BEBR
FACULTY WORKING
PAPER NO. 90-1693

A Double-Layered Learning Approach to Acquiring Rules for Financial Classification

Riyaz Sikora
Michael J. Shaw

College of Commerce and Business Administration
Bureau of Economic and Business Research
University of Illinois Urbana-Champaign

BEBR

FACULTY WORKING PAPER NO. 90-1693

College of Commerce and Business Administration

University of Illinois at Urbana-Champaign

September 1990

A Double-Layered Learning Approach to Acquiring Rules for Financial Classification

Riyaz Sikora
Michael J. Shaw

Department of Business Administration
and
Beckman Institute

University of Illinois

Abstract

In this paper we describe a new machine learning approach, based on a double-layered architecture and the Genetic Algorithms(GAs), to learning decision rules for financial classification. These rules can be implemented in an expert system for future consultation in the particular application area. GAs represent a new class of learning algorithms based on the model of the biological evolution process. Equipped with unique search behavior and solution-seeking properties, GAs provide an interesting new technique for such financial-classification tasks as bankruptcy prediction and credit analysis. However, some modifications to the basic GAs would be necessary in order to make the method suitable for solving the financial-classification problems. One of the objectives of this paper is to identify the aspects of GAs that need to be modified for the classification domain and how they can best be incorporated in the GAs. More importantly, we expand on the concept of GA and develop a learning method called Double-layered Learning System(DLS) that integrates GA with a similarity-based learning technique called Probabilistic Learning Systems(PLS1). DLS proves to be an effective improvement over both GA and PLS1. An analysis is included to evaluate the performance of DLS in terms of computational efficiency, prediction accuracy, conciseness of the concepts generated, and rule refinement.

1. Introduction

Machine learning methods can help automate the knowledge-acquisition process in building expert systems(Buchanan[1989]). In this paper we describe a hybrid approach combining the genetic algorithms and probabilistic learning to the problem of learning decision rules for financial classification. The problems considered can be described by the following decision task: Based on a set of financial attributes and qualitative information, assess a firm's financial performance and risks, and determine its ordinally ranked credit risk category. Examples of financial applications falling into this class of problems include bankruptcy prediction, bond rating, default prediction, and commercial loan classification.

GAs represent a new class of algorithms based on the model of the biological evolution process. Equipped with unique search behavior and solution-seeking properties, GA provides an interesting new technique for solving the financial-classification problems. It is also a rule-induction technique for acquiring classification knowledge(Holland et.al.[1986]). However, some modifications to the basic GA would be necessary in order to make the method more effective. One of the objectives of this paper is to identify the aspects of GA that need to be modified for the classification domain and how they can best be incorporated in the GA for rule learning.

In addition, some similarity-based learning techniques developed by artificial intelligence researchers recently, such as the tree induction method by Quinlan[1986] and the probabilistic learning method by Rendell[1986] have been applied to the same classification problem domain and have shown very promising results. These methods are non-parametric and have demonstrated many advantages as classification tools over the statistical methods traditionally used in this area. Buchanan[1989], for instance, pointed out that the traditional statistical techniques are " knowledge poor" procedures that are unable to use knowledge we may bring to the learning task. In addition, Currim et al. [1989], Carter & Cartlett[1987], and Hansen and Messier[1988] have shown the superiority of the tree induction method, ID3, over statistical methods. Shaw & Gentry[1989], Rendell et.al.[1989] have shown the comparable performance of PLS1.

Methods for similarity-based learning fall into two categories (Rendell[1990]):
(1) those which use the data to instantiate a parameterized model defined over *instance-space* (I-space)[1]. Thus the problem is to fit data to some model of the function; and
(2) those which use the data to select a candidate concept defined over *hypothesis-space* (H-space)[2]. Thus the problem here is to optimize some measure of hypothesis quality. Algorithms like PLS1(appendix A) and ID3(Quinlan[1986]) are of the first category whereas GA is of the second type. I-space algorithms are generally fast whereas H-space

algorithms are slow but stable. Though the structure of the H-space depends on the I-space, I-space algorithms do not fare well if the function in H-space is not unimodal (i.e., if it has multiple peaks). Also, algorithms like PLS1 are not stable if there is interaction among different attributes (feature interaction (Rendell[1983])). The data, and the training examples, in the financial-classification domain is characterized by high feature interaction (for example, increase in sales has a effect on profits), multiple peaks in I-space(for example, there might be several reasons for a company to go bankrupt), and lack of any underlying distribution. GA, a H-space method, does not have the above mentioned problems of dealing with these characteristics and thus is particularly appealing for financial classification .

To evaluate GA as a methodology for learning financial-classification rules, we compare our GA with two of the more successful similarity-based learning algorithms, Probabilistic Learning System (PLS1) and tree-induction method, ID3, and demonstrate the advantages of GA. However, GA is computationally slow compared with PLS1 or ID3. To overcome this shortcoming, we develop a new learning technique, referred to as the *double layered learning system*, for improving the performance of solving the financial-classification problems.

The Double-layered Learning System(DLS)[3] combines PLS1 and GA together and therefore incorporates the features of both types of methods(I-space and H-space algorithms). Empirical analysis confirms that DLS produces better classification results than PLS1 or GA. The hybrid technique points to a very promising direction for developing rule learning algorithms for financial classification.

We base our empirical analysis of the DLS on three different real world data sets from the financial domain (see appendix B for details): (1)Bankruptcy data set is from the *Standard and Poor' Compustat Industrial Annual Research file of Companies* and *Compustat Industrial files* , and is used to predict bankruptcy of firms; (2)Default loan data set (Abdel-Khalik and El-Sheshai[1980]) is used to classify a set of firms into those that would default and those that wouldn't default on loan payments; and (3)Loan data set consists information about loan risk classification of different firms from a regional bank, with a rank of 1 being for the lowest risk level and a rank of 5 for the highest risk level. With these different sets of financial data we hope to obtain consistency in evaluating the learning techniques under study.

From the empirical studies carried out to evaluate the performance of DLS, the following findings come to light: (1) DLS has the effect of *rule refinement* over the rules generated by PLS1 and GA, (2) the concepts generated by DLS are more concise; (3) it improves the prediction accuracy of the rules generated, over that of PLS1 and GA; and (4)

it improves the efficiency of GA because of the good initial population, thanks to PLS1. Thus the findings confirm that DLS, a hybrid technique, combines the advantages of both PLS1 and GA.

The organization of the paper is as follows: In the next section we give a brief overview of the GA and discuss some of its important properties; in § 3 we discuss concept representation with a GA; in § 4 we present the GA for classification problem with the appropriate modifications; in § 5 we illustrate the search process of the GA; in § 6 the empirical results and comparisons of the GA with ID3 and PLS1 are illustrated; § 7 describes the double layered learning system combining GA and PLS1; in § 8, two examples are used to demonstrate the working of DLS; and finally, in § 9, the empirical results of DLS are discussed.

2. Genetic Algorithm

2.1 Review

The learning of classification rules can be thought of as searching through the space of rules (or hypothesis) and thus has all the problems associated with a search problem, mainly the combinatorial explosion of the search space with the increase in the number of dimensions of instance space (i.e., number of attributes).

An efficient way of tackling the problem will be the use of a search algorithm which can simultaneously search different regions of the search space, in parallel, instead of the traditional algorithms which follow a single search path in the search space. Such an algorithm is more likely to find a global optimum if there are many peaks in the hypothesis space. This has important implication for the classification problem in terms of the prediction accuracy of the rules or hypothesis found. The traditional methods for classification perform very well in classifying the training data set but perform not so well on testing data, in other words they usually locate a local maxima if the training instances happen to be from around the local maxima.

Genetic Algorithms (GAs) are adaptive search algorithms which have the above desired properties of parallel search and ability to locate global maxima without getting trapped in local maxima. They represent a class of general purpose adaptive search techniques which have been used in a wide range of optimization problems. Goldberg(1989), describes GA as search algorithms based on the mechanics of natural selection and natural genetics. They combine survival of the fittest among string structures with a structured yet randomized information exchange to form a search algorithm with some of the flair of human search. In

every generation a new set of artificial creatures(strings) is created using bits and pieces of the fittest of the old; an occasional new part is tried for good measure. While randomized, genetic algorithms are no simple random walk, they efficiently exploit historical information to speculate on new search points with expected improved performance.

A GA should be equipped with the following four components for achieving the effect of rule learning:

(1) a chromosomal representation of solution to the problem.
(2) a way to create an initial population of solutions.
(3) an evaluation function that rates the solutions in terms of their "fitness".
(4) genetic operators that alter the composition of solutions during reproduction

In addition, in applying GA, one needs to decide the various values for the parameters that the genetic algorithm uses, such as the population size, the number of generations, and the probability of mutations. As will be seen later in this paper, varying the mutation rate greatly enhances the GA in terms of avoiding a local optimum and facilitating the convergence to a good solution.

Since the GA works with string structures (analogous to *chromosomes* in biological systems), the **hypothesis** (or rules or solutions) should be encoded and represented in a string form. This low level representation with which a GA works is called *genotype* , and the corresponding set of apparent characteristics is called *phenotype*. The individual elements of the genotype are called *genes* , and their possible values are *alleles*.

The GA's work with a population of hypothesis at a time, the number of hypothesis being a parameter of choice. Each hypothesis is evaluated using the training examples and a "fitness" score, usually measuring the accuracy of the hypothesis, is assigned to it. Starting from an initial population of hypothesis, the GA exploits the information contained in the present population and explores new hypothesis (abduction) by generating a new population of hypothesis from the old population through application of genetic operators. The genetic operators most often used are: (a) reproduction; (b) crossover, and (c) mutation.

The **reproduction** operator just duplicates the members of the population to be used to derive new members. The number of copies that each member (hypothesis) gets is proportional to its fitness score. Thus the fitness of an individual is clearly related to its influence upon its future development of the population. When many offspring of a given individual survive to reproduce, then many members of the resulting population, the "next

generation," will carry the alleles of that individual. Genotypes and phenotypes of the next generation will be influenced accordingly.

After reproduction, new individuals are generated by selecting two individuals at a time from the resulting population and applying the operator of **crossover**. Crossover exchanges the genes between the two selected individuals(parents) to form two different individuals. Crossover is the key to the power of the GAs as it helps in combining information from different hypothesis to discover more useful hypothesis. Usually crossover is applied with a constant probability P_c.

The **mutation** operator randomly changes some of the genes in a selected individual and is applied at a much lower rate(P_m) as compared to crossover operator (i.e., $P_m << P_c$) . The basic GA can be described by the following procedure:

PROCEDURE GA (population size n, max. number of generations Ng)

```
begin;
select an initial population of n genotypes {g};
no-of-generations = 0;
repeat;
        for each member b of the population;
                compute f( g), the fitness measure for each member; /* evaluation */
        repeat;
                stochastically select a pair of genotypes g1, g2
                        with probability increasing with their fitness f; /* reproduction */
                using the genotype representation of g1 and g2,
                        mutate a random bit with probability pu; /* mutation */
                randomly select a crossover point and
                        perform  crossover on g1 and g2 to
                         give new genotypes g'1 and g'2; /* crossover */
        until the new population is filled with n individuals g'i;
        no-of-generations = no-of-generation + 1;
until the number-of-generations has reached Ng or
        one of the genotype is good enough; /* termination */
end;
```

While most GAs would have this basic form, there are a variety of ways this procedure can be implemented. Some of the implementation issues are described in §4, in which the methods to specify the evaluation function, the sampling procedure for reproduction, the probability for mutation, and the crossover mechanism are discussed. It is important to note at this point that the specification of these parameters and mechanisms can affect the performance of a GA.

2.2 Properties of Genetic Algorithms

A GA incorporates the search procedure whose mechanics are based on those of natural genetic evolution. Fundamentally, a GA has features that make it robust and globally oriented search procedure. Mathematical analysis reveals that the GA's search power is due to computational leverage provided by the underlying process of schemata, or similarity templates(Goldberg[1989]). *Schemata* are substructures of parameter codings, and they describe subspaces of the coding space. Under the action of reproduction, schemata that are associated with high fitness receive exponentially increasing number of copies in a population. Crossover combines certain schemata in a randomized, yet structured way, such that new schemata can be evaluated.

Thus, at each step the GA faces the twin processes of ***exploitation*** vs. ***exploration.*** Exploitation refers to the process of exploiting the information (in the form of schemata) which the population has acquired. Typically, exploitation is carried out by reproduction operator by making copies of the coded structures(individuals) in proportion to their fitness. Exploration, on the other hand, refers to the process of randomly exploring new points in the search space. It is typically carried out by random mutations. Crossover operator is the unique operator which incorporates both exploitation and exploration, the exact tradeoff between the two depending on the type of crossover operator used. Eshelman et.al.(1989) discuss this tradeoff in the form of biases for different crossover operators, and suggest the use of a variety of crossover operators, such as the uniform crossover operator or the multiple-point crossover operator.

Exploitation and exploration together can explain some of the important properties of a GA. In the process of searching for the solution, the exploitation step helps put more emphasis on the more promising coded structures; the exploration step helps generate new candidates in the search space. There is a loss associated with either exploitation or exploration. If the association of a particular schema with high fitness is due to happenstance, then exploiting this association by giving increasing samples to the schema will lead to a loss of performance. On the other hand favoring exploitation by random mutations leads to loss of important information contained in the population. Thus, there should be an optimal balance between exploitation and exploration which minimizes these loses. Holland(1975) formalized this optimal balance by minimizing the expected loss due to either exploration or exploitation. He showed that "**Minimum expected loss**" was a more useful criterion of judging the algorithm than "convergence". In other words, A GA can achieve good solution quality by fully taking advantages of both the exploitation and the exploration steps.

GA has a number of algorithmic properties that may explain its ability to balance between exploitation and exploration. First, the expected number of trials given to the j^{th} value of an i^{th} attribute(for all i & j) is infinite. This ensures that every point in the search space is likely to be reached by the GA. Second, in executing a GA, for any schema, the number of copies given to a schema by the GA is an exponential function of its fitness. Third, the solution generated by a GA has minimized expected loss with respect to the opportunity cost related to exploration and exploitation. Fundamentally, a GA has the following properties that makes it computationally attractive as a classification technique.

(i) **Implicit Parallelism**: A GA works on a population of hypotheses instead of one at a time.It can be shown that in a generation the GA processes about n^3 schemata information(where n in the population size). As a result, GA is better able to search through large search space. Because they implement parallel search, GAs can better handle badly behaved objective functions.

(ii) **Robustness**: GAs operates on codings of parameters, rather than the parameters themselves. GAs search in the hypotheses space in the learning process, as opposed to in the instance space(Rendell[1990]). A GA is free from assumptions of other methods such as continuity, existence of derivative, unimodality, or any distribution. Since a GA use the fitness measure to guide the search in the hypotheses space, it has more tolerance for noise in the data and changes in the environment.

(iii) **Ability to perform global search**. Because of the use of genetic operators, GAs search around the hypotheses space not by following any functional form, gradient information, or heuristics, but by exploration and exploitation. This means more globally oriented search and less likelihood of getting stuck in local minimum. For example, De Jong(1975) and Caruana & Schaffer(1989) used several functions to test the GA's performance on search spaces. GA was able to find the global optimum in all the cases, although several other methods failed to find the global optimum on the same data.

(iv) **Conciseness**: Partly because of the searching advantages of a GA, it usually generates more concise concept descriptions. This is further confirmed by our empirical study.

(v) **Ability to learn concepts fast in high-dimension sparse spaces**: Many problems, including the financial classification problems, have the characteristics that it is

not known before hand which of the attributes considered are relevant for the concept being learned. If the concept is of a vary low dimension in a high dimension hypothesis space, then traditional similarity-based learning methods would waste lot of time in a high dimension space(Syswerda[1989]). GAs, on the other hand, can handle this problem nicely with their unique parallel search capability.

3. Knowledge Representation in Genetic Algorithms

3.1 Concept Representation

Since in this paper we are developing a GA for the purpose of performing classification tasks, which can in general be viewed as the **concept-learning** problem, we shall now first set the stage by addressing the representation issues related to concept learning and how they can be handled by the GA using its genetic representation.

A concept is either a categorical or an uncertain classification rule, corresponding to a binary or probabilistic class membership function over a space(i.e., the instance space) defined by a set of attributes. Given the range of attribute values and function values, we can express the correct concept as one of many candidates or *hypotheses*(Rendell[1990]). For a program to perform concept learning, it must be able able to formulate the hypotheses of the correct concept.

A concept can be logically represented as

$$C = C_1 \vee C_2 \vee \ldots \vee C_p$$
$$= (\xi_{1,1} \,\&\, \ldots \,\&\, \xi_{k_1,1}) \vee (\xi_{1,2} \,\&\ldots\, \&\, \xi_{k_2,2}) \vee \ldots \vee (\xi_{1,p} \,\&\, \ldots \,\&\, \xi_{k_p,p}),$$

where ξ corresponds to a proposition that specifies conditions for a particular coordinate, { $x_i = a_1$ or $x_i = a_2$ or ...}. The following example from financial domain will help explain the above representation:

Consider the concept C:

[(NET_INCOME TO TOTAL_ASSETS RATIO ≤ 0.048) and (TOTAL_DEBT TO TOTAL_ASSETS RATIO ≥ 0.534) OR
(TOTAL_DEBT TO TOTAL_ASSETS RATIO ≥ 0.34) and (CASH_FLOW TO TOTAL_DEBT RATIO ≥ 0.395)]

In terms of the above representation p=2, k_1=2, and k_2=2. The concept is thus in the form $C = C_1 \vee C_2$, where

$C_1 = \xi_{1,1} \,\&\, \xi_{2,1}$,

$C_2 = \xi_{1,2}$ & $\xi_{2,2}$,

$\xi_{1,1}$ = (NET_INCOME TO TOTAL_ASSETS RATIO ≤ 0.048) ,

$\xi_{2,1}$ = (TOTAL_DEBT TO TOTAL_ASSETS RATIO ≥ 0.534) ,

$\xi_{1,2}$ = (TOTAL_DEBT TO TOTAL_ASSETS RATIO ≥ 0.34) , and

$\xi_{2,2}$ = (CASH_FLOW TO TOTAL_DEBT RATIO ≥ 0.395)

This concept description is referred to as the *disjunctive normal form* . It is known that in mathematical logic that all possible expressions involving the elementary propositions are tautologically equivalent to one in disjunctive normal form. Let X_C represents the set of data points that satisfy C. Geometrically, all possible conditional sets X_C using conditions of the disjunctive normal form correspond to all possible subsets of the space of attributes $x_1 ... x_n$. Packard[1989] has a very good discussion on the logic and geometry related to concept representation.

3.2 Concept Representation with GA

To use the GA, we must be able to represent the domain knowledge by a vector called a *genotype*. Packard(1989) describes a method to represent concept descriptions by genotypes in a GA. Each of the genotypes is allowed to take on either a value of *, indicating no condition is set for the corresponding attribute, or a sequence of numbers $(c_1,..., c_k)$ indicating values for the corresponding attributes in a disjunctive term. For example, (*, *, (4,8), *, 5) is equivalent to the conditional set $X_C = \{x \mid (x_3 = 4$, or $x_3 = 8)$ & $(x_5 = 5)\}$.

Geometrically the set X_C for such conditions is a set of rectangles, instead of the more general disjunctive normal form. Alteration of the GA to conditions of disjunctive normal form would require a similar symbolic representation.

For example, a concept consisting of two variables : $(1 < x_1 < 3)$ & $(1.3 < x_2 < 3.4)$ can be represented by the coded structure, called phenotype as (1 3 1.3 3.4). Its corresponding genotype can be constructed by mapping the phenotype into binary form. The above concept can be similarly constructed to include the disjunctive forms, such as

(1 3 1.3 3.4 2 4 5.2 7.5)

This coded structure is equivalent to the following disjunctive form:

$[(1 < x_1 < 3)$ & $(1.3 < x_2 < 3.4)]$ v $[(2 < x_1 < 4)$ & $(5.2 < x_2 < 7.5)]$

4. GA for learning Classification Rules

4.1 Introduction

Classification problem concerns itself with finding of rules for pattern discrimination from examples of correctly classified patterns. The data or examples are of the form

$$(\underline{X}_i, y_i)$$

where $\underline{X}_i$ is a vector of independent variables $(x_1\ x_2 \ldots x_n)$ and y_i is the corresponding classification variable. The patterns learned from the examples will be of the form (C_i, y_i) where C_i is the conditions (the IF part of the IF-THEN rule) or the patterns classifying the variable y_i. C_i can also be viewed as the concept description correspond to y_i.The basic tool for discerning the patterns C_i from the data will be the GA here. The specific problem being considered in this paper is the class of financial-classification problems, where $\underline{X}_i$'s are the different types of financial variables of a company and y_i's are the corresponding credit risk classification of the company.

4.2 Representation

The conditions(hypothesis) are represented in the disjunctive normal form (DNF) as

$$C = (C_1 \vee C_2 \vee \ldots \vee C_p)$$

where each disjunct is a conjunction of conditions .

$$\text{i.e.,} \quad C_i = (\xi_{1,i} \,\&\ldots\& \, \xi_{k_i,i})$$

In order to handle *continuous variables* each condition $\xi_{j,i}$ will be in the form of a closed interval $[a_i\ b_i]$, i.e.,

$$\xi_{j,i} = (a_i \le X_i \le b_i) \quad \text{, if } a_i \le b_i$$
$$\text{or } (a_i \ge X_i \ge b_i) \quad \text{, if } a_i \ge b_i$$

The representation with which the GA works (genotype) is obtained by mapping the phenotype of $C_i = (\xi_{1,i} \,\&\ldots\& \, \xi_{k_i,i})$ into binary form. For example, if $k_i=2$ and $\xi_{1,i}$ = [1 3] $\xi_{2,i}$ = [4 2], then the representation of C_i is

((1 3) (4 2)) ---> phenotype

(0 0 1 0 1 1 1 0 0 0 1 0) ---> genotype,

(assuming each number is represented by a 3 bit binary number).

There are two ways of deriving the rules using a GA. One way is to let each member in the population represent a complete concept $(C_1 \vee C_2 \vee \ldots \vee C_p)$.
Alternatively, the other method is to let each member be just a single disjunct $C_i = (\xi_{1,i} \,\&\ldots\& \, \xi_{k_i,i})$. We use the second method in our GA because, unlike the first method, it does not have to assume the knowledge of the number of disjuncts a priori. In this method,

the GA tries to find the best possible hypothesis(or disjunct) (i.e., covering as many positive examples as possible). After it converges, the best hypothesis found is retained and the positive examples which it covers are removed. The process is again repeated to find a new hypothesis to cover as many of the remaining positive instances as possible. This process terminates after all the positive instances are covered. The final rule is then the disjunct of all the hypothesis found. This procedure of searching the instance space (I-space) is called *explanation based filtering* (Cohen & Feigenbaum[1982]).

4.3 Evaluation Function

We first present the theoretical foundation leading to the fitness function which has been used for Classification using GA (Packard[1989]), point out its drawbacks in present context, and then present a fitness function which has the desired properties.

The fitness function which evaluates the fitness of each member of the population should be based on the amount of the information about classification contained in it. Communication theory provides us with a precise measure of information called *entropy*. Applying this concept to the classification problem we find that if an object can be classified into one of the several different groups, the entropy is a measure of the uncertainty of the classification of that object. As the entropy increases, the amount of information that we gain by knowledge of the final classification increases. Mathematically, if an object can be classified into N different classes C_1, C_2, ... C_N and the probability of an object in class i is $p(C_i)$ then the entropy of classification, H(C) is

$$H(C) = -\sum_{i=1}^{N} p(C_i) \log_2 p(C_i)$$

To evaluate the fitness of any condition we should first get the conditional probability distribution of y values. Let C be a condition (individual in the population), then the conditional probability distribution of y values given that $\underline{X} \varepsilon C$ is

$$P_c(y) = \frac{1}{N_c} \sum_{\{\underline{X} \in C\}} \delta(y - y')$$

where N_C = number of training examples which satisfy C

and $\delta(y\text{-}y') = 1$ if the class of the training example matches that of the condition

$= 0$ else

We may now use $P_C(y)$ to evaluate the usefulness of the conditions in specifying y. This evaluation will allow us to assign a "value" or "fitness" of any condition C. An appropriate fitness measure is the *Kullback-Leibler function* (Packard[1989])

$$F(C) = d(P_c, P) = \sum_y P_c(y) \log_2 \left(\frac{P_c(y)}{P(y)}\right) = H_{max} - H(P_c)$$

where $P(y) = \frac{1}{K_D}$, $y \in \{1,2...K_D\}$

$H(P_c)$ is the entropy of the distribution $P_c(y)$

and $H_{max} = \frac{1}{K_D}$

For bankruptcy prediction, there are only two classes, positive and negative; in this case y is binary and can take only two values for any C, i.e.,

$$y \in \{p, n\}$$

Therefore, P(y) = 0.5, and

$$F(C) = P_c(p) \log_2 2P_c(p) + P_c(n) \log_2 2P_c(n) = P_c(p) \log_2 \left(\frac{P_c(p)}{(1 - P_c(p))}\right) + \log_2 2(1 - P_c(p))$$

where $P_c(p) = 1 - P_c(n) = \frac{p}{(p+n)}$, the accuracy of the condition C.

Figure 4.1(a) shows the plot of F(C) vs. $P_C(p)$.

Insert Figure 4.1 (a) Here

The desired condition C is the one which covers all the positive instances without covering any negative, i.e., one for which $P_C(p)$ is as close to 1 as possible. We see that F(C) is symmetric about $P_C(p)$=0.5 i.e., points X_1 and X_2 have the same fitness even though X_2 is more desirable than X_1, such ambiguity would create problems during the evolution process in selecting the fitter members. Thus the fitness function needs to be modified to have a monotonously increasing functional form, such as the one shown in Figure 4.1(b). To that end, a simplified version of F(C) is considered:

$$F'(C) = \frac{P_c^2(p)}{(1 - P_c(p))}$$

Figure 4.1(c) shows the plot of F'(C) Vs. $P_C(p)$

In addition, to take into account the poor statistics resulting from the dependence of the fitness function on the number of positive examples covered, we let

$$F''(C) = F'(C) / P, \text{ where P = total number of positive examples}$$

Normalizing the above function so as to have a maximum value of 1 when all the positive examples are covered without any negative examples, we get the final function $F_d(C)$, which is the one shown in Figure 4.1(b).

A close look at the function $F_d(C)$ reveals the following relationship:

$$F_d(C) = \left(\frac{p}{(p+n)}\right)\left(\frac{p}{P}\right)\left(\frac{1}{n}\right) \quad \text{if } n > 0$$
$$= \frac{p}{P} \quad \text{if } n = 0$$

The term (p / (p+n)) is just the accuracy of the condition C and is similar to the utility of the "regions' considered by PLS1, (p / P) is the reliability of the accuracy and is similar to the error term used in PLS1, and, (1 / n) is a penalty term for covering negative examples.

4.4 Modifications to the basic GA

In recent years there has been much work done studying the alternative ways to specify the parameters as well as to incorporate the mechanisms involved in implementing GAs. In the particular method we developed, we incorporated some of these modifications to the basic GA, each of them have been proven empirically to enhance the performance of GAs. Specifically, the major modifications include: (1) using Baker's (1987) stochastic universal sampling technique in the reproduction process; (2) adopting a new crossover operator, called uniform crossover, which is a more general, as well as more efficient, way to perform crossovers; and (3) using a varying mutation probability that approaches zero toward the end of the genetic process. The explanations of the three modifications follow.

1) Universal stochastic sampling:

The main culprit in terms of time complexity for a GA is the selection, or the sampling procedure, used in the application of the reproduction operator. The expected number of copies that each individual receives is proportional to its fitness. Specifically, the expected number of copies that individual i gets at time t is $\mu_i(t)/\underline{\mu}(t)$ where $\mu_i(t)$ is the observed fitness of individual i at time t and $\underline{\mu}(t)$ is the average fitness of the population at time t.

Traditional sampling techniques use the so-called "spinning wheel" method, wherein each wheel slice is proportional in size to some individual's expected value. The wheel has to be spun as many times as the number of samples required. This technique is typically implemented in $O(N^2)$ time, but can be implemented in O(N*logN) by using a B-tree. The traditional techniques fall into one of the following six:

(a) Deterministic sampling,

(b) Stochastic sampling with replacement,

(c) Stochastic sampling without replacement,

(d) Remainder stochastic sampling with replacement,

(e) Remainder stochastic sampling without replacement, and

(f) Stochastic tournament.

Baker(1987), uses three criteria for comparing the various sampling techniques. (1) bias : defined as the absolute difference between an individual's actual sampling probability and his expected value; (2) spread : defined as the range of possible values for the actual number of copies that an individual receives in a given generation; and (3) efficiency : the time complexity of the technique. Ideally, a sampling technique should have zero bias, minimum spread and should not increase the GA's overall time complexity (other phases of GA are O(LN), where L is the length of each individual, and N is the population size). The "spinning wheel" method has zero bias but unlimited spread, i.e., any individual with expected value > 0 could be chosen to fill the entire next population.

Baker(1987), then goes on to propose a sampling technique, namely "**Stochastic Universal Sampling**" (SUS), the only one which has zero bias, minimum spread and has O(N) time complexity. SUS is analogous to spinning wheel with N equally spaced pointers. Hence, a single spin results in N samples. We have used SUS in our GA, and thus the overall time complexity of our GA is O(LN) or better.

(2) <u>New crossover operator</u>:

The traditional crossover operator used is the one point operator, as shown below.

1001:01 --> 1001:10

0110:10 --> 0110:01

This can be extended to a two point (or in general k-point) operator:

10:01:01 --> 10:10:01

01:10:10 --> 01:01:10

These are special cases of a more general operator called *uniform crossover*. Instead of a crossover point it uses a *mask* which determines which bits are to be exchanged.

example. uniform operator:

100101 --> 110000

010101("mask")

011010 --> 001111

Syswerda(1989), showed the superiority (with respect to finding good solutions) of uniform crossover operator over one and two point traditional crossover operator on some experimental functions. We also use uniform crossover operator to enhance the performance of the GA.

(3) Varying mutation rate:

Instead of keeping the probability of mutation constant across the generations, we decreased the probability starting from an initial value to 0 uniformly over the total number of generations. Specifically the probability of mutation at time(generation) t is determined by(Holland[1975]):

$$P_m(t) = C_t.{}^0P_m$$

where,

$P_m(t)$ = probability of mutation at time t,

${}^0P_m$ = initial probability of mutation, and

C_t - a sequence of numbers satisfying the following:

(1) $0 < C_t < 1$

(2) $C_t \dashrightarrow 0$ as $t \dashrightarrow \infty$

(3) $\sum_t C_t \dashrightarrow \infty$ as $t \dashrightarrow \infty$

In our case, C_t was $(1-t / M)$ where M is a large constant (usually the limit of t). Fogarty(1989), showed empirical evidence (albeit for a particular application) that varying probability of mutation improved the performance of the GA significantly. This is verified in §5.3.

5. Search Process of the GA: an Example

5.1 Applying GA to the Classification Problems

We consider two separate data sets to demonstrate the search process of the GA. First is the default loan data set and the second is bankruptcy data set (appendix B). The detailed search process of the GA will be demonstrated for the bankruptcy data.

Default Loan Classification

The data set had 32 instances (16 positive and 16 negative) and 18 attributes. The final concept given by the GA had rule size of 3 (i.e., three disjuncts), as shown in Table 5.1.

Insert Table 5.1 Here

Thus the concept learned is (C_1 v C_2 v C_3). The above concept can be interpreted as the following rules:

C_1:

If [(NET_INCOME TO TOTAL_ASSETS RATIO ≤ 0.048) and (TOTAL_DEBT TO TOTAL_ASSETS RATIO ≥ 0.534) and (-0.357 ≤ LONG_TERM_DEBT TO NET_WORTH_TREND RATIO ≤ 0.894)]

then *'The firm would default on loan payment '*

C_2:

If [(TOTAL_DEBT TO TOTAL_ASSETS RATIO ≥ 0.340) and (CASH_FLOW TO TOTAL_DEBT RATIO ≥ 0.395) and (QUICK_ASSETS TO SALES RATIO ≤ 0.565)]

then *'The firm would default on loan payment '*

C_3:

If [(EARNINGS_TRENDS ≤ 3.669) and (CASH_FLOW TO TOTAL_DEBT_TREND RATIO ≤ -0.077)]

then *'The firm would default on loan payment '*

Bankruptcy Analysis

We consider a sample of instances from the bankruptcy data consisting of 58 training examples (29 positive and 29 negative) to demonstrate the search process. In this example the population size was 100, probability of crossover was 0.7, and the probability of mutation was held constant at 0.01 in one experiment and varied to 0 in other and their results are compared. The final concept given by the GA for these data had rule size of four (i.e., four disjuncts) is shown in Table 5.2. The rule thus learned is r = (C_1 v C_2 v C_3 v C_4). The detailed search process of the GA for the first disjunct C_1 is shown in Table 5.3. Note that this set of data have been transformed into the values in the range [0, 58]. That range was used in the initial population and the only schema in that generation is the the most general description.

The interesting aspect is that as more generations are generated, an important schema emerges. In this case, the schema is (*(3 45)*****(28 37)*), which corresponds to the description [($3 \leq x_2 \leq 45$) & ($28 \leq x_8 \leq 37$)]. This also emphasizes the *building block hypothesis* , a keystone of the genetic algorithm approach, wherein the GA treats *schemata* as the building blocks to the best hypothesis. It searches for better building blocks by giving increasing number of copies to the more fitter schema, and recombining them using the genetic operators. As is seen from the example above, when the GA converges, it has the best schema in all the members of the population.

Insert Tables 5.2 and 5.3 Here

5.2 The Learning process

The initial population used by the GA consists of all general members, each covering the *entire* instance-space. The GA then performs the equivalent of *specialization* on the members through the application of genetic operators. Table 5.3 shows snapshots of the top four members together with their most important schemata and fitness at different generations. The following plot shows the fitness of the top four members as the number of generations grows. The fitness measures converge at around the 60th generation among these four members..

Insert Figure 5.1 Here

Another important parameter judging the learning rate is the on-line performance measured by the *average* fitness of the population at any time t. The following plot shows the average fitness of the population with generations. The average fitness did not converge until at about the 90th generation.

Insert Figure 5.2 Here

5.3 Convergence process

By Figures 5.1 and 5.2 we have seen the convergence behavior of the GA. These experiments present a feasible quantitative measure of the convergence rate of the

algorithm. However, convergence does not guarantee a global optimum solution but can sometimes lead to the "premature" convergence of the algorithm. Among others, varying mutation rate is one way of avoiding premature convergence. As will be shown later, the initial population used would also affect the convergence process.

Convergence to a particular 'good' solution can be measured by the rate of change of the fraction of the population having the best solution. Since obtaining this exact fraction is not feasible(in terms of time complexity) at each generation, one way of measuring it is to look at *the ratio* of average fitness to maximum fitness of the population. The plot in Figure 5.3 shows the such a convergence curve. The convergence of this ratio indicates that there is no more improvement in the fitness measure and the population reaches stability. This is one way to determine the termination of the GA. Another commonly used termination condition is simply prespecify the total number of generations (Goldberg[1989]).

Insert Figure 5.3 Here

5.4 Varying Mutation vs. Constant Mutation Rate

A very interesting finding in applying the GA is that by simply using a varying mutation rate, the learning performance can greatly be improved. To verify the impact of the way the mutation rate is specified, we carried out the same experiment with a constant mutation rate of 0.01 and we compare the results with that of using a varying mutation rate (varying as a function of the generation number) as the one discussed in §4.4. The plots in Figure 5.4 compares the learning rate for both types of methods for specifying the mutation rate.

Insert Figure 5.4 Here

In Figure 5.4 the learning curves are specified in terms of the average fitness level and in terms of the maximum fitness level; the former is referred to as the on-line performance; the latter is referred to as the off-line performance. Varying mutation rate significantly enhances the performance of the GA both in terms of (1) the avoiding a local optimum and (2) converging to a 'good' solution. This can be explained by the fact that as the GA approaches the correct concept description, exploitation becomes more important than exploration in reaching the final solution. As a result, the probability of mutation should be decreased so as not to stray away from the emerging good hypotheses that appear to be fit.

In addition, the use of varying mutation rate would improve the level of fitness for the concept learned. This is because the varying-mutation-probability approach starts with a higher p_m value than the constant-mutation-probability approach - which means more exploration in the beginning, thus better ability to avoid local optimum.

6. Empirical Results

This section describes results of the experimental study carried out to study the relative performance of inductive learning methods ID3 and PLS1 vis-a-vis that of GA with respect to prediction accuracy and conciseness of concept representation as measured by rule size. The data used was bankruptcy data consisting of 104 instances (58 positive and 58 negative). *Cross validation* technique was used by drawing ten different random samples (each consisting of 58 training and 46 testing examples) from the data set and using the three methods to classify them. The following operators and parameter values were used for the GA:

operators: Uniform crossover and varying mutation rate

parameter values: Prob. of crossover = 0.7, initial prob. of mutation = 0.1, and population size = 100.

Table 6.1 shows the results of the 10 experiments for all the 3 methods.

Insert Tables 6.1 Here

A t-test for paired observations was carried out, and at 5% level of significance the difference between the prediction accuracy of GA and that of ID3 and PLS1 was found to be statistically significant. The same test for the rule size showed significant difference between the rule sizes of GA, ID3, and PLS1. Thus we can conclude that (1)GA outperformed ID3 and PLS1 both in terms of *prediction accuracy* and *conciseness* (rule size) and (2)GA was more consistent in the results as shown by the low variance in its results as compared to ID3 and PLS1.

7. Double Layered Learning Approach

GA holds promise as a robust machine learning method (especially for problems where the hypothesis space is not very smooth and has many peaks) which, as shown by the performance results just described, is able to find a concept representation with minimum rule size and with prediction accuracy which is better than ID3 and PLS1. In other words, it is able to locate a globally optimum concept when the traditional methods usually give a sub-optimal concept. On the other hand, GA is computationally more costly than either of the two inductive learning algorithms considered.

Based on the empirical study in the preceding section, one can conclude that there are tradeoffs between the GA and the similarity-based learning systems, such as ID3 or PLS1, in terms of *accuracy, conciseness, and the computation cost.* This gives rise to the idea of combining the features of both types of systems so as to have the advantages of both - the computational efficiency of these similarity-based learning algorithms and the ability of a GA to perform global search and have concise descriptions. The resulting system is what we refer to as the *double layered system,* wherein the first layer consists of a very efficient algorithm (in this case, PLS1) and the second layer which takes its input from the lower layer consists of a GA. A prototype of such a system was developed and tested, which we call Double Layered System (DLS), and is described as follows.

7.1 The Double Layered Learning System

A learning system, including the one described so far, can be generally characterized by the diagram shown in Figure 7.1 (a), Where $\{(\underline{X},u)\}$ is the input data set (better known as the training data set). For each instance of $(\underline{X},u)$, $\underline{X}$ is the vector of attribute values and u is the corresponding classification variable (usually a binary variable with 1 for positive and 0 for negative examples). The output of the system is the concept $u(\underline{X})$ i.e., the classification variable as a function of the attribute vector. The algorithm A takes as its input the training data set $\{(\underline{X},u)\}$ and finds a concept $u(\underline{X})$ explaining the input data set. Thus the problem of learning the concept can be thought of as the problem of finding the function $u(\underline{X})$.

The key idea of the DLS is to use PLS1 to generate a population of hypotheses to be used as the input to GA. These inputs provide good initial population for GA, thus saving a big chunck of GA's computational time for reaching the correct concept sooner. At the same time, GA provides a more globally oriented search with implicit parallelism, thus improving the quality of the learning results. The functional diagram of DLS is shown in

Figure 7.1 (b), where $\{(\underline{X},u)\}$ is the available data set and $\{(\underline{X},u)\}_i$ is the i^{th} sub-sample[4] generated from the data set using a random sampling technique. Jackknife technique (Efron[1982]) of drawing random samples was considered. In this technique one or more data points are removed from the data set S to get a sub-sample S_1. This is repeated to get more sub-samples S_i , i = 1,2 ... ω. We use a Jackknife technique wherein a fixed percent of the data points(say r%) are removed from S to get the subsamples S_i , i = 1,2 ... ω. We refer to this technique as the Jackknife technique of 'leave out r%'.

Insert Fig. 7.1 (a) & (b) here

7.2 Features of the System

At first the random sample generator takes the input data set and generates different random samples out of it using the Jackknife technique. The sample generator chooses 'n' instances out of the N randomly to be used as testing against the final concept generated by the system. From the remaining (N-n) instances, the jackknife technique of 'leave out r %' is used to generate different random samples $\{(\underline{X},u)\}_i$ which are then used by PLS1 to generate different concepts $u(\underline{X})_i$. These concepts are then used by the GA to form the initial population. The GA works in the same way as described in § 4.2. At each generation the GA uses the same (N-n) instances to evaluate its population of hypothesis. Before such a system could be designed, however, there is the issue of interface between the two algorithms PLS1 and GA.

In order for the GA to use the output of PLS1, the output of PLS1 has to be changed into the representation used by the GA (see § 4.2). The representation used by PLS1 is also similar but since the GA uses only single disjuncts in its population, the concept $u(\underline{X})$ given by PLS1 needs to be broken into individual disjuncts and given as input to the GA. This compatibility of the concept representation used by PLS1 and GA is the key linkage which ensure the success of the double layered approach.

Specifically, the concept descriptions generated by PLS1 are represented by the following form:

$$P = P_1 \vee \ldots \vee P_m$$

$$= \{ (d_{11} \le x_{i_{11}} \le e_{11}) \& \ldots \& (d_{1k} \le x_{i_{1k}} \le e_{1k}) \} \vee \ldots \vee \{ (d_{m1} \le x_{i_{m1}} \le e_{m1}) \& \ldots \& (d_{mk} \le x_{i_{mk}} \le e_{mk}) \}$$

$$= \{\xi_{11} \& \xi_{12} \& \ldots \& \xi_{1k}\} \vee \{\xi_{21} \& \xi_{22} \& \ldots \& \xi_{2k}\} \vee \ldots \vee \{\xi_{m1} \& \xi_{m2} \& \ldots \& \xi_{mk}\}$$

Rendell(1986) defined each P_i as a region geometrically. But the representation used by PLS1 is very compatible with the disjunctive normal form used by GA.

To generate enough samples for the initial population as the input to GA, the original data set is partitioned into ω subsamples by the jackknife technique. Each subsample is fed into the PLS1, for a total of ω runs. For each run i, PLS1 generates

$$P^i = P^i_1 \text{ v } \ldots P^i_{m_i}$$

The DLS system takes each of the P^i_j's from P^i as one member of the population, for i = 1,..., ω. Therefore, the initial population input to the GA component has a total of $m_1 + m_2 + \ldots + m_\omega$ members, represented in the form of genotypes.

8. Search Process of the DLS: an Example

We consider two data sets to show the working of the DLS. First, we use the Bankruptcy Data to show the improvement brought on by the search process of DLS over that of GA, and then use the Loan Data to show the improvement of DLS over PLS1.

Bankruptcy Analysis:

We consider a sample of instances from the bankruptcy data consisting of 58 training examples (same as in § 5) to demonstrate the search process. The final concept given by the DLS for these data had rule size of three (i.e., three disjuncts) as shown in Table 8.1.

Insert Table 8.1 Here

The rule thus learned is $r = (C_1 \text{ v } C_2 \text{ v } C_3)$. Comparing with § 5 we see that the rule size has been reduced from 4 to 3. Now we shall compare the search process of the DLS with that of the GA(§ 5) for the first disjunct C_1. In this example the following parameter values were used:

number of sub-samples used by PLS1 = 10
population size = 42 (this is determined by the output from PLS1)
probability of crossover = 0.7
probability of mutation = 0.01

Figure 8.1 shows the learning curve of DLS vs GA

Insert Figure 8.1 Here

Figure 8.1 shows the improvement in terms of solution (the improvement in the fitness measure in this case is about 0.1721, i.e. an improvement of 27.7%), population size (which in this case was 42 as compared to 100 used for the GA alone). Since population size is reduced to about half, it reduces the time complexity of GA (O(LN)) proportionally. This along with the 'good' initial population (thanks to PLS1) considerably improves the efficiency compared to using GA alone. These results are further confirmed by the empirical study.

Loan Classification:

The Loan data used consisted of information about different companies together with their loan risk class (or category), with class 1 being the lowest risk level and class 5 the highest risk. The data set had 100 instances. The concept generated by PLS1 had 47 rules (or disjuncts). For brevity we would not show all those rules here. Instead we give the rules generated by the DLS, which are 25 in number, and show the search process of the DLS for one such rule.

Table 8.2 shows the rules learned for each of the classes.

Insert Table 8.2 here

The above rules can be interpreted in the following way:

If R_{11} or R_{12} or R_{13} or R_{14} is true
then *the company belongs to the risk class 1.*
If R_{21} or R_{22} or R_{23} or R_{24} or R_{25} or R_{26} or R_{27} is true
then *the company belongs to the risk class 2 .*
If R_{31} or R_{32} or R_{33} or R_{34} or R_{35} or R_{36} or R_{37} is true
then *the company belongs to the risk class 3 .*
If R_{41} or R_{42} or R_{43} or R_{44} or R_{45} is true
then *the company belongs to the risk class 4 .*
If R_{51} or R_{52} is true
then *the company belongs to the risk class 5 .*

We will show the search process of DLS for rule R_{31} by showing snapshots of the top 4 members of the population at different generations. The initial population (i.e. at generation 0) is the output from PLS1. The first 12 variables correspond to the cash flow components as explained in appendix B, and the last three correspond to the qualitative variables related to the quality of the loan-granting decision, such as the quality of the collateral, whether the loan is secured, and whether the loan is guaranteed. The variable range (* a) for X means that $X \leq a$, and * means that that variable is irrelevant and can take any value (wild card).

Insert Table 8.3 here

This example shows the improvement brought on by the search process of the DLS over the rules generated by PLS1. We see that the best rule generated by PLS1 covers 22 instances. The GA part of DLS improves upon these rules, and takes only 3 generations to improve it to cover 25 instances and thus helps in reducing the total size of the concept from 47 rules to 25 rules. Thus DLS improves the rule size (or conciseness) over PLS1 by about 47% and the solution quality by about 9%.

9. Empirical Results with the Double Layered Learning System

9.1 The Empirical Study

We present the empirical results comparing the performance of DLS with GA and PLS1 based on the two data sets used before i.e., bankruptcy data set and default loan data set.

Bankruptcy Analysis

This empirical study uses the same bankruptcy data set as in §6. The same 10 samples of 58 training and 46 testing examples were used (n = 46 according to § 7.2). For each of the samples, the jackknife technique was used to randomly select 10 different data group, each with 60% of the 58 cases (i.e. each example in the set has 0.6 probability of being selected).These 10 groups of data were used as 10 different inputs to the PLS1 component to generate 10 different concept descriptions. As explained previously, the disjuncts in these concept descriptions were used as the members of the initial population by the GA component of the double layered system. The performance results are shown in Table 9.1.

Insert Table 9.1 here

A t-test for paired observations was carried out (as in §6), and the difference between the prediction accuracy of DLS and PLS1 was significant even at 2% level of significance. However, the difference between the accuracy of DLS and GA was not found to be statistically significant. The improvement in rule size by DLS over that of GA and PLS1 was found to be statistically significant at 5% level of significance.

Default Loan Classification

The Default data set used is the same as in § 5.1. Only one training set with 32 instances and one testing set with 16 positive instances was used. The Jackknife technique of leave out 40% was used to get 20 samples from the training set. The following parameter values were used

number of sub-samples used by PLS1 = 20

population size for the GA = 46 (this is determined by the output from PLS1)

probability of crossover = 0.7

probability of mutation = 0.01

The results are shown in Table 9.2

Insert Table 9.2 here

The following were the rules obtained by PLS1 and DLS, along with the number of instances they correctly predicted(from the total of 16).

PLS1:

C_1:

If [(TOTAL_DEBT TO TOTAL_ASSETS RATIO ≤ 0.763) and (CURRENT_ASSET TO CURRENT_LIABILITY RATIO ≤ 1.967)]

then *'the firm would default on loan payment'* ---------- correctly predicted 6

C_2:

If [(NET_INCOME TO TOTAL_ASSETS RATIO ≤ -0.015) and ((TOTAL_DEBT TO TOTAL_ASSETS RATIO > 0.763) and (CURRENT_ASSET TO CURRENT_LIABILITY RATIO ≤ 1.967)]

then *'the firm would default on loan payment'* ---------- correctly predicted 4

C_3:

If [(1.967 < CURRENT_ASSET TO CURRENT_LIABILITY RATIO≤ 2.574) and (WORKING_CAPITAL TO SALES RATIO ≥ 0.226)]

then *'the firm would default on loan payment'* ---------- correctly predicted 1

C_4:

If [(CURRENT_ASSET TO CURRENT_LIABILITY RATIO ≥ 4.578)]

then *'the firm would default on loan payment'* ---------- correctly predicted 0

DLS:

C'_1:

If [(CURRENT_ASSET TO CURRENT_LIABILITY RATIO ≤ 1.967)]

then *'the firm would default on loan payment'* ---------- correctly predicted 14

C'_2:

If [(1.967 < CURRENT_ASSET TO CURRENT_LIABILITY RATIO ≤ 2.574) and (WORKING_CAPITAL TO SALES RATIO ≥ 0.226)]

then *'the firm would default on loan payment'* ---------- correctly predicted 1

C'_3:

If [(CURRENT_ASSET TO CURRENT_LIABILITY RATIO ≥ 4.578)]

then *'the firm would default on loan payment'* ---------- correctly predicted 0

The above result demonstrates the *rule refinement* feature of the DLS, wherein it combines one or more rules from the output of PLS1 to generate a more concise and powerful rule. We can see that DLS combined the rules C_1 & C_2 from the output of PLS1 to get the simple rule C'_1:

If [(CURRENT_ASSET TO CURRENT_LIABILITY RATIO ≤ 1.967)]

then *'the firm would default on loan payment'*

which correctly predicted 14 out of the 16 default loans, more than the combined prediction of the output of PLS1.

A look at the rules C_1 & C_2 from the output of PLS1 shows that the attribute (TOTAL_DEBT TO TOTAL_ASSETS RATIO) has the value of ≤ 0.763 in C_1, and the value of >0.763 in C_2,with the value of the attribute (CURRENT_ASSET TO CURRENT_LIABILITY RATIO) remaining same in both the rules. Thus combining these rules would eliminate this variable to give a more concise rule, which is what the DLS achieves. Thus, DLS reduces

the rule size from 4 to 3 and improves the prediction accuracy from 68.8% to 93.8% vis-a-vis that of PLS1.

9.2 Discussion

The comparison of the inductive learning program PLS1 with GA shows that the accuracy of GA is better than PLS1 and GA is able to find the most concise (optimum) concept. On the other hand GA is very slow as compared to PLS1.The time complexity of a GA is O(LN) or better, where L is the length of each member of the population and N is the population size. There is also another important factor for the efficiency of a GA, and that is the initial population with which it starts the search. If, to begin with, the population consists of diverse members with 'good enough' fitnesses then the GA would be much more efficient as is explained below. Brady(1985) showed the improvement in performance brought on by maintaining *diversity* in the population, which in our case is provided by the input from PLS1.

Figure 9.1 explains, in general, how the learning curve of the GA will be effected if the initial population contains more fit members to start with. It attempts to generalize the learning behavior of the DLS in comparison with a regular GA. As shown, if the initial population has the maximum fitness of 'f' then the GA would be saving 't' generations worth of time, all else remaining same. It might also effect the final solution to which it converges, as shown by δ in the figure. Thus the main advantage of having a good initial population would be efficiency. It also has important effect on the required population size 'N'. Since the optimum value of N depends on L and since L is fixed hence it appears that there is no way of improving upon the time complexity of O(LN) without sacrificing performance. But, it turns out that if the initial population is a 'good one', then the value of N can be reduced without effecting the performance.

Insert Figure 9.1 Here

Thus, the double layered system has these advantages:

(1) it improves upon the time complexity of the GA by providing it with an initial population of good hypothesis, with greater diversity, generated by PLS1;

(2) it improves upon the accuracy of the PLS1 and GA by providing the ability of a GA to perform more global search, with a fitter population generated by PLS1; and

(3) it improves upon the rules(disjuncts) given by PLS1 and hence is able to cover all the examples in less number of rules.

The above results show the improvement in both accuracy and conciseness brought on by the Double Layered System over PLS1 1& GA (for the data set considered). Thus, from the results of both § 6 and § 9 we can conclude that double layered system performs best of all the three algorithms considered (ID3, PLS1 and GA), in terms of accuracy and conciseness. Figure 9.2 shows the qualitative comparison of the inductive learning algorithms ID3, PLS1 with GA on the three dimensions of *accuracy, rule size* (measuring conciseness), and, *efficiency* and shows how double layered system(DLS) takes the best from both PLS1 and GA. IDEAL in the figure refers to the (ideal)desired learning algorithm.

Insert Figure 9.2 Here

In applying GA, the quality of the solution when the leaning process converges is not guaranteed. For example, if we look at Figure 8.1, which is about the performance improvement of DLS over GA. We can see that both the speed of convergence as well as the fitness level when it converge have improved in the double layered system. Both of these can be explained by having a population of hypotheses, generated by PLS1, that are already pretty close to the final concept description. But the fact that DLS generates fitter concepts points to the logical conclusion that *the concept learned for classification is dependent upon the initial population given in GAs*. This unstable aspect of GA makes it difficult to derive the optimality of the solutions generated by GA. On the other hand, it also gives additional advantage to DLS which ensures that the GA starts out with a good population. DLS can be viewed as taking a decompositional approach(i.e., the original set of training example is decomposed into smaller subsets) to concept learning, which enables the system to be equipped with multiple mechanisms for pursuing several promising search directions. The performance results therefore are predictably good. In theoretical biology, this decomposition strategy is akin to the *speciation* phenomenon in biological evolution(Brady[1985]).

Acknowledgements:

We would like to thank Prof. Larry Rendell of the Department of Computer Science, University of Illinois for his comments and suggestions about the Double-layered Learning System.

References:

ABDEL-KHALIK,A.R., AND EL-SHESHAI,K.M., "Information choice and Utilization in an Experiment on Default Prediction," *Journal of Accounting Research* , Autumn, 1980, pp. 325-342.

BAKER,J.E.," Reducing Bias and Efficiency in the Selection Algorithm," *Proceedings of the Second International Conference on Genetic Algorithms*, MIT, Cambridge, MA, 1987, pp.14-21.

BRADY,R.M., "Optimization Strategies gleaned from Biological Evolution," *Nature* , Vol. 317, October, 1985, pp. 804-806.

BUCHANAN,B.G., " Can Machine Learning Offer Anything to Expert Systems?" *Machine Learning*, Vol.4, No. 3/4, 1989, pp. 251-254.

CARTER,C. AND CARTLETT,J., "Assessing Credit Card Applications Using Machine Learning," *IEEE Expert*, Fall, 1987, pp. 71-79.

COHEN,P. AND FEIGENBAUM,E.,*The Handbook of AI*, Vol. 3, Heuristech Press. Stanford, CA, 1982.

CURRIM,I.S., MEYER,R.J., AND, LE,N.T., "Disaggregate Tree-Structured Modelling of Consumer Choice Data," *Journal of Marketing Research*, August, 1988, pp. 253-265.

DAVIS,L.,*Genetic Algorithms and Simulated Annealing*, Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1987.

DE JONG,K.,1988, "Learning with GA: An overview," *Machine Learning*, pp.121-137.

DE JONG,K., "An Analysis of the Behaviour of a Class of Genetic Adaptive Systems," Doctoral dissertation, University of Michigan, 1975.

EFRON,B., *The Jackknife, the Bootstrap and Other Resampling Plans*., SIAM, Philadelphia, PA, 1982.

ESHELMAN , et.al., "Biases in the Crossover Landscape," *Proceedings of the Third International Conference on Genetic Algorithms*, George Mason University, June 4-7 1989, pp. 10-19.

FOGARTY,T.C., "Varying the Probability of Mutation in GA," *Proceedings of the Third International Conference on Genetic Algorithms*, George Mason University, June

4-7 1989, pp. 104-109.

GOLDBERG,D.,*Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Publishing Co., Inc., Reading, MA, 1989.

HANSEN,J.V. AND MESSIER,W.F.Jr., "Inducing Rules for Expert System Developement: An Example Using Default and Bankruptcy Data," *Management Science*, 34, 12, (1988), pp. 1403-1415.

HOLLAND,J., *Adaptations in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, Michigan, 1975.

HOLLAND,J. et. al., *INDUCTION: Processes of Inference, Learning, and Discovery* , The MIT Press, Cambridge, MA, 1986.

PACKARD,N., " A Genetic Learning Algorithm for the Analysis of Complex Data," Technical Report CCSR-89-10, Center for Complex Systems Research, University of Illinois, 1989.

QUINLAN,J.R., "Induction of Decision Trees," *Machine Learning*, 1, (1986), pp. 81-106.

RENDELL,L.A., "A Doubly Layered, Genetic Penetrance Learning System," *Proc. Third National Conference on Artificial Intelligence*, 1983, pp. 343-347.

RENDELL,L.A., "Genetic Plans and the Probabilistic Learning System: Synthesis and Results," *Proc ICGA*, 1985, pp. 60-73.

RENDELL,L.A., "A General Framework for Induction and a Study of Selective Induction," *Machine Learning*, 1,(1986), pp. 177-226.

RENDELL,L.A., CHO,H.H., AND SESHU,R., "Improving the Design of Similarity-Based Rule-Learning Systems," *International Journal of Expert Systems*, 2, (1989), pp. 97-133.

RENDELL,L.A., "Induction as Optimization", *IEEE Transactions on Syatems, Man, and Cybernetics*, Vol.20, no.2, 1990, pp.

SCHAFFER,J.D.," Learning Multiclass Pattern Discrimination," _____.

SHAW,M. AND GENTRY,J., "Inductive Learning for Risk Classification," *IEEE Expert*, February, 1990, pp. 47-53.

SYSWERDA,G., "Uniform crossover in GA," *Proceedings of the Third International Conference on Genetic Algorithms*, George Mason University, June4-7 1989, pp. 2-9.

---

[1] An *instance* is a data point or unit of a sample and is represented as a list of attribute values. The attributes define an *instance-space* in which each attribute is a distinct dimension.

[2]The H-space is the space of all possible concepts with each point representing a whole concept.

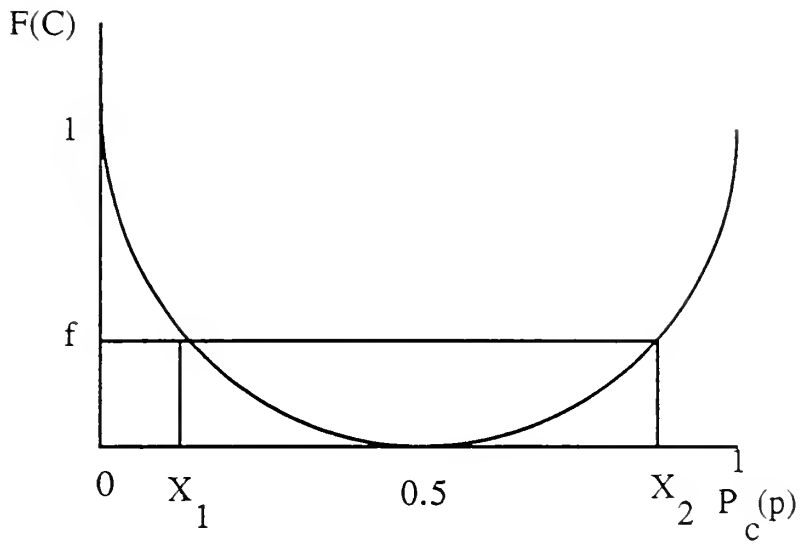
[3] DLS is similar to Rendell's PLS2 (Rendell, 1985) in the sense that it combines PLS1 and GA, but is functionally very different.

[4]It has been shown that averaging a statistic over several sub-samples gives a better estimate of the statistic than using the whole sample set.

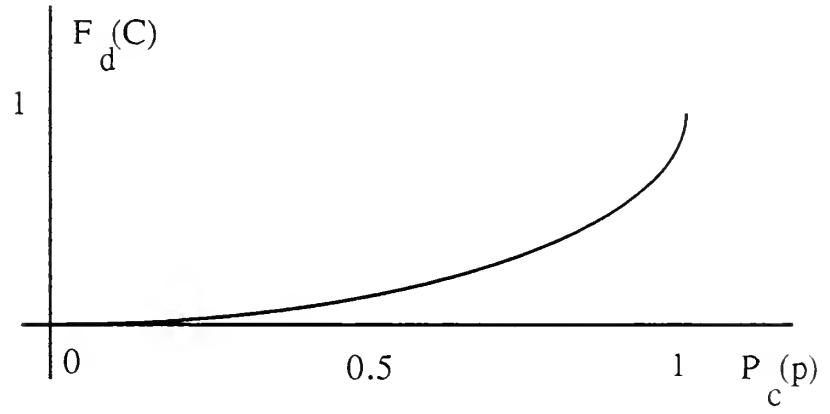
FIGURES

TABLES

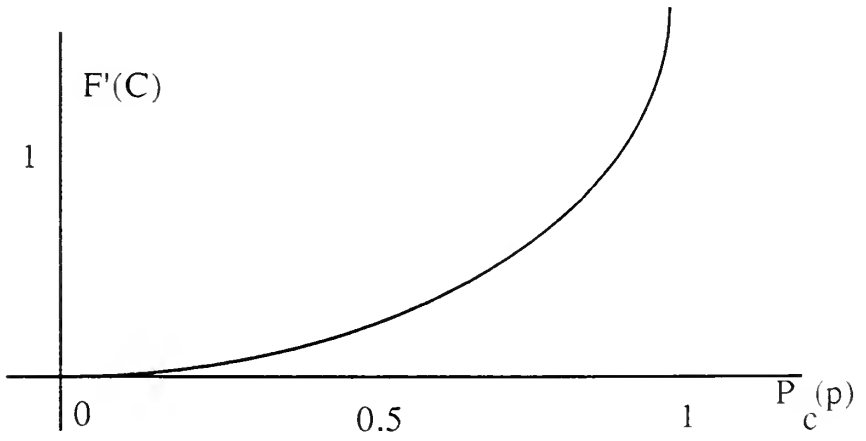
APPENDICES



(a)



(b)



(c)

Figure 4.1 Alternative Evaluation Functions

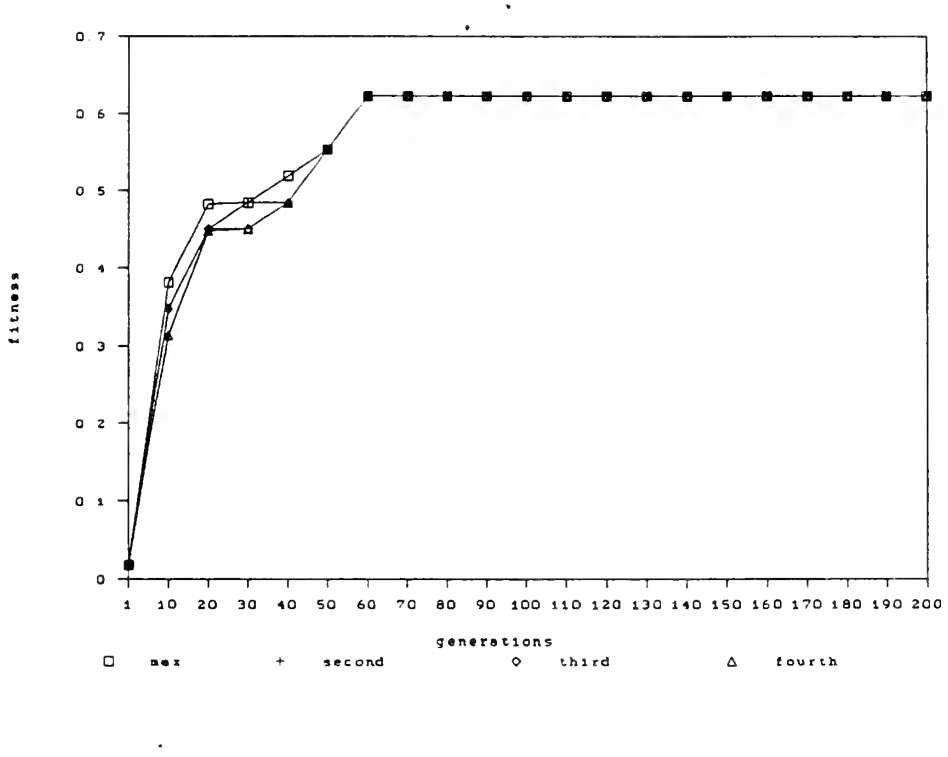


Figure 5.1 Learning Curves of the Top Four members

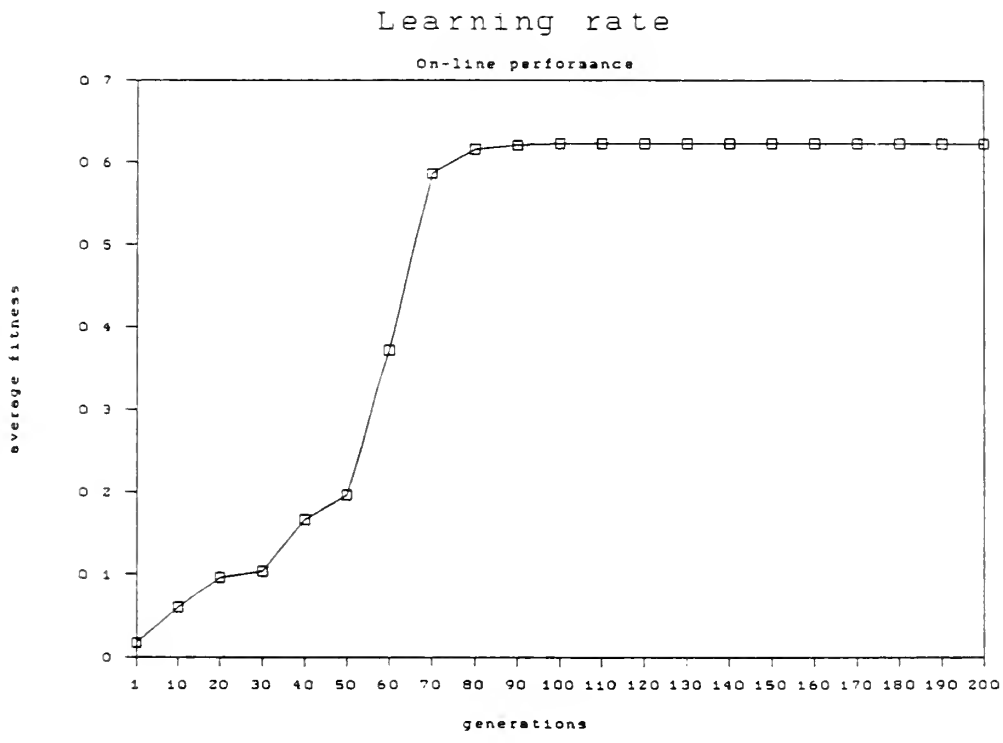


Figure 5.2 Learning Curve based on Average Fitness Level

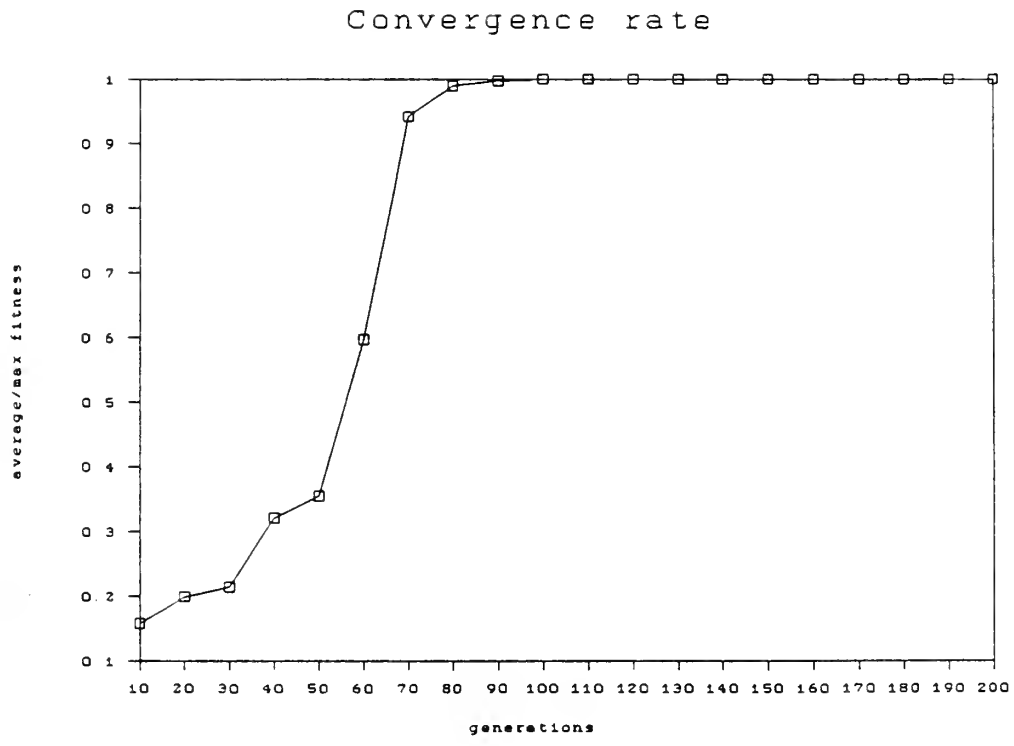


Figure 5.3 Learning Curve based on the Ratio of Average Fitness to Maximum Fitness

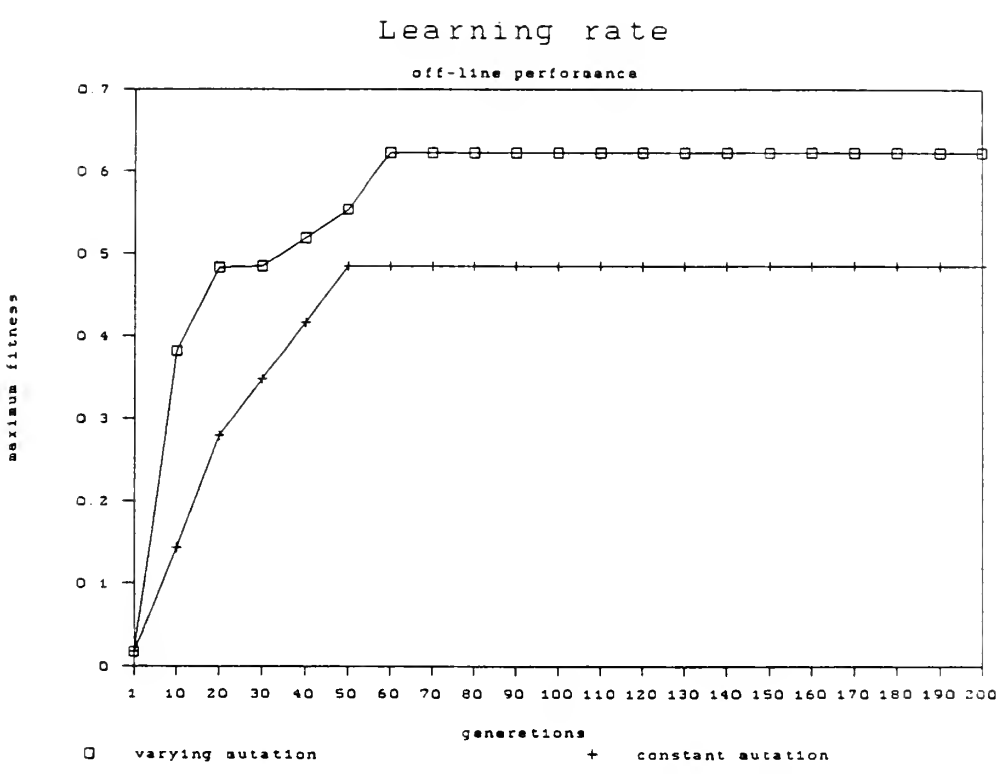
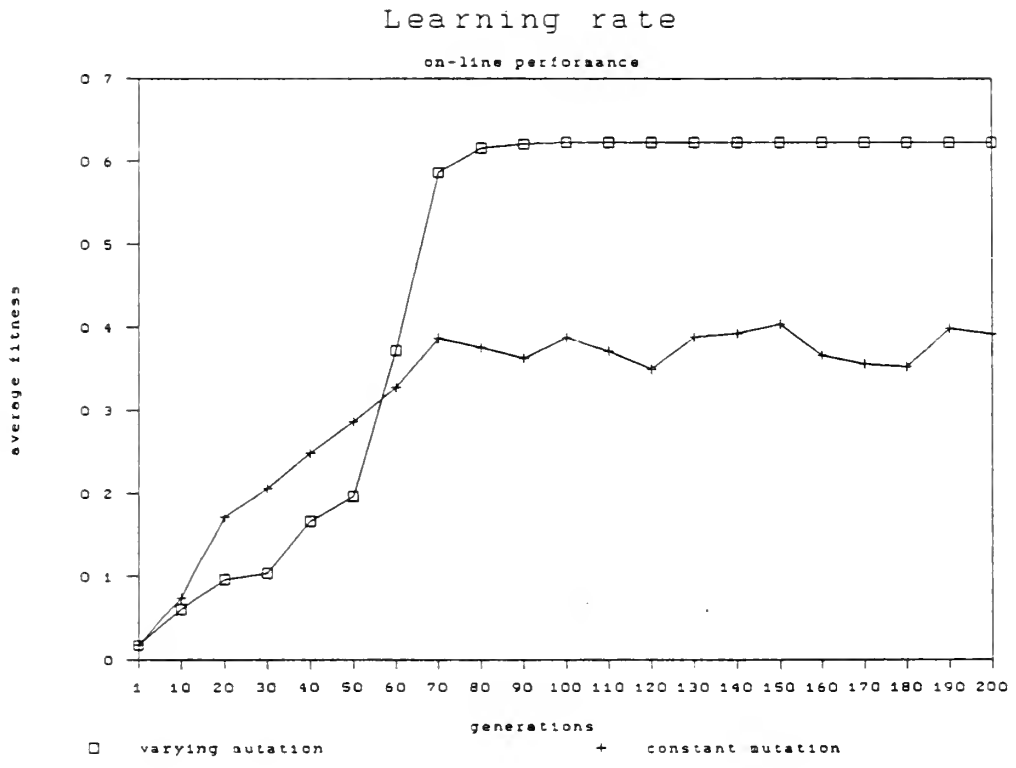

Learning rate
off-line performance
maximum fitness
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1 10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200
generations
varying mutation
constant mutation

Figure 5.4 (a)



(b)

Figure 5.4 A Comparison of Constant Mutation Rate Vs. Varying Mutation Rate measured by (a)Maximum Fitness, (b)Average Fitness

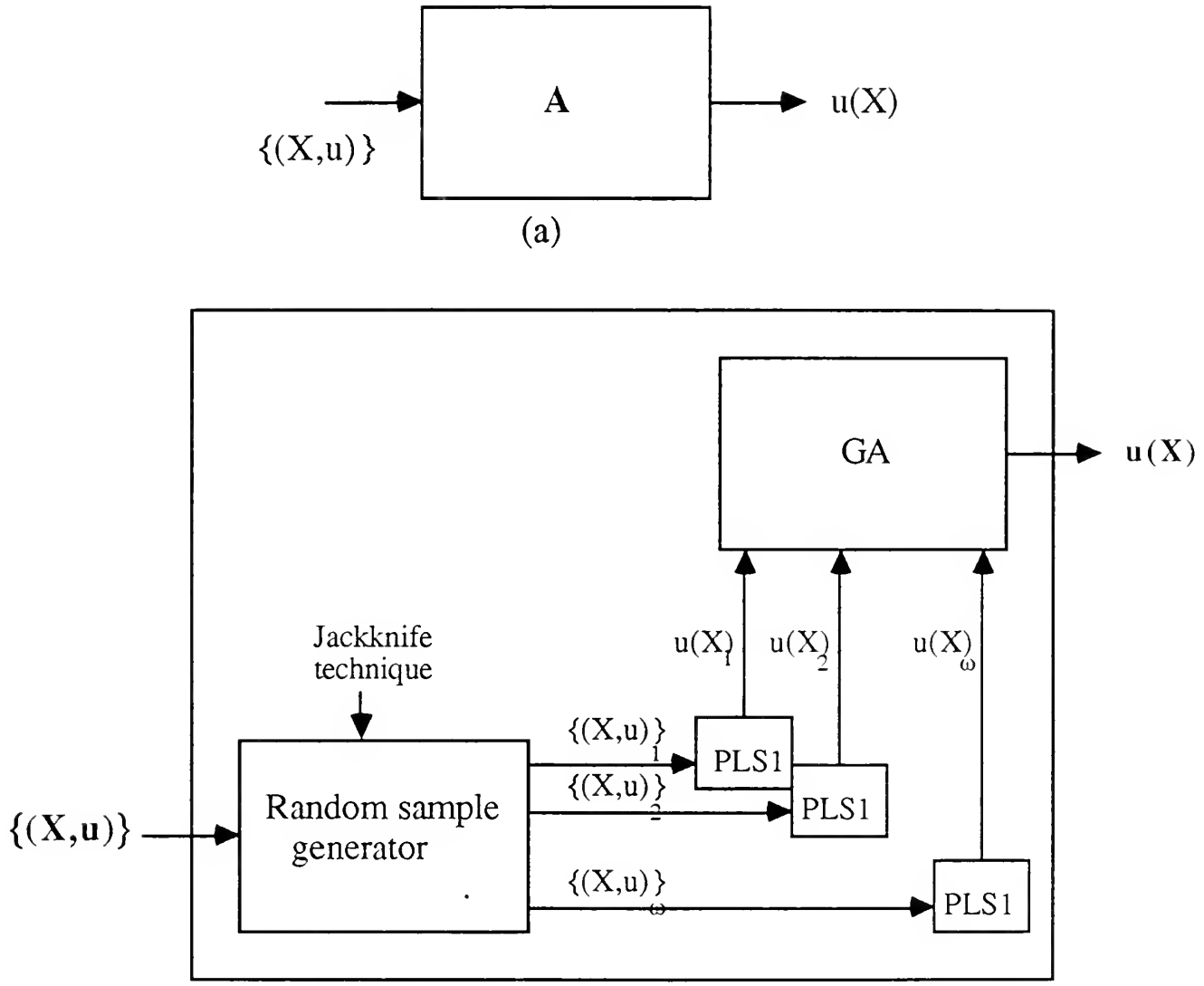


Figure 7.1 (a) The General Learning System; (b) The Double Layered Learning System

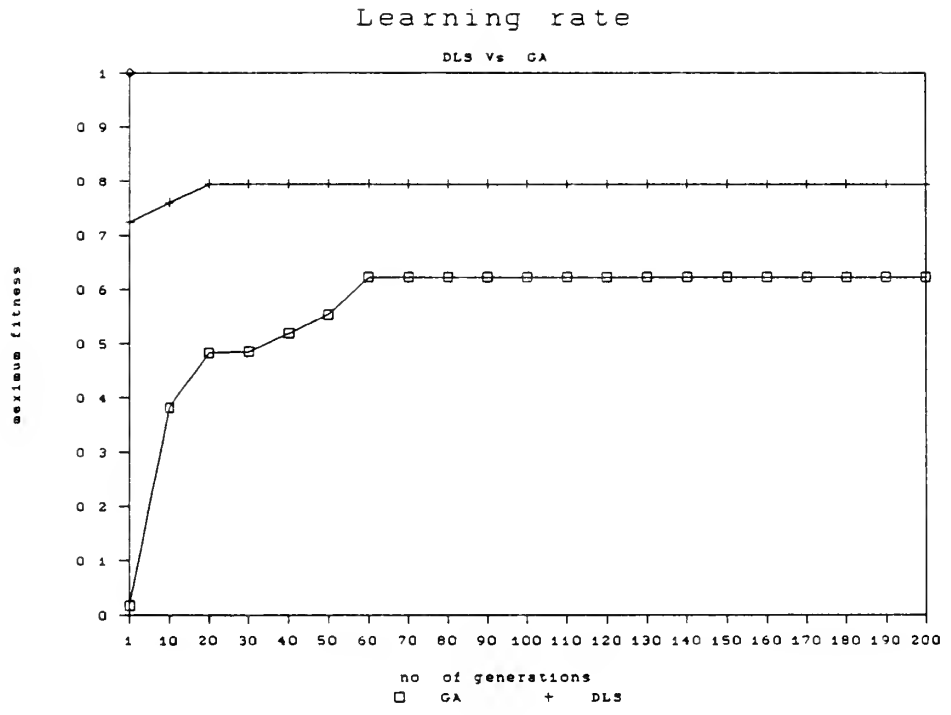


Figure 8.1 Comparison of Learning Rates of DLS Vs. GA

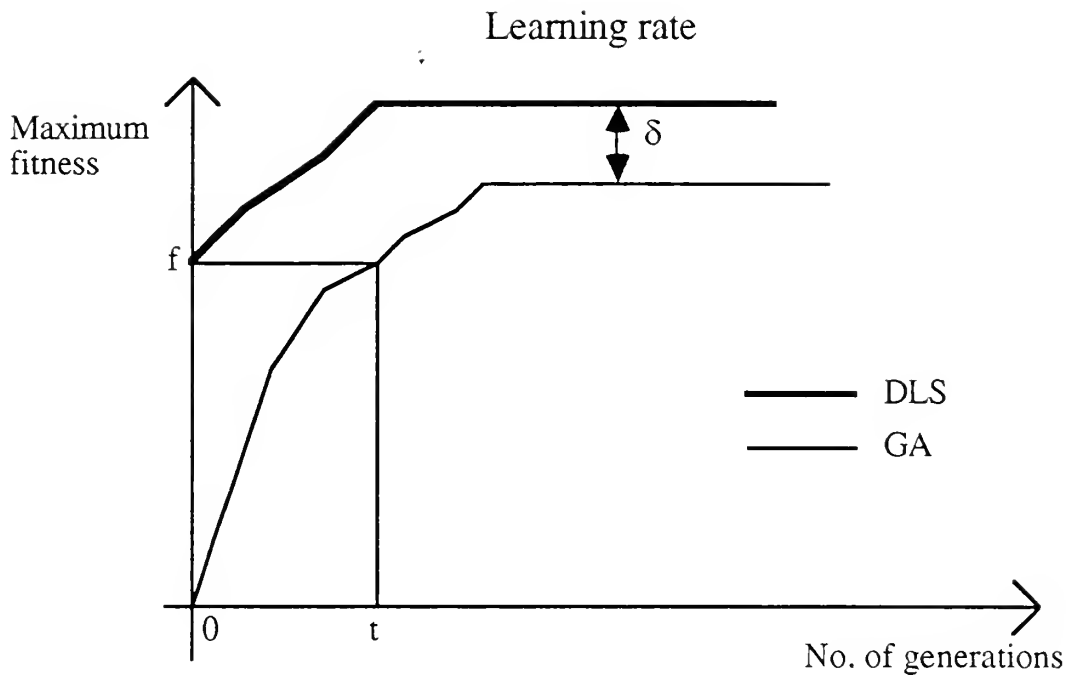


Figure 9.1 The Learning Rate Improvement by the Double Layered Learning System

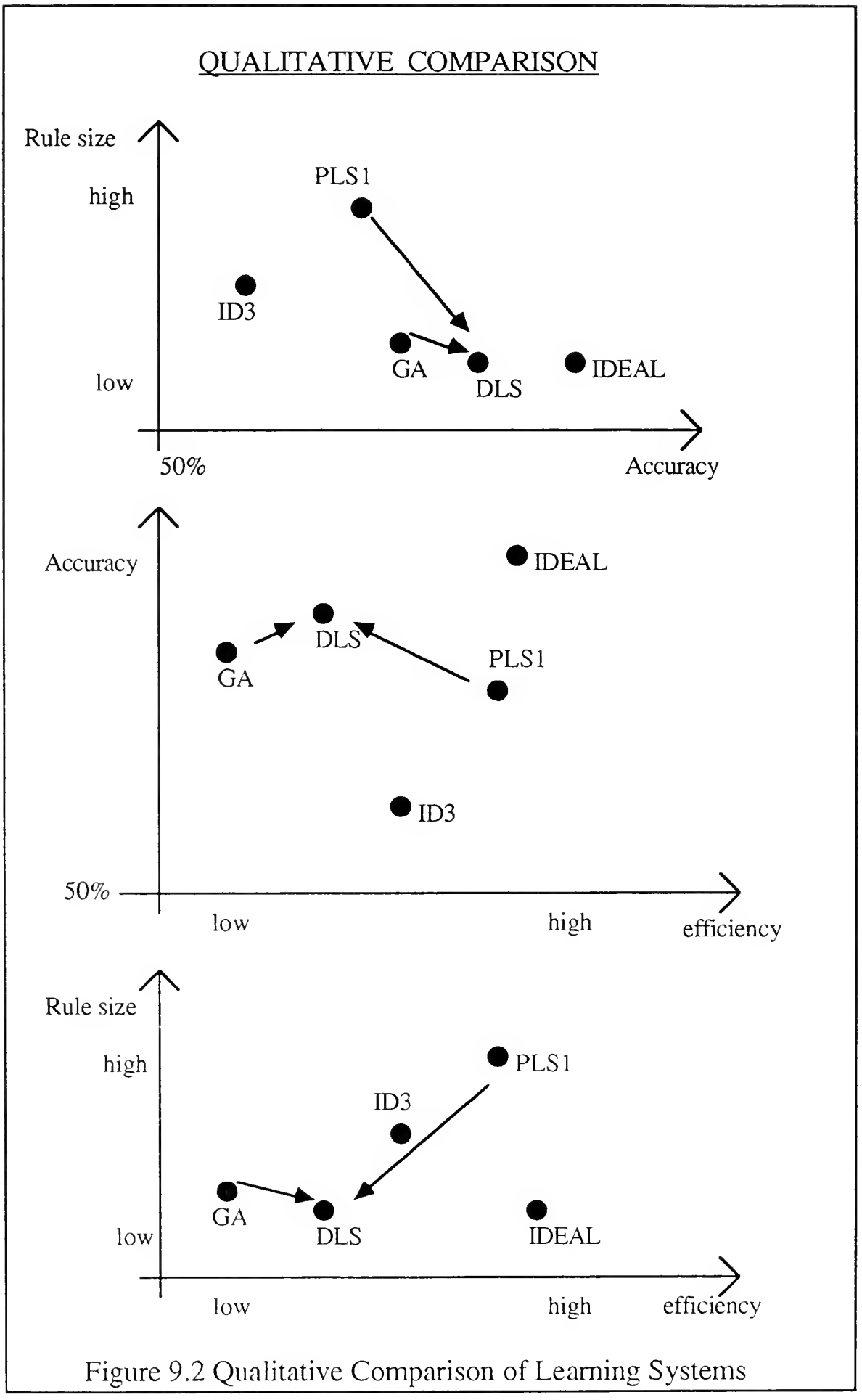


Figure 9.2 Qualitative Comparison of Learning Systems

Disjuncts	Fitness	No. of +ve & -ve
C_1 - ((-63 48) * (534 950) * * * * * * * * * * * * (-357 894) * *)[1]	- 0.51	9 +ve 1 -ve
C_2 - (* * (340 950) (395 685) * * (29 565) * * * * * * * * * *)	- 0.313	5 +ve 0 -ve
C_3 - (* * * * * * * * * * * (-2090 3669) * * (-294 -77) * * *)	- 0.125	2 +ve 0 -ve

1 - the * means that that particular attribute is not relevant and hence can take any value (wild card).

Table 5.1 Concept Descriptions Generated by the GA for Default Loan data

Disjuncts	Fitness	No. of +ve & -ve examples covered
C_1 - ((11 42) (3 45) (15 46) (0 46) (10 54) (3 48) (17 61) (28 37) (16 40))*	- 0.6224	19 +ve 1 -ve
C_2 - ((16 53) (8 58) (12 46) (7 54) (18 30) (9 47) (10 54) (27 55) (24 41))	- 0.276	8 +ve 0 -ve
C_3 - ((3 42) (0 60) (0 56) (21 63) (16 34) (10 13) (0 26) (1 59) (0 60))	- 0.035	1 +ve 0 -ve
C_4 - ((32 62) (3 60) (20 37) (25 58) (30 40) (24 59) (1 42) (22 24) (9 59))	- 0.035	1 +ve 0 -ve

* each element of the list is the range of values for the respective attributes(see section 4.2)

Table 5.2 Concept Descriptions Generated by the GA for Bankruptcy data.

Gen	Members	Important schema	Fitness	+ve	-ve
1)					
	((0 58)(0 58)(0 58)(0 58)(0 58)(0 58)(0 58)(0 58)(0 58))	(*********)	0.0172	29	29
	((0 58)(0 58)(0 58)(0 58)(0 58)(0 58)(0 58)(0 58)(0 58))	(*********)	0.0172	29	29
	((0 58)(0 58)(0 58)(0 58)(0 58)(0 58)(0 58)(0 58)(0 58))	(*********)	0.0172	29	29
	((0 58)(0 58)(0 58)(0 58)(0 58)(0 58)(0 58)(0 58)(0 58))	(*********)	0.0172	29	29
10)					
	((12 46)(3 38)(2 47)(0 58)(10 62)(3 42)(0 50)(2 58)(17 42))	(*(3 38)*******)	0.3820	12	1
	((12 56)(2 38)(2 44)(8 50)(2 62)(5 58)(0 48)(17 51)(17 50))	(*(3 38)*******)	0.3477	11	1
	((28 58)(2 38)(6 44)(1 62)(9 62)(18 42)(1 58)(8 42)(1 42))	(*(3 38)*******)	0.3477	11	1
	((0 40)(16 36)(2 61)(0 58)(10 62)(3 42)(0 58)(0 58)(0 42))	(*(16 36)*******)	0.3135	10	1
20)					
	((12 42)(3 39)(2 47)(0 62)(10 62)(3 32)(0 48)(17 35)(16 50))	(*(3 39)***(3 32)***)	0.4828	14	0
	((13 46)(7 39)(3 47)(0 62)(10 62)(3 32)(8 48)(17 33)(0 48))	(*(7 39)***(3 32)*(17 33)*)	0.4506	14	1
	((13 46)(7 39)(3 47)(0 62)(10 62)(3 32)(8 48)(17 33)(0 48))	(*(7 39)***(3 32)*(17 33)*)	0.4506	14	1
	((30 42)(3 39)(3 45)(1 58)(15 54)(12 34)(16 40)(3 35)(0 34))	(*(3 39)***(12 34)**(0 34))	0.4483	13	0
30)					
	((0 58)(3 39)(2 47)(0 62)(10 54)(5 31)(18 60)(18 50)(14 45))	(*(3 39)***(5 31)***)	0.4849	15	1
	((0 58)(3 39)(2 47)(0 62)(10 54)(5 31)(18 60)(18 50)(14 45))	(*(3 39)***(5 31)***)	0.4849	15	1
	((0 59)(3 39)(10 47)(8 62)(10 54)(13 31)(16 60)(3 51)(1 56))	(*(3 39)***(13 31)***)	0.4506	14	1
	((13 59)(11 39)(3 44)(0 60)(11 62)(5 31)(18 60)(18 50)(14 45))	(*(11 39)***(5 31)***)	0.4506	14	1
40)					
	((11 42)(3 45)(2 45)(0 42)(11 53)(3 59)(19 37)(28 37)(18 34))	(*(3 45)****(19 37)(28 37)*)	0.5193	16	1
	((0 59)(3 39)(10 47)(0 62)(10 54)(5 31)(16 52)(3 49)(1 56))	(*(3 39)***(5 31)***)	0.4849	15	1
	((0 59)(3 39)(10 47)(0 62)(10 54)(5 31)(16 52)(3 49)(1 56))	(*(3 39)***(5 31)***)	0.4849	15	1
	((0 63)(3 39)(11 47)(0 46)(10 54)(1 63)(16 60)(2 50)(11 52))	(*(3 39)*******)	0.4849	15	1
60)					
	((11 42)(3 45)(15 46)(0 46)(10 54)(3 48)(17 61)(28 37)(16 40))	(*(3 45)*****(28 37)*)	0.6224	19	1
	((11 42)(3 45)(15 46)(0 46)(10 54)(3 48)(17 61)(28 37)(16 40))	(*(3 45)*****(28 37)*)	0.6224	19	1
	((11 42)(3 45)(15 46)(0 46)(10 54)(3 48)(17 61)(28 37)(16 40))	(*(3 45)*****(28 37)*)	0.6224	19	1
	((11 42)(3 45)(15 46)(0 46)(10 54)(3 48)(17 61)(28 37)(16 40))	(*(3 45)*****(28 37)*)	0.6224	19	1

Table 5.3 An Example of the Learning Process by GA

	GA	ID3	PLS1
Prediction%			
1)	63.04%	43.5%	63%
2)	69.6%	52.2%	60.9%
3)	69.6%	56.5%	52.2%
4)	67.4%	63.0%	63%
5)	67.4%	69.6%	71.7%
6)	78.3%	56.5%	67.4%
7)	58.7%	63.0%	54.35%
8)	65.2%	45.7%	65.2%
9)	58.7%	69.6%	60.9%
10)	69.6%	47.8%	58.7%
Average:	**66.8%**	**56.74%**	**61.75%**
Std. deviation:	**5.52**	**8.98**	**9.34**
Rule size:			
1)	5	6	7
2)	5	4	6
3)	6	8	9
4)	6	9	13
5)	6	7	10
6)	5	7	10
7)	4	7	7
8)	5	7	8
9)	5	8	8
10)	5	7	9
Average:	**5.2**	**7**	**8.7**
Std. deviation:	**0.6**	**1.27**	**1.91**

Table 6.1 Comparison Results of the Empirical Study

Disjuncts	Fitness	No. of +ve & -ve examples covered
C_1 - ((32 53) (40 56) (11 48) (20 38) (26 61) (16 29) (8 33) (19 29) (20 40))*	- 0.7945	24 +ve 1 -ve
C_2 - ((39 48) (38 55) (11 49) (13 38) (27 53) (19 29) (8 23) (22 29) (20 35))	- 0. 104	3 +ve 0 -ve
C_3 - ((32 36) (40 54) (28 44) (16 34) (27 53) (13 29) (11 32) (19 29) (20 40))	- 0.069	2 +ve 1 -ve

Table 8.1 Concept Descriptions Generated by DLS

Class 1:

R_{11}: (TNF/TA ≥ 0.1261) and (AP ≤ 0.0364) and (OTHCL ≥ 0.00315) and (FCEXP ≥ -0.1572) and (INVST ≤ 0.0832) and (DIV < -0.1498).

R_{12}: (OPER ≤ 0.642) and (-0.1498 ≤ DIV < -0.1388).

R_{13}: (INV ≤ -0.1759) and (INVST ≤ 0.152) and (-0.1388 ≤ DIV < -0.0948).

R_{14}: (OTHA&L ≤ -0.1354) and (-0.691 ≤ INVST ≤ -0.631) and (DIV ≥ -0.0948)

Class 2:

R_{21}: (TNF/TA ≥ 0.168) and (AR ≥ -0.1427) and (OTHCL ≤ 0.1527) and (DIV < - 0.1388).

R_{22}: (AR ≥ -0.0776) and (INV ≥ -0.1759) and (INVST ≤ 0.152) and (-0.1388 ≤ DIV < -0.0948).

R_{23}: (AR ≥ -0.1427) and (-0.635 ≤ FIN ≤ -0.611) and (INVST ≤ 0.152) and (DIV ≥ -0.0948).

R_{24}: (TNF/TA < 0.168) and (AR ≥ 0.0418) and (OTHCL ≤ 0.1527) and (DIV < -0.1498).

R_{25}: (TNF/TA ≥ 0.2389) and (OTHCL ≤ 0.2562) and (OTHA&L ≤ -0.1354) and (FIN ≤ -0.1027) and (FCEXP ≥ -0.1028) and (DIV ≥ -0.1498).

R_{26}: (FIN ≥ 0.1393) and (FCEXP ≥ -0.1708) and (INVST ≤ -0.45) and (DIV ≥ -0.1388).

R_{27}: (AP ≤ -0.3836) and (INVST ≤ 0.152) and (DIV ≥ -0.0948).

Class 3:

R_{31}: (TNF/TA ≤ 0.6478) and (OTHCA ≤ 0.2325) and (OTHCL ≤ 0.2217) and (-0.1244 ≤ OTHA&L ≤ 0.3177) and (-0.4173 ≤ FIN ≤ 0.4781) and (FCEXP ≥ -0.1368) and (INVST ≤ 0.0488) and (DIV ≥ -0.0948).

R_{32}: (OPER ≤ 0.73) and (AR ≥ -0.0776) and (-0.5876 ≤ INVST ≤ -0.0028) and (DIV ≥ -0.0948)

R_{33}: (TNF/TA ≤ 0.3517) and (OTHCA ≤ 0.2013) and (AP ≤ 0.1372) and (OTHCL ≥ -0.1694) and (OTHA&L ≤ 0.3177) and (-0.3812 ≤ INVST ≤ 0.0488) and (DIV ≥ -0.0948).

R_{34}: (AR ≤ -0.0776) and (INV ≥ -0.0584) and (-0.1388 ≤ DIV < -0.0948).

R_{35}: (0.3094 ≤ TNF/TA ≤ 0.6478) and (OPER ≥ 0.07) and (OTHCA ≤ 0.2013) and (AP ≥ -0.3164) and (FCEXP ≥ -0.0892) and (INVST ≤ 0.1864) and (DIV ≥ -0.0948).

R_{36}: (OPER ≥ 0.642) and (-0.1498 ≤ DIV ≤ -0.1388).

R_{37}: (AR ≤ -0.1427) and (-0.635 ≤ FIN ≤ -0.611) and (INVST ≤ 0.152) and (DIV ≥ -0.0948).

Class 4:

R_{41}: (TNF/TA ≤ 0.676) and (0.356 ≤ OPER ≤ 0.664) and (AR ≤ 0.2697) and (INV ≤ 0.388) and (OTHCA ≤ 0.2325) and (AP ≥ -0.2492) and (OTHCL ≤ 0.2217) and (OTHA&L ≥ -0.257) and (FCEXP ≤ -0.1232) and (INVST ≥ -0.5532) and (DIV ≥ -0.0948).

R_{42}: (0.73 ≤ OPER ≤ 0.851) and (AR ≤ 0.2914) and (OTHCA ≤ 0.1077) and (OTHCL ≤ 0.3022) and (FIN ≤ 0.3087) and (DIV ≥ -0.0948).

R_{43}: (0.168 ≤ TNF/TA ≤ 0.45) and (-0.128 ≤ OPER ≤ 0.642) and (INV ≤ 0.3647) and (OTHCA ≤ 0.2013) and (AP ≥ -0.2996) and (-0.478 ≤ OTHA&L ≤ 0.2293) and (FIN ≥ 0.4539) and (-0.1708 ≤ FCEXP ≤ -0.001) and (INVST ≥ -0.3812) and (DIV ≥ -0.0948).

R_{44}: (TNF/TA ≤ 0.1966) and (OTHCL ≥ 0.0377) and (OTHA&L ≥ -0.2791) and (FCEXP ≥ -0.2864) and (DIV ≥ -0.0948).

R_{45}: (INV ≤ 0.3647) and (OTHA&L ≥ 0.3177) and (DIV ≥ -0.0948).

Class 5:

R_{51}: (OPER ≤ 0.356) and (AP ≤ 0.3052) and (-0.6548 ≤ OTHA&L ≤ 0.2735) and (FIN ≤ 0.5749) and (DIV ≥ -0.0948).

R_{52}: (OTHA&L ≤ -0.7211) and (DIV ≥ -0.1498).

Note: For each class the rules are in the order of importance, with the most important rule being the first.

Table 8.2 Rules Generated by DLS for Loan Data

Gen. no.	members	fitness	+ve	-ve
0)				
	((* 0.65) (* 0.73) * (* 0.37) (* 0.2) * * (-0.12 0.32) (-0.42 0.24) * (* 0.05) (-0.095 *) * * *)	0.458	22	0
	((* 0.65) (0.07 *) (-0.43 *) * (* 0.2) (-0.32 *) (-0.28 0.29) (-0.66 *) (-0.13 *) (-0.1 *) (-0.59 0.08) (-0.1 *) * * *)	0.376	19	1
	((* 0.65) (0.09 *) * (* 0.39) (* 0.23) * (* 0.22) (-0.12 *) (* 0.45) (-0.13 *) * (-0.095 *) * * *)	0.2204	23	2
	((* 0.63) * * * (-0.13 0.23) * (* 0.22) (-0.12 0.27) (* 0.48) (-0.13 *) (* 0.05) (-0.095 *) * * *)	0.2204	23	2
1)				
	((* 0.65) (* 0.73) * (* 0.37) (* 0.2) * * (-0.12 0.32) (-0.42 0.24) * (* 0.05) (-0.095 *) * * *)	0.458	22	0
	((* 0.65) (* 0.73) * (* 0.37) (* 0.2) * * (-0.12 0.32) (-0.42 0.24) * (* 0.05) (-0.095 *) * * *)	0.458	22	0
	((* 0.65) (* 0.73) * (* 0.37) (* 0.2) * * (-0.12 0.32) (-0.42 0.24) * (* 0.05) (-0.095 *) * * *)	0.458	22	0
	((* 0.65) (* 0.73) * (* 0.37) (* 0.2) * * (-0.12 0.32) (-0.42 0.24) * (* 0.05) (-0.095 *) * * *)	0.458	22	0
2)				
	((* 0.65) (* 0.73) * (* 0.37) (* 0.2) * * (-0.28 0.32) (-0.42 0.24) * (* 0.05) (-0.095 *) * * *)	0.48	24	1
	((* 0.65) (* 0.73) * (* 0.37) (* 0.2) * * (-0.12 0.32) (-0.42 0.24) * (* 0.05) (-0.095 *) * * *)	0.458	22	0
	((* 0.65) (* 0.73) * (* 0.37) (* 0.2) * * (-0.12 0.32) (-0.42 0.24) * (* 0.05) (-0.095 *) * * *)	0.458	22	0
	((* 0.65) (* 0.73) * (* 0.37) (* 0.2) * * (-0.12 0.32) (-0.42 0.24) * (* 0.05) (-0.095 *) * * *)	0.458	22	0
3)				
	((* 0.65) * * * (* 0.23) * (* 0.22) (-0.12 0.32) (-0.42 0.48) (-0.14 *) (* 0.05) (-0.095 *) * * *)	0.5	25	1
	((* 0.65) (* 0.73) * (* 0.37) (* 0.2) * * (-0.28 0.32) (-0.42 0.24) * (* 0.05) (-0.095 *) * * *)	0.48	24	1
	((* 0.65) (* 0.73) * (* 0.37) (* 0.2) * * (-0.28 0.32) (-0.42 0.24) * (* 0.05) (-0.095 *) * * *)	0.48	24	1
	((* 0.65) (* 0.73) * (* 0.37) (* 0.2) * * (-0.28 0.32) (-0.42 0.24) * (* 0.05) (-0.095 *) * * *)	0.48	24	1
4)				
	((* 0.65) * * * (* 0.23) * (* 0.22) (-0.12 0.32) (-0.42 0.48) (-0.14 *) (* 0.05) (-0.095 *) * * *)	0.5	25	1
	((* 0.65) * * * (* 0.23) * (* 0.22) (-0.12 0.32) (-0.42 0.48) (-0.14 *) (* 0.05) (-0.095 *) * * *)	0.5	25	1
	((* 0.65) (* 0.73) * (* 0.37) (* 0.2) * * (-0.28 0.32) (-0.42 0.24) * (* 0.05) (-0.095 *) * * *)	0.48	24	1
	((* 0.65) (* 0.73) * (* 0.37) (* 0.2) * * (-0.28 0.32) (-0.42 0.24) * (* 0.05) (-0.095 *) * * *)	0.48	24	1
50)				
	((* 0.65) * * * (* 0.23) * (* 0.22) (-0.12 0.32) (-0.42 0.48) (-0.14 *) (* 0.05) (-0.095 *) * * *)	0.5	25	1
	((* 0.65) * * * (* 0.23) * (* 0.22) (-0.12 0.32) (-0.42 0.48) (-0.14 *) (* 0.05) (-0.095 *) * * *)	0.5	25	1
	((* 0.65) * * * (* 0.23) * (* 0.22) (-0.12 0.32) (-0.42 0.48) (-0.14 *) (* 0.05) (-0.095 *) * * *)	0.5	25	1
	((* 0.65) * * * (* 0.23) * (* 0.22) (-0.12 0.32) (-0.42 0.48) (-0.14 *) (* 0.05) (-0.095 *) * * *)	0.5	25	1

Table 8.3 An Example of the Learning Process of DLS

	GA	PLS1	DLS
Prediction accuracy:			
1)	63.04%	63%	71.7%
2)	69.6%	60.9%	60.9%
3)	69.6%	52.2%	56.5%
4)	67.4%	63%	76.1%
5)	67.4%	71.7%	69.6%
6)	78.3%	67.4%	73.9%
7)	58.7%	54.35%	84.8%
8)	65.2%	65.2%	76.1%
9)	58.7%	60.9%	60.9%
10)	69.6%	58.7%	71.7%
Average:	**66.8%**	**61.75%**	**69.4%**
Rule size:			
1)	5	7	3
2)	5	6	4
3)	6	9	5
4)	6	13	6
5)	6	10	4
6)	5	10	4
7)	4	7	7
8)	5	8	5
9)	5	8	6
10)	5	9	3
Average:	**5.2**	**8.7**	**4.7**

Table 9.1 Comparison of the Performances of DLS with PLS1 and GA

	GA	PLS1	DLS
Prediction accuracy:	62.5%	68.8%	93.8%
Rule size:	3	4	3

Table 9.2 Comparison of Performance of DLS with PLS1 and GA on Default Loan Data.

Appendix A

PLS1 Algorithm : The inductive process followed by the PLS1 algorithm starts with the entire space of possible events (the 'feature space'). The space is then further split into two 'regions,' those of which have a greater likelihood to being in a specific class (positive events) and those which have a greater likelihood to being in the other classes (negative events). The process of splitting continues, each split using only one attribute that is chosen according to an information-theoretic approach, until a stopping criterion is satisfied. In each iteration, the region R in the feature space can be defined by the tuple (r,u,e), where r is region or disjunct represented as conjunction of conditions (similar to the representation given in sec. 4.2; a disjunction of regions would then constitute a concept or hypothesis); u is the *utility* function giving the fraction of positive events to the total events covered by the disjunct, and e is the *error* rate allowed by the disjunct which is based on the number of positive events covered by the disjunct as compared to the total number of positive events.

Since the purpose of the algorithm is to maximise the dissimilarity between the disjuncts, the split is made based upon maximising the difference in the utilities of the two disjuncts (known as the distant function). Each disjunct, also called a hyper-rectangle, is also associated with its error measure e . In proportion to the number of positive events covered, e has a lower value. The distant function (d) is defined as follows

$$d = |\log u_1 - \log u_2| - t * \log(e_1 * e_2)$$

where,

u_1, u_2 - utilities for a tentative region dichotomy,

e_1, e_2 - respective error factors,

t - a constant representing degree of confidence.

Larger values of d correspond to higher dissimilarity.

Let S be the set of positive and negative training events and R as the hyper-plane that contains all events in E, the PLS1 algorithm can be summarized as follows:

ALGORITHM PLS1:

While any trial hyper-plane remains untested, *do*

Begin

1. Choose a hyper-plane not previously selected to become a tentative boundary for two subregions of R, r_1 and r_2.

2. Using the events from S, determine the utilities u_1 and u_2 of r_1 and r_2, and their error factors e_1 and e_2.

3. If this tentative dichotomy produces a dissimilarity$d$ larger than any previous value for d

then : create two permanent regions $R_1 = (r_1, u_1, e_1)$ and $R_2 = (r_2, u_2, e_2)$ having the (previously recorded) common boundary that gives the most dissimilar probabilities;

else : place R in the defined region set R to be output, and quit.

End.

Appendix B

Default-loan data set:

Abdel-Khalik and El-Sheshai (1980) had previously used this data set to classify a set of firms into those that would default and those that wouldn't default on loan payments. This data set has 32 examples of which 16 belong to the default class and the other 16 examples belong to the non-default class. The 18 attributes in the example set are: (1) Net income / total assets, (2) Net income / sales, (3) Total debt / total assets, (4) Cash flow / total debt, (5) Long term debt / net worth, (6) Current assets / current liabilities, (7) Quick assets / sales, (8) Quick assets / current liabilities, (9) Working capital / sales, (10) Current year equity / total debt, (11) Sales trend, (12) Earnings trend, (13) Current ratio trend, (14)

Working capital / sales trend, (15) Cash flow / total debt trend, (16) Long-term debt / net worth trend, (17) Net income / total assets trend, and, (18) Net income / sales trend.

Bankruptcy data:

This data set is from the *Standard and Poors' Compustat 1981 Industrial Annual Research file of Companies* and *Compustat Industrial files,* and was used to determine the companies that failed during the period 1970-1981. The data was used to predict bankruptcy of firms and is discussed in detail in Gentry, et al.(1985). The data set has 104 examples and two classes (either bankrupt or not) and there are 9 continuous variables , which are (1) ratios of total net flow (TNF), (2) funds from operations (NOFF), (3) working capital which includes inventory, other current assets and liabilities and accounts payable (NWCFF), (4) financial (NFFF), (5) fixed coverage expenses (FCE), (6) capital expenditures (NIFF), (7) dividends (DIV), (8) other assets and liability flows (NOA&LF), and, (9) change in cash and marketable securities (CC) with total assets (TA).

Loan risk rating data set:

This Loan Data consists of information about different companies together with their loan risk class (or category), with class 1 being the lowest risk level and class 5 the highest risk. The information is in the form of these 12 cash flow components: Total net Flow / Total Assets(TNF/TA), operating(OPER), accounts receivables(AR), inventories(INV), other current assets(OTHCA), accounts payables(AP), other current liabilities(OTHCL), other asset and liability flows(OTHA&L), financial(FIN), fixed coverage expenditures(FCEXP), investment(INVST), and dividends(DIV); together with qualitative information that indicated whether the loan was guaranteed or not guaranteed, the liquidity status of the collateral(COL), and secured or unsecured. The data set has 100 instances.