A COMPARISON OF GENETIC ALGORITHMS AND OTHER MACHINE LEARNING SYSTEMS ON A COMPLEX CLASSIFICATION TASK FROM COMMON DISEASE RESEARCH

by
Clare Bates Congdon

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in The University of Michigan
1995

Doctoral Committee:

Professor John H. Holland, co-chair
Associate Professor John E. Laird, co-chair
Professor Keki Irani
Assistant Research Scientist Sharon L. Reilly

ABSTRACT

A COMPARISON OF GENETIC ALGORITHMS AND OTHER MACHINE LEARNING SYSTEMS ON A COMPLEX CLASSIFICATION TASK FROM COMMON DISEASE RESEARCH

by

Clare Bates Congdon

Co-Chairs: John H. Holland, John E. Laird

The thesis project is an investigation of some well-known machine learning systems and evaluates their utility when applied to a classification task from the field of human genetics. This common-disease research task, an inquiry into genetic and biochemical factors and their association with a family history of coronary artery disease (CAD), is more complex than many pursued in machine learning research, due to interactions and the inherent noise in the dataset. The task also differs from most pursued in machine learning research because there is a desire to explain the dataset with a small number of rules, even at the expense of accuracy, so that they will be more accessible to medical researchers who are unaccustomed to dealing with disjunctive explanations of data. Furthermore, there is assymetry in the task in that good explanations of the positive examples is of more importance than good explanations of the negative examples.

The primary machine learning approach investigated in this research is genetic algorithms (GA's); decision trees, Autoclass, and Cobweb are also included. The GA performed the best in terms of descriptive ability with the common-disease research task, although decision trees also demonstrated certain strengths. Autoclass and Cobweb were recognized from the onset as being inappropriate for the needs of common-disease researchers (because both systems are unsupervised learners that create probabilistic structures), but were included for their interest in the machine learning community; these systems did not perform as well as GA's and decision trees in terms of their ability to describe the data. In terms of predictive accuracy, all systems performed poorly, and the differences between any two of the three best systems is not significant. When positive and negative examples are considered separately, the GA does significantly better than the other systems in predicting positive examples and significantly worse in predicting negative examples.

The thesis illustrates that the investigation of "real" problems from researchers in other fields can lead machine learning researchers to challenge their systems in ways they may not otherwise have considered, and may lead these researchers to a symbiotic relationship that benefits multiple research communities.

To the memory of my mother

ACKNOWLEDGEMENTS

There are many people who have contributed to the ideas presented in this thesis project, but first and foremost, I thank Sharon Lee Reilly Kardia, without whom this thesis most certainly could not have existed. Sharon and I met at the Santa Fe Institute's Complex Systems Summer School in 1991. Although we were both at the University of Michigan (I as a graduate student in Computer Science, she as a postdoc in Human Genetics), we had to travel to Santa Fe to actually meet. It was in the nurturing and intellectually stimulating environment of the summer school that Sharon and I first began discussing possible applications of genetic algorithms to various research questions in human genetics. This led to her to convince Dr. Charles Sing in the Human Genetics department to hire me as a research assistant. I'm not sure that Charlie knew what he was getting into when he hired me; he and I have had our share of troubles finding a common language in which to discuss this research. And yet, Charlie's active participation in the initial development of the GA used in this research was invaluable. I am also indebted to Sharon and Charlie for access to a research problem that I find fascinating, even in the final throes of writing the dissertation, when I am supposed to be sick of it. There are so many facets to this research, so many different questions that could be addressed, that I feel I could work with this data for another dozen years and still not have exhausted its possibilities.

While Sharon has contributed constant input on this work, and in particular, the development of the GA model, a number of other "grown ups" have helped me along the way. Melanie Mitchell served as my thesis advisor initially on this project, until she departed Michigan for the Santa Fe Institute. John Laird has been my advisor in one capacity or another (both official and unofficial) throughout my graduate career, and is so good natured that he is willing to co-chair the committee of a thesis project that has nothing to do with his primary research interest, Soar. John's broad knowledge of machine learning research helped me to organize this document and to focus on relevant issues. John Holland generously provided time for meetings and email conversations although he generally does not advise students these days. His insightful comments on this research were most appreciated. Keki Irani provided assistance in particular with decision trees. I consider myself very fortunate that all of my committee members are not only scholarly role models for me, but also, terribly nice people and a pleasure to work with.

Others who have helped in specific ways include Rick Riolo, who cheerfully responded to multiple requests for relevant GA references, and who helped Sharon and me to know which approaches to consider in adapting the GA for our task. Usama Fayyad shared his time and opinions on this task in general and on decision trees in particular. Jim Wogulis patiently and thoroughly answered my questions about related work in machine learning. A number of people whom I met only by email conversations have also been patient and helpful.

Kevin Thompson at NASA Ames helped get me started with Cobweb and answered several questions about that system and about classification in general. Will Taylor at NASA Ames assisted with Autoclass; Wray Buntine at NASA Ames answered my questions about IND (which did not end up being used in the thesis). Larry Hunter, Jeff Schlimmer, and Michael Pazzani are also helpful people who had never met me but who were willing to answer my questions anyway. Institutions to be thanked include NASA's COSMIC software distribution center, which made beta-test versions of Autoclass and Cobweb available to me, and JPL (through Usama), which provided me with TreeMaker. The ability to use these systems without having to program them myself made this a feasible thesis project. The University of California at Irvine (UCI) repository of machine learning databases has also proved to be a handy resource.

In addition to professional contacts, I would also like to thank a number of friends, including present and former officemates, whose contributions in terms of casual conversations (both social and academic) cannot be overlooked. This includes the Reverend Doctor Peter Hastings (aka "the Bad Peter"), Michael Hucka, Renee Rottner, Craig Miller, Miriam Allersma, Eric Chown, Doug Pearson, Helen Kim, Scott Huffman, Tom Montgomery, Neal Rothleder, Ulrich Raschke, Bob Wray, Jim Herbsleb, and probably at least a dozen other people who will be mildly saddened to have been inadvertently omitted from this list. There has been much silliness in grad school, including the forming of the RDCA (red delicious are crappy apples) and the secret knock for keeping you-know-who out of our rooms at ML-90, and I thank my friends for contributing to this levity.

Finally, I would like to thank my family: Dad, Richard, Alice, and John, and all their partners and children, for encouraging me in my pursuit of a PhD, for letting me know they are proud of me, and for gently teasing me into finally finishing up. And most of all, I would like to thank my husband, Randy Jones, who does fall into the category of a friend who provided helpful conversation and silly diversions, but mostly makes the world a wonderful, magical place to live in. Finally, I would like to thank Benjamin Congdon-Jones for arriving on the planet, for timing his arrival so as not to interfere with my defense, and for bringing so much joy to our lives.

TABLE OF CONTENTS

LIST OF TABLES

Table

LIST OF FIGURES

Figure

LIST OF APPENDICES

Appendix

CHAPTER 1

INTRODUCTION

Coronary Artery Disease (CAD) is a major killer in our society: in 1989, 257,524 males and 240,497 females died of CAD in the United States [2]. Like other complex common diseases, such as hypertension, diabetes, and cancer, CAD is difficult to study because the disease "aggregates" in families, but does not "segregate" as if determined by a single Mendelian gene [93, 94]. That is, the disease tends to cluster in families, but the disease is associated with a multifactorial etiology, rather than a single-gene etiology. It is believed that in most cases, an individual's risk of disease is the consequence of the complex interactions (i.e. the nonadditive relationships) of many genetic and environmental factors [101]. Furthermore, different multiple-factor etiologies may be needed to explain the risk associated with different subgroups of individuals.

There exists an increasing inconsistency between what is known about the underlying biology of CAD and the research methods used to study the disease in the population at large [94, 101]. Studies of the risk factors associated with CAD and other complex common diseases have relied almost exclusively on traditional statistical modeling procedures, which do not take into account the possibility of interactions between risk factors and risk of disease, nor the possibility of multiple etiologies. New methods for studying the risk-factor associations with CAD are needed that yield more realistic models by allowing for non-additive relationships between risk factors and CAD risk, and by allowing for multiple etiologies [94, 96, 95]. The main research question from this perspective is: how many combinations of risk factors are needed to explain the frequency of disease in the population, and what are these risk-factor combinations (that is, what are the risk factors and what are their values).

The field of machine learning has developed approaches that may help study the risk-factor combinations associated with common diseases. In machine learning, models that allow for multiple etiologies are commonplace (in machine-learning parlance, this corresponds to multiple rules); a few machine learning methods allow for models with complex interactions between the risk factors and CAD risk (although several prohibit this). Furthermore, the classification task defined by this common-disease research poses worthwhile challenges for the field of machine learning. This task is different from most explored in the machine learning literature for two reasons: The measure of the quality of a rule set is not simple prediction accuracy (the percent of correct classifications on examples not in the original "training" dataset), and the interactions between the attributes and risk of CAD make classification particularly difficult.

The primary machine learning question addressed in this thesis is:

- What is the "best" approach for a complex classification problem such as the one presented here?

This raises two important subproblems:

- How do we measure the quality of a rule set for this task?
- How can we characterize the complexity of a classification task?

A comparison of the performance of different systems on a difficult task is an interesting project in itself, and not undertaken frequently enough in machine learning literature. However, the subproblems presented here are in many ways more profound. The field of machine learning rarely considers that prediction accuracy is not always an appropriate measure of quality, and rarely considers the complexity of the tasks used to investigate the strengths and weaknesses of systems. Many tasks commonly used in machine learning research to evaluate learning algorithms are relatively simple, but few researchers pause to evaluate how difficult their classification task is before they proceed with their research.

The research presented here challenges simple prediction accuracy as a metric in two primary ways:

1. A smaller rule set may be preferable to a larger rule set, even at the expense of prediction accuracy.
2. The classification errors are not necessarily symmetric: classifying a negative example as positive may not be as serious an error as classifying a positive example as negative.

In addition to challenging simple prediction accuracy, this thesis research illustrates the need for considering the complexity of classification tasks in machine learning.

The common-disease research task presented here poses interesting challenges for machine learning systems and challenges some oft-held assumptions in machine learning. Unlike the artificial problems often used to compare machine learning systems (datasets created to test specific features of a system), this problem has been defined by researchers in another field. That is, the constraints of the problem come from the needs of common-disease researchers, rather than researchers in machine learning. From the perspective of machine learning, the task provides an "objective" domain, not chosen or designed to highlight or test the features of any particular machine learning system, and across which different systems may be compared.

This study of risk-factor associations raises issues not generally considered in the development of machine learning systems, and for this reason, not all machine learning systems will be suitable for application to this problem. The primary approach investigated in this thesis research is genetic algorithms (GA's). They have demonstrated success on complex problems (see, for example, [77, 23, 22, 106]) and are more flexible than most machine learning approaches in that they do not impose rigid assumptions about the search space (for example, many machine learning systems assume that the attributes are independent, and many do not attempt to model interactions between attributes and the class). However, it is not clear from the onset that GA's will outperform other systems. It may well be that an approach such as decision trees is capable of finding better classifications even though it makes no explicit attempt to model attributes that interact. Furthermore, the flexibility of GA's does not come without a "cost": A considerable amount of time in this research was spent on experiments with varying parameters, in search of good settings for the GA with this particular task. This initial exploration of good parameter settings was not required for the other machine learning approaches investigated in this thesis. The other approaches investigated in this thesis include: Decision trees, because of their widespread use in machine learning research and their ease of implementation; and Autoclass and Cobweb, because they are well known and well established classification systems. That is, they have been in use for a number of years by a number of researchers.

Contributions of the thesis

This thesis benefits both the machine learning and the common-disease research communities:

Contributions to the machine learning community:

The use of real problems can spur the field into addressing hard questions that can otherwise be left as "future work". The research problem from the study of common diseases provides a complex dataset that cannot be easily classified by any machine learning system and is therefore a challenging domain for comparison. The task is also a compelling example of one in which simple prediction accuracy is not a sufficient metric for comparing systems, and therefore demands that other criteria be incorporated into the metrics used for comparison.

Specifically, the contributions to the machine learning community include:

- The thesis provides a comparison of several well-known machine learning systems on a real problem which is more complex than those commonly undertaken in machine learning research. Comparisons of different machine learning systems are not often undertaken, and certainly not with a problem as complex as this one.

- The thesis argues that prediction accuracy is not a "gold standard" applicable to all classification tasks, introduces additional metrics to be considered, and provides a mechanism by which multiple competing criteria may be combined into a single metric.

- The thesis proposes a set of criteria for evaluating the complexity of a dataset, and compares the complexity of the dataset used here to that of several datasets commonly used in machine learning research.

- Specific to the genetic algorithms community, the thesis provides an in-depth study of the robustness and usefulness of genetic algorithms on a complex problem, and includes investigations of some relatively new approaches (e.g. immigration and fitness sharing).

Contributions to the common-disease research community

The thesis provides an exploration of a new paradigm for research in common complex diseases, and the development of analytic tools that may be useful in pursuing this new paradigm. The tools investigated here are different from those commonly used to study the genetics of common diseases because they allow for (but do not force) *more than one model* for different subsets of the dataset, and because they allow for complex interactions between attributes and disease risk.

This thesis encourages common-disease researchers to consider the merits of the proposed paradigm shift and to envision some of the tools that may be useful in pursuing this new paradigm (including tools that have already been developed in the field of machine learning).

Overview of thesis

The second chapter of this thesis discusses the common-disease research question in more detail, providing information on the questions driving the common-disease research agenda and on the dataset itself. The third chapter discusses the problem from the perspective of machine learning, recasting it as a search for a good rule set and challenging the prevalent metrics in the machine learning community for comparing rule sets. The fourth chapter presents relevant metrics for evaluating and comparing rule sets. The fifth chapter discusses the genetic algorithms implementation and the sixth discusses the decision trees implementation. In the seventh chapter, a number of other approaches that were implemented are presented; these are not given as much attention as GA's or decision trees either because their implementation was straightforward (e.g., hillclimbing) or because assumptions of the system were counter to the constraints of the problem (e.g., Autoclass' requirement that the attributes be independent). The eighth chapter summarizes the performances of the different systems, presents some metrics for evaluating the complexity of a dataset, and applies these metrics to the common-disease research task presented here as well as to some datasets commonly used in machine learning research. Finally, the ninth chapter summarizes the thesis, reiterating results and projecting future research projects.

A note on terminology

In this thesis, an effort has been made to define terms and to avoid using multiple terms to express a single concept. The use of precise terminology is especially important when spanning two fields: Some terms will be unfamiliar to common-disease researchers, while others will be unfamiliar to computer scientists. What's more difficult is that the same term may be familiar to both computer scientists and common-disease researchers, but may mean something different in the context of each field. To avoid confusion, there are many definitions that must be presented in this thesis. Not all of the definitions will be needed by all readers, and they tend to break the flow of an idea when presented in the main text; however, flipping back and forth to a glossary seems even more tedious to me. I have therefore attempted a compromise: terminology that may be unfamiliar to some readers is set off in the text with an underline; the definitions appear at the bottom of the page, as footnotes. For example, terminology. A complete glossary appears in Appendix G, which begins on page 157.

terminology: The terms or system of terms used in a specific science, art, etc.; nomenclature [107].

CHAPTER 2

THE NEED FOR A NEW PARADIGM IN THE STUDY OF COMMON DISEASES

In Western industrialized societies, diseases of the heart rank as the number one cause of morbidity and mortality in the adult population [49]. Identification of individuals at risk for Coronary Artery Disease (CAD) is a major research problem: Although there is evidence that the disease begins early in life [8, 30, 54, 76, 111], symptoms do not generally appear until the fifth or sixth decades [93], and many individuals with CAD go undetected until they suffer a myocardial infarction, more commonly known as a heart attack. Excluding angina pectoris, sudden death is the first and only symptom of CAD in 18% of all heart attacks in males and 24% in females [61]. In addition to the physical and emotional toll this disease takes on society, CAD is estimated to be associated with \$8 billion in direct health care costs and with \$60 billion in economic costs per year in the United States [12]. Kannel and Schatzkin observe that few other diseases can kill in a matter of minutes, and that therefore, CAD is a disease that demands a preventive approach [61].

The etiology of CAD is difficult to study because in the majority of cases, an individual's risk of developing CAD is the consequence of complex interactions among the effects of *many* factors, both genetic and environmental, and cannot generally be ascribed to a single cause. The biological complexity of CAD will be discussed in Section 2.1.

The traditional paradigm used to study the etiology of CAD is based on the model developed

morbidity: The rate of disease or proportion of diseased persons in a given locality, nation, etc. [107].

mortality: The proportion of deaths to the population of a region, nation, etc.; death rate [107].

Coronary Artery Disease (CAD): Also called ischemic heart disease and coronary heart disease, this term is applied to heart ailments that are caused by narrowing (or obstruction) of the coronary arteries and therefore characterized by a decreased blood supply to the heart.

myocardial infarction: the damaging or death of an area of the heart muscle (myocardium) resulting from a reduced blood supply to that area.

angina pectoris: A condition marked by recurrent pain in the chest and left arm, caused by a sudden decrease of the blood supply to the heart muscle [107].

etiology: A theory of the causes of a particular disease.

for simple single-gene diseases, such as sickle-cell anemia. This paradigm assumes that all cases of a disease can be explained by one model and tends to overlook the possibility of interactions between attributes and disease status when developing this model. These methods are insufficient to address what is now known about the complexity of this disease. The traditional approach and the statistical methods used will be discussed in Section 2.2.

The shortcomings noted in the traditional paradigm have led some common-disease researchers to call for a paradigm shift, discussed in Section 2.3. The proposed paradigm shift is similar to those in other fields, as researchers are noting that the phenomena they seek to model are more complex than traditional approaches allow for; this interdisciplinary movement is called complexity studies [106, 73, 92, 86]. This shift to studying "complex adaptive systems" brings with it a need for new analytic tools that are suited to the new paradigm; the characteristics of these tools will be discussed in Section 2.3.

Finally, from the perspective of the new paradigm, two question may be posed, allowing for the possibility of interaction between attributes and allowing for a multiple-etiology model:

1. *What combinations* of risk factors are associated with a family history of CAD?
2. *How many combinations* are needed to describe the frequency of the family history of disease in the population?

The materials and methods used to approach these questions are presented in Section 2.4.

2.1 The Biological Complexity of CAD

Complex common diseases, such as Coronary Artery Disease (CAD), hypertension, diabetes, and cancer are difficult to study because they aggregate, but do not segregate in families [93, 94]. That is, these diseases tend to cluster in families, but none of these diseases is associated with a single Mendelian gene. The complex common diseases can be contrasted with many monogenic diseases, such as Duchenne muscular dystrophy and sickle cell anemia. Such diseases also cluster in families, but an alteration in a single gene is necessary and sufficient to cause disease.

In addition to the relatively simple monogenic model of disease inheritance, Strohman [101] defines two means of inheritance that are more complex. In a polygenic model of disease inheritance, there is a many-to-one relationship between genes and disease. Several different genes must all be present in an individual for the disease to be present. An epigenetic model of disease, according to Strohman,

Mendelian gene: following the laws of Gregor Mendel; of particular relevance here is the law of single unit characters, which states that characters (such as height, color, etc.) are inherited separately as units [107]. As used here, a Mendelian gene would imply that there is a one-to-one relationship between variation in a gene and disease status.

monogenic: A disease or other physical trait whose presence or absence is determined by a single gene.

polygenic: A disease or other physical trait whose presence or absence is determined by two or more genes, working together.

> "...may involve both single-gene and multi-gene interaction. Epigenesis implies a level of complexity beyond gene-gene interaction and extends to interaction between genes, between genes and gene products (proteins), and between all of these and environmental signals, including, of course, the individual organismal experience. But, in addition, epigenetic pathways are usually thought by developmental biologists to involve progressive states of organization, each succeeding state depending on the prior state. Epigenetic pathway, therefore, implies great complexity of interaction as well as the production of entire states of organization arising from that interaction."

The common complex diseases can be characterized as epigenetic. Furthermore, Sing and Reilly [94] note that the mapping between risk-factor interactions and risk of disease is many-to-one rather than one-to-one. For example, two individuals with the same clinical phenotype (say with respect to the onset and progression of disease) may have very different genotypes and exposures to very different environments.

2.1.1 CAD viewed as a complex adaptive system

Sing and colleagues [94, 95, 96] suggest that using a model of a complex adaptive system can help us to understand the etiology of a complex common disease; they note that there are five underlying characteristics of complex adaptive systems, which apply to CAD in the following manner:

1. *Many agents* — Up to 200 genes and over 100 other risk-factor traits have been implicated in the initiation, progression, and manifestation of CAD.
2. *A coherent network* — The biological traits that define normal biochemical and physiological processes are an interrelated system. Variation in each trait is bounded, and covariation between traits is constrained in order to maintain homeostasis in response to external perturbations.
3. *Emergent properties* — CAD is an emergent property of the network of biological traits in that there is no value of a particular trait, or values for a combination of traits, that is necessary and sufficient to predict the presence or absence of disease. Relationships *between* traits in the network, as well as the actual levels of particular traits, may be involved in determining risk of disease [3, 66].

epigenetic: A disease or other physical trait whose presence or absence is thought to be determined by multiple interacting genes, gene products, and environmental signals.

phenotype: The expressed traits of a person; their physical characteristics.

genotype: The genetic traits of a person; their genetic characteristics.

homeostasis: The tendency to maintain, or the maintenance of, normal, internal stability in an organism by coordinated responses of the organ systems that automatically compensate for environmental changes [107].

4. *Hierarchical organization* — The risk factors associated with CAD may be viewed as a hierarchy, where each level of the hierarchy is characterized by more agents than the level above it; activity at the lower levels occurs at a faster rate than at the higher levels; and effects at each level in the hierarchy exist in different scales and hence are measured very differently.
5. *Dynamic properties* — A living organism is necessarily dynamic. New cells are formed as others die off; diet and exercise, for example, change the homeostatic balance. As such, CAD does not appear "suddenly"; rather, it emerges gradually during the lifetime of an individual. (Although, often our knowledge of CAD is sudden in that the first outward symptom may be a heart attack.)

To Sing and Reilly's list, I would add the following attributes of complex adaptive systems:

5. *Importance of initial conditions* — There is evidence of a genetic component to CAD; conception might be considered the initial state of a person. Alternatively, since an individual's genes do not exist independently of an environment, the "initial state" for an individual in a study of the risk factors associated with CAD might be a specific age or age group, such as "five years old" or "18-22 years old". In terms of complex adaptive systems, the initial state is a set of conditions that restricts in some ways the endpoints that the system can achieve. For example, a human embryo will not mature into a frog.

   Note that there is considerable leeway in terms of what might be termed an initial state. For most complex adaptive systems, there are multiple states that might be considered the "initial state"; the important influence of the initial conditions is that where the system starts has a marked impact on where it can progress.
6. *Sensitivity to small perturbations* — This is sometimes refered to as the "butterfly effect" [40], taken from a complex adaptive systems model of weather patterns that notes that a butterfly flapping its wings in Japan might profoundly affect the weather in North America[1]. A more common phrase would be "the straw that broke the camel's back" to indicate a small change that has a large effect on the outcome of the system.

   In terms of a complex common disease, a seemingly small event, such as getting angry or anxious on one particular day (and having elevated blood pressure as a result of this), can tax the homeostasis mechanisms of the body, leading to a permanent physiological change that may put an individual at risk for disease.

2.1.2 Related nontraditional views of complex common diseases

There are other researchers whose work concurs with Sing and Reilly in their understanding of the complexity of CAD. One view is that the lifetime of an organism consists of an initial

[1] The "Butterfly Effect" was also fancifully illustrated in an episode of the Twilight Zone, in which time travelers went back in time and unknowingly stepped on a butterfly and then ceased to exist.

genetic structure under the influence of a particular environment. With this view, disease can be seen as the result of the homeostatic physiological network breaking down. According to Childs [19], "The reality (of disease) is the inability of one person's homeostasis, conditioned by his genotype and a lifetime of special experiences, to maintain equilibrium; neither genes nor environment 'cause' disease, it is simply that the organism is unsuited for adaptive action in one, or several environments."

Strohman [101] argues that we've evolved from a certain gene pool, which was (presumably) advantageous in the primitive environment in which it evolved, and that our genes are simply not adapted for the environment of the modern world. Strohman would call CAD a "disease of civilization" and points to evidence that such diseases were rare or nonexistent in primitive times and are rare in modern non-Western civilizations. According to Strohman, "disease is ... a result of frustrated attempts made by the organism to adapt phenotypically to a hostile environment or to some hostile set of elements for which there is not adequate response basis." Genes constrain the possible physiological responses to environmental signals, but do not directly cause a complex disease. Different genetic makeup and exposure to different environments lead to different anomalies in different individuals; there is no single pathway to disease.

The biological complexity of CAD and other complex diseases has come to be understood only in recent decades, but is generally accepted by common-disease researchers. However, the tools used to model these diseases lag far behind knowledge of the disease process itself. Sing and Reilly propose that by viewing CAD as a complex adaptive system, one is able to break away from the dominant paradigm in common-disease research so that our models of the common complex diseases resemble what is known about the complexity of these diseases. The new paradigm advocated by Sing and Reilly calls for new analytical methods to be developed. This will be discussed in Section 2.3, after a review of the traditional analytic methods.

2.2 Traditional Methods Used to Model Risk Factor Associations with Complex Diseases

The traditional statistical approaches used by genetic epidemiologists to model common diseases such as CAD assume that attributes are independent and make little or no attempt to model nonlinear effects between attributes and risk of disease. These methods often assume an additive relationship between the factors and the probability of CAD. Furthermore, and more strikingly, traditional statistical approaches assume that all people who are striken with a common disease can be explained by the same model. (In machine learning, this would be equivalent to assuming that all incidences of a disease could be explained by a single rule.) As the previous section explained, these assumptions are not appropriate, due to the incredible biological complexity of CAD.

Karlin [63] suggests that although the traditional approach follows from the paradigm established for the study of monogenic diseases (such as sickle cell anemia), it is not appropriate to the study of multifactorial diseases. While we do not want to rule out the possibility of a single-factor, single-etiology model in our study of CAD, nor do we want to be confined to such a model. The metrics and methods of the traditional approach are described below.

2.2.1 Traditional metrics

The fields of statistics and epidemiology have developed many metrics to analyze the association between risk-factor traits and risk of disease. In Chapter 4, I review some of these metrics, including sensitivity, specificity, and odds ratio, which are used to characterize the strength of the risk-factor disease association.

2.2.2 Traditional methods

The traditional method used to model a common complex disease is stepwise logistic regression (LR). This is an iterative hillclimbing model-building procedure, which looks for the "best" single model to explain the data, given specific statistical criteria. At the beginning of an LR run, the user defines a number of possible attributes to be considered in forming the model. These may be specific risk-factor traits from the dataset, or interaction terms formed by considering combinations of risk-factor traits. For example, risk-factor trait a, which has 3 degrees of freedom, may be combined with risk-factor trait b, which has 2 degrees of freedom to yeild a new attribute, $a * b$, which has 6 degrees of freedom. Only attributes that have been established by the user at the beginning of the run may be used by the LR in forming the model.

In forward regression, the model starts with a "null hypothesis" that the observed variation in disease status is due entirely to chance. With each iteration of the regression procedure, the system compares the current hypothesis to several alternative hypotheses; each of these alternative hypotheses are formed by adding one more attribute from the original list into the current model. When none of the alternative hypotheses are significantly better than the current hypothesis, the regression stops. Otherwise, the hypothesis that leads to the best improvement over the current model becomes the new current model, and the regression continues. (In backward regression, the process is similar, but the initial model assumes that all of the attributes established by the user at the beginning of the run are significant predictors, and with each iteration, an attribute is removed from the model. The alternative hypotheses in this case are the current model *minus* each of the remaining attributes.) The result of a logistic regression run is a multidimensional probability model that includes one or more attributes from the initial list.

Logistic regression is a heuristic but deterministic search method. The final explanation of the data found by this method is influenced by the list of attributes defined by the user at the beginning of the run as well as by the choice of forward or backward regression precedures. Since this method is typically used on problems with many attributes, it is not feasible for the user to include all possible interaction terms.

2.3 A New Paradigm for Studying Complex Diseases

The traditional tools and approaches used in the fields of genetics and epidemiology are incompatible with what is now known about the biology of CAD. According to Strohman [101], "the major research paradigm of modern biomedicine and disease is undermined by evidence from molecular biology itself, as well as from epidemiology." However, a paradigm shift raises the challenging task of developing analytic tools appropriate to the new paradigm. This section discusses some of the qualities desirable in the new analytic tools.

Two main criteria distinguish the paradigm shift from its predecessor[2] and should be capabilities of the new tools:

1. allow for *more than one model* for different subsets of the dataset
2. allow for complex interactions between attributes and the class

Additional considerations (which possibly apply to the old paradigms) should also be capabilities of the new tools:

3. allow for attributes that are not independent
4. minimize the error in each of the models
    (a) in describing the original data
    (b) in generalizing to unseen data (from the same population)
5. minimize the number of models needed to explain the data
6. allow for asymmetry in classification errors, that is, false positives might be more significant than false negatives, or vice versa.

A few notes on the above lists are called for because some items are not compatible with the dominant paradigm in common-disease research and therefore, may not be obvious. We want to allow for more than one model and for complex interactions because the one-model, overlook-interactions, approach is inconsistent with what is known about the underlying biology [101]. Previous one-model approaches sought to minimize the error in describing the original data set (that is, they attempted to find the one best model to "explain" the data), but it is unusual to use the separate "test set vs. training set" approach common in machine learning.

The field of machine learning has developed approaches that fit the above criteria. These approaches will be discussed in Chapter 3.

[2] There is a third feature not pursued in this research: ideally, new tools would allow for hierarchical relationships between the attributes.

2.4 The Research Question Addressed in this Thesis

Two research questions may now be posed from the perspective of the new paradigm:

1. *What combinations* of risk factors are associated with a family history of CAD?
2. *How many combinations* are needed to describe the frequency of the family history of disease in the population?

The investigation of this research question provides insight to the usefulness of the new paradigm for studying and understanding the etiology of common complex diseases. The following sections discuss the specific details surrounding this research question.

2.4.1 The Rochester Family Heart Study

As part of the Rochester Family Heart Study (RFHS) begun in 1984, 283 multi-generation pedigrees were ascertained without regard to health status through households with children enrolled in public and parochial schools in Rochester, Minnesota. There are 2004 individuals (99.6% Caucasian) distributed among these 283 pedigrees. Turner et al. [104] and Perusse et al. [81] describe the sampling methods in more detail.

The research reported here is based on 573 Caucasian individuals from the parental generation of these pedigrees. These 573 individuals were selected so that:

- they had no missing data for any of the risk factors used in this analysis,
- information about CAD status of both of their parents was available, and
- they had no clinical history of CAD (because the disease itself may affect values of risk-factor traits).

Each of the individuals in the sample were classified as either having a family history of disease (i.e., at least one affected parent) or not having a family history of disease, based on the medical records or death certificates of their parents. Of the 573 individuals in the sample, 254 have a family history of CAD.

The RFHS is different from many in that it is representative of the population of Rochester, MN. Many studies of disease recruit only sick or only healthy people for study.

2.4.2 Risk factors

The risk factors included in this study are:

Genes — We consider here three unlinked genes that code for apolipoproteins — Apo E, Apo H, and Apo AIV — which are polymorphic at the protein level. These polymorphic genes have been selected because they are candidates for influencing the

unlinked genes: When two genes are unlinked, they are found on different chromosomes; variation in one is not associated with variation in the other.

interindividual variation in lipid and apolipoprotein traits that have been associated with risk of CAD [21]. For each of these genes, there are three common allelic forms, denoted $\epsilon2$, $\epsilon3$, and $\epsilon4$ for the Apo E gene; H*1, H*2, and H*3 for the Apo H gene; and AIV*1, AIV*2, and AIV*3 for the Apo AIV gene. Kaprio et al. [62] describe how these polymorphisms were measured in the RFHS.

Intermediate traits — The following traits have been selected because they have been found to be associated with risk of CAD [21, 31, 47]: plasma levels of total cholesterol (CHOL), triglycerides (TRIG), high-density lipoprotein cholesterol (HDL), and apolipoproteins AI, AII, B, CII, CIII, and E (APO AI, APO AII, APO B, APO CII, APO CIII, and APO E, respectively), systolic blood pressure (SYS), diastolic blood pressure (DIA), and red blood cell sodium-lithium countertransport (CNT). Kaprio et al. [62] describe how each of these traits were measured. In the research presented here, each of these continuously distributed traits were transformed into three discrete classes representing the high, medium, and low tertiles of the sample distributions.

Anthropometric Factors — There are many other risk factors that have been associated with risk of CAD. In this study, we have included age, height (HT), weight (WT), waist-to-hip ratio (WHR), gender (M/F), and smoking status (SMOKE). Age, HT, WT, and WHR, are continuously distributed traits that were transformed into three discrete classes representing the high, medium, and low tertiles of the sample distribution. M/F and SMOKE are binary traits; individuals are classified as either male or female and either current smokers or current nonsmokers (denoted as YES or NO in figures and tables in this thesis).

There are three genes, 12 intermediate traits, and six anthropometric factors considered here, for a total of 21 risk factors.

2.4.3 Justification for discretization of traits

The common-disease research question at the center of this thesis, including the paradigm shift it hopes to encourage, is a very difficult problem that has not previously been approached in the literature. Although many of the risk factors considered in this study are measured on continuous scales, these have been transformed into discrete counterparts for use in the research reported here.

The research problem with discrete attributes maintains a great deal of complexity (as will be discussed in Chapter 8), and yet is easier to implement. This is seen as an important first step towards an understanding of the etiological complexity of CAD and towards evaluating the merits of the analytic approaches presented/evaluated here, and is not meant to presume that we would see exactly the same results on the continuous-valued problem as we do with the discrete-valued problem.

polymorphic at the protein level: Genes encode proteins. Thus, "polymorphic at the protein level" indicates that there is variation in the gene's DNA sequence, resulting in variation in the amino acid sequence of the protein.

Naysayers from machine learning have argued that this simplification of the task may actually make the task more *difficult* because the tertile divisions are "arbitrary" with respect to the actual biology (for example, a person with cholesterol value of n may regarded as in the middle tertile, while a person with cholesterol value of $n + 1$ may be regarded as in the highest tertile). There are also statisticians and epidemiologists who object strongly to the discretization of real-valued traits prior to the use of stastical modelling tools. To placate those who object to the discretization of traits, an appendix has been added, describing a brief experiment with the continuous-valued attributes and decision trees. This is described in Appendix D; the results are not markedly different from the version with discrete attributes.

2.4.4 Justification for studying family history of disease

Many computer scientists are baffled as to why one would study the family history of disease of an individual rather than the disease in the individual itself, though few common-disease researchers are troubled by this. Therefore, a brief discussion of why this is done is warranted here. (There are many other examples of studies that use family history as the dependent variable; for example, [72, 37, 100, 109].)

One consideration is that people who have already had heart attacks can not be included in the study. The first reason for this is that many people who have had a heart attack will have died as a result of it, and therefore our sample would consist not of people who have had heart attacks, but rather, people who have *survived* heart attacks. But the primary reason these individuals must be excluded is that a heart attack changes an individual's physiology along many of the dimensions that we are measuring. For example, after a heart attack, the cholesterol level often soars as the body attempts to repair itself. (This phenomenon is called biological feedback.) Also, people who have had heart attacks tend to make major changes to their behavior, such as by eating better and getting regular exercise; these behavioral changes will also tend to affect many of the traits we are measuring.

A longitudinal study would be ideal; individuals would be measured for risk factors at regular intervals and their causes of deaths recorded. This would allow multiple studies of people who are at high risk at many different ages. However, longitudinal studies are rarely funded because of their great expense and the length of time they tie up financial resources. (One notable exception to this is the Framingham Study [24].)

It has been shown that first-degree relatives of adults with CHD have a 2.5-7-fold increase in the risk of early coronary death compared with relatives of controls [72, 48], and that a person who has an early myocardial infarction (before 50 years of age) is likely to have a family history of myocardial infarction [27, 74]. Ten Kate et al. [103] argue that the familiar aggregation of heart disease is not entirely explained by the familial clustering of currently known coronary risk factors (perhaps because environments and anthropometric factors tend to follow the patterns of inheritance too, but are difficult to measure.) These explain some of the reasons that family history of disease is often used as a substitute for actual disease in studies of common complex diseases.

CHAPTER 3

CHALLENGES FOR MACHINE LEARNING

The common disease research question presented in Chapter 2 provides an interesting and unusual machine-learning task because it is more difficult than most that are used to compare systems in the machine learning literature and because it challenges the usual metrics used to compare performance of machine learning systems. The specific reasons for this will be discussed in detail in Section 3.2, after a discussion of the general problem of classification.

3.1 A Detour to Establish Terms

Before the machine learning issues can be elaborated upon, there is a great deal of terminology that must be clarified. This is essential because even within the field of machine learning, terminology is not always used consistently.

The risk factors described in Section 2.4.2 are generically called attributes in machine learning. There is a common blurring in the distinction between "attribute" and "attribute-value pair" (both in machine learning and in colloquial use) in that people will often call an attribute-value pair simply an attribute. One reason for this may be that the definition of what constitutes an attribute is somewhat arbitrary: we could call "cholesterol" an attribute, and use the possible values HIGH and LOW; or we could call the attribute "LOW cholesterol", and use the possible values YES and NO. Though the distinction between the two uses of "attribute" is often easily inferred from the context, we will attempt to distinguish these in the thesis, using "attribute-value pair" when appropriate.

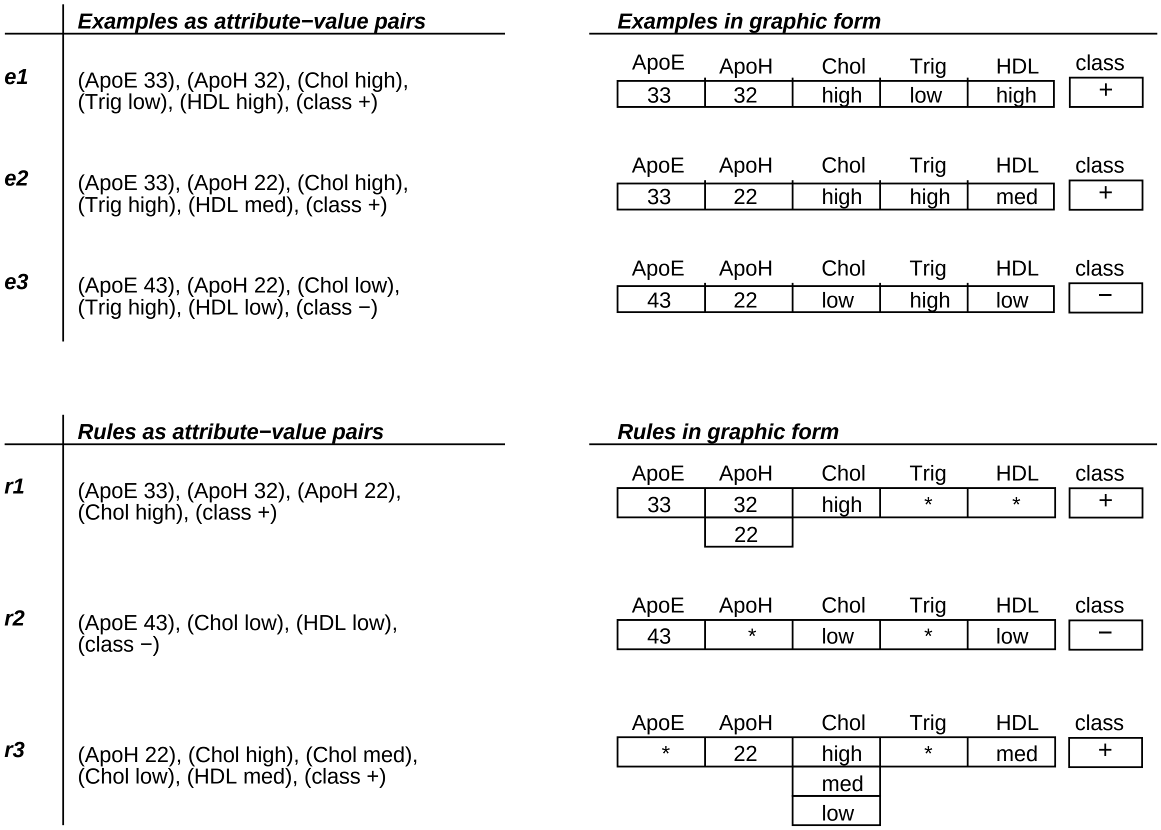
The 573 individuals in the dataset are generically referred to as examples in the context of machine learning. Each example and each rule is denoted by a set of attribute-value pairs; the top half of Figure 3.1 illustrates three examples from a hypothetical dataset. For the

classification: Given a set of examples (a dataset), the task of classification is to form a decision structure (e.g., a rule set) that divides the set into subgroups of similar examples and enables a new example to be placed with the examples that are most similar to it.

attribute: A feature or dimension that applies to each example in the dataset.

attribute-value pair: An attribute and a specific value of that attribute.

example: A set of attribute-value pairs that together comprise one entry or instance in the dataset.



	Examples as attribute−value pairs
e1	(ApoE 33), (ApoH 32), (Chol high), (Trig low), (HDL high), (class +)
e2	(ApoE 33), (ApoH 22), (Chol high), (Trig high), (HDL med), (class +)
e3	(ApoE 43), (ApoH 22), (Chol low), (Trig high), (HDL low), (class −)

Examples in graphic form

ApoE	ApoH	Chol	Trig	HDL	class
33	32	high	low	high	+
33	22	high	high	med	+
43	22	low	high	low	−

	Rules as attribute−value pairs
r1	(ApoE 33), (ApoH 32), (ApoH 22), (Chol high), (class +)
r2	(ApoE 43), (Chol low), (HDL low), (class −)
r3	(ApoH 22), (Chol high), (Chol med), (Chol low), (HDL med), (class +)

Rules in graphic form

ApoE	ApoH	Chol	Trig	HDL	class
33	32 22	high	*	*	+
43	*	low	*	low	−
*	22	high med low	*	med	+

Figure 3.1: An illustration of three examples from the dataset (e1, e2, e3) and three rules from the rule set (r1, r2, r3). Both are illustrated as a set of attribute-value pairs as well as graphically. In the graphical representation, the unspecified attributes of the rules are denoted with asterisks (*). Also, note that in the third rule (r3), the Cholesterol attribute has an attribute-value pair for all possible values, and therefore, could also have been unspecified.

classification task investigated in this thesis, there are a total of 21 attribute-value pairs for each example in the dataset (one attribute-value pair for each possible attribute). In this research, individuals are labelled as positive or negative with respect to family history of disease. In machine learning terminology, there are two classes (positive and negative), and this is a binary classification task. The class is sometimes called the “class attribute”, since it is another dimension that applies to all examples in the dataset.

In this thesis research, we are searching for a rule set to describe individuals with a family history of disease, where a rule set is one or more rules and each rule describes one or more

binary classification task: A classification task with exactly two classes.

class attribute: More commonly called simply the “class”; this is the category of interest for the classification task.

individuals from the data set. The bottom half of Figure 3.1 illustrates three rules from a hypothetical rule set. Similar to examples, rules are also denoted by a set of attribute-value pairs, but are a bit more difficult to describe and comprehend. For the rules in the rule set, there may be zero or more attribute-value pairs for each attribute; if there are zero attribute-value pairs for a given attribute, that attribute is said to be unspecified; otherwise, the attribute is said to be specified.

The specified attributes of a rule are used in the matching process, which determines which examples are described by each rule. Matching compares the specified attributes of the rule to each of the examples in the data set. An example that matches for all specified attributes of the rule is said to match the rule. Figure 3.2 illustrates which of the examples match the rules from Figure 3.1.

The rule format used here is what is sometimes called a "conjunct of disjuncts": The rules are *conjunctive* in that the rule is said to match an individual if the string and the example match for *all specified* attributes; the attributes are *disjunctive* in that an attribute of a rule is said to match an attribute of an example if for that attribute, *one of* the attribute-values represented in the rule is equal to the example's attribute-value, or if the attribute is unspecified. (Note that a specified attribute that has an attribute-value pair for all possible values is equivalent to an unspecified attribute. That is, this attribute will match for all examples in the dataset.)

Rules that match examples in the dataset are said to describe those examples. Collectively, a rule set is said to be a description of the data set, though this does not imply that the description is either complete (describing all examples) or correct (describing the examples accurately; that is, the rule may describe an example as positive, and the example could be negative). Thus, there is generally some metric used to compare the quality of different rule sets that may describe the same dataset in different ways. These metrics will be discussed in Chapter 4.

Most machine learning research in classification focuses on prediction: the available data is divided into a training set and a test set; the rule set is formed using the training set and is evaluated using the test set. Typically, the metric used is prediction accuracy: the percent of examples in the test set that are correctly classified by the rule set.

rule set: A collection of rules that describe a dataset.

rule: A set of attribute-value pairs that describe some subset of the dataset.

unspecified: An unspecified attribute does not appear in the set of attribute-value pairs that constitute a rule, and is not involved in the matching process.

specified: A specified attribute appears in one or more attribute-value pairs in the set that constitutes a rule, and is used in the matching process.

conjunct of disjuncts: The conjunct of disjuncts rule format is conjunctive in that the rule must match for all specified attributes and disjunctive in that any attribute may be specified by a disjunctive expression indicating one or more values.

prediction: The task of classifying examples not in the original training data set.

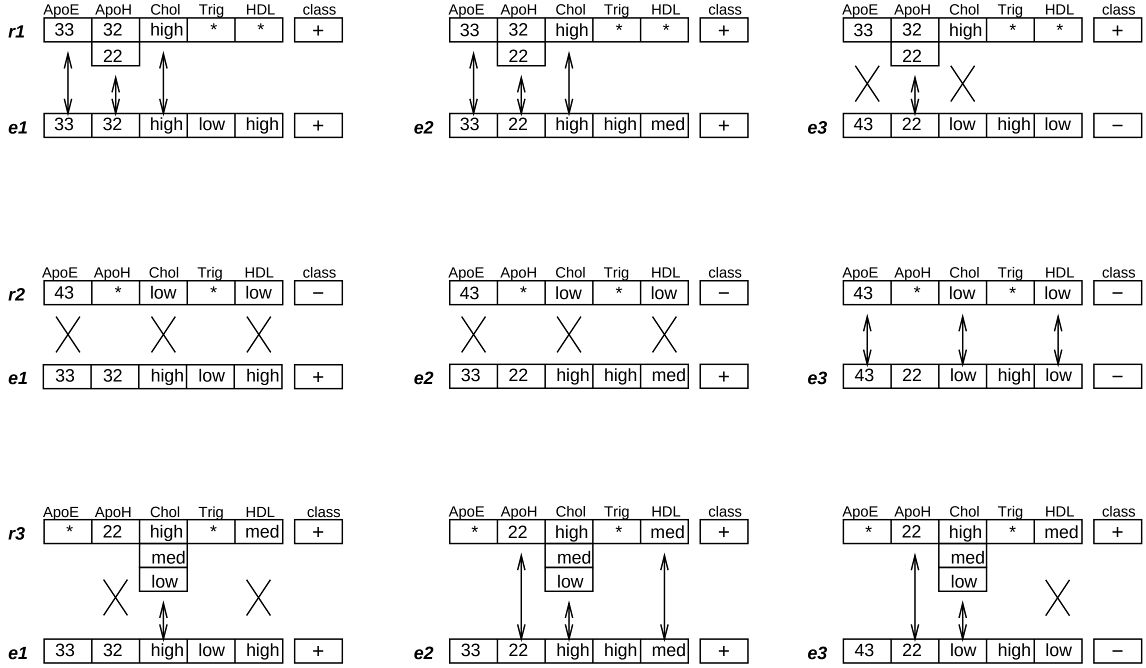


Figure 3.2: An illustration of the matching process, using the examples and rules from (the previous illustration). Rule 1 matches examples 1 and 2; rule 2 matches example 3; and rule 3 matches example 2. In this illustration, the class of the rule is the same as the class of the example for all matches, but this is not necessarily the case.

3.2 The Machine-Learning Research Questions

The common-disease research question posed in Chapter 2 provides a challenging classification task that allows the following machine learning question to be addressed in this thesis:

1. What is the "best" approach for a complex classification problem such as the one presented here?

This raises two important subproblems:

2. How do we measure the quality of a rule set for this task?
3. How can we evaluate the complexity of a classification task?

The first question constitutes the bulk of this thesis and is pursued in Chapters 5–8. The second question represents a departure from the usual approaches in machine learning, and

training set: The data set used to form the rule set in a classification task.

test set: The data set used to evaluate the rule set in a classification task.

prediction accuracy: The metric commonly used to evaluate classifications, equal to the percent of examples in the test set that are correctly classified by the rule set.

is addressed in this section as well as in Chapter 4. A complete answer to the third question is beyond the scope of this thesis, but relevant issues are presented in Chapter 8.

In this research, we are searching for a rule set to describe the class of individuals with a family history of disease. This is considered a classification task in machine learning; however, this specific task is different from the standard classification task for the following reasons:

1. The emphasis in this task is *description*, whereas most classification tasks place an emphasis on *prediction*.
2. The problem is asymmetrical in that it focuses on description of positive examples over description of negative examples. Most machine learning tasks attempt to describe (or more likely, predict) all classes in the dataset equally well.
3. There is a "human psychology" component to this task, which does not appear in many machine learning domains. The needs of the common-disease research community influence how we judge the quality of a rule set.
4. A typical data set is small (about 500 examples), relative to the size of the search space of possible rules (with 21 attributes, 2^{21} or larger). Many machine learning tasks have more examples available, relative to the size of the search space.

To elaborate on the points made in the above list, first, the emphasis on this task is on description, and not on prediction. In this research, we want to identify the combinations of risk-factor values associated with a family history of CAD, not just predict whether a given individual is likely to have a family history of CAD. The reasons for this include a desire to understand the relevant combinations of risk factors associated with a family history of disease. Some machine learning systems are able to do prediction without description (for example, neural nets); using such a system would not shed any light on the possible etiologies for CAD.

Evaluating the predictive abilities of a model is important from the machine learning perspective, however, the validity of a model in common disease research is more commonly checked by replicating studies: sampling more data (from the same population), forming a model, and comparing this model to those developed by other researchers. Recall that this paradigm assumes a single-etiology model, that is, that one rule is sufficient and appropriate to describe the data. With the assumption that one rule is sufficient to explain the data, it makes sense to use all of the available data to form that one rule and make it as accurate as possible. In a sense, this formulation *subsumes* the prediction task because it is a description of the data that also contains an estimate of the expected prediction accuracy. When multiple rules are allowed, as is typical in machine learning research, we no longer have an estimate of the expected prediction accuracy as a byproduct of forming the rule set because each rule corresponds to only a subset of the data. With multiple rules, checking the prediction accuracy of the rule set is partially a confirmation that the rule set has divided its explanatory power into appropriate subgroups.

The second reason this task is different from the standard classification task in machine learning is that the task is not symmetric; it is more important that the rule set describe positive examples well than it is that the rule set describe the negative examples well. This can be considered separately from the issue of how we measure a good rule set. (Metrics for describing rule sets will be discussed in the next chapter.) That is, of greatest importance is describing individuals with a family history of CAD; by identifying characteristics of individuals who are "at risk", we hope to gain some understanding of the factors that set them apart from people who are not at risk.

Thirdly, because the classifications must be interpretable to common disease researchers (and be understandable to them), there is a preference for rule sets that are small. The idea of multiple etiologies is unfamiliar and perhaps unpopular with the common-disease research community; part of the goals of this research from this point of view is a paradigm shift. If common disease researchers are presented with 100 rules, many will likely reject the model outright, whereas if they are presented with 5 rules, even if they are not "great" rules (from a machine learning perspective), they can see that the multiple-etiology model is an improvement over a single-etiology model. (Ideally, common-disease researchers will be presented with a range of rule sets, including the extremes of a one-rule set that is rather inaccurate and a 100+ rule set that is very accurate, to fully appreciate the benefit of a multiple-etiology model.) This is likely to be the point that causes the most contention among machine learning researchers; it is rare for researchers to consider that there may be reasons that a smaller, less accurate rule set is "better" than a larger, more accurate rule set.

Fourthly, the dataset is small relative to the search space; it is important not to overfit the data such that every individual in the data set is explained by a very specific rule (which may not explain any other data points). A small number of general rules, say 10, may be preferable to a larger number of specific rules, say 100. Even though the general rules will likely be less accurate, they are expected to be more robust in that they should do better at generalizing to examples not in the original dataset. Note that the desire not to overfit the dataset is compatible with the desire for small rule sets, but at odds with the desire to describe the positive examples well.

Finally, the classification task may be different from many in the literature in that there seems to be a large degree of interactions between the attributes and the classification task. Also, many machine learning systems assume that the attributes are independent.

Many computer scientists do not have a solid understanding of the implications of the terms "interaction" and "independence", so a brief discussion of this may be helpful. Two attributes are said to interact with the class attribute if their contribution to predicting the value of the class attribute is not additive. For example, in our dataset, 38% of the people who have LOW TRIG, 42% of the people who have MED TRIG, and 53% of the people who have HIGH TRIG have a family history of disease. This is illustrated by the solid line labelled "All HDL" in Figure 3.3. When we also consider HDL values in predicting the class attribute, the four lines in Figure 3.3 would be parallel if TRIG and HDL do not interact. Instead, what we see is that LOW HDL does not follow the same pattern as the other values

overfit: A rule set is said to overfit the data when it describes the training data at the expense of generalizing to test data.

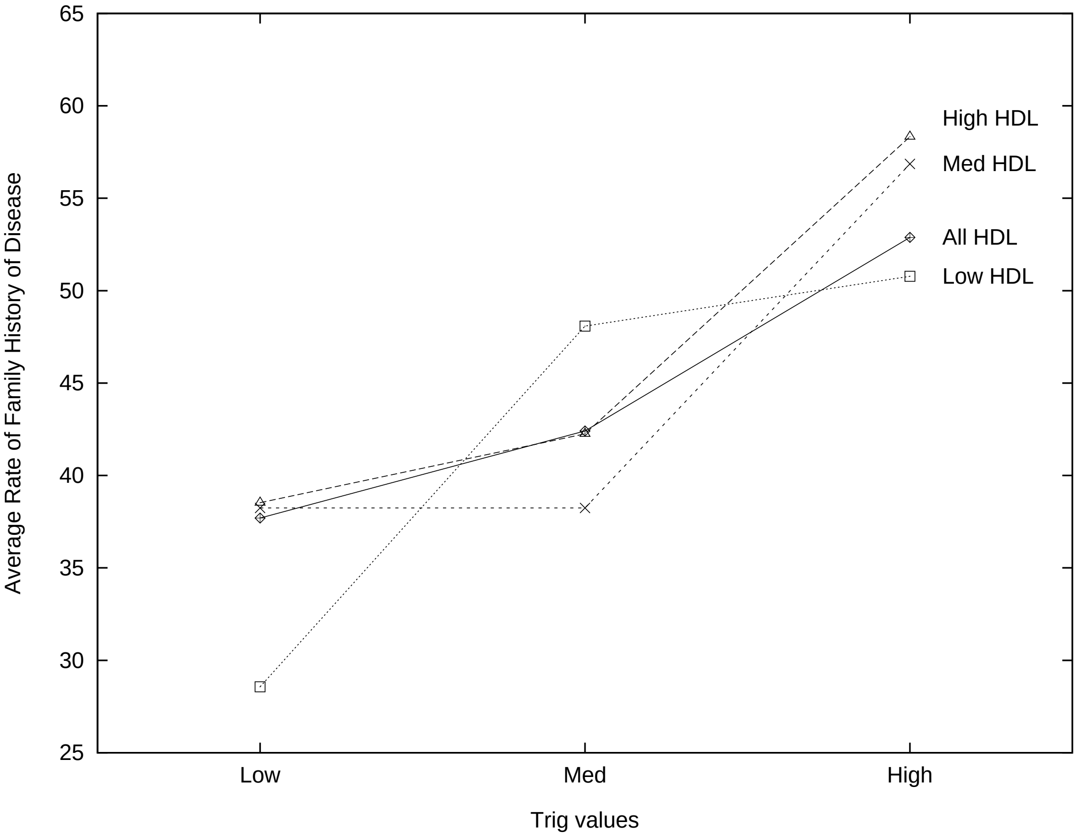


Figure 3.3: An illustration of the interaction between the TRIG attribute and the HDL attribute in predicting a family history of disease.

of HDL. This figure illustrates an interaction between TRIG and HDL in predicting family history of disease. Interactions between attributes and the class attribute may complicate the classification procedure. For example, looking just at the HDL attribute, LOW HDL is the best predictor of a family history of disease (48% of these individuals have a family history of disease); however, looking at a combined HDL and TRIG term, we find that HIGH TRIG, HIGH HDL is the best predictor of a family history of disease (58% of these individuals have a family history of disease). Since most machine learning systems look at one attribute at a time, they will tend to miss some of these interactions.

Two attributes are said to be independent if knowing the value of one does not help to predict the value of the other; mathematically, two attributes are independent if $P(A_i|B_j) = P(A_i) * P(B_j)$ holds for all attribute-values A_i and B_j corresponding to two different attributes A and B. The 573 examples in our dataset are divided nearly evenly into the LOW, MED, and HIGH, tertiles for each attribute; there are 191 examples for each value of TRIG, and 194, 187, and 192 examples for each of LOW, MED, and HIGH HDL, respectively. For TRIG and HDL to be independent, we would expect to see about 65 examples for each value of TRIG with LOW HDL, about 62 examples for each value of TRIG with MED HDL, and 64 examples

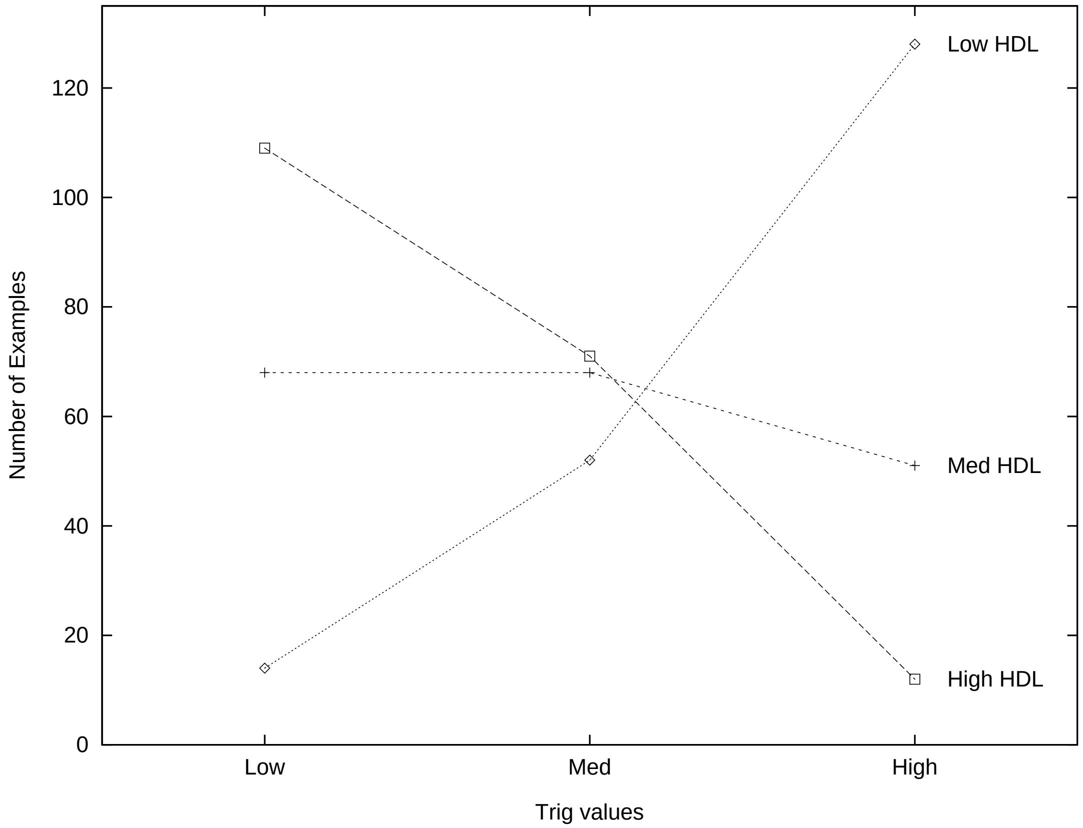


Figure 3.4: An illustration of the lack of independence between the TRIG attribute and the HDL attribute.

for each value of TRIG with HIGH HDL. However, the actual numbers of examples for each of these combinations varies quite a bit from this [62..65] range. For example the HIGH TRIG, HIGH HDL combination explains only 12 examples from the dataset. The TRIG and HDL attributes in our dataset are not independent, as illustrated in Figure 3.4. In the figure, we see that there is a tendency for LOW values of TRIG to be paired with HIGH values of HDL, and vice versa. Attributes that are not independent may complicate the classification procedure for systems that rely on statistical models, since most statistical models assume that each attribute can be considered independently when forming the classification.

In addition to interactions and lack of independence, an additional consideration that relates to the fourth point in the above list is that there is a degree of noise in the dataset. For example, the phenotype levels of an individual may vary from one day to the next (or one hour to the next), so that the values recorded for any one person at a particular time may be higher or lower than they are in general. In addition, lab errors may occasionally result in noise when measuring individuals; for example, a cholesterol level may be recorded as 210 when it should have been recorded as 120 (or an instrument may be faulty). Finally, we are using family history of CAD as the classification, which is not correlated 100% with CAD in the individual. Thus, we would expect there to be a certain number of individuals who have a family history of CAD, but who have attribute-value profiles that more closely resemble other individuals who have no family history of CAD (and vice versa). All of these factors contribute to noise, and add to the difficulty of the task.

3.3 Promising Machine-Learning Systems

The capabilities needed for this classification task were listed in Section 2.3:

1. allow for *more than one model* for different subsets of the dataset
2. allow for complex interactions between attributes and the class
3. allow for attributes that are not independent
4. minimize the error in each of the models
   (a) in describing the original data
   (b) in generalizing to unseen data (from the same population)
5. minimize the number of models needed to explain the data
6. allow for asymmetry in classification errors, that is, false positives might be more significant than false negatives, or vice versa.

Many of these are taken for granted in machine learning research, although many systems do not allow for complex interactions between the attributes and the class, and few consider the possibility of asymmetry.

Several machine learning approaches are investigated in this research. An attempt has been made to choose systems that produce rule sets interpretable by humans and that are well known in the machine learning community.

The systems that have been implemented in this thesis are:

- Genetic Algorithms
- ID3 and other decision-tree systems
- Autoclass
- Cobweb
- Random generation
- Steepest-ascent hillclimbing

noise: Noise in a dataset is a vague term that can apply to any of a number of factors that may make the dataset more difficult to classify. For example, there may be errors (typographical errors or measuring errors) in the dataset, or it may be that two examples in the dataset have identical values for all attributes but are in two different classes. The latter type of noise may be due to an error, but can sometimes be attributed to not considering enough attributes. That is, if additional attributes were included in the dataset, the two examples might not be identical for all attributes.

Each of these systems will be described below; specific issues related to using the system for our classification task will be described in later chapters. The random generation and hillclimbing approaches are included primarily as comparisons against genetic algorithms. Neural net approaches were excluded at this time because they do not lend themselves to human-interpretable rules.

3.3.1 Genetic algorithms

There are many variations on genetic algorithms, and it would be impossible to report on them all or investigate them all in this thesis work. In this section, I present a representative simple GA formulation.

The GA [51, 41, 23] is a search procedure inspired by principles of natural selection and population genetics. With this approach, a population of strings *evolves* so that over multiple generations, the population tends to include better and better solutions to an optimization problem.[1]

3.3.1.1 Components of a GA

The basic components of a genetic algorithm are:

A population of strings. These strings represent potential solutions to the optimization problem. The elements of the string (typically, bits) encode the important features of the solution to the optimization problem. (In the thesis research, each string represents a rule.)

An evaluation measure. This is usually called a "fitness function" in the GA literature. This function takes as input a string from the population, and returns a number that indicates the quality of the string as a solution to the optimization problem. The fitness function is provided by the user, and is selected specifically for a given task.

A method for parent selection. Typically, only some of the strings in the current population will participate in the reproduction process. The parent strings are chosen randomly (with replacement); in addition, most GA implementations use a weighted random selection scheme so that higher fitness strings are more likely to be selected as parents.

Reproduction operators. These are the mechanisms by which new strings are formed from parent strings; typically, the operators used are mutation and crossover. Mutation forms an offspring string from a parent string by changing one or more randomly selected elements; crossover swaps randomly selected portions of two strings, forming two new offspring strings.

A method for replacement. Because almost all GAs maintain a constant population size, each new offspring introduced into the population means that one existing string must be removed. The simplest approach to this is to create as many offspring strings

[1] GA's need not be used exclusively for optimization problems, but this is a typical application.

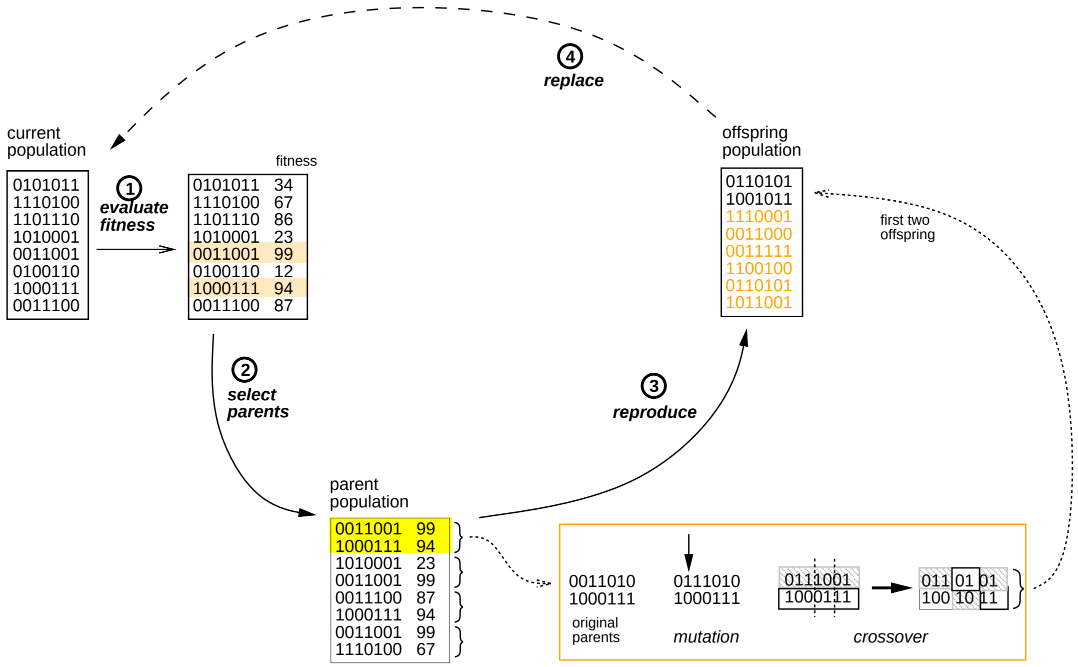


Figure 3.5: The GA proceeds through a number of generations. In each generation, first, the fitness of the strings in the current population are evaluated. Second, the population of parent strings is selected from the population of current strings; the probability of a string being selected is proportional to its fitness. The selected parents are paired off. Third, each pair of parent strings undergoes mutation and crossover; the new strings form the offspring population. Finally, the offspring population replaces the current population.

as parent strings; the population of offspring strings entirely replaces the current population.

3.3.1.2 Algorithm for the GA

The GA starts with an initial population of strings (which may be randomly generated, created by the user, or a combination of the two). Call this the *current population*. Each string in the current population is evaluated using the fitness function. A second population of strings, called the *parent population* is selected (with replacement) from the current population; the parent population is the same size as the current population. Each string in the current population has a probability proportional to its fitness that it will be added to the parent population, with the effect that high-fitness strings from the current population are likely to be represented multiple times in the parent population, while low-fitness strings are likely to be excluded from the parent population. The strings in the parent population are paired up for the reproduction process. Each pair of parent strings form two offspring strings via mutation and crossover; the two new strings are put into a third population, called the *offspring population*. Once all pairs of parents have reproduced, the offspring population is the same size as the current population. The offspring population replaces the current population. This cycle (evaluation, parent selection, reproduction, replacement) constitutes one generation of the GA. The reproduction cycle is illustrated in Figure 3.5.

The GA proceeds through many generations until a stopping criterion is met. For example, the stopping criterion may be that the population has converged to a single string (within some tolerance) or that a predetermined number of generations or fitness-function evaluations has been completed.

3.3.1.3 Result from a GA run

In the simple GA formulation, the "result" of the run is the single best string found in the course of the run; in the classification task presented here, this single best string represents the rule associated with the largest group of individuals with a family history of CAD. (This string will not necessarily be present in the final population because the offspring population entirely replaces the parental population with each generation.)

In the research presented here, the mechanics of the simple GA was changed in order to evolve rule sets, rather than find the best rule. The details of this are discussed in Chapter 5.

3.3.2 Decision trees

Like GA's, there are many variations of decision trees, and it would be impossible to report on them all or investigate them all in this thesis work. A decision-tree system recursively partitions the dataset into smaller subsets, at each level of the recursion choosing one attribute to "branch" on (creating two or more subsets); each branch is labelled with an attribute and value (or set of values). When the recursion stops, the final subsets are called the "leaves" of the decision tree that has been formed by the recursive process; each leaf is labelled with a class. Each level of branching represents an attribute-value pair, so the results from a decision-tree run may also be written as a rule set (as well as a decision tree).

3.3.2.1 Components of a decision tree system

The basic components of a decision tree system are:

An evaluation measure. This function assigns a value to the quality of the partition obtained when the current subset is branched on a specific attribute, using the specified splitting criterion. The evaluation measure may vary from one decistion tree system to another, but is not usually designed for any specific task.

A splitting criterion. The splitting criterion determines how a particular subset should be partitioned, using the specified attribute. Some decision tree systems create one branch for each possible value of each attribute; others may be binary, creating two branches with each split, and one or more values for each branch.

A stopping criterion. The stopping criterion signals that the recursion should end at this level. The simplest stopping criterion is that all the examples in the current node are the same class (or that there are no further attribute-values to branch on).

3.3.2.2 Algorithm for a decision tree system

The algorithm for a decision tree is recursive: A decision tree system begins with the entire data set; this may be called the root node. Using a binary splitting criterion with discrete attributes, each attribute is examined in turn. All possible binary splits are considered for all attributes, using the evaluation measure to find the best split. Two branches are formed, based on the best split, resulting in two new subnodes. Each of these subnodes may be further divided by comparing all possible binary splits. The stopping criterion is checked before each split, and no splitting occurs if the stopping criterion is met. The final set of nodes (those with no subnodes) are called leaf nodes, and are labelled with a class name. The decision tree cycle is illustrated in Figure 3.6.

One of the best known decision-tree systems is Quinlan's ID3 [83]. (Precursors to ID3 include CLS [56] and CART [13].) As an evaluation measure, ID3 uses a measure from information theory, called Shannon's uncertainty measure, intended to minimize the expected number of tests needed to classify an example. The highest evaluation is given to the attribute split that gains the most information, that is, the split for which the uncertainty is the least. (Which can also be viewed as the split in which the examples are distributed "least randomly" over the possible classes.)

The information gain of attribute a_i from the set of attributes A is the entropy of the set S, minus the entropy of attribute a_i with respect to S:

$$Gain(a_i, S) = Ent(S) - E(a_i, S) \tag{3.1}$$

The entropy of the set S when S has two classes C_1 and C_2 is defined as:

$$Ent(S) = -P_{C_1} \log_2 - P_{C_2} \log_2 \tag{3.2}$$

where P_{C_j} is the probability of occurence of class C_j in the set of examples S (more simply, the percent of examples from S that are in class C_j):

$$P_{C_j} = \frac{|e \in C_j|}{|S|} \tag{3.3}$$

When attribute a_i has j possible values from the set $V_i = \{v_{i1}, v_{i2}, ...v_{ij}\}$, a_i partitions S into j subsets, $S_1, ...S_j$, and the entropy of the attribute a_i with respect to set S is:

$$E(a_i, S) = \sum_{v_{ij} \in V_i} \frac{|S_j|}{|S|} Ent(S_j) \tag{3.4}$$

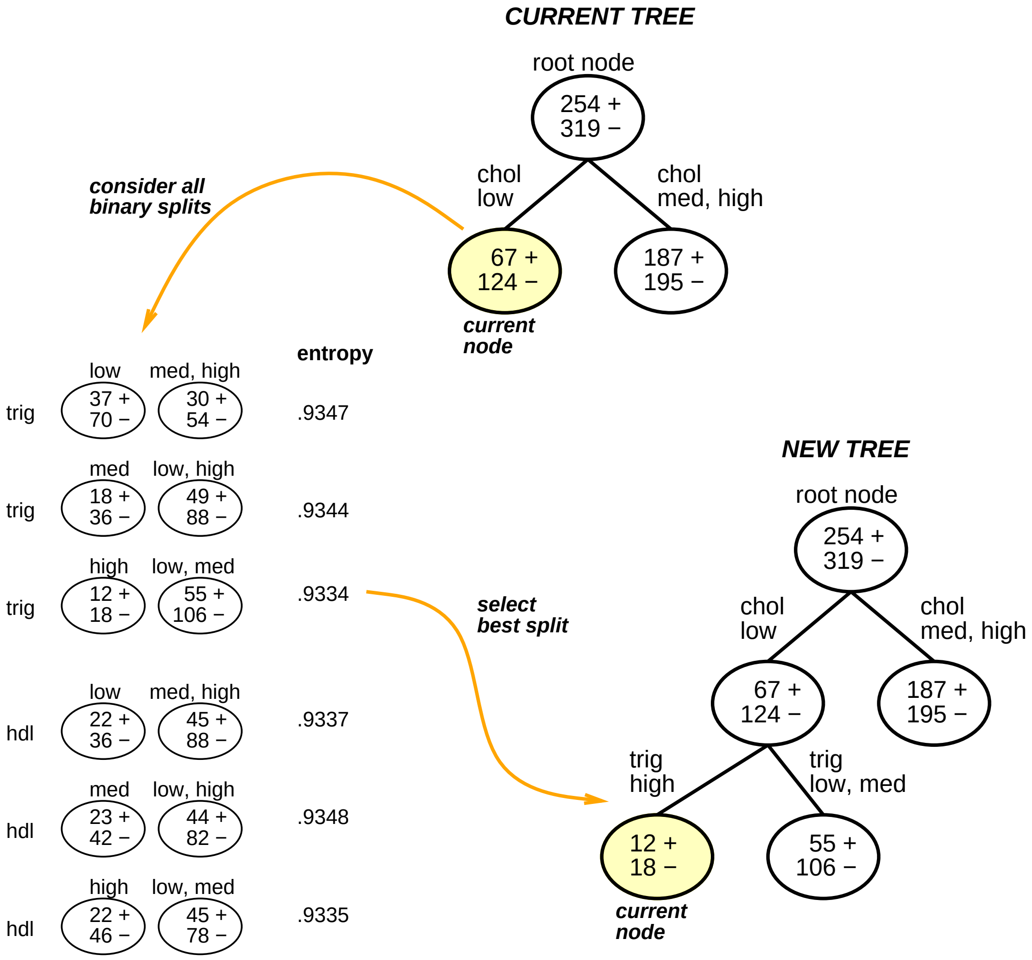


Figure 3.6: An illustration of a binary decision tree, using entropy as the evaluation measure (lower entropy is a better partition).

Intuitively, information gain is a measure of the information gained by splitting on a particular attribute. That is, the attribute that partitions the data into subsets that appear to be the least random. When ID3 splits on an attribute, it creates one branch for each possible value of that attribute. Other decision tree systems, such as GID3* [32], create binary trees, creating exactly two branches for each split. Some decision tree systems also use other evaluation measures. These variations will be discussed further in Chapter 6.

3.3.2.3 Results from a decision tree run

The resulting decision tree forms a partition of the dataset; each example falls uniquely into one leaf node. In classifying an example, the tree is traversed from the root node to one leaf node, following the appropriate attribute-value branch at each level. One can convert a decision tree into a rule set by repeatedly traversing the tree, once for each leaf node, and recording the attribute-value pairs at each level.

In Figure 3.6, the rule under formation (indicated by the "current node" at the bottom of the figure), could be read as "Cholesterol low, Triglycerides high".

3.3.3 Autoclass

Autoclass [17, 57] is a classification program that uses Bayesian statistical techniques to determine the probable number of classes in a dataset and probabilistic descriptions of these classes. Autoclass is an example of what is sometimes called an unsupervised learning system, which means that it does not expect the examples in the dataset to be identified as belonging to a particular class. (Our GA formulation and decision trees would be considered supervised learning systems, because both require that the class attribute be specified in the dataset. Instead, Autoclass defines its own classes, based on similarities found between examples in the dataset.)

Because Autoclass is an unsupervised learning system, it is not suited to the common disease research task presented in Chapter 2. However, it is included in the thesis for the sake of comparing it with other machine learning tasks in terms of predictive accuracy. Although this endeavor is nonsensical from the perspective of common disease research, it is consisent with research in machine learning.

3.3.3.1 Components of Autoclass

The basic components of Autoclass are:

An evaluation measure. A form of Bayes' law is used in Autoclass to compare competing descriptions of the data. This provides a well-established statistical method for combining probabilities, balancing a classification's complexity against its power to account for the data. This measure is fixed in the architecture.

Prior probabilities. The use of prior probabilities constitutes the fundamental difference between Bayesian and classical statistics. They measure how unlikely a classification is. More complex classifications (those with more classes, for example) are considered more unlikely than simpler ones.

Model functions. The model functions establish which of several available statistical models should be used with each attribute; different subsets of attributes may use different models.

unsupervised learning system: A classification system that does not regard the class attribute of the examples as more important than other attributes, or ignores the class attribute altogether. Such a system defines its own classes within the dataset, finding similarities in the values of attributes.

supervised learning system: A classification system that uses and requires a class attribute in the dataset. Such a system is looking for a description of the dataset based on the class attribute.

Expected number of classes. Autoclass begins its search with a specific number of classes and then tries fewer classes as it looks for a better description of the data.

3.3.3.2 Algorithm for Autoclass

Autoclass uses an iterative algorithm. Beginning with the number of expected classes[2], it randomly partitions the data into that number of classes. Then it repeatedly:

1. Searches for the best description of the current classes.
2. Reassigns each example to the class it belongs to with the highest probability.

This iterative process typically converges in about 20 iterations, according to Cheeseman et al. [17]. Each convergence is called a "try". Autoclass will save the classification obtained from the current try, and will start a new try, continually looking for a better classification. The system will continue until a maximum time limit is reached, or a maximum number of tries has been reached. The Autoclass iteration cycle is illustrated in Figure 3.7

The bulk of the work is done in Step 1 of the iteration, where Autoclass searches for the best description of the current classes; this will be described below. The initial assignment of examples to classes is done randomly and the assignment of examples to classes in Step 2 is also straightforward: the probability that example x_i belongs to class C_j is evaluated for all examples and for all classes, and each example is assigned to the class it belongs to with the highest probability.

In simplified form, the probability that example x_i belongs to class C_j is calculated:

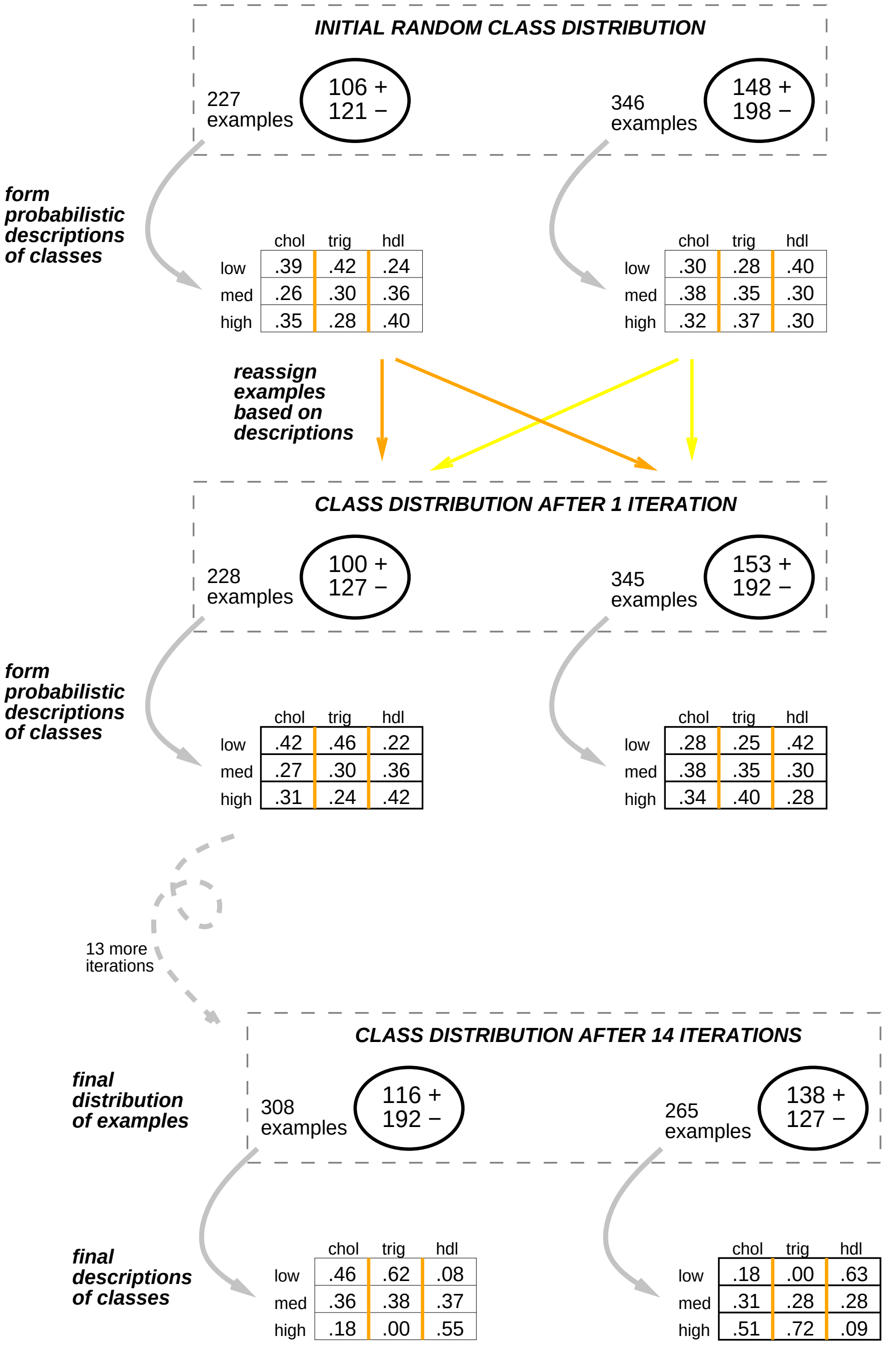
$$\mathcal{P}(x_i \in C_j|\vec{D}) = \frac{\mathcal{P}(C_j)\mathcal{P}(x_i|x_i \in C_j, \vec{D})}{\mathcal{P}(x_i|D)} \tag{3.5}$$

Where $\vec{D}$ is the description of the current classes. In English, the probability that the example is in the class, given the current class descriptions, is equal to the probability of *any* example being in that class times the probability of the example occuring, assuming both that the example *is* in the class and that the current descriptions hold, divided by the probability that example comes from the current set of classes and their descriptions.

The difficult part of the Autoclass iteration is Step 1, forming the class descriptions. The Autoclass model assumes that the real-valued data can be described by multi-dimensional Gaussian distributions[3]. That is, each class is described by a mean and a variance for each attribute (the classes are collectively described by a class parameter vector, $\vec{\theta_j}$, which includes means and variances for all attributes and a class probability vector π, which includes the probability of an example being drawn from class C_j for each C_j).

[2] The number of expected classes is provided by the user; Autoclass may be told to try a different number of expected classes with each try.

[3] Discrete-valued data is described more simply, with a probability for each possible attribute-value.


INITIAL RANDOM CLASS DISTRIBUTION
227 examples
106 +
121 −
346 examples
148 +
198 −
form probabilistic descriptions of classes
reassign examples based on descriptions
CLASS DISTRIBUTION AFTER 1 ITERATION
228 examples
100 +
127 −
345 examples
153 +
192 −
form probabilistic descriptions of classes
13 more iterations
CLASS DISTRIBUTION AFTER 14 ITERATIONS
final distribution of examples
308 examples
116 +
192 −
265 examples
138 +
127 −
final descriptions of classes

	chol	trig	hdl
low	.39	.42	.24
med	.26	.30	.36
high	.35	.28	.40

	chol	trig	hdl
low	.30	.28	.40
med	.38	.35	.30
high	.32	.37	.30

	chol	trig	hdl
low	.42	.46	.22
med	.27	.30	.36
high	.31	.24	.42

	chol	trig	hdl
low	.28	.25	.42
med	.38	.35	.30
high	.34	.40	.28

	chol	trig	hdl
low	.46	.62	.08
med	.36	.38	.37
high	.18	.00	.55

	chol	trig	hdl
low	.18	.00	.63
med	.31	.28	.28
high	.51	.72	.09

Figure 3.7: An illustration of Autoclass, using two classes. In this example, the distribution of examples and the class descriptions converged in 14 iterations.

Step 1 of the iteration is itself an iterative process, via heuristics and the application of Bayes' law. As noted by Cheeseman et al. [17], given observed data D and a hypothesis H, Bayes' law is often expressed as:

$$\mathcal{P}(H|D) = \frac{\mathcal{P}(H)\mathcal{P}(D|H)}{\mathcal{P}(D)} \tag{3.6}$$

In English, the probability that the hypothesis explains the data ($\mathcal{P}(H|D)$) is proportional to the probability of observing the data if the hypothesis were known to be true ($\mathcal{P}(D|H)$) times the prior probability of the hypothesis ($\mathcal{P}(H)$). The numerator in Equation 3.6 is the prior probability of the data, ($\mathcal{P}(D)$).

In applying Bayes' law using Autoclass, the hypothesis H is the number and descriptions of the classes from which we believe the data D has been drawn. Given D, we must select H to maximize $\mathcal{P}(H|D)$, called the *posterior probability.*

The specific terms used to derive the posterior probability are:

$\mathcal{P}(D|H)$ — This is the *likelihood* of the data, given the hypothesis; it is explained below.

$\mathcal{P}(H)$ — This is the *prior probability* of the hypothesis; this term measures how unlikely the current classification is. In Autoclass, this term is approximated from looking at the data itself. Autoclass assumes a priori that descriptions with more classes/rules are less likely than those with fewer rules.

$\mathcal{P}(D)$ — This is *prior probability* of the data. Given a dataset, this term will be constant. Therefore it need not be calculated since it does not affect the relative values of the posterior probabilities of different hypotheses.

In Step 1 of the iteration, we apply Bayes' law (Equation 3.6), trying to maximize the posterior probability, $\mathcal{P}(H|D)$. To maximize the posterior probability, a variation of Dempster and Laird's EM algorithm [26] is used. This algorithm transforms Equation 3.6 into a system of nonlinear equations which hold at the maximum of the posterior probability. To find a solution to this system of equations, Autoclass iterates between these nonlinear equations (while holding $\vec{w}$ constant) and Equation 3.5 (while holding $\vec{\pi}$ and $\vec{\theta}$ constant); $\vec{w}$ is the vector of all w_{ij}, where w_{ij} is the probability that the x_i was drawn from class C_j, from Equation 3.5.

If classification is begun with more classes than necessary, the class weights drop to 0 for some classes, thus, Autoclass may arrive at a description with fewer classes than suggested by the user.

3.3.3.3 Results from an Autoclass run

The result of an Autoclass run is a set of probabilistic class descriptions. For real-valued attributes, each class is described in terms of a mean and a variance for each of its attributes, and each object belongs to each class with some probability. For discrete-valued attributes, each class is described by the probability of each attribute-value. Each attribute also has an "influence" that measures how significant the attribute's value is for assigning an instance to the class. In addition, each class has an influence value indicating how many data points it covers.

Figure 3.7 illustrates the result from one try with Autoclass, using a reduced set of attributes. (Since the results from any given try is strongly tied to the initial random class distribution, recall that Autoclass continually generates new tries until asked to stop by the user, saving the best try.) In this exercise, Autoclass was told to look for two classes. The final distribution of examples does not represent an improvement in terms of segretating positive and negative examples, which is not surprising with an unsupervised learning system. Using all attributes, including the class attribute equally, Autoclass was able to find a stronger division of the dataset into two classes by relying more strongly on the other available attributes. The two classes found here represent one class of people who tend to have low triglyceride levels and high HDL levels and a second class of people who tent to have high triglyceride levels and low HDL levels. Also, the people in the "low triglycerides" class tend to have low cholesterol, and the people in the "high triglycerides" class tend to have high cholesterol. The associations among these three attributes is well known to medical researchers. Still, it is good to know that Autoclass was able to discover these associations on its own, and serves as an illustration of the fact that with the full dataset, the attributes are not independent.

3.3.4 Cobweb

Cobweb [35] forms "classification trees", which are similar to decision trees described above, but with a probabilistic component. Classification trees are "polythetic" (the choice of which branch to follow is based on many attributes), whereas decision trees are "monothetic" (the choice of which branch to follow is based on a single attribute). To classify an example, a partial match is made against each of the possible nodes at the next level of the classification tree. The best node determines which branch of the tree to follow.

Unlike previous systems, Cobweb is an incremental learning system which means that the classification tree is formed by considering one example at a time, rather than looking at the entire dataset. (Note that our GA formulation, decision trees, and Autoclass are called nonincremental learning systems.)

Like Autoclass, Cobweb is an unsupervised learning system; it does not distinguish the class attribute. And, therefore, like Autoclass, Cobweb is not suited to the common disease

incremental learning system: An incremental learning system constructs its classification by considering one example at a time and modifying its current classification based on the new example. Therefore, changing the order of the examples in the dataset may change the resulting classification.

nonincremental learning system: A nonincremental learning system forms its classification using all the examples available in the dataset (or training set). The order of the examples is not usually relevant to a nonincremental learning system.

research task presented in Chapter 2. However, it is included in the thesis for the sake of comparing it with other machine learning tasks in terms of predictive accuracy.

3.3.4.1 Components of Cobweb

The basic components of Cobweb are:

An evaluation measure. The evaluation measure is a heuristic used to compare competing trees; in Cobweb, this metric is fixed in the architecture; the metric used is called category utility, described below.

Tree-building operators. As each example is considered, the classification tree may grow or shrink: a new class may be formed, two classes may be merged together, or one class may be split into two.

The evaluation measure used in Cobweb is called category utility, and rewards similarity of objects within the same class and dissimilarity of objects in different classes. This measure provides a tradeoff between intra-class similarity (similarity within each class) and inter-class dissimilarity (differences across classes) of examples.

In the following equations, $A_i = V_{ij}$ is an attribute-value pair, and C_k is a class. Intra-class similarity is measured using the conditional probability of the attribute-value pair, given the class: $\mathcal{P}(A_i = V_{ij}|C_k)$. Larger probabilities correspond to more predictable attribute-values within a given class. Inter-class dissimilarity is measured using the conditional probability of the class, given the attribute-value pair: $\mathcal{P}(C_k|A_i = V_{ij})$. Larger probabilities correspond to attribute-values that are more predictive of class membership. Using these two measures, Cobweb users talk about the tradeoffs between predictability (predicting an attribute-value, given a class) and predictiveness (predicting a class, given an attribute-value).

The quality of a partition can be measured by combining predictability and predictiveness multiplicatively into a single term, and weighted by the prior probability of each attribute so that frequently occuring attribute values are given more weight than infrequently occurring attribute values:

$$\sum_{k=1}^{n} \sum_{i} \sum_{j} \mathcal{P}(A_i = V_{ij}) \mathcal{P}(C_k|A_i = V_{ij}) \mathcal{P}(A_i = V_{ij}|C_k) \tag{3.7}$$

Using Bayes rule (see Fisher [35] for details), Equation 3.7 is equal to[4]:

$$\sum_{k=1}^{n} \mathcal{P}(C_k) \sum_{i} \sum_{j} \mathcal{P}(A_i = V_{ij}|C_k)^2 \tag{3.8}$$

[4] According to Fisher, "this expectation assumes a guessing strategy that is *probability matching*, meaning that an attribute value is guessed with probability $\mathcal{P}(A_i = V_{ij}|C_k)$ and that this guess is correct with the same probability."

The category utility CU of a partition into classes $C_1, ..., C_n$ is the increase in the expected number of attribute-values that can be correctly guessed given the partition (Equation 3.8) beyond the expected number of correct guesses *without* this partition $(\mathcal{P}(C_k) \sum_i \sum_j \mathcal{P}(A_i = V_{ij})^2$. Therefore, the category utility $CU(C_1, ..., C_n)$ is:

$$\frac{1}{n}\sum_{k=1}^{n} \mathcal{P}(C_k)[\sum_i \sum_j \mathcal{P}(A_i = V_{ij}|C_k)^2 - \sum_i \sum_j \mathcal{P}(A_i = V_{ij})^2] \tag{3.9}$$

Since n is the number of categories in the partition, making category utility an average over n allows comparision of different-sized partitions.

3.3.4.2 Algorithm for Cobweb

Cobweb incrementally adds each example into the current classification tree, and is iterative in this respect. While adding an example to the current tree, Cobweb is a recursive algorithm.

Initially, there is one node in the classification tree, which constitutes a single class. As each example is added to the tree, the example is filtered through the tree; each level of the tree corresponds to a new level of the recursion. At each level of the recursion, the counts of the current node are updated, and the category utility is calculated for each tree that would result if the example were added to each of the children of the current node. The highest category utility identifies the best child (the node that is the best host of the example). The following options are then considered:

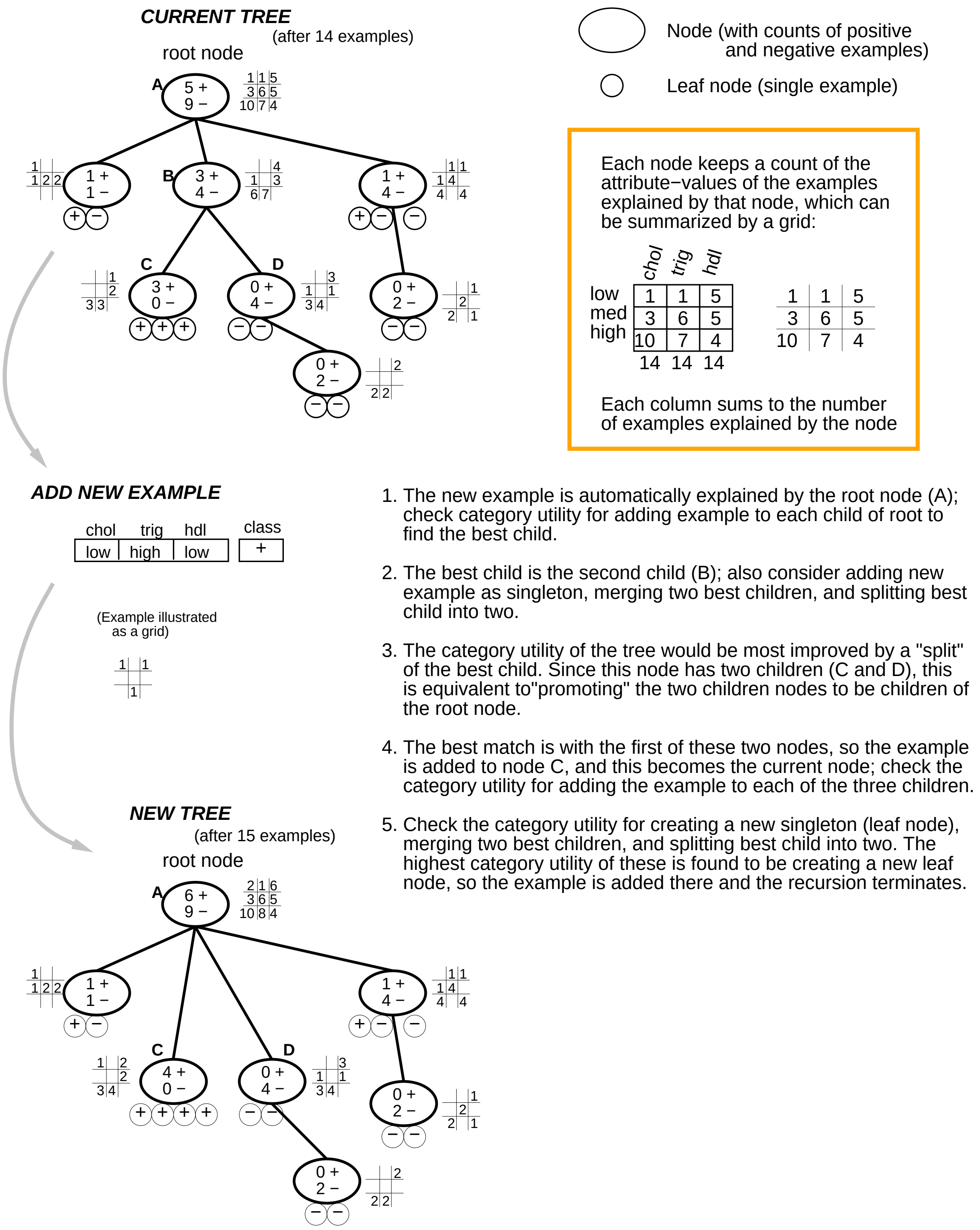
1. Should the new example be added as a singleton class (a sibling of the best child)?
2. Should the two best children be merged into a single node?
3. Should the best child be split into two separate nodes?

The category utility is calculated for each of these possibilities. If any result in higher category utility than adding the example to the best child, the option that results in the highest category utility is chosen, and the example is added in the appropriate fashion (to the best child, to the new singleton class, to the merged node, or to one of the split nodes). The recursion stops when the example is added as a new singleton node.

The Cobweb cycle is illustrated in Figure 3.8

3.3.4.3 Results from a Cobweb run

The result from a Cobweb run is a classification tree; though each node of the tree is a probabistic description of the examples classified by that node, each example will fall uniquely into one leaf node. Like a decision tree, the tree may be traversed from root to leaf node; however, since the descriptions of the nodes are probabilistic rather than absolute, one does not read off a rule from traversing the tree. Instead, one uses the entire tree as the "rule set"; filtering in a new example from root to leaf. The process is similar to the formation



Each node keeps a count of the attribute−values of the examples explained by that node, which can be summarized by a grid:

	chol	trig	hdl
low	1	1	5
med	3	6	5
high	10	7	4
	14	14	14

Each column sums to the number of examples explained by the node

1. The new example is automatically explained by the root node (A); check category utility for adding example to each child of root to find the best child.

2. The best child is the second child (B); also consider adding new example as singleton, merging two best children, and splitting best child into two.

3. The category utility of the tree would be most improved by a "split" of the best child. Since this node has two children (C and D), this is equivalent to"promoting" the two children nodes to be children of the root node.

4. The best match is with the first of these two nodes, so the example is added to node C, and this becomes the current node; check the category utility for adding the example to each of the three children.

5. Check the category utility for creating a new singleton (leaf node), merging two best children, and splitting best child into two. The highest category utility of these is found to be creating a new leaf node, so the example is added there and the recursion terminates.

Figure 3.8: An illustration of Cobweb, showing how the addition of a new example causes node B to "split".

of the tree, but the structure of the tree is not altered. Instead, the best child is chosen at each level of recursion, which stops when the example is placed in a leaf node (containing one example). The class of the leaf node is then the predicted class of the new example.

It is possible to read off probabilistic rules defined by the descriptions of the parents of leaf nodes. For example, in Figure 3.8, the rule for node C in the new tree would be: (Cholesterol: 75% high, 25% low), (Triglycerides: 100% high), (HDL: 50% low, 50% medium). This is perhaps helpful in terms of our human understanding of the classification tree, but is not recommended for further classifications. Proper classifications with Cobweb are best obtained by filtering examples through the tree.

Though Cobweb is sensitive to the order of the data, the merging and splitting operations help lessen the effects of this.

3.3.5 Random generation and hillclimbing

Two simple approaches were implemented primarily as a comparison against the GA: random string generation and hillclimbing.

3.3.5.1 Components of random generation and hillclimbing procedures

The basic components of random generation are:

An evaluation measure. This function is used to compare strings; it is selected specifically for the application.

A generator. This is some method of creating random strings.

The basic components of hillclimbing are:

An evaluation measure. This function is used to compare strings; it is selected specifically for the application.

A climbing mechanism. This is some method of generating the "neighbors" of the current solution.

3.3.5.2 Algorithms for random generation and hillclimbing procedures

In random string generation, a number of strings were randomly generated. The number of strings generated is equal to the number of trials per GA run times the number of GA runs per experiment. With 10,000 trials (a total of 10,000 strings evaluated before the GA terminates) and 100 runs, 1,000,000 random strings are generated and evaluated; the best string found is compared to the best found by the GA.

In the hillclimbing generation, a random string is generated, and is initially the current string. All one-step neighbors are compared against the current string; the best of these replaces the current string; the process repeats until none of the one-step neighbors are better than

	Supervised	Symbolic	Deterministic	Incremental	Partition
	or	or	or	or	dataset
	Unsupervised	Probabilistic	Stochastic	Nonincr.	or Not?
	learning	rules	algorithm	learning	
GA's	S	S	S	N	N
D. Trees	S	S	D	N	P
A'class	U	P	S	N	N
Cobweb	U	P	D	I	P
Random	S	S	S	N	N
Hill	S	S	S	N	N

Table 3.1: A coarse overview of the differences between systems evaluated in this thesis.

the current string. (One-step neighbors are those that are identical to the current string in all but one attribute; the value for the one attribute that differs may be any of the remaining possible values for that attribute, including "don't cares".)

3.3.5.3 Results from random generation and hillclimbing runs

These methods do not produce rule sets, but instead look for the single best rule, and this is compared to the single best rule found by GA runs in a comparable number of evaluations. As with GA's, these approaches could be extended to produce rule sets, but initial results indicated that such an exercise would be tedious and fruitless.

3.4 Comparison of Machine-Learning Systems

The systems implemented in this research differ in the representation they use, and in the assumptions that drive the design of the system. The syntactic structure of the representation used is not terribly important (for example, the fact that the GA represents a rule as a bit string instead of using literal attribute and value names). However, the *semantics* of the representation used may have a large effect in terms of the kinds of solutions the system can discover.

The different assumptions behind each system are also relevant, for example, whether the system does supervised or unsupervised learning, and whether it is incremental or nonincremental. The differences in representation and assumption for each of these systems is summarized in Table 3.1.

Additional assumptions of individual systems that are not reflected in the table include the following:

- Most decision tree systems consider attributes one at a time.

- Autoclass assumes that the attributes are independent and can be modeled as multidimensional Gaussians.
- Cobweb was designed to learn categories that are useful for predicting unknown properties of examples.

One reason GA's were favored for this research initially is that they are more flexible in terms of the models that may be implemented by the system. The other systems described here may be modified to an extent to suit our assumptions, but not necessarily to the point that the underlying assumptions do not seem to interfere with our intended task.

Because the systems implemented in this research differ in their representations and underlying assumptions, comparing them on any one dimension is sometimes "like comparing apples and oranges". However, the overall systems can still be compared in terms of their utility as an approach to the common-disease research task.

The next chapter will discuss metrics to be used for the classification task presented in this research; following chapters will describe the specific implementations of the various systems and the results obtained with each. Chapter 8 will then overview the performance observed with all systems.

CHAPTER 4

DESIGNING NEW METRICS FOR THE EVALUATION OF RULE SETS

Almost universally, the measure of the quality of a rule set in machine learning research has been its ability to classify examples that were not in the original training data. This measure, often called prediction accuracy, is quite useful for comparing one machine learning system to another on a particular task. However, it is not appropriate to all classification tasks.

This chapter will discuss the inappropriateness of simple prediction accuracy for our classification task, suggest additional criteria for evaluating a rule set, present a method for combining conflicting criteria into a single metric, and illustrate this metric applied to the task of classifying individuals with a family history of CAD.

4.1 Shortcomings of Prediction Accuracy

One of the great advantages of prediction accuracy is that it is a simple metric that can be applied to nearly any classification system and nearly any classification task. However, this does not imply that it's a *good* metric for all tasks. With the common-disease research task in this thesis, prediction accuracy fails in two primary ways:

1. It penalizes all classification errors equally, and does not allow for assymetry between the classes.
2. It fails to acknowledge or incorporate our desire for smaller rule sets.

These concerns will be discussed in more detail in the following sections.

4.1.1 All errors are not necessarily equivalent

One problem with the simple prediction accuracy metric is that it penalizes all classification errors equally: An instance of class A that is classified as class B is considered an equivalent error to an instance of class B that is classified as class A. However, this assumption of symmetry in the classification task is not always an appropriate assumption to make. For example, with the well-know "mushroom" dataset [87, 105], classifying an edible mushroom as poisonous may be an error, but classifying a poisonous mushroom as edible could be fatal.

As discussed in Section 3.2, the common-disease research task investigated in this thesis is assymetrical in that a *correct* description of the positive examples in the data set is more important than a *complete* description of the positive examples. The degree of this assymetry is not as severe as the poisonous mushroom example given above; when eating mushrooms, one would likely want to avoid any that couldn't be classified with 100% certainty as being nonpoisonous. In our case, we prefer to avoid classifying negative examples as positive, but doing so will not be a a "fatal" mistake.

In this research, we are primarily looking for descriptions of those individuals who have a family history of CAD. We would like to know which risk factors (or combinations of risk factors) are associated with a family history of CAD so that we might gain some insight into what characteristics of people with a family history of disease set them apart from those who do not have a family history of disease.

Recall that we anticipate that a multiple-rule (disjunctive) description of these examples will be more correct than a single-rule description. When we say that we prefer correctness over completeness, we mean that (to some extent) we would like the descriptions that we find to be as good as possible, even though there may be subsets of the positive examples that remain undescribed. We don't necessarily need descriptions of the people who have no family history of disease; our emphasis is on describing the attributes that distinguish the people who have a family history of CAD from those who have no family history.

A further complication in this task is that we don't have a 50-50 split of positive and negative examples in the dataset. In this sense, it is somewhat "easier" to classify the negative examples. This issue is more easily illustrated in a hypothetical dataset of 90% negative examples — with one rule that says that all examples are negative, the classification can be 90% correct. But this does not provide any insight into the classification task. With our dataset, we can get 56% correct with one rule that labels all examples as negative.

4.1.2 Smaller rule sets are sometimes preferable

The second problem with prediction accuracy is that it ignores our interest in smaller rule sets. The criterion that the rule set be as small as possible might be taken for granted by many machine learning researchers until they grasp what is actually intended here. This is not quite "Occam's Razor" [10] as typically discussed in machine learning literature; as usually considered, the Occam's Razor problem is to find the smallest rule set of maximal correctness (however correctness is defined). In our work, we are willing to have less correct rule sets if they are small. Criteria for "correct" and "small" will be discussed in Sections 4.3.1 and 4.3.2, along with methods of formalizing the tradeoff between the two.

Sacrificing the correctness of a rule set for the sake of its simplicity is not usually even considered in the machine learning community, and therefore must be defended here. (Iba, Wogulis, and Langley provide an exception in [58], which discusses experiments in the use of a weighting function to trade off between correctness and simplicity. Bohanec and Bratko in [11] consider pruning decision trees to increase simplification at the expense of accuracy.) For the purposes of illustration, let the correctness of a rule set be the percent of correctly classified positive examples, and let the simplicity be the inverse of the number of rules (one over the number of rules). A one-rule set that explains exactly the positive examples is ideal (highest possible correctness, highest possible simplicity), but unlikely in our domain. A one-rule set that explains every example (both positive and negative) in the data set is low correctness, but is high simplicity; a rule set that contains one rule for each (positive) data point is 100% correct, but will contain 254 rules (lowest possible simplicity). Of course, smaller rule sets that are 100% correct are also possible; for example, the ID3 algorithm finds 131 rules to explain the 254 positive examples with 100% correctness (a total of 266 rules to explain all 573 examples).

ID3 and other decision tree systems will be discussed in detail in Chapter 6. My reason for "showing my hand" at this point is to give a concrete example, using a well-known system, that finds a rule set that is 100% correct, but that we do not consider to be very "good". Some of the reasons that this is not a good rule set are discussed in the following paragraphs.

- The rule set found by ID3 averages slightly less than 2 examples per rule. This does not lead to any sense of statistical confidence in the rule set. It is not likely that such a rule set would be very good at prediction, either.

- 131 rules is far too large for a human to internalize; furthermore, a rule set that large gives us no understanding of the relevant factors and their interactions. Though this emphasis on understanding the rules is not common to all classification tasks, it does exist in many. For example, the designers of CART [13] argue that "An important criterion for a good classification procedure is that it not only produce correct classifiers (within the limits of the data) but that it also *provide insight and understanding into the predictive structure of the data*" (emphasis in the original).

- An additional consideration is that noise has to be expected in this domain. For one thing, the dataset is based on a single observation for each individual, and may differ from the average values expected for that person. Secondly, clinical errors are also a possibility. A large and highly correct rule set is suspect; one is reminded of fitting points on a curve by using higher and higher order polynomials. Often, it's better to approximate.

4.2 Metrics Commonly Used in Statistics and Epidemiology

The fields of statistics and epidemiology regularly use metrics that can be applied to our task. This section contains a description of some of the more relevant of these approaches. Like prediction accuracy, none of these metrics is fully suited to our task. However, they provide many useful concepts which will be useful in constructing our metric.

4.2.1 A brief description of traditional statistical tools

There are many functions used by statisticians and epidemiologists to evaluate the quality of a test for describing a dataset. As used in epidemiology, for example, these tests might be used to evaluate how good a test is for identifying individuals who have been exposed to (or have contracted) a particular disease.

With a binary classification task, the first step in evaluating a rule set is to partition the dataset into four groups of data points, based on which class the data point belongs to, and which class the rule set assigns to the data point.[1]

		Data Set	
		$D+$	$D-$
Rule Set	$R+$	a	b
	$R-$	c	d
		$a+c$	$b+d$

The data is divided into two groups ($D+$ and $D-$) based on whether each data point is a positive or negative example of the class. It is also divided into two groups ($R+$ and $R-$) based on whether the rule set classified it as a positive or negative example. The resulting four groups, a, b, c, and d, correspond respectively to: true positives, false positives, false negatives, and true negatives. A rule set that describes the dataset perfectly will have both b and c equal to 0.

Note that $a+c$ is the number of positive examples (those in $D+$); $b+d$ is the number of negative examples (those in $D-$); $a+b$ is the number of examples classified as positive by the rule set (those in $R+$); and $c+d$ is the number of examples classified as negative by the rule set (those in $R-$).

The metrics described below come from the literature in statistics and epidemiology (see, for example, Kramer [65]). Each is defined in terms of a, b, c, and d, and illustrated for our dataset. There are a total of $a+b+c+d = 573$ examples in our data set; 254 of these are positive examples, and 319 are negative examples. Once the dataset is fixed, with a given number of positive and negative examples, these metrics can be described exclusively in terms of a and b. That is, $c = 254 - a$, and $d = 318 - b$.

Sensitivity Sensitivity is a measure of the number of examples in $D+$ actually classified as positive by the rule set; it is defined as $\frac{a}{a+c}$. Using conditional probabilities this is $P(R + |D+)$. Sensitivity is illustrated in Figure 4.1. Note that this metric is independent of b (and d).

Specificity Specificity is a measure the number of examples in $D-$ actually classified as

[1] The classification task approached in this thesis is binary; the examples in the dataset are one of two possible classes. The metrics discussed in this section will address only binary classification tasks, but most of the methods discussed here are extendable to multiple-class classification tasks.

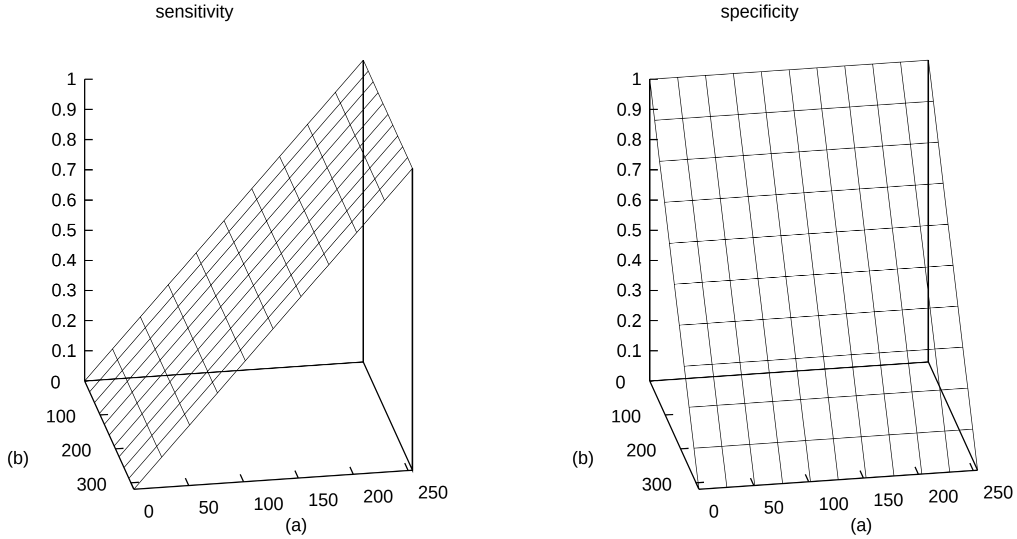


Figure 4.1: Graphs of sensitivity and specificity for a dataset with 254 positive examples and 319 negative examples. For example, sensitivity increases with the number of true positives, a, and is insensitive to the number of false positives, b. False negatives, c and true negatives d are implicitly represented on these graphs since $c = 254 - a$ and $d = 319 - b$.

negative by the rule set; it is defined as $\frac{d}{b+d}$. Using conditional probabilities this is $P(R - |D-)$. Specificity is illustrated in Figure 4.1. Note that this metric is independent of a (and c).

Accuracy Accuracy combines sensitivity and sensitivity, weighing the contribution of each (respectively) by the percent positive examples (*PPE*) and percent negative examples (*PNE*) in the dataset. It is defined as $\frac{a+d}{a+b+c+d}$, which is derived as follows:

$$\begin{aligned} &\mathit{sensitivity} * \mathit{PPE} + \mathit{specificity} * \mathit{PNE} \\ &\quad = \left(\tfrac{a}{a+c} * \tfrac{a+c}{a+b+c+d}\right) + \left(\tfrac{d}{b+d} * \tfrac{b+d}{a+b+c+d}\right) \\ &\quad = \left(\tfrac{a}{a+b+c+d}\right) + \left(\tfrac{d}{a+b+c+d}\right) \\ &\quad = \tfrac{a+d}{a+b+c+d} \end{aligned}$$

Accuracy is illustrated in Figure 4.2.

Odds Ratio The "odds" of an example being in a particular class is the ratio of the probability that it belongs to that class to the probability that it doesn't belong to that class. For example, the odds that a given datapoint in $D+$ is described by $R+$ is $\frac{a}{c}$. Likewise, the odds that a given datapoint in $D-$ is described by $R+$ is $\frac{b}{d}$. The odds ratio takes the ratio of these two odds: $\frac{\frac{a}{c}}{\frac{b}{d}} = \frac{ad}{bc}$, the ratio of the odds that $R+$

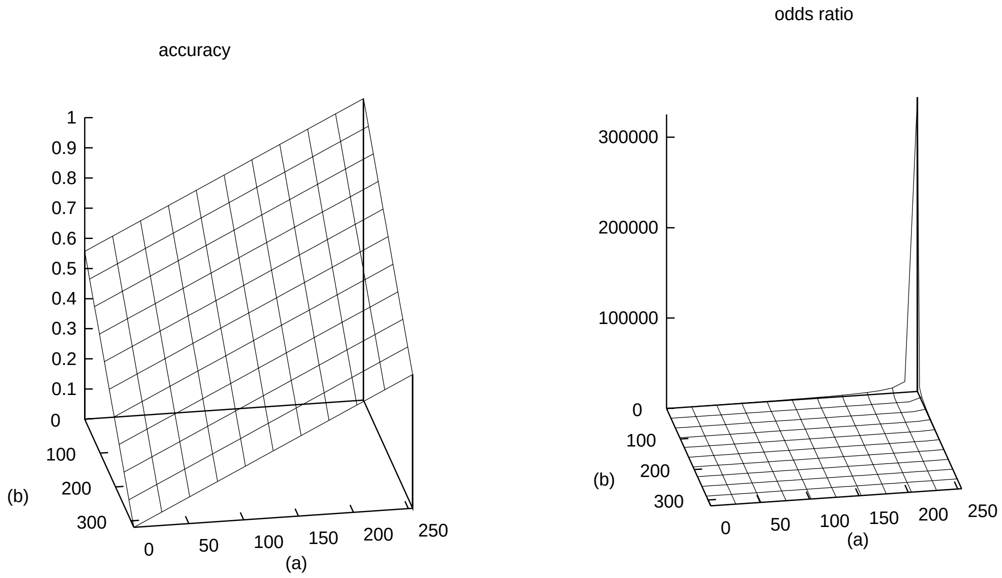


Figure 4.2: Graphs of accuracy and odds ratio for a dataset with 254 positive examples and 319 negative examples.

describes a positive example to the odds that $R+$ describes a negative example.[2] Odds ratio is illustrated in Figure 4.2.

4.2.2 Problems with the traditional statistical approaches

The metrics discussed above all focus on different aspects of the quality of the rule set as a description of the dataset, consequently, they show different strengths and weaknesses with regard to our task. Sensitivity is flawed for our uses because it is independent of the number of false positives (b), although it is good in that it gives higher values to rule sets that explain more positive examples. Likewise, specificity is flawed because it is independent of the number of true positives (a), although it is good in that it gives higher values to rule sets that explain fewer negative examples.

Accuracy and odds ratio are also problematic, but the problems they present for us are more subtle. The problem with accuracy is that it is linear: a rule set with $a = 200$ and $b = 0$ has the same value as one with $a = 253$ and $b = 53$ (in both cases, $a + d = 519$, the number of correctly classified examples from the total of 573). The reason that this is undesirable is the assymetry of our task: it is more important that the rule set explain positive examples well than it is that the rule set explain the positive examples *completely*. For our uses, we

[2] To eliminate problems such as division by 0, 0.5 is added to each of a, b, c, and d.

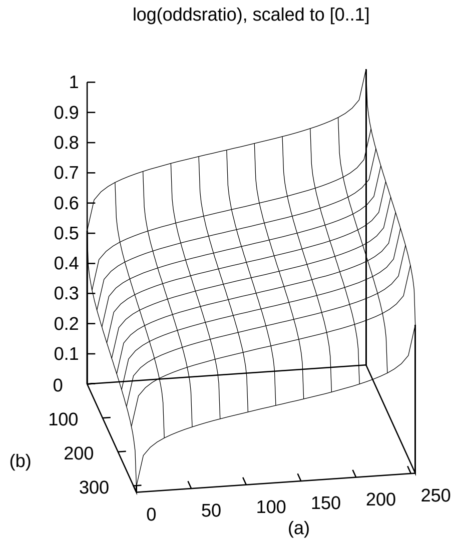


Figure 4.3: Graph of *log(odds ratio)* for a dataset with 254 positive examples and 319 negative examples, scaled so that it falls on the [0..1] interval.

would generally prefer the $a = 200, b = 0$ rule set to the $a = 253, b = 53$ rule set (contingent upon the number of rules in each, which will be discussed in the following sections).

At first glance, odds ratio seems problematic: When we look at its graph (in Figure 4.2), it is nearly flat with one extremely large spike. However, the metric is actually quite useful for our task, and can be made more palatable with a simple modification. By taking the logarithm of this function, the sharp spike is tempered, and we are able to better see the variation in this function (note that this transformation has no effect on the rankings of the evaluations, but only on the magnitude of each). This metric is useful for our task because it balances the tradeoff between describing the positive examples correctly and describing them completely, and does so using a metric that is already well accepted by the medical community. The *log(odds ratio)* function is illustrated in Figure 4.3.

Figure 4.4 presents a cross section of the *log(odds ratio)* graph. From this illustration, we can see that there is a sharp dropoff in the evaluations of rule sets with $b > 0$ as compared to those with $b = 0$ (note that the $b = 10$ curve is halfway between the $b = 0$ curve and the $b = 159$ curve).

Incorporating any negative examples into the rule set comes with a sharp penalty. The emphasis on describing the positive examples completely can be also seen in this figure; there is also a sharp dropoff in the evaluations of rule sets with $a < 254$ as compared to those with $a = 254$. Omitting any positive examples also comes with a sharp penalty.

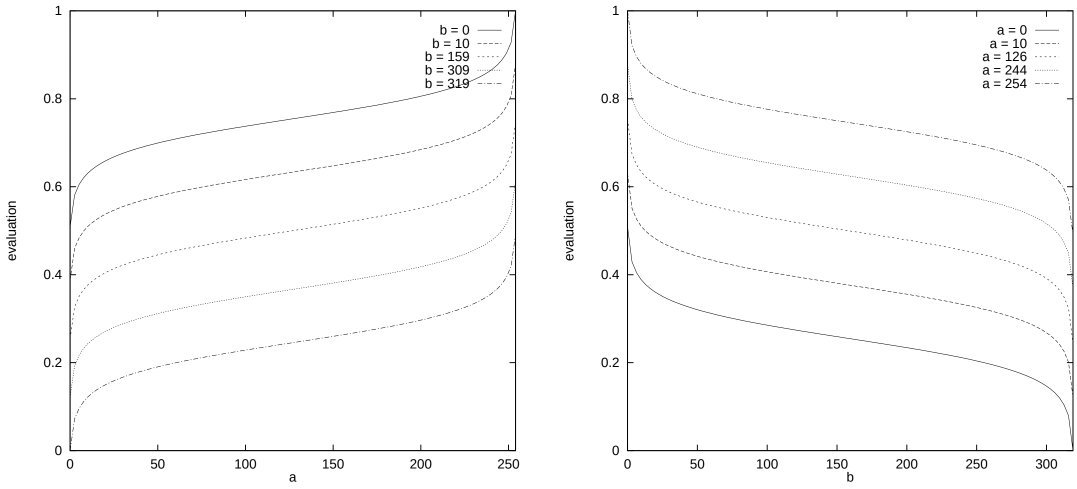


Figure 4.4: Two-dimensional illustrations of the scaled *log(odds ratio)* function, showing the "ledge" that occurs near $b = 0$ and near $a = 254$, respectively. (Note, for example that the difference between $b = 0$ and $b = 10$ is roughly the same as the difference between $b = 10$ and $b = 159$.)

4.3 Designing an Appropriate Metric for Our Task

The full list of desiderata for this task were given in Section 2.3, but these can be distilled to two primary concerns:

1. that the rule set explain as many positive examples as possible, while excluding as many negative examples as possible and
2. that the rule set be as small as possible, or to look at it another way, that the average number of examples explained by each rule is maximized.

Making the tradeoff between these two criteria will be discussed in this section. Because our desiderata are at odds with each other, we need an evaluation criterion that incorporates these two conflicting concerns. There are three main approaches used to combine different functions into a single metric:

addition: In this approach, the competing metrics are simply added together (or equivalently, averaged).

multiplication: In this approach, the competing metrics are multiplied together.

weighting: In this approach, each competing metric is multiplied by a weight; these terms are then added together.

Typically, when using any of these approaches, each competing metric will first be scaled to be on the [0..1] interval; when using weighting, there is usually a requirement that the weights sum to 1. For example, using a weight w, we would combine a function that describes the first criterion, called $criterion1(a, b, ...)$, and one that describes the second criterion, called $criterion2(x, y, ...)$, as:

$$weighted(a, b, ..., x, y, ..., w) = w * criterion1(a, b, ...) + (1 - w) * criterion2(x, y, ...)$$

Note that averaging two criteria is equivalent to weighting them, with a .5 weight given to each criterion.

Simple addition and multiplication do not work well for our task because they allow for only equal contributions of the two criteria. The weighting approach is more appropriate because it allows us to examine the effects of varying the contribution of each criterion (by changing the value of w). To use this function, we need two component functions to describe our criteria. These functions will be presented in the following sections.

4.3.1 Measuring the error of the rule set ("correctness")

The *log(odds ratio)* function presented in Section 4.2.2 embodies our first criterion (that the rule set explain as many positive examples as possible, while excluding as many negative examples as possible), and when scaled to be on the [0..1] interval, is suited for use with the weighting function.

Note that there are many possible functions that could have been used for this purpose, and I do not wish to argue that the *log(odds ratio)* function is the single best one to choose. However, it seems the best to choose from among those readily familiar to the medical community, including accuracy (which is a metric that is also familiar to the machine learning community).

4.3.2 Measuring the size of the rule set ("simplicity")

Our second criterion is that the rule set be as small as possible. The simplest metric for the size of the rule set is the number of rules, r; to make smaller rule sets have higher value (and to put the function on the [0..1] range), we might invert this to $\frac{1}{r}$. However, this function is exponentially decreasing: one rule is twice as highly rated as two rules, etc, and we do not intend to penalize two-rule sets so severely.

As an alternative, observe that the maximum number of rules needed to explain 254 people is 254, and the minimum is 1. A linear metric of the size of the rule set is, therefore $(254 - r)$; to bring this into the [0..1] range, we divide by the number of possible values, 253: $\frac{(254-r)}{253}$. This metric will be called the *simplicity* of the rule set. (Of course, it is possible to have a rule set with more than 254 rules to explain 254 examples, but such rule sets will be given the lowest value of 0.0.)

The $\frac{1}{r}$ function and the *simplicity* function are illustrated in Figure 4.5

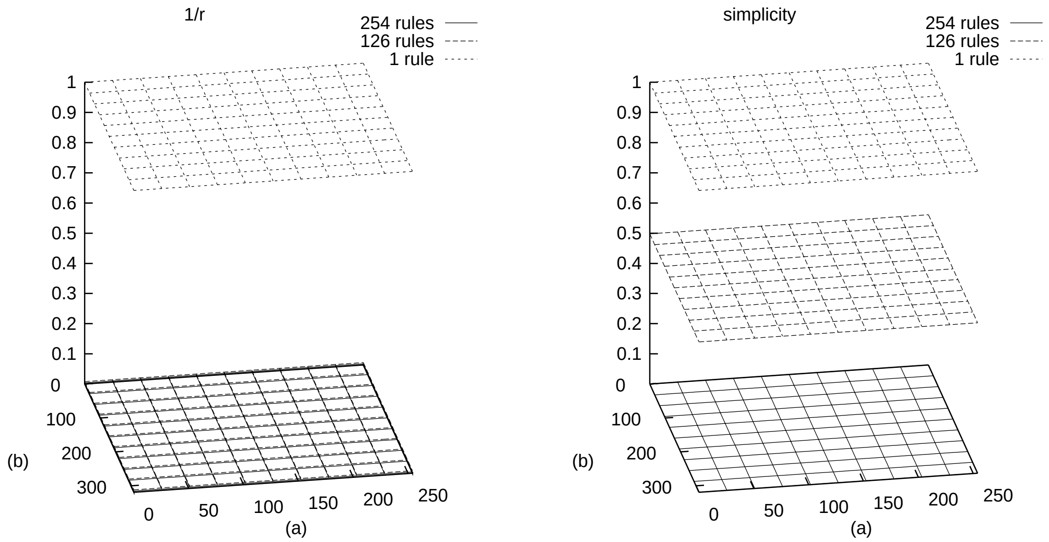


Figure 4.5: The $1/r$ metric and the *simplicity* metric, for a dataset of 254 positive examples, are both independent of the values of a and b. Graphs are shown for $r = 1$, $r = 126$, and $r = 254$. Note that in the graph of $1/r$, the surfaces for $r = 126$ and $r = 254$ are nearly the same.

4.3.3 Combining correctness and simplicity

As mentioned in Section 4.3, our two conflicing functions (*correctness* and *simplicity*) can be combined using a weighting function. With this approach, the generic function:

$$weighted(a, b, ..., x, y, ..., w) = w * criterion1(a, b, ...) + (1 - w) * criterion2(x, y, ...)$$

becomes:

$$evaluation(a, b, r, w) = w * correctness(a, b) + (1 - w) * simplicity(r)$$

The weight, w, is on the range [0..1]; when $w = 1.0$, *evaluation* is the same as *correctness*, and when $w = 0.0$, *evaluation* is the same as *simplicity*. The *evaluation* function is illustrated for sample values of $w = .25$, $w = .5$ and $w = .75$, and for rule set sizes of 1 and 254, in Figure 4.6. For each fixed value of w, we have a four-dimensional function, which is graphed on a three-dimensional grid; therefore, each value of the number of rules is graphed on a separate surface. Note that as w increases, these surfaces move closer together.

4.4 Discussion of the New Metric

The metric put forth in Section 4.3.3 for evaluating the quality of a rule set can be looked at as a five-dimensional function; I find it less confusing to consider it a family of four-dimensional

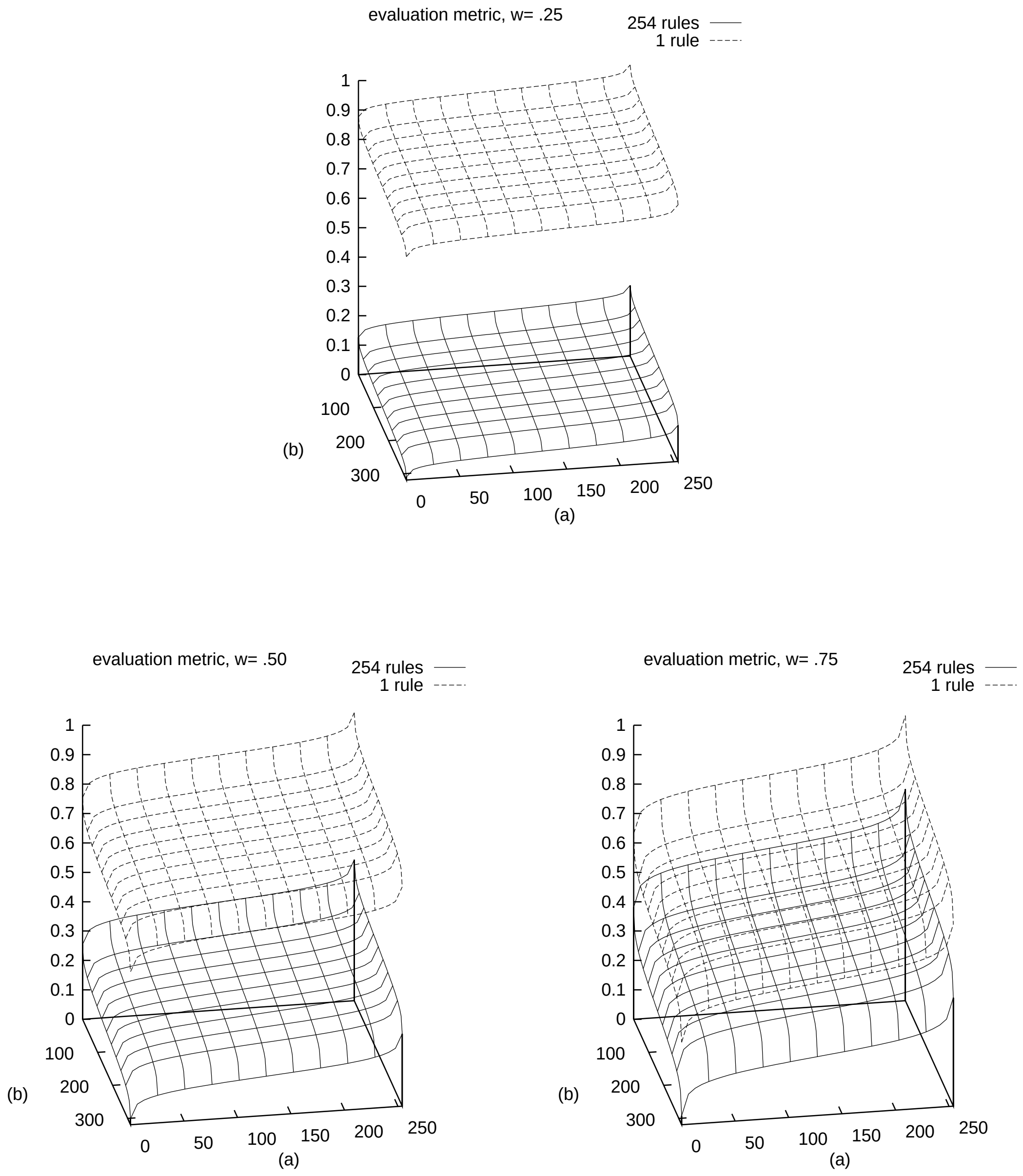


Figure 4.6: The proposed *evaluation* function for weights .25, .5 and .75. Graphs are shown for $r = 1$ and $r = 254$.

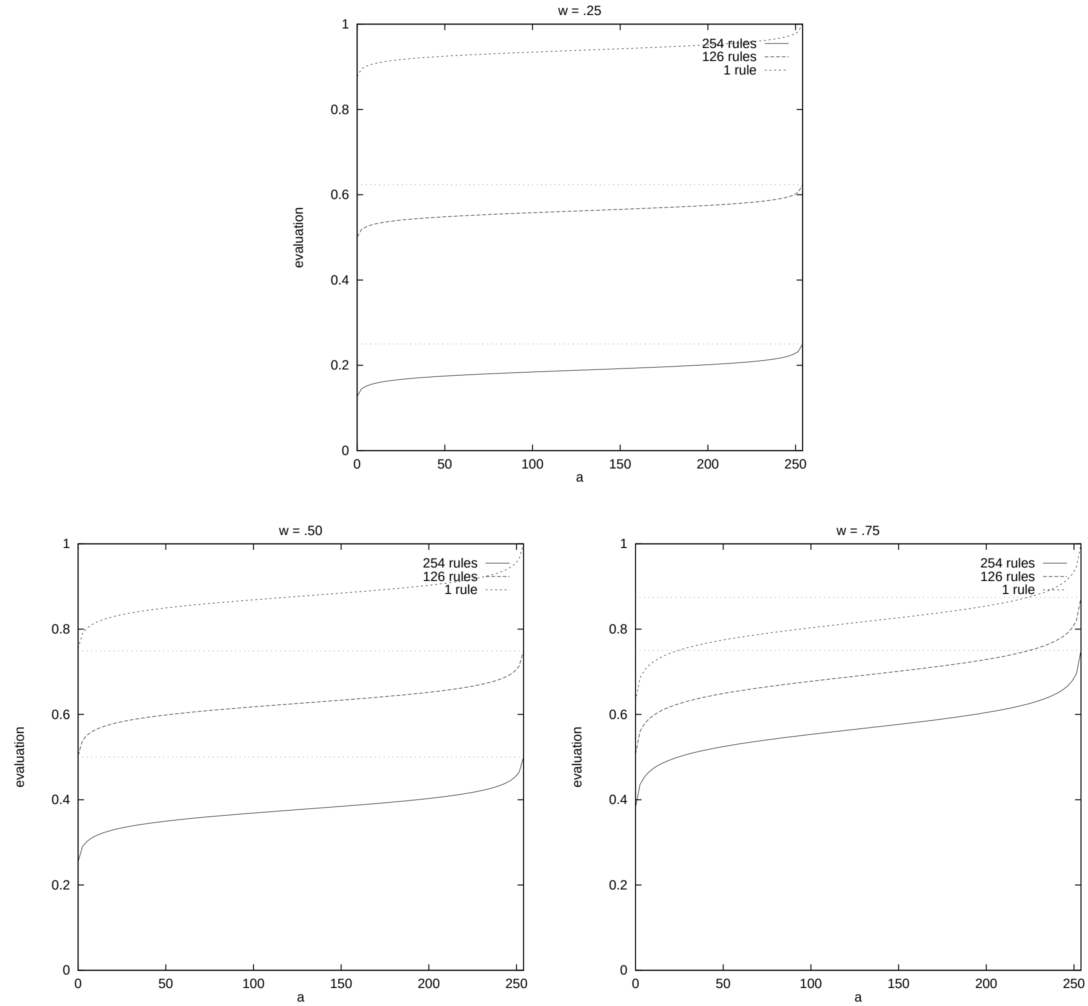


Figure 4.7: Two-dimensional illustration of the proposed *evaluation* function for weights .25, .50 and .75, with b fixed at 0. Graphs are shown for $r = 1$, $r = 126$, and $r = 254$. The horizontal dotted lines show how the *evaluation* value of explaining all the examples compares to the other curves. For example, with $w = .50$, explaining all 254 examples with 126 rules has roughly the same *evaluation* as explaining only 1 example with 1 rule.

functions (a different function for each value of w). Each four-dimensional function consists of 253 different layers of three-dimensional functions — there is one layer for each different value of r. Figure 4.6 illustrates these layers for the extremes of $r = 1$ and $r = 254$; showing every possible value of r would make the graphs unreadable.

When $w = 1.0$, the function is independent of the number of rules in the rule set, and there is only one layer (as illustrated in Figure 4.3); As w gets smaller, the *log(odds ratio)* shape is flattened, until when $w = 0.0$, the function is entirely dependent on the number of rules, and appears as a series of planes (as illustrated in Figure 4.5).

rule set	weights				
	1.0	0.75	0.50	0.25	0.0
200-0, 1 rule	.8058 (1)	.8543 (1)	.9029 (1)	.9514 (1)	1.000 (1)
200-5, 1 rule	.7107 (7)	.7830 (7)	.8554 (7)	.9277 (5)	1.000 (1)
200-10, 1 rule	.6846 (10)	.7635 (10)	.8423 (10)	.9211 (9)	1.000 (1)
195-0, 1 rule	.8013 (5)	.8510 (2)	.9007 (2)	.9503 (2)	1.000 (1)
195-5, 1 rule	.7063 (8)	.7797 (8)	.8531 (8)	.9266 (6)	1.000 (1)
195-10, 1 rule	.6801 (11)	.7601 (11)	.8401 (11)	.9200 (10)	1.000 (1)
190-0, 1 rule	.7971 (6)	.8479 (4)	.8986 (3)	.9493 (3)	1.000 (1)
190-5, 1 rule	.7021 (9)	.7765 (9)	.8510 (9)	.9255 (7)	1.000 (1)
190-10, 1 rule	.6760 (12)	.7570 (12)	.8380 (12)	.9190 (11)	1.000 (1)
200-0, 5 rules	.8058 (1)	.8504 (3)	.8950 (4)	.9396 (4)	.9842 (10)
200-0, 10 rules	.8058 (1)	.8454 (5)	.8851 (5)	.9248 (8)	.9644 (11)
200-0, 15 rules	.8058 (1)	.8405 (6)	.8752 (6)	.9099 (12)	.9447 (12)

Table 4.1: Sample rule sets and their *evaluation*, for different values of w. The evaluations should be compared only in relation to others with the same value of w, i.e., those in the same column.

Figure 4.7 illustrates cross sections of the graphs from Figure 4.6, with the value of b held constant at $b = 0$. These graphs illustrate that when w=.25, the highest values are for small rule sets, but with w=.75, the highest values are with large a's. For example, with $w = .25$, a one-rule set with $a = 0$ and $b = 0$ has an evaluation of .875, while a 50-rule set with $a = 254$ and $b = 0$ has an evaluation of .855 (the former is considered a better rule set; a rule set with $w = .25$ must have fewer than 44 rules to be better than .875). But when $w = .75$, a one-rule set with $a = 0$ and $b = 0$ has an evaluation of .625, while a 254-rule set with $a = 254$ and $b = 0$ has an evaluation of .750 (the latter is considered a better rule set).

Table 4.1 illustrates how the ranking of some example rule sets vary as the value of w is varied. Twelve example rule sets are shown, with their a and b values, and the number of rules. Each column of the table corresponds to a different value of w; the value assigned by the *evaluation* function is given, along with the ranking of the rule set compared to other example rule sets for that value of w.

Each column lists the *evaluation* of the rule set, followed by the ranking of that rule set within the column (as compared to other rule sets for the same value of w). As we would expect, the 200-0, 1-rule set has the highest fitness for each of the values of w.

The *evaluation* metric will be used in Chapter 5 to evolve rule sets using the GA and in subsequent chapters to compare the GA rule sets to those found by other systems that have their own evaluation functions.

CHAPTER 5

THE GENETIC ALGORITHMS IMPLEMENTATION

As mentioned in Chapter 3, Genetic Algorithms have shown success with complex problems and are expected to outperform other machine learning systems on the task presented here. The main reason for this is that GA's may be able to exploit the interactions between attributes and the risk of disease, whereas the other machine learning systems may ignore interactions. In addition, GA's do not require that the attributes be independent.

In the common-disease-research classification task presented here, we might like to consider all possible pairwise interactions, all possible three-way interactions, etc. However, an exhaustive survey of all possible interactions would be extremely time consuming. (There are $\binom{21}{2} = 210$ possible pairs of attributes, $\binom{21}{3} = 1330$ triples of attributes, $\binom{21}{4} = 5985$ quadruples of attributes, etc.) Any two attributes, each with three possible values, has nine different attribute-value interactions to be considered. The GA, through random processes such as parent selection, mutation, and crossover, investigates only a subset of these possible interactions. However, having discovered an important interaction, it is often able to preserve this pattern into future generations. It is because of the GA's ability to discover and utilize interactions (often referred to as the GA's ability to balance between exploration and exploitation) that we expect it to do well on this task.

GA's were introduced in Section 3.3.1. In the first section of this chapter, I present some of the many possible variations on GA's. Section 5.2 discusses the general approach we have taken; Section 5.3 then discusses in more detail the modifications made to enable the GA to evolve rule sets and to experiment with means of improving these rule sets. Finally, Section 5.4 presents experiments designed to find a good parameter set for the GA.

5.1 Variations on GA Mechanisms and Parameters

There are a number of mechanisms in the GA that may be altered from those described in Section 3.3.1, and number of parameters that may be set in a GA program. This section describes some of these mechanisms and parameters.

Variations in replacement strategy:

Generation Gap — Rather than replacing the entire current population with the offspring population, a percent of the parent population may be preserved from one generation to the next. Typically, the strings that are preserved are chosen randomly; this tends to prevent premature convergence by helping to maintain the diversity of the string population.

Elitism — The elitist strategy is similar to using a generation gap, except that the strings that are preserved are those with the highest fitness. This guarantees that the best string seen in the run will be present in the final population. A common version of elitism is that the single best string may be preserved into the next generation, independently of the generation gap (i.e., there may be no additional strings preserved via a generation gap).

Steady-state GA — Rather than using discrete "generations" (sometimes called a generational GA), a steady-state GA selects two parents and produces two offspring, which are immediately inserted into the population. They may replace the two worst strings in the population, or two random strings.

Variations in parent selection:

Random parent selection — Rather than choosing parents preferentially (based on fitness), parents are chosen randomly (still with replacement). This scheme works only when an elitist replacement strategy is used (that is, the evaluation provided by the fitness function must come in *somewhere*).

Ranking — Rather than using the fitness function evaluation to determine the probability of selecting a string into the parent population, the fitness function evaluation provides a ranking of the strings, and this ranking is used to determine probabilistic parent selection. The chance of being selected as a parent is then proportional to the string's index in population, rather than proportional to its fitness. According to Grefenstette [45], "Ranking helps prevent premature convergence by preventing 'super' individuals from taking over the population within a few generations. However, ranking often produces slower improvement than proportional selection."

Variations in operators:

Mutation Scheme — A mutation is considered by some GA researchers to be selecting a random allele and then changing it to something different; by others, it is considered to be selecting a random allele and then setting it to be a random value (which may be identical to the original value). This distinction is important, but perhaps primarily to those who seek to duplicate other's results.

Crossover Scheme — There exist a wide variety of crossover schemes. In one-point crossover, the two parents crossover at only one point; in two-point crossover, the two parents crossover at two points. In uniform crossover, each location of the string is regarded independently, and a random "coin flip" determines whether or not the strings will swap information at that location.

New operators — It is not unusual for GA researchers to create new operators for forming offspring from parent strings. For example, when the elements in the string represent number ranges, it is common to introduce an operator (similar to mutation) that increments or decrements a value by a small number.

Variations in fitness evaluation:

Windowing — Windowing is a method for increasing the relative variance of the fitness of strings within the population (the term "window" refers to the number of previous generations that are to be considered when scaling the fitness). This approach is thought to increase the selection pressure towards more fit individuals. [45]

Niching — When niching is used, strings are divided into subpopulations (either for the entire GA run or for the purpose of fitness evaluation only), and the fitness of each string is calculated relative to others in the subpopulation. The method is often used with a steady-state GA, for example: a subpopulation of strings is determined, and the two new strings replace the worst of the strings in the subpopulation. This has the advantage of being faster than considering the whole population, and less risky than random replacement (which might replace a high-fitness string).

Variations in other mechanisms:

Variations in representation — Although most GAs describe the elements of the population as bit strings, over the alphabet {0,1}, other representations are possible. For example, real numbers might be used. In this case, mutation changes one real number to another real number, and variations of mutation may be introduced that will mutate a number to a neighboring number.

Immigration — Each generation, a number of randomly generated strings are introduced into the population (taking the place of offspring created via reproduction). (This has the effect of maintaining diversity in the population of strings and possibly helping to prevent premature convergence.)

Seeding initial population — The initial population is often composed of randomly generated strings, but may instead be composed of heuristically chosen structures (or a mix of random and heuristically chosen structures).

Variations in parameter settings:

The parameters to be set at runtime will vary with each GA implementation; this listing describes runtime parameters relevant to the GA used in this thesis research:

Crossover Rate — The percentage of the parent population that undergo crossover.

Mutation Rate — The percentage of elements in a string that undergo mutation.[1]

[1] There could also be a second meaning of mutation rate: the number of parents that undergo mutation, but this does not appear in the software used in this thesis, so this use is omitted to avoid confusion.

Generation Gap — The percentage of strings from generation N that are not passed directly to generation N+1 (i.e., 100 minus the generation gap is the percent that will be saved.) The strings preserved into the next generation are typically chosen at random.

Elitism rate — The number of best strings from generation N that are passed directly to generation N+1. The elitism rate may work with the generation gap (which typically selects random strings), causing instead the best strings to be saved, or may work in addition to the generation gap, causing an additional number of strings to be saved.

Rank minimum — When ranking is used as a parent selection scheme, this parameter sets a lower bound on the value that will be assigned to the worst string in the population.

Scaling window — When windowing is used, this parameter determines the number of previous generations to consider in scaling the fitness of the current generation.

Immigration Rate — When the immigration strategy is used, this is the percentage of strings in each generation that are immigrants.

Random Seed — The seed to the random number generator.

Number of Experiments — The number of distinct runs done with different random seeds.

Convergence criteria — The percent of strings in the current population that must be identical, to signal that the population has converged.

Total trials/generations — The maximum number of trials (fitness function evaluations) to complete or generations to run before the run is terminated (assuming it does not converge first).

Population size — The number of strings in the current population each generation.

Note that many of the rates described above may be altered as the run proceeds. For example, some GA implementations increase the mutation rate as the run proceeds to encourage the system to "jump off" local optima.

5.2 Our GA Formulation

Section 3.3.1 presented a typical GA implementation; Section 5.1 presented several possible variations on this. This section will present the specific implementation of the GA used for this thesis work. The variations of the GA that were explored in the course of the thesis will be explained as the specific experiments are discussed. The GA described here was implemented using the Genesis software package [45], described in Section 5.2.2.

5.2.1 Representation

Each string in the GA population represents a rule to be matched against our data set. There is one bit for each possible value for each attribute: There are six bits for each of the three apolipoprotein genotypes, two bits for each of smoking status and gender, and three

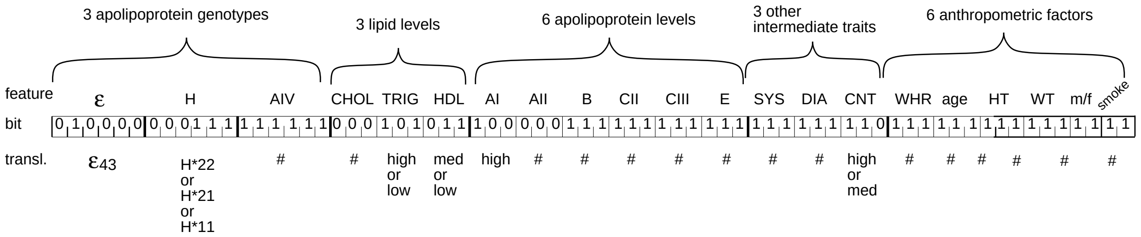


Figure 5.1: An example string, showing the representation of genotypes and other attributes. The first part of the string represents each of the genotypes of the three genes; the second part of the string represents three lipid traits; the third part of the string represents six apolipoprotein traits; the fourth part of the string represents three other intermediate-trait attributes; and the last part represents the six anthropometric trait attributes. The pound sign is used to denote a "don't care" value — the attribute may take on any of its possible values.

Binary	Alleles
100000	ϵ44, H*33, AIV*33)
010000	ϵ43, H*32, AIV*32)
001000	ϵ42, H*31, AIV*31)
000100	ϵ33, H*22, AIV*22)
000010	ϵ32, H*21, AIV*21)
000001	ϵ22, H*11, AIV*11)

Binary	Trait
001	lower tertile
010	middle tertile
100	higher tertile

Binary	Smoking	Gender
01	yes	male
10	no	female

Table 5.1: Translations from binary to symbolic risk-factor values for our GA.

bits for each of the other 16 risk factors. This is a total of 70 bits in each string. An example string and its translation are illustrated in Figure 5.1.

The example string in Figure 5.1 represents those people with APO ϵ43 genotype, an APO H genotype of H*22 or H*21 or H*11, high or low TRIG, medium or low HDL, high APO AI, and high or medium CNT. The other risk factors are all "don't cares" (represented as '#'), which means that these risk factors are not considered when comparing a string to the data set.

Table 5.1 provides translations from binary to "symbolic" risk-factor values. For each genotype or trait, two or more possible values may be logically "ANDed" together, interpreted as an OR in the resulting rule. For example, the string value for the APO H genotype attribute is 011000, indicates that the APO H allele can be either H*32 or H*31. A value of all 1's (111111, 111, or 11, depending on the attribute) is interpreted as a "don't care", that is, the attribute may assume any of the possible values and still match the rule. Likewise, a string of all 0's (000000, 000, or 00, depending on the attribute) is interpreted as a "don't care".

(The string of all 1's can be interpreted as listing all possible values for this attribute; the string of all 0's can be interpreted as an unspecified attribute. Functionally, these two are equivalent.)

This representation results in rules that are "conjuncts of disjuncts". That is, *all* of the attributes-values must be matched for the rule to match a data point, but each attribute-value may match any *one of* the possible values specified in the rule.

5.2.2 Software

We used the software package Genesis [45] for the implementation. To use Genesis, the user provides the system with the string representation and the fitness function; a number of parameters must also be set. Genesis is a highly modular software package, which simplifies the process of adding or changing mechanisms in the GA implementation.

5.2.3 Initial population

We use a mix of 50% "don't care" strings and 50% randomly generated strings as the initial population. Seeding with "don't care" strings (a "don't care" string will match all 573 people) encourages the GA to search from general towards specific rules, rather than vice versa.

Though it is possible to seed the initial population with strings that exactly describe individuals in the data set, we found empirically that this did not help our GA to find high-fitness strings. Instead, what seems to help is a mix of "don't care" strings (every risk factor is a "don't care"), and randomly generated strings. We believe this is due to the complexity of the search space: Crossing over two strings that correspond to two individuals in the data set is likely to produce a "lethal" — a string that describes no one in the dataset.

5.2.4 Fitness function

The weighted *evaluation* function presented in Section 4.3.3 was used as a fitness function. Most experiments were done with $w = 1.0$, that is, the experiments were looking for a good parameter set to find the best rule set, regardless of the number of rules (as is typically done in machine learning research). These experiments are presented in Section 5.4. Once a good parameter set was found, these parameters were used in investigating the effects of varying w, presented in Section 5.4.3.

5.2.5 Parent selection policy

Parents are selected with replacement from the current population, using a ranking strategy. Ranking is thought to slow the effects of convergence, but is also a component of our modification to the GA to evolve rule sets (see Section 5.3.1). Slowing convergence is important in a complex task such as this, with many local optima; it is further helpful when we are trying to find a set of rules, rather than a single optimum, because it helps to maintain diversity in the population.

A .8 rank minumum was used, which means that the lowest-ranked string is expected to be selected as a parent .8 times and the highest ranked parent is expected to be selected as a parent 1.2 times. There are theoretical advantages to setting this value at or above .8 [53], so as to prevent premature convergence.

5.2.6 Operators

The operators used are mutation and two-point crossover. All parent strings are candidates for mutation; the mutation operator determines a bit to mutate (the frequency of this is determined by the mutation rate parameter) and then selects a random 0 or 1 for this bit. (Note that it is possible for a "mutated" bit to be identical to the original bit, so the string may not change during the process of mutation.) The percentage of the parent strings selected to reproduce that undergo crossover is determined by the "crossover rate" parameter. The crossover operator used in this research is two-point crossover; furthermore, the crossover occurs only between risk factors and not within risk factors. The two crossover points are selected randomly, with the restriction that they cannot occur at the same location.

Crossover was restricted to occur only between risk factors to facilitate experiments on varying crossover and mutation rates. Without this restriction, a crossover has an effect similar to a mutation, and the effects of varying the rates are confounded. The default mutation and crossover rates in Genesis are .001 and .6, respectively, which means that each bit of each string has a .1% chance of being mutated, and 60% of the offspring population undergo crossover. Experiments were done to investigate the effects of varying these two parameters.

5.2.7 Replacement policy

A generation gap of 75% was used, where 25% of the population at generation N is preserved into generation N+1 (chosen randomly, without replacement), and the remaining 75% of the population is comprised of offspring formed by the GA operators. This fraction was chosen based on several initial empirical test runs.

5.2.8 Other GA parameters

A population size of 100 was used, 100,000 total trials. The latter parameter means that 100,000 strings will be evaluated before the run ends (barring premature convergence, which will not happen in our formulation for evolving rule sets). Genesis (see Section 5.2.2) always completes a generation before stopping, so there may be more than 100,000 trials in a single run (but not more than 100,075, with 100 population size and .75 generation gap). This translates to roughly 1000 generations.

5.2.9 The random number generator

The GA has a considerable dependence on the random number generator used. A seed for the random number generator determines many of the attributes of the search, and therefore can have a large influence on the portion of the solution space that is searched. The random number generator determines the particular strings created for the initial population as well as the places where crossover and mutation occur each generation; it also plays a role in selection of parents for reproduction.

Moreover, these stochastic effects may be more pronounced on a complex problem with discontinuities (where all of the solutions are not observed in the sample being studied), such as the research problem presented here. Because we are using a stochastic process in a complex space, it is not surprising that runs with different random number seeds may converge to different optima.

Because of this effect and the complexity of our problem, we did 100 runs with each parameter setting. This typically results in 100 different rule sets; in our experiments, we report on the range of values seen in these 100 runs as well as the best of the 100 runs for each experiment.

5.2.10 Summary of parameter settings

Parameter settings for our initial GA experiments included those given in Table 5.2[2].

5.3 Alterations to Genesis

There were two primary modifications made to the Genesis code to pursue the experiments presented here:

1. The code was changed to evolve rule sets, rather than to look for the single best rule.

2. The code was changed to allow experiments with "immigration" of strings not formed by crossover and mutation.

These two modifications are described in the following sections.

[2] The crossover and mutation rates are the default settings in Genesis. The other parameters were determined by several initial runs.

Parameters	Explanations
Population Size = 100	Use 100 strings in the population
Total Trials = 100,000	Evaluate 100,000 strings in the entire run
Experiments = 100	Do 100 runs with different random seeds
Crossover Rate = 0.6	60% of the parents will undergo crossover each generation
Mutation Rate = 0.001	Each bit of each offspring has a 0.1% chance of undergoing mutation each generation
Generation Gap = 0.75	75% of the population is replaced each generation
Rank Min = 0.8	the worst structure in the population will have .8 offspring, on the average
Percent DCares = 50	50 percent of the initial population is "don't care" strings; the remaining 50 percent are randomly generated

Other Options
two-point crossover
crossover only between attributes
evolve rule sets, not single best rule
use elitism: best strings are saved to next generation, not random 25%
use ranking

Table 5.2: Table showing initial parameter settings and options for the GA.

5.3.1 Adding population-based fitness to the GA

The typical GA formulation is designed to find a single solution (an optimization of the fitness function). Development of theory and mechanisms to encourage the GA to evolve a set of coadapted rules, rather than a single optima is a current "hot topic" in the GA community.

While there are a number of approaches that might have been used, the one we've chosen is based on the approach used by Greene and Smith in their COGIN system [44]. Some other approaches are described in Appendix A. The COGIN approach was selected because it explicitly evolves a population of rules that cover different niches, without requiring an explicit statement of the number of rules expected.

We did not use the "Pitt approach" [60] (in which each string represents an entire rule set) because use of the Pitt approach would require strings roughly 75 times longer than our current 70 bits. Our runs to find a *single* rule were already taking a rather long time (approximately 30 minutes for a single run and 2 days for 100 runs with different seeds). Nonetheless, this approach is interesting, and we leave it for future work. The "iterative runs" approach was investigated briefly, but was also prohibitively time consuming. Other approaches were ruled out because of concerns such as that they used syntactic (rather than semantic) measures of similarity between strings or that they required that the number of rules be known ahead of time.

Using the population-based fitness approach

The designers of COGIN call their approach to evolving rule sets "population-based fitness", which means that the fitness of a string is based on the entire population of strings. This is in contrast to the typical GA formulation, which evaluates a string independently of others in the population.

In this approach, each string in the population is first evaluated using the fitness function as would be used by a typical GA; this is called the *raw fitness* of the string. The strings are then sorted according to their raw fitness. Finally, a second fitness is calculated for each string, in an iterative manner. This second fitness is called the *operative fitness*:

1. For the first (highest ranked) string, the operative fitness is the same as the raw fitness.

2. The positive examples explained by the first string are removed from consideration, and the operative fitness of the second string is calculated, using the fitness function.

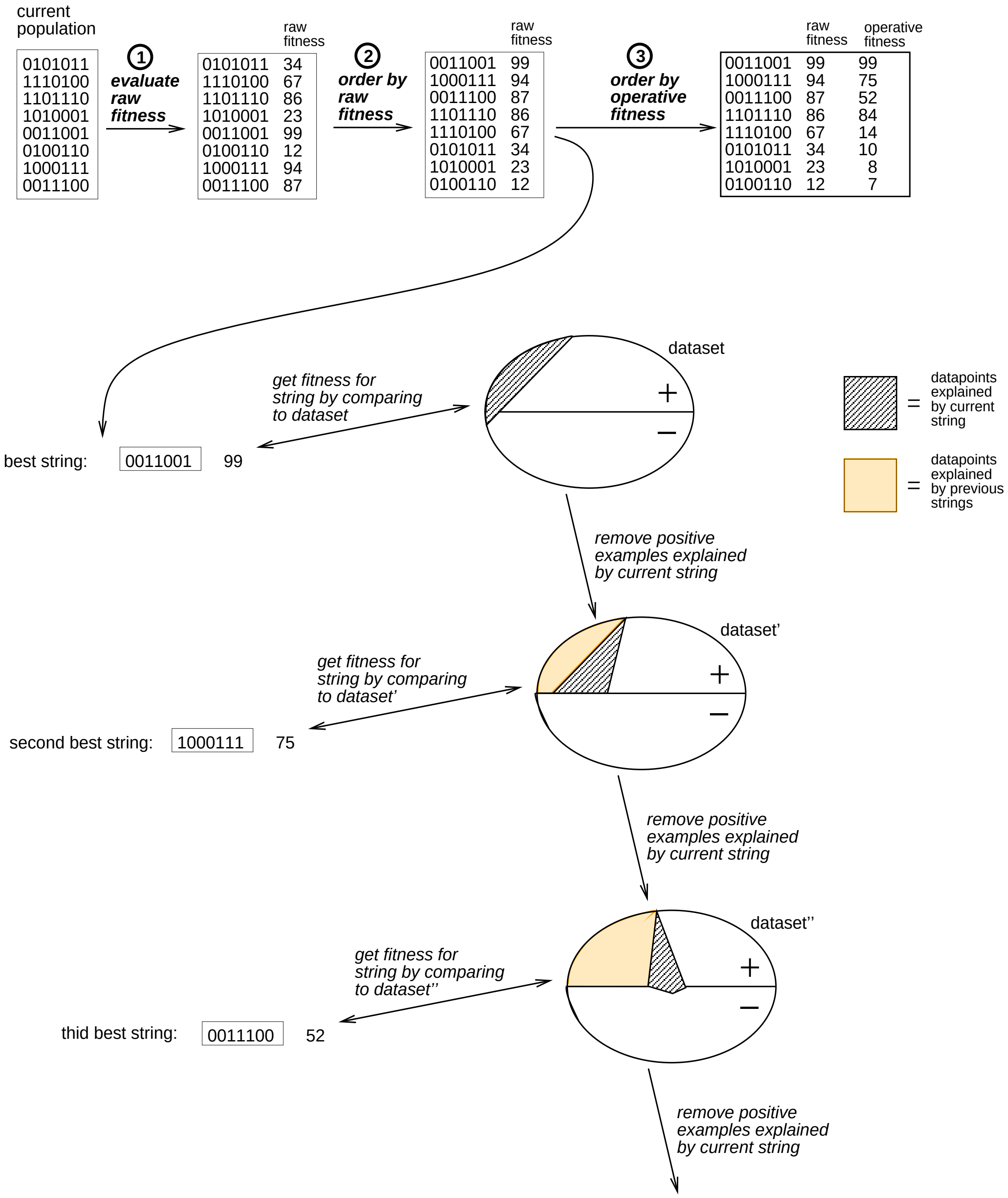
3. The positive examples explained by the second string are also removed from consideration, and the operative fitness of the third string is calculated, using the fitness function.
4. etc., until the operative fitness of all strings has been calculated.

Finally, the strings are ranked according to their operative fitness, and this ranking is used for parent selection. In addition, the best strings are preserved into the next generation, using elitism with the generation gap. With our initial parameters, the 25 best strings are preserved to the next generation. Using population-based fitness with our task is illustrated in Figure 5.2.

Thus, the pool of positive examples to be explained may be decremented by each successive string, and (after the first string), a string does not get "credit" for explaining any positive examples that have been explained by better strings.

In most cases, the ranking according to operative fitness will differ from the ranking according to raw fitness because strings with high raw fitness are likely to explain many of the same examples explained by the string with the highest raw fitness. These strings, therefore, may have much lower operative fitness than raw fitness.

Note that it is nearly impossible for our population-based fitness GA to converge to a single string before the requisite number of trials has elapsed (nearly impossible on our task, that is, where many rules are needed to explain the data). The use of population-based fitness provides a built-in mechanism to encourage diversity within the population of strings and avoid the problem of "premature convergence", where one very high fitness string takes over



current population
0101011
1110100
1101110
1010001
0011001
0100110
1000111
0011100

	raw fitness
0101011	34
1110100	67
1101110	86
1010001	23
0011001	99
0100110	12
1000111	94
0011100	87

	raw fitness
0011001	99
1000111	94
0011100	87
1101110	86
1110100	67
0101011	34
1010001	23
0100110	12

	raw fitness	operative fitness
0011001	99	99
1000111	94	75
0011100	87	52
1101110	86	84
1110100	67	14
0101011	34	10
1010001	23	8
0100110	12	7

1
evaluate raw fitness
2
order by raw fitness
3
order by operative fitness
dataset
+
−
▨ = datapoints explained by current string
□ = datapoints explained by previous strings
get fitness for string by comparing to dataset
best string: 0011001 99
remove positive examples explained by current string
dataset'
get fitness for string by comparing to dataset'
second best string: 1000111 75
remove positive examples explained by current string
dataset''
get fitness for string by comparing to dataset''
thid best string: 0011100 52
remove positive examples explained by current string

Figure 5.2: Using population-based fitness.

the population (effectively terminating the search process prematurely). However, some of the issues associated with premature convergence are still relevant to our GA. For example, the best strings in our GA population are most likely to be selected as parents; when two such strings crossover, it is not unusual for the offspring strings that are formed to explain groups of examples that have already been explained by other strings (perhaps even their parents). Thus, issues such as what Holland calls "the balance between exploitation and innovation" [51, 52] apply to our population-based fitness GA as well.

5.3.2 Adding immigration to the GA

Immigration is an approach developed by Bean and colleagues [5, 7, 6, 46] that may help the GA to maintain diversity in the population of strings. This should help counter the effects of premature convergence, which tends to decrease the diversity of the population of strings.

When using immigration, every N generations, P% of the population are "immigrants", i.e., new strings that are introduced into the population of strings, but not generated by the usual reproduction process. (N and P are parameters set by the user.) When immigration is added, an extra step is added to the basic GA cycle, first illustrated in Figure 3.5. The new cycle is illustrated in Figure 5.3.

After reproduction via crossover and mutation, immigration produces one or more new strings, which are inserted into the population. These strings take the place of strings produced by crossover and mutation, rather than those saved via the generation gap. (The reason for this is that the generation gap saves the best strings from the previous generation, while crossover and mutation may or may not produce good strings.)

For our implementation, we set $N = 1$, so that immigration occurs very generation. We investigated values of $P = 1$ and $P = 5$. We investigated a few variations of immigration, which will be described in the next Section 5.4.2.

5.4 Experiments with Varying the Parameters to the GA

Four basic variations on the GA were investigated in this research:

1. an investigation of the effects of varying crossover and mutation rates.
2. an investigation of the effects of adding immigration.
3. an investigation of the effects of varying w in the weighting function.
4. an evaluation of the prediction accuracy for the best parameter settings.

These experiments are described below.

5.4.1 The effects of varying crossover and mutation rates

The crossover operator is one distinguishing feature of the GA that sets it apart from most other approaches, and so our first experiments investigated the effects of varying the crossover

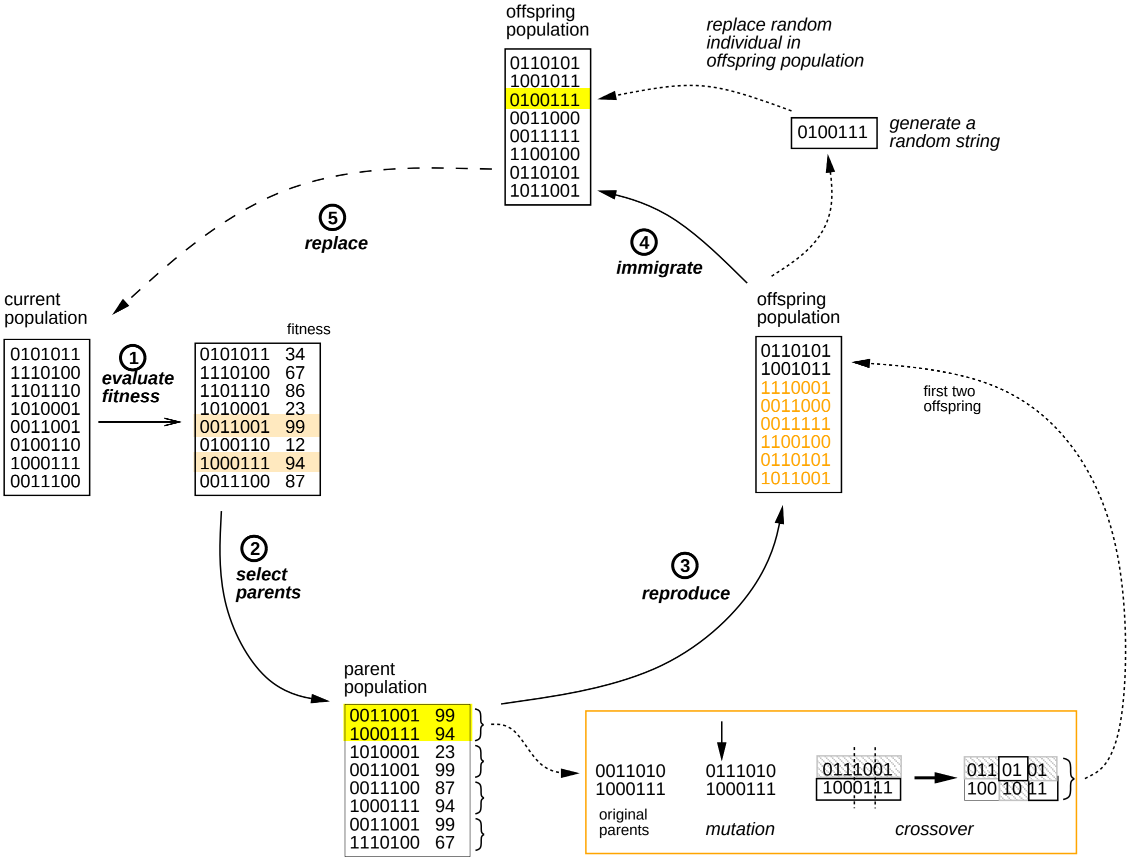


Figure 5.3: The GA cycle with immigration of random strings.

rate parameter. We also recognized that the mutation rate may interact with the crossover rate, so these experiments investigated the effects of different mutation rates as well.

Recall that the default settings in Genesis are .6 crossover rate and .001 mutation, and that we did 100 runs with each variation of the two parameters. We did 15 different experiments, using crossover rates of .25, .50, .60, .75, and .90, and mutation rates of .1, .01, and .001.

For each of these 15 experiments, the 100 resulting rule sets were evaluated. Each rule set contains roughly 25 rules (the number of rules saved via elitism and the generation gap), and explains anywhere from 6 to 174 of the 254 positive examples. The distributions of these runs are illustrated in Figure 5.4.

A rule set is extracted from the final population by ranking the rules and adding each successive rule to the rule set until adding the next best rule lowers the fitness of the rule set. Because the 25 best rules are saved each generation, there will typically be 25 rules in the final rule set.

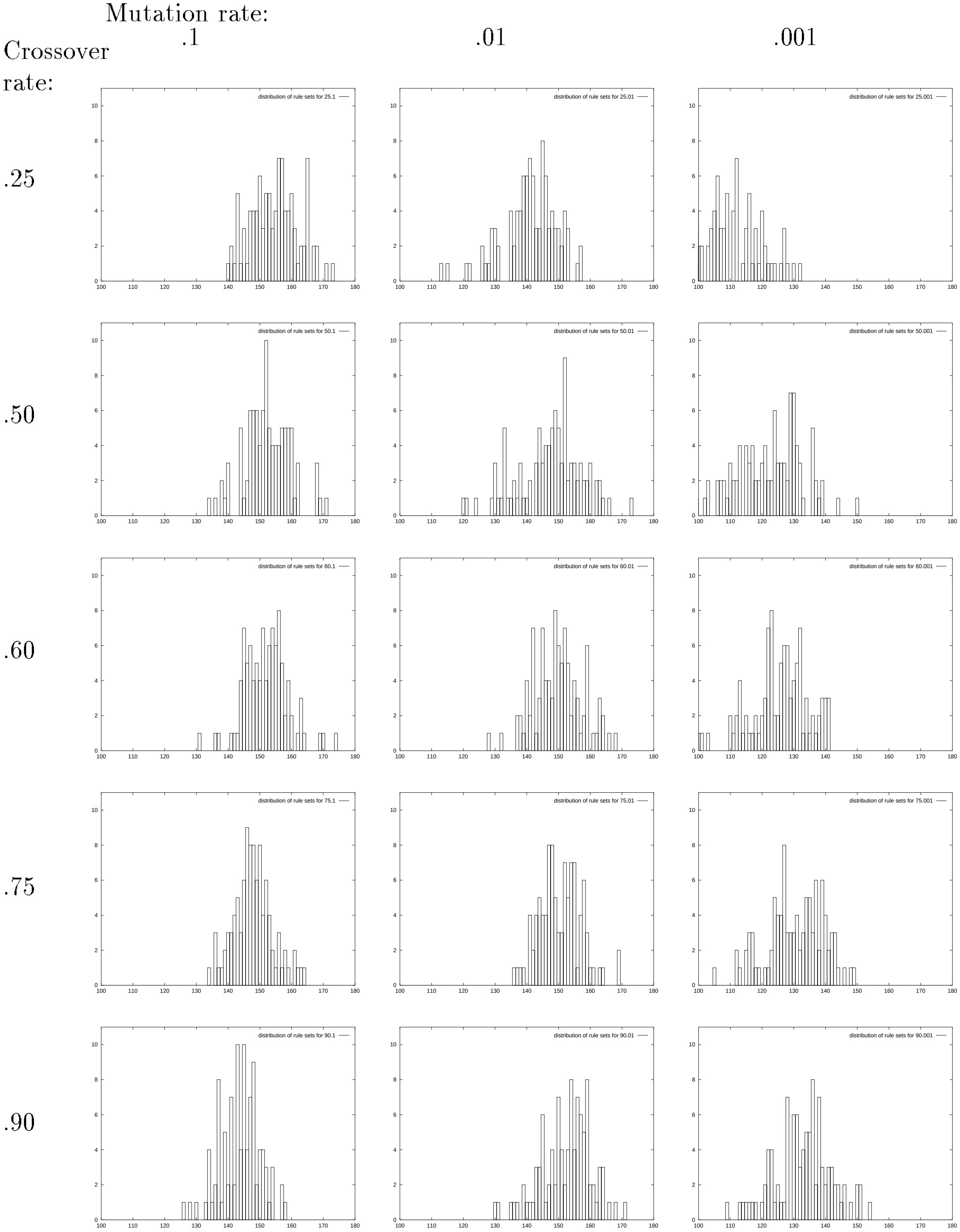


Figure 5.4: Distributions of the best rule sets found in 100 runs, for each of 15 experiments with different parameter settings (varying mutation and crossover rates). The horizontal axis of each bar graph shows the number of positive examples explained (above 100); the vertical axis shows the number of runs that explained a given number of positive examples.

The highest number of positive examples, and the average number of positive examples explained by a rule set in 100 runs with each of the 15 parameter settings is shown in the following tables:

			mutation rate		
			.1	.01	.001
Best	crossover	.25	$\boxed{173}$	157	132
Rule	rate	.50	171	$\boxed{173}$	150
Set		.60	$\boxed{174}$	168	141
		.75	164	169	149
		.90	158	171	154

			mutation rate		
			.1	.01	.001
Average	crossover	.25	$\boxed{154.67}$	141.08	104.88
Rule	rate	.50	152.38	147.09	122.32
Set		.60	151.94	149.64	125.41
		.75	147.93	150.49	130.54
		.90	143.86	152.39	132.95

While there is a wide variation in the "best rule set" seen in these experiments, a few patterns emerge:

- The distribution of runs with the lowest mutation rate (.001) has a lower average fitness than the runs with the highest and middle mutation rates.
- The best rule set explained 174 positive examples (and no negative examples), with parameter settings of .60 crossover and .1 mutation.
- The next best rule sets explained 173 positive examples (and no negative examples), with parameter settings of .25 crossover and .1 mutation, and with parameter settings of .50 crossover and .01 mutation.
- There is no clear "best" crossover rate and no clear "best" mutation rate.

The results from a two-way analysis of variance indicate that the crossover rate, the mutation rate, and the interaction between crossover and mutation rates each have a statistically significant ($p < .0001$) effect on the average rule set found by the GA. These results favor choosing the parameter setting that has the best average performance to be used in future experiments (in this case, .25 crossover and .1 mutation).

The lowest mutation rate (.001) was rejected at this point, and not carried to future experiments, but other mutation rates and all crossover rates were investigated in the next set of experiments. The reason for carrying along all these different parameter settings to future experiments is these parameters might interact with others that are being investigated. (A brief experiment was done to investigate using even lower crossover rates and even higher mutation rates; this is reported in Appendix B.)

5.4.2 The effects of adding immigration

The next set of experiments addressed the issue of adding immigration to our GA. We thought this would help improve the rule sets found by the GA by possibly introducing rules to cover new examples.

The first variation of immigration we tried was that reported by Bean [5]. In this variation, there is no mutation, and immigration of random strings serves to introduce variation into the population. Furthermore, parent selection is random; elitism is required to preserve the best strings (though elitism is already used in our GA). We tried immigration rates of 1% and 5% (one immigrant per generation and five immigrants per generation, with a population size of 100), each with the five different crossover rates from the previous experiments.

For each of these 10 experiments, the 100 resulting rule sets were evaluated. Each rule set contains roughly 25 rules (the number of rules saved via elitism and the generation gap), and explains anywhere from 91 to 166 of the 254 positive examples. The distributions of these runs are illustrated in Figure 5.5.

The highest number of positive examples, and the average number of positive examples, explained by a rule set in 100 runs with each of the 10 parameter settings is shown in the following tables:

			immigration rate	
			.01	.05
Best	crossover	.25	133	143
Rule	rate	.50	144	150
Set		.60	158	159
		.75	156	$\boxed{166}$
		.90	161	162

			immigration rate	
			.01	.05
Average	crossover	.25	107.38	116.80
Rule	rate	.50	124.44	134.75
Set		.60	127.74	137.13
		.75	134.45	144.66
		.90	138.48	$\boxed{147.56}$

The results from a two-way analysis of variance indicate that the crossover rate, and the mutation rate each have a statistically significant ($p < .0001$) effect on the average rule set found by the GA (the combined crossover and mutation interaction has no significant effect).

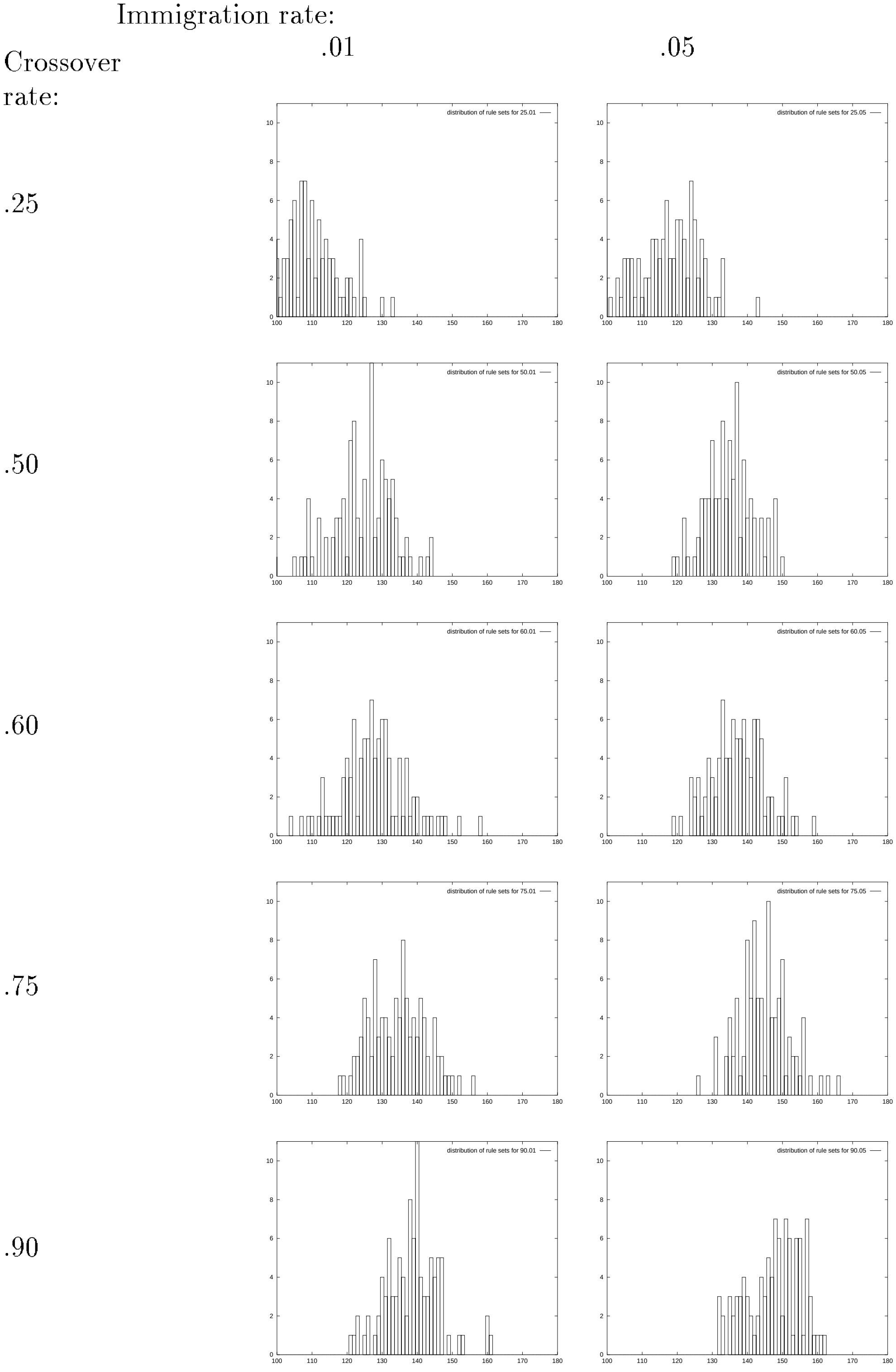


Figure 5.5: Investigation of immigration: Distributions of the best rule sets found in 100 runs, for each of 10 runs with different parameter settings. There is no mutation used with this version of immigration; the horizontal axis represents different immigration rates.

The rule sets found using immigration are on the average worse than those found with the standard GA, and the best rule set found in all runs explained only 166 positive examples. So we tried two other variations on immigration. In both variations, we returned to preferential parent selection, and added mutation. In the first variation, we used immigration of random individuals (as in the initial immigration experiments), and in the second, we used immigration of individuals found by a hillclimbing procedure. For both variations, we used immigration rates of 5%.

For each immigrant, the hillclimbing procedure starts with a randomly generated string, and then mutates this string to find all "one-step" neighbors (those strings that differ in only one attribute-value, for all possible values of all possible attributes). The best of these is chosen as the current string, and again the one-step neighbors are compared to the current string (if there are multiple neighbors that have the same "best" fitness, one is randomly chosen). The process stops when all of the neighbors are worse than the current string; the current string is then used for immigration.

For each of these 20 experiments, the 100 resulting rule sets were evaluated. Each rule set contains roughly 25 rules (the number of rules saved via elitism and the generation gap), and explains anywhere from 119 to 174 of the 254 positive examples. The distributions of these runs are illustrated in Figure 5.6 and Figure 5.7.

The highest number of positive examples, and the average number of positive examples, explained by a rule set in 100 runs with each of the 20 parameter settings is shown in the following tables:

			mutation rate			
			random immigration		hillclimbing immigration	
			.01	.01	.1	.01
Best	crossover	.25	$\boxed{171}$	159	$\boxed{170}$	160
Rule	rate	.50	169	163	$\boxed{171}$	168
Set		.60	165	$\boxed{174}$	164	169
		.75	162	165	158	$\boxed{171}$
		.90	157	166	163	168

			mutation rate			
			random immigration		hillclimbing immigration	
			.1	.01	.1	.01
Average	crossover	.25	152.42	138.72	$\boxed{153.53}$	138.85
Rule	rate	.50	149.98	145.44	150.02	147.04
Set		.60	148.99	149.57	149.26	149.60
		.75	145.00	151.66	143.54	151.56
		.90	140.15	$\boxed{153.28}$	140.03	$\boxed{153.60}$

The results from a three-way analysis of variance indicate that the crossover rate and the interaction between crossover and mutation rates each have a statistically significant

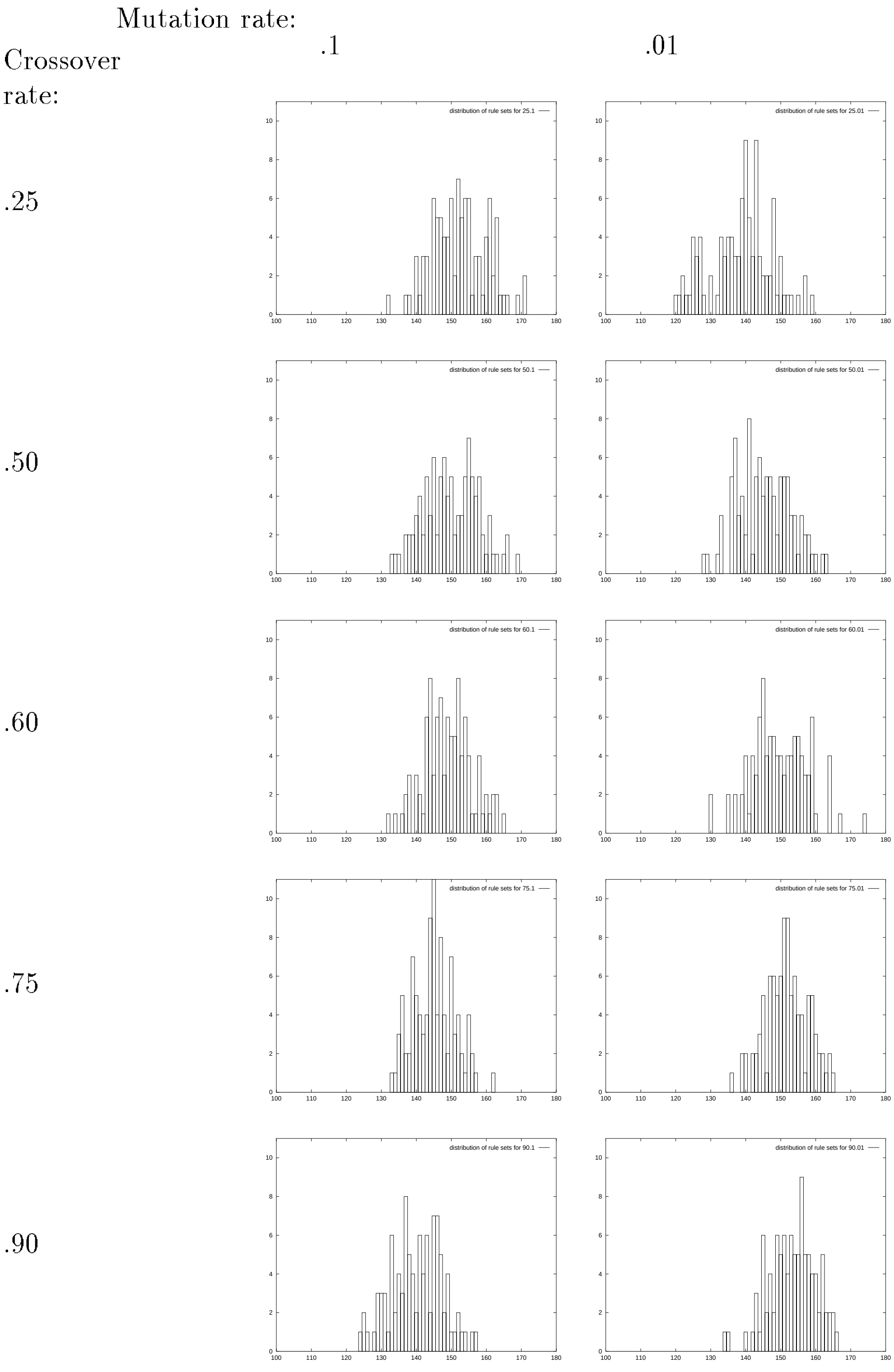


Figure 5.6: Investigation of the effects of 5% random immigration, with mutation and preferential parent selection: Distributions of the best rule sets found in 100 runs, for each of 10 runs with different parameter settings (varying mutation and crossover rates).

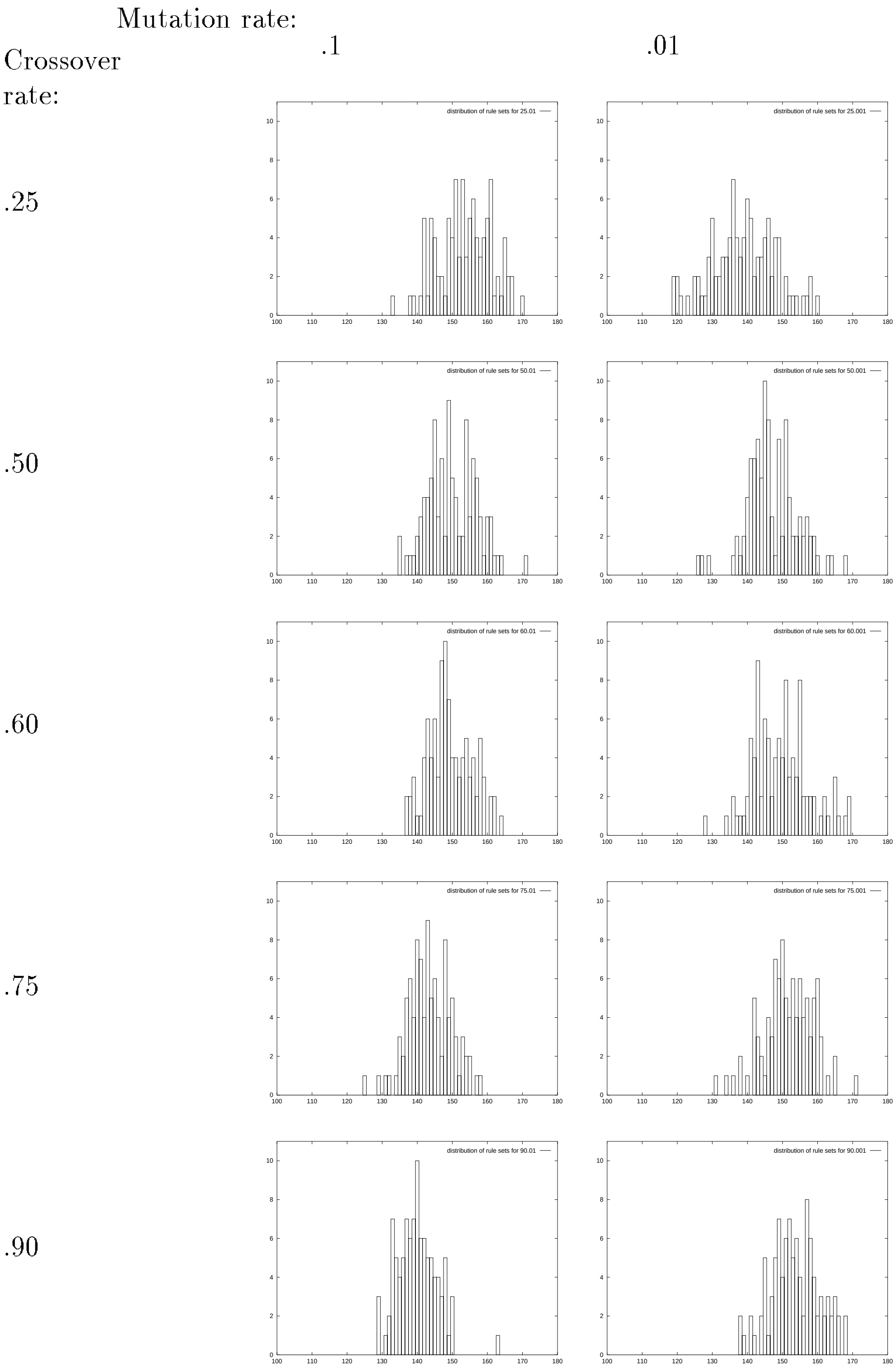


Figure 5.7: Investigation of the effects of 5% hillclimbing immigration: Distributions of the best rule sets found in 100 runs, for each of 10 runs with different parameter settings (varying mutation and crossover rates).

($p < .0001$) effect on the average rule set found by the GA. (The other four terms in the analysis of variance are the mutation rate, the type of immigration (random or hillclimbing), the interaction between crossover and type of immigration, and the interaction between crossover, mutation, and type of immigration. None of these four terms had a significant effect on the average rule set found by the GA.)

The best rule set found in these runs explained 174 positive examples (with random immigration and .60 crossover and .01 mutation), but that run still had a lower average fitness than the best runs without immigration. In addition, the runs with immigration took much longer than without immigration (nearly twice as long with hillclimbing immigration, which is a tremendous increase in time when 100 runs takes 2 days to complete).

Immigration at this point must be considered a promising idea that proved not to be helpful for our task. One of the main advantages of immigration is thought to be its ability to prevent premature convergence by maintaining diversity in the population. It is quite possible that our population-based fitness approach already maintains considerable diversity. Our GA may also be good at maintaining diversity due to the relatively high mutation rates used and the use of ranking.

5.4.3 The effects of varying w

In this set of experiments, all parameters were held constant except for the value of w in the fitness function from Section 4.3.3:

$$evaluation(a, b, r, w) = w * correctness(a, b) + (1 - w) * simplicity(r)$$

where $correctness(a, b)$ is the *log(odds ratio)* function, and $simplicity(r)$ is $\frac{(254-r)}{253}$ (a linear measure of the number of rules). Previous experiments have used the value $w = 1.0$, following the usual machine learning approach to find the best rule set, independent of the number of rules.

In this set of experiments, parameter settings are fixed with a crossover rate of .25 and a mutation rate of .1; no immigration is used. Five different values of w were explored: $w = 1.0$, $w = 0.75$, $w = 0.50$, $w = 0.25$, and $w = 0.0$. (Note that $w = 1.0$ corresponds to an experiment that has already been done.)

Parameters used in previous experiments produced rule sets with about 25 experiments, which is not enough to explain the entire rule set. So additional experiments were done at this point, increasing the population size to 150 and decreasing the generation gap to 50%. This means that with the previous settings, 25 strings were saved each generation and 75 new ones were created, and with new settings, 75 strings are saved each generation and 75 new ones are created. The number of trials was also increased to 150,000, to keep the number of generations roughly constant. (Runs using the larger population sizes with $w = .25$ and $w = 0.0$ were not done because these runs clearly would not benefit from the additional rules.)

Looking at the distributions of the rule sets found in 100 runs is less informative for this set

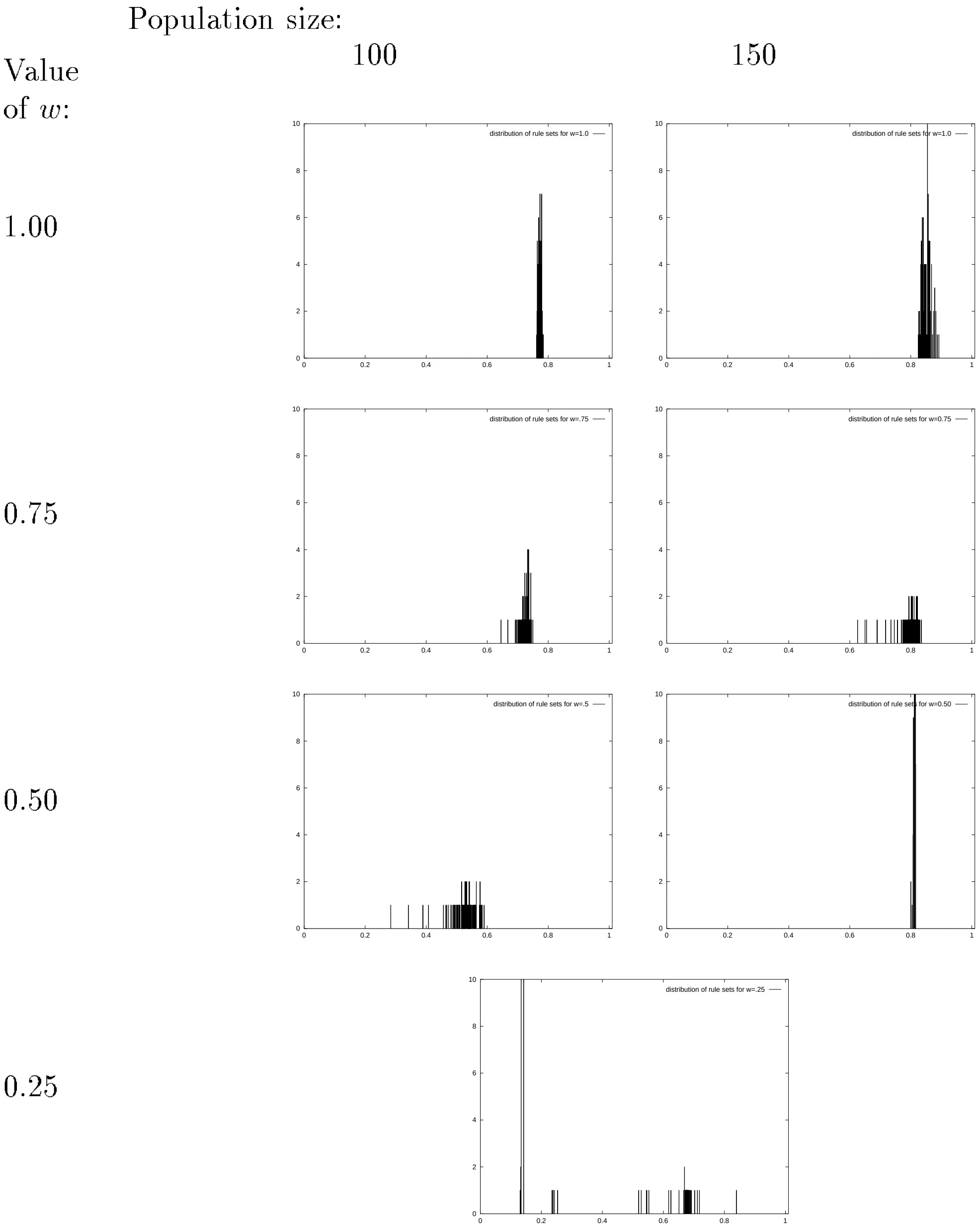


Figure 5.8: Distributions of the best rule sets found in 100 runs, for each of 8 runs, using different values of w in the fitness function. The horizontal axis is the fitness on the range [0..1]; the vertical axis is the number of rule sets on the range [0..10]. These graphs look much different from previous graphs because the rule sets now contain false positives. Therefore, instead of graphing the number of positive examples explained, we graph the *fitness* of each rule set. (The graph for $w = 0.0$ is not shown because at that weight there is no selection pressure in the GA for *correctness*, and all runs converge to a single string that says all examples are positive.)

of experiments than for previous experiments, but these graphs are provided in Figure 5.8 for completeness. The graphs are also more difficult to interpret because we can no longer use the number of positive examples as the horizontal axis: the number of negative examples must also be considered. (In previous graphs, the number of negative examples was always 0.) The horizontal axis for these graphs is now the fitness, on the scale [0..1.0] (where lower fitness is a better rule set). However, note that the fitness cannot meaningfully be compared across different values of w because of the differing contributions of *correctness* and *simplicity*.

The average fitnesses of each of the runs is not very meaningful, again, because we cannot compare different values of w, but it may still be helpful to look at the best rule set found in each of the 100 runs:

	w	fitness	correctness	simplicity	a	b	rules
Population	1.00	.7842	.7842	.5771	173	0	27
Size	0.75	.7498	.7916	.6245	183	0	25
100	0.50	.6872	.5365	.8379	221	231	9
	0.25	.8982	.6282	.9882	252	304	3
	0.00	1.000	.4910	1.000	254	319	1
Population	1.00	.8922	.8922	.7391	247	0	60
Size	0.75	.8058	.8303	.7323	222	0	36
150	0.50	.7815	.5985	.9646	253	298	8

From this table, note that the *correctness* of the best rule set tends to decrease with the value of w and the *simplicity* of the best rule set tends to increase as the value of w decreases. Also notice that with a population size of 100, $w = .75$ actually did “better” than $w = 1.0$ in that it explained more positive examples with fewer rules, although this anomaly does not exist with a population size of 150.

A useful way of illustrating the differences between the best rule sets obtained using different values of w is with the use of a ROC curve, which plots the *sensitivity* of the rule set against the *specificity* as each new rule is added (ROC is an acronym for “receiver optimizing characteristic”; see Metz [68] for a discussion of the use of this tool in epidemiology). Recall from Section 4.2.1 that sensitivity is the fraction of positive examples that have been explained — a measure of completeness — and specificity is the fraction of examples explained that are positive — a measure of correctness.

In a ROC curve, each successive rule added to the rule set is represented by a tick mark; each rule tends to increase the *sensitivity* of the rule set and decrease the *specificity.* The ROC curves for the five different values of w are superimposed on the same graph in Figure 5.9. Graphs are shown for population sizes of 150 for $w = 1.0$ and $w = .75$ and for population sizes of 100 for other values of w.

From Figure 5.9, we see that the two highest values of w do not stray from the vertical line defined by *specificity* = 1.0, and are therefore, difficult to distinguish. Therefore, Figure 5.10,

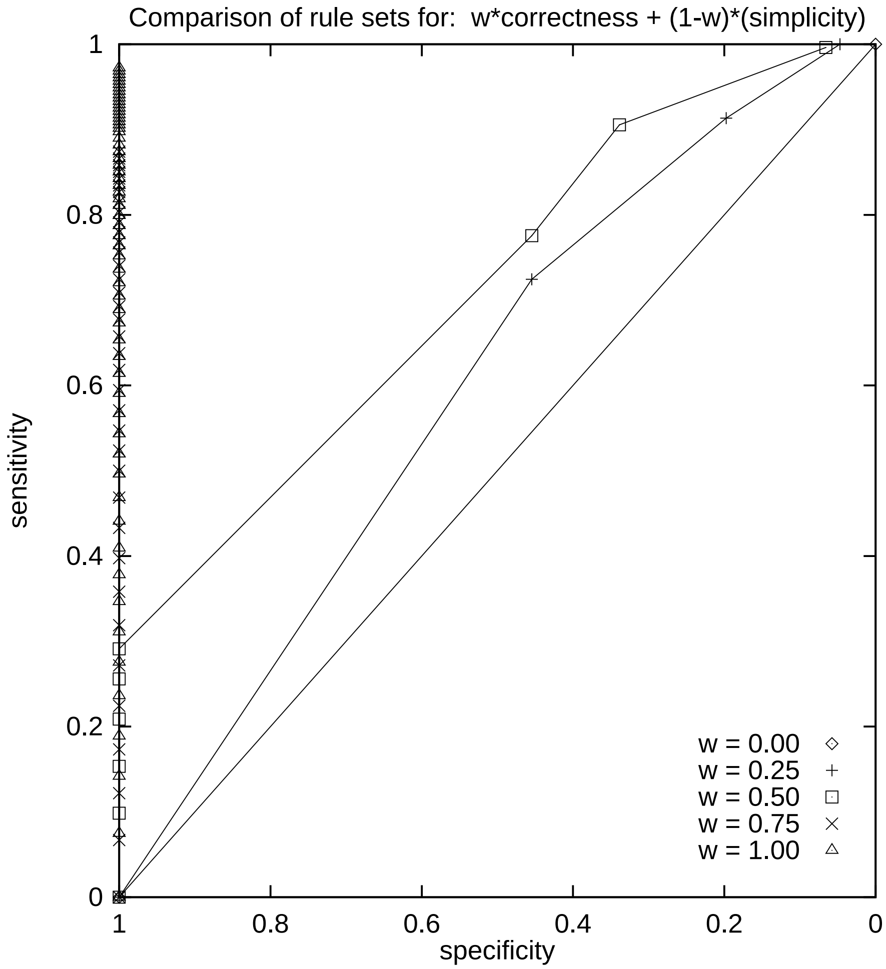


Figure 5.9: Graph of *sensitivity* vs. *specificity* in describing the entire dataset for each of the values of w.

shows a variation of the ROC graph, which plots *sensitivity* on the vertical axis, and the number of rules on the horizontal axis. By looking at the two graphs together, we are able to get a better feeling for the differences in the rule sets obtained using different values of w.

5.4.4 An evaluation of prediction accuracy

To evaluate the prediction accuracy of the GA, we used the "leave out n" strategy: 20% of the data were randomly selected (without replacement) from the original dataset, a rule set was formed using the remaining 80% (the "training" data), and the rule set was evaluated using the 20% (the "testing" data). This was repeated five times, each with a different random split of the dataset into training and testing sets. The prediction accuracy for each

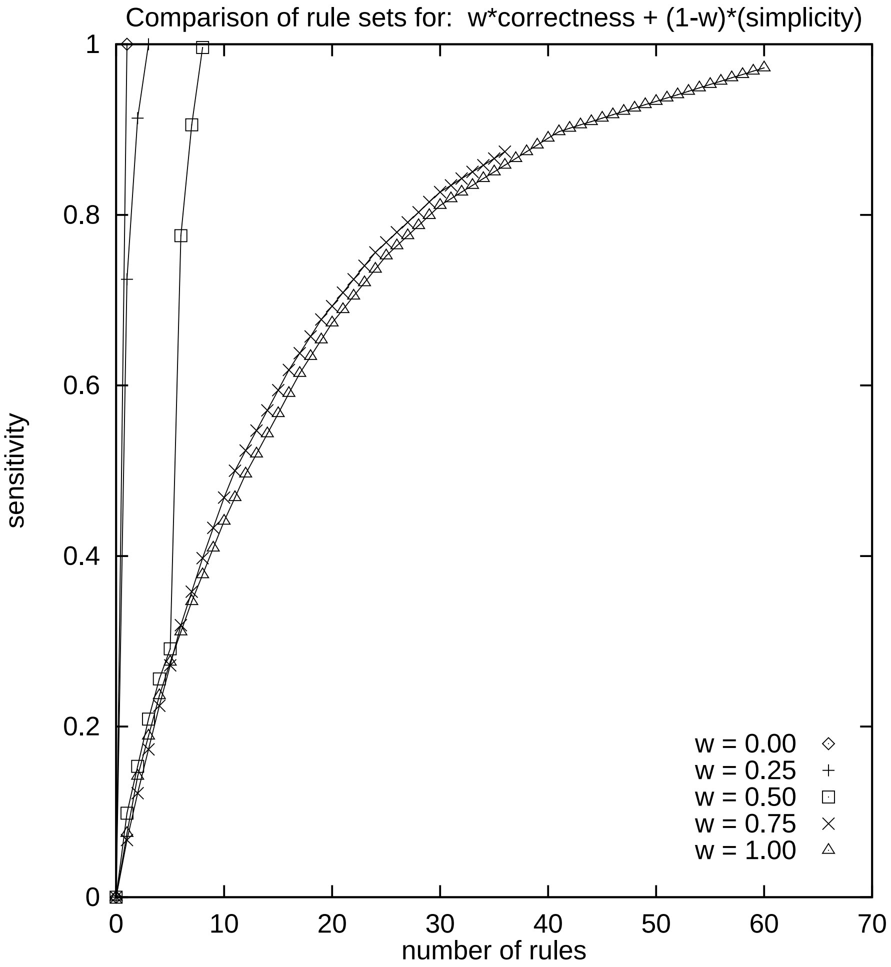


Figure 5.10: Graph of *sensitivity* vs. number of rules in describing the entire dataset for each of the values of w.

run is the percent of examples in the testing set that are correctly classified (as positive or negative examples).

The structure of the rule sets formed by the GA is that only positive examples are described by the rule set; anything not described by the rule set is assumed to be a negative example. Therefore, the "number of rules" for each of these rule sets might be incremented by one to include a default rule for "all other examples are negative".

		training set			testing set		
	number of	number of	number	percent	number of	number	percent
run	rules	examples	correct	correct	examples	correct	correct
1	53	458	458	100%	115	57	50%
2	55	458	458	100%	115	61	53%
3	55	458	458	100%	115	54	47%
4	50	458	458	100%	115	63	55%
5	57	458	458	100%	115	65	57%
(totals)	270	2290	2290	100%	575	300	52.2%

In the table above, prediction accuracy is the same as the percent correct on the training set. The GA achieves only 52% prediction accuracy in these experiments, which is poor performance indeed.

The runs with varied values of w fared even worse in the test of prediction accuracy, as shown in the tables below:

$w = .75$		training set			testing set		
	number of	number of	number	percent	number of	number	percent
run	rules	examples	correct	correct	examples	correct	correct
1	52	458	453	99%	115	54	47%
2	49	458	452	99%	115	54	47%
3	50	458	454	99%	115	52	45%
4	43	458	453	99%	115	54	47%
5	48	458	454	99%	115	57	50%
(totals)	242	2290	2266	99.0%	575	271	47.1%

$w = .50$		training set			testing set		
	number of	number of	number	percent	number of	number	percent
run	rules	examples	correct	correct	examples	correct	correct
1	16	458	261	57%	115	43	37%
2	8	458	214	47%	115	50	43%
3	9	458	221	48%	115	48	42%
4	14	458	227	50%	115	54	47%
5	9	458	224	49%	115	52	45%
(totals)	56	2290	1147	50.1%	575	247	43.0%

$w = .25$		training set			testing set		
	number of	number of	number	percent	number of	number	percent
run	rules	examples	correct	correct	examples	correct	correct
1	2	458	217	47%	115	43	37%
2	3	458	219	48%	115	48	42%
3	2	458	213	46%	115	46	40%
4	3	458	211	46%	115	54	47%
5	3	458	213	46%	115	50	43%
(totals)	13	2290	1073	46.9%	575	241	41.9%

In these runs, the average percent correct on training runs was 99%, 50%, and 47%, for weights of .75, .50, and .25, respectively; the average prediction accuracy was 47%, 43%, and 42%. These percentages can be compared against a single rule that said "all examples are negative", which would get an average 55.2% correct on training runs and 57.7% correct on testing runs. This single rule is slightly better than some of our GA rule sets in describing the training data, but is much worse than the GA runs with $w = .75$ and $w = 1.0$. This rule is better than *any* of the GA runs in describing the testing data. However, since our emphasis is on describing positive examples, a fairer comparison might be one rule that said "all examples are positive"; this would yield an average 44.8% correct on training sets and 42.3% correct on testing sets.

The results of these prediction accuracy experiments will be discussed further in Chapter 8, which compares the relative performace of different approaches.

CHAPTER 6

THE DECISION TREES IMPLEMENTATION

Decision trees are perhaps the most frequently used approach in machine learning classification research. This approach is attractive because it is conceptually and mechanically simple as well as very fast, especially as compared with many other approaches. It is also generally very successful at finding good rule sets for many machine learning tasks.

Decision trees are, therefore, an important approach to use as a comparison to our GA system (described in Chapter 5), which is not an approach commonly used in classification work. Both approaches produce deterministic rules, which can be readily understood by humans (as opposed to probabilistic structures, which cannot). Both systems are capable of discovering interactions between attributes that are important for the classification task. However, since the typical decision tree system investigates one attribute at a time, such discoveries are perhaps more serendipidous than in the GA, which is continually (but not exhaustively) searching the space of attribute-value combinations. For this reason, we expected our GA to find better rule sets than decision tree systems for our task, which has an expected high degree of interaction between attributes and the class.

Decision trees were introduced in Section 3.3.2. In the first section of this chapter, I present some of the possible variations of decision tree mechanisms. In Section 6.2, I discuss the particulars of the variations pursued in this research. Finally, in Section 6.3, I present the results of these runs.

6.1 Variations on Decision Trees

Virtually every GA implementation is different because there are so many parameters that may be varied (and so many different mechanisms), but the mechanism of decision trees is quite simple and there are fewer variations. Furthermore, there are a number of variations that have been established, and are relatively well-known within the machine learning community.

There are three primary elements that may vary from one decision tree system to another: the evaluation metric used, the splitting criterion used, and the representation used (whether the system works with discrete or continuous traits, or both). In addition, some decision tree systems use pruning or other methods to increase the statistical confidence in the resulting rule set or to increase the prediction accuracy on testing sets.

Variations in the evaluation metric:

The information gain function (presented in Section 3.3.2) is perhaps the most well-known metric for evaluating and comparing the classifications produced by splitting on particular attributes, but many other possibilities exist for this heuristic. Research on decision trees often focuses on how variations of this metric affect the resulting trees (and their associated predictive accuracy).

Examples of other metrics include:

Gain Ratio — One problem with the information gain function is that it is biased in favor of attributes with a larger number of values, as reported by Quinlan [83]. Quinlan's solution to this was to alter the information gain function by dividing by a term that factored in the number of examples for each attribute-value in the current node.

Orthogonality measure — In [32], Fayyad develops a metric called the orthogonality measure, applicable to binary trees, which is specifically designed to favor partitions that keep examples of the same (or similar) class together.

Several studies have been done regarding the effects of using different metrics. Mingers [71] finds that the accuracy of the final tree does not change significantly with the choice of metrics; the metric used seems to affect the size of the final tree, but not its accuracy.

Variations in the splitting criterion:

The traditional decision tree formulation splits on one attribute at a time and creates branches for all possible values of that attribute, but variations on this are certainly possible.

Default branches — Fayyad [32, 34] shows that there are a number of disadvantages to creating branches for all values of an attribute (such as the "irrelevant values problem", the "reduced training data problem", and the "missing branches" problem). One possibility is to branch on specific values of the attribute, and leave other values as a single "default" branch.

Binary trees — It is also possible to always create exactly two branches at a split. Fayyad [32] shows that splitting a single attribute-value from all other attribute values produces a binary tree that is logically equivalent to the default branches approach described above (the same number of leaf nodes, etc.), and that the binary approach is preferable for a number of reasons.

Combined attributes — It is possible to create decision trees that split on more than one attribute at a time. This creates a computational problem in that exploring all possible attribute combinations would be extremely time-consuming; an alternative, exploring only some combinations (chosen either heuristically or stochastically), is possible, and was explored by Pagallo and Haussler in their Fringe system [78].

Variations in the representation used:

The traditional decision tree formulation required discrete attributes, but many decision tree systems now allow for continuous-valued attributes. At issue is whether the attribute is split into subranges before the system executes, or is split as part of the tree-formation process. The advantage to determining these subranges at run time is that the splitting process can take into account the subset of examples at the particular node being considered. That is, the entire range of values will not be represented for all attributes at all nodes. The disadvantage is that the discretization process may increase the run time.

This distinction is not pursued further in this thesis, since we have restricted this research to a dataset that has already been discretized.

Variations to increase statistical validity

Another dimension that often varies in decision trees has to do with the statistical confidence of the resulting tree. The typical decision tree system recursively partitions the tree until all examples are in the same class or until there are no additional attribute-values to branch on. However, on some datasets, this may result in leaf nodes that explain only one or two examples. The rules corresponding to these leaf nodes have low descriptive ability as well as low predictive ability. Some decision tree systems have introduced mechanisms to counter this:

Pruning — After the decision tree has been formed, some of the interior nodes (not leaf nodes) may be made into leaf nodes by removing all subnodes. Alternatively, individual leaf nodes may be removed, such as those that explain no examples.

Stopping criteria — During the tree-generation process, the recursion may terminate before all examples are the same class and before all possible attribute-values have been exhausted.

6.2 Specific Decision Tree Systems Used in this Thesis

Three variations on decision trees were investigated in this research: ID3, GID3*, and O-Btree. The first was chosen because it is somewhat of a standard in the machine learning community; the latter two were chosen because they were developed at the University of Michigan and because they compare favorably to other decision tree systems in terms of their predictive accuracy on a range of tasks [32, 33, 18].

ID3 and GID3* both use information gain as the evaluation measure; O-Btree uses the orthogonality measure. ID3 creates standard decision trees that branch on every value of every attribute; GID3* allows for "default" branches, and O-Btree creates binary trees. Due to the "default" branches allowed by both GID3* and O-Btree, both of these systems produces rule sets with the same expressive power as the GA string representation. That is, it is possible to say (in effect) "this value or this value, but not that value" for a particular attribute. This does not hold for ID3.

Decision tree software used

The software used for all three decision tree systems was TreeMaker from the Jet Propulsion Laboratory at the California Institute of Technology (JPL), originally developed at the University of Michigan. The advantages to this software are that it can run any of these three systems by setting a parameter, and that it is very fast to run. The disadvantage of this software is that the source was not available from JPL at the time of this thesis work. This prevented some experiments, such as trying alternative evaluation metrics, from being done.

6.3 Results from Decision Trees

Since there is no stochastic component to any of the decision tree systems investigated in this thesis, unlike our GA experiments, there is no need to do multiple runs with different random seeds and report the distributions observed. Instead, for each system, we can present the performance on the entire dataset as well as on the prediction task, with a few tables.

The first table summarizes the performance of these three systems on the description task. The GA results are also listed as a comparison. In these tables, a and d are the number of true positive and true negatives, respectively, from Section 4.2.1, and fitness is the scaled $log(oddsratio)$ function:

system	number of rules	a	d	fitness	accuracy	average ex./rule	average pos. ex./rule
ID3	266	254	319	1.0	100%	2.15	1.95
GID3*	183	254	319	1.0	100%	3.13	2.73
O-Btree	207	254	319	1.0	100%	2.77	2.51
GA	60	247	319	0.89	98.8%	9.55	4.12

The next table summarizes the performance of these three systems on the prediction accuracy task:

	averages across five testing/training runs					
system	number of rules	a	d	number correct	prediction fitness	prediction accuracy
ID3	208	21	35	56	0.49	48.3%
GID3*	159	20	39	59	0.52	51.7%
O-Btree	157	22	36	58	0.50	50.6%
GA	54	25	35	60	0.51	52.2%

The following three tables present the results from five testing and training runs for each of the three systems.

ID3		training set			testing set		
	number of	number of	number	percent	number of	number	percent
run	rules	examples	correct	correct	examples	correct	correct
1	209	458	458	100%	115	56	49%
2	201	458	458	100%	115	60	52%
3	210	458	458	100%	115	51	44%
4	208	458	458	100%	115	59	51%
5	212	458	458	100%	115	52	45%
(totals)	1040	2290	2290	100%	575	278	48.3%

GID3*		training set			testing set		
	number of	number of	number	percent	number of	number	percent
run	rules	examples	correct	correct	examples	correct	correct
1	150	458	458	100%	115	59	51%
2	158	458	458	100%	115	65	57%
3	172	458	458	100%	115	57	50%
4	159	458	458	100%	115	57	50%
5	157	458	458	100%	115	59	51%
(totals)	796	2290	2290	100%	575	297	51.7%

O-Btree		training set			testing set		
	number of	number of	number	percent	number of	number	percent
run	rules	examples	correct	correct	examples	correct	correct
1	148	458	458	100%	115	56	49%
2	162	458	458	100%	115	50	43%
3	152	458	458	100%	115	64	56%
4	158	458	458	100%	115	64	56%
5	163	458	458	100%	115	57	50%
(totals)	783	2290	2290	100%	575	291	50.6%

ID3 showed the worst performance of the decision tree systems, requiring more rules to describe the data and achieving a lower prediction accuracy. GID3* is better than O-Btree in describing the original dataset (in that it requires fewer rules and therefore achieves a higher example/rule ratio), but is marginally worse than O-Btree in describing the training data. GID3* achieves a higher average prediction accuracy than O-Btree, however, a paired t-test of GID3* as compared to O-Btree shows that there is no significant difference between these systems on the five runs of the prediction task. The difference between the prediction results of GID3* and ID3 are statistically significant ($p < .05$). (The difference between the prediction results of ID3 and O-Btree is not significant.)

When one compares the GA's performance to GID3* and O-Btree, a similar pattern is observed: although the GA achieves the highest prediction accuracy across all tasks, a paired t-test comparing the five prediction runs from the GA and from GID3* shows no significant difference between these systems.

The discussion of the relative strengths and weaknesses of these systems will be discussed again in Chapter 8.

CHAPTER 7

A SUMMARY OF OTHER APPROACHES IMPLEMENTED

Four additional approaches were investigated in this thesis research, and are presented together in this chapter because from the onset they were recognized as not quite appropriate for our task. Random string generation and hillclimbing were implemented as a comparison specifically against the GA, as a test of the GA's search abilities on this task. Autoclass and Cobweb were implemented simply because they are well known machine learning systems; though it was recognized from the start that these two would not do well on the description component of the task, they were investigated as possibly good alternatives for the prediction component of the task. However, since both of these systems are designed to do unsupervised learning (that is, they don't distinguish the class attribute from the other attributes), it was not clear that they would do well on the prediction task either.

The approaches investigated in this chapter were first presented in Chapter 3. Unlike GA's or decision trees, the systems presented in this chapter are not privy to a large number of variations; this chapter simply presents the results of running these systems.

The software for Cobweb and Autoclass was provided by NASA Ames' software distribution center, COSMIC.

7.1 Random Generation and Hillclimbing

As mentioned in Chapter 3, the random generation and hillclimbing approaches were implemented as a comparison against the GA. The main body of work done with the GA involved evolving a population of strings (the rule set), rather than a single rule; however, the random generation and hillclimbing approaches find a single rule at a time. To best compare these, then, the GA was run without the modification for evolving rule sets. That is, the GA was run in the more traditional mode of optimizing to find the single best string (in this case, the single best rule), and this was compared against the random generation and hillclimbing approaches, which also look for the single best string.

In these runs, 5,000,000 strings were generated. For the GA, this was achieved using 100 runs, each with 50,000 trials. With random generation, this was done by generating 5,000,000 random strings. The appropriate number of hillclimbing runs cannot be predetermined since any given hillclimbing run will evaluate at least 287 strings (the initial string plus its 286 one-step neighbors), but possibly many more. Therefore, the number of hillclimbing runs is determined at the time of execution, so that at least 5,000,000 strings will be evaluated in all (in practice, this turned out to be 16690 runs).

Note that each of these three approaches is a variation of the "generate and test" approach; the variation occurs in the method used for generation of candidate solutions. With the random generation approach, the candidates are generated at random; with the GA, the candidates are generated via the crossover and mutation operators; and with hillclimbing the candidates are generated by creating all one-step neighbors of the current string.

Also note that although each of these approaches as compared here is searching for a best string, rather than the best rule set, they could easily be extended to finding complete rule sets. The method for this is called "Iterative Runs", and is described in Appendix A.

The table below uses *a* and *b*, the number of true positives and the number of true negatives explained, rather than the *a* and *d* of previous tables. This is because we are looking at the equivalent of a single rule, rather than an entire rule set. All of these rules are good rules in that they explain only positive examples.

	Best rule				Number of times
Approach	*a*	*b*	fitness	run time	best is found
GA	24	0	.666	8 hrs	3
Random	6	0	.611	4 hrs	1
Hill	20	0	.658	4.5 hrs	1
Hill-P5	22	0	.662	4.5 hrs	1
Hill-P10	16	0	.649	4.5 hrs	1
Hill-P100	20	0	.658	5 hrs	2

The last three approaches, Hill-P5, Hill-P10, and Hill-P100, were an experiment to try to improve the performance of hillclimbing. It is not uncommon for the hillclimber to reach a "plateau" on this problem — where several neighbors have the same fitness as the current solution, but none are better. One of the ways that I varied the algorithm was to add a parameter, P; when a plateau is reached, it is allowed P steps on the plateau (random steps) to look for higher peaks. In addition to the default of P=1, I tried P=5, P=10, and P=100.

While hillclimbing did better than I'd expected in this comparison against the GA, the GA still outperforms the other approaches, explaining more of the positive examples and finding these high-fitness rules more frequently. In its 100 runs, the GA also found an a=23, b=0 rule 4 times, and an a=22, b=0 rule 10 times. The next best rule found by the Hill-P5 approach is a=17, b=0, which was found only one time in 3352 runs. The next best rule found by the Hill-P100 approach is a=19, b=0, which was found only one time in 127 runs.

7.2 Autoclass

Autoclass, first introduced in Section 3.3.3, is an example of an unsupervised learning system, and does not distinguish the class attribute when forming its classes. Like Cobweb, Autoclass forms a probabilistic description of the data; however, in Autoclass, this description *does* correspond to rules. Autoclass therefore, falls into a grey area on the description task because these rules are not easy for humans to internalize, but they may still be useful in terms of shedding insight on the underlying structure of the data. Also, Autoclass is a well-known machine learning system, and it is interesting to evaluate its predictive abilities on this task.

The evaluation of Autoclass cannot proceed without the knowledge that using the system on this task violates an important assumption of Autoclass, namely, the assumption of independence of the attributes. (However, according to the Autoclass documentation, this assumption "is often violated in practice", so we need not feel we are conducting a useless exercise.) Autoclass was run with the single-multinomial model (which models discrete attributes as multinomials; this is the only model available in Autoclass for using discrete attributes), and told to look for from 30-60 classes in the data (different tries looked for a different number of classes). Without this restriction, Autoclass consistently found two or three classes in the data, since it assumes that fewer classes is preferable and since it ignores the class attribute. The range of classes to look for (30-60) was selected after the GA runs, using the 60 rules found by the GA as a guide. Since Autoclass would run forever if allowed to, each Autoclass run was terminated after 24 hours. The results from Autoclass on the description and prediction tasks are given in the tables below:

system	number of rules	*a*	*d*	fitness	accuracy	average ex./rule	average pos. ex./rule
A-CLASS	30	160	220	0.55	66.3%	19.1	18.14

	averages across five testing/training runs					
system	number of rules	*a*	*d*	number correct	prediction fitness	prediction accuracy
A-CLASS	30	18	44	62	0.51	53.6%

The results from the five training and testing runs are given in the table below:

A-CLASS		training set			testing set		
run	number of rules	number of examples	number correct	percent correct	number of examples	number correct	percent correct
1	29	458	308	67%	115	55	48%
2	29	458	341	74%	115	60	52%
3	30	458	317	69%	115	63	55%
4	32	458	312	68%	115	62	54%
5	29	458	314	69%	115	68	59%
(totals)	149	2290	1592	69.5%	575	308	53.6%

A class description formed by Autoclass is a list of the probabilities for each attribute-value; therefore, the class attribute lists a probability for being a positive example and a probability for being a negative example. Because this attribute is not distinguished, across the 30 classes found by Autoclass for the description task, 9 of the classes are divided nearly 50-50 for the two possible values (that is, one value has a probability of less than .60, while the other has a probability greater than .40). This is the reason that Autoclass is not very accurate on the description task.

The curious thing is that although nine of the 30 classes are divided nearly 50-50 for the class attribute, Autoclass performs better than any other system on the prediction task. However, using a paired t-test, this difference is not significant. Furthermore, although the rule sets found by Autoclass are able to describe the data with fewer rules than any other system, these descriptions are not very accurate.

A brief experiment

Although Autoclass does allow the option of specifying that the dataset has been "preclassified" (i.e., is a supervised learning task), when I tried running this option with this task, Autoclass stuck with the two classes as defined in the dataset and did not subdivide further than that. The result of this run was probabilistic descriptions of the two classes (positive examples and negative examples).

One ad hoc attempt was made to force Autoclass to acknowledge the class attribute: the class attribute was duplicated so that it appeared in the dataset a total of 5 times. (Note that this further violates the assumption of independence of variables.) The results from this experiment are reported in the tables below; these runs are labelled as A-class-5:

system	number of rules	a	d	fitness	accuracy	average ex./rule	pos.	average ex./rule
A-CLASS-5	30	96	305	0.60	70.0%	19.1		42.2

	averages across five testing/training runs					
system	number of rules	a	d	number correct	prediction fitness	prediction accuracy
A-CLASS-5	30	21	40	61	0.51	52.9%

The results from the five training and testing runs are given in the table below:

A-CLASS-5		training set			testing set		
run	number of rules	number of examples	number correct	percent correct	number of examples	number correct	percent correct
1	30	458	458	100%	115	63	55%
2	29	458	458	100%	115	52	45%
3	29	458	458	100%	115	59	51%
4	30	458	458	100%	115	69	60%
5	30	458	458	100%	115	61	53%
(totals)	148	2290	2290	100.0%	575	304	52.9%

The results here are somewhat inconclusive; although the experimental Autoclass-5 runs (with the duplicated class attribute) achieved a higher fitness and accuracy on the description task than the original run, Autoclass-5 fared worse on the prediction task. Using a paired t-test, the five prediction runs are not significantly different than those from Autoclass. Autoclass-5 did turn out to achieve the stated objective of forcing Autoclass to acknowledge the class attribute: all of the 30 rules had probabilities greater than .90 of being one class or the other (only 2 of the rules in Autoclass had this characteristic).

7.3 Cobweb

Cobweb was first introduced in Section 3.3.4. Like Autoclass, Cobweb is an unsupervised learning system (one that does not distinguish the class attribute), and one cannot extract rules from the resulting concept hierarchy in any meaningful way. Therefore, it was known from the onset that this system was not suited to our task. The results of running Cobweb on the description and prediction tasks are given in the tables below. (Because rules cannot be extracted from the concept hierarchy, these entries are left as "?" in the tables.)

system	number of rules	a	d	fitness	accuracy	average ex./rule	average pos. ex./rule
Cobweb	?	231	291	0.68	91.1%	?	?

	averages across five testing/training runs					
system	number of rules	a	d	number correct	prediction fitness	prediction accuracy
Cobweb	?	22	33	55	0.49	48.0%

The results from the five training and testing runs are given in the table below:

Cobweb		training set			testing set		
run	number of rules	number of examples	number correct	percent correct	number of examples	number correct	percent correct
1	?	458	418	91%	115	49	43%
2		458	403	88%	115	50	43%
3		458	410	90%	115	54	47%
4		458	419	91%	115	61	53%
5		458	407	89%	115	62	54%
(totals)	?	2290	2057	89.9%	575	276	48.0%

From the onset, Cobweb failed on the description task; the 48% prediction accuracy shows it to be worse than both the GA and decision trees on the prediction task as well. Although it did only marginally worse than ID3 on the prediction task, Cobweb was not significantly worse than any of the decision tree systems. Cobweb was significantly worse than the GA ($p < .05$) and Autoclass ($p < .01$) on the prediction task

Cobweb was designed to form its concept hierarchy to achieve a balance between the predictiveness of each attribute (its ability to predict the class, where “class” is established by Cobweb and not by the class attribute provided in the dataset) and the predictability of each attribute (its ability to predict the correct value for an attribute of an example that is missing this information). In this formulation, our “class” attribute is not distinguished, and is treated just like all other attributes in the dataset.

Two approaches for modifying Cobweb to be more suited to a supervised learning task were briefly considered. In the first, the class attribute is duplicated, which should increase both its predictiveness and its predictability. In the second, the internals of Cobweb are modified so as to increase the predictability of the class attribute relative to other attributes, and to decrease its predictiveness relative to the other attributes. The first modification was attempted, but resulted in a concept hierarchy that split into two subtrees from the root node — one subtree explained the positive examples and the other explained the negative examples; there is nothing to prevent these trees from being similar or even identical. The second modification was not attempted based on reports from others who had tried it and found it to be largely unsuccessful [79].

CHAPTER 8

THE COMPLEXITY OF THE PROBLEM

Chapters 5, 6, and 7 reported the details and the results of implementing the common-disease-research classification task (presented in Chapter 2) on a variety of machine learning systems. This chapter overviews the performance of these various systems on the task, and discusses the complexity of this task, relative to others that are commonly studied in machine learning.

8.1 A Summary of the Performance of Different Approaches

As discussed in Chapter 3, the task pursued in this thesis differs from most in machine learning in that the primary emphasis is on *describing* the original dataset well, rather than *predicting* the class for a separate training set. However, because prediction accuracy is such a standard metric in the machine learning community, this has been evaluated in the thesis as well.

The description task

The three best systems were the GA, GID3*, and Autoclass. The results of these systems on the description task are summarized in the table below:

	number					average
system	of rules	*a*	*d*	fitness	accuracy	examples/rule
GA	60	247	319	0.89	98.8%	9.55
GID3*	183	254	319	1.0	100%	3.13
A-CLASS	30	160	220	0.55	66.3%	19.1

Recall that a good description is based not only on accurately describing the dataset, but also on describing the dataset *well*. That is, the number of rules required to explain the data (and comparably, the average examples per rule), is of major importance, as is the fact that these rules must be intepretable by humans

Looking at the above table, the reader will notice that GID3* did the best in terms of accuracy of description, while Autoclass did the best in terms of using the smallest number of rules. However, the GA should likely be considered to have achieved the best performance on this task. It is far preferable to Autoclass because Autoclass achieved such a low accuracy on the task, even though it required half as many rules to do so. Furthermore, Autoclass produces probabilistic rules, which are not easily interpretable by humans.

To see why the GA should be considered better than decision trees, one must consider a few factors not apparent in the above table. The first is that although the GA missed seven of the positive examples, finding one rule to explain each of these examples is a trivial exercise, requiring only a translation of the example into a rule. (Furthermore, with only slightly more effort, each of these rules could be written in a generalized form, containing some "don't cares", so that it would not be a literal representation of the one example, and would still exclude all negative examples.) This exercise was not performed because a rule that explains only one example is not very useful. As an extreme example of this, the entire dataset could have quite easily been explained with 100% accuracy by using 573 rules. But this does not shed any insight into the underlying structure of the data. Likewise, any rule that explains only one example does not serve to identify any patterns.

It would not be unreasonable for our task to exclude from consideration any rule that explains only one example, which would result in the following table:

system	number of rules	a	d	fitness	accuracy	average examples/rule
GA	59	246	319	0.89	98.6%	9.71
GID3*	113	215	288	0.65	87.8%	5.07
A-CLASS	30	160	220	0.55	66.3%	19.1

This is an extreme comparison, and is not fair to GID3* because it ignores the decision tree formulation itself. That is, any rule that explained only one example was simply removed from the rule set. A more realistic comparison would be obtained by first pruning the tree and then using the new tree to form a new rule set. For GID3*, this is estimated to produce a rule set with 124 rules that correctly explains 218 positive examples and 296 negative examples, for a fitness of .67 and an accuracy of 89.7%. (Pruning as a method of increasing the simplicity of a decision-tree classification is explored by Bohanec and Bratko in [11].) To be even more fair to decision trees in this comparison, we would want to prevent the tree from ever making a split that put only one example in a node in the first place. This might lead GID3* to choose different attributes to branch on. However, removing the one-example rules is still a very illustrative exercise because it demonstrates that although the decision tree systems achieved the highest accuracy, they did so by using a large number of rules that explain only one example.

A breakdown of how many rules explain a given number (or range) of examples is given in the table below:

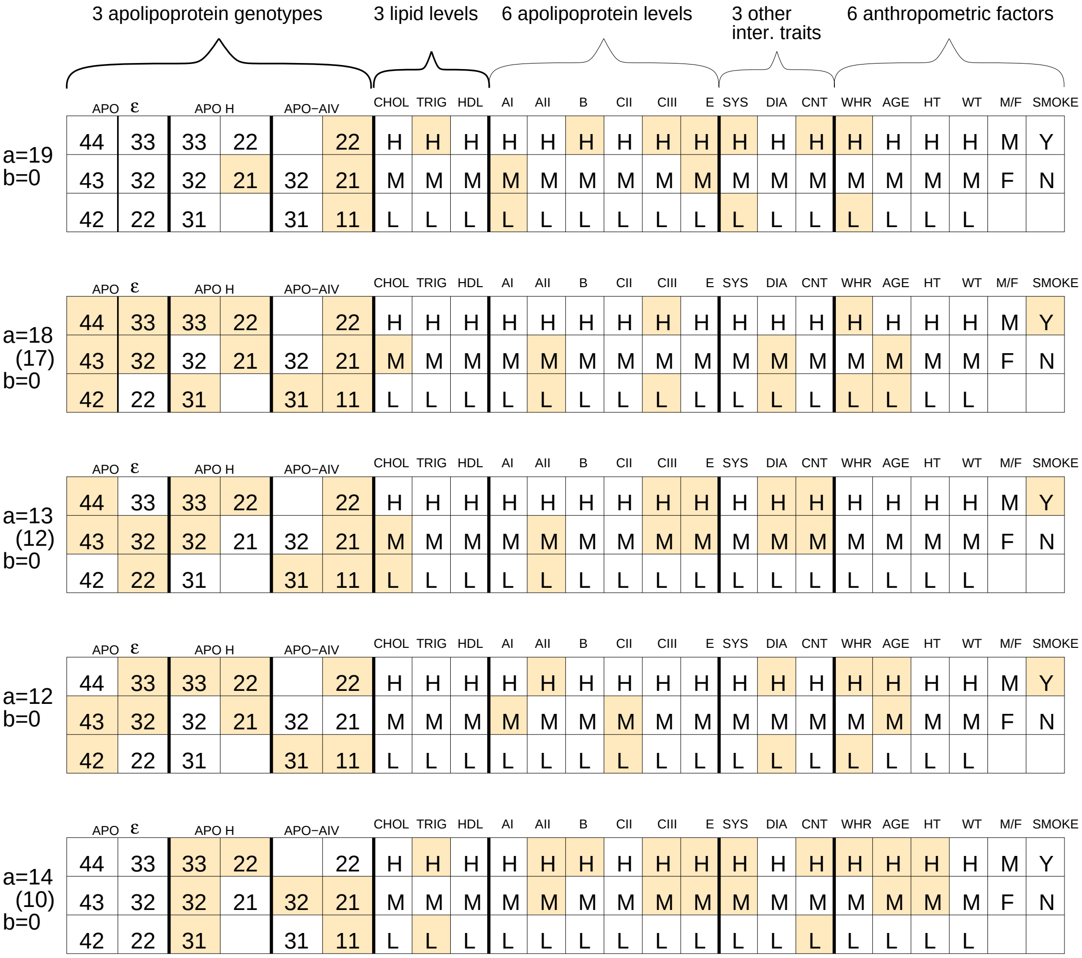


Figure 8.1: An illustration of the five best rules for the GA. The grid represents all the possible attribute-values that might be specified in a rule; the shaded areas indicate the rule itself. To the left of each grid are the number of true positives, a, and false negatives, b, explained by each rule; the numbers in parentheses are true positives that nave not been explained by a better rule.

	total	number of rules that explain n people					
system	rules	1	2	3	4	5-10	10+
GA	60	1	0	0	3	40	16
GID3*	183	70	35	22	14	36	6
A-CLASS	30	0	0	0	0	7	23

The above table illustrates that for this task GID3* relies heavily on rules that do not describe very many examples, whereas the GA and Autoclass tend to find rules that explain 5 or more examples. This difference highlights another reason that smaller rule sets are considered preferable (in addition to being easier for humans to understand): the more examples

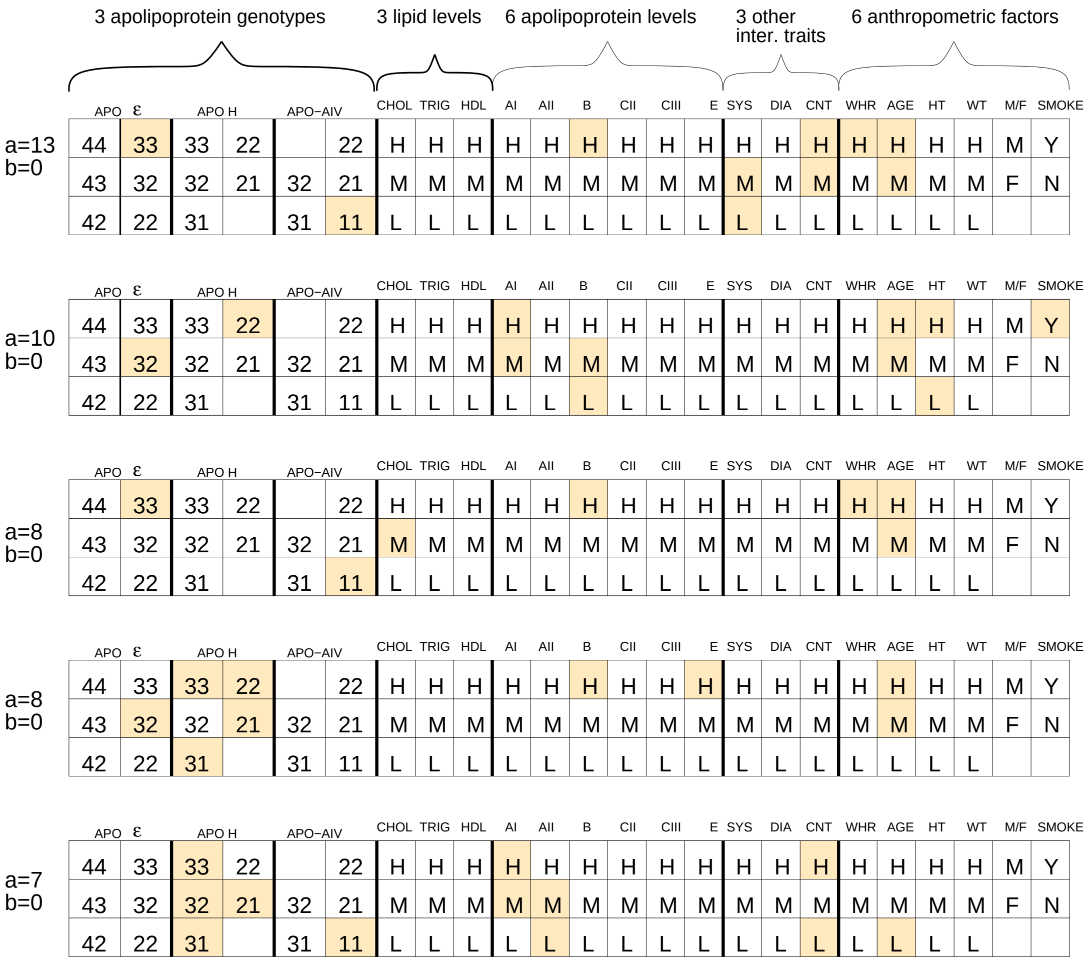
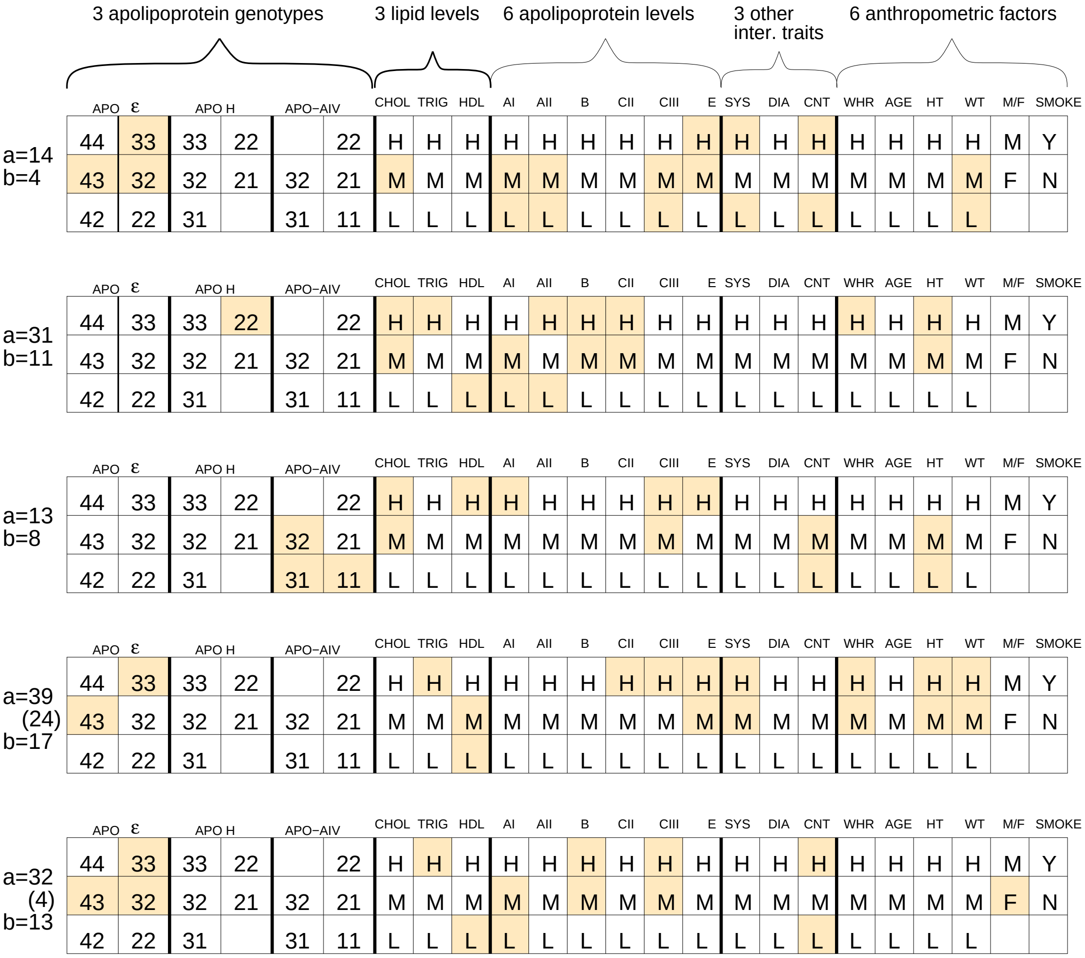


Figure 8.2: An illustration of the five best rules for GID3*. The grid represents all the possible attribute-values that might be specified in a rule; the shaded areas indicate the rule itself. Since the rule set found by GID3* partitions the data, there are no examples explained by more than one rule.

---

explained by a rule, the higher confidence we can have in that rule as being descriptive of a pattern in the data. These results may reveal a strength of the GA formulation in that it does not partition the dataset. There is a pressure in the GA to save those rules that explain examples not explained by other rules, but a rule that explains one new example and one previously explained example will have a slightly better chance of being saved into the next generation than a rule that explains one new example and no previously explained examples. (This is a byproduct of the population-based fitness implementation, since the rules are first ranked according to the total number of examples they explain, regardless of whether these examples have been explained by other rules or not.)

There is one regard in which the rules from GID3* may be considered better for the de-



	APO ε		APO H		APO−AIV		CHOL	TRIG	HDL	AI	AII	B	CII	CIII	E	SYS	DIA	CNT	WHR	AGE	HT	WT	M/F	SMOKE
a=14 b=4	44	**33**	33	22		22	H	H	H	H	H	H	H	H	**H**	**H**	H	**H**	H	H	H	H	M	Y
	43	**32**	32	21	32	21	**M**	M	M	**M**	**M**	M	M	**M**	**M**	M	M	M	M	M	M	**M**	F	N
	42	22	31		31	11	L	L	L	**L**	**L**	L	L	**L**	L	**L**	L	**L**	L	L	L	**L**		

	APO ε		APO H		APO−AIV		CHOL	TRIG	HDL	AI	AII	B	CII	CIII	E	SYS	DIA	CNT	WHR	AGE	HT	WT	M/F	SMOKE
a=31 b=11	44	33	33	**22**		22	**H**	**H**	H	H	**H**	**H**	**H**	H	H	H	H	H	**H**	H	**H**	H	M	Y
	43	32	32	21	32	21	**M**	M	M	**M**	M	**M**	**M**	M	M	M	M	M	M	M	**M**	M	F	N
	42	22	31		31	11	L	L	**L**	**L**	**L**	L	L	L	L	L	L	L	L	L	L	L		

	APO ε		APO H		APO−AIV		CHOL	TRIG	HDL	AI	AII	B	CII	CIII	E	SYS	DIA	CNT	WHR	AGE	HT	WT	M/F	SMOKE
a=13 b=8	44	33	33	22		22	**H**	H	**H**	**H**	H	H	H	**H**	**H**	H	H	H	H	H	H	H	M	Y
	43	32	32	21	**32**	21	**M**	M	M	M	M	M	M	**M**	M	M	M	**M**	M	M	**M**	M	F	N
	42	22	31		**31**	**11**	L	L	L	L	L	L	L	L	L	L	L	**L**	L	L	**L**	L		

	APO ε		APO H		APO−AIV		CHOL	TRIG	HDL	AI	AII	B	CII	CIII	E	SYS	DIA	CNT	WHR	AGE	HT	WT	M/F	SMOKE
a=39 (24) b=17	44	**33**	33	22		22	H	**H**	H	H	H	H	**H**	**H**	**H**	**H**	H	H	**H**	H	**H**	**H**	M	Y
	43	32	32	21	32	21	M	M	**M**	M	M	M	M	M	**M**	**M**	M	M	**M**	M	**M**	**M**	F	N
	42	22	31		31	11	L	L	**L**	L	L	L	L	L	L	L	L	L	L	L	L	L		

	APO ε		APO H		APO−AIV		CHOL	TRIG	HDL	AI	AII	B	CII	CIII	E	SYS	DIA	CNT	WHR	AGE	HT	WT	M/F	SMOKE
a=32 (4) b=13	44	**33**	33	22		22	H	**H**	H	H	H	**H**	H	**H**	H	H	H	**H**	H	H	H	H	M	Y
	43	**32**	32	21	32	21	M	M	M	**M**	M	**M**	M	**M**	M	M	M	M	M	M	M	M	**F**	N
	42	22	31		31	11	L	L	**L**	**L**	L	L	L	L	L	L	L	**L**	L	L	L	L		

Figure 8.3: An illustration of the five best rules for Autoclass. The grid represents all the possible attribute-values that might be specified in a rule; the shaded areas indicate the rule itself.

scription task than the other systems: when one looks at the rules formed by each of these systems, one sees that the rules formed by GID3* are simpler than the rules formed by the other two systems in that they generally specify fewer attributes. The best five rules from the GA, GID3*, and Autoclass are illustrated in Figures 8.1–8.3. Autoclass produces probabilistic rules that mention the probability of each value for each attribute; these have been translated into discrete rules using the very ad hoc process of looking at each attribute and the probabilities for each of the values, and selecting the values that have the highest probabilities. The entire rule set from the GA is provided in Appendix E; the entire rule set from GID3* is provided in Appendix F.

In each of the illustrations of the five best rules, the grid indicates the possible attribute-values that could be specified by a rule and the shaded area indicates the attribute-values actually specified. (There are two genotypes, APO H*11 and APO AIV*33, that are not

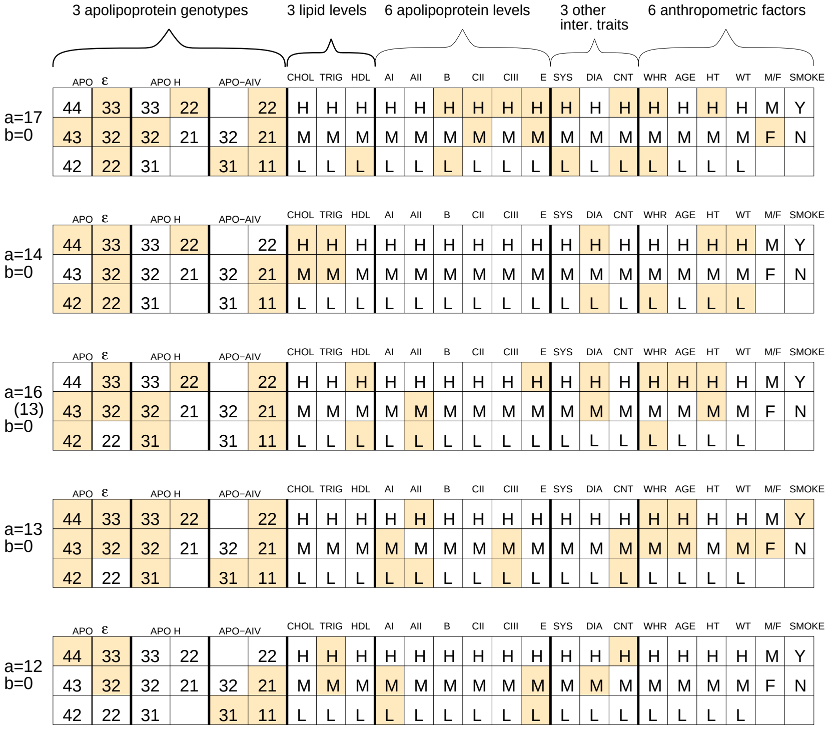


Figure 8.4: An illustration of the five best rules for the GA with $w = .75$ in the fitness function.

represented in our dataset; these squares of the grids are left blank.) The first thing to notice is that GID3* required the fewest attributes per rule, averaging about 6 attributes specified per rule (for the best five rules), while the GA averaged about 10 specified attributes per rule and Autoclass averaged 9 specified attributes per rule. Using fewer attributes is preferable for the "human interpretability" component of the task. The second thing to notice is the number of attributes specified across all five of the best rules. GID3* refered to only 14 of the 21 available attributes in the five best rules, whereas the GA and Autoclass both refered to 18. (However, it is well to note that when one looks at the entire rule set, rather than the five best rules, all systems used all of the available 21 attributes). It is also worthwhile to remember that the GA explained 70 positive examples, with its five best rules, while GID3* explained 46 positive examples. That is, it is perhaps *because* the GA used more attributes that it was able to find better rules.

For completeness, the five best rules for the GA runs with the weighting function are illustrated in Figures 8.4–8.6 (when $w = .25$, there are only three rules in the entire rule set). It is interesting to note that the five best rules found by the GA with weights $w = .75$ and $w = .50$ are of approximately the same descriptive power as the five best rules found with the GA using $w = 1.0$. These five best reuls explain 69 examples with $w = .75$ and 74 examples with $w = .50$, as compared with the 70 examples explained for $w = 1.0$. Note that changing

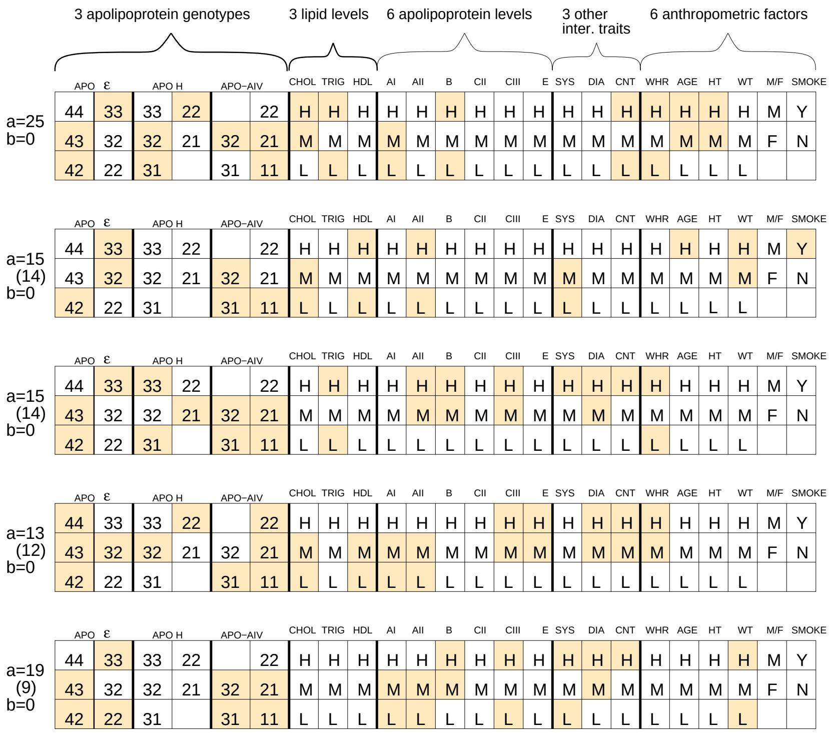


Figure 8.5: An illustration of the five best rules for the GA with $w = .50$ in the fitness function.

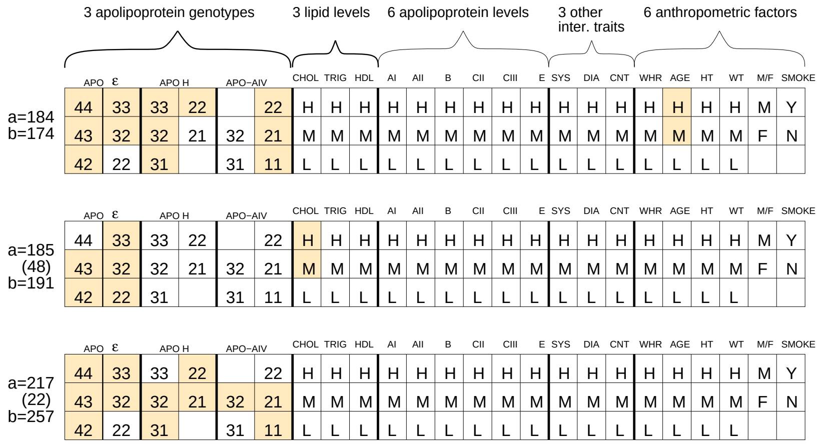


Figure 8.6: An illustration of the rule set found by the GA with $w = .25$ in the fitness function. These three rules represent an entire rule set for this weight.

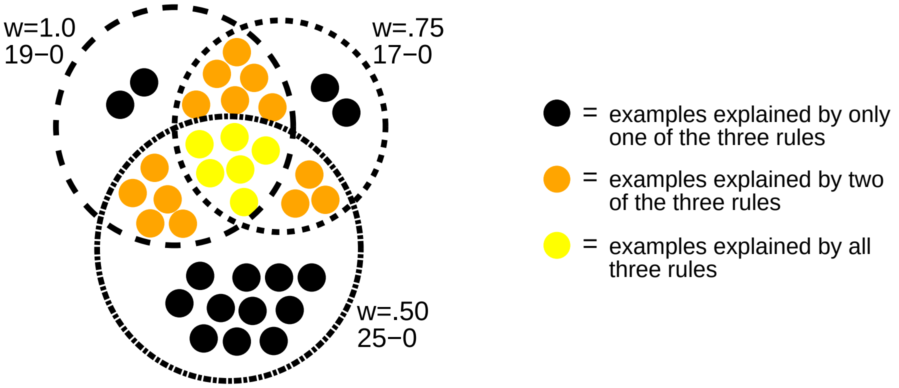

w=1.0
19−0
w=.75
17−0
w=.50
25−0
● = examples explained by only one of the three rules
● = examples explained by two of the three rules
● = examples explained by all three rules

Figure 8.7: A Venn diagram illustrating the overlap in the examples explained by the best rule from the GA runs with weights set to $w = 1.0$, $w = .75$, and $w = .50$.

the value of w does not seem to affect the average number of attributes per rule for the five best rules at $w = 1.0$, $w = .75$, and $w = .50$ (the averages are 10.4, 10.2, and 10.6 examples per rule, respectively). Also note that although the GA with $w = .50$ was able to explain more examples in its five best rules than the GA with $w = 1.0$, when one looks at the entire rule sets, the GA with $w = .50$ is clearly worse at description (see the ROC curve in Figure 5.9, for example).

It is interesting to look at the examples explained by the best rule from the weighting runs with weights set to $w = 1.0$, $w = .75$, and $w = .50$ ($w = .25$ is not included because it explains so many examples). The Venn diagram in Figure 8.7 illustrates that there is a large overlap in the examples explained by these three rules. For example, the best rule from the $w = 1.0$ run and the best rule from the $w = .75$ run explain 12 of the same examples.

The Venn diagram in Figure 8.8 illustrates that the rules found by different variations on the GA have more examples in common that the rules found by different systems. This raises an interesting question that will not be explored further in this thesis: are the different systems exploring different parts of the search space? Intuitively, we might expect the answer to this question to be yes, since each system uses a different approach to search through the space of possible rules.

The prediction task

The results of the prediction task are no doubt what most machine learning researchers will focus their attention on, even though it has been stated many times that this is not a primary concern from the perspective of common-disease research (i.e., it is not really a component of this task). These results are summarized in the following table:

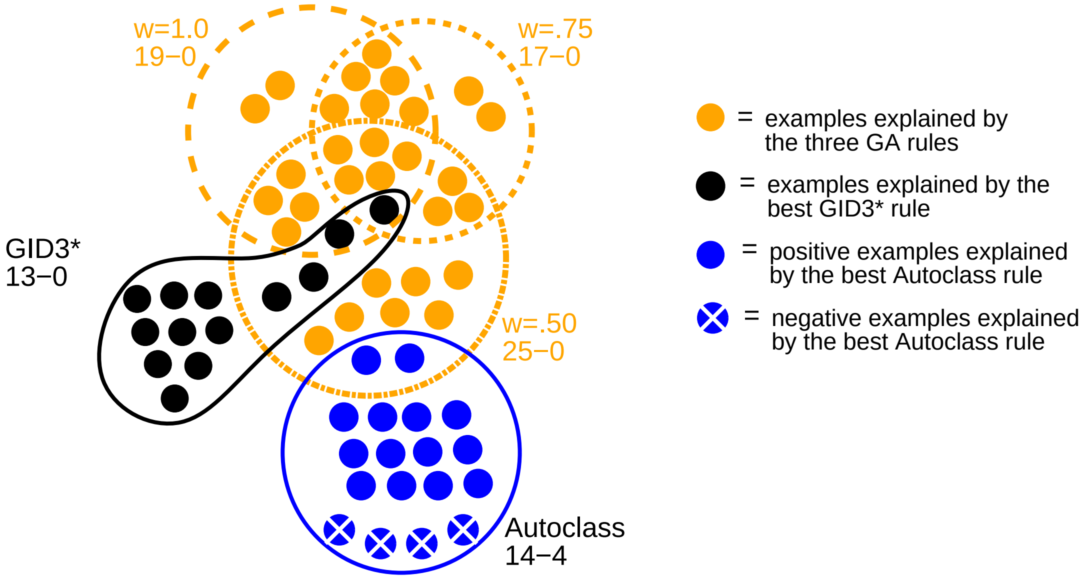

w=1.0
19–0
w=.75
17–0
GID3*
13–0
w=.50
25–0
Autoclass
14–4
= examples explained by the three GA rules
= examples explained by the best GID3* rule
= positive examples explained by the best Autoclass rule
= negative examples explained by the best Autoclass rule

Figure 8.8: A Venn diagram illustrating the overlap in the examples explained by the best rules from the GA runs and the best rule found by GID3* and Autoclass.

	averages across five testing/training runs					
	number			number	prediction	prediction
system	of rules	a	d	correct	fitness	accuracy
GA	54	25	35	60	0.51	52.2%
GID3*	159	20	39	59	0.50	51.7%
A-CLASS	30	19	43	62	0.51	53.6%

From this table, it is clear that all of the systems performed poorly on the prediction task. This is probably due to the complexity of the problem itself, which will be discussed in the next section. Using paired t-tests, the differences between any two of these three systems is not significant. Using a t-test, only Autoclass is significantly better than an unbiased coin ($p < .1$), and all are significantly ($p < .025$) worse than a single rule that classifies all examples as negative.

The performance of these systems might also be compared to a biased coin, which selects "negative" as the class 55.7% of the time, reflecting the distribution of examples in the full dataset. Using this biased coin, we would expect to classify 55.7% of the negative examples correctly, and 55.7% of the positive examples incorrectly (44.3% of the positive examples correctly). This biased coin would be expected to get an average of 21.54 positive examples and 36.97 of the negative examples correct — a total average of 58.51 examples correct, and 50.65% correct. None of the three systems performance is significantly different from this biased coin.

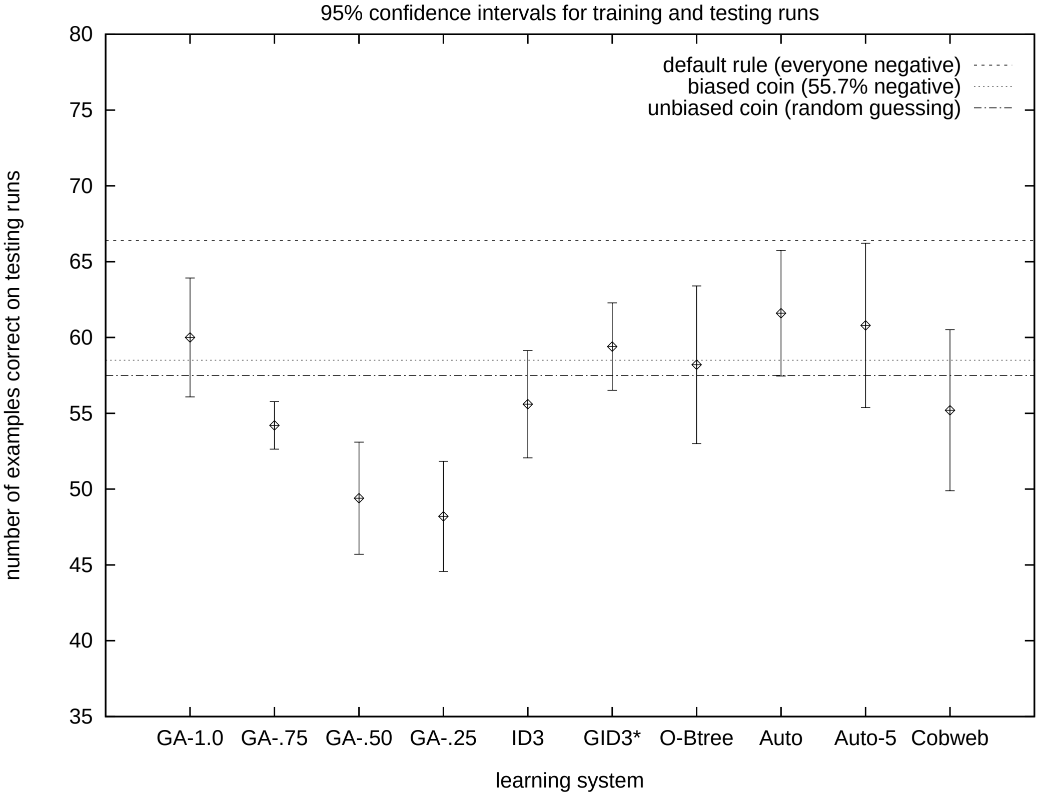


Figure 8.9: 95% confidence intervals for the prediction task are shown for all systems, including different weights for the GA fitness function and different decision tree systems.

The 95% confidence intervals for the prediction task are shown in Figure 8.9 for all systems. The topmost horizontal line indicates the expected prediction accuracy of a single rule that assigns all examples to the negative class; none of the systems evaluated in this thesis were able to achieve this minimal level of performance on the prediction accuracy task. The lower horizontal line indicates the expected performance of an unbiased coin; only five systems achieved a performance better than this. The middle horizontal line indicates the expected performance of a biased coin; only four systems achieved a performance better than this.

We see a slightly different pattern across the performance abilities of the different systems if we consider the abilities of the systems to predict positive examples separately from their abilities to predict negative examples:

system	correct positive examples	correct negative examples	percent positive correct	percent negative correct
GA	25	35	50.6%	53.3%
GID3*	20	39	41.6%	59.0%
A-CLASS	19	43	37.9%	65.1%

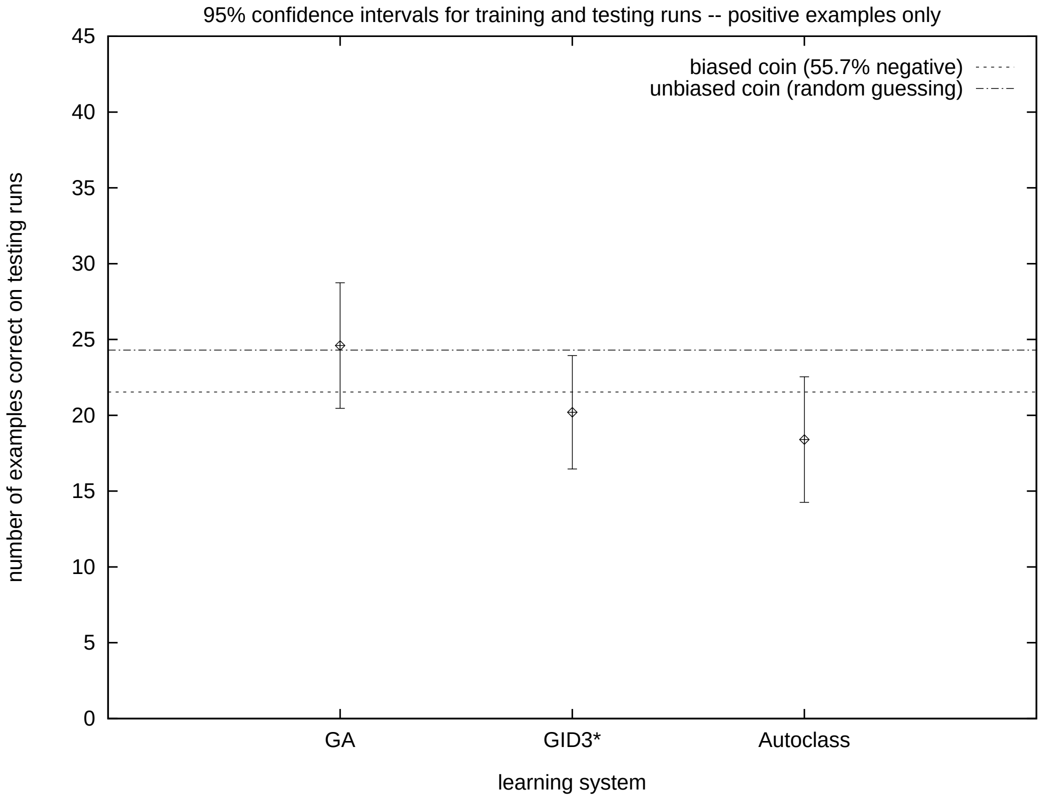


Figure 8.10: 95% confidence intervals for the prediction task with positive examples only are shown for the three best systems (GA, GID3*, and Autoclass).

From the table above, we see that the GA was the best of these systems in terms of its ability to predict positive examples correctly. GID3* and Autoclass did significantly worse ($p < .05$) than random chance on predicting positive examples, while the GA did not. The GA also did not do significantly better than random chance on predicting negative examples, while both GID3* and Autoclass were significantly better than random chance when predicting the class of negative examples, with ($p < .05$) and (($p < .005$), respectively.

A biased coin that favors negative examples at the same rate as the original dataset (55.7% negative examples) would be expected to get an average of 21.54 positive examples correct; none of the three systems performance is significantly different from this biased coin in classifying positive examples. The biased coin would be expected to get 36.97 of the negative examples correct; only Autoclass' performance is significantly ($p < .001$) better than this biased coin in classifying negative examples. The confidence intervals for positive examples only is shown in Figure 8.10; the confidence intervals for negative examples only is shown in Figure 8.11.

This analysis illustrates the classification biases of these three systems, as they relate to a dataset with an uneven number of positive and negative examples. We designed our GA to

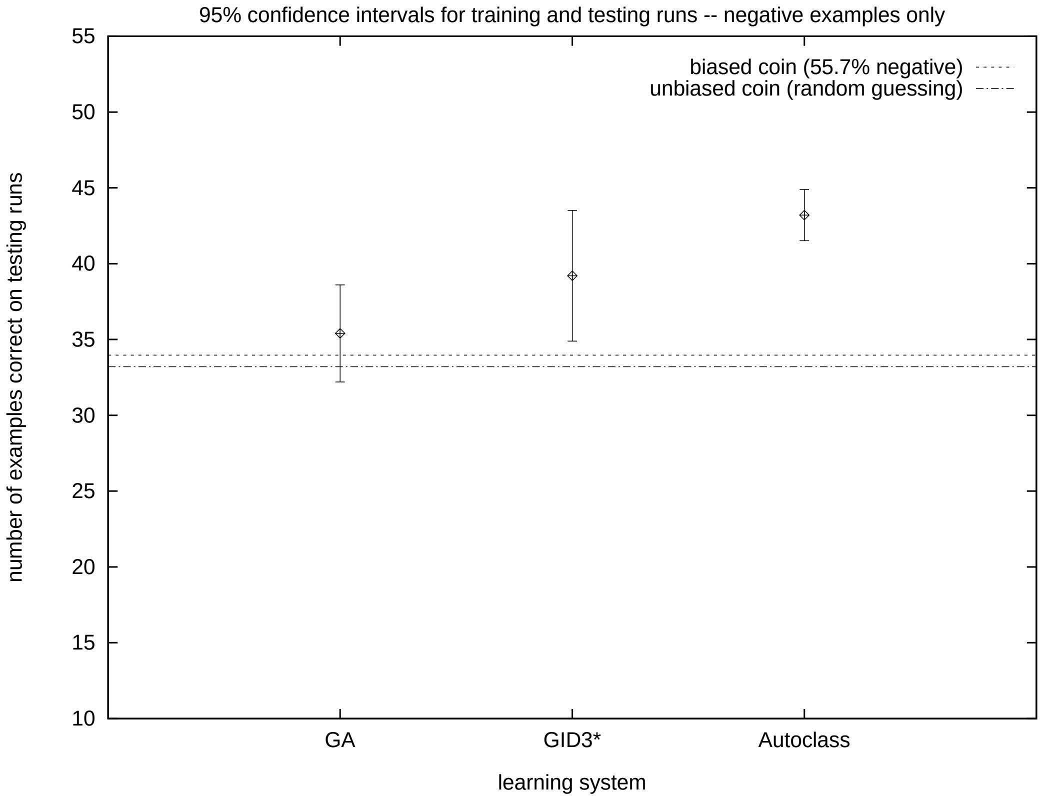


Figure 8.11: 95% confidence intervals for the prediction task with negative examples only are shown for the three best systems (GA, GID3*, and Autoclass).

favor the classification of positive examples over the classification of negative examples, for example, and as it turns out, our GA is slightly better at classifying positive examples and slightly worse at classifying negative examples, than one would expect with the biased coin described above. The information gain function used by GID3* tends to classify positive and negative examples correctly in proportions roughly equivalent to that of the original dataset. And Autoclass is much better at classifying negative examples correctly, probably because it can find stronger classifications by relying more heavily on the class that is more strongly represented in the dataset.

Paired t-tests between any two of these three systems show that the GA does significantly better than GID3* and Autoclass in predicting positive examples correctly ($p < .05$), and significantly worse than GID3* and Autoclass in predicting negative examples correctly ($p < .1$ and $p < .025$, respectively). The differences between decision trees and Autoclass are not significant when predicting the class of either positive or negative examples.

Other points of comparison

From a practical standpoint, another consideration when comparing systems is the length of time each of these systems takes to run. The following table gives an approximate estimate of the run time for each system when run on a Sun IPX Sparcstation:

system	relative run time
GA	3 days
GID3*	1 minute
A-CLASS	1 day

It is important to note when considering the relative run times that of all the systems run, only TreeMaker, on which the decision trees were run, was written to be fast. The other systems used were designed to be portable to various hardware platforms and to be modifiable by the user.

8.2 Criteria to be Considered in Comparing the Complexity of Tasks

The results presented in the previous section illustrate the complexity of the task investigated in this thesis. Although several systems were able to describe the dataset (albeit with varying success), none exibited better than a cursory capacity for finding rule sets that generalize to unseen examples. Before this research was undertaken, we realized that we were dealing with a "complex" task, but had no real sense of how much more difficult this task would be than most pursued in machine learning, nor how the complexity of the task would manifest itself in the research results.

A formal definition of complexity is beyond the scope of this thesis. However, this section of the thesis investigates several of the criteria that should be considered when evaluating the complexity of a machine learning task.

In the course of the thesis work, I've realized a number of factors that one should consider when evaluating the complexity of a classification task:

1. *Is one attribute highly correlated with the class attribute?*
   If the answer to this question is yes, the data should be relatively easy to classify, regardless of other factors.

2. *How many attributes are included in the dataset?*
   With a small number of attributes, it may be possible to enumerate all of the possible classifications in a relatively short period of time. (Although a very small number of attributes is likely to make the problem more difficult in that there may not be enough attributes to differentiate the examples.) However, a large number of attributes increases the size of the search space; a very large number of attributes may do more

to complicate the classification rather than simplify it. This is sometimes referred to as the "curse of dimensionality": the higher the dimensionality, the sparser and more spread apart are the data points.

3. *How big is the search space?*
   The size of the search space for n attributes is the number of values for the first attribute times the number of attributes for the second attribute, etc., etc., times the number of values for the nth attribute. (For the common-disease research task, this is 6*6*6*3*3*3*3*3*3*3*3*3*3*3*3*3*3*3*3*2*2 $= 3.719 * 10^{10}$.) Obviously, the larger the search space, the harder it will be for a classification system to find good rules.

4. *How many examples are included in the dataset?*
   A classification system will generally be able to do a better job of classifying data if provided with more examples. This applies both to the description and to the prediction task; fewer examples make generalization more difficult because it is harder for the system to find patterns in the data.

5. *How big is the data set, relative to search space?*
   More relevant than the number of examples in the dataset is the ratio of the size of the search space to the number of examples in the dataset. If this ratio is high, the search space will likely be too sparsely populated with points for the classification system to arrive at a good classification.

6. *What is the distribution of the classes in the dataset?*
   A dataset that is split 50-50 between the positive and negative examples will be easier for many systems to classify than one that is not split evenly. A lopsided dataset, say with 90% positive examples, may be harder to classify because there may not be enough negative examples to find a good classification. This concern is also relative to the size of the search space.

7. *Is there noise in the dataset?*
   The term "noise" refers to attribute-values (including the class attribute) that may be incorrectly measured or recorded in the dataset. This can considerably complicate a classification task because a classification system has no means of determining which attribute-values are correct and which are erroneous. Most systems assume noise-free data, and attempt to find a consistent classification of the data as it appears in the dataset.

   Another type of "noise" refers to the situation that arises when not enough attributes have been included in the dataset; this often leads to a situation in which there are two examples that have the same attribute-value profile, but belong to different classes. When this happens, the datset is called ambigious; the classification system has no means of determining whether ambiguity in the dataset is due to errors or due to not enough attributes. It is impossible to find a consistent classification of the data in this situation.

8. *How many rules are needed to explain the data?*
   This metric is specific to any particular classification system, but is useful for comparing different datasets. More rules/example implies a more difficult task.

9. *Are the attributes independent?*
   Many classification systems assume that the attributes are independent as a simplifying assumption because the attributes may then safely be considered individually. However, if two attributes are not independent to the point that they are very highly correlated, they are redundant, which would greatly simplify the problem. The difficulty arises somewhere in the middle, when attributes are neither independent nor highly correlated.

10. *Do the attributes interact with the class attribute?*
    Not only may attributes interact to different degrees, but one should consider not only pairwise interactions, but also interactions of three or more attributes. Most machine learning systems ignore the issue of interaction between attributes.

Additional considerations raised in this thesis address the classification task itself, rather than the dataset:

11. *Is it important to have correct rules for one particular class?*
    Although an assymetry in the importance of correctly classifying different classes does not necessarily make the task itself more complex, it may make the task more difficult for a particular system. Few classification systems consider this possibility, and hence, have a more difficult time classifying data with this additional consideration.

12. *Is it important to have a small rule set?*
    Again, this concern does not necessarily make the task itself more difficult, but may make it more difficult for a particular classification system.

The above list has been restricted to issues relevant to the thesis task; there are many other factors that may contribute to the complexity of a task. Among these are:

13. *How many classes are represented in the dataset?*
    In the thesis, I've focused on a binary classification task; a task with more than two classes may be more difficult to classify, given the same number of examples, simply because each class will have fewer examples to represent it.

14. *Does the task include continuous attributes?*
    The thesis addresses a task with discrete attributes, which considerably simplifies the classification task. Continous attributes require that the classification system determine for itself where to subdivide an attribute into relevant ranges, as well as how many ranges to use.

15. *Does the task includes a mix of continuous and discrete attributes?*
    A mix of continuous and discrete attributes may be more difficult for some classification systems. For example, systems that rely on statistical models may have a

more difficult time because different statistical models are usually used to interpret continuous data versus discrete data.

16. *Does the task includes data with missing values?*
A "missing value" occurs when the value of an attribute is unknown for a particular example. This does not present a major difficulty for a discrete classification task, because the value of "missing" can be added to the range of values for the attribute. It may present a considerable difficulty, however, for a continuous classification task.

8.3 Comparison of a Few Typical Machine Learning Datasets Using GID3*

Although establishing concrete metrics to compare the complexity of datasets is not within the scope of this thesis, a quick gauge of the relative complexity can be achieved by running a decision tree system on the various tasks. This approach was suggested to me by Schlimmer [88]; the approach is also recommended by Buntine in [15].

This section presents the results from running GID3* on some well-known machine learning datasets, to be compared against the common-disease research task. Note that this evaluation does not take into account additional considerations such as asymmetry or size of rule set.

The additional datasets used here were chosen from the UCI repository of datasets [105] based on two considerations:

1. That the dataset use discrete attributes
2. That the dataset be relatively well-known in the machine learning community.

The datasets compared in this section are described briefly below, along with a list of some of the references to using each dataset in the machine learning literature.

Mushroom — This dataset includes examples taken from the Audubon Society Field Guide to North American Mushrooms. The data includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota family; each example is classified as edible or poisionous. The dataset was first used by Schlimmer [87], but has also been used by many other machine learning researchers [38, 39, 80, 84, 75, 110, 58]

Soybean — This dataset includes examples of soybean plants, each afflicted by one of 19 different diseases. The dataset was first used by Michalski and Chilausky [70], and has also been used frequently in the machine learning literature [69, 84, 97, 16, 102, 17].

Congressional voting record — This dataset lists data taken from the Congressional Quarterly Almanac, including votes for each of the members of the U.S. House of Representatives on 16 key votes identified by the CQA; there are two classes: democrat and republican. The dataset was first used by Schlimmer [87], but has also been used frequently in the literature [38, 59, 75, 102].

The table below gives the results from running GID3* on each of these datasets, as well as the original common-disease research task (labelled "CAD" in the tables below).

	number of examples	number of attributes	number of classes	size of search space
CAD	573	21	2	$3.719 * 10^{10}$
Mushroom	8124	22	2	$1.219 * 10^{14}$
Soybean	307	35	19	$4.514 * 10^{19}$
Voting	435	16	2	$4.305 * 10^{7}$

	number of rules	number of examples/rule	description accuracy	prediction accuracy
CAD	183	3.13	100%	51.7%
Mushroom	16	507.75	100%	100.0%
Soybean	46	6.67	100%	86.2%
Voting	32	13.59	100%	94.9%

The three datasets commonly used in machine learning research are relatively easy to classify, whereas the common-disease research task pursued in this thesis is relatively difficult.[1] My intuitions for the reasons for this difference are described in the following paragraphs.

The mushroom dataset is by far the largest and by far the easiest to classify. The reason for this is almost certainly the fact that one of its attributes (odor) is highly correlated with the class attribute (roughly 90% of the dataset can be classified correctly on the description task using this one attribute alone). (The final rule set found by GID3* uses only 9 of the 22 attributes.)

The soybean dataset has a large number of classes, but is still relatively easy to classify; I believe this is due at least in part to the fact that 8 of the 35 attributes are not needed for the final rule set. (Note that with 19 classes, at least 19 rules would be required in the final rule set, so 46 rules can still be considered small.)

The congressional voting dataset in many regards seems to be the most similar to the common-disease research task in that it has roughly the same number of examples, the same number of classes, and the roughly the same number of attributes. And yet, GID3* is able to find a rule set of only 32 rules to describe this dataset, using 14 of the 16 possible attributes. The voting task turns out to be much easier: 221 of the 267 democrats are described by a single rule (adoption-of-the-budget yes; immigration no vote; physician-fee-freeze not yes), and 118 of the 168 republicans can be described by a single rule (adoption-of-the-budget not yes; duty-free-exports not yes; synthetic-fuels-corporation-cutback not yes; physician-fee-freeze yes). In fact, this dataset can be described with 95% accuracy by using the physician-fee-freeze attribute alone.

[1] Holte [55] calls these three tasks (as well as several others commonly used in machine learning research) "easy", but defends their use in machine learning research as natural problems, not representative of the entire range of "real" problems, but still representative of a class of problems that naturally arise in practice.

The common-disease research task would seem to be more difficult for several reasons. The first is the size of the search space, relative to the number of examples (and that all of the attributes are used in the final rule set found by GID3*). The second is that the attributes *do* appear to interact with the class attribute, based on the fact that the GA was able to find better rules than GID3* by using more attributes for each rule. But perhaps the largest contribution to the difficulty of the common-disease research task on the performance task is the presence of noise in the dataset. Most classification systems, including our GA and GID3*, place an emphasis on finding "pure" rules: those that explained only positive or negative examples. In our GA, this was manifest by the use of the fitness function (derived from odds ratio), which places a steep penalty on rules that do not have $b = 0$. In GID3*, this is manifest by the stopping criteria; the system will keep subdividing nodes of the tree until all the examples in a node are the same class (or until it runs out of attributes). In retrospect, this emphasis is probably undesirable for our task.

CHAPTER 9

SUMMARY OF THE THESIS

In the early chapters of the thesis, I presented a problem from the field of common-disease research, and outlined the issues raised for common-disease researchers as well as those raised for machine learning researchers. The general problem is to describe individuals who have a family history of CAD. From the perspective of common-disease research, the research attempts to encourage a paradigm shift because traditional approaches in common-disease research assume that the dataset is to be explained by a single description, whereas the thesis research allows for a conjunctive description of the individuals with a family history of CAD. From the perspective of machine learning, the thesis research encourages researchers to consider smaller rule sets than those needed to describe a dataset with 100% accuracy. From the perspective of common-disease research, we want to find *larger* rule sets than researchers usually consider, as a means of increasing descriptive accuracy; from the perspective of machine learning, we want to find *smaller* rule sets than researchers usually consider, in the hopes of increasing statistical confidence in the rules.

From the perspective of machine learning, the thesis hopes to illustrate that many of the tasks commonly investigated in the course of machine learning are relatively simple, and suggests that research would be strenghthened by using more complex tasks. Furthermore, there is a tendency in machine learning research to homogenize the tasks undertaken, so that they have similar structure; these simplifications may be unwarranted if one considers the underlying “real” task associated with the data. The use of a real dataset obtained from researchers in another field serves to prevent machine learning researchers from making simplifying assumptions about how the data is to be used, thus presenting an additional challenge for machine learning researchers.

9.1 What We Can Learn from the Thesis Work

The last few chapters of the thesis presented a number of results, which I will briefly summarize here, and also raised a number of issues that may not be so obvious.

9.1.1 The obvious results

The main results from the previous chapters are summarized in the following list:

1. *The GA outperformed other systems investigated in this research.*
   This is undoubtedly due in large part to the fact that GA's are not used exclusively (or even usually) for classification, and therefore have fewer built-in assumptions that constrain the rule sets the system is able to find. Instead, the constraints come largely from the user-defined fitness function.

2. *Decision trees still have a lot to offer for this task.*
   Even though the decision tree systems used a rather large number of rules to describe the data, they are markedly faster than the GA. Furthermore, the individual rules found by GID3* were simpler than the individual rules found by the GA in that they refered to fewer attributes.

3. *All systems did poorly on the prediction task.*
   Autoclass did the "best" on the prediction task, but this was not significantly better than the performance of other systems investigated in this thesis. Furthermore, the rule set found by Autoclass was still worse than a single rule that predicts all examples are negative.

4. *It is likely that the inability of any of the systems to perform well on the prediction task is due in large part to the noise in the dataset.*
   In our task, there is a large degree of noise that is due to the fact that the physiology of the children does not correspond exactly to disease in the parents. That is, the class attribute itself is very noisy, and we should not even be trying to find classifications that are "100% correct". This issue seems to present problems both with the fitness function we selected for the GA as well as with the evaluation metric and decision procedure used by GID3*.

5. *The weighting function did not lead to the smooth gradation of "grain size" in the rule sets that we might have hoped for.*

   The weighting function developed in Chapter 4 worked fine for its purpose as stated (and originally viewed), in that decreasing the weight produced increasingly smaller (and less accurate) rule sets when used in the GA. However, the resulting rule sets tended to have a few large and largely inaccurate rules mixed in with a few smaller and more accurate rules. In retrospect, what we really would have liked is a means of controlling the expected number of examples explained by each rule.

6. *The GA is significantly better at predicting the class of positive examples (and significantly worse at predicting the class of negative examples) than the other systems investigated in this research.*

   Although the performance of the three best systems (the GA, GID3*, and Autoclass) was not significantly different on the overall prediction task, a difference was observed when the testing data was divided into separate test sets of positive and negative examples. While the GA's performance on either subset was not significantly different from that of an unbiased coin, both GID3* and Autoclass did significantly worse than an unbiased coin in predicting the class of positive examples ($p < .05$) and significantly better than an unbiased coin in predicting the class of negative examples ($p < .05$ for

GID3* and $p < .005$ for Autoclass). While the GA was not better at predicting the class of positive examples than it was at predicting the class of negative examples (which is what we might have hoped), it was at least significantly better than the other systems at predicting positive examples.

9.1.2 The not-so-obvious results

I have also learned a number of things that may not be obvious from the previous sections:

1. *The odds ratio (and its modifications) is probably not the best function to use for our task.*

   The *odds ratio* function, and its modification *log(odds ratio)*, used in the thesis, is a curious function. It was selected for use at the beginning of this research project because it was very familiar to epidemiologists and because its use in the GA seemed to encourage "good" rules and discourage "bad" rules. However, I now feel that the use of this function was perhaps misplaced. When used in epidemiology, the *odds ratio* function is used to *evaluate* the quality of a test for a disease; what we are doing is comparable to using it to *develop* the test for a disease. In its usual usage, an odds ratio of over 25 is rarely seen. That is, the usual usage avoids the steep inclines that occur when a is at a minimum and when d is at a maximum.

   This discrepancy is in part due to the "one-rule" models of the traditional paradigm in epidemiology. Because the data is usually complex, a single explanation of the etiology of a disease is unlikely to be very accurate. In our use of the odds ratio with the GA, we are looking for an undetermined number of rules, and search for the rules with the highest odds ratio; the search naturally tends to follow the ridge of $b = 0$ solutions.

2. *The odds ratio function does not allow us to adjust the balance between concerns for false positives and false negatives.*

   The *odds ratio* function strikes an appropriate balance *for us* between discouraging false positives and discouraging false negatives; however, it does not address this issue in a general way that would allow changing the balance.

3. *Immigration could still be useful for our task if used with a different intent.*

   Immigration was originally used as a means of preventing premature convergence within a GA population. However, the population-based fitness modification to the GA in order to evolve rule sets already does a wonderful job of preventing premature convergence. Where immigration might prove to be a useful addition *for our task* is in introducing strings that explain examples not already explained by the current population of strings.

   In hindsight, we should almost expect that immigration of random strings would not help our GA. Rather, what might help would be immigration of strings that represent examples from the dataset. Such strings might or might not explain any examples not already explained by higher-fitness strings, but they would certainly stand a better chance than a randomly generated string, which is likely to explain no new examples.

The fact that the hillclimbing procedure was able to find some reasonably good rules was the inspiration for incorporating it into the GA immigration procedure as an alternative to random strings. However, what I failed to notice at that time was that the hillclimbing procedure often turns up rules that explain NO examples — only 308 of the 16690 hillclimbing runs explained any examples at all.

9.2 Future Work

It's a strange feeling to "finish" a thesis project. One is struck by a feeling not only of the accomplishment of what has been done, but also by an overwhelming sense of what remains undone. And so it is with this project. There are a number of open questions that I hope to address in future research, but that could not be pursued within in the temporal and physical bounds of a reasonable thesis project.

I've found this project fascinating, and I hope to pursue some of these topics in the coming months:

1. *Addressing the issue of noise in the dataset.*

   Although I have come to believe that the noise in the dataset is one of the most complicating aspects of this task, both for description and for prediction, further investigation is needed to confirm this. This might entail experiments with different fitness functions, which might allow the GA to settle on rules that are not 100% accurate, but that describe more examples; the increased generalization of these rules might also lead to an improved performance on the prediction task.

2. *Finding or developing a better fitness function for use with the GA.*

   I am not entirely happy with the odds ratio as a fitness function, but nor am I clear as to a good alternative. It is not difficult to derive new functions using sensitivity and specificity as building blocks. However, since the new function would have no obvious meaning to common-disease researchers, it would be extremely difficult to communicate the merits of this unfamiliar metric. It is likely that performing these experiments first with the odds ratio and following up with an alternative function is a better course of action than beginning with the self-designed function.

3. *Finding or developing a better way to balance between false positives and false negatives.*

   This is of concern to me, as a machine learning researcher, rather than me, as someone who interacts with the common-disease research community. The issue of trading off between false positives and false negatives is not usually considered in machine learning research; the thesis presented it as an issue, but also presents the odds ratio function incorporated into the GA as the sole means of resolution to this issue. I suspect that a weighting function could be put to good use toward this end.

4. *Experimenting with variations on decision trees.*

   There are several variations of decision trees I would like to investigate with this task; I hope to soon have access to the source code for TreeMaker in order to enable these. First, I would like to try using the *log(odds ratio)* function and the weighting function developed in Chapter 4 as evaluation metrics for the decision trees, to see what effect this has on the resulting rule sets. Secondly, I would like to try changing the stopping criteria, as a means of preventing GID3* from finding rule sets that are 100% accurate. (It may be that a parameter to GID3* could be used to adjust the size of the rule sets found.) Finally, I would like to experiment with a splitting mechanism that uses combinations of attributes, rather than a single attribute, as is done with Fringe [78].

5. *Doing additional experiments with immigration.*

   The use of immigration is an appealing approach to me; at an intuitive level, it seemed that it should be helpful for our task. Before I give up on it completely, I would like to try experiments such as using immigration of examples from the dataset.

6. *Encouraging the GA to use fewer attributes to specify rules.*

   This might entail yet another level of modification to the GA fitness function, perhaps a small penalty for each attribute mentioned in a given rule. Alternatively, but with considerably more trouble, it could be incorporated into the sorting mechanism in the population-based fitness adaptation.

   It is difficult to anticipate how encouraging the GA to use fewer attributes might interact with whatever fitness function is in place at the time. However, the current fitness function does not distinguish between two rules that describe exactly the same subsets of the data, even if one relies on 10 attributes and the other relies on 20 attributes. Because of our concern that the rules be interpretable to humans, the 10-attribute rule should probably be considered better than the 20-attribute rule. However, this concern may interact with the issue of predictive accuracy; it is possible that the 10-attribute rule, which is more general than the 20-attribute rule, would be worse in the context of a prediction task.

APPENDIX A

DIFFERENT APPROACHES TO EVOLVING RULE SETS WITH THE GA

The typical GA formulation is designed to find a single solution (an optimization of the fitness function). In fact, it has been shown that even when two optima of equal fitness exist, a GA run will tend to converge on one or the other of these optima (see, for example, Goldberg and Segrest [43] or Smith et al. [99]). This phenomenon is called "genetic drift" (borrowing a term from natural genetics). Development of theory and mechanisms to encourage the GA to evolve a set of co-adapted rules, rather than a single optima is a current research topic in the GA community.

There are several approaches in the GA literature that describe methods of evolving a co-adapted population of strings, rather than the usual GA behavior of converging to a single solution. In this appendix, I'll present a representative sample of these approaches, labelled: "The Pitt Approach", "Iterative Runs", "Crowding", "Fitness Sharing I", "Fitness Sharing II", and "COGIN". Each of these approaches has advantages and disadvantages, which may make it suited to a particular type of application. In discussing each approach, I will discuss expected advantages and disadvantages of using the approach on the thesis work.

One thing to keep in mind with these various approaches is that several of them require a different fitness function; the new fitness function needs to evaluate the fitness of a *rule set*, rather than the fitness of an individual rule. This is but one approach to preventing the population from converging on a single rule; other approaches evaluate the fitness of the rule set implicitly, rather than explicitly.

The Pitt Approach

The GA formulation described in Section 3.3.1 is sometimes referred to as "the Michigan approach" because it corresponds to the formulation developed by John Holland and others at the University of Michigan [51]. In contrast to this is "the Pitt approach", more commonly used at the University of Pittsburgh (see, for example, DeJong [60]).

In the "Pitt" approach, each string in the population represents a rule set, rather than a single rule. In some implementations of this, the strings are of a fixed length (that is, there are a fixed number of rules in a rule set), and in other implementations, the strings are of variable length.

Advantage: Rule set evolves as a set (rather than as independent rules), so a fitness function can be explicitly applied to the rule set. (In most other approaches, the fitness function is applied to a single rule.)

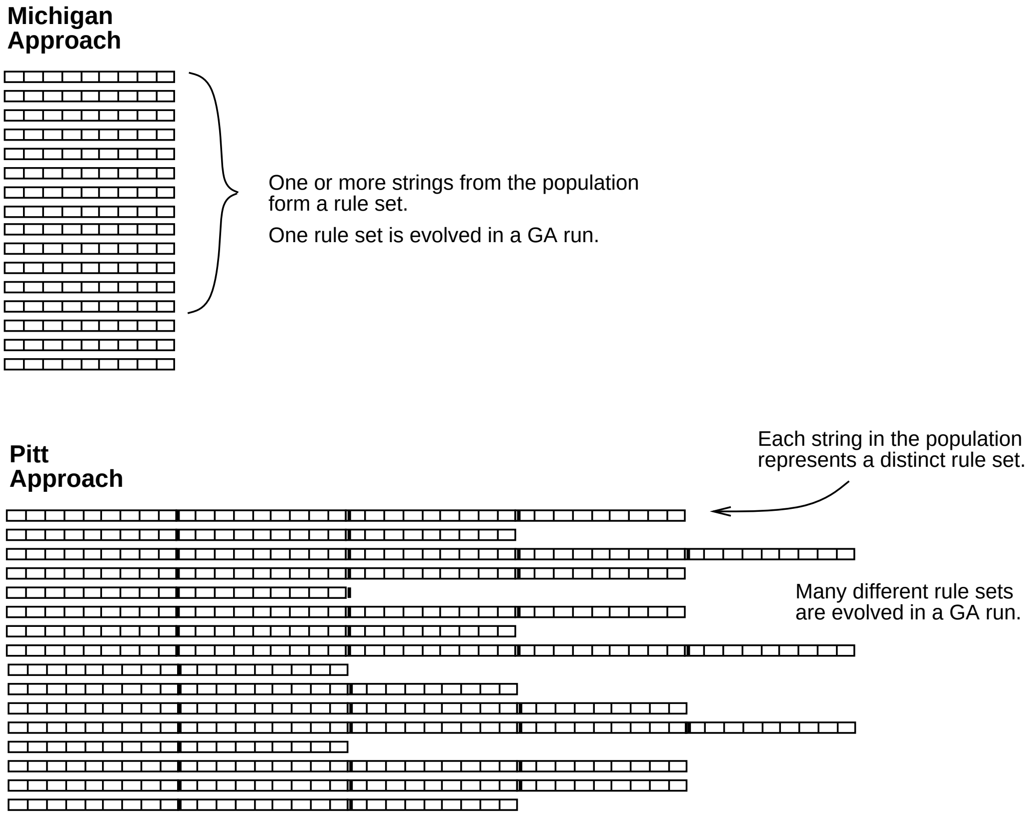

Michigan Approach
One or more strings from the population form a rule set.
One rule set is evolved in a GA run.
Pitt Approach
Each string in the population represents a distinct rule set.
Many different rule sets are evolved in a GA run.

Figure A.1: The Michigan approach vs. the Pitt approach

Disadvantage: Much longer evaluation times per string, because an entire rule set must be evaluated.

Disadvantage: Much longer evolution time, because the strings are so long.

The Pitt approach is illustrated in Figure A.1.

Iterative Runs

In the "Iterative Runs" approach, used by Sikora and Shaw [91], the GA is run multiple times. After each run, the best rule is saved, and the data points that match that string are removed from the dataset. The rule set is the collection of best rules from the series of runs.

Advantage: Solutions should not have a high degree of overlapping data points.

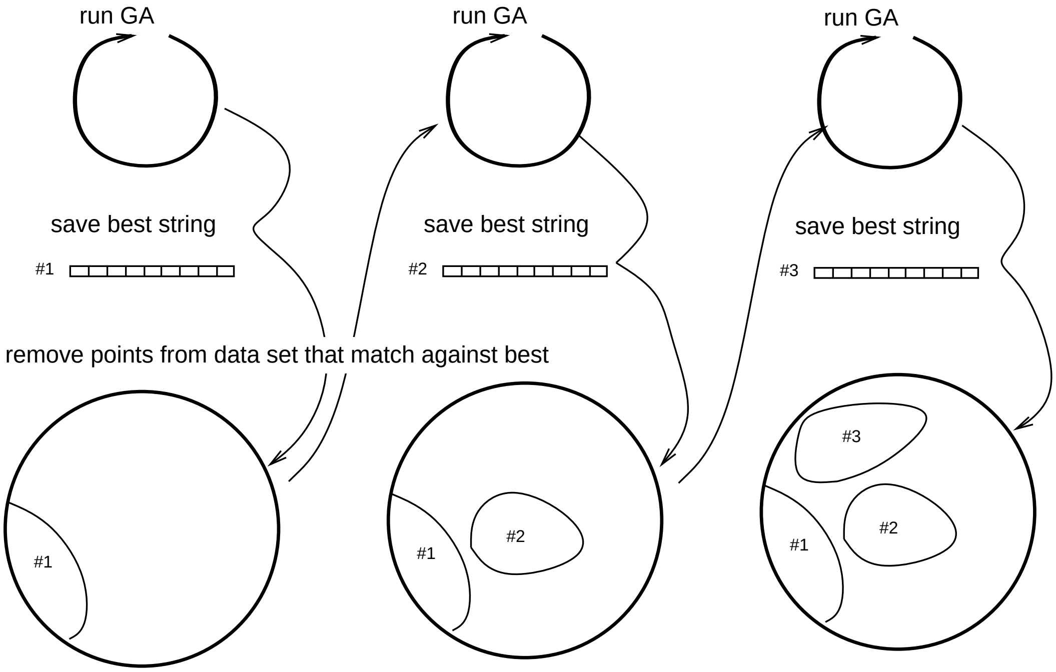


Figure A.2: Iterative runs

Advantage: The fitness of a string can be evaluated independently of the fitness of other strings. (Some approaches to evolving rule sets with the GA require that strings be compared to other strings.)

Disadvantage: Multiple runs are required, and the resulting "best" strings are not necessarily the best collective explanation of the data.

The "Iterative Runs" approach is illustrated in Figure A.2.

Crowding

"Crowding" is an approach explored by DeJong [60]. In Crowding, each new string is inserted into the population as soon as it is created (as discussed in Section 5.1, this is a Steady-State GA, as opposed to a Generational GA). For each new string to be added to the current population, a small subset (e.g. 3) of the current population is sampled; the new string replaces the string from the sample that it is most similar to.

Advantage: Strings replace strings that are genetically similar to themselves, possibly preserving greater diversity in the population.

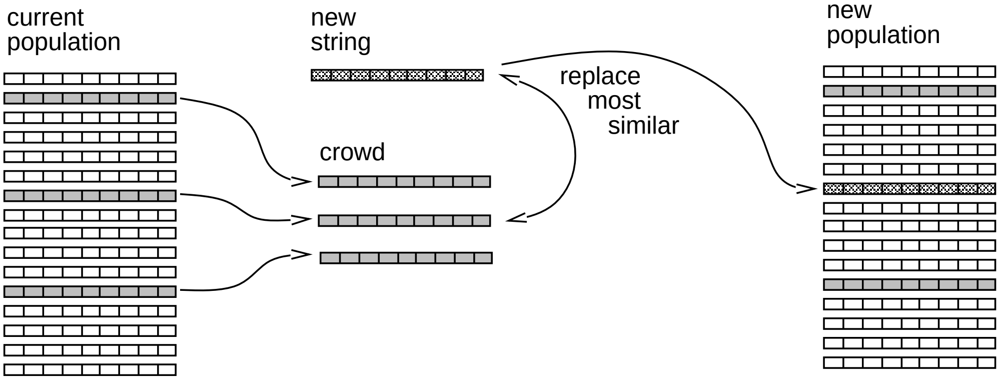


Figure A.3: Crowding

Advantage: The fitness of a string can be calculated without comparing it to all other strings in the population.

Disadvantage: The measure of similarity is syntactic, not semantic. That is, similarity is determined at the level of what the strings look like, rather than how they function.

The Crowding approach is illustrated in Figure A.3.

Fitness Sharing I

In the original formulation of fitness sharing by Goldberg and Richardson [42], strings are penalized for the presence of other similar strings in the population. (That is, their fitness is decremented.) This approach defines a neighborhood for each string (for example, a boundary of all points at the same Euclidean distance from the string). The fitness of each string is calculated using the fitness function, but then this value is degraded according to the number of strings that are in the same neighborhood. The amount to decrement the string's fitness is based on a parameter, σ_{share}, to determine the size of the neighborhood; this parameter is based on the number of expected peaks in the solution.

Advantage: This method induces stable niches.

Advantage: This method incorporates idea of limited resources into the GA model.

Disadvantage: The number of peaks must be known (or guessed) ahead of time; the peaks must be equidistant from one another.

Disadvantage: An enormous number of comparisons must be done every generation (popsize*popsize).

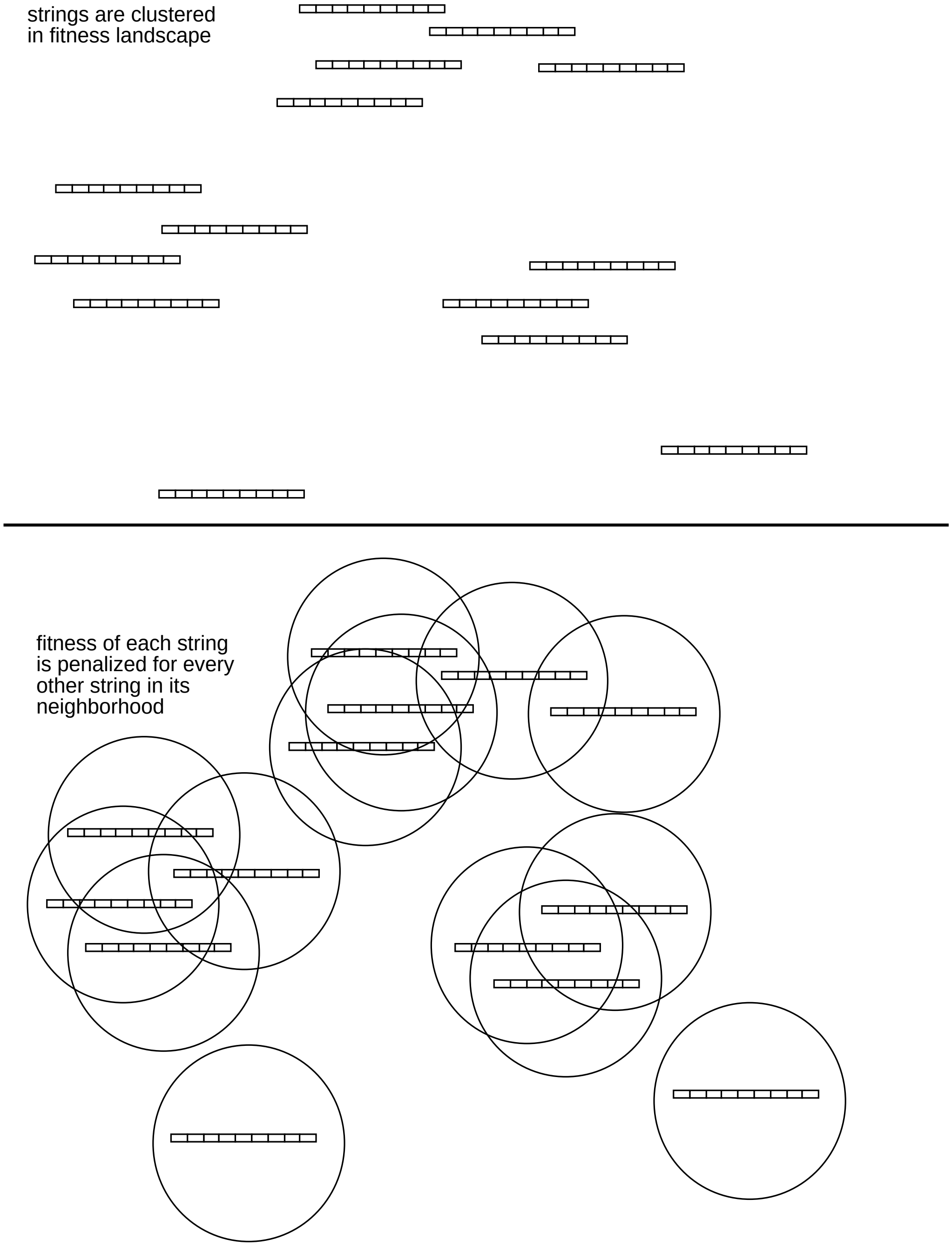


Figure A.4: Fitness sharing

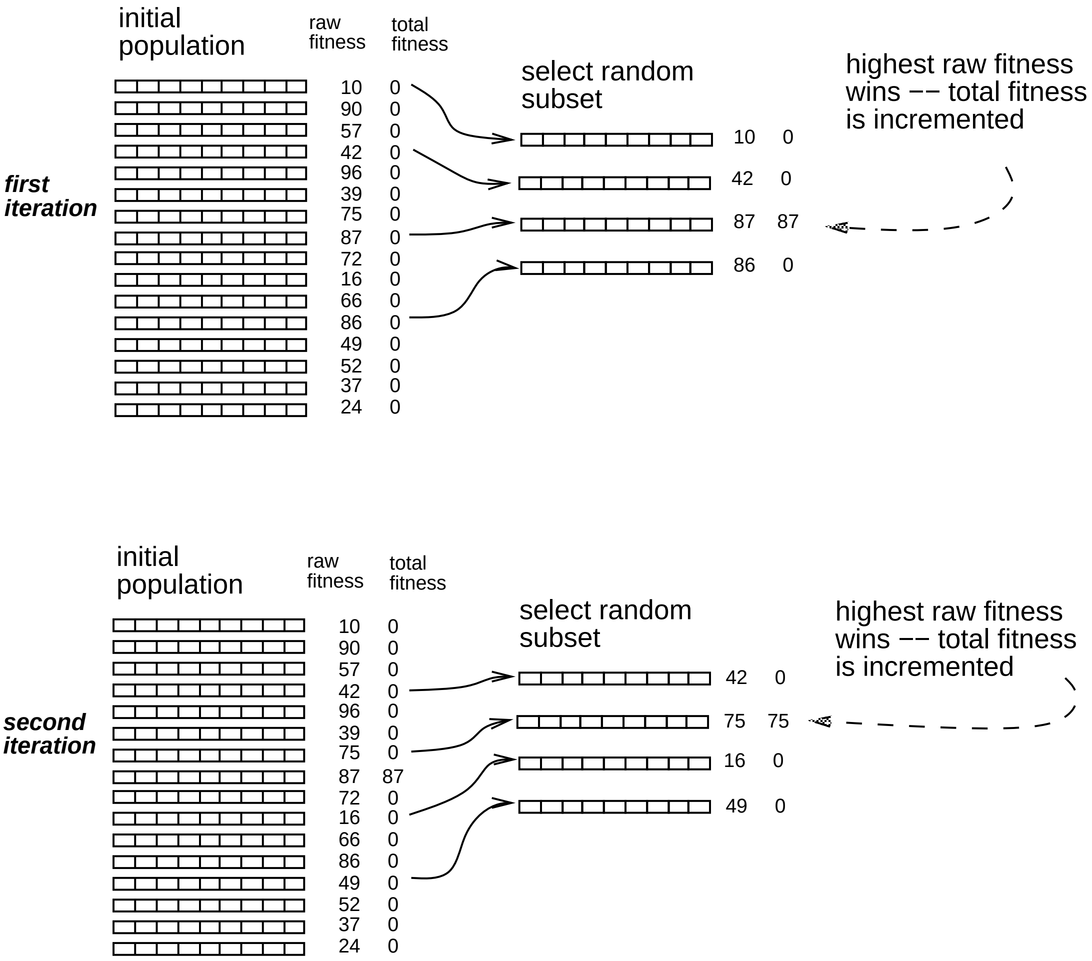


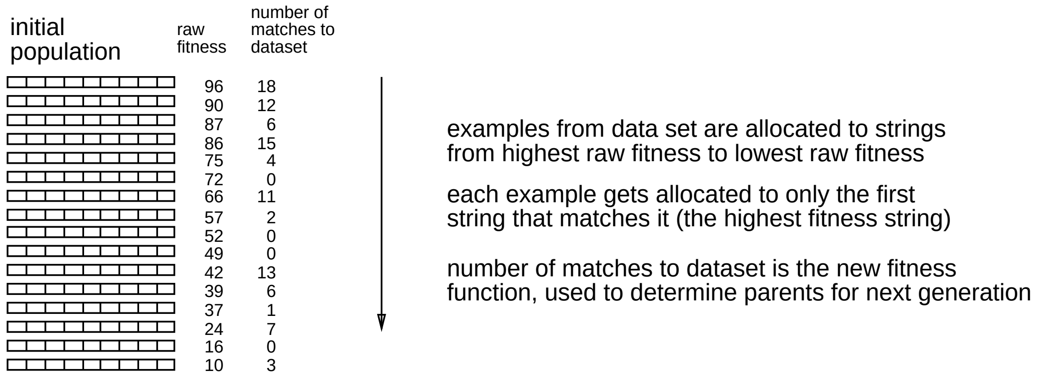
Figure A.5: A second approach to fitness sharing.

Although fitness sharing uses primarily a syntactic measure of string similarity, Deb and Goldberg [25] outline an approach for a semantic measure of similarity.

This approach to fitness sharing is illustrated in Figure A.4.

Fitness Sharing II

An alternative version of fitness sharing is described in by Smith, Forrest, and Perelson, in their project on the immune system [99]. In this approach, the calculation of fitness is an iterative process. First the "raw fitness" of each string is determined, using the fitness function. Then, a subset of the population is selected, and the raw fitnesses of these strings compared; the string with the highest raw fitness (in the subset) gets its "operative fitness" incremented. (Note that their are two fitness measures here.) This is repeated for C cycles,



raw fitness	number of matches to dataset
96	18
90	12
87	6
86	15
75	4
72	0
66	11
57	2
52	0
49	0
42	13
39	6
37	1
24	7
16	0
10	3

examples from data set are allocated to strings from highest raw fitness to lowest raw fitness

each example gets allocated to only the first string that matches it (the highest fitness string)

number of matches to dataset is the new fitness function, used to determine parents for next generation

Figure A.6: The COGIN approach

where C is a parameter set by the user. After the C cycles have completed, the operative fitness is used to determine parent selection and/or survivors to the next generation.

Advantage: The number of peaks does not need to be known ahead of time, nor do they need to be equidistant.

Advantage: There is not an excessive amount of computation required in the computation of fitness for a generation.

Disadvantage: Seems that the peaks must be the same height or they will not get picked up.

As used by Smith et al. this approach uses a syntactic measure of fitness, but they claim that it wouldn't have to be.

This approach to fitness sharing is illustrated in Figure A.5.

COGIN

Greene and Smith [44] developed another approach to encouraging niche formation. This method also uses two fitnesses for each string. The first, the "raw fitness" is calculated from the fitness function. The strings are ranked according to the raw fitness (from best to worst), and then the "operative fitness" is determined: the number of examples from the dataset matched by a string that have not already been explained by a higher fitness string.

Advantage: Allows a variable number of niches.

Disadvantage: Second fitness is the fitness function as far as the GA is concerned, and it results in a large number of 0-fitness strings, for our task.

The COGIN approach was used in the thesis and is illustrated in Figure A.6 and Figure 5.2.

APPENDIX B

ADDITIONAL GA RUNS

The different crossover and mutation rates investigated here represent a good range of the parameter space; however, since the GA runs tended to find better rule sets when using lower crossover rates and higher mutation rates, a brief experiment was done to see the effect of using even lower crossover rates and even higher mutation rates on this problem. An additional set of experiments were conducted, using crossover rates of .15, .20, and .25 and mutation rates of .2, .1 and .01. This is a total of nine different experiments, however, two of these experiments (crossover rate of .25 with mutation rates of .1 and .01) were already done at the beginning of Chapter 5. The nine runs are illustrated in Figure B.1. The highest number of positive examples for each of these four runs, and the average number of positive examples explained by a rule set in 100 runs, is shown in the following tables:

			mutation rate		
			.2	.1	.01
Best	crossover	.15	146	$\boxed{180}$	162
Rule Set	rate	.20	139	176	153
		.25	146	173	157

			mutation rate		
			.2	.1	.01
Average	crossover	.15	128.43	$\boxed{156.02}$	137.81
Rule Set	rate	.20	127.69	155.60	138.19
		.25	127.21	154.67	141.08

The results from a two-way analysis of variance indicate that the mutation rate has a statistically significant ($p < .0001$) effect on the average rule set found by the GA, but that the crossover rate and the interaction crossover by mutation rates have no significant effect.

However, when one looks at mutation rates of .1 and .01 only, one can do a larger analysis of variance. Using data from these experiments and from the experiments done at the beginning of Chapter 5, we can include 14 experiments (two mutation rates and seven crossover rates: .15, .20, .25, .50, .60, .75, .90) in two-way analysis of variance. The results from this analysis indicate that the crossover rate, the mutation rate, and the interaction between crossover and mutation rates each have a statistically significant ($p < .0001$) effect on the average rule set found by the GA.

The best parameter settings from this additional set of experiments is with .15 crossover and .1 mutation. Although I neglected to do my usual test of the GA's descriptive ability (using the full dataset) with this parameter setting, I did do a set of experiments to evaluate its

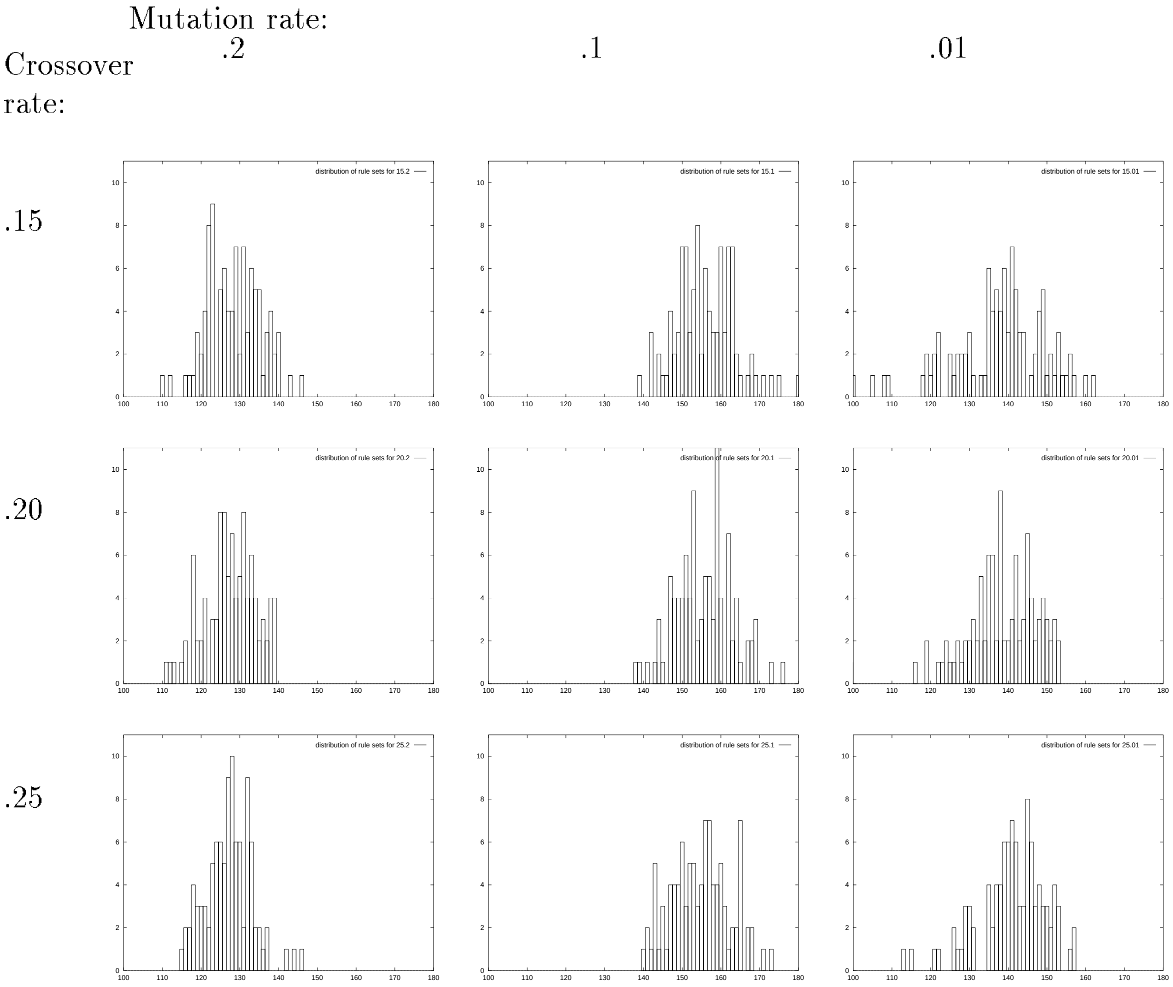


Figure B.1: Distributions of the best rule sets found in 100 runs, for nine additional experiments with different parameter settings (varying mutation and crossover rates).

predictive abilities:

		training set			testing set		
run	number of rules	number of examples	number correct	percent correct	number of examples	number correct	percent correct
1	53	458	421	91.9%	115	51	44.3%
2	55	458	401	87.6%	115	51	44.3%
3	55	458	416	90.8%	115	54	47.0%
4	50	458	396	86.5%	115	63	54.8%
5	57	458	425	92.8%	115	59	51.3%
(totals)	270	2290	2059	89.9%	575	278	48.3%

These results show that the GA with .15 crossover and .1 mutation performed significantly worse ($p < .05$) than the GA runs used in the main part of the thesis (which were done with .25 crossover and .1 mutation). In fact, a paired t-test comparing the .25 crossover GA to the .15 crossover GA on classifying the training data show that the .15 crossover GA did significantly worse ($p < .005$) than the .25 crossover GA on the five training runs.

The experiments in crossover versus mutation rates presented here and in Chapter 5 compared GA's that were limited to about 25 rules (since that was the number saved via the generation gap and elitism). Although the GA with .15 crossover was able to explain more examples with those 25 rules (180) than the GA with .25 crossover (which explained only 173 examples), the experiments with the training data indicate that the lower crossover rate may make it more difficult for the GA to discover rules that explain the remaining 74 positive examples.

APPENDIX C

THE GA WITH A “SIMPLE” STRING REPRESENTATION

Chapter 5 discussed the main experiments done with the GA, using what we’ve come to call the “complex” string representation, the conjunct of disjuncts format. However, we soon realized that although this representation has wide expressive power, it is not entirely consistent with our intent to develop a rule set that is easily interpretable by humans because people seem to have a difficult time internalizing the conjunct of disjunct form. Therefore, we also did a number of experiments using what we call the “simple” string representation, which allows at most one value to be specified per attribute in a rule. The results of these experiments are presented in this appendix.

C.1 The Representation

Using the simple representation, there are four values per attribute (each specified in the GA string using two bits). The genotypes are specified as two alleles (that is, each allele is considered a distinct attribute), each of which has three possible values (plus a don’t care). Smoking and gender, each of which have only two possible values (YES or NO and MALE or FEMALE), have two don’t care values. The other 16 attributes all have three possible values (HIGH, MEDIUM and LOW) and one don’t care. This is a total of 24 attributes, or 48 bits in the string. An example string and its translation are illustrated in Figure C.1.

The example string in Figure C.1 represents those people with APO ϵ43 genotype, an APO H genotype of H*23 or H*22 or H*21, high TRIG, medium HDL, and low APO AI. The other risk factors are all “don’t cares” (represented as ‘#’), which means that these risk factors are not considered when comparing a string to the data set.

The tables below provide translations of bits to risk-factor values. Note that the binary values are gray coded so that adjacent values differ by only one bit.

Binary	Integer	Allele
10	3	ϵ4, H*3, AIV*3)
11	2	ϵ3, H*2, AIV*2)
01	1	ϵ2, H*1, AIV*1)
00	0	“don’t care”

Binary	Integer	Trait
10	3	“don’t care”
11	2	higher tertile
01	1	middle tertile
00	0	lower tertile

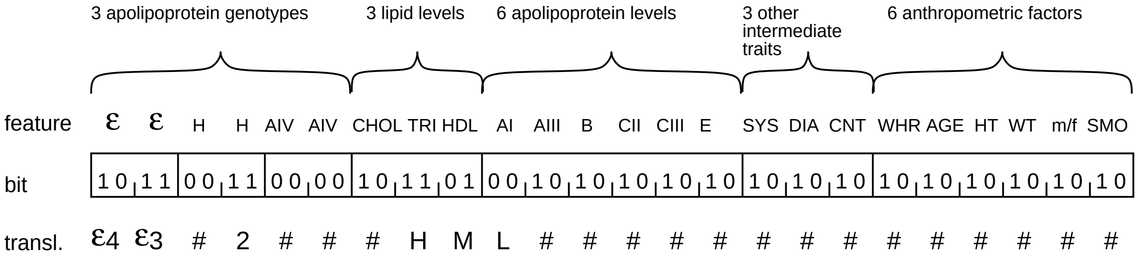

3 apolipoprotein genotypes
3 lipid levels
6 apolipoprotein levels
3 other intermediate traits
6 anthropometric factors
feature
ε ε H H AIV AIV CHOL TRI HDL AI AIII B CII CIII E SYS DIA CNT WHR AGE HT WT m/f SMO
bit
1 0 1 1 0 0 1 1 0 0 0 0 1 0 1 1 0 1 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
transl.
ε4 ε3 # 2 # # # H M L # # # # # # # # # # # # # #

Figure C.1: An example string using the simple representation, showing the representation of genotypes and other risk factors. The first part of the string represents the genotype of the three genes; the second part of the string represents three lipid traits; the third part of the string represents six apolipoprotein traits; the fourth part of the string represents three other intermediate-trait attributes; and the last part represents the six anthropometric trait attributes. The pound sign is used to denote a “don’t care” value — the attribute may take on any of its possible values.

Binary	Integer	Smoking Status	Gender
11	3	“don’t care”	“don’t care”
10	2	“don’t care”	“don’t care”
01	1	smoker	male
00	0	nonsmoker	female

Like the original experiments with the “complex” representation, the experiments with the simple representation used a mix of 50% don’t cares strings and 50% randomly generated strings as the initial population, and used the same fitness function, parent selection policy, and crossover and mutation rates. Also, like the initial experiments with the complex representation, a population size of 100 was used, with 100,000 total trials and a generation gap of 75%.

C.2 Variations to crossover and mutation rates

Ten experiments were done to investigate the effects of varying the crossover and mutation rates for the simple GA. As with the first experiments with the complex representation, crossover rates of .25, .50, .60, .75, and .90 were investigated; mutation rates were .1 and .01. The distributions of the 10 runs with the simple representation are shown in Figure C.2. The highest number of examples explained, and the average number of positive examples explained, is shown in the following tables:

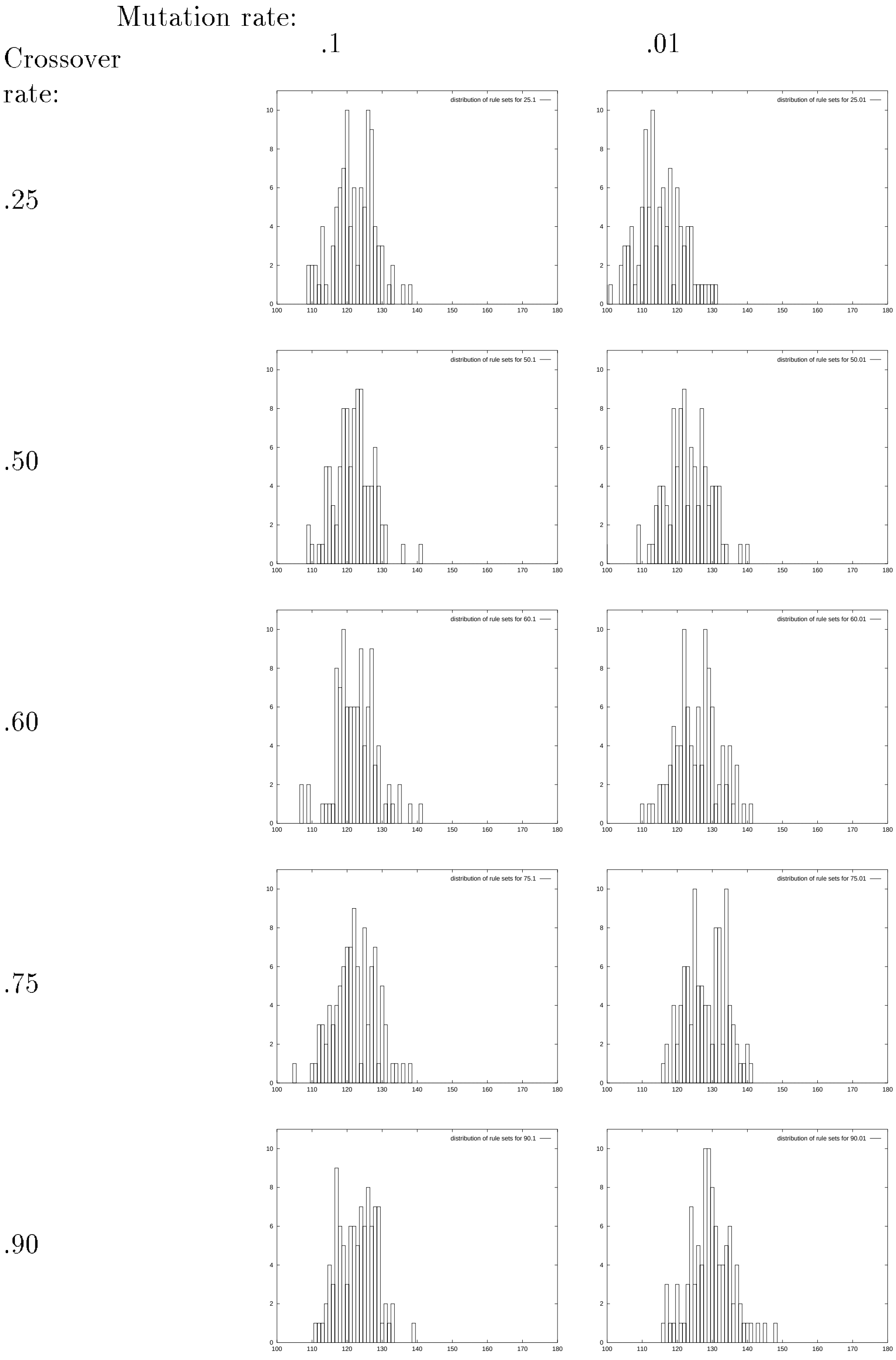


Figure C.2: Distributions of the best rule sets found in 100 runs, for each of 10 experiments with different parameter settings (varying mutation and crossover rates), using the simple string representation.

			mutation rate	
			.1	.01
Best	crossover	.25	138	131
Rule	rate	.50	141	140
Set		.60	141	141
		.75	138	141
		.90	139	$\boxed{148}$

			mutation rate	
			.1	.01
Average	crossover	.25	122.07	115.05
Rule	rate	.50	121.81	122.89
Set		.60	122.50	125.64
		.75	122.05	128.25
		.90	122.77	$\boxed{129.45}$

The results from a two-way analysis of variance indicate that the crossover rate, the mutation rate, and the interaction between crossover and mutation rates each have a statistically significant ($p < .0001$) effect on the average rule set found by the GA. These results favor choosing the parameter setting that has the best average performance to be used in future experiments. In this case, .90 crossover and .01 mutation are the best parameter settings, unlike the complex string representation, which favored low crossover and high mutation.

The .90 crossover rate and .01 mutation rate experiment was repeated, using the larger population size (150), number of trials (150,000), and generation gap (50%) that were used with the complex string representation to explain more of the dataset. The results of this are compared to the equivalent run using the complex representation in the table below; the five best rules are illustrated in Figure C.3, which can be compared against Figure 8.1 from Chapter 8.

GA	number					average	average
rep.	of rules	a	d	fitness	accuracy	ex./rule	pos. ex./rule
simple	64	220	319	0.83	94.1%	8.95	3.97
complex	60	247	319	0.89	98.8%	9.55	4.12

Note that although the descriptive accuracy is slightly lower using the simple representation, the rules found are considerably easier to understand. The distribution of rules that explain a given number of examples is compared in the table below; from this we can see that the rule set found using the simple representation avoided rules that explained 10 or more people, but also avoided rules that explained fewer than 3 people.

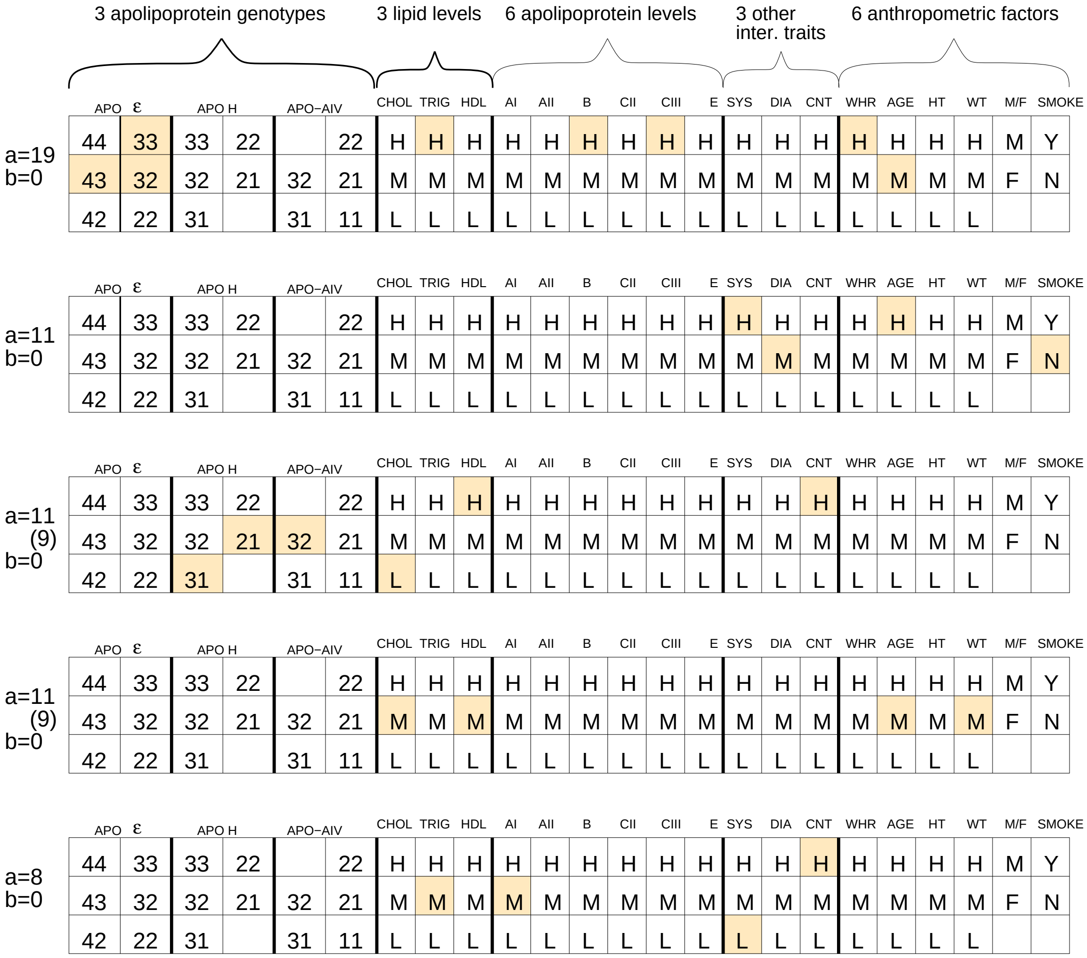


Figure C.3: An illustration of the five best rules for the GA, when run with the simple representation. The grid represents all the possible attribute-values that might be specified in a rule; the shaded areas indicate the rule itself.

GA rep.	total rules	number of rules that explain n people 1	2	3	4	5-9	10+
simple	64	0	0	2	17	42	3
complex	60	1	0	0	3	40	16

The prediction task

The prediction accuracy task for the simple representation is summarized in the table below. A population size of 150 was used, with 150,000 total trials, and a generation gap of 50%.

w = 1.0		training set			testing set		
	number of	number of	number	percent	number of	number	percent
run	rules	examples	correct	correct	examples	correct	correct
1	56	458	445	97%	115	59	51%
2	66	458	450	98%	115	61	53%
3	66	458	450	98%	115	61	53%
4	61	458	451	98%	115	61	53%
5	69	458	452	99%	115	51	44%
(totals)	318	2290	2248	98.2%	575	293	51.0%

The table below compares the GA using the simple representation to the GA using the complex representation on the prediction task; using the simple representation, the GA performed marginally worse, however, using a paired t-test, this difference is not significant. Paired t-tests also show that the GA with simple attributes did not perform significantly differently than the regular GA when positive and negative examples are considered separately.

GA	averages across five testing/training runs					
	number			number	prediction	prediction
rep.	of rules	*a*	*d*	correct	fitness	accuracy
simple	64	25	34	59	0.505	51.0%
complex	54	25	35	60	0.508	52.2%

The 95% confidence intervals for the prediction task with all of the different GA runs are shown in Figure C.4.

C.3 The Effects of Varying w

The next set of experiments looked at the effects of varying w in the weighting function from Chapter 4. In this set of experiments, parameter settings are fixed with a crossover rate of .90 and a mutation rate of .01. Five different values of w were explored: $w = 1.0$, $w = 0.75$, $w = 0.50$, $w = 0.25$, and $w = 0.0$. (Note that $w = 1.0$ corresponds to an experiment that was performed in the previous section of this appendix.) For these experiments, a population size of 150 was used, with 150,000 total trials and a generation gap of 50%, since these were the parameters found to be necessary to find a full rule set with the complex representation.

The best rule set found for each of the values of w is shown in the following table:

	w	fitness	correctness	simplicity	a	b	rules
Population	1.00	.8276	.8276	.6166	220	0	64
Size	0.75	.6816	.6162	.8780	253	287	30
150	0.50	.7862	.6040	.9685	253	295	7
Simple	0.25	.8609	.4910	.9843	254	319	3
Rep.	0.00	1.000	.4910	1.000	254	319	1

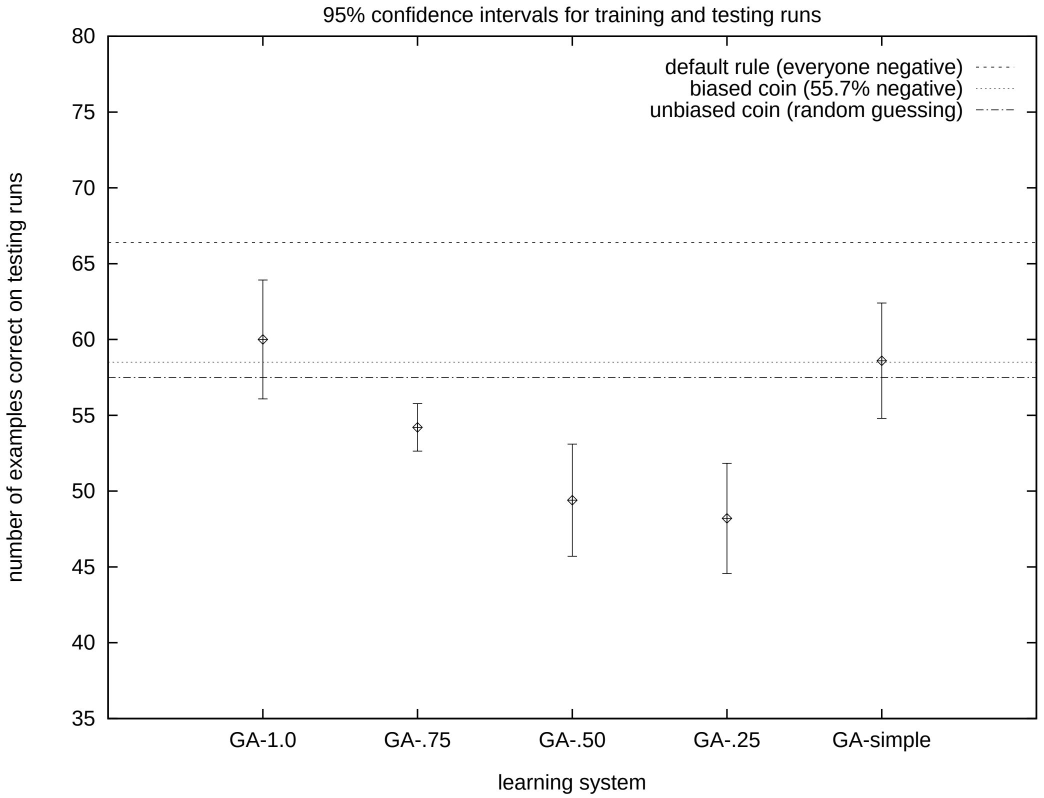


Figure C.4: 95% confidence intervals for the prediction task are shown for all GA runs, including the “simple” representation.

The most notable difference in the results above, as compared with the weighted runs using the complex representation, is the $w = .75$ run. Using the complex representation, the GA found a rule set with $a = 220$ and $b = 0$, using 36 rules. The fitness of that rule set was .8058, as compared with the .6816 fitness of the rule set found with $w = .75$ using the simple representation. The other runs are roughly comparable to the runs found using the complex representation.

The ROC curves from these rule sets are shown in Figure C.5. Figure C.6 shows the varation of the ROC graph, which plots *sensitivity* on the vertical axis, and the number of rules on the horizontal axis. By looking at the two graphs together, we are able to get a better feeling for the differences in the rule sets obtained using different values of w.

The five best rules for the GA runs with the simple representation and the weighting function are illustrated in Figures C.7–C.9 (when $w = .25$, there are only four rules in the entire rule set). Note that the best rules found by the $w = .75$ runs and the $w = .50$ runs matches the same 19 examples from the dataset. Also, the best rule from the $w = .25$ runs matches the same examples as the fourth rule from the $w = .50$ runs.

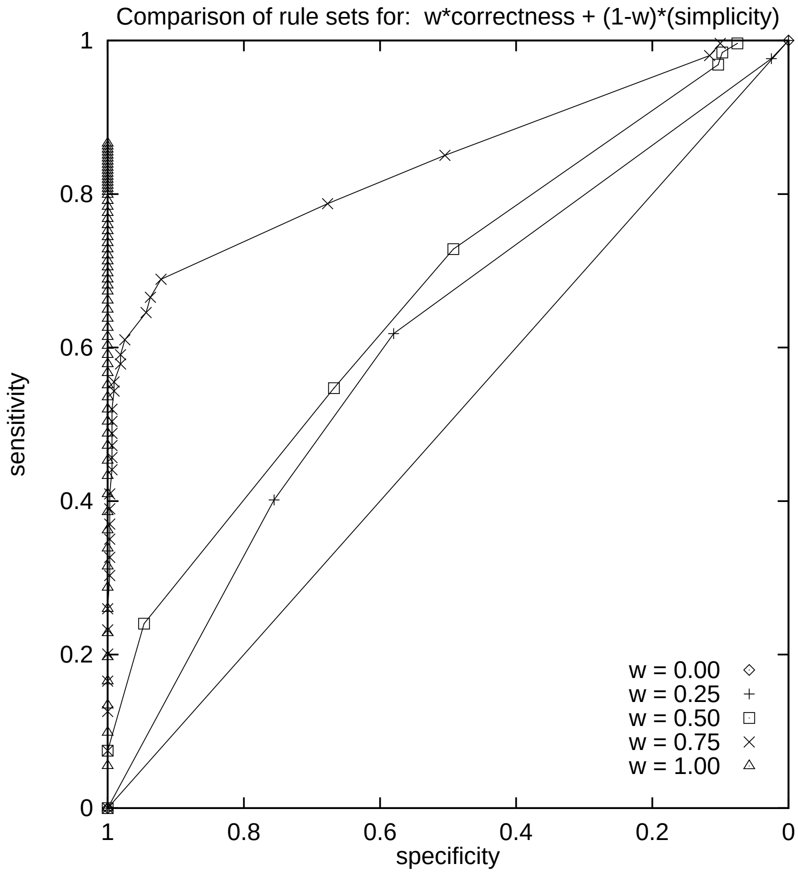


Figure C.5: Graph of *sensitivity* vs. *specificity* in describing the entire dataset for each of the values of w, using the simple representation with the GA.

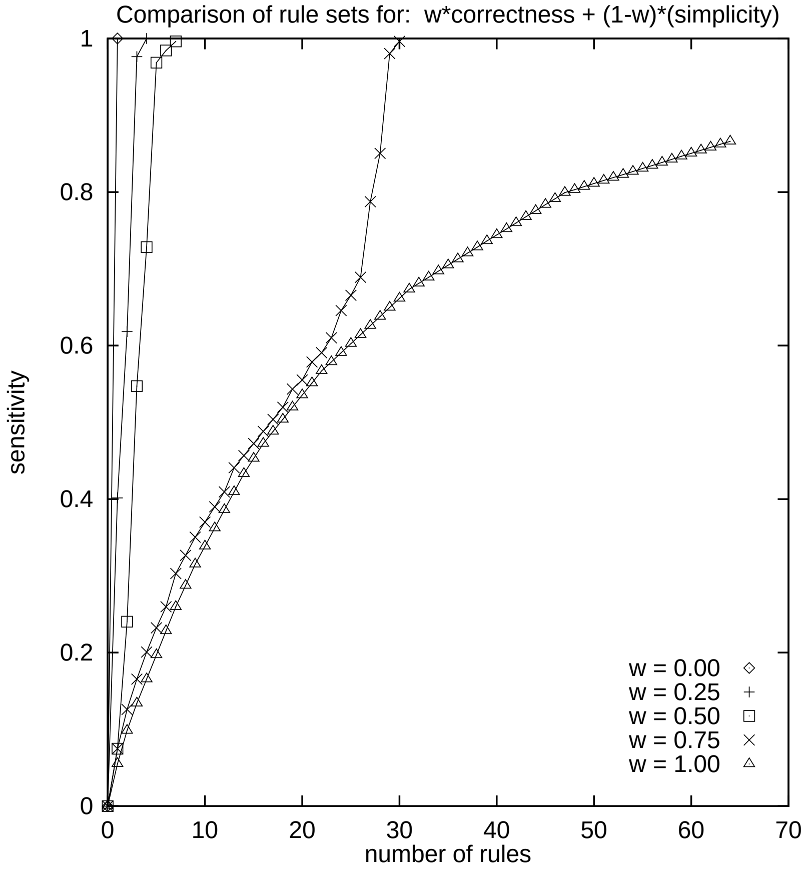


Figure C.6: Graph of *sensitivity* vs. number of rules in describing the entire dataset for each of the values of w, using the simple representation with the GA.

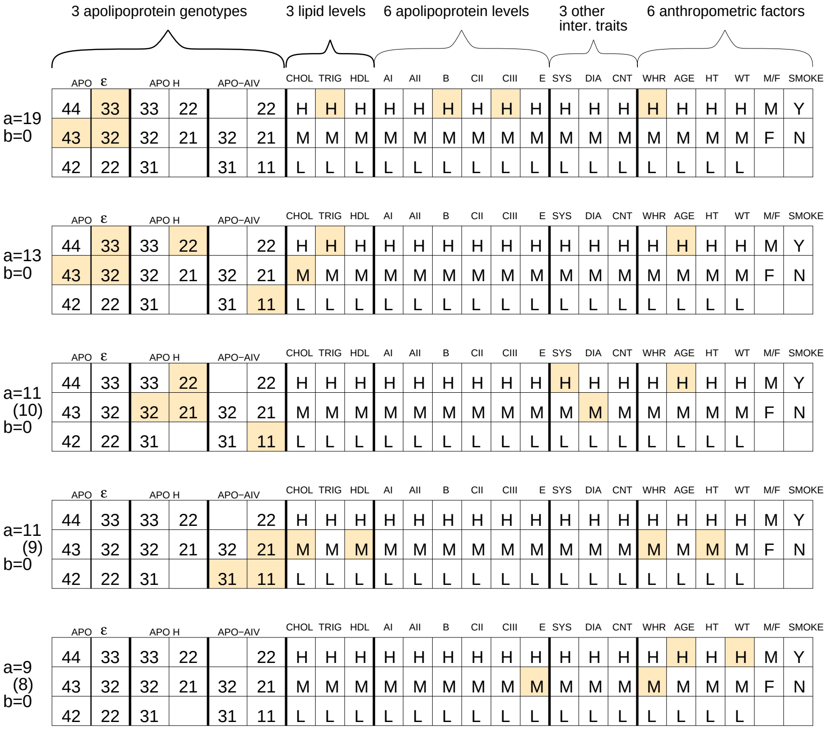


Figure C.7: An illustration of the five best rules for the GA when run with the simple representation and with $w = .75$ in the fitness function.

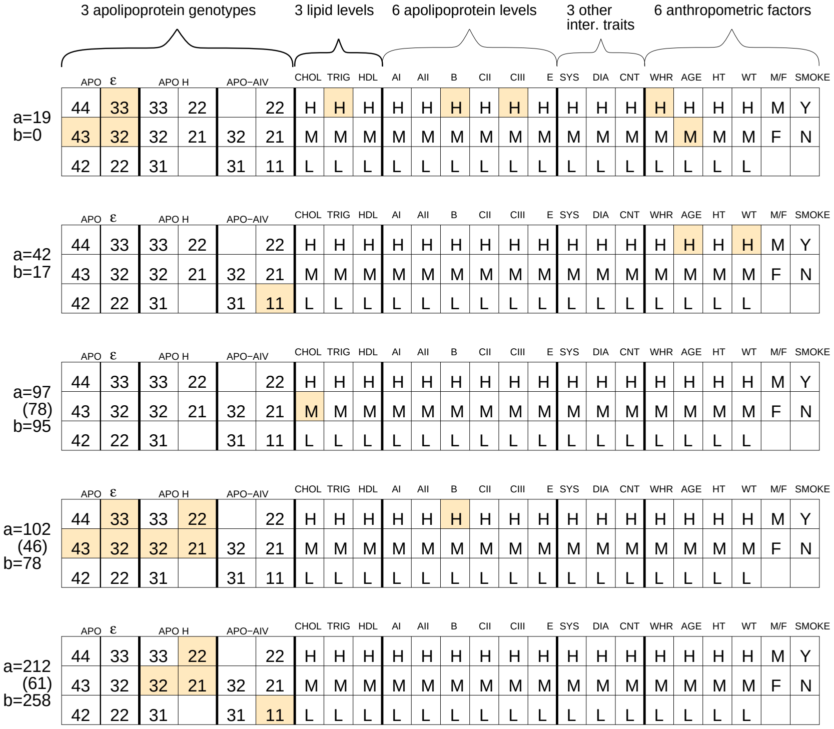


Figure C.8: An illustration of the five best rules for the GA when run with the simple representation and with $w = .50$ in the fitness function.

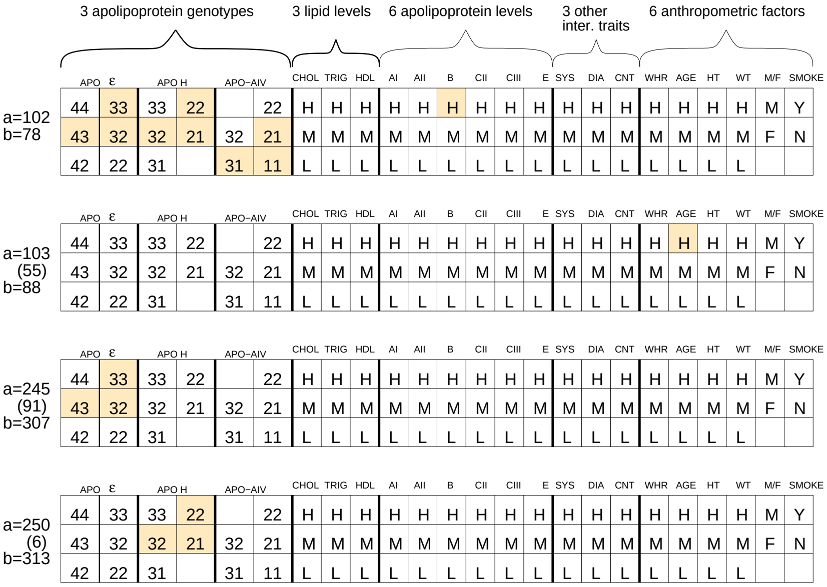


Figure C.9: An illustration of the rule set found by the GA when run with the simple representation and with $w = .25$ in the fitness function. These four rules represent an entire rule set for this weight.

APPENDIX D

GID3* EXPERIMENTS WITH REAL NUMBERS

In order to investigate how we may have influenced the descriptive and predictive nature of the task by discretizing the attributes ahead of time, we also did some brief experiments running GID3* with the continuous attributes, for those attributes that this information was available. That is, the three genotypes, gender, and smoking status, are naturally discrete attributes, and were not transformed. The results from this are reported in this appendix.

The table below summarizes the performance of GID3* with continuous data on the descriptive task; the results from GID3* when run with discretized data are also listed, as a comparison.

GID3* dataset	number of rules	*a*	*d*	fitness	accuracy	average ex./rule	average pos. ex./rule
contin.	127	254	319	1.0	100%	4.51	4.10
discrete	183	254	319	1.0	100%	3.13	2.73

The next table summarizes the performance of GID3* on the prediction task:

GID3*	averages across five testing/training runs					
	number			number	prediction	prediction
dataset	of rules	*a*	*d*	correct	fitness	accuracy
contin.	104	23	34	57	0.497	49.4%
discrete	159	20	39	59	0.500	51.7%

Finally, the last table presents the results from five testing and training runs, using the continuous-valued data:

continuous		training set			testing set		
	number of	number of	number	percent	number of	number	percent
run	rules	examples	correct	correct	examples	correct	correct
1	98	458	458	100%	115	59	51%
2	89	458	458	100%	115	56	49%
3	117	458	458	100%	115	64	56%
4	106	458	458	100%	115	49	43%
5	108	458	458	100%	115	56	49%
(totals)	518	2290	2290	100%	575	284	49.4%

From the above tables, we note that GID3* was better on the description task when provided with the continuous-valued attributes in that it found fewer rules to describe the data

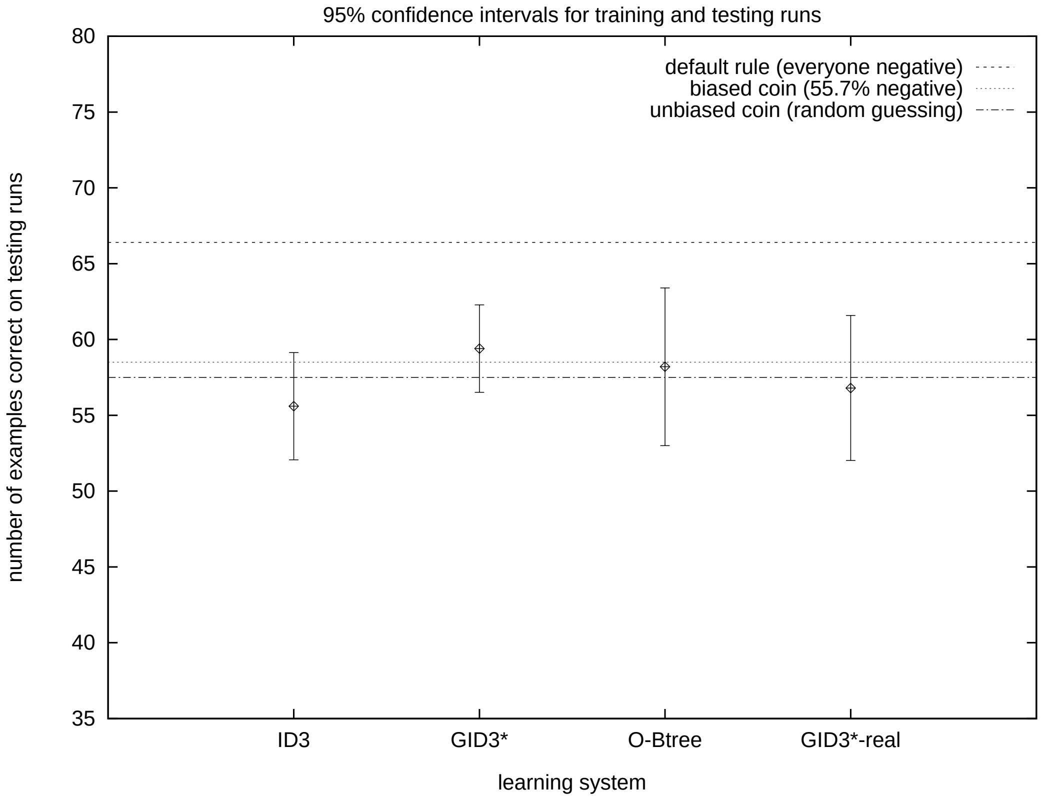


Figure D.1: 95% confidence intervals for the prediction task are shown for all decision-tree runs, including the runs using real-valued data.

than it found with the discretized attributes. It fared slightly worse in terms of prediction task, achieving both a lower prediction accuracy and a lower prediction fitness. However, using a paired t-test, the difference in prediction accuracy across the five runs is not significant. When positive examples and negative examples are considered separately, however, the GA with discretized attributes performed significantly better ($p < .05$) than the GA with continuous attributes when classifiying negative examples.

The 95% confidence intervals for the prediction task with all of the different decision-tree runs are shown in Figure D.1.

It is worth noting, perhaps, that the smaller number of rules found by using the continuous-valued dataset is not without a "cost" in terms of the simplicity of the rules found. The five best rules found by GID3* using the continuous-valued dataset are shown in Figure D.2. (Note that the gridding in the figure does not correspond exactly to that used with discrete attributes. The division of the data into tertiles places approximately one third of the dataset into each of the high, medium, and low categories, whereas the division here is based on the range of values seen in the data, independent of number of examples in each subdivision.) These rules illustrate two curious things about the rule set. First, GID3* made no use of the

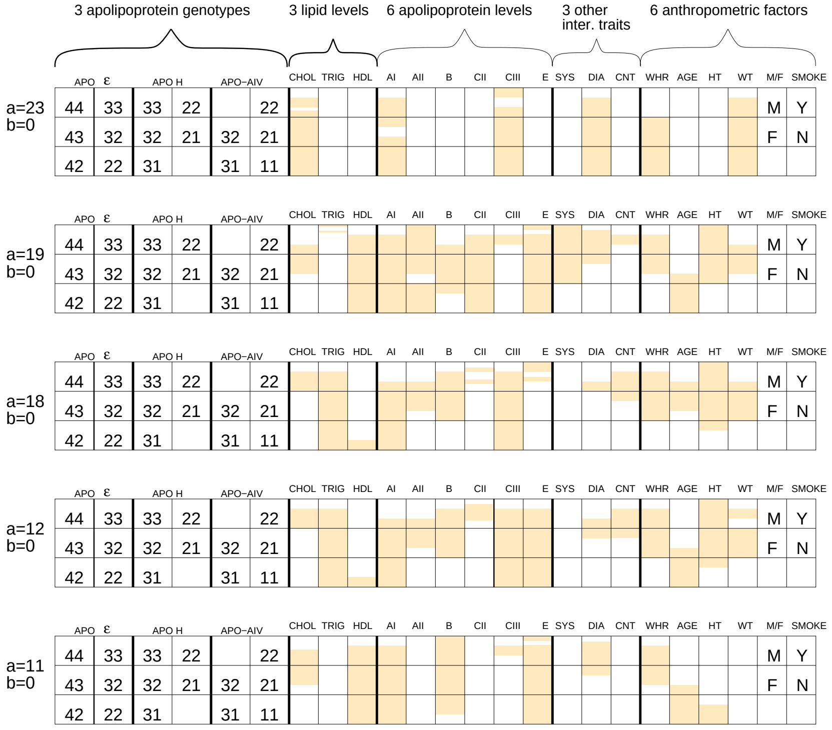


Figure D.2: An illustration of the five best rules for GID3*, when run with the continuous-valued dataset. The grids are left as they were for the discretized versions; the shaded area approximates the actual values specified in the rules.

discrete-valued attributes in any of the 127 rules in the rule set. Second, GID3* demonstrated a tendency to carve out small subranges of attributes to exclude a single individual. For example, in the best rule, the cholesterol attribute has excluded the range from 160 to 165.5 to exclude two individuals with a cholesterol value in that range who did not have a family history of disease. It may be that this tendency to carve out small subranges is the reason that GID3* avoided the discrete attributes, which do not allow for such fine subdivisions.

Like the discrete-valued rule set, the continuous-valued rule set contained a large number of rules that explained a small number of examples (although the continuous-valued rule set was better in terms of the number of examples explained by the average rule):

GID3*	total	number of rules that explain n people					
dataset	rules	1	2	3	4	5-10	10+
contin.	127	34	31	13	15	19	15
discrete	183	70	35	22	14	36	6

The continuous-valued dataset produced fewer rules than the discrete-valued dataset. In fact, the number of rules required to explain the positive examples is comparable to the number of rules found by the GA: GID3* with real numbers found 62 rules to explain the positive examples, while the GA with discrete attributes found 60 rules to explain the positive examples. However, this hardly constitutes a ringing endorsement of GID3* with real numbers. The rule set found by GID3* with real numbers is quirky in that it avoids the discrete-valued attributes, is more complicated for humans to understand (than the GA run with discrete values) due to its use of small subdivisions in the values of attributes, and is less accurate on the prediction task (than either GID3* or the GA run with discrete attributes).

APPENDIX E

THE RULE SET FOUND BY THE GA

The rule set found by GA is included here for those who have the interest and patience to look at it. These rules are presented in nearly the format used by the GA, that is, as bit strings. Translations from bits to values is provided in Table 5.1 in Chapter 5; recall that for each attribute there are two ways to specify "don't care" strings. Each string is followed by three numbers: the number of positive examples it explains (not explained by a higher fitness string); the number of negative examples it explains (always zero); and the number of positive examples it explains that has already been explained by a higher fitness string.

The order of the attributes in the strings does not correspond exactly to that given in the illustrations in the thesis, although it is nearly the same. In the strings presented here, the order is APO ϵ, APO H, APO AIV, CHOL, TRIG, HDL, AI, AII, B, CII, CIII, E, SYS, DIA, WHR, M/F, SMOKE, CNT, AGE, HT, WT. The interested reader may want to compare the five best rules with the illustration in Figure 8.1 to better understand the translations.

```
  Genes    Lipids   Apo's        other traits          a   b  a'
 0  5 39   0 4 0   3 7 4 7 4 6   5 0 5 3 0 4   7 7 0    19   0 (0)
62 47 47   2 0 0   7 3 0 7 5 7   7 3 5 0 1 7   3 0 7    17   0 (1)
51 52 15   3 7 0   7 3 7 7 6 6   0 6 0 3 1 6   7 7 0    12   0 (1)
30 38 13   7 7 0   2 4 7 3 7 0   7 5 5 3 1 7   6 7 7    12   0 (0)
 0 61 51   0 5 0   0 6 4 7 6 6   6 0 4 0 3 5   6 6 0    10   0 (4)
30 20 61   1 1 7   7 0 2 1 7 3   6 7 7 3 1 0   3 0 7     9   0 (0)
62 63 45   6 7 7   5 7 5 5 6 6   7 0 0 1 0 3   7 0 3     9   0 (0)
21 55 59   2 4 7   5 7 6 7 0 7   7 7 7 3 3 6   5 0 0     8   0 (5)
 6 54 25   6 7 6   7 7 4 7 3 7   7 0 0 0 0 7   5 5 0     8   0 (2)
 0 54 63   7 7 7   3 3 6 7 7 7   1 1 6 0 0 6   7 3 0     8   0 (1)
27 38 33   7 6 0   7 0 7 7 6 6   6 6 0 0 3 6   6 0 4     7   0 (6)
 7 55  5   7 0 7   7 7 7 6 2 7   7 0 5 1 0 5   7 1 0     7   0 (1)
22 38 33   7 6 3   7 0 5 3 7 6   3 7 7 2 0 7   0 0 7     6   0 (7)
12 44 17   0 0 0   0 1 0 0 5 0   3 3 0 3 0 0   0 4 0     6   0 (1)
15 62 27   6 5 7   4 6 2 3 0 7   5 0 3 3 1 3   7 0 7     6   0 (1)
20 17 45   5 5 6   6 0 0 3 7 0   7 0 5 3 3 0   7 3 0     6   0 (1)
 0 37 55   7 0 6   5 4 7 3 7 4   6 7 6 3 0 0   0 3 0     6   0 (0)
62 62  7   3 7 7   1 3 6 7 2 0   7 1 0 0 0 0   0 7 0     5   0 (2)
 5 60 37   7 7 5   0 7 3 7 0 7   5 0 1 3 1 3   7 2 1     5   0 (1)
45 53  7   2 7 6   6 5 7 7 7 7   7 0 2 3 1 3   3 0 7     5   0 (1)
47 63 41   7 7 3   7 6 5 6 5 7   5 6 5 2 0 5   7 0 6     4   0 (8)
44 53 57   7 7 4   5 6 6 7 0 7   7 7 7 3 1 6   5 0 7     4   0 (4)
```

```
63  6 41   3 7 7   2 3 7 7 0 6   0 6 6 2 0 6   7 0 0      4   0 (4)
 0  0  0   1 7 1   3 4 7 3 3 7   6 0 7 3 0 0   5 0 0      4   0 (3)
 0  0  0   0 0 3   4 0 0 0 0 1   6 0 1 1 0 0   5 0 0      4   0 (0)
30 54 15   7 7 0   3 4 7 3 7 0   7 5 5 3 0 7   6 7 3      3   0 (9)
31 62 41   4 6 5   7 0 7 7 7 7   6 6 0 3 3 4   6 7 4      3   0 (9)
46 45 15   6 0 0   3 6 5 0 7 0   3 7 4 3 1 7   6 7 7      3   0 (6)
 0 63 31   6 7 7   6 7 5 7 6 7   7 0 7 1 0 0   6 2 6      3   0 (5)
27  6 41   0 6 0   7 0 7 7 7 6   2 6 0 2 3 6   7 0 5      3   0 (4)
54 22 51   0 4 0   7 7 4 7 4 6   7 0 4 2 0 0   2 7 0      2   0 (13)
21 55 59   2 5 3   5 7 7 6 0 6   4 6 7 3 0 6   0 7 0      2   0 (8)
31 46  7   7 7 0   3 6 6 5 7 0   7 7 4 3 3 7   6 5 5      2   0 (7)
55 58 57   7 7 6   6 1 7 7 7 7   7 0 0 3 0 0   5 7 0      2   0 (7)
21 31 59   2 5 7   5 5 6 7 0 6   6 7 7 3 3 0   5 7 0      2   0 (6)
12 63 11   7 0 6   7 7 3 0 4 6   7 6 6 3 0 7   5 5 0      2   0 (5)
22 38 39   7 7 3   6 0 5 3 7 6   2 7 7 2 0 7   0 0 7      2   0 (5)
62  7 47   2 2 0   2 3 7 7 0 6   0 7 3 0 1 7   3 0 7      2   0 (5)
 0  5 39   7 5 3   1 6 6 7 7 6   7 0 6 3 1 0   7 2 0      2   0 (5)
 0 37 55   7 2 0   5 4 7 3 7 0   6 5 7 3 0 0   0 3 0      2   0 (3)
 7 37  5   7 0 7   7 7 7 6 3 7   7 0 6 1 0 5   5 1 0      2   0 (2)
19 52 47   3 6 1   7 3 7 7 6 6   0 6 0 3 0 6   7 7 0      1   0 (11)
23 45 43   0 4 0   7 7 4 7 6 6   5 0 4 0 2 6   7 6 0      1   0 (10)
63 63 41   7 7 3   7 6 5 6 5 7   5 6 5 3 0 5   7 4 6      1   0 (10)
 6 63 25   7 7 3   0 0 6 0 0 0   5 2 0 0 0 4   7 0 6      1   0 (8)
47 23 57   7 7 4   7 7 7 6 6 3   7 0 3 1 1 0   5 0 7      1   0 (8)
 0 55 13   7 6 7   3 3 7 7 7 7   5 1 6 0 0 6   7 3 0      1   0 (8)
54 38 33   3 6 3   7 0 5 3 7 6   3 6 0 3 1 5   0 0 7      1   0 (7)
55 58 57   7 7 7   6 1 7 7 7 7   7 0 0 3 3 6   5 3 0      1   0 (7)
 0 63 63   6 7 5   3 7 5 7 6 7   7 0 7 1 0 0   6 2 7      1   0 (6)
62 55 11   6 7 6   4 3 0 3 5 3   3 7 7 3 1 3   7 0 7      1   0 (6)
46 14 63   2 2 0   7 3 0 3 5 7   7 3 7 1 1 7   7 0 7      1   0 (6)
15 23  7   3 7 7   3 1 4 0 7 6   5 0 5 0 0 4   6 7 0      1   0 (5)
54 62  7   3 7 5   3 3 6 7 2 0   1 1 0 0 0 0   0 7 0      1   0 (5)
31 46  9   6 6 5   5 0 7 7 7 3   6 7 0 3 3 4   6 7 5      1   0 (5)
46 62 47   6 6 6   5 7 7 5 6 6   7 0 0 0 0 3   3 1 7      1   0 (4)
21  4 19   0 0 0   0 1 0 0 7 0   3 3 0 2 0 0   2 4 0      1   0 (4)
50 52 47   7 3 0   7 3 7 5 6 6   0 7 0 3 1 6   5 7 0      1   0 (4)
 4 22 55   2 7 6   5 7 7 7 6 5   7 0 0 0 0 2   5 0 0      1   0 (3)
54 38 13   7 5 0   3 4 5 3 7 1   3 5 0 3 1 5   0 0 3      1   0 (0)
```

APPENDIX F

THE RULE SET FOUND BY GID3*

The rule set found by GID3* is included here for those who have the interest and patience to look at it. These rules are presented in nearly the format output by TreeMaker. Specifically, the rules are in the order they would be read off the decision tree, and not separated into subsets for the rules that explain positive and negative examples. Each rule is described by a list of attribute-value pairs, the last of which is the class attribute, and followed by the number of examples explained by that rule.

Although the tree itself is too branchy to be illustrated coherently, the first three levels of this tree are illustrated in Figure F.1.

Rule 1 (HT not 1) (APOAIV 31) (AGE 0) (CLASS +) EXAMPLES: 3

Rule 2 (HT 1) (APOAIV 31) (AGE 0) (CLASS -) EXAMPLES: 1

Rule 3 (HT not 2) (APOE 32) (DIA not 1) (CII not 1) (AII 2) (APOAIV 11) (AGE 0) (CLASS +) EXAMPLES: 2

Rule 4 (HT 2) (APOE 32) (DIA not 1) (CII not 1) (AII 2) (APOAIV 11) (AGE 0) (CLASS -) EXAMPLES: 1

Rule 5 (HT not 0) (WT 0) (AI 1) (E not 2) (APOE not 32) (DIA not 1) (CII not 1) (AII 2) (APOAIV 11) (AGE 0) (CLASS -) EXAMPLES: 1

Rule 6 (HT 0) (WT 0) (AI 1) (E not 2) (APOE not 32) (DIA not 1) (CII not 1) (AII 2) (APOAIV 11) (AGE 0) (CLASS +) EXAMPLES: 1

Rule 7 (WT not 0) (AI 1) (E not 2) (APOE not 32) (DIA not 1) (CII not 1) (AII 2) (APOAIV 11) (AGE 0) (CLASS -) EXAMPLES: 4

Rule 8 (APOE not 42) (APOE not 32) (WHR not 2) (SMOKE not 1) (AI 2) (E not 2) (DIA not 1) (CII not 1) (AII 2) (APOAIV 11) (AGE 0) (CLASS -) EXAMPLES: 5

Rule 9 (APOE 42) (WHR not 2) (SMOKE not 1) (AI 2) (E not 2) (DIA not 1) (CII not 1) (AII 2) (APOAIV 11) (AGE 0) (CLASS +) EXAMPLES: 1

Rule 10 (WHR 2) (SMOKE not 1) (AI 2) (E not 2) (APOE not 32) (DIA not 1) (CII not 1) (AII 2) (APOAIV 11) (AGE 0) (CLASS +) EXAMPLES: 1

Rule 11 (SMOKE 1) (AI 2) (E not 2) (APOE not 32) (DIA not 1) (CII not 1) (AII 2) (APOAIV 11) (AGE 0) (CLASS +) EXAMPLES: 2

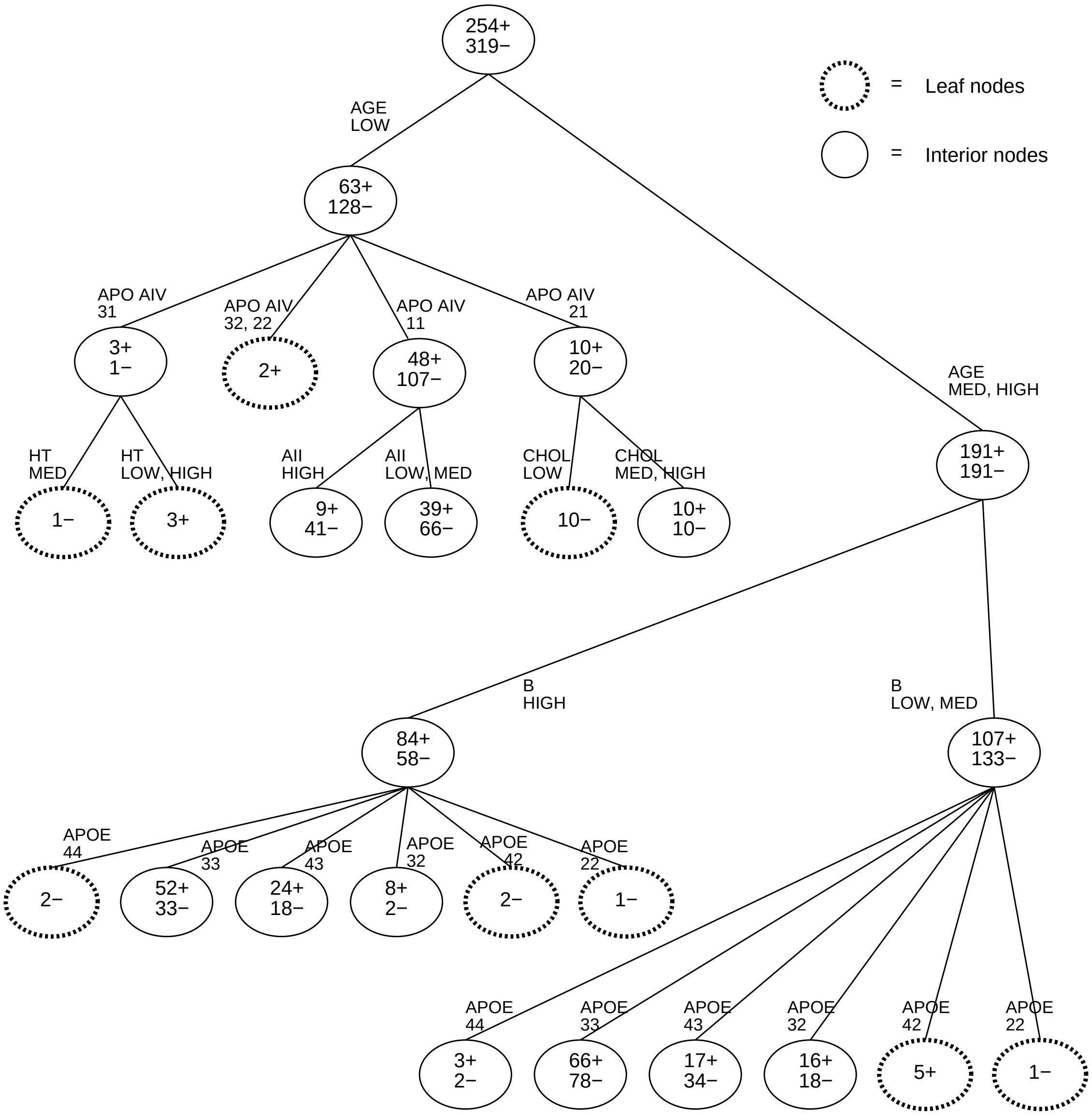


Figure F.1: An illustration of the first three levels of the decision tree found by GID3*.

Rule 12 (WT not 0) (AI 0) (E not 2) (APOE not 32) (DIA not 1) (CII not 1) (AII 2) (APOAIV 11) (AGE 0) (CLASS +) EXAMPLES: 2

Rule 13 (WT 0) (AI 0) (E not 2) (APOE not 32) (DIA not 1) (CII not 1) (AII 2) (APOAIV 11) (AGE 0) (CLASS -) EXAMPLES: 1

Rule 14 (E 2) (APOE not 32) (DIA not 1) (CII not 1) (AII 2) (APOAIV 11) (AGE 0) (CLASS -) EXAMPLES: 6

Rule 15 (DIA 1) (CII not 1) (AII 2) (APOAIV 11) (AGE 0) (CLASS -) EXAMPLES: 6

Rule 16 (CII 1) (AII 2) (APOAIV 11) (AGE 0) (CLASS -) EXAMPLES: 17

Rule 17 (SYS not 2) (HT 2) (E 2) (CNT 1) (AII not 2) (APOAIV 11) (AGE 0) (CLASS -) EXAMPLES: 4

Rule 18 (SYS 2) (HT 2) (E 2) (CNT 1) (AII not 2) (APOAIV 11) (AGE 0) (CLASS +) EXAMPLES: 1

Rule 19 (HT not 2) (E 2) (CNT 1) (AII not 2) (APOAIV 11) (AGE 0) (CLASS +) EXAMPLES: 2

Rule 20 (B not 2) (WT not 0) (APOH 21) (E not 2) (CNT 1) (AII not 2) (APOAIV 11) (AGE 0) (CLASS -) EXAMPLES: 3

Rule 21 (B 2) (WT not 0) (APOH 21) (E not 2) (CNT 1) (AII not 2) (APOAIV 11) (AGE 0) (CLASS +) EXAMPLES: 1

Rule 22 (WT 0) (APOH 21) (E not 2) (CNT 1) (AII not 2) (APOAIV 11) (AGE 0) (CLASS +) EXAMPLES: 1

Rule 23 (APOH not 21) (E not 2) (CNT 1) (AII not 2) (APOAIV 11) (AGE 0) (CLASS -) EXAMPLES: 18

Rule 24 (WT not 2) (CII 2) (GENDER 1) (SYS 1) (APOH 22) (CNT not 1) (AII not 2) (APOAIV 11) (AGE 0) (CLASS +) EXAMPLES: 1

Rule 25 (WT 2) (CII 2) (GENDER 1) (SYS 1) (APOH 22) (CNT not 1) (AII not 2) (APOAIV 11) (AGE 0) (CLASS -) EXAMPLES: 5

Rule 26 (WT not 1) (CII not 2) (GENDER 1) (SYS 1) (APOH 22) (CNT not 1) (AII not 2) (APOAIV 11) (AGE 0) (CLASS +) EXAMPLES: 4

Rule 27 (WT 1) (CII not 2) (GENDER 1) (SYS 1) (APOH 22) (CNT not 1) (AII not 2) (APOAIV 11) (AGE 0) (CLASS -) EXAMPLES: 3

Rule 28 (GENDER not 1) (SYS 1) (APOH 22) (CNT not 1) (AII not 2) (APOAIV 11) (AGE 0) (CLASS -) EXAMPLES: 10

Rule 29 (CII not 0) (HT not 1) (TRIG not 1) (SYS 2) (APOH 22) (CNT not 1) (AII not 2) (APOAIV 11) (AGE 0) (CLASS +) EXAMPLES: 4

Rule 30 (CII 0) (HT not 1) (TRIG not 1) (SYS 2) (APOH 22) (CNT not 1) (AII not 2) (APOAIV 11) (AGE 0) (CLASS -) EXAMPLES: 1

Rule 31 (HT 1) (TRIG not 1) (SYS 2) (APOH 22) (CNT not 1) (AII not 2) (APOAIV 11) (AGE 0) (CLASS -) EXAMPLES: 3

Rule 32 (TRIG 1) (SYS 2) (APOH 22) (CNT not 1) (AII not 2) (APOAIV 11) (AGE 0) (CLASS +) EXAMPLES: 4

Rule 33 (AI not 0) (DIA not 1) (AII not 1) (AII not 2) (CHOL not 1) (WT 0) (B 0) (SYS 0) (APOH 22) (CNT not 1) (APOAIV 11) (AGE 0) (CLASS -) EXAMPLES: 2

Rule 34 (AI 0) (DIA not 1) (AII not 1) (AII not 2) (CHOL not 1) (WT 0) (B 0) (SYS 0) (APOH 22) (CNT not 1) (APOAIV 11) (AGE 0) (CLASS +) EXAMPLES: 1

Rule 35 (DIA 1) (AII not 1) (AII not 2) (CHOL not 1) (WT 0) (B 0) (SYS 0) (APOH 22) (CNT not 1) (APOAIV 11) (AGE 0) (CLASS +) EXAMPLES: 1

Rule 36 (AII 1) (CHOL not 1) (WT 0) (B 0) (SYS 0) (APOH 22) (CNT not 1) (APOAIV 11) (AGE 0) (CLASS -) EXAMPLES: 4

Rule 37 (CHOL 1) (WT 0) (B 0) (SYS 0) (APOH 22) (CNT not 1) (AII not 2) (APOAIV 11) (AGE 0) (CLASS +) EXAMPLES: 1

Rule 38 (WT not 0) (B 0) (SYS 0) (APOH 22) (CNT not 1) (AII not 2) (APOAIV 11) (AGE 0) (CLASS -) EXAMPLES: 3

Rule 39 (HT not 1) (TRIG 1) (APOE not 32) (WHR not 2) (AI not 2) (B not 0) (SYS 0) (APOH 22) (CNT not 1) (AII not 2) (APOAIV 11) (AGE 0) (CLASS +) EXAMPLES: 2

Rule 40 (HT 1) (TRIG 1) (APOE not 32) (WHR not 2) (AI not 2) (B not 0) (SYS 0) (APOH 22) (CNT not 1) (AII not 2) (APOAIV 11) (AGE 0) (CLASS -) EXAMPLES: 1

Rule 41 (TRIG not 1) (APOE not 32) (WHR not 2) (AI not 2) (B not 0) (SYS 0) (APOH 22) (CNT not 1) (AII not 2) (APOAIV 11) (AGE 0) (CLASS -) EXAMPLES: 6

Rule 42 (APOE 32) (WHR not 2) (AI not 2) (B not 0) (SYS 0) (APOH 22) (CNT not 1) (AII not 2) (APOAIV 11) (AGE 0) (CLASS +) EXAMPLES: 2

Rule 43 (WHR 2) (AI not 2) (B not 0) (SYS 0) (APOH 22) (CNT not 1) (AII not 2) (APOAIV 11) (AGE 0) (CLASS +) EXAMPLES: 3

Rule 44 (AI 2) (B not 0) (SYS 0) (APOH 22) (CNT not 1) (AII not 2) (APOAIV 11) (AGE 0) (CLASS +) EXAMPLES: 4

Rule 45 (AI not 0) (APOH not 22) (CNT not 1) (AII not 2) (APOAIV 11) (AGE 0) (CLASS +) EXAMPLES: 7

Rule 46 (AI 0) (APOH not 22) (CNT not 1) (AII not 2) (APOAIV 11) (AGE 0) (CLASS -) EXAMPLES: 3

Rule 47 (DIA not 1) (WT 0) (HDL 1) (CHOL not 0) (APOAIV 21) (AGE 0) (CLASS +) EXAMPLES: 1

Rule 48 (DIA 1) (WT 0) (HDL 1) (CHOL not 0) (APOAIV 21) (AGE 0) (CLASS -) EXAMPLES: 1

Rule 49 (WT not 0) (HDL 1) (CHOL not 0) (APOAIV 21) (AGE 0) (CLASS -) EXAMPLES: 5

Rule 50 (SYS not 0) (DIA 0) (APOE not 32) (HDL not 1) (CHOL not 0) (APOAIV 21) (AGE 0) (CLASS +) EXAMPLES: 2

Rule 51 (SYS 0) (DIA 0) (APOE not 32) (HDL not 1) (CHOL not 0) (APOAIV 21) (AGE 0) (CLASS -) EXAMPLES: 3

Rule 52 (DIA not 0) (APOE not 32) (HDL not 1) (CHOL not 0) (APOAIV 21) (AGE 0) (CLASS +) EXAMPLES: 7

Rule 53 (APOE 32) (HDL not 1) (CHOL not 0) (APOAIV 21) (AGE 0) (CLASS -) EXAMPLES: 1

Rule 54 (CHOL 0) (APOAIV 21) (AGE 0) (CLASS -) EXAMPLES: 10

Rule 55 (APOAIV not 21) (APOAIV not 11) (APOAIV not 31) (AGE 0) (CLASS +) EXAMPLES: 2

Rule 56 (CIII not 0) (CIII not 1) (CNT 1) (GENDER not 2) (SYS 2) (CHOL not 1) (WHR 2) (APOAIV 11) (APOE 33) (B 2) (AGE not 0) (CLASS -) EXAMPLES: 2

Rule 57 (CIII 0) (CNT 1) (GENDER not 2) (SYS 2) (CHOL not 1) (WHR 2) (APOAIV 11) (APOE 33) (B 2) (AGE not 0) (CLASS +) EXAMPLES: 1

Rule 58 (AI not 2) (AII 0) (CNT not 1) (GENDER not 2) (CIII not 1) (SYS 2) (CHOL not 1) (WHR 2) (APOAIV 11) (APOE 33) (B 2) (AGE not 0) (CLASS +) EXAMPLES: 3

Rule 59 (AI 2) (AII 0) (CNT not 1) (GENDER not 2) (CIII not 1) (SYS 2) (CHOL not 1) (WHR 2) (APOAIV 11) (APOE 33) (B 2) (AGE not 0) (CLASS -) EXAMPLES: 1

Rule 60 (AII not 0) (CNT not 1) (GENDER not 2) (CIII not 1) (SYS 2) (CHOL not 1) (WHR 2) (APOAIV 11) (APOE 33) (B 2) (AGE not 0) (CLASS +) EXAMPLES: 7

Rule 61 (GENDER 2) (CIII not 1) (SYS 2) (CHOL not 1) (WHR 2) (APOAIV 11) (APOE 33) (B 2) (AGE not 0) (CLASS -) EXAMPLES: 1

Rule 62 (CIII 1) (SYS 2) (CHOL not 1) (WHR 2) (APOAIV 11) (APOE 33) (B 2) (AGE not 0) (CLASS -) EXAMPLES: 3

Rule 63 (AGE not 2) (AGE not 0) (CNT 0) (SYS not 2) (CHOL not 1) (WHR 2) (APOAIV 11) (APOE 33) (B 2) (CLASS +) EXAMPLES: 1

Rule 64 (AGE 2) (CNT 0) (SYS not 2) (CHOL not 1) (WHR 2) (APOAIV 11) (APOE 33) (B 2) (CLASS -) EXAMPLES: 1

Rule 65 (CNT not 0) (SYS not 2) (CHOL not 1) (WHR 2) (APOAIV 11) (APOE 33) (B 2) (AGE not 0) (CLASS +) EXAMPLES: 13

Rule 66 (CHOL 1) (WHR 2) (APOAIV 11) (APOE 33) (B 2) (AGE not 0) (CLASS +) EXAMPLES: 8

Rule 67 (DIA not 0) (E 1) (WHR not 2) (APOAIV 11) (APOE 33) (B 2) (AGE not 0) (CLASS +) EXAMPLES: 7

Rule 68 (DIA 0) (E 1) (WHR not 2) (APOAIV 11) (APOE 33) (B 2) (AGE not 0) (CLASS -) EXAMPLES: 1

Rule 69 (CNT not 0) (DIA 1) (HT 0) (E not 1) (WHR not 2) (APOAIV 11) (APOE 33) (B 2) (AGE not 0) (CLASS -) EXAMPLES: 3

Rule 70 (CNT 0) (DIA 1) (HT 0) (E not 1) (WHR not 2) (APOAIV 11) (APOE 33) (B 2) (AGE not 0) (CLASS +) EXAMPLES: 1

Rule 71 (DIA not 1) (HT 0) (E not 1) (WHR not 2) (APOAIV 11) (APOE 33) (B 2) (AGE not 0) (CLASS +) EXAMPLES: 3

Rule 72 (HT not 1) (HT not 0) (WT 2) (SMOKE not 1) (E not 1) (WHR not 2) (APOAIV 11) (APOE 33) (B 2) (AGE not 0) (CLASS -) EXAMPLES: 1

Rule 73 (HT 1) (WT 2) (SMOKE not 1) (E not 1) (WHR not 2) (APOAIV 11) (APOE 33) (B 2) (AGE not 0) (CLASS +) EXAMPLES: 1

Rule 74 (WT not 2) (SMOKE not 1) (HT not 0) (E not 1) (WHR not 2) (APOAIV 11) (APOE 33) (B 2) (AGE not 0) (CLASS -) EXAMPLES: 10

Rule 75 (SMOKE 1) (HT not 0) (E not 1) (WHR not 2) (APOAIV 11) (APOE 33) (B 2) (AGE not 0) (CLASS +) EXAMPLES: 1

Rule 76 (WHR not 0) (WT not 2) (DIA not 1) (CII not 0) (APOAIV not 11) (APOE 33) (B 2) (AGE not 0) (CLASS -) EXAMPLES: 5

Rule 77 (WHR 0) (WT not 2) (DIA not 1) (CII not 0) (APOAIV not 11) (APOE 33) (B 2) (AGE not 0) (CLASS +) EXAMPLES: 2

Rule 78 (WT 2) (DIA not 1) (CII not 0) (APOAIV not 11) (APOE 33) (B 2) (AGE not 0) (CLASS +) EXAMPLES: 3

Rule 79 (DIA 1) (CII not 0) (APOAIV not 11) (APOE 33) (B 2) (AGE not 0) (CLASS -) EXAMPLES: 5

Rule 80 (CII 0) (APOAIV not 11) (APOE 33) (B 2) (AGE not 0) (CLASS +) EXAMPLES: 1

Rule 81 (CNT not 2) (WT 1) (SMOKE 3) (SYS 0) (APOE 43) (B 2) (AGE not 0) (CLASS +) EXAMPLES: 1

Rule 82 (CNT 2) (WT 1) (SMOKE 3) (SYS 0) (APOE 43) (B 2) (AGE not 0) (CLASS -) EXAMPLES: 1

Rule 83 (WT not 1) (SMOKE 3) (SYS 0) (APOE 43) (B 2) (AGE not 0) (CLASS -) EXAMPLES: 5

Rule 84 (SMOKE not 3) (SYS 0) (APOE 43) (B 2) (AGE not 0) (CLASS +) EXAMPLES: 1

Rule 85 (E not 2) (CNT 2) (AGE 1) (HT 1) (SYS not 0) (APOE 43) (B 2) (CLASS -) EXAMPLES: 1

Rule 86 (E 2) (CNT 2) (AGE 1) (HT 1) (SYS not 0) (APOE 43) (B 2) (CLASS +) EXAMPLES: 1

Rule 87 (CNT not 2) (AGE 1) (HT 1) (SYS not 0) (APOE 43) (B 2) (CLASS +) EXAMPLES: 2

Rule 88 (AGE not 1) (AGE not 0) (HT 1) (SYS not 0) (APOE 43) (B 2) (CLASS +) EXAMPLES: 6

Rule 89 (HDL not 1) (CIII 1) (HT not 1) (SYS not 0) (APOE 43) (B 2) (AGE not 0) (CLASS -) EXAMPLES: 5

Rule 90 (HDL 1) (CIII 1) (HT not 1) (SYS not 0) (APOE 43) (B 2) (AGE not 0) (CLASS +) EXAMPLES: 1

Rule 91 (WT not 2) (SYS 1) (TRIG not 0) (TRIG not 1) (CHOL not 1) (CIII not 1) (HT not 1) (APOE 43) (B 2) (AGE not 0) (CLASS -) EXAMPLES: 4

Rule 92 (WT 2) (SYS 1) (TRIG not 0) (TRIG not 1) (CHOL not 1) (CIII not 1) (HT not 1) (APOE 43) (B 2) (AGE not 0) (CLASS +) EXAMPLES: 1

Rule 93 (AII not 0) (SYS not 1) (SYS not 0) (TRIG not 0) (TRIG not 1) (CHOL not 1) (CIII not 1) (HT not 1) (APOE 43) (B 2) (AGE not 0) (CLASS +) EXAMPLES: 5

Rule 94 (AII 0) (SYS not 1) (SYS not 0) (TRIG not 0) (TRIG not 1) (CHOL not 1) (CIII not 1) (HT not 1) (APOE 43) (B 2) (AGE not 0) (CLASS -) EXAMPLES: 1

Rule 95 (TRIG 0) (CHOL not 1) (CIII not 1) (HT not 1) (SYS not 0) (APOE 43) (B 2) (AGE not 0) (CLASS -) EXAMPLES: 1

Rule 96 (TRIG 1) (CHOL not 1) (CIII not 1) (HT not 1) (SYS not 0) (APOE 43) (B 2) (AGE not 0) (CLASS +) EXAMPLES: 2

Rule 97 (CHOL 1) (CIII not 1) (HT not 1) (SYS not 0) (APOE 43) (B 2) (AGE not 0) (CLASS +) EXAMPLES: 4

Rule 98 (APOH not 32) (E 2) (APOE 32) (B 2) (AGE not 0) (CLASS +) EXAMPLES: 8

Rule 99 (APOH 32) (E 2) (APOE 32) (B 2) (AGE not 0) (CLASS -) EXAMPLES: 1

Rule 100 (E not 2) (APOE 32) (B 2) (AGE not 0) (CLASS -) EXAMPLES: 1

Rule 101 (APOE not 42) (APOE not 32) (APOE not 43) (APOE not 33) (APOE not 44) (B 2) (AGE not 0) (CLASS -) EXAMPLES: 1

Rule 102 (APOE 42) (B 2) (AGE not 0) (CLASS -) EXAMPLES: 2

Rule 103 (APOE 44) (B 2) (AGE not 0) (CLASS -) EXAMPLES: 2

Rule 104 (WT not 1) (E not 0) (SYS 1) (DIA not 0) (AII 1) (HT 1) (APOE 33) (B not 2) (AGE not 0) (CLASS -) EXAMPLES: 5

Rule 105 (WT 1) (E not 0) (SYS 1) (DIA not 0) (AII 1) (HT 1) (APOE 33) (B not 2) (AGE not 0) (CLASS +) EXAMPLES: 2

Rule 106 (E 0) (SYS 1) (DIA not 0) (AII 1) (HT 1) (APOE 33) (B not 2) (AGE not 0) (CLASS +) EXAMPLES: 2

Rule 107 (SYS not 1) (DIA not 0) (AII 1) (HT 1) (APOE 33) (B not 2) (AGE not 0) (CLASS +) EXAMPLES: 3

Rule 108 (DIA 0) (AII 1) (HT 1) (APOE 33) (B not 2) (AGE not 0) (CLASS -) EXAMPLES: 6

Rule 109 (E not 0) (AI 0) (APOAIV not 21) (AII not 1) (HT 1) (APOE 33) (B not 2) (AGE not 0) (CLASS +) EXAMPLES: 7

Rule 110 (E 0) (AI 0) (APOAIV not 21) (AII not 1) (HT 1) (APOE 33) (B not 2) (AGE not 0) (CLASS -) EXAMPLES: 1

Rule 111 (DIA not 1) (AII 0) (CNT 1) (SMOKE 3) (AI not 0) (APOAIV not 21) (HT 1) (APOE 33) (B not 2) (AGE not 0) (CLASS +) EXAMPLES: 2

Rule 112 (DIA 1) (AII 0) (CNT 1) (SMOKE 3) (AI not 0) (APOAIV not 21) (HT 1) (APOE 33) (B not 2) (AGE not 0) (CLASS -) EXAMPLES: 2

Rule 113 (AII not 0) (AII not 1) (CNT 1) (SMOKE 3) (AI not 0) (APOAIV not 21) (HT 1) (APOE 33) (B not 2) (AGE not 0) (CLASS +) EXAMPLES: 7

Rule 114 (WT not 0) (AGE 2) (SYS not 2) (CIII not 0) (CNT not 1) (SMOKE 3) (AI not 0) (APOAIV not 21) (AII not 1) (HT 1) (APOE 33) (B not 2) (CLASS -) EXAMPLES: 1

Rule 115 (WT 0) (AGE 2) (SYS not 2) (CIII not 0) (CNT not 1) (SMOKE 3) (AI not 0) (APOAIV not 21) (AII not 1) (HT 1) (APOE 33) (B not 2) (CLASS +) EXAMPLES: 1

Rule 116 (AGE not 2) (AGE not 0) (SYS not 2) (CIII not 0) (CNT not 1) (SMOKE 3) (AI not 0) (APOAIV not 21) (AII not 1) (HT 1) (APOE 33) (B not 2) (CLASS -) EXAMPLES: 4

Rule 117 (SYS 2) (CIII not 0) (CNT not 1) (SMOKE 3) (AI not 0) (APOAIV not 21) (AII not 1) (HT 1) (APOE 33) (B not 2) (AGE not 0) (CLASS +) EXAMPLES: 1

Rule 118 (CIII 0) (CNT not 1) (SMOKE 3) (AI not 0) (APOAIV not 21) (AII not 1) (HT 1) (APOE 33) (B not 2) (AGE not 0) (CLASS +) EXAMPLES: 1

Rule 119 (WT not 0) (SMOKE not 3) (AI not 0) (APOAIV not 21) (AII not 1) (HT 1) (APOE 33) (B not 2) (AGE not 0) (CLASS -) EXAMPLES: 3

Rule 120 (WT 0) (SMOKE not 3) (AI not 0) (APOAIV not 21) (AII not 1) (HT 1) (APOE 33) (B not 2) (AGE not 0) (CLASS +) EXAMPLES: 1

Rule 121 (APOAIV 21) (AII not 1) (HT 1) (APOE 33) (B not 2) (AGE not 0) (CLASS +) EXAMPLES: 4

Rule 122 (WHR not 2) (WT 1) (CHOL 2) (HT not 1) (APOE 33) (B not 2) (AGE not 0) (CLASS +) EXAMPLES: 2

Rule 123 (WHR 2) (WT 1) (CHOL 2) (HT not 1) (APOE 33) (B not 2) (AGE not 0) (CLASS -) EXAMPLES: 2

Rule 124 (WT not 1) (CHOL 2) (HT not 1) (APOE 33) (B not 2) (AGE not 0) (CLASS -) EXAMPLES: 11

Rule 125 (AII not 0) (HT 2) (AI not 0) (CNT not 1) (SMOKE 3) (E 2) (CHOL not 2) (APOE 33) (B not 2) (AGE not 0) (CLASS -) EXAMPLES: 2

Rule 126 (AII 0) (HT 2) (AI not 0) (CNT not 1) (SMOKE 3) (E 2) (CHOL not 2) (APOE 33) (B not 2) (AGE not 0) (CLASS +) EXAMPLES: 1

Rule 127 (HT not 2) (HT not 1) (AI not 0) (CNT not 1) (SMOKE 3) (E 2) (CHOL not 2) (APOE 33) (B not 2) (AGE not 0) (CLASS +) EXAMPLES: 6

Rule 128 (AI 0) (CNT not 1) (SMOKE 3) (E 2) (CHOL not 2) (HT not 1) (APOE 33) (B not 2) (AGE not 0) (CLASS -) EXAMPLES: 2

Rule 129 (CNT 1) (SMOKE 3) (E 2) (CHOL not 2) (HT not 1) (APOE 33) (B not 2) (AGE not 0) (CLASS +) EXAMPLES: 6

Rule 130 (SMOKE not 3) (E 2) (CHOL not 2) (HT not 1) (APOE 33) (B not 2) (AGE not 0) (CLASS -) EXAMPLES: 1

Rule 131 (CII not 0) (CIII 2) (TRIG 2) (APOAIV 11) (E not 2) (CHOL not 2) (HT not 1) (APOE 33) (B not 2) (AGE not 0) (CLASS +) EXAMPLES: 1

Rule 132 (CII 0) (CIII 2) (TRIG 2) (APOAIV 11) (E not 2) (CHOL not 2) (HT not 1) (APOE 33) (B not 2) (AGE not 0) (CLASS -) EXAMPLES: 1

Rule 133 (CIII not 2) (TRIG 2) (APOAIV 11) (E not 2) (CHOL not 2) (HT not 1) (APOE 33) (B not 2) (AGE not 0) (CLASS +) EXAMPLES: 3

Rule 134 (WT not 1) (B not 1) (B not 2) (AII 0) (DIA not 2) (APOH not 21) (APOH not 32) (SMOKE 3) (E 0) (TRIG not 2) (APOAIV 11) (CHOL not 2) (HT not 1) (APOE 33) (AGE not 0) (CLASS -) EXAMPLES: 2

Rule 135 (WT 1) (B not 1) (B not 2) (AII 0) (DIA not 2) (APOH not 21) (APOH not 32) (SMOKE 3) (E 0) (TRIG not 2) (APOAIV 11) (CHOL not 2) (HT not 1) (APOE 33) (AGE not 0) (CLASS +) EXAMPLES: 1

Rule 136 (B 1) (AII 0) (DIA not 2) (APOH not 21) (APOH not 32) (SMOKE 3) (E 0) (TRIG not 2) (APOAIV 11) (CHOL not 2) (HT not 1) (APOE 33) (AGE not 0) (CLASS +) EXAMPLES: 3

Rule 137 (B not 0) (B not 2) (WT 1) (AII not 0) (DIA not 2) (APOH not 21) (APOH not 32) (SMOKE 3) (E 0) (TRIG not 2) (APOAIV 11) (CHOL not 2) (HT not 1) (APOE 33) (AGE not 0) (CLASS -) EXAMPLES: 2

Rule 138 (B 0) (WT 1) (AII not 0) (DIA not 2) (APOH not 21) (APOH not 32) (SMOKE 3) (E 0) (TRIG not 2) (APOAIV 11) (CHOL not 2) (HT not 1) (APOE 33) (AGE not 0) (CLASS +) EXAMPLES: 1

Rule 139 (WT not 1) (AII not 0) (DIA not 2) (APOH not 21) (APOH not 32) (SMOKE 3) (E 0) (TRIG not 2) (APOAIV 11) (CHOL not 2) (HT not 1) (APOE 33) (B not 2) (AGE not 0) (CLASS -) EXAMPLES: 4

Rule 140 (DIA 2) (APOH not 21) (APOH not 32) (SMOKE 3) (E 0) (TRIG not 2) (APOAIV 11) (CHOL not 2) (HT not 1) (APOE 33) (B not 2) (AGE not 0) (CLASS +) EXAMPLES: 3

Rule 141 (APOH 21) (SMOKE 3) (E 0) (TRIG not 2) (APOAIV 11) (CHOL not 2) (HT not 1) (APOE 33) (B not 2) (AGE not 0) (CLASS -) EXAMPLES: 1

Rule 142 (SMOKE not 3) (APOH not 32) (E 0) (TRIG not 2) (APOAIV 11) (CHOL not 2) (HT not 1) (APOE 33) (B not 2) (AGE not 0) (CLASS -) EXAMPLES: 2

Rule 143 (APOH 32) (E 0) (TRIG not 2) (APOAIV 11) (CHOL not 2) (HT not 1) (APOE 33) (B not 2) (AGE not 0) (CLASS +) EXAMPLES: 3

Rule 144 (CIII not 2) (CNT not 2) (DIA 1) (E not 0) (E not 2) (TRIG not 2) (APOAIV 11) (CHOL not 2) (HT not 1) (APOE 33) (B not 2) (AGE not 0) (CLASS -) EXAMPLES: 3

Rule 145 (CIII 2) (CNT not 2) (DIA 1) (E not 0) (E not 2) (TRIG not 2) (APOAIV 11) (CHOL not 2) (HT not 1) (APOE 33) (B not 2) (AGE not 0) (CLASS +) EXAMPLES: 1

Rule 146 (CNT 2) (DIA 1) (E not 0) (E not 2) (TRIG not 2) (APOAIV 11) (CHOL not 2) (HT not 1) (APOE 33) (B not 2) (AGE not 0) (CLASS +) EXAMPLES: 2

Rule 147 (GENDER not 2) (APOH 21) (SMOKE not 1) (DIA 0) (E not 0) (E not 2) (TRIG not 2) (APOAIV 11) (CHOL not 2) (HT not 1) (APOE 33) (B not 2) (AGE not 0) (CLASS -) EXAMPLES: 1

Rule 148 (GENDER 2) (APOH 21) (SMOKE not 1) (DIA 0) (E not 0) (E not 2) (TRIG not 2) (APOAIV 11) (CHOL not 2) (HT not 1) (APOE 33) (B not 2) (AGE not 0) (CLASS +) EXAMPLES: 1

Rule 149 (APOH not 21) (SMOKE not 1) (DIA 0) (E not 0) (E not 2) (TRIG not 2) (APOAIV 11) (CHOL not 2) (HT not 1) (APOE 33) (B not 2) (AGE not 0) (CLASS -) EXAMPLES: 6

Rule 150 (SMOKE 1) (DIA 0) (E not 0) (E not 2) (TRIG not 2) (APOAIV 11) (CHOL not 2) (HT not 1) (APOE 33) (B not 2) (AGE not 0) (CLASS +) EXAMPLES: 1

Rule 151 (DIA 2) (E not 0) (E not 2) (TRIG not 2) (APOAIV 11) (CHOL not 2) (HT not 1) (APOE 33) (B not 2) (AGE not 0) (CLASS -) EXAMPLES: 11

Rule 152 (APOAIV not 11) (E not 2) (CHOL not 2) (HT not 1) (APOE 33) (B not 2) (AGE not 0) (CLASS -) EXAMPLES: 5

Rule 153 (AI not 0) (HT 1) (SMOKE 3) (APOE 32) (B not 2) (AGE not 0) (CLASS -) EXAMPLES: 6

Rule 154 (AI 0) (HT 1) (SMOKE 3) (APOE 32) (B not 2) (AGE not 0) (CLASS +) EXAMPLES: 2

Rule 155 (WHR not 0) (HT 0) (WT not 0) (AI 0) (SMOKE 3) (APOE 32) (B not 2) (AGE not 0) (CLASS +) EXAMPLES: 1

Rule 156 (WHR 0) (HT 0) (WT not 0) (AI 0) (SMOKE 3) (APOE 32) (B not 2) (AGE not 0) (CLASS -) EXAMPLES: 1

Rule 157 (HT not 0) (HT not 1) (WT not 0) (AI 0) (SMOKE 3) (APOE 32) (B not 2) (AGE not 0) (CLASS -) EXAMPLES: 4

Rule 158 (WT 0) (AI 0) (HT not 1) (SMOKE 3) (APOE 32) (B not 2) (AGE not 0) (CLASS +) EXAMPLES: 1

Rule 159 (APOH 21) (AI not 0) (HT not 1) (SMOKE 3) (APOE 32) (B not 2) (AGE not 0) (CLASS +) EXAMPLES: 2

Rule 160 (APOH 22) (AI not 0) (HT not 1) (SMOKE 3) (APOE 32) (B not 2) (AGE not 0) (CLASS +) EXAMPLES: 10

Rule 161 (APOH 32) (AI not 0) (HT not 1) (SMOKE 3) (APOE 32) (B not 2) (AGE not 0) (CLASS -) EXAMPLES: 1

Rule 162 (SMOKE not 3) (APOE 32) (B not 2) (AGE not 0) (CLASS -) EXAMPLES: 6

Rule 163 (AII not 1) (SYS 1) (B 1) (APOE 43) (AGE not 0) (CLASS +) EXAMPLES: 7

Rule 164 (AII 1) (SYS 1) (B 1) (APOE 43) (AGE not 0) (CLASS -) EXAMPLES: 1

Rule 165 (CNT not 1) (CIII not 1) (DIA not 2) (WT not 0) (APOAIV 11) (SYS not 1) (B 1) (APOE 43) (AGE not 0) (CLASS -) EXAMPLES: 3

Rule 166 (CNT 1) (CIII not 1) (DIA not 2) (WT not 0) (APOAIV 11) (SYS not 1) (B 1) (APOE 43) (AGE not 0) (CLASS +) EXAMPLES: 1

Rule 167 (CIII 1) (DIA not 2) (WT not 0) (APOAIV 11) (SYS not 1) (B 1) (APOE 43) (AGE not 0) (CLASS +) EXAMPLES: 2

Rule 168 (DIA 2) (WT not 0) (APOAIV 11) (SYS not 1) (B 1) (APOE 43) (AGE not 0) (CLASS -) EXAMPLES: 5

Rule 169 (WT 0) (APOAIV 11) (SYS not 1) (B 1) (APOE 43) (AGE not 0) (CLASS +) EXAMPLES: 2

Rule 170 (APOAIV not 11) (SYS not 1) (B 1) (APOE 43) (AGE not 0) (CLASS -) EXAMPLES: 4

Rule 171 (DIA not 2) (CNT 0) (AI not 0) (AI not 1) (HT not 0) (TRIG not 1) (WHR not 1) (B not 1) (B not 2) (APOE 43) (AGE not 0) (CLASS +) EXAMPLES: 1

Rule 172 (DIA 2) (CNT 0) (AI not 0) (AI not 1) (HT not 0) (TRIG not 1) (WHR not 1) (B not 1) (B not 2) (APOE 43) (AGE not 0) (CLASS -) EXAMPLES: 1

Rule 173 (CNT not 0) (CNT not 2) (AI not 0) (AI not 1) (HT not 0) (TRIG not 1) (WHR not 1) (B not 1) (B not 2) (APOE 43) (AGE not 0) (CLASS -) EXAMPLES: 2

Rule 174 (AI 0) (HT not 0) (TRIG not 1) (WHR not 1) (CNT not 2) (B not 1) (B not 2) (APOE 43) (AGE not 0) (CLASS +) EXAMPLES: 1

Rule 175 (HT 0) (AI not 1) (TRIG not 1) (WHR not 1) (CNT not 2) (B not 1) (B not 2) (APOE 43) (AGE not 0) (CLASS -) EXAMPLES: 2

Rule 176 (AI 1) (TRIG not 1) (WHR not 1) (CNT not 2) (B not 1) (B not 2) (APOE 43) (AGE not 0) (CLASS -) EXAMPLES: 5

Rule 177 (TRIG 1) (WHR not 1) (CNT not 2) (B not 1) (B not 2) (APOE 43) (AGE not 0) (CLASS +) EXAMPLES: 1

Rule 178 (WHR 1) (CNT not 2) (B not 1) (B not 2) (APOE 43) (AGE not 0) (CLASS +) EXAMPLES: 2

Rule 179 (CNT 2) (B not 1) (B not 2) (APOE 43) (AGE not 0) (CLASS -) EXAMPLES: 11

Rule 180 (CIII not 1) (APOE 44) (B not 2) (AGE not 0) (CLASS +) EXAMPLES: 3

Rule 181 (CIII 1) (APOE 44) (B not 2) (AGE not 0) (CLASS -) EXAMPLES: 2

Rule 182 (APOE not 44) (APOE not 43) (APOE not 32) (APOE not 33) (APOE not 42) (B not 2) (AGE not 0) (CLASS -) EXAMPLES: 1

Rule 183 (APOE 42) (B not 2) (AGE not 0) (CLASS +) EXAMPLES: 5

APPENDIX G

GLOSSARY OF TERMS

angina pectoris — A condition marked by recurrent pain in the chest and left arm, caused by a sudden decrease of the blood supply to the heart muscle [107]., 7

attribute — A feature or dimension that applies to each example in the dataset., 21

attribute-value pair — An attribute and a specific value of that attribute., 21

binary classification task — A classification task with exactly two classes., 23

class attribute — More commonly called simply the "class"; this is the category of interest for the classification task., 23

classification — Given a set of examples (a dataset), the task of classification is to form a decision structure (e.g., a rule set) that divides the set into subgroups of similar examples and enables a new example to be placed with the examples that are most similar to it., 21

conjunct of disjuncts — The conjunct of disjuncts rule format is conjunctive in that the rule must match for all specified attributes and disjunctive in that any attribute may be specified by a disjunctive expression indicating one or more values., 23

epigenetic — A disease or other physical trait whose presence or absence is thought to be determined by multiple interacting genes, gene products, and environmental signals., 9

Coronary Artery Disease (CAD) — Also called ischemic heart disease and coronary heart disease, this term is applied to heart ailments that are caused by narrowing (or obstruction) of the coronary arteries and therefore characterized by a decreased blood supply to the heart., 7

etiology — A theory of the causes of a particular disease., 8

example — A set of attribute-value pairs that together comprise one entry or instance in the dataset., 22

genotype — The genetic traits of a person; their genetic characteristics., 10

heart attack — death of, or damage to, part of the heart muscle due to an insufficient blood supply., 7

homeostasis — The tendency to maintain, or the maintenance of, normal, internal stability in an organism by coordinated responses of the organ systems that automatically compensate for environmental changes [107]., 10

incremental learning system — An incremental learning system constructs its classification by considering one example at a time and modifying its current classification based on the new example. Therefore, changing the order of the examples in the dataset may change the resulting classification., 44

Mendelian gene — following the laws of Gregor Mendel; of particular relevance here is the law of single unit characters, which states that characters (such as height, color, etc.) are inherited separately as units [107]. As used here, a Mendelian gene would imply that there is a one-to-one relationship between variation in a gene and disease status., 9

monogenic — A disease or other physical trait whose presence or absence is determined by a single gene., 9

morbidity — The rate of disease or proportion of diseased persons in a given locality, nation, etc. [107]., 7

mortality — The proportion of deaths to the population of a region, nation, etc.; death rate [107]., 7

myocardial infarction — the damaging or death of an area of the heart muscle (myocardium) resulting from a reduced blood supply to that area., 7

noise — Noise in a dataset is a vague term that can apply to any of a number of factors that may make the dataset more difficult to classify. For example, there may be errors (typographical errors or measuring errors) in the dataset, or it may be that two examples in the dataset have identical values for all attributes but are in two different classes. The latter type of noise may be due to an error, but can sometimes be attributed to not considering enough attributes. That is, if additional attributes were included in the dataset, the two examples might not be identical for all attributes., 30

nonincremental learning system — A nonincremental learning system forms its classification using all the examples available in the dataset (or training set). The order of the examples is not usually relevant to a nonincremental learning system., 44

overfit — A rule set is said to overfit the data when it describes the training data at the expense of generalizing to test data., 27

phenotype — The expressed traits of a person; their physical characteristics., 10

polygenic — A disease or other physical trait whose presence or absence is determined by two or more genes, working together., 9

polymorphic at the protein level — Genes encode proteins. Thus, "polymorphic at the protein level" indicates that there is variation in the gene's DNA sequence, resulting in variation in the amino acid sequence of the protein., 17

prediction — The task of classifying examples not in the original training data set., 25

prediction accuracy — The metric commonly used to evaluate classifications, equal to the percent of examples in the test set that are correctly classified by the rule set., 25

rule — A set of attribute-value pairs that describe some subset of the dataset., 23

rule set — A collection of rules that describe a dataset., 23

specified — A specified attribute appears in one or more attribute-value pairs in the set that constitutes a rule, and is used in the matching process., 23

supervised learning system — A classification system that uses and requires a class attribute in the dataset. Such a system is looking for a description of the dataset based on the class attribute., 38

terminology — The terms or system of terms used in a specific science, art, etc.; nomenclature [107]., 6

test set — The data set used to evaluate the rule set in a classification task., 25

training set — The data set used to form the rule set in a classification task., 25

unlinked genes — When two genes are unlinked, they are found on different chromosomes; variation in one is not associated with variation in the other., 17

unspecified — An unspecified attribute does not appear in the set of attribute-value pairs that constitute a rule, and is not involved in the matching process., 23

unsupervised learning system — A classification system that does not regard the class attribute of the examples as more important than other attributes, or ignores the class attribute altogether. Such a system defines its own classes within the dataset, finding similarities in the values of attributes., 38

BIBLIOGRAPHY

[1] A. J. Anderson, R. F. Loeffler, J. J. Barboriak, and A. A. Rimm. Occlusive coronary artery disease and parental history of myocardial infarction. *Prev Med*, 8:419–428, 1979.

[2] American Heart Association. 1993 heart and stroke fact statistics, 1993.

[3] M. A. Austin, J. D. Brunzell, W. L. Fitch, and R. M. Krauss. Inheritance of low density lipoprotein subclass patterns in familial combined hyperlipidemia. *Arteriosclerosis*, 10:520–530, 1990.

[4] E. Barrett-Connorr and K. Khaw. Family history of heart attack as an independent predictor of death due to cardiovascular disease. *Circulation*, 69:1065–9, 1984.

[5] James C. Bean. Genetics and random keys for sequencing and optimization. Technical Report TR-92-43, Department of Industrial and Operations Engineering, The University of Michigan, 1992; revised 1993.

[6] James C. Bean and Atidel Ben Hadj-Alouane. A dual genetic algorithm for bounded integer programs. Technical Report TR-92-53, Department of Industrial and Operations Engineering, The University of Michigan, 1992.

[7] James C. Bean and Bryan A. Norman. Random keys for job shop scheduling. Technical Report TR-93-7, Department of Industrial and Operations Engineering, The University of Michigan, 1993.

[8] Gerald S. Berenson. Epidemiologic investigations of cardiovascular risk factor variables in childhood — an overview. *Acta Peadiatr Scand Suppl*, 318:7–9, 1985.

[9] Gerald S. Berenson, Sathanur R. Srinivasan, and Larry S. Webber. Prognostic significance of lipid profiles in children. In R. M. Lauer and R. B. Shekelle, editors, *Childhood Prevention of Atherosclerosis and Hypertension*, pages 75–86. Raven Press, New York, 1980.

[10] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Mandred K. Warmuth. Occam's razor. *Information Processing Letters*, 24(6):377–380, 1987.

[11] Marko Bohanec and Ivan Bratko. Trading accuracy for simplicity in decision trees. *Machine Learning*, 15(3):223–250, 1994.

[12] E. Braunwald. *Heart Disease*. W. B. Saunders Company, Philadelphia, 1988.

[13] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth & Brooks, Monterey, CA, 1984.

[14] L. Brieman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth & Brooks, 1984.

[15] Wray Buntine. Classifiers: A theoretical and empirical study. In *Proceedings of the Twelvth International Joint Conference on Artificial Intelligence (IJCAI), Sydney, Australia.* Morgan Kaufmann, San Mateo, CA, 1991.

[16] Claudio Carpiento and Giovanni Romano. Galois: An order-theoretic approach to conceptual clustering. In *Proceedings of the Tenth International Conference on Machine Learning, Amherst, MA*, pages 33–40. Morgan Kaufmann, San Mateo, CA, 1993.

[17] P. Cheeseman, J. Kelly, M. Self, J. Stutz, W. Taylor, and D. Freeman. AutoClass: A Bayesian classification system. In *Proceedings of the Fifth International Conference on Machine Learning, Ann Arbor, MI.* Morgan Kaufmann, San Mateo, CA, 1988.

[18] J. Cheng, U. M. Fayyad, K. B. Irani, and Z. Quian. Improved decision trees: A generalized version of ID3. In *Proceedings of the Fifth International Conference on Machine Learning, Ann Arbor, MI*, pages 100–108. Morgan Kaufmann, San Mateo, CA, 1988.

[19] B. Childs. Persistent echos of the nature-nurture argument. *American Journal of Human Genetics*, 29:1–13, 1977.

[20] C. B. Congdon, C. F. Sing, and S. L. Reilly. Genetic algorithms for identifying combinations of genes and other risk-factors associated with coronary artery disease. In *Proceedings of the Workshop on Artificial Intelligence and the Genome, International Joint Conference on Artificial Intelligence, Chambery, France*, 1993.

[21] J. Davignon, R. Dufour, and M. Contin. Atherosclerosis and hypertension. In G. Jacques *et al*, editor, *Hypertension, Physiopathology, and Treatment*, pages 810–852. McGraw-Hill, New York, 1983.

[22] Lawrence Davis, editor. *Genetic Algorithms and Simulated Annealing.* Pitman/Morgan Kaufmann, Los Altos, CA, 1987.

[23] Lawrence Davis. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, New York, NY, 1991.

[24] T. R. Dawber. *The Framingham Study. The epidemiology of atherosclerotic disease.* Harvard University Press, Cambridge, 1980.

[25] Kalyanmoy Deb and David E. Goldberg. An investigation of niche and species formation in genetic function optimization. In *Proceedings of the Third International Conference on Genetic Algorithms.* Morgan Kaufmann, San Mateo, CA, 1989.

[26] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[27] Barbara A. Dennison, David A. Kikuchi, Sathanur R. Srinivasan, Larry S. Webber, and Gerald S. Berenson. Parental history of cardiovascular disease as an indication for screening for lipoprotein abnormalities in children. *The Journal of Pediatrics*, 115(2):186–194, August 1989.

[28] S. Deutscher, L. D. Ostrander, and F. H. Epstein. Familial factors in premature coronary heart disease — preliminary report from the Tecumseh community health study. *American Journal of Epidemiology*, 91:233+, 1970.

[29] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, New York, 1973.

[30] W. F. Enos, R. H. Holmes, and J. Beyer. Coronary disease among United States soldiers killed in action in Korea. *Journal of the American Medical Association*, 152:1090–1093, 1953.

[31] G. Fager, O. Wiklund, S. O. Olofsson, L. Wilhelmsen, and G. Bondjers. Multivariate analyses of serum apolipoproteins and risk factors in relation to acute myocardial infarction. *Arteriosclerosis*, 1:273–279, 1981.

[32] Usama M. Fayyad. *On the induction of decision trees for multiple concept learning.* PhD thesis, Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, September 1991.

[33] Usama M. Fayyad and Keki B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of Thirteenth International Joint Conference on Artificial Intelligence (IJCAI), Chambery, France.* Morgan Kaufmann, San Mateo, CA, 1993.

[34] Usama M. Fayyad, Nicholas Weir, and S. Djorgovski. Skicat: A machine learning system for automated cataloging of large scale sky surveys. In *Proceedings of the Tenth International Conference on Machine Learning, Amherst, MA*, pages 112–119. Morgan Kaufmann, San Mateo, CA, 1993.

[35] Douglas H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172, 1987.

[36] Lawrence J. Fogel, Alvin J. Owens, and Michael J. Walsh. *Artificial Intelligence through simulated evolution.* John Wiley & Sons, New York, NY, 1966.

[37] David S. Freedman, Sathanur R. Srinivasan, Charles L. Shear, Frank A. Franklin, Larry S. Webber, and Gerald S. Berenson. The relation of apolipoproteins a-i and b in children to parental myocardial infarction. *New England Journal of Medicine*, 315(12):721–726, September 18 1986.

[38] Johannes Furnkranz and Gerhard Widmer. Incremental reduced error pruning. In *Proceedings of the Eleventh International Conference on Machine Learning, New Brunswick, NJ*, pages 70–77. Morgan Kaufmann, San Francisco, 1994.

[39] Attilio Giordana, Lorenza Saitta, and Floriano Zini. Learning disjunctive concepts by means of genetic algorithms. In *Proceedings of the Eleventh International Conference on Machine Learning, New Brunswick, NJ*, pages 96–97. Morgan Kaufmann, San Francisco, 1994.

[40] James Gleick. *Chaos: Making a New Science.* Viking Penguin, Inc., 1987.

[41] D. E. Goldberg. *Genetic Algorithms in Search Optimization & Machine Learning.* Addison-Wesley, Reading, MA, 1989.

[42] David E. Goldberg and Jon Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms (ICGA-87)*, pages 41–49. Lawrence Erlbaum Associates, 1987.

[43] David E. Goldberg and Philip Segrest. Finite markov chain analysis of genetic algorithms. In *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms (ICGA-87)*, pages 1–8. Lawrence Erlbaum Associates, 1987.

[44] D. P. Greene and S. F. Smith. Competition-based induction of decision models from examples. *Machine Learning*, 13(2/3):229–257, November/December 1993.

[45] J. J. Grefenstette. A user's guide to GENESIS. Technical report, Navy Center for Applied Research in AI, Washington, DC, 1987.

[46] Atidel Ben Hadj-Alouane and James C. Bean. A genetic algorithm for the multiple-choice integer program. Technical Report TR-92-50, Department of Industrial and Operations Engineering, The University of Michigan, 1992.

[47] A. Hamsten, G. Walldius, A. Szamosi, G. Dahlen, and U. de Faire. Relationship of angiographically defined coronary artery disease to serum lipoproteins and apolipoproteins in young survivors of myocardial infarction. *Circulation*, 73:1097–1110, 1986.

[48] Charles H. Hennekens, Mary Jane Jesse, Barbara E. Klein, Janet E. Gourley, and Sidney Bluementhal. Cholesterol among children of men with myocardial infarction. *Pediatrics*, 58(2):211–217, August 1976.

[49] M. W. Higgins and R. V. Luepker. Trends and determinants of coronary heart disease mortality: International comparisons. *International Journal of Epidemiology*, 18(3, supplement 1), 1989.

[50] Daniel Hillis. Co-evolving parasites improves simulated evolution as an optimization procedure. In C. Langton, C. Taylor, J. Farmer, and S. Rasmussen, editors, *Artificial Life II*, Reading, MA, 1992. Addison-Wesley.

[51] John H. Holland. *Adaptation in Natural and Artificial Systems.* MIT Press, Cambridge, MA, 1992.

[52] John H. Holland. Innovation in complex adaptive systems: Some mathematical sketches. Technical report, Santa Fe Institute Working Paper, Santa Fe, NM, 1993.

[53] John H. Holland. Personal communication, 1993 & 1994.

[54] R. L. Holman, H. C. McGill Jr., J. P. strong, and J.C. Geer. The natural history of atherosclerosis. *American Journal of Pathology*, 34(2):209–235, 1958.

[55] Robert C. Holte. Very simple classification rules perform well on most commonly used data sets. *Machine Learning*, 11(1):63–91, 1993.

[56] Earl B. Hunt, Janet Marin, and Philip J. Stone. *Experiments in Induction.* Academic Press, Inc., New York, 1966.

[57] Lawrence Hunter and David J. States. Bayesian classification of protein structure. *IEEE Expert*, pages 67 –75, 1992.

[58] W. Iba, J. Wogulis, and P. Langley. Trading off simplicity and coverage in incremental concept learning. In *Proceedings of the Fifth International Conference on Machine Learning, Ann Arbor, MI*, pages 73–79. Morgan Kaufmann, San Mateo, CA, 1988.

[59] George H. John, Ron Kohavi, and Karl Pfleger. Irrelevant features and the subset selection problem. In *Proceedings of the Eleventh International Conference on Machine Learning, New Brunswick, NJ*, pages 121–129. Morgan Kaufmann, San Francisco, 1994.

[60] K. De Jong. Learning with genetic algorithms: An overview. *Machine Learning*, 3(2/3):121–138, 1988.

[61] W. B. Kannel and A. Schatzkin. Sudden death: Lessons from subsets in population studies. *Journal of American Coll Cardiology*, 5:141B–148B, 1985.

[62] J. Kaprio, R. E. Ferrell, B. A. Kottke, M. I. Kamboh, and C. F. Sing. Effects of polymorphisms in apolipoproteins E, A-IV, and H on quantitative traits related to risk for cardiovascular disease. *Arteriosclerosis and Thrombosis*, 11(5), 1991.

[63] S. Karlin. Comments on statistical methodology in medical genetics. In *Genetic Analysis of common diseases: Applications to predictive factors in coronary disease*, pages 497–520. Alan R. Liss, Inc., New York, 1979.

[64] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, Cambridge, MA, 1992.

[65] M. S. Kramer. *Clinical Epidemiology and Biostatistics: A Primer for Clinical Investigators and Decision-Makers.* Springer-Verlag, Berlin, 1988.

[66] R. M. Krauss. Low-density lipoprotein subclasses and risk of coronary artery disease. *Current Opinions in Lipidemiology*, 2:248–252, 1991.

[67] V. A. McKusick. *Mendelian inheritance in man. Calatogus of autosomal dominant, autosomal recessive and X-linked phenotype.* The Johns Hopkins University Press, Baltimore, 1990.

[68] Charles E. Metz. Basic principles of ROC analysis. *Seminars in Nuclear Medicine*, VIII(4), October 1978.

[69] R. S. Michalski. An experimental comparison of several many-valued logic inference techniques in the context of computer diagnosis of soybean diseases. *International Journal of Man-Machine Studies*, 1981.

[70] R. S. Michalski and R. L. Chilausky. Learning by being told and learning from examples: An experimental comparison of the two methods of knowledge acquisition in the context of developing an expert system for soybean disease diagnosis. *International Journal of Policy Analysis and Information Systems*, 4(2):125–161, 1980.

[71] John Mingers. An empirical comparison of selection measures for decision-tree induction. *Machine Learning*, 3:319–342, 1989.

[72] Patricia P. Moll, Charles F. Sing, William H. Weidman, Hymie Gordon, Ralph D. Ellefson, Patricia A. Hodgson, and Bruce A. Kottke. Total cholesterol and lipoproteins in school children: Prediction of coronary heart disease in adult relatives. *Circulation*, 67(1):127–134, January 1983.

[73] Gregoire Nicolis and Ilya Prigogine. *Exploring Complexity: An Introduction.* W. H. Freeman and Company, New York, 1989.

[74] J. J. Nora, R. H. Lortscher, R. D. Spangler, A. H. Nora, and W. J. Kimberling. Genetic-epidemiology study of early onset ischemic heart disease. *Circulation*, 61:503–8, 1980.

[75] Steven W. Norton and Haym Hirsh. Learning DNF via probabilistic evidence combination. In *Proceedings of the Tenth International Conference on Machine Learning, Amherst, MA*, pages 220–227. Morgan Kaufmann, San Mateo, CA, 1993.

[76] Pathobiological Determinants of Atherosclerosis in Youth (PDAY) Research Group. Natural history of aortic and coronary atherosclerotic lesions in youth: Findings from the PDAY study. *Arteriosclerosis and Thrombosis*, 13(9):1291–1298, September 1993.

[77] N. H. Packard. A genetic learning algorithm for the analysis of complex data. *Complex Systems*, 4:543–572, 1990.

[78] G. Pagallo and D. Haussler. Feature discovery in empirical learning. Technical Report UCSC-CRL-90-27, University of California at Santa Cruz, Santa Cruz, CA, 1990.

[79] Michael Pazzani. Personal communication (email), 1994.

[80] Michael Pazzani, Christopher Merz, Patrick Murphy, Kamal Ali, Timothy Hume, and Clifford Brunk. Reducing misclassification costs. In *Proceedings of the Eleventh International Conference on Machine Learning, New Brunswick, NJ*, pages 217–225. Morgan Kaufmann, San Francisco, 1994.

[81] L. Perusse, P. P. Moll, and C. F. Sing. Evidence that a single gene with gender- and age-dependent effects influences systolic blood pressure determination in a population-based sample. *American Journal of Human Genetics*, 49:94–105, 1991.

[82] R. L. Phillips, A. M. Lilienfeld, E. L. Diamond, and A. Kagan. Frequency of coronary heart disease and cerebrovascular accidents in parents and sons of coronary heart disease index cases and controls. *American Journal of Epidemiology*, 100:87–100, 1974.

[83] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[84] John Rachlin, Simon Kasif, Steven Salzberg, and David Aha. Towards a better understanding of memory-based reasoning systems. In *Proceedings of the Eleventh International Conference on Machine Learning (New Brunswick, NJ)*, pages 242–250. Morgan Kaufmann, San Francisco, 1994.

[85] A. M. Rissanen. Familial occurrence of coronary disease: Effect of age at diagnosis. *American Journal of Cardiology*, 44:60–6, 1979.

[86] S. N. Salthe. *Evolving Hierarchical Systems.* Columbia University Press, New York, 1985.

[87] J. S. Schlimmer. *Concept Acquisition Through Representational Adjustment.* PhD thesis, Department of Information and Computer Science, University of California, Irvine, 1987.

[88] Jeffrey Schlimmer. Personal communication (email), 1994.

[89] S. Shea, R. Ottman, C. Gabrieli, Z. Stein, and A Nichols. Family history as an independent risk factor for coronary artery disease. *J Am Coll Cardiol*, 4:93–101, 1984.

[90] R. I. Sholtz, R. H. Rosenman, and R. J. Brand. The relationship of reported parental history to the incidence of coronary heart disease in the western collaborative group study. *American Journal of Epidemiology*, 102:350–356, 1975.

[91] R. Sikora and M. J. Shaw. A distributed problem-solving approach to inductive learning. Technical Report Faculty Working Paper No. 91-0109, University of Illinois at Urbana-Champaign, College of Commerce and Business Administration, 1991.

[92] Herbert A. Simon. The architecture of complexity. *Proceedings of the American Philosophical Society*, 106:467–482, 1962.

[93] C. F. Sing and P. P. Moll. Genetics of atherosclerosis. *Annual Review of Genetics*, 24:171–87, 1990.

[94] C. F. Sing and S. L. Reilly. Genetics of common diseases that aggregate, but do not segregate, in families. In C. F. Sing and C. L. Hanis, editors, *Genetics of cellular, individual, family, and population variability.* Oxford University Press, New York, 1993.

[95] Charles F. Sing, Martha B. Haviland, Alan R. Templeton, Kim E. Zerba, and Sharon L. Reilly. Biologial complexity and strategies for finding DNA variations responsible for inter-individual variation in risk of a common chronic disease, coronary artery disease. *Annals of Medicine*, 24:539–547, 1992.

[96] Charles F. Sing, Kim E. Zerba, and Sharon L. Reilly. Traversing the biological complexity in the hierarchy between genome and CAD endpoints in the population at large. *Clinical Genetics*, 44, 1993.

[97] David B. Skalak. Prototype and feature selection by sampling and random mutation hill climbing algorithms. In *Proceedings of the Eleventh International Conference on Machine Learning, New Brunswick, NJ*, pages 293–301. Morgan Kaufmann, San Francisco, 1994.

[98] J. Slack and K.A. Evans. The increased risk of death from ischaemic heart disease in first degree relatives of 121 men and 96 women with ischaemic heart disease. *J Med Genet*, 3:239–57, 1966.

[99] R. E. Smith, S. Forrest, and A. S. Perelson. Searching for diverse, cooperative populations with genetic algorithms. *Evolutionary Computation*, 1(2):127–149, 1993. An earlier version of this paper appeared as Technical Report number LA-UR-92-947 from Los Alamos National Laboratories (1992).

[100] Sathanur R. Srinivasan, Gosta H. Dahlen, Rafael A. Jarpa, Larry S. Webber, and Gerald S. Berenson. Racial (black-white) differences in serum lipoprotein (a) distribution and its relation to parental myocardial infarction in children. *Circulation*, 84(1):160–167, July 1991.

[101] R. C. Strohman. Ancient genomes, wise bodies, unhealthy people: Limits of a genetic paradigm in biology and medicine. *Perspect Biol Med*, 1993, in press.

[102] M. Tan and L. Eshelman. Using weighted networks to represent classification knowledge in noisy domains. In *Proceedings of the Fifth International Conference on Machine Learning, Ann Arbor, MI*, pages 121–134. Morgan Kaufmann, San Mateo, CA, 1988.

[103] Leo P. TenKate, Helge Boman, Stephen P. Daiger, and Arno G. Motulsky. Familial aggregation of coronary heart disease and its relation to known genetic risk factors. *The American Journal of Cardiology*, 50:945–953, Novemeber 1982.

[104] S. T. Turner, W. H. Weidman, V.V. Michels, T. J. Reed, C. L. Ormson, T. Fuller, and C. F. Sing. Distribution of sodium-lithium countertransport and blood pressure in caucasians five to eighty-nine years of age. *Hypertension*, 13:378–391, 1989.

[105] The UCI repository of machine learning databases. Machine-readable data repository. Available via anonymous ftp to ics.uci.edu, in directory pub/machine-learning-databases.

[106] M. Mitchell Waldrop. *Complexity: The Emerging Science at the Edge of Order and Chaos.* Simon & Schuster, New York, 1992.

[107] *Webster's New World Dictionary of the American Language, Second College Edition*, 1972.

[108] Sholom M. Weiss and Casimir A Kulikowski. *Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning and Expert Systems.* Morgan Kaufmann, San Mateo, CA, 1991.

[109] David E. L. Wilcken, Xing Li Wang, Jennifer Greenwood, and Judith Lynch. Lipoprotein(a) and apolipoproteins B and A-1 in children and coronary events in their grandparents. *The Journal of Pediatrics*, 123(4):519–526, October 1993.

[110] Jarryl Wirth and Jason Catlett. Experiments on the costs and benefits of windowing in ID3. In *Proceedings of the Fifth International Conference on Machine Learning, Ann Arbor, MI*, pages 87–99. Morgan Kaufmann, San Mateo, CA, 1988.

[111] P. Zeek. Juvenile arteriosclerosis. *Arch Patho*, 10:417–446, 1930.