USING GENETIC ALGORITHMS FOR INSTANCE REDUCTION

تعتمد كلية الدراسات العليا
هذه النسخة من الرسالة
التوقيع ..... التاريخ ٢٦/١٧/٢٠٠٥

BY
Eman Faris Mohammad Issa

Supervisor
Dr. Khalil M. el Hindi

This Dissertation was Submitted in Partial Fulfillment of the Requirments for the Master's Degree of Computer Science

Faculty of Graduate Studies
University of Jordan

January, 2005

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

COMMITTEE DECISION

This thesis (Using Genetic Algorithms for Instance Reduction) was successfully defended and approved on, 10, 2005.

Examination Committee	**Signature**
Dr. Khalil el Hindi (Chairman) Assist. Prof. of Artificial Intelligence	K. Hindi
Prof. Nadeem Obaid Prof. of Artificial Intelligence	
Dr. Bassam Hammo Assist. Prof. of Natural Language Processing	Bassam Hammo
Dr. Riyad Shalabi Assist. Prof. of Artificial Intelligence (Yarmouk University).	

تعتمد كلية الدراسات العليا
هذه النسخة من الرسالة
التوقيع ..... التاريخ ٢٦/٧/٢٠٠٥

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

To the memory of my father,

To my mother,

To Maysoon

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

ACKNOWLEDGMENTS

I owe a dept of gratitude to many people who have been important for my studies and the completion of this thesis.

In particular, I am deeply grateful to my supervisor Dr. Khalil el Hindi for waken my interest in machine learning and inspiring the idea of this thesis. I would like to thank him for devoting a vast amount of hours to review, guidance, discussion, moral support, and mostly for believing in me. I would like to express my sincere gratitude to him for teaching me how to be a good student, researcher, and person.

Special gratitude is also extended to all KASIT staff for supporting me in every possible way.

Many thanks are also due to Safa, Tamara, Dania, Bashar, Luay, and Mohammad for being patient, supportive and helpful during this work.

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

List of Contents

Dedication........ii

Acknowledgements........iii

List of Contents........iv

List of Tables........vii

List of Figures........viii

Abstract........ix

Introduction........1

1. Induction Learning........1

2. Eager and Lazy Learners........2

3. Structure of The Thesis........4

Instance-Based Learning And Genetic Algorithms View........6

1. Introduction........6

2. Instance-Based Learning........7

2.1 The K Nearest Neighbor Algorithm........8

2.2 Distance Function........9

2.3 Drawbacks of Instance-Based Learning Algorithms........10

3. Reduction Techniques........12

3.1 A Revision of Reduction Algorithms........13

3.1.1 Condensed Nearest Neighbor Rule........13

3.1.2 Selective Nearest Neighbor Rule........14

3.1.3 Reduced Nearest Neighbor Rule........14

3.1.4 Edited Nearest Neighbor Rule........15

3.1.5 Encoding Length........15

3.1.6 Instance-Based learning algorithm2........15

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

3.1.7 Decremental Reduction Optimization Procedures (DROPs)....................16

4. Evolutionary algorithms....................18

4.1 Basic Concepts....................19

4.2 Representation (Encoding)....................20

4.3 Population Initialization....................21

4.4 Selection....................21

4.5 Crossover (Reproduction)....................23

4.6 Mutation....................25

4.7 Fitness Function....................25

4.8 Applications of Genetic Algorithms in Machine Learning....................26

A GENETIC ALGORITHM APPROACH FOR INSTANCE REDUCTION....................30

1. Introduction....................30

2. Genetically Reduced Instance-Based Learning System....................31

2.1 Individual Representation....................33

2.2 Population Initialization....................33

2.3 Evolution process....................34

2.4 Evolution operators....................35

2.5 Fitness Function....................38

2.6 Individual Classification Accuracy....................39

3. Seeded Genetically Reduced Instance-Based Learning System....................40

EMPIRICAL WORK....................42

1. Introduction....................42

2. Experimenting with GRIBL....................46

3. The Seeded-GRIBL Algorithm....................49

3.1 Seeded-GRIBL with Seeds From Ten Reduction Techniques....................50

3.2 DROPs Seeded-GRIBL....................55

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

4. The Effect of Evolution Parameters..58

4.1 The Effect of Population Size..58

4.2 The Effect of Changing Number of Generations Considered by Termination Criterion..59

4.1 The Effect of the Size Weight Used in Fitness Function 60

CONCLUSION AND FUTURE WORK..62

1. Conclusion..62

2. Future Work..65

REFERENCES..66

ABSTRACT IN ARABIC..71

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

LIST OF TABLES

Table 1. Data sets used in experiements 44

Table 2. The classification accuracy and size of reduced set of DROPs family for the 18 datasets compared to KNN..45

Table 3. The classification accuracy and size of reduced set of GRIBL and DROPs family..48

Table 4 The classification accuracy and size of reduced set of Seeded-GRIBL and 10 other reduction techniques..51

Table 5. The classification accuracy and size of reduced set of Seeded-GRIBL and DROPs family..53

Table 6 The classification accuracy, size of reduced set, and number of generations of GRIBL and Seeded-GRIBL..54

Table 7. The classification accuracy, size of reduced set, and number of generations of DROPs Seeded-GRIBL and Seeded-GRIBL 55

Table 8. The classification accuracy and size of reduced set of DROPs Seeded-GRIBL, Seeded-GRIBL and DROPs family..57

Table 9. The classification accuracy, size of reduced set, and number of generations of GRIBL using 10 and 50 individuals in the population..........................58

Table 10. The classification accuracy, size of reduced set, and number of generations of GRIBL using 5, 10, and 15 as sizes for the record used in termination criterion..59

Table 11. The classification accuracy, size of reduced set, and number of generations of GRIBL using 0.2 and 0.5 as a weight for the size in the fitness function..61

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

List of Figures

Figure 1: KNN Algorithm 9

Figure 2: The basic genetic algorithm...20

Figure 3: Tournament Selection 22

Figure 4: Single point crossover 24

Figure 5: Two points crossover 24

Figure 6: Binary Mutation..25

Figure 7: The data flow in GRIBL...32

Figure 8: GRIBL Algorithm 32

Figure 9: GRIBL population initialization algorithm 34

Figure 10: The data flow in Seeded GRIBL..41

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

Using Genetic Algorithms for Instance Reduction

BY
Eman Faris Mohammad Issa

Supervisor
Dr. Khalil M. el Hindi

ABSTRACT

Instance-based Learning algorithm (IBL) is a widely used inductive learning method that learns simply by storing instances of the problem. A new instance is classified by retrieving the most similar training instance(s) that is used to predict the class of the new instance. It has proven to be successful in terms of generalization accuracy for a wide range of real-world problem. However, to achieve good classification accuracy IBL requires storing a large number of training instances, which increases the classification time and memory requirements.

To avoid the excessive storage and long classification time, many instance reduction techniques were proposed in the literature. These techniques retain the most informative instances instead of the whole training set.

In this Thesis, the problem of instance reduction is considered as an optimization problem, which allows us to utilize genetic algorithms. Two genetically based reduction techniques were developed: Genetically reduced Instance-based Learning (GRIBL) and Seeded-GRIBL. The proposed techniques were tested over 18 benchmark real-world

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

datasets, and compared with the best-known reduction techniques in terms of size reduction and classification accuracy.

The developed techniques proved to compare favorably with other instance-based data reduction algorithms. Over eighteen real world problems Seeded-GRIBL achieved higher classification accuracy than the best-known reduction technique (which is DROP2) by 3.1%. This came at a slight cost of 0.6% (on average) increase in the size of the reduced set, compared with the same technique.

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

INTRODUCTION

1. Induction Learning

Computers provide the means for storing huge amounts of data in a form that allows fast random retrieval. In addition, to provide a convenient method for recording past events, this technology opens a new possibility; using historic data to aid in future decisions. Such tasks are trivial; the high computational power of computers opened new research areas that attracted much attention in the last few decades. These researches were directed to achieve systems that can metaphor human and animal learning to give the computer systems the ability to learn from experience.

Machine learning studies the ability of systems to improve over experience. Machine learning is any change in the computer system that causes an improvement in its performance (Simon, 1983).

In many fields what is really needed is to build a computer program that can learn from experience that is usually stored as set of examples in database. Inductive learning is a subfield of machine learning that is concerned with this issue (i.e. learning from examples). The user presents the system with a dataset of past cases (examples), together with a feature (target function) that it must learn how to predict. The system then uses this dataset to learn how to classify future examples. For example, a new patient is diagnosed by presenting his particulars to the system, which uses its knowledge base to predict the diagnosis.

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

2. Eager and Lazy Learners

There are many different types of Inductive learning approaches. They can be classified into two main categories: eager learners and lazy learners.

Eager learners, such as neural networks and decision trees, use the training examples to generate a classifier. This classifier is later used to classify new examples. They use the same classifier for all the unseen examples it may meet in classification time.

Eager learners produce global optimization of the target function that is used to classify any unseen example. Therefore, they need high learning time. However, the classification time is low.

On the other hand, lazy learners, such as Instance-based learners and Cased-based learners, store the training examples and perform most of their work at classification time. During training time, they simply store the training data without any further computation. Then at classification time, they generate classifiers for each unseen example; hence, they are suitable for applications that have complex target function that cannot be approximated by a single classifier. In effect, lazy learners find several simple local classifiers for each unseen example.

Lazy learners do not attempt to find approximation of a target function that can be used in general. They are good for applications with complex target functions but can be approximated using several local simple functions. However, they have some disadvantages such as their need for high storage requirements, large classification time, and that they use all the attributes of the examples in classification, which is unsuitable in some cases.

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

Neural networks were an early approach. They use numeric functions to weight each connection of the network. The user presents new examples to the system, causing the weights to alter. The final state of the output nodes determines the result. Although their usefulness is limited due to the difficulty of determining the best topology for a given problem, much research is still carried out in this area as it is thought that they learn in a similar way to neurons in the human brain (Fu,1994).

Induction of decision trees is another commonly used eager learners. C4.5 is a popular example that is often used as the benchmark for comparing new learning methods (Quinlan, 1986). The decision trees C4.5 induces, while not often intelligible to people, prove to be efficient classifiers. C4.5 has been used on wide variety of real datasets with much success, demonstrating a high degree of generality. There are many other similar rule-inducing systems, but they generate production rules instead of decision trees.

Psychologists studying the way that people use memory to perform tasks conclude that we often recall past experiences to guide us to solve to new problems. Instance-based learners do this by determining which case in memory is the most similar to the new situation (Kibler and Aha, 1987).

Instance-Based Learners (IBL) are "lazy" in the sense that they perform little work when learning from the dataset, and do most of the work at classification time. Unlike eager learners, IBL are incremental in the sense that they can use newly available examples without having to re-do any work. This gives them the freedom to learn over time, and so the set of instances in memory continues to grow. If allowed to learn indefinitely, the database eventually becomes too large to use, either because it exceeds memory capacity,

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

or because the time taken to classify new examples becomes prohibitively long. It is therefore desirable, sometimes even necessary, to prune (reduce the size) database.

Many reduction techniques were proposed in the literature; for a comprehensive survey see (Wilson and Martinez, 2000b).

In this thesis, we introduce two new reduction techniques. These techniques try to retain the strength of instance-based learners but at the same time solve the problem of storage requirement and slow classification. It employs the genetic algorithms optimization search, borrowed from the biological survival for the fittest theory, in finding the best set of instances in a dataset and discards the less relevant ones, maintaining a certain level of classification accuracy. The new technique uses genetic algorithms as a “front end” to traditional Instance-based learners in order to identify and select the best subset of examples to be used by the learner at classification time.

The techniques are discussed and empirically tested using many benchmarked datasets. They are also compared to other reduction techniques.

3. Structure of the Thesis

In chapter 2 of this thesis, we review the necessary background information needed in this work. It represents the Instance-based learning technique, discusses its strength and weaknesses, and reviews some of the reduction techniques used with Instance-based learning. It also reports the main concepts of Genetic algorithms and its applications in optimization problems.

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

Chapter 3 introduces Genetically Reduced Instance-based learning (GRIBL), a new reduction technique, and seeded-GRIBL, a version of GRIBL.

The experiments and results are reported in chapter 4. The conclusion and future work are presented in chapter 5.

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

INSTANCE-BASED LEARNING AND GENETIC ALGORITHMS

1. Introduction

Since the dawn of the computer age, researchers have been attempting to create computer programs that can improve their performance through experience. This intelligent behavior is the main goal of machine learning.

Many machine learning methods have been developed that can be categorized into broad categories of reinforcement, deductive, and inductive learning (Mitchell, 1997).

Learning what to do and how to map situations to actions to maximize a reward signal is called reinforcement learning. Unlike other types of learners, the learner is not told which action to take; instead it must discover which actions yield the most reward by trying possible actions. Reinforcement learning addresses the problem of learning control strategies for autonomous agents (Sutton and Barto, 1998).

Deductive learning is the process of reaching a conclusion that is guaranteed to follow if the evidence provided is true and the reasoning used to reach the conclusion is correct. The conclusion also must be based only on the evidence previously provided; it cannot contain new information about the subject matter .

Inductive learning methods, such as decision trees, rule induction, and exemplar-based learning, utilize examples of the problem, called a training set.

Though simple, exemplar-based learners proved to be competitive to more sophisticated learning methods, such as neural networks and decision trees, in terms of classification accuracy (Cost and Salzberg, 1993, Stanfill and Waltz, 1986, Hindi et al, 2003). These methods learn new concepts by storing past cases in such a way that new

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

examples can be directly compared with them. Based on this comparison, the similarity of cases (instances) is determined. The system then uses the most similar case(s) to predict the class of the new example. The learning methods included under this category differ from each other by the way they represent stored examples (i. e. representation method), and the similarity measure they use. There are different approaches of exemplar-based learning. Instance-based learning, (Aha et al, 1991), which uses a distance function to measure the similarity between the new instance and those in memory. Other approaches also exists such as case-based reasoning, memory-based reasoning and exemplar-based generalization (Stanfill and Waltz, 1986, Wettschereck and Dietterich, 1995),

Section 2 of this chapter provides a revision of instance-based learning. Section 3 presents a survey of some well-known instance reduction techniques. Section 4 presents the concepts of genetic algorithms and their applications.

2. Instance-Based Learning

Instance-based Learning algorithm (IBL) is a simple inductive learning algorithm. Unlike most learning algorithms, IBL does not construct an abstract hypothesis of the target function; instead it just stores the training examples (instances) and bases the target function approximation for the instance on the similarity between this instance and the stored instances (Aha et al, 1991).

The learning step simply requires storing the instances of the training set, with no further work on the generalization of the target function that is why IBL sometimes called lazy learners. Each instance is represented by an input vector x which consists of several attributes, and an output class c. During generalization, which is postponed until classification time, an unseen instance is classified by retrieving a set of similar training

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

instances and uses them to predict the class of the new instance. Therefore, IBL forms a local representation of the target function instead of a global one as most of machine learning methods do.

IBL has proven to be successful in terms of generalization accuracy over a wide area of real-world benchmark data sets. It is competitive to more sophisticated learning techniques such as neural networks in many applications (Cost and Salzberg, 1993, Stanfill and Waltz, 1986, Hindi et al, 2003).

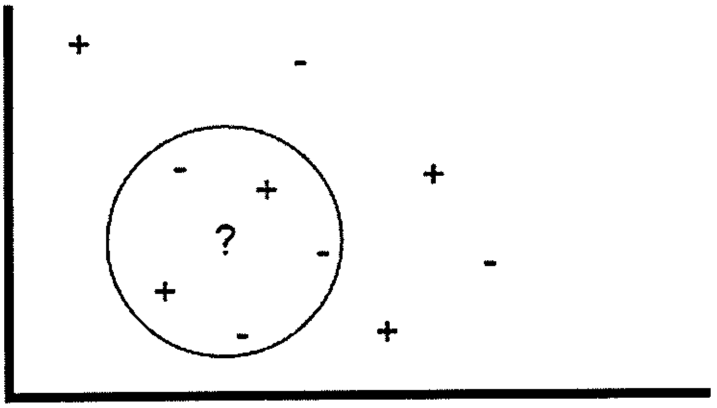
One of IBL characteristics is its ability to construct a different local approximation of the target function for each distinct unseen instance. This characteristic makes the IBL adequate for tasks where the target function is very complex but can be described by a collection of less complex local approximations.

Moreover, IBL can use more complex, symbolic representation of instances, which qualifies it to be used in many real-world learning tasks (Mitchell, 1997).

2.1 The K Nearest Neighbor Algorithm

The K Nearest Neighbor Algorithm (KNN) is a simple form of the IBL, (Aha, 1992, Aha et al, 1991). In its simplest form, KNN stores all classified instances in a training set *T* at the learning time. Then at classification time, it finds the K nearest instances and let them vote for the class of the unseen instance. The predicted class is the class with the majority of votes. The choice of K affects the predicted class as can be seen in figure [1], which represents a 2-dimensional space of instances where (+) represents a positive instance and (–) a negative instance. If K=1 the unseen instance, denoted by (?), will be classified as (+), depending on its nearest neighbor class. Where as it would be classified as (-), if K=5 since 3 of the 5 nearest neighbors holds the class (-).

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit



[Fig.1] KNN Algorithm
The figure represents a 2-dimintional space of instances where + represents a positive instance and – a negative instance

2.2 Distance Function

To measure the distance (similarity) between instances, KNN uses a distance function that measures the distance between two instances depending on the values of the different attributes.

The distance function used by the KNN was the Euclidean Distance function:

$$D(x, y) = \sqrt{\sum_{a=1}^{m} (x_a - y_a)^2} \qquad [1]$$

where x_a and y_a are the values of the attribute a in instances x and y respectively, and m is the number of attributes in the instances.

The Euclidean Distance is a commonly used function, but its use is limited to linear attributes. (i. e. attributes with numeric values that have an ordering relationship between them).

The Value Difference Metric (VDM) is a distance function that is appropriate for symbolic attributes such as color, shape, etc (Stanfill and Waltz, 1986),

$$vdm_a(x, y) = \sum_{c=1}^{C} \left| \frac{N_{a,x,c}}{N_{a,x}} - \frac{N_{a,y,c}}{N_{a,y}} \right|^q \qquad [2]$$

where:

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

- vdm_a in the distance between the values x and y of an attribute a.
- $N_{a,x}$ is the number of instances in T that have value x for attribute a.
- $N_{a,x,c}$ is the number of instances in T that have value x for attribute a and belongs to class c.
- C is the number of the classes in the problem domain.
- q is a constant, usually 1 or 2.

The Heterogeneous Value Distance Metric (HVDM) is a distance function that combines both the Euclidean Distance and the VDM (Wilson and Martinez, 1997),

$$HVDM(X,Y) = \sqrt{\sum_{a=1}^{m} d_a^2(X,Y)} \qquad [3]$$

where

$$d_a(x,y) = \begin{cases} vdm_a(x,y), & \text{if } a \text{ is symbolic else} \\ \dfrac{|x-y|}{a_{\max} - a_{\min}} & \text{if } a \text{ is numeric} \end{cases}$$

and a_{max}, a_{min} are the maximum and minimum values of attribute a.

There are many other extensions to the previous distance function that handles cases like encountering a nominal value in the unseen instance that does not exist in T. It also handles the missing values in the training set instances. Wilson and Martinez (2000a) propose distance functions that handle most of such cases.

2.3 Drawbacks of Instance-Based Learning Algorithms

As any other machine-learning algorithm, despite its valuable advantages, IBL has its drawbacks. As mentioned before IBL stores all the training set instances at learning time, which raises the need for large memory and causes the slow of classification process.

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

Classification accuracy achieved by IBL highly depends on the number of training instances stored at learning time. Storing too many instances can result in reducing the classification speed, since each instance would be visited to measure its similarity with the unseen instance, and it also can increase the memory needed. Therefore IBL algorithms are usually faced with the problem of choosing which instances to store to maintain a reasonable level of balance between the generalization accuracy and memory requirement and classification time.

These problems have been addressed in the literature using different methods: indexing techniques (Deng and Moore, 1995), and instance reduction techniques (Wilson and Martinez, 2000b).

Many reduction techniques were used to solve this problem. These techniques use different criteria to decide which instance to store for classification time. A survey of different reduction techniques will be introduced in section 2.3.

Curse of dimensionality is the second obstacle that faces the IBL algorithms, as well as many other machine learning approaches. When the number of attributes in the input vector is high, the probability of the presence of redundant and irrelevant attributes increases. These irrelevant attributes can mislead the classification of the machine learning algorithms, especially IBL algorithms. In such algorithms, the irrelevant attributes dilute the effectiveness of informative useful attributes in the distance function, therefore a misclassification may occur. At the same time the presence of a large number of attributes reduces the classification time.

Two techniques were used to overcome this shortcoming: feature selection techniques (Vafaie and De Jong, 1993), and feature weighting techniques (Wettschereck and Dietterich, 1997). In the former, the techniques used to choose the relevant features

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

(attributes), and exclude other attributes. The latter techniques weigh the features depending on their relevance to the learning task without excluding any.

3. Reduction Techniques

All machine-learning algorithms need a set of examples, negative and positive, to learn from. IBL uses these examples as prototypes to classify unseen examples. Most of the time, the larger the training examples the better generation accuracy achieved.

As mentioned in the previous section, large training sets require a large memory footprint, slow the execution time, and increase the sensitivity to noise. A technique is needed to determine how many instances to store for usage during generalization and what portion of space it should cover in order to avoid the excessive storage, time complexity, and maybe improve the accuracy by noise filtering.

Many reduction techniques were proposed in the literature; in this section we review some of the most widely used techniques, for a more comprehensive survey see (Wilson and Martinez, 2000b). Prior to that review a framework of the common aspects needed for discussing reduction techniques is presented.

The first aspect is the representation used in the reduction algorithm to represent the retained instances. The designer may choose to use a certain structure that represents a cluster of instances. We have introduced such structure in (Hindi et. al. 2004), where a prototype is used to represent a cluster of hand written digits instances with the same class. Other types of such clusters were introduced in the literature like hyprerectangles (Wettschereck and Dietterich, 1995), and rules (Domingos, 1996). The other common representation is to retain a subset of the original instances by removing less informative instances as in most of the instance reduction techniques.

603435

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

that CNN find S such that "***for every instance in T, the nearest neighbor in S is closer than its nearest enemy in S***", where the enemy is the nearest instance with a different class.

This algorithm does not guarantee a minimal subset of T. It is also sensitive to noise since S would always misclassify the noisy instances, hence, those instances would be added to it with a bad effect because it would cover more portion of the input domain.

3.1.2 Selective Nearest Neighbor Rule

In Ritter et al. (1975) the authors proposed an extension to the CNN algorithm, which overcomes the CNN drawback by ensuring that a minimal subset S would be found. This proposed method, named Selective Nearest Neighbor (SNN), handles the problem in CNN by updating the reduction condition such that "***for every instance in T, the nearest neighbor in S is closer than its nearest enemy in T.***"

This method is also sensitive to noise; it tends to maintain accuracy more than storage in the presence of noise.

3.1.3 Reduced Nearest Neighbor Rule

The Reduced Nearest Neighbor Rule (RNN), (Gates, 1972), is a decremental algorithm that starts with $S=T$ and removes any instance if doing so does not hurt the generalization accuracy. In other words, "***removes any instance if its removal does not cause any instance in T to be misclassified by the remaining instances in S***".

This method is able to remove noisy instances, thus, produces a subset of the CNN reduced set.

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

3.1.4 Edited Nearest Neighbor Rule

Wilson (1972) proposed a new decremental reduction algorithm called Edited Nearest Neighbor Rule (ENN) that initialize S by all instances in T and ***"removes from S any instance that is not classified correctly using the KNN algorithm".***

It removes noisy instances to leave smoother decision boundaries, and it does not reduce the memory requirement as much as other reduction techniques since it retains central points.

An extension of this algorithm is also presented that applies the ENN repeatedly until no further reduction is possible. This extension is called Repeated ENN (RNN).

3.1.5 Encoding Length

An encoding length heuristic was used in Cameron-Jones (1995) to measure how well S is describing T. This method consists of two phases. The first is the growing phase where each instance in T is added to S if that reduce the cost. The following phase is called the pruning phase in which the algorithm removes an instance if its removal lowers the cost. This method is also called ELGROW.

Explore is a method that applies ELGROW with its two stages, and then does a 1000 mutations hoping of improving the classifier.

3.1.6 Instance-Based learning algorithm2

An Incremental algorithm called IB2, or *Growth,* was introduced by Aha and Kibler (1987). This algorithm initializes *S* with an empty set. Then it ***"adds to S every instance in T that is not classified correctly by the instances already in S".***

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

3.1.7 Decremental Reduction Optimization Procedures (DROPs)

Wilson and Martinez (2000b) proposed a group of reduction techniques that takes into consideration the order of removal. The DROPs family work on a training set *T* contains n instances ($X_1..X_n$). A nearest enemy of an instance is the nearest instance with a different class. An instance's associate is the instance that has *X* in their *K* nearest neighbors.

The different DROP techniques are:

- **DROP1:** This technique is nearly identical to RNN, but the accuracy is checked in *S* instead of *T*. The algorithm ***"removes instance X if at least as many of its associates in S would be classified correctly".***

  Before removing any instance, DROP1 tests to see if removing *X* would degrade leave-one-out cross-validation generalization accuracy. If the results in the same level of generalization with lower storage requirements the instance is removed.

  This algorithm removes noisy instances, and instances in the center of the clusters to leave a non-noisy border instances. However, this algorithm is sensitive to the order in which the instances are removed; therefore, If the associates of a noisy instance were removed, that noisy instance would cover a large portion of the input space, and at that point when it is tested for removal a distant associate may be misclassified so the removal would be canceled.

- **DROP2:** An extension to DROP1 that handles the problem of noisy instance, mentioned above, by testing the affect of removal on the generalization accuracy

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

in T instead of S. That means it ***"removes instance X if at least as many of its associates in T would be classified correctly without X".***

This algorithm changes the order of removal of instances. It first sorts the instances depending on their distance to the nearest enemy. Then it removes those are farthest from their enemy; hence it removes the non-border points. From this point of view the noisy instances will be considered as border points. Therefore, if the noisy instance was in the center of the cluster, the algorithm will consider the central instances around it as border instances and would not be removed.

- **DROP3:** This member of the DROPs family overcomes DROP2 noise sensitivity by performing a noise-filtering pass. This filtering is done using a rule similar to the ENN, where any instance that is misclassified by its k nearest neighbors is removed. The second pass is applying DROP2. This yields smoother decision boundaries and immunity to overfitting.

- **DROP4:** An enhancement to DROP3 where a second condition is added to the noise filtering condition. It request that the removal of the noisy instance would not hurt the generalization accuracy, in order to limit the number of noisy instances removed in the filtering to a level that affords a good generalization accuracy.

- **DROP5:** An algorithm that modifies DROP2 by adding a noise reduction pass. In that pass the instances are removed depending on the distance to their nearest enemy from nearest to farthest. Then a typical DROP2 is applied.

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

4. Evolutionary algorithms

Optimization algorithms had received special interest in the last decade because of their ability to find approximate solution to NP-hard problems and problems where no analytic method exists.

Optimization algorithms are usually classified into two categories. The first category is the deterministic local search algorithms such as Steepest Descent, which usually stuck at local optima. That happens when the optimization problem has multiple local optima or when the search space is huge enough not to be able to define exact local optima. The other alternative is stochastic search such as Simulated Annealing, Tabu Search, and Evolutionary Algorithms (Louis 1993, Bäck T. 1996).

Among the different stochastic algorithms known, evolutionary algorithms propose the most promising solution for optimization problems. It can be applied to a wide area of problems and not restricted to certain applications.

The most commonly used type of evolutionary algorithms is Genetic Algorithms (GA) (Goldberg, 1989, Davis, 1991). GA is considered as a model of machine learning, which derives its behavior from a metaphor of the processes of evolution in nature. This is done by the creation within a machine of a population of individuals represented by chromosomes. Then, individuals in the population go through a process of evolution (comp.ai.genetics FAQ).

John Holland, from the University of Michigan, was the pioneering founder of much work in genetic algorithms. The first achievement was the publication of Adaptation in Natural and Artificial System in 1975 (Holland, 1975).

Genetic Algorithms inspire Darwinian survival for the fittest theory in searching the problem domain by evolving a population of solutions until a certain level of goodness (fitness) is achieved. Unlike most stochastic search methods, GA operates on

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

population of solutions instead of a single solution. It is a very simple optimization algorithm, yet it performs well on many different types of problems.

Typically GA maintains a population of individuals (chromosomes). Each individual has a fitness value and consists of a number of genes, where every gene represents a certain optimization characteristic, and together represents a solution for the problem. After creating the population, GA iteratively evolves it to a better population.

4.1 Basic Concepts

The population is usually initialized with randomly created chromosomes (solutions), which represent different solutions for the optimization problem. After initializing the population, the evolution process starts by iterating over several generations. During each successive generation, each individual is evaluated and a value of goodness or fitness is returned by a fitness function. Then the evolution continues until some termination criterion is met. The termination criterion varies from application to another; it may be a condition that a certain level of fitness is achieved or a certain number of iterations are performed.

Inside the evolution loop three main operators are applied (Goldberg, 1989, Miller et al., 1993, Grefenstrette et. al., 1989, Mitchell, 1996):

(1) Selection, where individuals with the better fitness are more likely to survive to the next generation.

(2) Recombination (cross over), where two parents are crossed over to create one or two children.

(3) Mutation, which alters the chromosomes to create new individuals.

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

The algorithm in figure [2] shows the basic operations used in Genetic algorithms.

Initialize population of individuals *P* randomly
Evaluate fitness of all individuals in *P*
Test for termination criterion (time, fitness, etc.)
While termination condition is not met do

1. P_{new} ← select the sub-population that will be passed to the next generation

2.*Parents* ← select a sub-population to participate in crossover to produce two offspring

P_{new}← P_{new}+ recombination of the "genes" of selected *parents*

3. Mutate P_{new} stochastically

4. P←P_{new};

[Fig. 2] The basic genetic algorithm

The basic algorithm can be modified in many ways depending on the optimization problem, and many parameters can be tuned to obtain the desired results. In general, if you choose the appropriate fitness function, the right representation, and the suitable operators, then the variations in the algorithm or on the parameters will have minor effect on the results.

4.2 Representation (Encoding)

The first decision must be taken before using Genetic algorithms is to determine the representation scheme to encode solutions of the optimization problem.

Different techniques for encoding chromosomes were used. Some of these techniques use high-level problem representation and implements specialized crossover and mutation operators. Such techniques use trees, lists, objects, or any other data

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

structure to encode solutions. However the simplest and most frequently used representation is the Binary representation.

Binary encoding is the traditional way to represent parameters in GA. The data structure used is a bit-vector with length L, where L is equal to the number of parameters and 2^L is the number of possible solutions.

4.3 Population Initialization

The initialization of the population specifies the starting point of the search. Many approaches were used to setup the population. One of the most commonly used approaches is the random initialization using uniform distribution in order to proceed from an unbiased sample of the search space. Another approach works by scattering the search space into regular grid-layout and generate a chromosome that represent a square in the grid. Furthermore, the domain knowledge can be incorporated to create chromosomes that represent already known solutions, (Louis, 2003). Louis and Johnson (1997), proposed a technique that seeds the initial population with solutions of similar previously solved problems, which can reduce the time taken to find a quality solution.

4.4 Selection

Selection is an important stage in the new population evolving (Whitley, 1989). In this stage the individual that does not serve the desired solution or have a low fitness value is discarded. In other words, it maintains the good individuals in order to keep a certain level of fitness. This way GA, generation after another, directs the search into the promising areas in the solution domain.

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

The selection operator picks a certain percentage of individuals and passes them to the next generation. These individuals are usually chosen probabilistically depending on their fitness.

The determination of the selected percentage of individuals, also called survival rate, is an important aspect of the evolution process. A large survival rate could direct the algorithm to converge into a small area of the solution space. On the other hand a small survival rate slows down the convergence process.

The selection technique should be biased to good individuals, but it should also pick some less good individuals to guarantee that the population will not quickly converge to a local optima solution.

Different techniques are used for selection. Here we list 2 of them.

1. Tournament Selection: This technique holds a tournament of *K* random individuals and copies the one of the best fitness among these K individuals to the next generation. The tournament length K is usually equals to two or three and is rarely above 5. Figure [3] shows the psuedocode for the tournament selection.

   Tournament selection is the most commonly used selection technique because it is simple, does not need long computation time, and gives good results.

```
Tournament selection (P) // P is the current population
        j=0;
        While j< (length (p)* survival_rate) do
        {
         Pick K random individuals I1,...,IK from P;
         Compare the fitness of the picked individuals;
         Insert a copy of the fitter individual into Pnew; // Pnew is the evolved  Population
          j++
        }
```

[Fig.3] Tournament Selection

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

2. Roulette Wheel Selection: It is also called proportional selection because individuals are given a probability of being selected that is directly proportional to their fitness. The probability is calculated by dividing the fitness of the individual by the total fitness of the population.

4.5 Crossover (Reproduction)

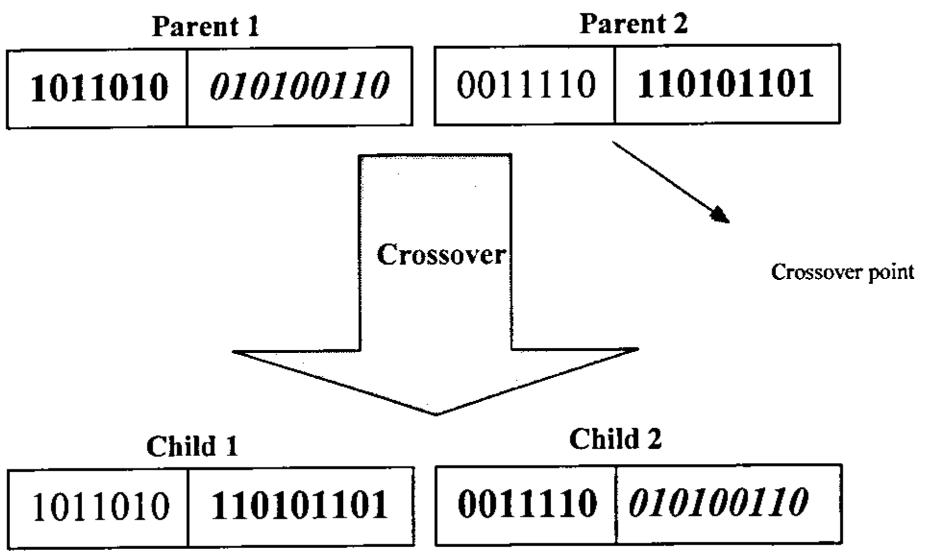
Crossover is a genetic operator that is used to add variation to chromosomes from one generation to the next one. Actually, the evolution process without crossover turns to a copying process that duplicates individuals without any enhancements on the new individuals (Qi and Palmieri, 1993). It is an analogy to the biological reproduction. The crossover process starts with two parents independently selected according to a probability distribution that takes their fitness into consideration. It produces two new offspring, where each offspring contains some of the genetic materials of each of its parents. The two offspring are usually different from their two parents and from each other. In the new generation the two offspring could be considered or only the fittest is included and the other one is discarded.

The technique used to select the parents that will participate in the crossover is usually the one used for selection.

There are different crossover techniques that can be used:

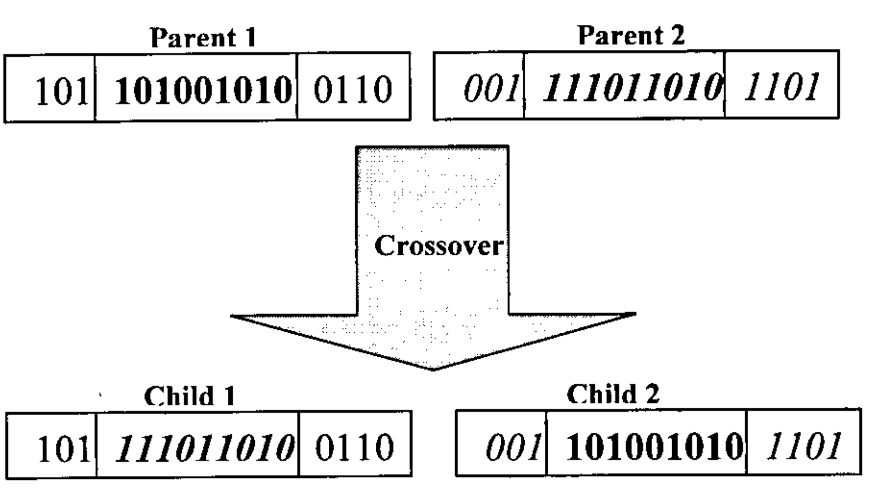
1. Single Point Crossover: The most common cross over technique, where a point in the chromosome is randomly selected, and the genes beyond that point is swapped between the two parents to produce two children, as shown in figure [4].

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit



[Fig.4] Single point crossover

2. Two Point Crossover: Two points are selected in the chromosome. Everything between the two points is swapped between the two parents to produce two children, as shown in figure [5].



[Fig.5] Two points crossover

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

4.6 Mutation

Crossover is the basis of genetic algorithms; there is nevertheless, another important operator, which is mutation. In fact, the desired solution may not be present inside a given population, even if it is a large population. Mutations allow the emergence of new genetic configurations, which improve the chances to find the optimal solution (Bäck, 1993).

The mutation process allows new individuals to be created. It begins by picking an individual, depending on its fitness, and then randomly chooses a gene and changes it. In Binary representation, this happens by choosing a random bit and flipping it, as shown in figure [6].

Before Mutation	0101001100000011001
After Mutation	0101001100100011001

[Fig.6] Binary Mutation

4.7 Fitness Function

The evolutionary process is driven by the fitness measure used. The fitness measure assigns each chromosome a fitness value that quantifies the optimality of a solution in a genetic algorithm, so that a particular chromosome may be ranked against all the other chromosomes. Relatively optimal chromosomes are allowed to breed and mix their datasets by any of several techniques, producing a new generation that will (hopefully) be even more optimal (Smith et. Al., 1993).

An ideal fitness function correlates closely with the algorithm's goal. Therefore it is important to choose a suitable fitness function. In most numarical problems, the fitness function is explicitly given by a mathmatical equation. However, in problems

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

that are not well-defined, the designer of the GA should make sure that the choosen function properly ranks the individuals so that the most desirable solution is assigned the best fitness. Otherwise, selection operator will choose the worng individuals when forming the next generation.

4.8 Applications of Genetic Algorithms in Machine Learning

Genetic Algorithms have been used in a wide variety of optimization tasks (Grefenstette, 1987), including numerical optimization, and combinatorial optimization problems such as the traveling salesman problem (TSP) (Louis, 1999), circuit design, job shop scheduling (Goldstein, 1991), planning, induction of decision trees for classification, and other optimization tasks related to machine learning. Additional information and examples can be found in Koza (1992).

Moreover, Genetic algorithms were employed to improve the behavior, to handle the drawbacks, and to solve the problems of some learning algorithms. Handling the learning algorithms weaknesses is an optimization problem after all.

Two examples of how GA was employed in machine learning are summarized below.

- ***Using GAs in Feature Selection and Weighting***

Most machine learning algorithms are sensitive to irrelevant attributes. Before the classification process the algorithm should determine the useful subset of features (attributes), to be used in the classification process, from a larger set of mutually redundant, possibly irrelevant attributes.

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

Yang and Honavar (1997) explore a wrapper-based multi-criteria approach for feature subset selection using a genetic algorithm in conjunction with a relatively fast inter pattern distance-based neural network learning algorithm.

They have represented the input pattern attributes vector with a binary string in which each bit corresponds to an attribute. The fitness function, that controls the generation evolving process, is determined by evaluating the neural network using training set whose input patterns are represented using only the selected subset of features.

They have examined the combined approach on 10 datasets, 9 real datasets and an artificial dataset, which was used to explore the feasibility of using genetic algorithms for the addressed problem. The generalization accuracy achieved by the neural network constructed using the GA-selected subset of attributes was remarkably increased compared to that achieved by the neural networks constructed using the original set of attributes.

Wilson and Martinez (1996) address the same problem of redundant and irrelevant attributes by using attribute weighting to lessen the influence of such attributes. They proposed a system that combines genetic algorithms with instance-based learning; the system is called Genetic Instance-Based Learning (GIBL).

The system uses the GA to guide the search in the weight space, and IBL to evaluate each combination of feature weights and determine its fitness.

The GIBL system uses a real values vector representation for the individual's chromosomes, where each real valued gene represents a weight for a certain attribute. These chromosomes formulate the population, which consists of 40 individuals, and initialized almost randomly, a vector of ones is also included as a default setting for the attribute weights.

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

At each iteration of the genetic revolution, the whole generation is replaced by new individuals created via recombination (crossover). Two operations were used: crossover and mutation. A percentage of 30% of the population were reproduced by the crossover operator. The parents that will participate in the crossover operation are selected probabilistically. The rest of the population is produced by mutating the original individuals. The mutation percentage is set to 50%, which corresponds to the possibility that each gene would be mutated.

The fitness value of a chromosome in GIBL system represents the classification accuracy when using the weights in that chromosome.

The GIBL system was tested on 16 datasets using 10-fold cross validation. In each fold the training set is used to find the chromosome with optimal attributes weights, then those weights were used during the classification of the test set.

The results reported in the paper shows that GIBL system gave slightly higher classification accuracy on regular data sets, and significantly higher classification accuracy with datasets with irregular and redundant attributes.

➢ ***Using GAs in clustering***

Unsupervised classification is a type of pattern classification technique that was frequently addressed; one example of those techniques is *Clustering*. In Clustering, sets of similar patterns are grouped in a cluster. The definition of the similarity between patterns is the main task of the clustering technique. Then the technique starts with an initial cluster centers and searches a very complex space in order to find the best possible cluster centers.

Many earlier versions of clustering techniques were proposed during the last decade. One of the simplest and most frequently used is the *K-mean* algorithm. Maulik

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

and Bandyopadhyay (2000) propose a new technique that derives the K-mean simplicity. At the same time employs the capabilities of the GAs to search through complex spaces, its implicit parallelism, and its ability to provide good results irrespective of the starting configuration to avoid local optima, where K-mean may stuck at.

The GA-clustering algorithm proposed Maulik and Bandyopadhyay used to appropriately determine a fixed number of cluster centers. They have followed the basic steps usually used in different GAs optimization tasks, such as using floating-point representation of chromosomes, defining the clustering metric and using the inverse of it as a fitness function, Roulette wheel selection, crossover, and mutation.

The experimental results presented, provided for four artificial data sets and three real-life data sets, show in general that the GA-clustering technique performs more uniformly than the older clustering algorithm (K-mean). Moreover it didn't exhibit any unwanted behavior regarding sub-optimal solutions where the K-mean may face a problem. It was clear that the GA-clustering fitness results were usually close to the best value in different initial populations. It provided a performance that is significantly superior to that of the K-means algorithm for the data sets considered.

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

A Genetic Algorithm Approach for Instance Reduction

1. Introduction

Instance-Based Learning techniques are widely used for different classification problems especially when the target function is hard to be represented by a single classifier (Aha et al, 1991). Therefore, IBL proved to be competitive in terms of classification accuracy to more complicated learning techniques such as neural networks in many applications (Cost and Salzberg, 1993, Stanfill and Waltz, 1986, Hindi et al, 2003).

Instance-Based learners are able to learn quickly from a very small dataset. Whereas other induction methods require reasonable number of examples before they can induce, IBL can begin to make useful predictions from as little as one example per class.

However, in many applications, to achieve reasonable classification accuracy large number of instances is needed, which not only increases the memory requirements but also slows the classification process. The large number of instances stored increases the classification time simply because every new instance needs to be compared with a large number of instances before the nearest instance is found.

To resolve the problem of large training set stored by IBL, many instance reduction techniques were proposed in the literature (see section 2.2 for a review of such techniques). Instance reduction techniques reduce the number of stored instances, but unfortunately, this usually reduces the classification accuracy.

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

In this chapter, new reduction techniques are presented. These techniques, discussed in section 2 and 3, consider the problem of dataset size as an optimization problem, permitting the use of GAs to find a reduced set that is informative enough to represent the original training set.

2. Genetically Reduced Instance-Based Learning System

Genetic algorithms are used in this thesis to find a good subset of training instances that is informative enough to represent the original training set. There are two criteria to be optimized: the classification accuracy and the size of the reduced set.

The proposed technique, Genetically Reduced Instance-Based Learning (GRIBL), uses genetic algorithms to search the space of all possible subsets of the original dataset. The technique balances between the exploration of the search space, using crossover and mutation, versus the exploitation of particular areas of the space, using selection operators.

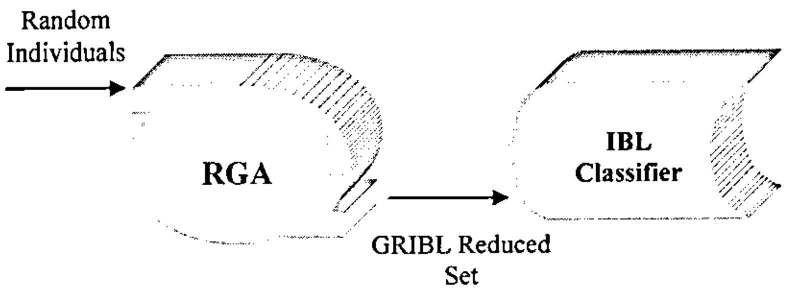
The evolution process starts with a population of different subsets of a dataset and continues for several generations (applying the different genetic operators) until no further improvement can be achieved in the fitness. Fitness is measured by a certain function that takes into consideration both the classification accuracy of the subset and its size.

At the end of the evolution process, a whole population of individuals becomes available. Each individual is a subset of the training set that is hopefully informative enough to represent it; hence, it represents a reduced set of the original dataset that is

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

expected, hopefully, to perform as well as the original in terms of classification accuracy but with lower number of instances.

The best individual in the population is the outcome of GRIBL, which is passed to the IBL system to use it in classification. Figure 7 shows the data flow in GRIBL.



[Fig.7] The data flow in GRIBL

The GRIBL algorithm is shown in fig [8]

GA_Reduction (*T*)

// where T is the Training Set

1. *P*= InitializePopulation(*k*); ***where k is the number of individuals in the population***

2. Evaluate each individual in *P* using a random subset of *T*

loop

3. *Best* ← find the fittest chromosome in P

4. *Pnew* ←Apply the tournament selection with k=3 to 20% of *P*

5. *Pnew* ←Apply single point crossover to 40% pairs of the population chosen using tounament selection

6.*Pnew* ← Apply Bit-Flip mutation to 20% of *P*

7. Evaluate each individual in *Pnew* using *T*

8. *P* ← *Pnew*

until (termination criterion is met)

return *Best*

[Fig.8] GRIBL Algorithm

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

2.1 Individual Representation

Each chromosome (individual) in GRIBL population represents a candidate reduced set of the training set. Each gene in the chromosome represents an instance in the training set. Binary representation is used to encode the chromosomes. A chromosome is represented by a binary string of length **L**, where **L** is the number of instances in the training set. Each binary bit represents agene and corresponds to an instance in the original training set. If the bit is on, (i. e. set to one), then the corresponding instance is a member of the chromosome; otherwise, the instance is not a member.

2.2 Population Initialization

In GRIBL, initial population consists of *k* individuals. Each individual represents a randomly selected set of instances. The individual is initialized with randomly generated binary values (0,1).

To achieve a good level of diversity, the population is initialized with individuals of different sizes. The range of the size is between 20% and 70% of the original training set size. This produces individuals from different areas of the domain space to maximize the portion of the space represented by the population.

To determine whether an instance is in an individual (a reduced set) or not, a random number between 0 and 1 is generated and compared to the threshold value (that

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

varies between 0.2 and 0.7). if the random number is greater then the gene value is set to 1, otherwise to 0.

The population initialization process used in GRIBL is shown in fig [9].

```
InitializePopulation(k)
    For each individual do
        Generate a random threshold value between 0.2 and 0.7
        For each instance in the training set do
        {
          X= a random number
          If x<= threshold
              Set the corresponding gene to 1
          else
              Set the corresponding gene to 0
        }
```

[Fig.9] GRIBL population initialization algorithm

2.3 Evolution control

After initializing the population of chromosomes the evolution process starts by iterating over several generations. During each generation, the fitness of each individual is evaluated using a fitness function. Then the evolution continues until some termination criterion is met. The termination criterion may be a condition that a certain level of fitness is achieved, a certain number of iterations are performed, or no further improvement in the fitness is achieved.

In GRIBL the evolution process continues as long as an improvement in the fitness is achieved in the last generation. This is tested by comparing the best chromosome, in terms of fitness, in the last generation with the best in the previous one. However, if no improvement is detected in the new generation that does not necessarily mean that

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

Populations in GRIBL contain the three types mentioned above. Therefore, it uses three types of operators: Selection, Crossover, and Mutation.

- **Selection Operator:**

The selection operator is used to choose the individuals with a good fitness and pass them to the next generation. This allows the individuals with acceptable fitness to survive for more than one generation; hence, the good genes will not fade away. This ensures that a minimum level of goodness will be maintained through the evolution.

The Tournament Selection, used in GRIBL, is a technique for choosing the surviving individuals. A tournament of 3 random individuals is held and the fittest one is passed to the new generation (Whitley, 1989).

To avoid losing the best individual in the previous generation, it is passed automatically to the new generation.

- **Crossover Operator:**

The new individuals in the generations represent new areas in the search space. The presence of these individuals supplies populations with chromosomes holding new gene combination, which increases the possibilities of exploring new areas in the search space and discovering new candidate solutions.

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

In GRIBL we used single-point crossover operator, where a point in the chromosome is randomly selected, and the genes beyond that point is swapped between the two parents to produce two children (Qi and Palmieri, 1993).

The crossover rate is set to (0.8), since the selection rate is (0.2), which implies that 80% of the produced generation are new individuals while 20% are copied by selection. The number of crossover processes equals to half the number of children (new individuals), since each crossover process produces two individuals.

The selection of the parents that will participate in the production of each couple of children is an important part of the crossover process. Therefore, tournament Selection is also used for choosing the parents. Two tournaments of 3 random individuals each are held and the fittest one in each is considered as parent and participates in the crossover process.

- **Mutation Operator:**

After the selection and crossover processes a new population full of individuals would be available. Some of these individuals are directly copied and others are children of the crossed over parents. In order to allow the emergence of new genetic configurations that improves the chances to find the optimal dataset, mutation operator is used. In some cases, when all the individuals become very similar, mutation is the only way to explore other areas of the search space.

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

The Bit-Flip mutation is the mutation operator used in GRIBL. This mutation operator randomly selects an individual from the current generation, and flips a random gene in that chromosome (Bäck, 1993).

The number of individuals mutated in each generation depends on the mutation rate. The rate in GRIBL is (0.2), i.e. around 20% of the population is mutated.

2.5 Fitness Function

Choosing an appropriate fitness function is a step of an extreme importance for successful applications of GAs to any problem domain. The fitness function measures the quality of the individuals, which affects the decision of selecting the individuals those will be copied to the new generation, or participates in the crossover process. This step is more difficult and important for instance reduction problem we are tackling, simply because the fitness function must make good balance between two factors: classification accuracy and size of the reduced set.

The fitness function used in GRIBL was adopted from a formula proposed by Nunez (1988), which was used to balance the information gain achieved by using a specific attribute and its cost, in building decision tree. The original formula is,

$$\frac{2^{Gain}-1}{(Cost+1)^{w}}$$

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

where w is a constant between 0 and 1, which determines the relative importance of cost versus information gain.

In GRIBL, the Information gain corresponds to the classification accuracy gained from using an individual, and the size is the cost of using it.

$$\frac{2^{Accuracy} - 1}{(size\_ratio + 1)^{w}}$$

The constant w varies among the different versions of GRIBL, but in most of the versions it is set to 0.2 since the classification accuracy is more important.

2.6 Individual Classification Accuracy

For each individual in the population during the evolution process a fitness value is assigned. This fitness value depends on the individual classification accuracy.

The individual classification accuracy is measured by classifying a randomly chosen group of instances from the original training set using the chromosome instances. Each individual in the population is used to classify the randomly chosen group using KNN algorithm, where k=3. The individual classification accuracy is the ratio of the number of correctly classified instances to the number of the instances used in the test (the size of the group).

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

3. Seeded Genetically Reduced Instance-Based Learning System

The initial experiments showed that GRIBL takes a large number of generations before terminating and returning the fittest reduced set, which takes alot of time.

As was mentioned in section 3.2, GA search makes a balance between the exploration of the search space, using crossover and mutation, and the exploitation of particular areas of the search space, using selection operators. In some cases, time and effort is wasted to maintain this balance by exploring areas in the search space where no optimal solutions are available.

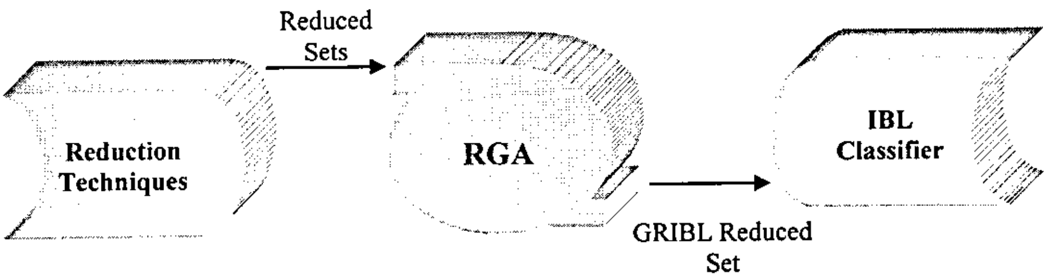
The random population initialization may mislead the search for multiple successive generations, where an era of generations could be wasted before an acceptable population is presented. This may cause the GRIBL technique to iterate for a considerably large number of generations before finding reasonably fit individuals.

Louis and Johnson (1997) proposed a technique that uses the idea of case-based reasoning (CBR) to seed the initial population with solutions to similar previously solved problems. They have suggested to seed the initial population with previous GA search solutions, or use some analytical information collected in these searches to control the current one.

Louis and Johnson idea inspired a modification that may solve the time problem of the original GRIBL. The initial population is seeded with solutions yielded by other reduction techniques. There are many reduction techniques used for IBL training set size reduction, (Wilson and Martinez, 2000b), we use the reduced set returned by some of these

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

techniques as solutions to initialize the population. The data flow in the suggested technique shown in fig [10].



[Fig.10] The data flow in Seeded GRIBL

Seeding the initial population with individuals that represent reduced sets obtained using other techniques solutions, gives the system a head start, enabling it to converge to a good reduced set in less number of generations. It also helps the algorithm to avoid the local optima that GRIBL may fall in simply because it consider search in areas that probably contains the global optima or at least good local optimas that are close to the global one.

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

EMPIRICAL WORK

1. Introduction

In chapter 3, two new reduction techniques were proposed: GRIBL and Seed-GRIBL. These techniques are used to reduce the training set size by deciding the most informative subset that can be retained instead of the original dataset.

Both techniques, GRIBL and Seeded-GRIBL, employ genetic algorithms to search for the best subset of instances that would act well in classifying unseen instances; they use genetic operators to explore more area of the search space in order to ensure that only instances with minimal effect on the general classification accuracy are discarded.

In all experiments in the following sections 10-folds cross validation is used. Each dataset is randomly divided into 10 separate partitions of the same size. At each iteration, 9 different partitions (90% of the dataset instances) are considered as training set, *T*, and the remaining partition is used as a test set *S* (10% of the dataset instances).

At each fold, GRIBL takes *T* as input, initializes the population depending on the size of *T*. *T* is also used to evaluate the fitness of each individual in the population for evolution purposes. Then when GRIBL finishes, it uses *S* in testing the classification accuracy of the reduced set. The classification accuracy for the reduced set is measured using the KNN algorithm where k= 3. The classification accuracy is found by calculating the average accuracy of the ten folds.

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

The techniques proposed were implemented using MATLAB7 development tool. MATLAB is widely used in scientific and technical computing, development, and programming since it provides a wide collection of supporting tools for different fields.

The datasets are partitioned and stored in text format. The partitions for each fold are stored in one file. Different old reduction techniques were applied to these files, and then the reduced set is stored to be used by Seeded-GRIBL.

The GRIBL and seeded-GRIBL were tested using 18 benchmark real-world datasets. These datasets were obtained from the machine learning data repository available from the University of California at Irvine, http://www.ics.uci.edu/AI/ML/MLDBRepository.html.

Table 1 gives further details on each of the datasets such as the number of attributes, the number of examples (instances), and the number of classes.

Table 2 shows the classification accuracy and percentage of reduced set size to the original training set size of the DROPs family. These algorithms will be used next in the statistical test of the proposed techniques. The last two columns show the best reduction technique in terms in accuracy, and size reduction

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

Table 1. Datasets used in experiments

DataSet	Number of instances	Number of attributes	Classes
Breast-cancer-wisconsin	699	9	2
Bridges	106	11	7
Echocardiogram	74	9	2
Flag	194	28	8
Glass	214	9	7
Heart	270	13	2
Heart.Long-beach-va.2	200	13	5
Heart.cleveland.2	303	13	2
Heart.hungarian.2	294	13	2
Heart.swiss.2	123	13	5
Hepatitis	155	19	2
Horse-colic	301	23	2
Iris	150	4	3
Liver.bupa	345	6	2
Pima-indians-diabetes	768	8	2
Promoters	106	57	2
Wine	178	13	3
Zoo	90	16	7

In section 2, we discuss the details of GRIBL technique, report the experiments, and discuss the results. In section 3 two versions of Seeded-GRIBL are represented with the experiments and results for both.

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

Table 2. The classification accuracy and size of reduced set of DROPs family for the 18 datasets compared to KNN

DataSet	KNN		DROP1		DROP2		DROP3		DROP4		DROP5		Best acc	Best size%
	size%	Acc	size%	Acc	size%	Acc	size%	Acc	size%	Acc	size%	Acc		
Breast-cancer-wisconsin	100	0.961	1.70	0.970	5.31	0.963	3.05	0.966	3.15	0.967	2.93	0.966	DROP1	DROP1
Bridges	100	0.688	23.58	0.472	27.58	0.496	15.79	0.471	25.68	0.506	20.42	0.438	DROP4	DROP3
Echocardiogram	100	0.920	12.99	0.932	14.33	0.932	14.93	0.932	14.93	0.932	13.43	0.932	DROP1	DROP1
Flag	100	0.707	21.43	0.686	28.11	0.686	21.89	0.687	25.60	0.686	24.23	0.670	DROP3	DROP1
Glass	100	0.690	20.49	0.541	26.23	0.636	18.20	0.587	23.83	0.646	23.28	0.603	DROP4	DROP3
Heart	100	0.841	9.55	0.807	17.65	0.822	11.44	0.848	12.30	0.837	11.98	0.811	DROP3	DROP1
Heart.Long-beach-va.2	100	0.710	8.50	0.640	11.56	0.715	4.11	0.580	6.72	0.545	7.39	0.550	DROP2	DROP3
Heart.cleveland.2	100	0.809	11.14	0.766	18.28	0.802	11.32	0.825	13.11	0.815	13.85	0.828	DROP5	DROP1
Heart.hungarian.2	100	0.775	12.92	0.826	16.89	0.802	8.79	0.813	12.31	0.823	15.68	0.823	DROP1	DROP3
Heart.swiss.2	100	0.937	2.70	0.962	6.31	0.951	3.42	0.962	3.42	0.962	2.97	0.968	DROP5	DROP1
Hepatitis	100	0.787	6.00	0.799	12.57	0.813	4.07	0.481	5.64	0.616	6.14	0.337	DROP2	DROP3
Horse-colic	100	0.731	6.83	0.449	16.64	0.734	1.99	0.627	8.01	0.545	9.23	0.472	DROP2	DROP3
Iris	100	0.953	9.63	0.960	15.48	0.947	14.74	0.940	14.96	0.940	12.74	0.933	DROP1	DROP1
Liver.bupa	100	0.617	25.16	0.595	34.84	0.618	22.65	0.609	28.97	0.621	26.42	0.591	DROP4	DROP3
Pima-indians-diabetes	100	0.720	16.90	0.707	25.12	0.712	14.59	0.738	18.93	0.702	18.38	0.724	DROP3	DROP3
Promoters	100	0.943	7.79	0.875	15.05	0.847	14.42	0.903	14.74	0.903	10.21	0.887	DROP3	DROP1
Wine	100	0.961	8.94	0.950	15.31	0.950	14.69	0.950	14.69	0.950	9.38	0.961	DROP5	DROP1
Zoo	100	0.944	19.14	0.900	20.25	0.822	19.51	0.811	21.23	0.800	19.51	0.711	DROP1	DROP1
Average	100.00	0.816	12.52	0.769	18.20	0.792	12.20	0.763	14.90	0.766	13.79	0.734	DROP2	DROP3

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

2. Experimenting with GRIBL

The GRIBL technique, as discussed in the previous chapter, employs genetic algorithm concepts to optimize the size of training set and classification accuracy combination in IBL.

It starts by initializing a population of size 10 random individuals. Each individual corresponds to a reduced set. The number of 1's in the individual indicates the number of instances in the suggested reduced set. To ensure a diverse population, individuals were initialized to represent reduced sets with different sizes. This was done by changing the threshold used by the algorithm to decide whether to include or exclude an instance. Different individuals vary in size between 20- 70% of the original dataset size.

After initializing the population, each individual is evaluated using the fitness function discussed in section 3.2.5. The individual accuracy is measured by applying KNN algorithm, with k=3, on 20% randomly chosen instances of the original training set using the individual instances. Testing over this random subset aims to improve the efficiency of evolution process. Increasing the size of the sample may improve the obtained results but that would increase evolution time.

During each generation, the GA operators are applied. Tournament selection, with a tournament of size 3, is used to choose 20% of the current generation that is copied to the next one, making sure that the best individual is one of the passed individuals. The remaining 80% of the new generation are produced by single-point crossover, which recombines two good parents, selected with the same selection operator, to produce 2 new individuals. Bit-flip mutation is then applied to 20% of the new generation individuals.

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

Once a new generation is formulated, the individual with the best fitness in that generation is found. Then it is compared to every "best individual" obtained in the last 10 generations, if it is not better than anyone of them the evolution process is terminated. Then, when GRIBL exits the evolution loop, it returns the individual with best fitness as the reduced set.

Table 3 shows the classification accuracy and the size of the reduced set achieved by applying GRIBL and the 5 DROP techniques over the 18 datasets. Recall that we are using 10-fold cross validation, so the classification and size figures in the table represent the average obtained in the 10 experiments. The table also shows the average accuracy, average size, technique that achieved the best accuracy, and the technique that achieved the best size reduction for each dataset.

Compared with the best-known reduction techniques, GRIBL achieved a higher average accuracy by 0.6%. However, the cost of that improvement was an increase in the size of the reduced set by 29.9% (on average).

A statistical significance test with 95% confidence level was applied to results. The test compares the significance of GRIBL classification accuracy with the best DROP technique (which is DROP2). The test showed that GRIBL's classification accuracy was statically significant for 11 datasets (marked in table 3 with a +), and not statically significant for 5 datasets (marked in table 3 by a -).

Several factors might have contributed to this result:

1. We used a small number of individuals in the initial population (i. e. 10 individuals). A larger number of individuals is expected to improve the results.

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

Table 3. The classification accuracy and size of reduced set of GRIBL and DROPs family

DataSet	KNN		GRIBL		DROP1		DROP2		DROP3		DROP4		DROP5		Average Acc.	Average size%.	Best acc	Best size%
	size%	Acc	size%	Acc	size%	Acc	size%	Acc	size%	Acc	size%	Acc	size%	Acc				
Breast-cancer-wisconsin	100	0.961	33.62	**0.967+**	1.70	0.970	5.31	0.963	3.05	0.966	3.15	0.967	2.93	0.966	0.966	8.293	DROP1	DROP1
Bridges	100	0.688	57.23	**0.612+**	23.58	0.472	27.58	0.496	15.79	0.471	25.68	0.506	20.42	0.438	0.477	28.381	GRIBL	DROP3
Echocardiogram	100	0.920	21.17	**0.932**	12.99	0.932	14.33	0.932	14.93	0.932	14.93	0.932	13.43	0.932	0.932	15.295	GRIBL	DROP1
Flag	100	0.707	51.95	0.676+	21.43	0.686	28.11	0.686	21.89	0.687	25.60	0.686	24.23	0.670	0.683	28.867	GRIBL	DROP1
Glass	100	0.690	49.64	**0.630-**	20.49	0.541	26.23	0.636	18.20	0.587	23.83	0.646	23.28	0.603	0.603	26.944	DROP4	DROP3
Heart	100	0.841	37.37	**0.837+**	9.55	0.807	17.65	0.822	11.44	0.848	12.30	0.837	11.98	0.811	0.825	16.715	DROP3	DROP1
Heart.Long-beach-va.2	100	0.710	43.00	**0.715**	8.50	0.640	11.56	0.715	4.11	0.580	6.72	0.545	7.39	0.550	0.606	13.546	GRIBL	DROP3
Heart.cleveland.2	100	0.809	50.94	**0.815+**	11.14	0.766	18.28	0.802	11.32	0.825	13.11	0.815	13.85	0.828	0.807	19.771	DROP5	DROP1
Heart.hungarian.2	100	0.775	47.96	0.768-	12.92	0.826	16.89	0.802	8.79	0.813	12.31	0.823	15.68	0.823	0.817	19.092	DROP1	DROP3
Heart.swiss.2	100	0.937	35.59	0.937-	2.70	0.962	6.31	0.951	3.42	0.962	3.42	0.962	2.97	0.968	0.961	9.070	DROP5	DROP1
Hepatitis	100	0.787	38.85	**0.820+**	6.00	0.799	12.57	0.813	4.07	0.481	5.64	0.616	6.14	0.337	0.609	12.214	GRIBL	DROP3
Horse-colic	100	0.731	48.06	**0.694-**	6.83	0.449	16.64	0.734	1.99	0.627	8.01	0.545	9.23	0.472	0.565	15.126	DROP2	DROP3
Iris	100	0.953	27.85	**0.953+**	9.63	0.960	15.48	0.947	14.74	0.940	14.96	0.940	12.74	0.933	0.944	15.901	DROP1	DROP1
Liver.bupa	100	0.617	62.42	0.569-	25.16	0.595	34.84	0.618	22.65	0.609	28.97	0.621	26.42	0.591	0.607	33.408	DROP4	DROP3
Pima-indians-diabetes	100	0.720	51.04	**0.727+**	16.90	0.707	25.12	0.712	14.59	0.738	18.93	0.702	18.38	0.724	0.717	24.161	DROP3	DROP3
Promoters	100	0.943	26.31	**0.885+**	7.79	0.875	15.05	0.847	14.42	0.903	14.74	0.903	10.21	0.887	0.883	14.753	DROP3	DROP1
Wine	100	0.961	30.84	**0.961+**	8.94	0.950	15.31	0.950	14.69	0.950	14.69	0.950	9.38	0.961	0.952	15.639	GRIBL	DROP1
Zoo	100	0.944	43.09	**0.844+**	19.14	0.900	20.25	0.822	19.51	0.811	21.23	0.800	19.51	0.711	0.809	23.786	DROP1	DROP1
Average	100.00	0.816	42.05	0.797	12.52	0.769	18.20	0.792	12.20	0.763	14.90	0.766	13.79	0.734	0.765	18.942	DROP3	GRIBL

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

2. The termination condition takes into account the last 10 generations only. If no improvement is achieved during these generations the evolution stops. Increasing the number of generations taken into consideration could improve the results.
3. The fitness of each individual is evaluated using a small subset of the training set (20%), which might not accurately reflect the fitness of an individual. Increasing the size of subset may, therefore, improve the results.

However, these changes may require considerably more evolution time.

3. The Seeded-GRIBL Algorithm

In order to solve GRIBL's time problem, we thought about initializing the population with quality individuals hoping that this will give the system ahead start, enabling it to converge to a good reduced set in less number of generations.

Instead of starting with a random population, Seeded-GRIBL makes use of the previous solutions obtained by other reduction techniques. It initializes the population with individuals that represent the reduced set found by 10 other techniques. The reduction techniques we considered were the 5 DROP techniques, ENN, RENN, EXPOLRE, ELGROW, and AllKnn.

The chosen reduction techniques guarantee a diverse population with individuals produced by different categories of reduction techniques, covering a large area from the solution space. DROPs provide competitive reduced sets in terms of size and classification accuracy, which expected to be the nucleus for good solutions. ENN and RENN offers reduced sets with good classification accuracy but with large size. This is because these techniques are good as noise elimination techniques. On the other

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

hand, EXPOLRE and ELGROW reduced sets are very small but with bad classification accuracy. While AllKnn provides a reduced set which is fair in both terms.

3.1 Seeded-GRIBL with Seeds From Ten Reduction Techniques

Starting with a good initial population, this technique is expected to outperform GRIBL technique with respect to: size of the reduced set, classification accuracy of it, and number of generations passed before finding the best reduced set (i. e. the evolution time).

Providing population with quality individuals from the beginning will save much of the generations wasted before arriving to such stage. It considers a search in areas that probably contains the global optima; hence, it helps the algorithm to avoid the local optima GRIBL may fall in causing a termination with solution that has a lower quality.

Therefore, Seeded-GRIBL is expected to be competitive to other reduction techniques in terms of classification accuracy and size of the reduced set.

Table 4 shows the classification accuracy and the size of reduced set obtained by Seeded-GRIBL the 10 reduction techniques used to initialize Seeded-GRIBL population. The table shows that the classification accuracy obtained by Seeded-GRIBL for 17 datasets was better than the average classification accuracy obtained by the 10 techniques. Moreover, the size of the reduced set obtained by Seeded-GRIBL was better than the size of the reduced set obtained by the 10 techniques for 15 datasets.

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

Table 4. The classification accuracy and size of reduced set of Seeded-GRIBL and 10 other reduction techniques

DataSet	KNN		Seeded-GRIBL		Drop1		Drop2		Drop3		Drop4		Drop5		ENN		RENN		EXPLORE		EIGROW		All KNN	
	size%	Acc	size%	Acc	size%	Acc	size%	Acc	size%	Acc	size%	Acc	size%	Acc	size%	Acc	size%	Acc	size%	Acc	size%	Acc	size%	Acc
Breast-cancer-wisconsin	100	0.961	2.257	**0.970**	1.70	0.970	5.31	0.963	3.05	0.966	3.15	0.967	2.93	0.966	87.58	0.970	87.55	0.970	0.32	0.655	0.32	0.655	86.06	0.968
Bridges	100	0.688	24.214	**0.631**	23.58	0.472	27.58	0.496	15.79	0.471	25.68	0.506	20.42	0.438	42.00	0.499	35.47	0.464	1.89	0.341	1.58	0.350	30.53	0.463
Echocardiogram	100	0.920	9.309	**0.932**	12.99	0.932	14.33	0.932	14.93	0.932	14.93	0.932	13.43	0.932	84.18	0.920	83.28	0.920	2.69	0.693	2.39	0.693	68.36	0.920
Flag	100	0.707	37.457	**0.722**	21.43	0.686	28.11	0.686	21.89	0.687	25.60	0.686	24.23	0.670	68.74	0.712	65.60	0.696	4.69	0.391	4.23	0.323	60.40	0.697
Glass	100	0.690	33.443	**0.641**	20.49	0.541	26.23	0.636	18.20	0.587	23.83	0.646	23.28	0.603	61.64	0.626	56.72	0.587	3.83	0.355	3.11	0.335	56.34	0.651
Heart	100	0.841	17.160	**0.844**	9.55	0.807	17.65	0.822	11.44	0.848	12.30	0.837	11.98	0.811	77.53	0.848	77.00	0.848	0.82	0.556	0.82	0.556	67.24	0.837
Heart.Long-beach-va.2	100	0.710	5.389	**0.750**	8.50	0.640	11.56	0.715	4.11	0.580	6.72	0.545	7.39	0.550	63.28	0.725	59.72	0.745	0.56	0.745	0.56	0.745	45.22	0.735
Heart.cleveland.2	100	0.809	14.411	**0.815**	11.14	0.766	18.28	0.802	11.32	0.825	13.11	0.815	13.85	0.828	75.05	0.822	74.21	0.818	0.73	0.534	0.73	0.534	63.77	0.812
Heart.hungarian.2	100	0.775	11.300	**0.829**	12.92	0.826	16.89	0.802	8.79	0.813	12.31	0.823	15.68	0.823	63.48	0.751	60.95	0.724	0.83	0.449	0.61	0.449	52.35	0.749
Heart.swiss.2	100	0.937	0.903	0.937	2.70	0.962	6.31	0.951	3.42	0.962	3.42	0.962	2.97	0.968	88.02	0.962	88.02	0.962	1.08	0.907	1.08	0.907	86.13	0.962
Hepatitis	100	0.787	8.674	**0.812**	6.00	0.799	12.57	0.813	4.07	0.481	5.64	0.616	6.14	0.337	65.86	0.799	62.86	0.793	0.71	0.793	0.71	0.793	46.71	0.812
Horse-colic	100	0.731	13.437	**0.777**	6.83	0.449	16.64	0.734	1.99	0.627	8.01	0.545	9.23	0.472	59.85	0.697	58.63	0.671	0.37	0.671	0.37	0.671	49.56	0.701
Iris	100	0.953	10.000	**0.947**	9.63	0.960	15.48	0.947	14.74	0.940	14.96	0.940	12.74	0.933	86.15	0.953	85.93	0.947	2.22	0.333	2.22	0.333	85.26	0.960
Liver.bupa	100	0.617	24.734	**0.632**	25.16	0.595	34.84	0.618	22.65	0.609	28.97	0.621	26.42	0.591	60.94	0.600	57.06	0.606	0.45	0.580	0.32	0.580	47.00	0.588
Pima-indians-diabetes	100	0.720	22.758	**0.746**	16.90	0.707	25.12	0.712	14.59	0.738	18.93	0.702	18.38	0.724	69.12	0.742	66.74	0.741	0.29	0.620	0.27	0.620	57.93	0.744
Promoters	100	0.943	8.910	**0.941**	7.79	0.875	15.05	0.847	14.42	0.903	14.74	0.903	10.21	0.887	88.00	0.905	87.89	0.905	2.11	0.537	2.11	0.537	88.00	0.905
Wine	100	0.961	8.302	**0.955**	8.94	0.950	15.31	0.950	14.69	0.950	14.69	0.950	9.38	0.961	85.94	0.955	85.94	0.955	1.88	0.332	1.88	0.332	84.81	0.955
Zoo	100	0.944	17.531	**0.922**	19.14	0.900	20.25	0.822	19.51	0.811	21.23	0.800	19.51	0.711	84.20	0.900	83.83	0.900	9.01	0.667	7.04	0.678	84.94	0.911
Average	100.00	0.816	15.01	**0.822**	12.52	0.769	18.20	0.792	12.20	0.763	14.90	0.766	13.79	0.734	72.86	0.799	70.97	0.792	1.92	0.564	1.69	0.561	64.48	0.798

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

The results of comparing Seeded-GRIBL with the 5 DROP techniques are shown in table 5. The figures in the table show that the average accuracy achieved by Seeded-GRIBL is higher than the best-known reduction techniques (which is DROP2) by 3.1%, at a cost of 2.8% (on average) increase in the size of the reduced set, compared with the same technique.

A statistical significance test with 95% confidence level was applied to the results. The test compares the significance of Seeded-GRIBL classification accuracy with DROP2. The test showed that Seeded-GRIBL's classification accuracy was statically significant for 14 datasets (marked in table 5 with a +), and not statically significant for 1 datasets (marked in table 5 by a -).

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

Table 5. The classification accuracy and size of reduced set of Seeded-GRIBL and DROPs family

DataSet	KNN		Seeded-GRIBL		GRIBL		Drop1		Drop2		Drop3		Drop4		Drop5		Average Acc.	Average size%.	Best acc	Best size%
	size%	Acc	size%	Acc	size%	Acc	size%	Acc	size%	Acc	size%	Acc	size%	Acc	size%	Acc				
Breast-cancer-wisconsin	100	0.961	2.26	**0.970+**	33.62	0.967	1.70	0.970	5.31	0.963	3.05	0.966	3.15	0.967	2.93	0.966	0.966	3.227	DROP1	DROP1
Bridges	100	0.688	24.21	**0.631+**	57.23	0.612	23.58	0.472	27.58	0.496	15.79	0.471	25.68	0.506	20.42	0.438	0.477	22.611	Seeded-GRIBL	DROP3
Echocardiogram	100	0.920	9.31	**0.932**	21.17	0.932	12.99	0.932	14.33	0.932	14.93	0.932	14.93	0.932	13.43	0.932	0.932	14.119	Seeded-GRIBL	Seeded-GRIBL
Flag	100	0.707	37.46	**0.722+**	51.95	0.676	21.43	0.686	28.11	0.686	21.89	0.687	25.60	0.686	24.23	0.670	0.683	24.251	Seeded-GRIBL	DROP1
Glass	100	0.690	33.44	**0.641+**	49.64	0.630	20.49	0.541	26.23	0.636	18.20	0.587	23.83	0.646	23.28	0.603	0.603	22.404	Seeded-GRIBL	DROP3
Heart	100	0.841	17.16	**0.844+**	37.37	0.837	9.55	0.807	17.65	0.822	11.44	0.848	12.30	0.837	11.98	0.811	0.825	12.584	DROP3	DROP1
Heart.Long-beach-va.2	100	0.710	5.39	**0.750+**	43.00	0.715	8.50	0.640	11.56	0.715	4.11	0.580	6.72	0.545	7.39	0.550	0.606	7.656	Seeded-GRIBL	DROP3
Heart.cleveland.2	100	0.809	14.41	**0.815+**	50.94	0.815	11.14	0.766	18.28	0.802	11.32	0.825	13.11	0.815	13.85	0.828	0.807	13.538	DROP5	DROP1
Heart.hungarian.2	100	0.775	11.30	**0.829+**	47.96	0.768	12.92	0.826	16.89	0.802	8.79	0.813	12.31	0.823	15.68	0.823	0.817	13.318	Seeded-GRIBL	DROP3
Heart.swiss.2	100	0.937	0.90	0.937-	35.59	0.937	2.70	0.962	6.31	0.951	3.42	0.962	3.42	0.962	2.97	0.968	0.961	3.766	DROP5	Seeded-GRIBL
Hepatitis	100	0.787	8.67	**0.812**	38.85	0.820	6.00	0.799	12.57	0.813	4.07	0.481	5.64	0.616	6.14	0.337	0.609	6.886	GRIBL	DROP3
Horse-colic	100	0.731	13.44	**0.777+**	48.06	0.694	6.83	0.449	16.64	0.734	1.99	0.627	8.01	0.545	9.23	0.472	0.565	8.539	Seeded-GRIBL	DROP3
Iris	100	0.953	10.00	0.947	27.85	0.953	9.63	0.960	15.48	0.947	14.74	0.940	14.96	0.940	12.74	0.933	0.944	13.511	DROP1	DROP1
Liver.bupa	100	0.617	24.73	**0.632+**	62.42	0.569	25.16	0.595	34.84	0.618	22.65	0.609	28.97	0.621	26.42	0.591	0.607	27.606	DROP4	DROP3
Pima-indians-diabetes	100	0.720	22.76	**0.746+**	51.04	0.727	16.90	0.707	25.12	0.712	14.59	0.738	18.93	0.702	18.38	0.724	0.717	18.784	Seeded-GRIBL	DROP3
Promoters	100	0.943	8.91	**0.941+**	26.31	0.885	7.79	0.875	15.05	0.847	14.42	0.903	14.74	0.903	10.21	0.887	0.883	12.442	Seeded-GRIBL	DROP1
Wine	100	0.961	8.30	**0.955+**	30.84	0.961	8.94	0.950	15.31	0.950	14.69	0.950	14.69	0.950	9.38	0.961	0.952	12.600	GRIBL	Seeded-GRIBL
Zoo	100	0.944	17.53	**0.922+**	43.09	0.844	19.14	0.900	20.25	0.822	19.51	0.811	21.23	0.800	19.51	0.711	0.809	19.926	Seeded-GRIBL	Seeded-GRIBL
Average	100.00	0.816	15.01	**0.822**	42.05	0.7968	12.52	0.769	18.20	0.792	12.20	0.763	14.90	0.766	13.79	0.734	0.765	14.321	Seeded-GRIBL	Seeded-GRIBL

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

3.2 DROPs Seeded-GRIBL

DROPs Seeded-GRIBL is an extension of Seeded-GRIBL in which the initial population is seeded (initialized) with individuals representing the reduced set obtained by the DROP techniques only.

The intuition is that since the DROP family of techniques provides solutions with good combination of classification accuracy and amount of reduction. Perhapse seeding GRIBL with such solutions would allow it to improve on them.

The experiments performed, reported in table 7, showed that DROPs Seeded-GRIBL outperforms Seeded-GRIBL in with respect to the size of the reduced set for 12 datasets. The average size of reduced set obtained by DROPs Seeded-GRIBL is better than Seeded-GRIBL's by 2.3%.

Table 7. The classification accuracy, size of reduced set, and number of generations of DROPs Seeded-GRIBL and Seeded-GRIBL

DataSet	KNN		DROPs Seeded-GRIBL		Seeded-GRIBL	
	size%	Acc	size%	Acc	size%	Acc
Breast-cancer-wisconsin	100	0.961	**2.146**	0.967	2.257	0.970
Bridges	100	0.642	**21.698**	0.631	24.214	0.631
Echocardiogram	100	0.988	9.309	0.932	9.309	0.932
Flag	100	0.722	**24.112**	0.701	37.457	0.722
Glass	100	0.694	**21.894**	0.642	33.443	0.641
Heart	100	0.811	**11.770**	0.856	17.160	0.844
Heart.Long-beach-va.2	100	0.770	7.000	0.745	5.389	0.750
Heart.cleveland.2	100	0.789	**13.531**	0.822	14.411	0.815
Heart.hungarian.2	100	0.813	11.943	0.843	11.300	0.829
Heart.swiss.2	100	0.937	0.903	0.937	0.903	0.937
Hepatitis	100	0.799	**7.599**	0.813	8.674	0.812
Horse-colic	100	0.604	**11.443**	0.751	13.437	0.777
Iris	100	0.960	**9.259**	0.960	10.000	0.947
Liver.bupa	100	0.643	27.440	0.643	24.734	0.632
Pima-indians-diabetes	100	0.694	**16.189**	0.745	22.758	0.746
Promoters	100	0.943	**8.805**	0.941	8.910	0.941
Wine	100	0.719	8.302	0.955	8.302	0.955
Zoo	100	0.944	**14.938**	0.911	17.531	0.922
Average	100	0.802	**12.682**	**0.822**	15.011	0.822

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

The results in table 8 show that over the 18 datasets DROPs Seeded-GRIBL achieved higher classification accuracy than the best-known reduction technique (which is DROP2) by 3.1%. This came at a slight cost of 0.6% (on average) increase in the size of the reduced set, compared with the same technique.

The table also shows that DROPs Seeded-GRIBL achieved significantly higher classification accuracy than the accuracy achieved by DROP2 for 15 datasets, and significantly lower accuracy for 1 datasets.

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

Table 8. The classification accuracy and size of reduced set of DROPs Seeded-GRIBL, Seeded-GRIBL and DROPs family

DataSet	Size%		DROPs Seeded-GRIBL		Seeded-GRIBL		DROP1		DROP2		DROP3		DROP4		DROP5		Average Acc.	Average size%.	Best acc	Best size%
	size%	Acc	size%	Acc	size%	Acc	size%	Acc	size%	Acc	size%	Acc	size%	Acc	size%	Acc				
Breast-cancer-wisconsin	100	0.961	2.15	**0.967+**	2.257	0.970	1.70	0.970	5.31	0.963	3.05	0.966	3.15	0.967	2.93	0.966	0.966	3.227	DROP1	DROP1
Bridges	100	0.688	21.70	**0.631+**	24.214	0.631	23.58	0.472	27.58	0.496	15.79	0.471	25.68	0.506	20.42	0.438	0.477	22.611	D Seeded-GRIBL	DROP3
Echocardiogram	100	0.920	9.31	**0.932**	9.309	0.932	12.99	0.932	14.33	0.932	14.93	0.932	14.93	0.932	13.43	0.932	0.932	14.119	D Seeded-GRIBL	D Seeded-GRIBL
Flag	100	0.707	24.11	**0.701+**	37.457	0.722	21.43	0.686	28.11	0.686	21.89	0.687	25.60	0.686	24.23	0.670	0.683	24.251	D Seeded-GRIBL	DROP1
Glass	100	0.690	21.89	**0.642+**	33.443	0.641	20.49	0.541	26.23	0.636	18.20	0.587	23.83	0.646	23.28	0.603	0.603	22.404	D Seeded-GRIBL	DROP3
Heart	100	0.841	11.77	**0.856+**	17.160	0.844	9.55	0.807	17.65	0.822	11.44	0.848	12.30	0.837	11.98	0.811	0.825	12.584	D Seeded-GRIBL	DROP1
Heart.Long-beach-va.2	100	0.710	7.00	**0.745+**	5.389	0.750	8.50	0.640	11.56	0.715	4.11	0.580	6.72	0.545	7.39	0.550	0.606	7.656	DROP2	DROP3
Heart.cleveland.2	100	0.809	13.53	**0.822+**	14.411	0.815	11.14	0.766	18.28	0.802	11.32	0.825	13.11	0.815	13.85	0.828	0.807	13.538	D Seeded-GRIBL	DROP1
Heart.hungarian.2	100	0.775	11.94	**0.843+**	11.300	0.829	12.92	0.826	16.89	0.802	8.79	0.813	12.31	0.823	15.68	0.823	0.817	13.318	DROP1	DROP3
Heart.swiss.2	100	0.937	0.90	**0.937-**	0.903	0.937	2.70	0.962	6.31	0.951	3.42	0.962	3.42	0.962	2.97	0.968	0.961	3.766	D Seeded-GRIBL	Seeded-GRIBL
Hepatitis	100	0.787	7.60	**0.813**	8.674	0.812	6.00	0.799	12.57	0.813	4.07	0.481	5.64	0.616	6.14	0.337	0.609	6.886	D Seeded-GRIBL	DROP3
Horse-colic	100	0.731	11.44	**0.751+**	13.437	0.777	6.83	0.449	16.64	0.734	1.99	0.627	8.01	0.545	9.23	0.472	0.565	8.539	DROP2	DROP3
Iris	100	0.953	9.26	**0.960+**	10.000	0.947	9.63	0.960	15.48	0.947	14.74	0.940	14.96	0.940	12.74	0.933	0.944	13.511	D Seeded-GRIBL	D Seeded-GRIBL
Liver.bupa	100	0.617	27.44	**0.643+**	24.734	0.632	25.16	0.595	34.84	0.618	22.65	0.609	28.97	0.621	26.42	0.591	0.607	27.606	D Seeded-GRIBL	DROP3
Pima-indians-diabetes	100	0.720	16.19	**0.745+**	22.758	0.746	16.90	0.707	25.12	0.712	14.59	0.738	18.93	0.702	18.38	0.724	0.717	18.784	Seeded-GRIBL	DROP3
Promoters	100	0.943	8.81	**0.941+**	8.910	0.941	7.79	0.875	15.05	0.847	14.42	0.903	14.74	0.903	10.21	0.887	0.883	12.442	D Seeded-GRIBL	Seeded-GRIBL
Wine	100	0.961	8.30	**0.955+**	8.302	0.955	8.94	0.950	15.31	0.950	14.69	0.950	14.69	0.950	9.38	0.961	0.952	12.600	D Seeded-GRIBL	DROP1
Zoo	100	0.944	14.94	**0.911+**	17.531	0.922	19.14	0.900	20.25	0.822	19.51	0.811	21.23	0.800	19.51	0.711	0.809	19.926	DROP1	D Seeded-GRIBL
Average	100.00	0.816	12.68	**0.822**	15.01	0.822	12.52	0.769	18.20	0.792	12.20	0.763	14.90	0.766	13.79	0.734	0.765	14.321	DROP2	DROP3

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

4. The Effect of Evolution Parameters:

The empirical work showed in the pervious two sections assumed certain values for the different parameters for the evolution process. In this section, we introduce some experiments that are held using different parameter values in order to justify the choice, we made in the original experiments.

4.1 The Effect of Population Size

The original experiments of GRIBL, shown in section 4.2, assumed a population of 10 individuals. Increasing the number of individuals is expected to increase the diversity of the solutions considered (i. e. increases the area covered in the search space). Table 9 shows the classification accuracy and the size percentage of the reduced set for GRIBL using a population of fifty individuals.

Table 9. The classification accuracy, size of reduced set, and number of generations of GRIBL using 10 and 50 individuals in the population.

DataSet	KNN		GRIBL with 10			GRIBL with 50		
	size%	Acc	size%	Acc	gen	size%	Acc	gen
Breast-cancer-wisconsin	100	0.961	33.62	0.967	15.0	28.64	0.967	38.9
Bridges	100	0.688	57.23	0.612	16.4	45.91	0.661	33.1
Echocardiogram	100	0.920	21.17	0.932	29.0	15.02	0.932	67.2
Flag	100	0.707	51.95	0.676	19.9	51.20	0.676	28.7
Glass	100	0.690	49.64	0.630	15.6	54.84	0.651	32.3
Heart	100	0.841	37.37	0.837	16.1	43.91	0.822	32.5
Heart.Long-beach-va.2	100	0.710	43.00	0.715	15.4	35.78	0.740	27.1
Heart.cleveland.2	100	0.809	50.94	0.815	16.2	45.40	0.825	33.0
Heart.hungarian.2	100	0.775	47.96	0.768	17.6	34.54	0.785	25.4
Heart.swiss.2	100	0.937	35.59	0.937	18.3	8.76	0.937	78.3
Hepatitis	100	0.787	38.85	0.820	17.7	33.26	0.840	35.4
Horse-colic	100	0.731	48.06	0.694	16.3	48.50	0.714	33.6
Iris	100	0.953	27.85	0.953	26.8	11.11	0.973	82.4
Liver.bupa	100	0.617	62.42	0.569	18.2	55.33	0.594	30.5
Pima-indians-diabetes	100	0.720	51.04	0.727	13.4	45.36	0.639	23.6
Promoters	100	0.943	26.31	0.885	37.5	8.81	0.537	62.2
Wine	100	0.961	30.84	0.961	34.9	13.17	0.515	85.1
Zoo	100	0.944	43.09	0.844	33.2	23.95	0.878	45.5
Average	100	0.816	42.05	0.797	21.0	33.53	0.760	44.2

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

The experiment results show that GRIBL with 50 individuals in the population achieved lower average classification accuracy by 3.6%. However, it also achieved better average size reduction by 8.52%. This improvement in reduction came at the cost of the time consumed by the evolution process before converging to a fit population, where an increase of 23 generations was noticed when using the 50 individual population. This implies that the use of larger population may result in finding better solutions since the technique will search more areas in the search space, however, it may be time consuming for a certain extent.

4.2 The Effect of Changing Number of Generations Considered by Termination Criterion

As mentioned in section 3.2.3, in order to reduce the likelihood that GRIBL would fall in local optima, GRIBL terminates when there is no improvement in fitness for a number of successive generations.

In the original experiments, GRIBL terminates when no improvement is achieved in 10 successive generations. Table 10 shows the results obtained using 5, 10, and 15 generations. The results show that increasing the number of generations gives smaller reduced sets and relatively better classification accuracy, but it also increases the evolution time as the number of generation increases.

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

Table 10. The classification accuracy, size of reduced set, and number of generations of GRIBL using 5, 10, and 15 as sizes of the record on termination criterion.

DataSet	KNN		GRIBL with 5			GRIBL with 10			GRIBL with 15		
	size%	Acc	size%	Acc	gen	size%	Acc	gen	size%	Acc	Gen
Breast-cancer-wisconsin	100	0.961	33.71	0.718	7.6	33.62	0.967	15.0	33.62	0.967	21.5
Bridges	100	0.688	57.34	0.612	10.8	57.23	0.612	16.4	56.71	0.612	26.1
Echocardiogram	100	0.920	25.38	0.750	11.2	21.17	0.932	29.0	15.02	0.932	67.2
Flag	100	0.707	52.41	0.676	10.1	51.95	0.676	19.9	51.72	0.676	30.1
Glass	100	0.690	49.86	0.636	8.5	49.64	0.630	15.6	49.59	0.631	27.4
Heart	100	0.841	37.45	0.837	9.2	37.37	0.837	16.1	37.33	0.830	27.8
Heart.Long-beach-va.2	100	0.710	43.00	0.715	9.2	43.00	0.715	15.4	42.94	0.710	22.8
Heart.cleveland.2	100	0.809	46.31	0.577	9.8	50.94	0.815	16.2	50.90	0.812	22.3
Heart.hungarian.2	100	0.775	49.28	0.681	10.4	47.96	0.768	17.6	48.07	0.768	26.1
Heart.swiss.2	100	0.937	33.60	0.937	17.4	35.59	0.937	18.3	29.18	0.937	61.6
Hepatitis	100	0.787	39.00	0.820	9.3	38.85	0.820	17.7	38.85	0.820	22.7
Horse-colic	100	0.731	47.07	0.671	7.5	48.06	0.694	16.3	48.06	0.694	24.0
Iris	100	0.953	29.56	0.953	9.4	27.85	0.953	26.8	24.30	0.973	58.9
Liver.bupa	100	0.617	62.51	0.566	7.8	62.42	0.569	18.2	62.42	0.569	26.3
Pima-indians-diabetes	100	0.720	50.74	0.629	8.3	51.04	0.727	13.4	51.04	0.727	18.4
Promoters	100	0.943	32.29	0.574	11.9	26.31	0.885	37.5	23.58	0.856	61.5
Wine	100	0.961	32.71	0.961	14.6	30.84	0.961	34.9	26.40	0.950	85.8
Zoo	100	0.944	47.78	0.856	10.3	43.09	0.844	33.2	40.37	0.822	51.4
Average	100	0.816	42.78	0.732	10.2	42.05	0.797	21.0	40.56	0.794	37.9

4.3 The Effect of the Size Weight Used in Fitness Function

In GRIBL the following fitness function was used:

$$\frac{2^{Accuracy}-1}{(size\_ratio+1)^{w}}$$

The constant *w* determines the relative importance of the size of the reduced set compared to the classification accuracy achieved by this set. Hence higher values of *w* gives more weight for the size.

Table 11 shows the results of using two different values for *w*: 0.2 and 0.5. As expected, a higher value for *w* caused GRIBL to search through solution areas with

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

smaller reduced set paying less attention to the classification accuracy achieved by that set.

Table 11. The classification accuracy, size of reduced set, and number of generations of GRIBL using 0.2 and 0.5 as a weight for the size in the fitness fuction

DataSet	KNN		GRIBL with w=0.2			GRIBL with w=0.5		
	size%	Acc	size%	Acc	gen	size%	Acc	gen
Breast-cancer-wisconsin	100	0.961	33.62	0.967	15.0	30.85	0.863	18.9
Bridges	100	0.688	57.23	0.612	16.4	44.34	0.610	19.6
Echocardiogram	100	0.920	21.17	0.932	29.0	20.57	0.750	30.1
Flag	100	0.707	51.95	0.676	19.9	41.81	0.360	21.3
Glass	100	0.690	49.64	0.630	15.6	40.07	0.436	21.1
Heart	100	0.841	37.37	0.837	16.1	30.86	0.689	18.4
Heart.Long-beach-va.2	100	0.710	43.00	0.715	15.4	40.11	0.715	23.9
Heart.cleveland.2	100	0.809	50.94	0.815	16.2	39.75	0.580	20.9
Heart.hungarian.2	100	0.775	47.96	0.768	17.6	37.04	0.554	17.1
Heart.swiss.2	100	0.937	35.59	0.937	18.3	31.35	0.937	35.8
Hepatitis	100	0.787	38.85	0.820	17.7	32.33	0.799	22.2
Horse-colic	100	0.731	48.06	0.694	16.3	45.77	0.668	15.4
Iris	100	0.953	27.85	0.953	26.8	25.48	0.660	46.4
Liver.bupa	100	0.617	62.42	0.569	18.2	47.18	0.603	17.2
Pima-indians-diabetes	100	0.720	51.04	0.727	13.4	37.47	0.675	19.4
Promoters	100	0.943	26.31	0.885	37.5	29.25	0.584	26.6
Wine	100	0.961	30.84	0.961	34.9	27.65	0.624	36.2
Zoo	100	0.944	43.09	0.844	33.2	34.69	0.422	35.0
Average	100	0.816	42.05	0.797	21.0	35.37	0.640	24.8

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

CONCLUSION AND FUTURE WORK

1. Conclusion

Instance-based Learning algorithm is a simple inductive learning method. The learning step simply requires storing the instances of the training set, with no further work on the generalization of the target function. An unseen instance is classified by retrieving a set of the most similar training instances. This set is used to predict the class of the new instance. In effect, IBL forms a local representation of the target function instead of a global one as eager learners do, which makes it suitable for problems with complex target function that are better described by several less complex local approximations. Moreover, IBL can use more complex, symbolic representation of instances, which qualifies it to be used in many real-world learning tasks (Mitchell, 1997).

IBL has proven to be successful, in terms of classification accuracy, over a wide area of real-world benchmark data sets. It is competitive to more sophisticated learning techniques such as neural networks in many applications (Cost and Salzberg, 1993, Stanfill and Waltz, 1986, Hindi et al, 2003).

However, Classification accuracy achieved by IBL highly depends on the number of training instances stored at learning time. Storing too many instances can reduce the classification speed and increase memory requirements.

To remedy these problems of the large training set stored by IBL, different reduction techniques were proposed in the literature (see section 2.2 for a revision of such techniques) (Wilson and Martinez, 2000b).

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

In this work, we presented the instance reduction as an optimization problem and utilized the genetic algorithms to address it. We used genetic algorithms to search the space of possible reduced sets in order to find a good reduced set with respect to both size and classification accuracy.

Two genetically-based techniques were developed. The first is called Genetically Reduced Instance-Based Learning (GRIBL), in which an evolution process iterates starting with a randomly initialized population. This process continues for successive generations by applying different genetic operators until a fit reduced set is obtained. Fitness is measured by a function that takes into consideration both the classification accuracy of the subset and its size (see section for more details 3.2.5).

The second technique, named Seeded-GRIBL, initializes the population with solutions obtained by other reduction techniques such as the 5 DROP algorithms, ENN, RENN, EXPOLRE, ELGROW, and AllKnn. Initializing the population with quality individuals gives the system ahead start, enabling it to converge to a good reduced set in fewer of generations. It also helps the algorithm to avoid the local optima that GRIBL may fall in simply because it considers search areas that probably contains the global optima.

The proposed techniques were tested over 18 bench-mark real-world datasets, and compared with the best reduction techniques with respect to the reduction in size and classification accuracy.

Experiments show that GRIBL achieved an average accuracy higher than the best reduction technique (which is DROP2) by 0.6%. However, that improvement was at the cost of the reduced set size, which is higher than the average reduced size of the same technique by 29.9%.

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

Furthermore, a statistical significance test with 95% confidence level was applied to the results. The test compares the significance of GRIBL classification accuracy with the best DROP technique. The results showed that GRIBL was significantly higher than the best DROP (which is DROP2) for 11 datasets with 95% confidence level, and lower for 5 datasets.

Experiments with Seeded-GRIBL show that its classification accuracy was better than the average classification accuracy achieved by 10 other reduction techniques for 17 datasets (out of the 18 datasets), and the size of the reduced set was better than the average in 15 datasets. It achieved the best average accuracy among the different reduction techniques. Moreover, Seeded-GRIBL was significantly higher than the best DROP technique for 14 datasets with 95% confidence level.

In other set of experiments to reduce the size of the reduced set, Seeded-GRIBL was initialized with solutions obtained from the 5 DROP techniques. The intuition is that we provide Seeded-GRIBL with solutions that are good in both classification accuracy and size of reduced set. Experiments show that Seeded-GRIBL achieved an average accuracy higher than the best-known reduction techniques (which is DROP2) by 3.1%. This came at a cost of 2.8% (on average) increase in the size of the reduced set, compared with the same technique. Moreover, classification accuracies obtained by DROPs Seeded-GRIBL were better than those obtained by the best DROP technique (which is DROP2) for 15 datasets with 95% confidence.

The GRIBL techniques in general and DROPs Seeded-GRIBL in particular, proved to compare favorably with other instance-based data reduction algorithms. Over eighteen real world problems, DROPs Seeded-GRIBL achieved the highest average generalization accuracy, and comparable percentage in size of the reduced set.

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

2. Future Work

Genetic algorithms have a large number of operators and parameters. In this work, we have covered a small portion of the available options. GRIBL and Seeded-GRIBL used evolution operators such as: tournament selection and single-point crossover, and bit-flip mutation. They, also, assume certain values for some parameters. Other operators and parameter values may give better results.

As future work, we intend to look for other ways to initialize the population, for example, several solutions of a good reduction technique, such as DROP2 might be used to initialize the population. These solutions can be obtained by applying such a reduction technique on a randomly generated sample of the original training set. Bagging and Boosting techniques for building ensemble of classifiers might provide good initial solutions (Dietterich, 1997).

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

REFERENCES

Aha, D. W. (1992). **Tolerating Noisy, Irrelevant and Novel Attributes in Instance-Based Learning.** International Journal of Man-Machine Studies, vol. 36, pp. 267-287.

Aha, D. W., Kibler D. and Albert. M. K. (1991). **Instance-Based Learning Algorithm.** Machine Learning, vol. 6-1, pp. 37-66.

Bäck, T. (1993). **Optimal Mutation Rates in Genetic Algorithms.** Proceedings of the Fifth ICGA, pp. 2-8.

Bäck T. (1996). **Evolutionary Algorithms in Theory and Practice.** Oxford, NY.

Cameron-Jones, R. M. (1995). **Instance Selection by Encoding Length Heuristic with Random Mutation Hill Climbing.** Proceedings of the Eighth Australian Joint Conference on Artificial Intelligence, pp. 99-106.

Cost, S. and Salzberg, S. (1993). **A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features.** Machine Learning, vol. 10-1, pp. 57 - 78.

Davis, L. (1991). **Handbook of Genetic Algorithms.** New York: Van Nostrand Reinhold.

Deng, K., and Moore A. W. (1995). **Multiresolution Instance-Based Learning.** Proceedings of the Twelfth International Joint Conference on Artificial Intellingence, pp. 1233-1239.

Dietterich, T. G. (1997). **Machine Learning Research: Four Current Directions.** AI Magazine, vol. 18-4, pp. 97-136.

Domingos, Perdo (1996). **Unifying Instance-Based and Rule-Based Induction.** Machine Learning, vol. 24, pp. 141-168.

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

Fu, L. (1994). **Neural Networks in Computer Intelligence**. New York: McGraw Hill.

Gates, G. W. (1972). **The Reduced Nearest Neighbor Rule**. IEEE Transactions on Information Theory, vol. 18-3, pp. 431-433.

Goldberg, D. E. (1989). **Genetic Algorithms in Search, Optimization and Machine Learning,** Addison-Wesley Publishing Company, Inc., Reading, MA.

Goldstein, J. (1991). **Genetic Algorithm Simulation of the SHOP Scheduling Problem**. ICMS/Shell Oil Business Consultancy, Central Library of Imperial College.

Grefenstette, J. (1987). **Genetic Algorithms and Their Applications.** Proceedings of the Second International Conference on Genetic Algorithms. Hillsdale, NJ: Lawrence Erlbaum Associates.

Grefenstette, J. and Baker, J. E. (1989). **How Genetic Algorithms Work: A Critical Look at Implicit Parallelism**. Proceedings of the Third ICGA, pp. 20-27.

Hart, P. E. (1968). **The Condensed Nearest Neighbor Rule**. IEEE Transactions on Information Theory, 14, pp. 515-516.

Hindi, K., Abu Kar A., and Al Qaddomi G. (2003). **Comparing the Machine Ability to Recognize Hand-Written Hindu and Arabic Digits**. To Appear in Pattern Recognition Letters.

Hindi, Khalil, Issa Eman, and Abu Kar Areej (2005) **Instance Reduction Techniques for Digit Recognition.** To appear in Dirasat, Pure Sciences, vol. 32(1).

Holland J.H. (1975). **Adaptation in Natural and Artificial System**. The University of Michigan Press.

Kibler, D. and Aha, D. (1987). **Learning Representative Exemplars of Concepts: An Initial Case Study**. Proceedings of the Fourth International Workshop on Machine Learning, pp. 24-30.

603435

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

Koza, J. (1992). **Genetic Programming: On the Programing of Computers by Means of Natural Selection.** MIT Press, Cambridge MA.

Louis, J. (2003). **Genetic Learning from Experience**. Proceedings of the International Congress on Evolutionary Computation, Canberra, Australia.

Louis, S. and Tang R. (1999). **Interactive Genetic Algorithms for the Traveling Salesman Problem**. Proceedings of the 1999 Genetic and Evolutionary Computing Conference (GECCO 1999), Orlando, Florida.

Louis, S. J. (1993). **Genetic Algorithms as a Computational Tool for Design.** Doctoral Thesis, Indiana University

Louis, S., and Johnson, J. (1997). **Solving Similar Problems Using Genetic Algorithms and Case-Based Memory**. Proceedings of the Seventh ICGA, pp. 283-290.

Maulik, U., and Bandyopadhyay, S. (2000). **Genetic algorithm-based clustering technique.** Pattern Recognition, 33, pp.1455-1465

Miller, J. A., Potter W. D., Gandham R. V., and Lapena C. N. (1993). **An Evaluation of loca improvement operators for genetic algorithms.** IEEE Transactions on Systems, Man and Cybernetics, vol. 23-5, pp.1340-1351.

Mitchell, M. (1996). **An Introduction to Genetic Algorithms**. MIT Press, Cambridge, MA Addison-Wesley.

Mitchell, T. M. (1997). **Machine Learning,** (1st ed.), New York, McGraw-Hill.

Nunez, M. (1991). **The Use of Background Knowledge in Decision Tree Induction.** Machine Learning, vol.6-3, pp.231-250.

Qi, X., and Palmieri, F. (1993). **The Diversification Role of Crossover In Genetic Algorithms**. Proceedings of the Fifth ICGA, pp. 132-137.

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

Quinlan, J.R. (1986). **Induction of Decision Trees**. Machine Learning, vol. 1, pp.81–106.

Ritter, G. L., Woodruff H. B., Lowry S. R., and Isenhour T. L. (1975). **An Algorithm for a Selective Nearest Neighbor Decision Rule**. IEEE Transactions on Information Theory, vol. 21-6, pp. 665-669.

Simon, H. A. (1983). **Why Should Machines Learn?**, In R.S Michalski, J. G. Garbonell, and T. M. Mitchell (editors). Machine Learning: An Artificial Intelligence Approach. Morgan Kaufmann, pp. 25-37.

Smith, A. E. and David M. (1993). **Genetic Optimization Using A Penalty Function.** Proceedings of the Fifth ICGA, p 499 – 505.

Stanfill, C., and Waltz D. (1986). **Toward Memory-Based Reasoning.** Communications of the ACM, vol. 29, 1213-1228.

Sutton, R., and Barto, A. (1998). **Reinforcement Learning.** MIT Press.

Vafaie, H., and De Jong, k. (1993). **Robust Feature Selection algorithms**. Proceeding of IEEE International Conference on Tools with Artificial Intelligence, pp.356-363.

Wettschereck, D., Aha D. and Takao M. (1997). **A Review and Comparative Evaluation of Feature Weighting Methods for Lazy Learning Algorithms.** Technical Report AIC-95-012, Washington, D.C, Naval Research laboratory, Navy Center for Applied Research in Artificial Intelligence.

Wettschereck, D., and Dietterich T.G. (1995). **An Experimental Comparison of Nearest-neighbor and Nearest-Hyperrectangle Algorithms.** Machine learning, vol. 19-1, pp.5-28.

Whitley, D. (1989). **The Genitor Algorithm And Selection Pressure: Why Rank-Based Allocation Of Reproductive Trials Is Best**. Proceedings of the Third ICGA, pp. 116-121.

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

Wilson, D. L. (1972). **Asymptotic Properties of Nearest Neighbor Rules Using Edited Data**. IEEE Transactions on Systems, Man, and Cybernetics, vol. 2-3, pp. 408-411.

Wilson, D. R.,Martinez T. R.(1996). **Instance-Based Learning with Genetically Derived Attribute Weights**. Proceeding of the international conference on Artifitial Intelligence, Expert Systems and Nueral Networks, pp.11-14.

Wilson, D. R., and Martinez T. R. (2000a). **An Integrated Instance-Based Algorithm.** Computational Intellegence, vol. 16-1, pp. 1-28.

Wilson, D. R., and Martinez, T. R. (2000b). **Reduction Techniques for Instance-Based Learning Algorithms**. Machine Learning, vol. 38-3, pp. 257-286.

Wilson, D. R.,Martinez T. R.(1997). **Improved Hetrogeneous Distance Functions.** Journal of Artifitial Intelligence Research (JAIR), vol. 6-1,pp. 1-34.

Yang, J., Honavar V. (1997). **Feature Subset Selection Using a Genetic Algorithm.** Computer science Department, Lowa State University.

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit

استخدام الخوارزميات الجينية في اختزال الأمثلة

إعداد
إيمان فارس عيسى

المشرف
الدكتور خليل الهندي

ملخص

التعلم المعتمد على الأمثلة يعتبر واحداً من أكثر طرق التعلم الاستقرائي استخداماً، حيث يتعلم عن طريق تخزين الأمثلة السابقة. و عند الحاجة لتصنيف مثال جديد يقوم باسترجاع أكثر الأمثلة شبهاً به و يستخدمها للتنبؤ بصنفه.

وقد أثبت هذا النوع من التعلم الاستقرائي نجاحه في تحقيق دقة تصنيف عالية في عدة تطبيقات عملية. ولكن لتحقيق هذا المستوى العالي من الدقة يتوجب عليه تخزين عدد كبير من الأمثلة مما يؤدي إلى زيادة في وقت التصنيف و حجم الذاكرة التي يحتاجهما.

لقد طرحت كثير من الأبحاث طرقاً مختلفة لتقليل عدد الأمثلة، حيث تقوم هذه الطرق بتحديد الأمثلة الأكثر تعبيراً بين الأمثلة المتوفرة لتجاوز الحاجة لذاكرة كبيرة و تقليل زمن التصنيف.

يقوم هذا البحث بتوظيف الخوارزميات الجينية لحل هذه المشكلة، حيث تم تطوير طريقتين جديدتين هما GRIBL و Seeded-GRIBL. و لقد قمنا بإجراء العديد من الدراسات التجريبية عليها و وجدنا أنها أثبتت فاعلية عالية مقارنة بالطرق المعروفة، حيث أنها تفوقت على أفضل الطرق الأخرى، وهي،DROP2 من حيث دقة التصنيف بمعدل 3.1% ولكن بزيادة طفيفة في عدد الأمثلة المختزلة بمعدل 0.6%.

All Rights Reserved - Library of University of Jordan - Center of Thesis Deposit