulation Threshold Variation (iris)

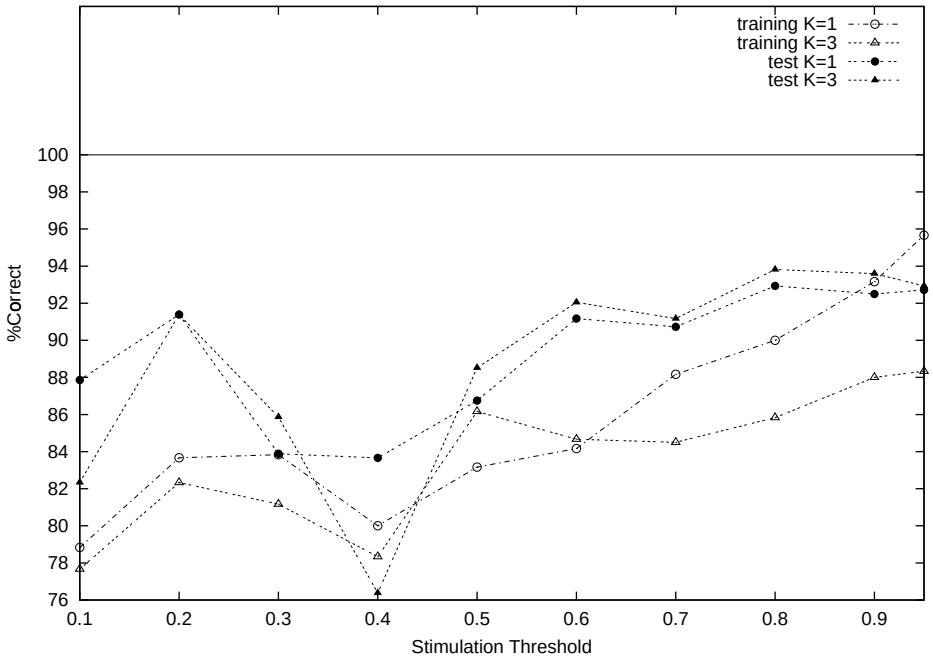

Figure 5.10: Stimulation Threshold Variation (ionosphere)

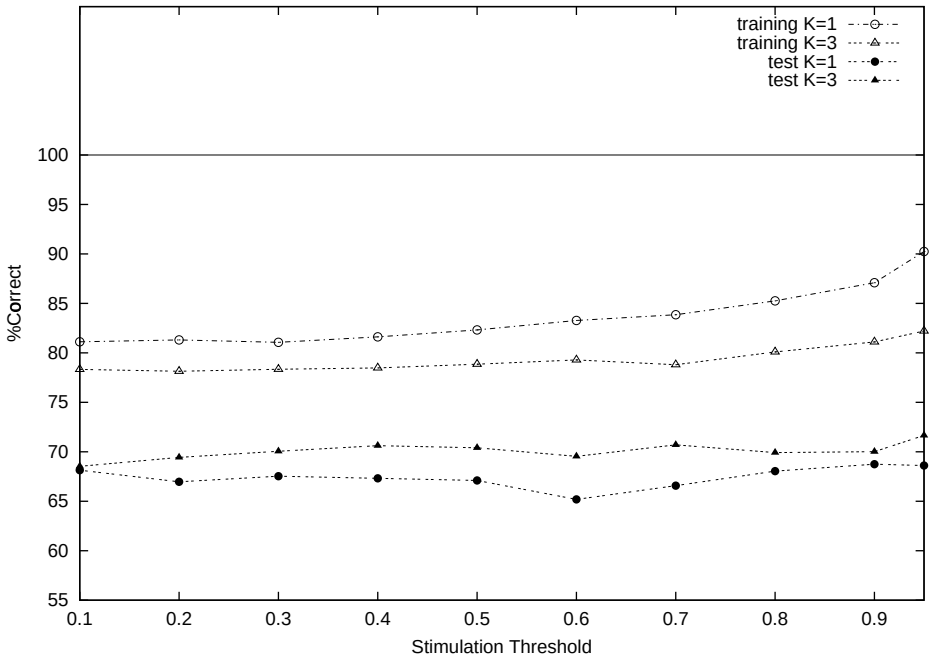

Figure 5.11: Stimulation Threshold Variation (diabetes)

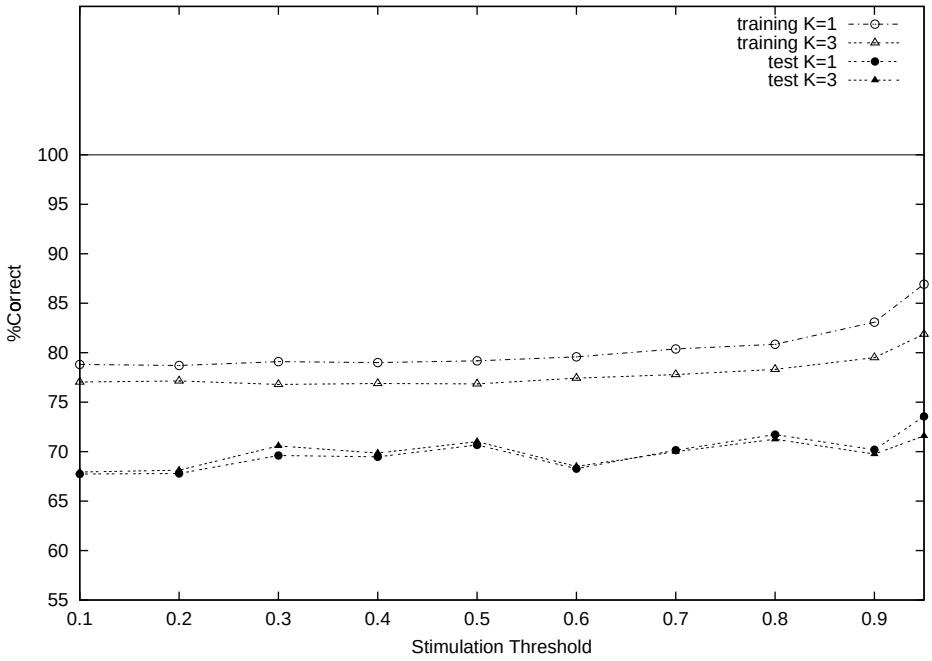

Figure 5.12: Stimulation Threshold Variation (sonar)

5.5 Mutation Rate

The mutation rate is expected to affect the diversity of the evolving ARB population. However, since the mutations in the current implementation are truly random it is possible for an increased mutation rate to have a negative effect on the performance of the algorithm both in terms of classification accuracy and execution time. The hypothesized negative impact on the classification accuracy is due to the fact that random mutations can result in ARBs that are further away from the target search space. This also can directly impact execution time as the training stopping criterion is based on the average stimulation value for a set of ARBs. Since the mutations are random, a high mutation rate can result in the production of numerous offspring that will not be highly stimulated by the training antigen.

Figures 5.13 through 5.16 present the results for mutation rate variation on the four data sets. The range of values explored was $\{0.1, 0.2, \ldots 1.0\}$. As can be seen from these figures, there is a general degradation in the classification accuracy with the increase in the mutation rate. Also, for the ionosphere and sonar data sets the results presented are incomplete. This is due to the fact that the execution time for both of these data sets increased beyond a manageable level (read multiple days of running). A notable exception to this general degradation in the classification accuracy is found in the iris data set (figure 5.13). For this data set, the peak test set accuracy was found when the mutation rate was 0.8. This phenomenon is somewhat inexplicable given the trends noticed throughout this and the other data sets.

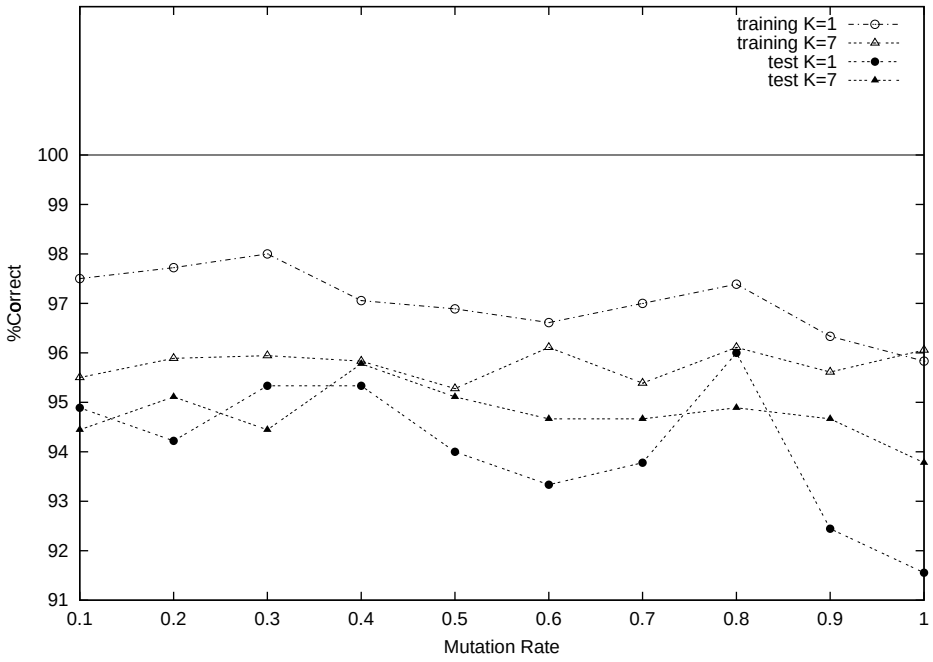

Figure 5.13: Mutation Rate Variation (iris)

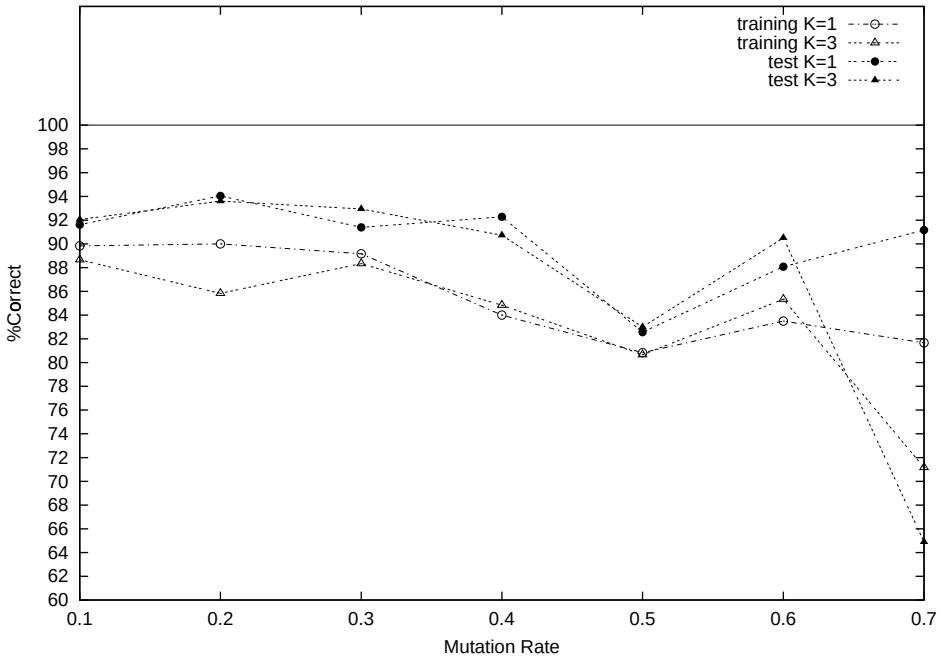

Figure 5.14: Mutation Rate Variation (ionosphere)

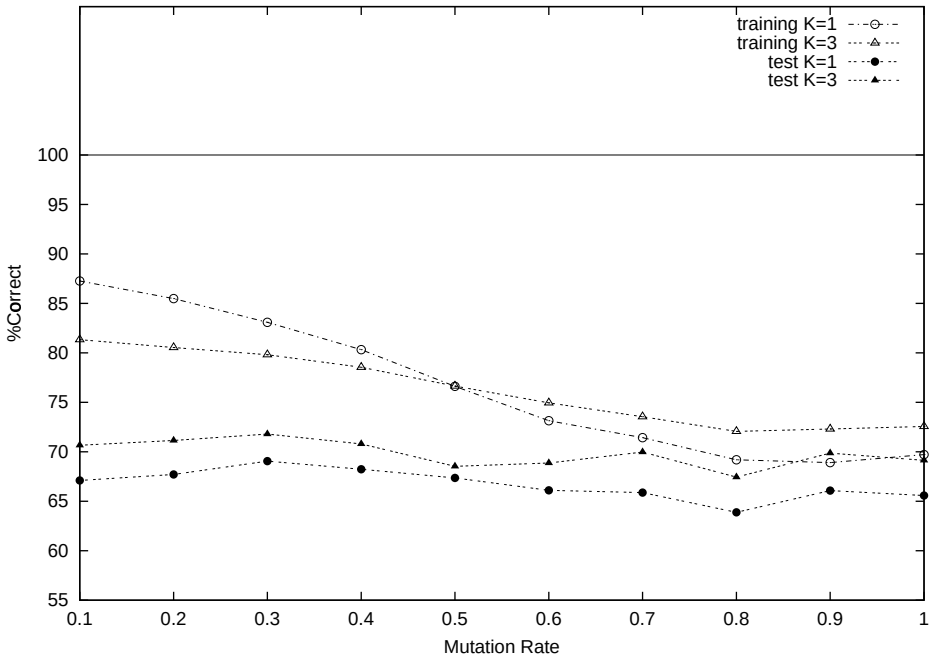

Figure 5.15: Mutation Rate Variation (diabetes)

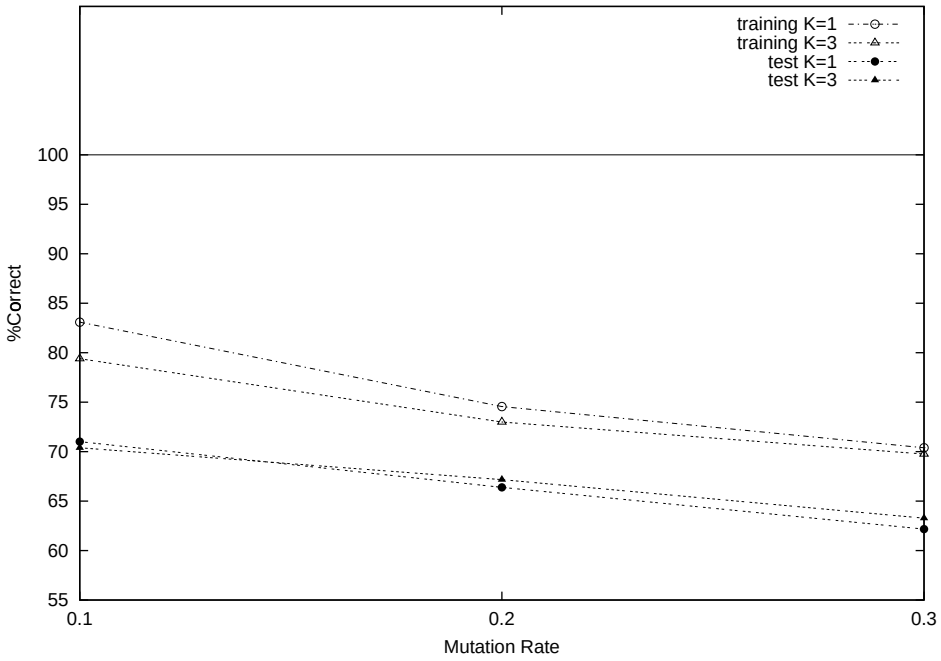

Figure 5.16: Mutation Rate Variation (sonar)

5.6 Affinity Threshold Scalar

The affinity threshold scalar (ATS) has the most direct effect on memory cell replacement. Recall from section 3.2 that the affinity threshold is defined as the average affinity (Euclidean distance) value of the pairwise affinity among the training set data items. The ATS is a user-defined scalar which is multiplied by the affinity threshold to determine memory cell replacement. That is, a newly evolved memory cell replaces an established memory cell if and only if it is more stimulated by the training antigen and the affinity between the two memory cells (candidate and match) is less than the product of the affinity threshold and the ATS. As the ATS increases, it is expected that the number of memory cells left in the system will decrease. While this is expected to have little impact on the classification accuracy of the system for relatively small values of ATS, the increase in the ATS (and thus decrease in the number of memory cells) is expected to have a dramatic impact on the classification accuracy for larger ATS values. This should be particularly true when system classification is based on a k value greater than 1.

Figures 5.17 through 5.20 present the results obtained by varying the ATS on each of the four data sets. The range of values explored was $\{0.1, 0.2, \ldots 1.0\}$. As expected, we find a fairly rapid decrease in the classification accuracy as the ATS is increased, particularly for values of k greater than one. Again, the ionosphere data set appears to be somewhat of an anomaly to these general trends. For the ionosphere data set, we find that the peak test set accuracy is achieved when the ATS is 0.8 and k is one.

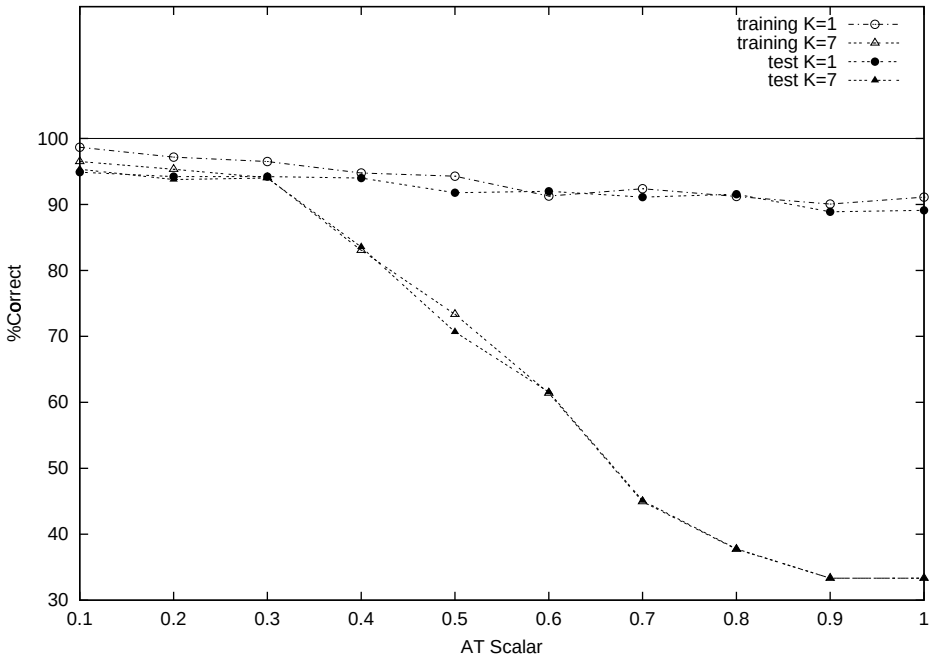

Figure 5.17: ATS Variation (iris)

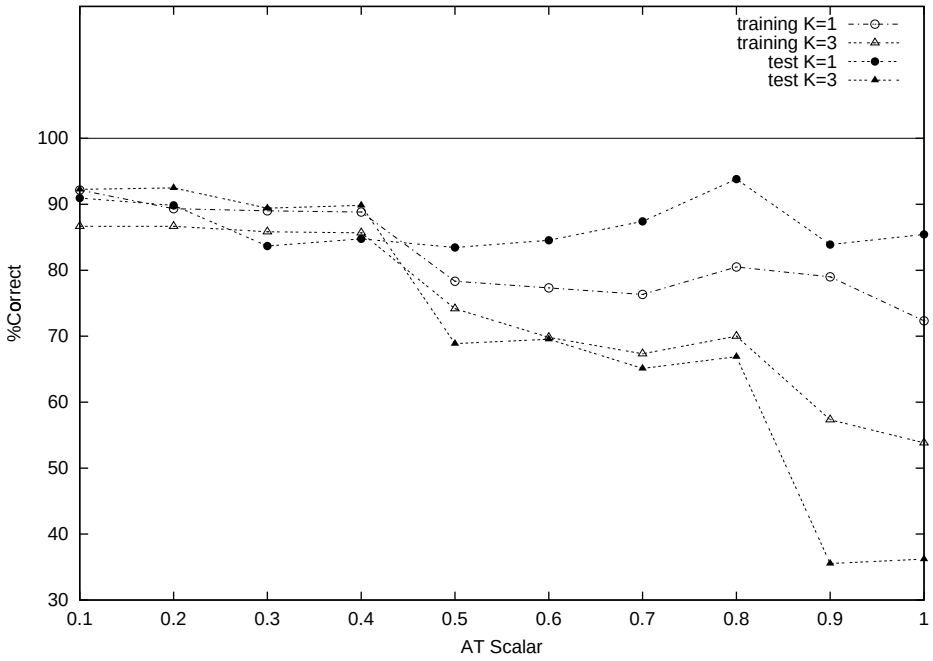

Figure 5.18: ATS Variation (ionosphere)

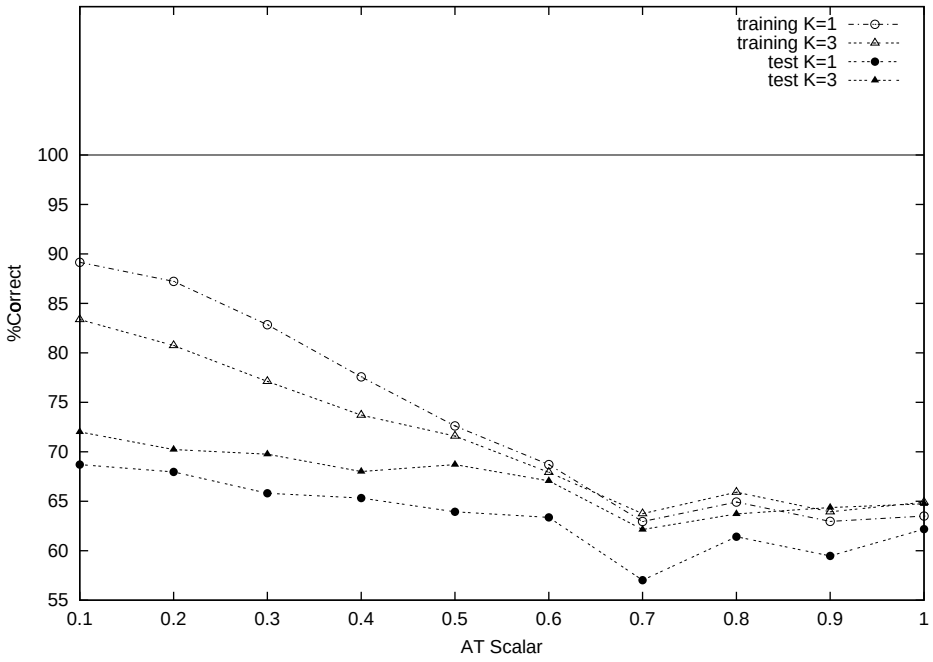

Figure 5.19: ATS Variation (diabetes)

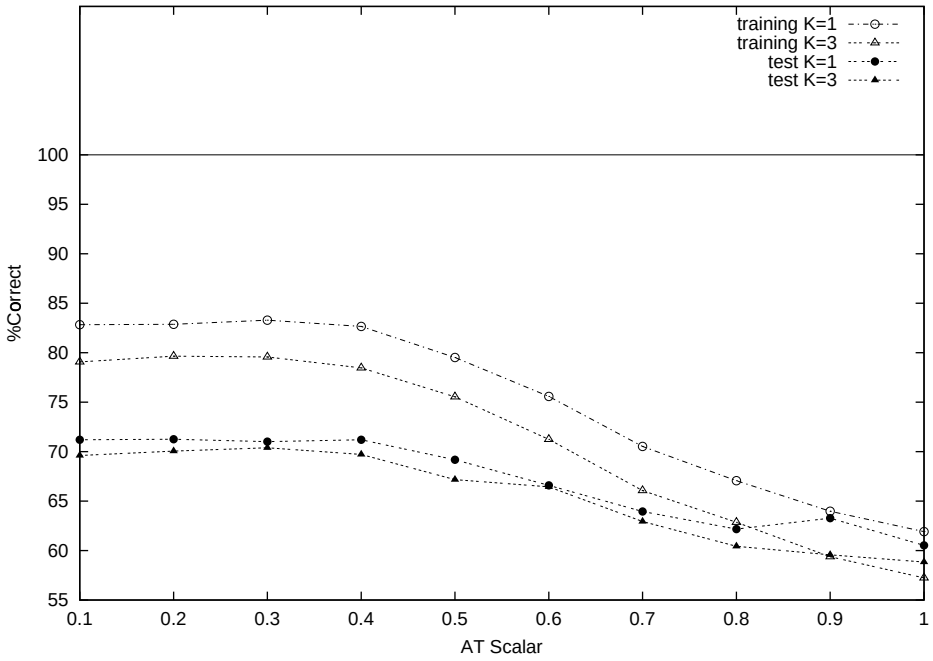

Figure 5.20: ATS Variation (sonar)

5.7 K Value

The final user parameter investigated in this set of experiments was the k value used for classification. Recall that the classification of the trained classifier is based on a k-nearest neighbor approach. That is, the classification of any data item is based on a majority vote among the k closest memory cells to the data item, or, in the case of a three-class problem, the classification could be based on the largest plurality. If there is no majority and no dominant plurality, then the classification of the system is chosen on a strictly ordered basis with the lowest number class among the "tied" classes always being chosen. In general, there is no clear way to predict the "best" k value for a given data set. It is extremely dependent on the nature of the data set itself.

Figures 5.21 through 5.24 present the results obtained when varying the k values. The range of values tested was $\{1, 2, \ldots 10\}$. Examining these results, we find that there are no real trends across all four data sets to observe. For the iris data set, there appear to be two k values that provide the best test set accuracy, namely 3 and 7. For the ionosphere data set, 3 appears to be a good choice again, along with, surprisingly, 10. For the diabetes test set there seems to be a general trend of increased k value translating into increased test set accuracy. This finding will be revisited in section 5.8. Finally, for the sonar test set the k value appears to have little effect except a couple of small peaks in the test set accuracy at k values of 4 and 7.

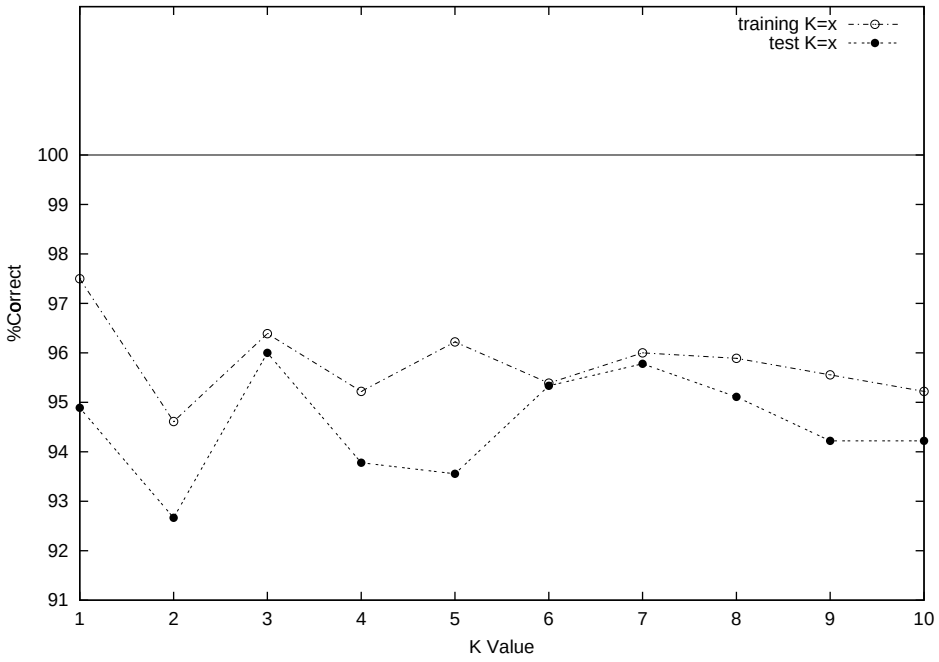

Figure 5.21: K Value Variation (iris)

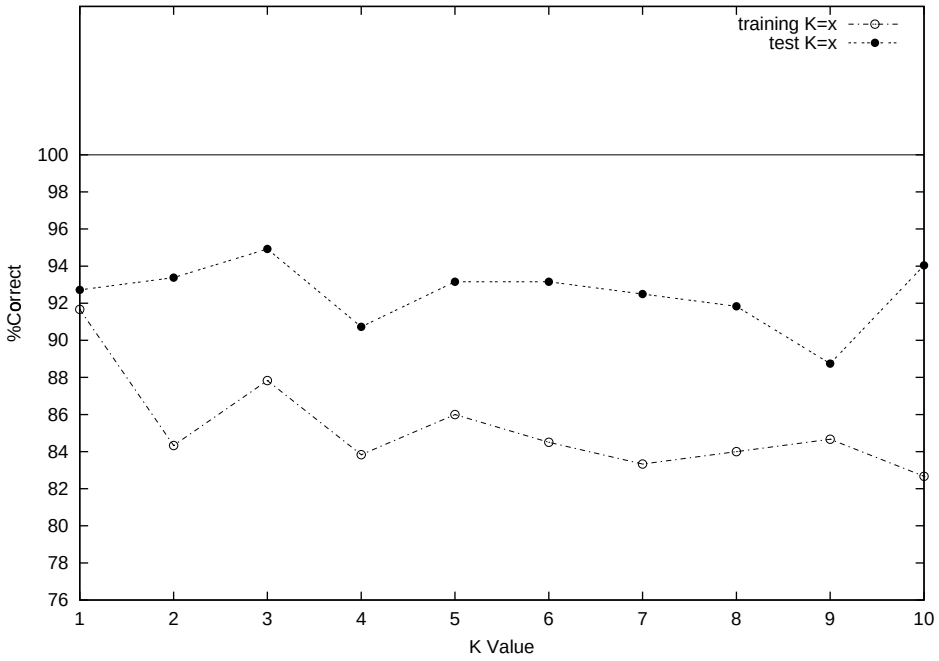

Figure 5.22: K Value Variation (ionosphere)

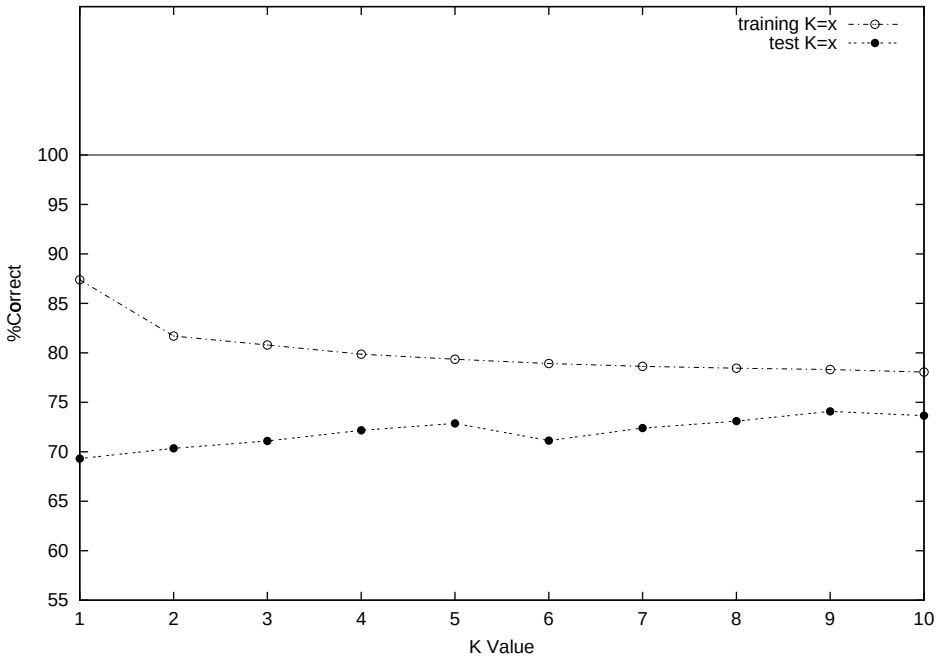

Figure 5.23: K Value Variation (diabetes)

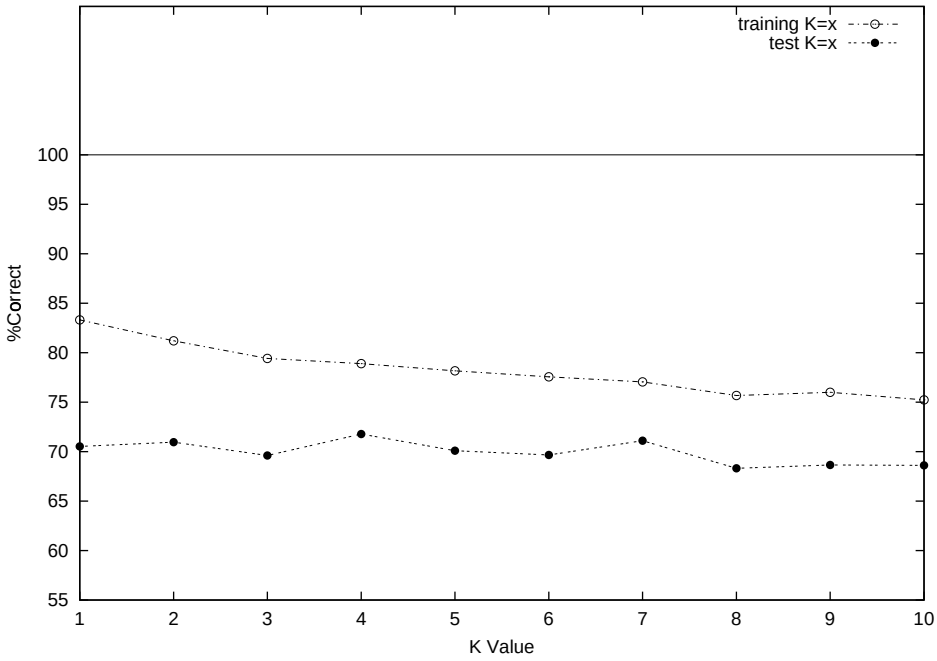

Figure 5.24: K Value Variation (sonar)

5.8 Comparison of Results with Other Classifiers

This section provides a brief comparison of the results obtained on the four data sets to other results reported in the literature. First, however, table 5.1 displays the parameter settings for the most accurate classifiers presented in sections 5.2 through 5.7.

Table 5.1: Most Accurate Classifiers

Data Set	Seed	Res.	Stim.	Mut.	ATS	K	Training	Test
Iris	1	200	0.9	0.1	0.2	7	95.83%	96.67%
Ionosphere	1	500	0.8	0.1	0.2	3	87.83%	94.92%
Diabetes	1	200	0.9	0.1	0.2	9	74.08%	74.09%
Sonar	100	200	0.9	0.1	0.2	1	95.34%	83.99%

5.8.1 Iris Data Set

As previously mentioned, Fisher's iris data set is one of the most well-known in the machine learning/pattern recognition literature. A brief sampling of these results are presented here in table 5.2 on page 67. This table was adapted from Duch (2000b). It should be noted that the data division and methodology is not fully specified for all of the methods in table 5.2. As previously mentioned, the AIRS experiments for the Iris data set were conducted in a five-fold cross validation manner averaged over three runs. With this data division, each test set consisted of 30 data items. Therefore, an accuracy of approximately 96.7% for AIRS is a misclassification of on average only one of the 30 test data items.

Table 5.2: Comparison of Iris Results

Method	Test Set Accuracy
Grobian (rough)	100%
SSV	98.0%
C-MLP2LN	98.0%
PVM 2 rules	98.0%
PVM 1 rule	97.3%
AIRS	**96.7%**
FuNe-I	96.7%
NEFCLASS	96.7%
CART (dec. tree)	96.0%
FUNN	95.7%

5.8.2 Ionosphere Data Set

Table 5.3 on page 68 adapted from Duch (2000a) provides a comparative table of various classifiers on the Ionosphere data set. As with the comparative results for the Iris data set, these results show that the AIRS algorithm performs comparatively well in relation to other classifiers. The accuracy rate of 94.9% obtained by the AIRS algorithm on this test set represents correctly classifying approximately 143 out of the 151 data items in the test set. The best classifier reported here correctly classifies approximately 149 of the 151 items in the test set. Compared to the somewhat akin sub-symbolic approach of multi-level perceptrons, the AIRS algorithm performs only slightly worse than the MLP, which correctly classifies approximately 145 of the 151 items in the test set.

Table 5.3: Comparison of Ionosphere Results

Method	Test Set Accuracy
3-NN + simplex	98.7%
3-NN	96.7%
IB3	96.7%
MLP+BP	96.0%
AIRS	**94.9%**
C4.5	94.9%
RIAC	94.6%
C4(no windowing)	94.0%
SVM	93.2%
Non-linear perceptron	92.0%
FSM + rotation	92.8%
1-NN	92.1%
DB-CART	91.3%
Linear perceptron	90.7%
OC1 DT	89.5%
CART	88.9%
GTO DT	86.0%

5.8.3 Pima Indians Diabetes Data Set

Table 5.4 on page 69 adapted from Duch (2000a) presents a comparison of test set accuracy results on the Diabetes data set. In general, this data set appears to be a much more difficult domain with the highest accuracy of only 77.7% (correct

classification of approximately 60 of the 77 items in the average test set). While there are several classifiers that outperformed AIRS on this data set, the algorithm, nevertheless, performed on par with other established classifiers. AIRS correctly classified approximately 57 of the 77 items in the average test set. One point in particular to note was the performance of two of the kNN classifiers on this data set. Both examples of this type of classifier that outperformed AIRS on this data set did so with fairly high k values. This seems to provide more evidence for the trend observed in section 5.7 where it appeared that the higher the k number the better AIRS performed on the Diabetes data set.

Table 5.4: Comparison of Diabetes Results

Method	Test Set Accuracy
Logdisc	77.7%
IncNet	77.6%
DIPOL92	77.6%
Linear Discr. Anal.	77.5-77.2%
SMART	76.8%
GTO DT (5xCV)	76.8%
ASI	76.6%
Fisher discr. analysis	76.5%
MLP+BP	76.4%
LVQ	75.8%
LFC	75.8%
RBF	75.7%
NB	75.5-73.8%

Table 5.4: Comparison of Diabetes Results *cont.*

Method	Test Set Accuracy
kNN, k=22, Manh	75.5%
MML	75.5%
SNB	75.4%
BP	75.2%
kNN, k=18, Euclid	75.2%
CART	74.7%
DB-CART	74.4%
ASR	74.3%
AIRS	**74.1%**
C4.5	73.0%
Bayes	72.2%
C4.5 (5xCV)	72.0%
Kohonen	72.7%
kNN	71.9%
ID3	71.7%
IB3	71.7%
IB1	70.4%
C4.5 rules	67.0%
OCN2	65.1%
Default	65.1%
QDA	59.5%

5.8.4 Sonar Data Set

Finally, table 5.5 adapted from Duch (2000a) provides a comparison of various classifiers on the sonar data set. The best classifier correctly classified approximately 15 out of the 16 test data items, whereas AIRS correctly classified approximately 13 out of this set of 16. Again, we find that AIRS performs comparatively well, while still not providing the most accurate classifier. One interesting note from this and the previous two tables is that AIRS' performance is roughly equivalent to the multi-layered perceptrons using the back-propagation algorithm. However, the MLP seems, in general, to slightly outperform AIRS.

Table 5.5: Comparison of Sonar Results

Method	Test Set Accuracy
TAP MFT Bayesian	92.3%
Naive MFT Bayesian	90.4%
SVM	90.4%
Best 2-layer MLP+BP, 12 hidden	90.4%
MLP+BP, 12 hidden	84.7%
MLP+BP, 24 hidden	84.5%
1-NN, Manhattan	84.2%
AIRS	**84.0%**
MLP+BP, 6 hidden	83.5%
FSM - methodology ?	83.6%
1-NN Euclidean	82.2%
DB-CART, 10xCV	81.8%
CART, 10xCV	67.9%

CHAPTER VI

CONCLUSION

6.1 Summary

This thesis has presented a new supervised learning paradigm based on observations of natural immune systems and previous work in artificial immune systems. Several key components from these immune systems were employed in the development of this paradigm. One of these components is the abstraction of B cells as a means of data representation. The reactions between an invading pathogen's antigen and a B cell's antibody were simulated through the use of Euclidean distance. This follows directly from the work of Timmis and Neal (2000), Timmis, Neal and Hunt (2000), Timmis (2001), and Knight and Timmis (2001). The concept of artificial recognition balls to represent exact clones in the search space coupled with competition for system-wide resources was also adopted from this work by Timmis and his colleagues.

Fundamental to the work presented here is the development of memory cells. Similar in some ways to the use of memory cells presented in de Castro and Von Zuben (2001a) and de Castro and Von Zuben (2001b), memory cells provide the primary mechanism for pattern recognition in resource limited artificial immune classifiers. The development of these memory cells is closely linked to the ideas of clonal selection and affinity maturation seen in many of the surveyed works on AIS.

The early work on artificial immune networks presented in Timmis (2001) also provided inspiration for this work. While the representation of idiotypic immune

networks is not employed in resource limited artificial immune classifiers, the concept of inter-cell affinity is highly important. Inter-cell affinity plays a key role in the data reduction and memory retention criteria of the system and, thus, fosters the generalization capabilities that are important to any classifier.

The contribution of this thesis to the fields of Artificial Immune Systems and Machine Learning is the introduction of a supervised learning method employing immune system principles. While many of the components used for the development of resource limited artificial immune classifiers had been previously explored, none have been exploited for supervised learning. Supervised learning was introduced in several ways. First, the stimulation of a system cell in reaction to a training antigen is based not only on its affinity to the antigen but also on the cell's classification in comparison to the antigen's classification. Second, resources are allocated to cells based on classification and stimulation level with more resources being allocated to cells of the same class as the given antigen. Third, a training stopping criterion based both on stimulation level and class is employed to ensure that high-quality classification cells are retained in the system. Fourth, memory cell identification, retention, and replacement is based primarily on the classification of the memory cell. Finally, classification of previously unseen data items is based on the reactions and classifications of the evolved set of memory cells.

This thesis has provided a detailed discussion of a resource limited artificial immune classification algorithm, named AIRS for Artificial Immune Recognition System. The fundamental mechanisms of the AIRS algorithm were presented. Experimental results of the AIRS algorithm on simulated data sets were explored to demonstrate the primary learning capabilities of the system. AIRS exhibited the ability to evolve a representation of the classes in the training data set and then use

this representation to successfully classify previously unseen data items to a high degree of accuracy. The various learning parameters of the AIRS algorithm were investigated through the use of real-world machine learning benchmark data sets. Finally, a comparison of the AIRS algorithm on these benchmark data sets to more established classifiers was presented. This comparison showed that the performance of the AIRS algorithm is comparable, and in some cases superior, to the performance of other highly-regarded supervised learning techniques for these benchmarks.

6.2 Future Work

The exploration of resource limited artificial immune classifiers presented here has by no means been exhaustive. This work has pointed to several areas that should prove fruitful for future investigations of this new learning paradigm. This section discusses a few of these areas in more detail.

In the AIRS algorithm, the current method for memory cell replacement is heavily biased in two areas. First, the metric for determining the threshold for memory cell replacement is based on the average affinity value among all of the training cells. Second, AIRS has a bias toward the most recently encountered antigen. While this second bias is somewhat true to nature, it is potentially detrimental for developing classifier cells. There are several modifications to AIRS that could be undertaken to investigate the effects of these biases. One potential modification would be determining the affinity threshold based on the average affinity of training items within the same class rather than the broad method of using all of the training items. This could possibly lead to a more refined representation of each class in the data set. Another modification, in response to the second bias, could be a presentation of the training data to the system a second time; however, on the

second pass only the memory cells would be involved. The idea is to keep track of the developed memory cells' responses to the training data. If a memory cell is never the closest, or most stimulated, cell to any of the training data items, then this cell should be removed from the set of memory cells. This modification would potentially reduce the number of outlier memory cells that were seen in section 4.3. However, one potential pitfall for this modification could be the development of a set of memory cells that are overly specific to the training data set and thus have lost the capacity to generalize to previously unseen data items.

Another area of future investigation is the method of resource allocation. Currently, resources are allocated in a somewhat naïve manner with one half of the available resource being designated for the ARBs of the same class as the current antigen and the remaining half being evenly divided among the ARBs of the remaining classes. This method could be refined to account for the known, or discovered, proportion of classes naturally occurring in the data set. With this method, the system-wide resources would be divided among the different class ARBs proportional to the class distribution in the training set. It is unclear what effect this would have on the developed memory cell set, but it could potentially lead to greater accuracy.

There is also the matter of mutations and mutation rate. AIRS employs a uniform mutation rate which is blind to the quality of the ARB or memory cell to be mutated. Additionally, the mutation mechanism in AIRS is based on random changes within a predefined range. Both of these features could be modified in future studies. In de Castro and Von Zuben (2001b), the authors employ a method of mutation that is dependent upon the quality of the cell to be mutated. That is, for highly stimulated cells the mutation rate is decreased and for less stimulated cells the mutation rate

is increased. In this way, only small changes are made to cells that are known to be good recognizers. This allows for the exploration of the search area immediately surrounding the high quality cells. On the other hand, for lower quality cells, the higher mutation rate allows for greater leaps in the search space to facilitate moving away from less productive areas. The actual mutation mechanism warrants further investigation as well. Rather than purely random mutations, mutations of features to a discrete set of values could be employed. This could be extremely useful when exploring data sets that consist of features that are something other than continuous values.

The investigation of resource limited artificial immune classifiers on non-continuous featured data sets also warrants greater attention. The current implementation makes this investigation difficult; however, the actual formalism should be easily adaptable. In particular, methods of using both nominal and binary-valued data sets need to be explored to allow for greater applicability of this learning paradigm.

This leads naturally to the need for investigation of stimulation and affinity functions. AIRS relies heavily on Euclidean distance as a metric for both stimulation and affinity. While this has proven to be adequate for the experiments presented here, Euclidean distance would probably not be as useful for non-continuous featured data sets. The methodology presented in this thesis should allow for the exploration of other functions, such as Hamming distance for binary valued data sets or string matching for discrete data sets. Such an investigation will be essential as different types of data sets are explored.

Finally, the addition of parallel or distributed computing techniques to the current implementation could result in extended benefits as well. Cantú-Paz (1998)

presents a survey of methods used for parallelizing genetic algorithms. While there are many differences between AIS and GAs, both rely on evolutionary mechanisms to introduce diversity into the system. Some of the mechanisms discussed in Cantú-Paz's article should prove useful in investigating the effects of parallelizing AIRS.

REFERENCES

Blake, C., and C. Merz. 1998. *UCI repository of machine learning databases.* http://www.ics.uci.edu/∼mlearn/MLRepository.html (Accessed 09 July 2001).

Cantú-Paz, E. 1998. A survey of parallel genetic algorithms. *Calculators Parallels* 10(2):141–171.

Carter, J. H. 2000. The immune system as a model for pattern recognition and classification. *Journal of the American Medical Informatics Association* 7(1):28–41.

Dasgupta, D. 1997. Artificial neural networks and artificial immune systems: similarities and differences. In *Proceedings of IEEE international conference on systems, man, and cybernetics*, Volume 1, 873–878. IEEE Press.

Dasgupta, D, ed. 1998a. *Artificial immune systems and their applications.* Berlin: Springer.

Dasgupta, D. 1998b. An overview of artificial immune systems and their applications. In *Artificial Immune Systems and Their Applications*, edited by D. Dasgupta, 3–21. Berlin: Springer.

Dasgupta, D. 1998c. Preface. In *Artificial Immune Systems and Their Applications*, edited by D. Dasgupta, v–xi. Berlin: Springer.

Dasgupta, D. 1999. Immunity-based intrusion detection system: A general framework. In *Proceedings of the 22nd national information systems security conference held in Arlington, Virginia, October 18-21, 1999*, 147–159.

Dasgupta, D., and S. Forrest. 1998. An anomaly detection algorithm inspired by the immune system. In *Artificial Immune Systems and Their Applications*, edited by D. Dasgupta, 262–277. Berlin: Springer.

de Castro, L. N., and F. J. Von Zuben. 2001a. *ainet: An artificial immune network for data analysis.*

ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/lnunes/DMHA.pdf (Accessed 21 Aug 2001).

de Castro, L. N., and F. J. Von Zuben. 2001b. *Learning and optimization using the clonal selction principle.* ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/lnunes/ieee_tec01.pdf (Accessed 21 Aug 2001).

Deaton, R., M. Garzon, J. A. Rose, R. C. Murphy, S. E. Stephens, and D. R. Franceschetti. 1997. A dna based artificial immune system for self-nonself discrimination. In *Proceedings of IEEE international conference on systems, man, and cybernetics*, Volume 1, 862–866. IEEE Press.

D'haeseleer, P., S. Forrest, and P. Helman. 1996. An immunological approach to change detection: Algorithms analysis and implications. In *Proceedings of the IEEE computer society symposium on research in security and privacy held in Oakland, California, May 6-8, 1996*, 110–119. IEEE Computer Society Press.

Duch, W. 2000a. *Datasets used for classification: Comparison of results.* http://www.phys.uni.torun.pl/kmk/projects/datasets.html (Accessed 20 Aug 2001).

Duch, W. 2000b. *Logical rules extracted from data.* http://www.phys.uni.torun.pl/kmk/projects/rules.html (Accessed 20 Aug 2001).

Farmer, J. D., N. H. Packard, and A. S. Perelson. 1986. The immune system, adaptation and machine learning. *Physica* 22D:187–204.

Forrest, S., S. A. Hofmeyer, A. Somayaji, and T. A. Longstaff. 1996. A sense of self for unix processes. In *Proceedings of the IEEE computer society symposium on research in security and privacy held in Oakland, California, May 6-8, 1996*, 120–128. IEEE Computer Society Press.

Forrest, S., S. A. Hofmeyr, and A. Somayaji. 1997. Computer immunology. *Communications of the ACM* 40(10):88–96.

Forrest, S., A. S. Perelson, L. Allen, and R. Cherukuri. 1994. Self-nonself discrimination in a computer. In *Proceedings of the IEEE computer society symposium on research in security and privacy held in Oakland, California, May 16-18, 1994*, 202–212. IEEE Computer Society Press.

Hofmeyr, S. A., and S. Forrest. 1999. Immunity by design: An artificial immune system. In *Proceedings of the genetic and evolutionary computation conference (GECCO) held in Orlando, Florida, July 13-17, 1999*, 1289–1296. Morgan-Kaufmann.

Hofmeyr, S. A., S. Forrest, and A. Somayaji. 1998. Intrusion detection using sequences of system calls. http://www.cs.unm.edu/immsec/publications/ids.ps (Accessed 25 Aug 2000).

Hunt, J., J. Timmis, D. Cooke, M. Neal, and C. King. 1998. Jisys: The development of an artificial immune system for real world applications. In *Artificial Immune Systems and Their Applications*, edited by D. Dasgupta, 157–186. Berlin: Springer.

Hunt, J. E., and D. E. Cooke. 1996. Learning using an artificial immune system. *Journal of Network and Computer Applications* 19:189–212.

Jerne, N. K. 1974. Towards a network theory of the immune system. *Annals of Immunology (Inst. Pasteur)* 125C:373–389.

Jun, J.-H., D.-W. Lee, and K.-B. Sim. 1999. Realization of cooperative swarm behavior in deistributed autonomous robotic systems using artificial immune system. In *Proceedings of IEEE international conference on systems, man, and cybernetics, held in Tokyo, Japan, November, 1999*, Volume 6, 614–619.

Knight, T., and J. Timmis. 2001. *Assessing the performance of the resource limited artificial immune system AINE.* Computing Laboratory, University of Kent at Canterbury. Technical Report 3-01.

Perelson, A. 1989. Immune network theory. *Immunological Review* 110:5–36.

Timmis, J. 2001. Artificial immune systems: A novel data analysis technique inspired by the immune network theory. Ph. D. thesis, University of Wales, Aberystwyth.

Timmis, J., and M. Neal. 2000. Investigating the evolution and stability of a resource limited artificial immune system. In *Proceedings of gentic and evolutionay computtion conference (GECCO) 2000, Workshop Program, held in Las Vegas, Nevada, July 2000*, edited by A. S. Wu, 40–41. AAAI Press.

Timmis, J., M. Neal, and J. Hunt. 1999. Data analysis using artificial

immune systems, cluster analysis and kohonen networks: Some comparisons. In *Proceedings of IEEE international conference on systems, man, and cybernetics held in Tokyo, Japan, November, 1999*, Volume 3, 922–927.

Timmis, J., M. Neal, and J. Hunt. 2000. An artificial immune system for data analysis. *Biosystems* 55(1/3):143–150.

Watanabe, Y., A. Ishiguro, and Y. Uchikawa. 1998. Decentralized behavior arbitration mechanism for autonomous mobile robot using immune network. In *Artificial Immune Systems and Their Applications*, edited by D. Dasgupta, 187–209. Berlin: Springer.