2010

# The Impact of Overfitting and Overgeneralization on the Classification Accuracy in Data Mining

Huy Nguyen Anh Pham
*Louisiana State University and Agricultural and Mechanical College*, hpham15@lsu.edu

# THE IMPACT OF OVERFITTING AND OVERGENERALIZATION ON THE CLASSIFICATION ACCURACY IN DATA MINING

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The Department of Computer Science

by
Huy Nguyen Anh Pham
B.S., University of Science of Ho Chi Minh City, Viet Nam, 1995
M.S., University of Science of Ho Chi Minh City, Viet Nam, 1999
May 2011

# ACKNOWLEDGEMENTS

Many people contributed to the successful completion of this dissertation. I am grateful to Dr. Evangelos Triantaphyllou for serving as committee chair and mentor. Committee members Drs. S. Sitharama Iyengar, Jianhua Chen, T. Warren Liao, and Martin Feldman guided the dissertation and provided valued advice. Drs. Mary L. Kelley, John Hopkins, Joe Abraham, Mrs. Janet Lemoine, and Judy Haslitt assisted for English proof-reading.

This dissertation was also supported by grants from Vietnamese Overseas Scholarship Program funded by Vietnamese Ministry of Education and Training.

Above all, I am deeply grateful to my parents, wife, and children. Their sacrifice for this dissertation was much more than anticipated and far more than fair.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# TABLE OF NOTATION

| Symbol | Explanation |
|---|---|
| *RateFP*, *RateFN*, *RateUC* | The false-positive, the false-negative, and the unclassifiable rates |
| $C_{FP}$, $C_{FN}$, $C_{UC}$ | The penalty costs for the false-positive, the false-negative, and the unclassifiable cases |
| *TC* | The total misclassification cost |
| *HD* and *CD* | The homogeneity degree and the convex-density |
| *E, F, G, M, P* | Regions |
| *T*, $T_1$, $T_2$ | Datasets |
| *n*, $n_E$, , $N_E$, $n_P$, $N_P$, $n_F$, $N_F$ | The number of data points |
| $\alpha^+$, $\alpha^-$, $\beta^+$, $\beta^-$, $\gamma$, *L* | Parameters |
| *D* | The number of dimensions |
| $M_1$, $M_2$, *C* | Classification models |
| *h, d* | Distances |
| $R_F$, $R_G$, *R*, $R_M$ | Radii |
| HBA | Homogeneity-Based Algorithm |
| CBA | Convexity-Based Algorithm |
| BSA | Binary Search Approach |
| SVM | Support Vector Machine |
| DT | Decision Tree |
| ANN | Artificial Neural Network |

# ABSTRACT

Current classification approaches usually do not try to achieve a balance between fitting and generalization when they infer models from training data. Such approaches ignore the possibility of different penalty costs for the false-positive, false-negative, and unclassifiable types. Thus, their performances may not be optimal or may even be coincidental.

This dissertation analyzes the above issues in depth. It also proposes two new approaches called the Homogeneity-Based Algorithm (HBA) and the Convexity-Based Algorithm (CBA) to address these issues. These new approaches aim at optimally balancing the data fitting and generalization behaviors of models when some traditional classification approaches are used. The approaches first define the total misclassification cost (*TC*) as a weighted function of the three penalty costs and their corresponding error rates. The approaches then partition the training data into regions. In the HBA, the partitioning is done according to some homogeneous properties derivable from the training data. Meanwhile, the CBA employs some convex properties to derive regions. A traditional classification method is then used in conjunction with the HBA and CBA. Finally, the approaches apply a genetic approach to determine the optimal levels of fitting and generalization. The *TC* serves as the fitness function in this genetic approach.

Real-life datasets from a wide spectrum of domains were used to better understand the effectiveness of the HBA and CBA. The computational results have indicated that both the HBA and CBA might potentially fill a critical gap in the implementation of current or future classification approaches. Furthermore, the results have also shown that when the penalty cost of

an error type was changed, the corresponding error rate followed stepwise patterns. The finding of stepwise patterns of classification errors can assist researchers in determining applicable penalties for classification errors. Thus, the dissertation also proposes a binary search approach (BSA) to produce those patterns. Real-life datasets were utilized to demonstrate for the BSA.

**Keywords:** Data mining, classification, prediction, overfitting, overgeneralization, false-positive, false-negative, unclassifiable, homogeneous region, homogeneity degree, convex set, convexity density, optimization, genetic algorithms.

# CHAPTER 1. INTRODUCTION

The importance of collecting enormous amounts of data related to science, engineering, business, governance, and almost any endeavor of human activity or the natural world is well recognized today. Powerful mechanisms for collecting and storing data and managing them in enormous datasets are in place in many large- and mid-range companies, not to mention research labs and various agencies. There is, however, a serious challenge in making the best use of such massive datasets and trying to gain new knowledge of the system or phenomenon that created these datasets. Human analysts cannot process and comprehend such datasets unless they have special computational tools at their disposal.

The emerging field of data mining and knowledge discovery seeks to develop reliable and effective computational tools for analyzing such large datasets to extract new knowledge from the data. Such new knowledge can be derived from patterns that are embedded in the data. Many applications of data mining involve the analysis of data that describe the state of nature of a hidden system of interest. Such a system could be natural or artificial phenomenon (such as the state of the weather or the result of a scientific experiment), a mechanical system (such as the engine of a car), an electronic system (such as an electronic device), and so on. Each data point describes the state of the phenomenon or system in terms of a number of attributes and their values for a given realization of the phenomenon or system. Furthermore, each data point is associated with a class value which describes a particular state of nature of this phenomenon or system. The typical setting of interest in this dissertation involves the indication that each data point belongs to one of two classes, which will be referred to without loss of generality, as the positive and negative classes. This is not a real restriction as any multi-class problems revert to a

number of two-class problems (see, for instance, [Pujol, et. al., 2006], [Dietterich, et. al., 1995], and [Crammer, et. al., 2002]). Then the main problem is to use such historic data (also known as the training data) to infer a model which would accurately classify new observations of unknown class values.

For instance, a bank administrator could be interested in knowing whether a loan application should be approved or not based on some characteristics of those applying for credit. There are two classes: "approve" or "do not approve." Attributes in this hypothetical scenario might include the age of the applicant, the income level of the applicant, the education level, and the employment status. In this case, the goal of the data mining process would be to extract any patterns that might occur in the data of successful credit applicants and also any patterns that might be present in the data of non-successful applicants. Successful applicants are defined here as those who can repay their loans without any negative complications, while non-successful applicants are those who default on their loans.

Many questions could be asked, but only a few of them would be important for the decision. With the abundance of data available in this area, a careful analysis could provide a pattern that discovers the main characteristics of reliable loan applicants. The data mining analyst often identifies such patterns from past data for which the final outcome is known, then uses those patterns to decide whether a new application for credit should be approved or not.

In other words, many applications of data mining involve the analysis of historic data (also known as the training data) for which we know the class value of each data point. We wish to infer some patterns from these data which in turn help us to infer the class value of new points for which the class value is unknown. These patterns may be defined on the attributes used to describe the available training data. For instance, for the previous bank example the patterns may

be defined on the level of education, years on the same job, and level of income of the applicants.

This kind of data mining analysis is called classification or class prediction of new data points because it uses patterns inferred from training data to aid in the correct classification / class prediction of new data points for which we do not know their class value. We only know the values of the attributes (perhaps not all of them) of the new data points. This description implies that this type of data mining analysis, besides the typical data definition, data collection, and data cleaning steps, involves the inference of a model of the phenomenon or system of interest to the analyst. This model consists of the patterns mentioned above. The data involved in deriving this model are the training data. Next, this model is used to infer the class value of new points.

Many theoretical and practical developments have been made in the last two decades or so regarding the development of approaches for inferring classification models from training data. The most recent approaches include the Statistical Learning Theory [Vapnik, 1998], Artificial Neural Networks (ANNs) [Abdi, 2003] and [Hecht-Nielsen, 1989], Decision Trees (DTs) [Quinlan, 1987, 1993, and 1996], and Support Vector Machines (SVMs) [Cristianini, et. al., 2000] and [Vapnik, 1998]. In such classification approaches, there are three different types of possible errors:

- The false-negative type where a data point, which in reality is positive, is predicted as negative.
- The false-positive type where a data point, which in reality is negative, is predicted as positive.
- The unclassifiable type where the classification approach cannot predict the class value of a new point. This occurs when insufficient information is extracted from the historic dataset.

Current classification approaches may work well with some training datasets, while they may perform poorly with other datasets for no obvious reason. If the classification approach is accurate, then we praise the mathematical model and claim that it is a good model. However, there is no solid understanding of why such models are accurate or not. The performance of such models is often coincidental.

A growing belief is that such approaches usually do not try to achieve a balance between fitting and generalization when they infer models from datasets. Thus, the models they infer may suffer from the overfitting and overgeneralization problems, and this causes their poor performance. Overfitting occurs when a model can accurately classify data points which are very closely related to the training data but performs poorly with data which are not closely related to the training data. Overgeneralization occurs when a model erroneously claims to be able to accurately classify vast amounts of data which are not closely related to the training data.

In general, current classification approaches attempt to minimize the sum of the false-negative and false-positive error rates without considering these two error rates in a weighted fashion. They also do not consider the case of having unclassifiable instances. To appreciate the magnitude of this situation, let us consider, for instance, the case of a diagnostic system for a serious disease (say some kind of aggressive cancer). In a situation like this, a false-positive diagnosis would subject a patient to some unnecessary medical tests and treatments, along with emotional distress. On the other hand, a false-negative diagnosis may cause loss of critical time which in turn may prove to be fatal to the patient. It is reasonable to argue here that these two cases of diagnostic errors should be associated with significantly different penalty costs (i.e., much higher for the false-negative case). Similar situations may occur when approving large lines of credit (as the current financial crisis is demonstrating), in oil exploration, issuing evacuation orders to avoid natural disasters (such as when a hurricane is approaching a

vulnerable area), classification of targets as enemy or not, and so on. The penalty associated with unclassifiable cases is more subtle. Currently the system does not make any diagnosis due to limited input information. However, in an extreme case a system may avoid any false-positive and false-negative types by reverting to unclassifiable outcomes for most diagnostic instances. That is, such a system would only offer advice when a new instance is of an obvious nature (i.e., either clearly positive or clearly negative) and avoid any challenging instance. That would result in high numbers of unclassifiable cases. Thus, this outcome should also be assigned a penalty cost as well.

This dissertation carefully analyzes the following issues: the balance between fitting and generalization, the differences in penalty costs for the false-positive and false-negative types, and the consideration of the unclassifiable type. Two new approaches, called the Homogeneity-Based Algorithm (HBA) and the Convexity-Based Algorithm (CBA), are proposed to address the above issues. The new approaches attempt to balance both fitting and generalization by attempting to minimize the total misclassification cost (*TC*) of the final system. These approaches first define the *TC* as an optimization problem in terms of the false-positive, false-negative, and unclassifiable rates along with their penalty costs. They then identify regions in the space of the training data where the training data points are located in either a homogeneous or convex manner. The new approaches are used in conjunction with traditional classification approaches. Next, a genetic approach is applied to determine the optimal levels of fitting and generalization. The *TC* expression is used as the fitness function in this genetic approach. The final models are an aggregate of the models inferred from such regions. By employing these new approaches, it is hoped that the classification / prediction accuracy of the inferred models will be very high, or at least as high as can be achieved, with the available training data.

Furthermore, one of the significant results which the HBA and CBA obtained shows that rates of classification errors may be arranged in stepwise patterns. Stepwise patterns of classification errors can be useful tools for researchers in determining penalties for classification errors. In fact, assume that for given penalties a classification approach returns some error rates. We want to know whether an error rate is retained, if the corresponding penalty is increased or decreased. If the rate is not retained, it should show penalties in which the level of the rate is changed. Stepwise patterns can assist in answering these concerns. Each step of a stepwise pattern shows an interval of penalties in which the error rate is identical. The different steps of the pattern depict penalties where the error rate varies across levels. This dissertation also proposes a Binary Search Approach (BSA) to determine such patterns.

The remainder of the dissertation is organized into seven chapters. The second chapter provides a preliminary description of the main research problem. The third chapter gives a summary of the major developments in the related literature. The HBA approach is highlighted in the fourth chapter. Meanwhile, the CBA approach is discussed in the fifth chapter. The sixth chapter discusses some promising results. These results give a good indication that these approaches may improve the performance of traditional classification approaches. The seventh chapter gives a description of the BSA and its results. Finally, the dissertation ends with some conclusions and proposed future work.

# CHAPTER 2. A FORMAL PROBLEM DESCRIPTION[1]

## 2.1 Some Basic Definitions

In order to clarify some important ideas, we first consider the hypothetical data depicted in Figure 1. Let us assume that the circles and squares in this figure correspond to sampled observations from two classes defined in 2-dimensions.

We assume that a data point is a vector defined on *D* variables along with their values. In Figure 1, *D* is equal to 2 and the two variables are indicated by the X and Y axes. In general, not all values may be known for a given data point. Data points describe the behavior of the system of interest to the analyst. The state space is the universe of all possible data points. In terms of Figure 1, the state space is any point in the X-Y plane.

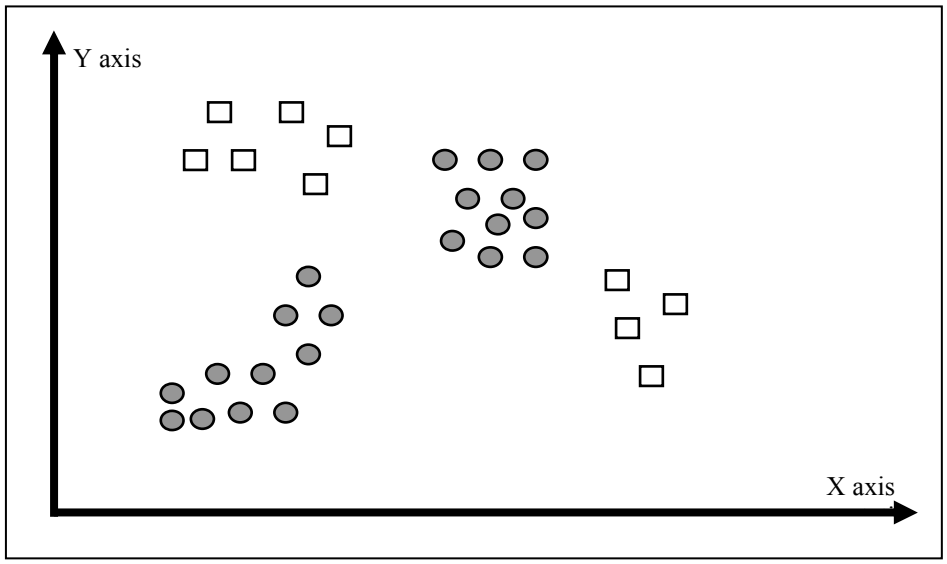



**Figure 1:** Sample data from two classes in 2-dimensions.

[1] A partial part of this chapter was in "Soft Computing for Knowledge Discovery and Data Mining" (O. Maimon and L. Rokach, Editors), Part 4, Chapter 5, Springer, New York, NY, USA, 2008, pp. 391-431.

We assume that there are only two classes. Arbitrarily, we will call one of them the positive class, while we will call the other one the negative class. Thus, a positive data point**,** also known as a positive example**,** is a data point that has been evaluated to belong to the positive class. A similar definition exists for negative data points (or negative examples).

Given a set of positive and negative examples, such as the ones depicted in Figure 1, this set is called the training data (or training examples) or the classified examples. The remaining data from the state space is called the unclassified data (or unclassified examples).

### 2.2. Problem Description

We start the problem description with a simple analysis of the sample data depicted in Figure 1. Suppose that a classification approach (such as a DT, ANN, or SVM) has been applied on these training data points. Next, we assume that two classification models have been inferred from these training data points. Usually, such classification models arrange the training dataset into groups described by the parts of a decision tree or classification rules. These groups of the training data points define regions inferred from the classification algorithm. For this hypothetical scenario, we assume that the classification algorithm has inferred the system regions depicted in Figure 2(a).

In general, one classification system describes the positive data points (thus we will call it the positive model), while the other system describes the negative data points (thus we will call it the negative model). In Figure 2(a), the positive model corresponds to regions A and B (which define the positive regions), while regions C and D correspond to the negative model (which defines the negative regions).

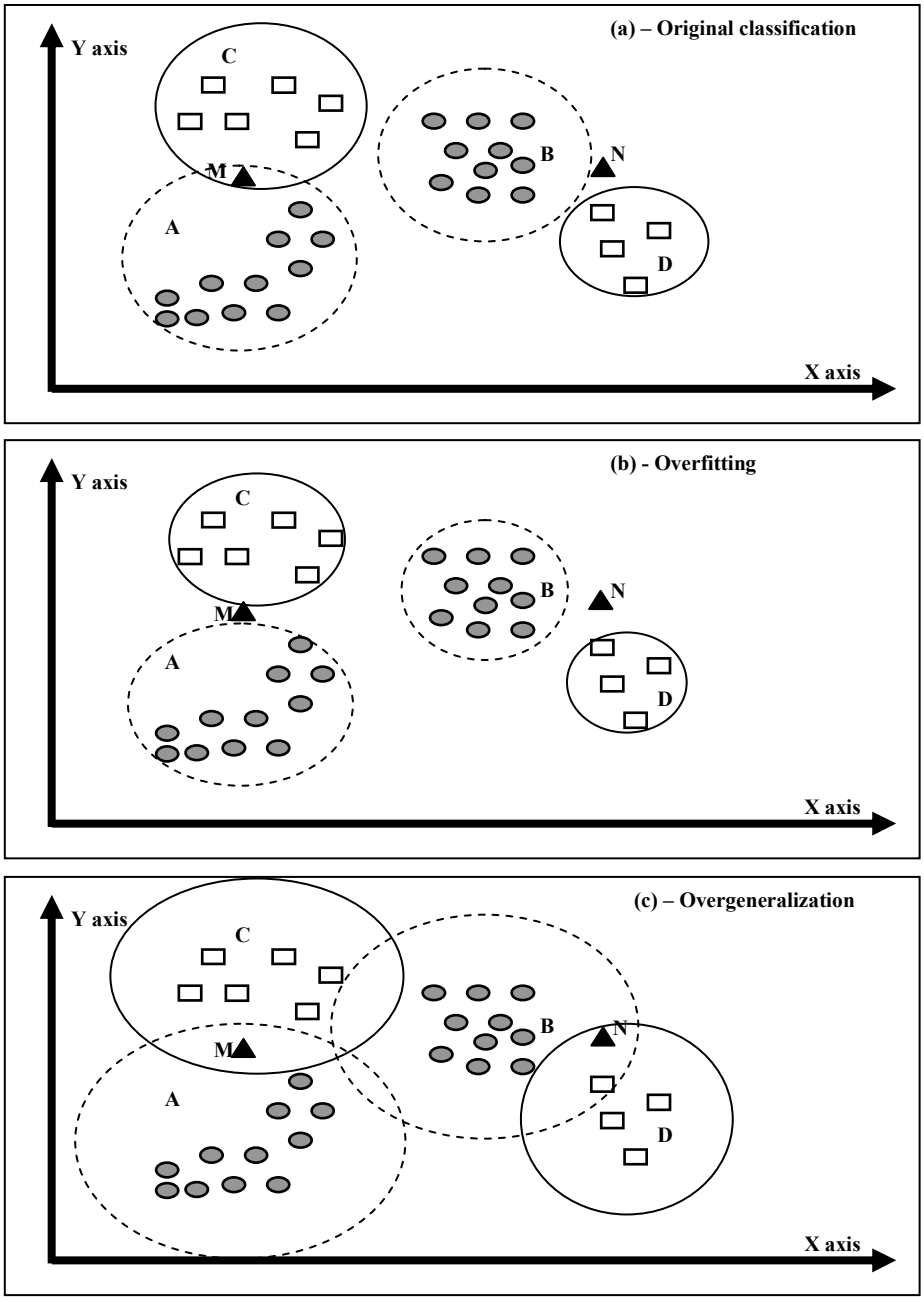



**Figure 2:** An example of the overfitting and overgeneralization problems.

Please recall that there are three different types of possible errors if one erroneously classifies a true positive point as negative or if one classifies a true negative point as positive. The first case is known as false-negative, while the second case is known as false-positive. Furthermore, a closer examination of Figure 2(a) indicates that there are some unclassifiable points which either are not covered by any of the regions or are covered by regions that belong to both models. For

instance, point N in Figure 2(a) (indicated as a triangle) is not covered by any of the regions, while point M (also a triangle) is covered by regions A and C which belong to the positive and the negative regions, respectively.

For the first case, since point N is not covered by any of the regions, the inferred model may declare point N as an unclassifiable point. In the second case, there is a direct disagreement by the inferred model as the new point (i.e., point M) in Figure 2(a) is covered simultaneously by regions of both classes. Again, such a point may also be declared as unclassifiable. Thus, in many real-life applications of classification one may have to consider three different penalty costs as follows: one cost for the false-positive type, one cost for the false-negative type, and one cost for the unclassifiable type.

Next consider Figure 2(b). Suppose that all regions A, B, C, and D have been reduced significantly but still cover the original training data. A closer examination of this figure indicates that now neither point M nor point N are covered by any of the inferred regions. In other words, these points and several additional points which were classified before by the inferred regions now become unclassifiable.

Furthermore, the data points which before were simultaneously covered by regions from both classes, and thus were unclassifiable, are now covered by only one type of region or none at all. Thus, it is very likely that the situation depicted in Figure 2(b) may have a higher total misclassification cost than the original situation depicted in Figure 2(a). If one takes this idea of reducing the covering regions as much as possible to the extreme, then there would be one region (i.e., just a small circle) around each individual training data point. In this extreme case, the total misclassification cost due to unclassifiable points would be maximal as the model would be able to classify the training data points only and nothing else. The previous scenarios are known as overfitting of the training data.

On the other hand, suppose that the original regions depicted as regions A, B, C, and D (as shown in Figure 2(a)) are now expanded significantly as in Figure 2(c). A closer examination of this figure demonstrates that points M and N are now covered simultaneously by regions of both classes. Also, more points are now covered simultaneously by regions of both classes. Thus, under this scenario we also have a large number of unclassifiable points because this situation creates many cases of disagreement between the two classification models (i.e., the positive and the negative models). This realization means that the total misclassification cost due to unclassifiable points will also be significantly higher than under the situation depicted in Figure 2(a). This scenario is known as overgeneralization of the training data.

Thus, we cannot separate the control of fitting and generalization into two independent studies. That is, we need to find a way to simultaneously balance fitting and generalization by adjusting the inferred models (i.e., the positive and the negative models) obtained from a classification algorithm. The balance of the two models will attempt to minimize the total misclassification cost (*TC*) of the final model.

In particular, let us denote $C_{FP}$, $C_{FN}$, and $C_{UC}$ as the penalty costs for the false-positive, the false-negative, and the unclassifiable cases, respectively. Let *RateFP*, *RateFN*, and *RateUC* be the false-positive, the false-negative, and the unclassifiable rates, respectively. Then, the problem is to achieve a balance between fitting and generalization that would minimize, or at least significantly reduce the total misclassification cost. The problem is defined in the following expression:

$$TC = \min\ (C_{FP} \times RateFP + C_{FN} \times RateFN + C_{UC} \times RateUC\ ). \qquad (1)$$

This methodology may assist the data mining analyst in creating classification models that would be optimal in the sense that their total misclassification cost would be minimized.

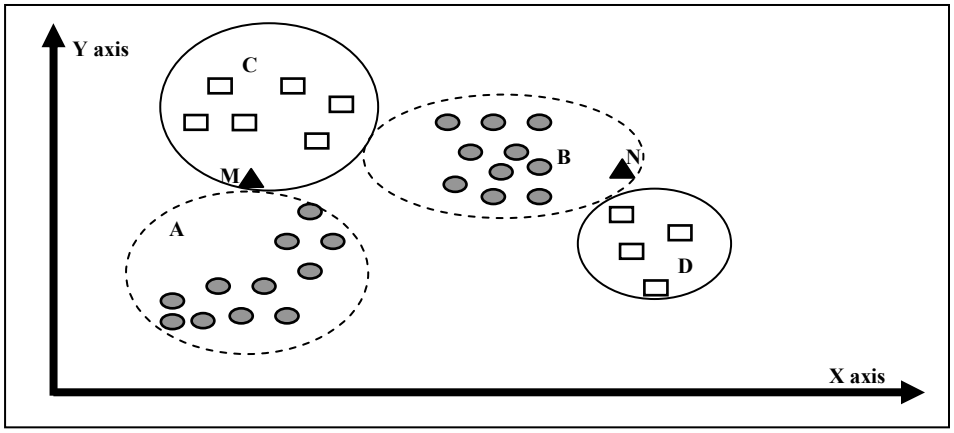



**Figure 3:** An example of a better classification.

In terms of Figures 2(a), (b), and (c), let us now consider the situation depicted in Figure 3. At this point, assume that in reality point M is negative, while point N is positive. Figure 3 shows different levels of fitting and generalization for the two classification models. For the sake of illustration, regions C and D are kept the same as in the original situation (i.e., as depicted in Figure 2(a)), while region A has been reduced (i.e., it fits the training data more closely) and now does not cover point M. On the other hand, region B is expanded (i.e., it generalizes the training data more) to cover point N. The new situation may correspond to a total misclassification cost that is smaller than the cost in any of the previous three scenarios. The following chapter provides a literature review of approaches designed for controlling the fitting and the generalization problems.

# CHAPTER 3. LITERATURE REVIEW[2]

The following classification approaches have typically focused on minimizing the sum of the false-positive and false-negative error rates by controlling either fitting or generalization. However, these approaches have failed to consider the two error rates in a weighted fashion.

## 3.1 Decision Trees (DTs)

There are two methods for controlling the overfitting problem in DTs: pre-pruning methods in which the growing tree approach is halted by some early stopping rules before generating a fully grown tree and post-pruning in which the DT is first grown to its maximum size and then some partitions of the tree are trimmed.

Recently much effort has focused on improving the pre-pruning methods. [Kohavi, 1996] proposed the NBTree (a hybrid of decision trees and naive-classifiers). The NBTree provides some early stopping rules by comparing two alternatives: partitioning the instance-space further on (i.e., continue splitting the tree based on some gain ratio stopping criteria) versus stopping the partition and producing a single Naïve Bayes classifier. [Zhou and Chen, 2002] suggested the hybrid DT approach for growing a binary DT. A feed-forward neural network is used to subsequently determine some early stopping rules. [Rokach, et. al., 2005] proposed the cluster-based concurrent decomposition (CBCD) algorithm. That algorithm first decomposes the training set into mutually exclusive sub-samples and then uses a voting scheme to combine these sub-samples for the classifier's predictions. Similarly, [Cohen, et. al., 2007] proposed an

[2] A partial part of this chapter was in "Soft Computing for Knowledge Discovery and Data Mining" (O. Maimon and L. Rokach, Editors), Part 4, Chapter 5, Springer, New York, NY, USA, 2008, pp. 391-431.

approach for building a DT by using a homogeneous criterion for splitting the space. However, the above approaches encounter difficulty in choosing the threshold value for early termination. A too large threshold value may result in underfitting models, while a too low threshold value may not be sufficient to overcome overfitting models.

Under the post-pruning approaches described in [Breiman, et. al., 1984] and [Quinlan, 1987], the pruning process eliminates several partitions of the tree. The reduction of the number of partitions makes the remaining tree more general. In order to demonstrate the main idea, we consider the simple example depicted in Figure 4. Suppose that Figure 4(a) shows a DT inferred from some training examples. The pruning process eliminates some of the DT 's nodes as depicted in Figure 4(b). The remaining part of the DT, as shown in Figure 4(c), implies some rules which are more general. For instance, the left most branch of the DT in Figure 4(a) implies the rule "if D $\wedge$ A $\wedge$ B $\wedge$ C, then …" On the order hand, Figure 4(c) implies the more general rule "if D $\wedge$ A, then …"

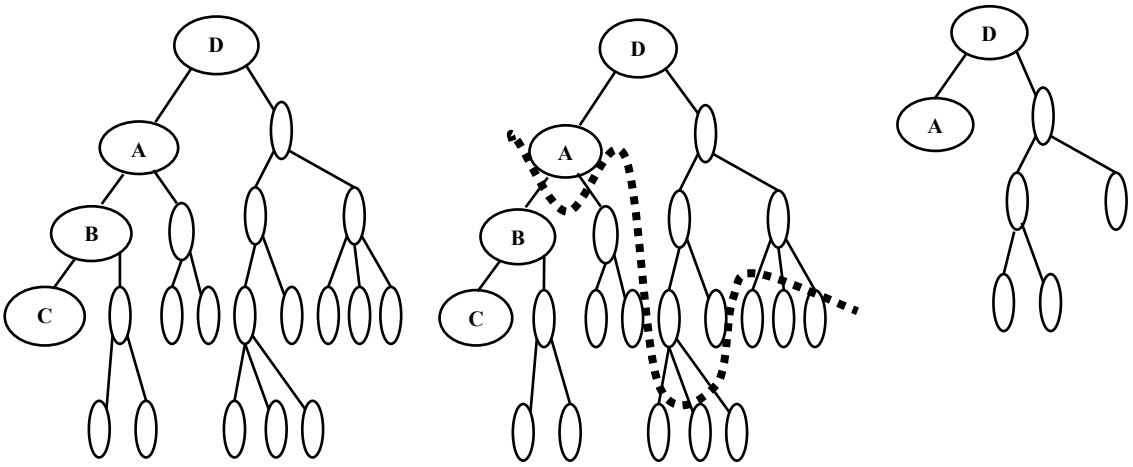



**Figure 4:** An example of DT pruning.

However, more generalization is not always required nor is it always beneficial. A more complex arrangement of partitions has been proved to increase the complexity of DTs in some applications. Furthermore, the treatment of generalization of a DT may lead to the overgeneralization problem since pruning conditions are based on localized information.

In addition to the pruning methods, there have been other developments to improve the accuracy of DTs. [Webb, 1996] attempted to graft additional leaves to a DT after its induction. This method does not leave any area of the instance space in conflict because each data point belongs to only one class. Obviously, the overfitting problem may arise from this approach. [Mansour, et. al., 2000] proposed another way to deal with the overfitting problem by using the learning theoretical method. In that method, the bounds on the error rate for DTs depend both on the structure of the tree and on the specific sample. [Kwok and Carter, 1990], [Schapire, 1990], [Wolpert, 1992], [Dietterich and Bakiri, 1994], [Ali, et. al., 1994], [Oliver and Hand, 1995], [Nock and Gascuel, 1995] and [Breiman, 1996] allowed multiple classifiers used in a conjunction. The above methods are similar to using a Disjunctive Normal Form (DNF) Boolean function. Furthermore, [Breiman, 2001] also used the so-called random forest approach for multiple classifiers. However, the above approaches might create conflicts among the individual classifiers' partitions, as in the situation presented in C4.5 [Quinlan, 1993].

### 3.2 Rule-Based Classifiers

A rule-based classifier uses a collection of "if … then …" rules that identify key relationships between the attributes and the class values of a training dataset. There are two methods which infer classification rules. First, direct methods which infer classification rules directly from the training dataset and, secondly, indirect methods which infer classification rules from other classification methods such as DTs, SVMs, or ANNs and then translate the final

model into a set of classification rules [Tan, et. al., 2005]. An extensive survey of rule-based methods can be found in [Triantaphyllou and Felici, 2006]. A new rule-based approach, which is based on mathematical logic, is described in [Triantaphyllou, 2010].

A well-known algorithm of direct methods is the Sequence Covering algorithm and its later enhancement, the CN2 algorithm [Clark and Niblett, 1989]. To control the balance of fitting and generalization while generating rules, these algorithms first use one of two strategies for growing the classification rules: general-to-specific or specific-to-general. Then, the rules are refined through the pre- and post-pruning methods mentioned in DTs.

Under the general-to-specific strategy, a rule is created by finding all possible candidates and using a greedy approach to choose the new conjuncts to be added into the antecedent part of the rule in order to improve its quality. This approach ends when some stopping criteria are met.

Under the specific-to-general strategy, a classification rule is initialized by randomly choosing one of the positive data points as the initial step. Then, the rule is refined by removing one of its conjuncts so that this rule can cover more positive points. This refining approach ends after the satisfaction of certain stopping criteria. A similar method exists for the negative data points. There are some related developments regarding these strategies. Such developments include the beam search algorithm [Clark and Boswell, 1991] which avoids the overgrowing of rules as a result of the greedy behavior and the RIPPER algorithm [Cohen, 1995] which uses a rule induction algorithm.

Furthermore, the rules derived by the above methods are then treated by an approach described in [Mastrogiannis, 2009]. Specifically, the approach first estimates the resemblance between each data point in the calibration dataset and the rules. Next, the results are used to exclude redundant attributes and rules.

However, the use of the two strategies for growing classification rules has its disadvantages. First, the complexity for finding optimal rules is of exponential size of the search space. Secondly, although some rule pruning methods are used to minimize their generalization error, they also leave weaknesses as mentioned in the case of DTs.

### 3.3 *K*-Nearest Neighbor Classifiers

While DTs and rule-based classifiers are examples of eager learners, *K*-Nearest Neighbor Classifiers [Cover and Hart, 1967] and [Dasarathy, 1979] are known as lazy learners. That is, this approach finds *K* training points that are relatively similar to attributes of a testing point to determine its class value.

The importance of choosing the right value for *K* directly affects the accuracy of this approach. A wrong value for *K* may lead to either the overfitting or the overgeneralization problems [Tan, et. al., 2005]. One method to reduce the impact of *K* is to weight the influence of the nearest neighbors according to their distance from the testing point. One of the most well-known schemes is the distance-weighted voting scheme [Dudani, 1976] and [Keller, Gray and Givens, 1985].

However, the use of *K*-Nearest Neighbor Classifiers also has its limitations. First, classifying a test example can be quite expensive due to the need to compute a similarity degree between the testing point and each training point. Secondly, these classifiers are unstable since they are based only on localized information. Finally, it is difficult to find an appropriate value for *K* to avoid the overfitting and overgeneralization problems.

### 3.4 Bayes Classifiers

Bayes classifiers use the modeling probabilistic relationships between the attribute set and the class variable for solving classification problems. There are two well-known implementations of Bayesian classifiers: Naïve Bayes (NBs) and Bayesian Belief Networks (BBNs).

NBs assume that all the attributes are conditionally independent, given the value of the class variable. Next, they estimate the class conditional probabilities. This independence assumption, however, is problematic because in many real applications there are strong conditional dependencies between the attributes. Furthermore, when using the independence assumption, NBs may suffer from the overfitting problem since they are based on localized information.

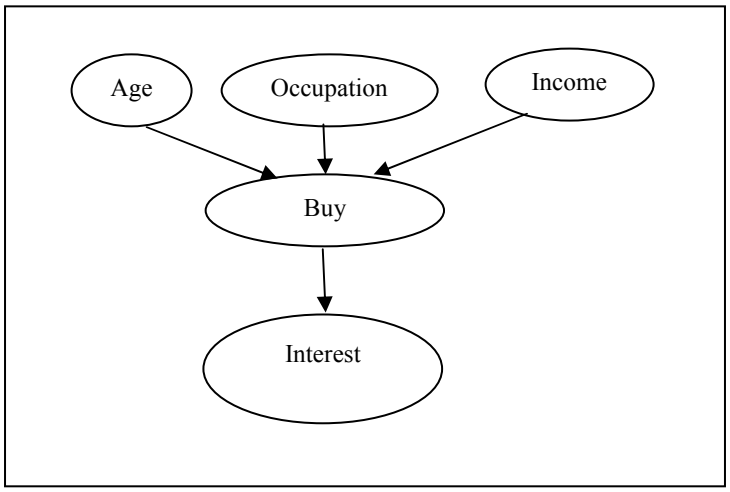



**Figure 5:** An example of a BBN [Rada, 2004].

BBNs [Duda and Hart, 1973] allow for pairs of attributes to be conditionally independent when the value of the class variable is known. Let us consider the following example. Suppose that we have a training dataset consisting of the following attributes: age, occupation, income, buy (i.e., buy some product X), and interest (i.e., "interest in purchasing insurance for product X"). The attributes age, occupation, and income may determine if a customer will buy some

product X. Given is a customer who has bought product X. There is an interest in buying insurance when we assume this is independent of age, occupation, and income. These constraints are presented by the BBN depicted in Figure 5. Thus, for a certain data point described by a 5-tuple (age, occupation, income, buy, interest), its probability based on the BBN should be:

P(age, occupation, income, buy, interest) =

P(age) × P(occupation) × P(income) × P(buy | age, occupation, income) × P(interest | buy).

Considerable efforts have been focused on improving BBNs. These efforts followed two general approaches: selecting a feature subset [Langley and Sage, 1994], [Pazzani, 1995], and [Kohavi and John, 1997] and relaxing the independent assumptions [Kononenko, 1991] and [Friedman, et. al., 1997]. However, these approaches have the following weaknesses. First, they require a large amount of effort when constructing the network. Secondly, the approaches quietly degrade to the overfitting problem because they combine probabilistically the data with prior knowledge.

### 3.5 Artificial Neural Networks (ANNs)

Recall that an ANN is a model that is an assembly of inter-connected nodes and weighted links. The output node sums up each of its input values according to the weights of its links. The output node is compared against a threshold value *t*. Such a model is illustrated in Figure 6. The ANN in this figure consists of the three input nodes $X_1$, $X_2$, and $X_3$ which correspond to the weighted links $W_1$, $W_2$, and $W_3$, respectively, and one output node $Y$. The sum of the input nodes can be $Y = \text{sign} \sum_i (X_i W_i - t)$, and is called the perceptron model [Abdi, 2003].

In general, an ANN has a set of input nodes $X_1$, $X_2$, …, $X_m$ and one output node $Y$. Given are $n$ values for the $m$-tuple ($X_1$, $X_2$, …, $X_m$). Let $\hat{Y}_1$, $\hat{Y}_2$, …, $\hat{Y}_n$ be the predicted outputs and $Y_1$, $Y_2$,

…, $Y_n$ be the expected outputs from the $n$ values, respectively. Let $E = \sum_{i=1}^{n}[Y_i - \hat{Y}_i]^2$ denote the total sum of the squared differences between the expected and the predicted outputs. The goal of the ANN is to determine a set of the weights in order to minimize the value of $E$. During the training phase of an ANN, the weight parameters are adjusted until the outputs of the perceptron become consistent with the true outputs of the training points. In the weight update process, the weights should not be changed too drastically because $E$ is computed only for the current training point. Otherwise, the adjustments made during earlier iterations may be undone.

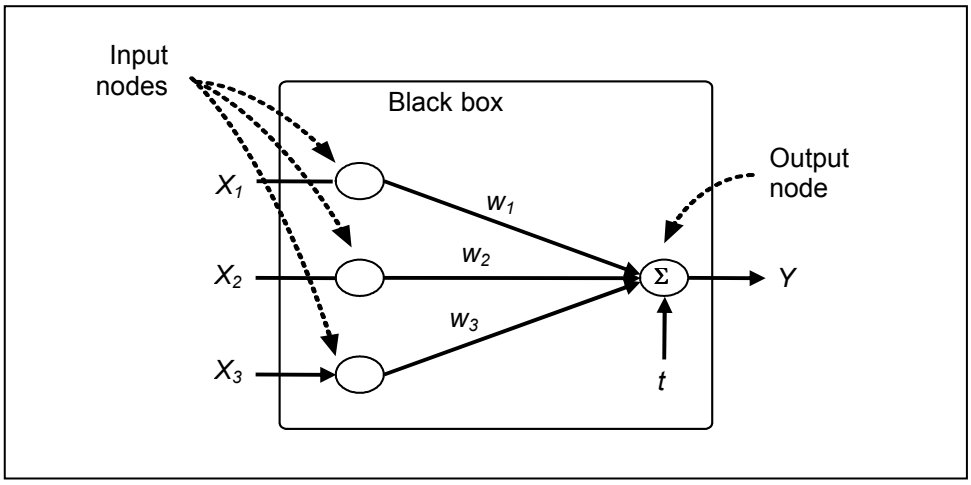



**Figure 6:** An example of an ANN [Tan, et. al., 2005].

In order to avoid the overfitting and overgeneralization problems, the design for an ANN must be considered. A network that is not sufficiently complex may fail to fully detect the input in a complicated dataset, leading to the overgeneralization problem. On the other hand, a network that is too complex may not only fit the input but also the noisy points, thus leading to the overfitting problem. According to [Geman, et. al., 1992] and [Smith, 1996], the complexity of a network is related both to the number of the weights and to the size of the weights. Geman and Smith were either directly or indirectly concerned with the number and size of the weights.

That is, the number of the weights relates to the number of hidden units and layers. The more weights there are, relative to the number of the training cases, the more overfitting amplifies noise in the classification systems [Moody, 1992]. Reducing the size of the weights may reduce the effective number of the weights leading to weight decay [Moody, 1992] and early stopping [Weigend, 1994].

In summary, ANNs have the following disadvantages. First, it is difficult to find an appropriate network topology for a given problem in order to avoid the overfitting and overgeneralization problems. Secondly, it takes considerable time to train an ANN when the number of hidden nodes is large.

### 3.6 Support Vector Machines (SVMs)

Another classification technique that has received considerable attention is known as SVMs [Cortes and Vapnik, 1995]. The basic idea behind SVMs is to find a maximal margin hyperplane, $\theta$, that will separate points considered as vectors in the $D$-dimensional space. The maximum margin hyperplane can be essentially represented as a linear combination of the training points. Consequently, the decision function for classifying new data points with respect to the hyperplane only involves dot products between data points and the hyperplane.

In order to illustrate the ideas, we consider the simple example depicted in Figure 7. Suppose that we have a training dataset defined on two given classes (represented by the squares and circles) in 2-D. In general, the approach can find many hyperplanes, such as $B_1$ or $B_2$, separating the training dataset into the two classes. The SVM, however, chooses $B_1$ to classify this training dataset since $B_1$ has the maximum margin. Roughly speaking, $B_1$ maximally separates the two groups of training examples.

Decision boundaries with maximal margins tend to lead to better generalization. Furthermore, SVMs attempt to formulate the learning problem as a convex optimization problem in which efficient algorithms are available to find a global solution. For many datasets, however, an SVM may not be able to formulate the learning problem as a convex optimization problem because of excessive misclassifications. Thus, the attempts for formulating the learning problem may lead to the overgeneralization problem. The following chapter will describe one of the proposed approaches which can address the limitations of current classification approaches.

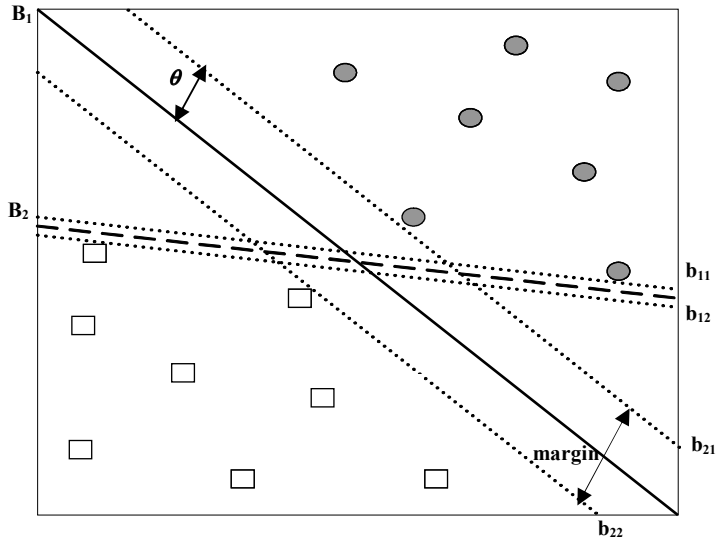



**Figure 7:** An example of an SVM [Tan, *et. al.*, 2005].

# CHAPTER 4. THE HOMOGENEITY-BASED ALGORITHM (HBA)[3]

## 4.1 Some Key Observations

The HBA assumes that all attributes in a dataset are numerical. If the data are not numerical, then there are procedures for converting them into equivalent numerical data. This cannot be done for all cases as ordering relations in the converted numerical data may impose undesirable effects into the data. The interested reader may refer to the recent book [De Vaus, 2002] for a discussion on some related key problems in data analysis.

In order to explain the HBA, we first consider the situation depicted in Figure 8(a). This figure presents two inferred regions. These are the circular areas that surround groups of training data (shown as small circles). Actually, these data are part of the training data shown earlier in Figure 1. The circles in Figure 1 represent positive points. Moreover, in Figure 8(a) there are two additional data points shown as small triangles which are denoted as points P and Q. At this time it is assumed that we do not know the actual class values of these two new points. We would like to use the available training dataset and inferred regions to classify these two points. Because points P and Q are covered by regions A and B, respectively, both of these points may be assumed to be positive examples.

Let us look more closely at region A. This region covers areas of the state space that are not adequately populated by positive training points. Such areas, for instance, exist in the upper left corner and the lower part of region A (see Figure 8(a)). It is possible that the unclassified points which belong to such areas are erroneously assumed to be of the same class as the positive

[3] A partial part of this chapter was in Expert Systems with Applications, Vol. 36, No. 5, 2009, pp. 9240-9249.

training points covered by region A. Point P is in one of these sparsely covered areas under region A. Thus, the assumption that point P is a positive point may not be very accurate.

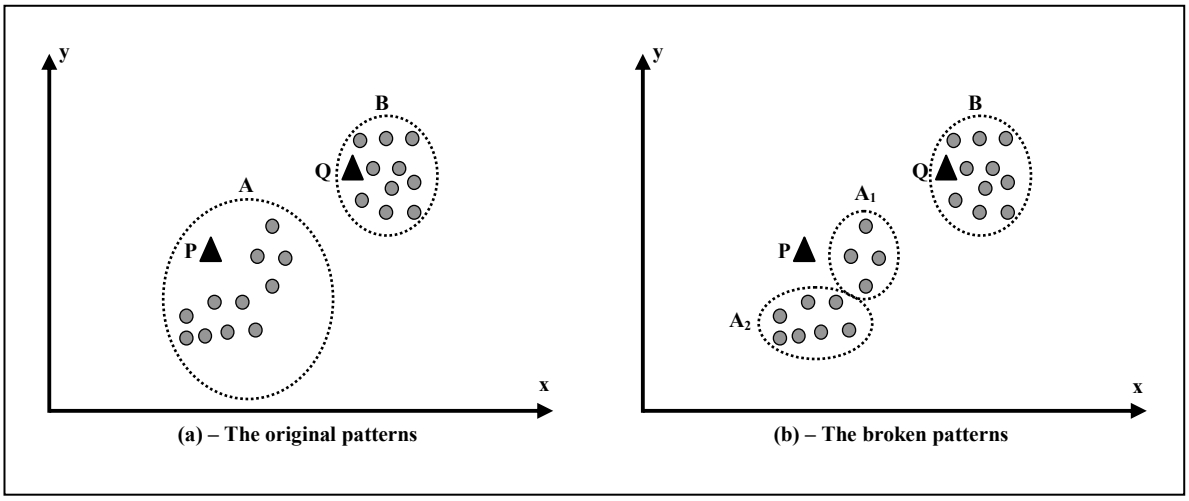



**Figure 8:** Region B is a homogeneous region, while region A is a non-homogeneous region. Region A can be broken into the two homogeneous regions A1 and A2 as shown in part (b).

On the other hand, region B does not have such sparsely covered areas (see also Figure 8(a)). Thus, it may be more likely that the unclassified points covered by region B are more accurately assumed to be of the same class as the positive training points covered by the same region. For instance, the assumption that point Q is a positive point may be more accurate.

The above simple observations lead to an assumption that the accuracy of the inferred models may be increased if the derived regions are somehow more compact and homogeneous [Pham and Triantaphyllou, 2008a, 2008b, and 2009a]. Based on [Webster Dictionary, 2010], given a certain class (i.e., positive or negative), a homogeneous region describes a steady or uniform distribution of a set of distinct points.

The point pattern analysis [Greig-Smith, 1952], which is called the Quadrat analysis approach, has also discussed the concept of homogeneous regions. This approach states that within a pattern there are no regions (also known as bins) with unequal concentrations of

classified (i.e., either positive or negative) and unclassified points. In other words, if a pattern is partitioned into smaller bins of the same unit size and the density of these bins is almost equal to each other (or, equivalently, the standard deviation is small enough), then this pattern is a homogeneous region. According to [Pham and Triantaphyllou, 2008a, 2008b, and 2009a], an axiom and a theorem are derivable from the definition of a homogeneous region as follows:

**Axiom 1**: Given an inferred region $P$ of size one, then $P$ is a homogeneous region.

**Theorem 1**: Let us consider a homogeneous region $P$. If $P$ is divided into two parts, $P_1$ and $P_2$, then these two parts are also homogeneous regions.

Proof: We prove Theorem 1 by using contradiction. Since $P$ is a homogeneous region, there is a uniform random variable $Z$ which represents the distribution of points within $P$. Similarly, $Z_1$ and $Z_2$ are the two random variables that represent the distribution of points within $P_1$ and $P_2$, respectively. Obviously, Z is the sum of $Z_1$ and $Z_2$. Assume that either $P_1$ or $P_2$ is a non-homogeneous region. Then, $Z_1 + Z_2$ is not a uniform random variable. This contradicts the fact that $Z$ is a uniform random variable.

Looking once more at Figure 8, assume that the region which is represented by the non-homogeneous region A in Figure 8(a) can be replaced by two more homogeneous regions denoted as $A_1$ and $A_2$ in Figure 8(b). Now region A is covered by the two new smaller regions $A_1$ and $A_2$ which are more homogeneous than the area covered by the original region A. Given this consideration, point P may be assumed to be an unclassifiable point, while point Q is still a positive point.

As presented in the previous paragraphs, the homogeneous property of regions may influence the number of misclassification cases of the inferred models. Furthermore, if a region is homogeneous, then the number of training points covered by this region may be another factor which affects the accuracy of the overall inferred models. For instance, Figure 9 shows the case illustrated in Figure 8(b) (i.e., region A has been replaced by two more homogeneous regions denoted as $A_1$ and $A_2$). Suppose that all regions $A_1$, $A_2$, and B are homogeneous regions and a new point S (indicated as a triangle) is covered by region $A_1$.

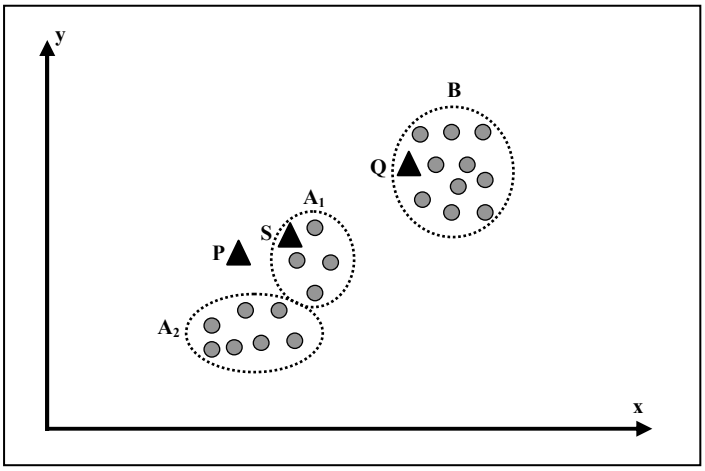



**Figure 9:** An example of homogeneous regions.

A closer examination of Figure 9 shows that the number of points in B is greater than those in $A_1$. Although both points Q and S are covered by homogeneous regions, the assumption that point Q is a positive point may be more accurate than the assumption that point S is a positive point. This observation leads one to surmise that the accuracy of the inferred models may also be affected by a density measure. Such a density could be defined as the number of points in each inferred region per unit of area or volume. This density is also called the homogeneity degree (*HD*) [Pham and Triantaphyllou, 2008a, 2008b, and 2009a].

In summary, a fundamental assumption here is as follows: if an unclassified point is covered by a region which is homogeneous and also happens to have a high value for *HD*, then it may be more accurately assumed to be of the same class as the points covered by that region. In other words, the accuracy of the inferred models may be increased when their regions are more homogeneous and have high values for *HD*s.

### 4.2 Non-Parametric Density Estimation

Please recall that a region $P$ of size $n$ is a homogeneous region if the region can be partitioned into smaller bins of the same unit size $h$ and the standard deviation of the density of the bins is small enough. In other words, if $P$ is superimposed by a hypergrid of unit size $h$ and the density of the bins inside $P$ is almost equal to each other, then $P$ is a homogeneous region.

The density estimation of a typical bin plays an important role in determining whether a region is homogeneous or not. According to [Duda, et. al., 2001], the density estimation is the construction of an estimate. That is, it is based on observed data points and on unobservable data points under a probability density function. There are two basic approaches for the density estimation:

- Parametric in which we assume a given form of the density function (e.g., Gaussian and Gamma) and its parameters (i.e., its mean and variance) such that this function may optimally fit the model to the training dataset.
- Non-parametric in which we cannot assume a functional form for the density function, and the density estimates are driven entirely by the available training dataset.

In particular, the non-parametric approaches first divide region $P$ into a number of small bins, called *R*s, of unit size $h$. Next, these approaches estimate the density for a center point $x$ of bin $R$ by using Equation (2). Please note that $d(x)$ denotes the density for center $x$.

$$d(x) = \frac{1}{n}[\frac{\text{The number of data points falling in } R \text{ with center } x}{R' s \text{ volume}}]. \quad (2)$$

As seen in the above equation, $d(x)$ relies on the probability that points will fall in $R$. By using this idea, $d(x)$ is indicated as follows:

$$d(x) \approx \frac{k}{n \times V} \quad , \quad (3)$$

where $k$ is the number of points which fall in $R$ and $V$ is the volume enclosed by $R$.

One of the most appropriate approaches for the non-parametric density estimation is Parzen Windows [Duda, et. al., 1973]. The Parzen Windows approach temporarily assumes that a typical bin is a $D$-dimensional hypercube of unit size $h$. To find the number of points which fall within $R$, the Parzen Windows approach defines a kernel function $\varphi(u)$ as follows:

$$\varphi(u) = \begin{cases} 1, & |u| \le 1/2. \\ 0, & otherwise. \end{cases} \quad (4)$$

It follows that the quantity $\varphi(\frac{x - x_i}{h})$ is equal to unity if the point $x_i$ is inside the hypercube (centered at $x$) of unit size $h$, and zero otherwise. In the $D$-dimensional space, the kernel function can be presented as follows:

$$\varphi(\frac{x - x_i}{h}) = \prod_{j=1}^{D} \varphi(\frac{x^j - x_i^j}{h}). \quad (5)$$

Substitute Equation (5) into Equation (3), then:

$$d(x) \approx \frac{1}{n \times h^D} \sum_{i=1}^{n} \prod_{j=1}^{D} \varphi(\frac{x^j - x_i^j}{h}). \quad (6)$$

The following sections in this chapter will use the Parzen Windows approach for the density estimation. A right value for $h$ plays the role of a smoothing parameter in the Parzen Windows approach. That is, if $h \to \infty$, then the density at point $x$ in $P$ (i.e., $d(x)$) approaches a false density.

As $h \rightarrow 0$, then the kernel function approaches the Dirac Delta Function and *d(x)* approaches to the true density [Bracewell, 1999].

Suppose that we determine all distances between all possible pairs formed by taking any two points from region *P*. For easy illustration, assume that for region *P* which contains 5 data points these distances are as follows: 6, 1, 2, 2, 1, 5, 2, 3, 5, 5. Then, we define *S* as a set of the distances which have the highest frequency. For the previous illustration, we have set *S* equal to {2, 5} as both distances 2 and 5 occur with frequency equal to 3. By using set *S*, [Pham and Triantaphyllou, 2008a, 2008b, and 2009a] proposed Heuristic Rule 1 to find an appropriate value for *h* when estimating the density *d(x)*. In particular, the heuristic rule uses the minimum value in *S* (which is equal to 2 in the previous illustration) as follows:

**Heuristic Rule 1**: If *h* is equal to the minimum value in set *S* and this value is used to compute $d(x)$ by using Equation (6), then $d(x)$ can approach to the true density.

This heuristic rule is applicable for the following reason. In practice, since region *P* has a finite number of points, the value for *h* cannot be made arbitrarily small. Obviously, an appropriate value for *h* is between the maximum and the minimum distances that are computed by all pairs of points in region *P*. If the value for *h* is the maximum distance, then *P* would be inside a single bin. Thus, *d(x)* approaches to a false density. In contrast, if the value for *h* is the minimum distance, then the set of the bins would degenerate to the set of the single points in *P*. This situation also leads to a false density and thus it is misleading.

According to [Bracewell, 1999], as $h \rightarrow 0$, then *d(x)* approaches to the true density. Furthermore, a small value for *h* would be appropriate to approach to the true density [Duda, et. al., 2001]. Thus, the value for *h* described in Heuristic Rule 1 is a reasonable selection because it

is close to the minimum distance, but at the same time the bins would not degenerate to regions which only include *P*'s points. The following section describes the HBA which applies the concepts discussed above.

### 4.3 The HBA

We assume that two classification models denoted as $M_1$ (one for the positive data and the other for the negative data) and the training dataset *T* are given. The desired goal of the HBA is to enhance models $M_1$ in order to obtain an optimal *TC* as described in Equation (1). There are five parameters used in the HBA:

- Two expansion coefficients $\alpha^+$ and $\alpha^-$ to be used when expanding positive and negative homogeneous regions, respectively.
- A breaking threshold value $\beta^+$ to be used when determining whether a positive homogeneous region is broken. A similar concept is applied on the breaking threshold value $\beta^-$ for a negative homogeneous region.
- A density threshold value $\gamma$ to be used when determining whether a region is homogeneous.

The main steps of the HBA are described in Figure 10 and are summarized in terms of the following phases:

- **Phase 1** (Steps 1 and 2): Normalize the values of the attributes in *T*. Normalization is due to the fact that the Euclidean distance is used in the HBA. This distance requires that *T* 's attributes should be defined in the same unit (i.e., as percentages). Next, the HBA divides *T* into two random sub-datasets: $T_1$ whose size is, say 90% of *T* 's size and $T_2$ whose size, in this case, is the remaining 10% of *T* 's size. These percentages can be determined empirically with trial and error. Suppose that models $M_1$ group $T_1$'s training points into a set of regions. Finally,

the HBA randomly initializes the four parameters ($\alpha^+$, $\alpha^-$, $\beta^+$, $\beta^-$) whose ranges depend on each individual application.

- **Phase 2** (Step 3): Break regions into hyperspheres. We use the concept of hyperspheres because each hypersphere is determined by a center and a radius. Such hyperspheres do not depend on the dimensionality of the training dataset.
- **Phase 3** (Steps 4 to 6): Determine whether the hyperspheres derived in Phase 2 are homogeneous. If so, then compute their *HD* values. Otherwise, break non-homogeneous regions into smaller hyperspheres. Some homogeneous regions are broken again if their *HDs* are less than either $\beta^+$ (for positive regions) or $\beta^-$ (for negative regions).
- **Phase 4** (Step 7): Expand the hyperspheres obtained in Phase 3 in decreasing order of the values for *HD*. The expansion algorithms are shown in Section 4.3.3.
- **Phase 5** (Step 8): Apply the genetic algorithm (GA) described in Section 4.3.4 to Phases 3 and 4 with Equation (1) as the fitness function and $T_2$ as the calibration dataset. The GA approach finds the four optimal parameters ($\alpha_*^+$, $\alpha_*^-$, $\beta_*^+$, $\beta_*^-$), for the three given cost coefficients $C_{FP}$, $C_{FN}$, and $C_{UC}$.
- **Phase 6** (Step 9): Use the optimal parameters ($\alpha_*^+$, $\alpha_*^-$, $\beta_*^+$, $\beta_*^-$) on the entire training dataset *T* to repeat Phases 3 to 4 and infer the final pair of models $M_2$.

The above phases can lead to the formulation of five sub-problems as follows:

- Sub-Problem 1: Break $M_1$'s regions into hyperspheres (see an example depicted in Figure 11).
- Sub-Problem 2: Determine whether a hypersphere is a homogeneous region.
- Sub-Problem 3: Break a non-homogeneous region into hyperspheres.
- Sub-Problem 4: Break a homogeneous region into smaller homogeneous regions.
- Sub-Problem 5: Expand a hypersphere.

**Input**: The positive and negative classification models $M_1$, the training dataset $T$, the density threshold value $\gamma$, and the three cost coefficients $C_{FP}$, $C_{FN}$, and $C_{UC}$.

1. Normalize $T$ and divide $T$ into two random sub-datasets: the actual training dataset $T_1$ and the calibration dataset $T_2$. Suppose that models $M_1$ group $T_1$ into a set of regions.
2. Randomly initialize the values of the four parameters ($\alpha^+$, $\alpha^-$, $\beta^+$, $\beta^-$).
3. Break $M_1$'s regions into hyperspheres. This task is called Sub-Problem 1.
4. Determine whether the hyperspheres are homogeneous regions. This task is called Sub-Problem 2. Then we break non-homogeneous regions (if any) into homogeneous regions. This task is called Sub-Problem 3.
5. Compute the *HD*s for the homogeneous regions.
6. Break the homogeneous regions into sub-homogeneous regions, if their *HD*s are less than $\beta^+$ or $\beta^-$ for the positive and negative data, respectively. This task is called Sub-Problem 4.
7. In decreasing order of the *HD*s, expand hypersphere $P$ by using *HD(P)* and either $\alpha^+$ or $\alpha^-$ for the positive and negative data, respectively. This task is called Sub-Problem 5.
8. Apply the GA approach to Steps 6 and 7 using Equation (1) as the fitness function and the calibration dataset $T_2$. The GA approach finds the optimal values of the four parameters denoted as $(\alpha_*^+, \alpha_*^-, \beta_*^+, \beta_*^-)$.
9. Use the optimal values $(\alpha_*^+, \alpha_*^-, \beta_*^+, \beta_*^-)$ on the entire training dataset $T$ to repeat Steps 6 and 7 and infer the final pair of models $M_2$.

**Output**: The positive and negative classification pair of models $M_2$.

**Figure 10:** The HBA [Pham and Triantaphyllou, 2008a, 2008b, and 2009a].

Because the task of Sub-Problems 1 and 3 are to break a region into hyperspheres, they use the same algorithm. Furthermore, according to Theorem 1, sub-regions broken from a homogeneous region are also homogeneous. Thus, Sub-Problem 4 also uses the same algorithm as Sub-Problems 1 and 3. The following sections present some procedures for solving Sub-Problems 1, 2, 5, and also the GA approach.

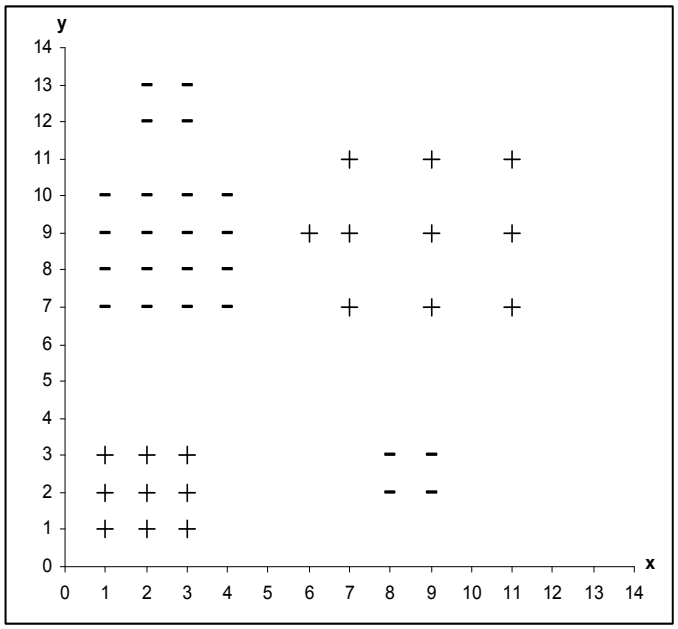
### 4.3.1 Sub-Problem 1

In order to illustrate the following sections, we consider the situation depicted in Figure 11(a). This figure presents two training sets (called the training dataset $T_1$) of positive and negative training points in 2-D. Suppose that the classification models $M_1$ has applied a DT algorithm on the training dataset $T_1$ to infer a decision tree as depicted in Figure 11(b). This decision tree separates the training data into the four regions delineated by the two bold lines depicted in Figure 11(c).

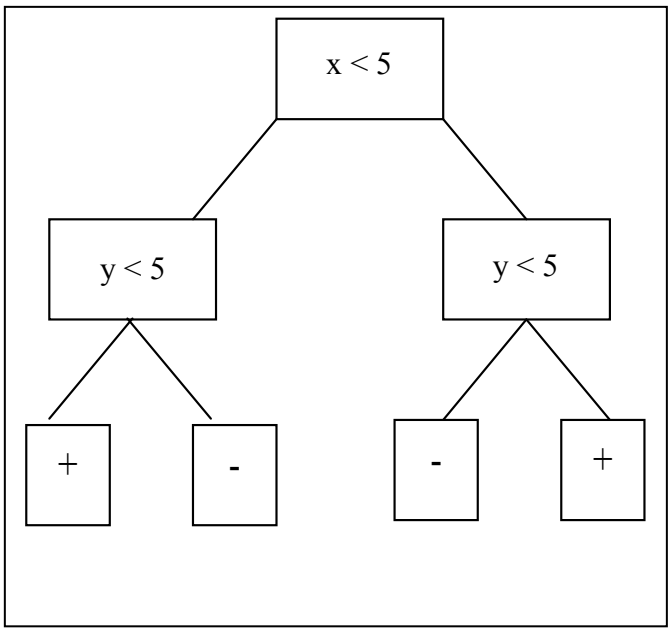
Let us consider the example in Figure 11(c). For each region in this figure, somehow Sub-Problem 1 finds the minimum number of hyperspheres (i.e., circles in 2-D) which can cover all the points in the original region. For instance, this situation is depicted in Figure 11(d) in which the positive and negative points are covered by the corresponding circles (please note that in 2-D hyperspheres are circles): A, B, C, D, and E.

The problem of finding the minimum number of hyperspheres that can cover a region $P$ of size $n$ is similar to a form of the set cover problem, an NP-complete problem [Karp, 1972]. In this dissertation, a heuristic algorithm is proposed as depicted in Figure 12.

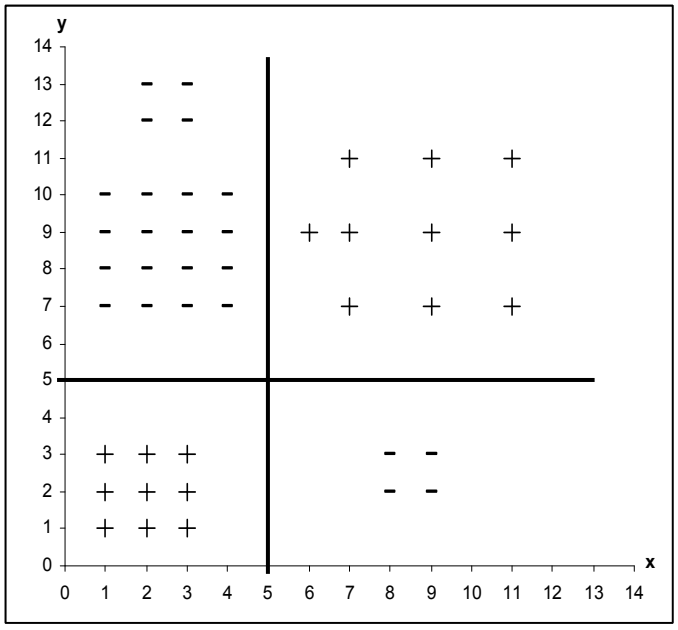
The algorithm starts by first estimating the densities of the $n$ points by using Equation (6) in Section 4.2. We assume that the value for $K$ is going from 1 to $n$. The algorithm will pick $K$ points in $P$ with the highest densities. Next, it uses these $K$ points as centroids in the $K$-means



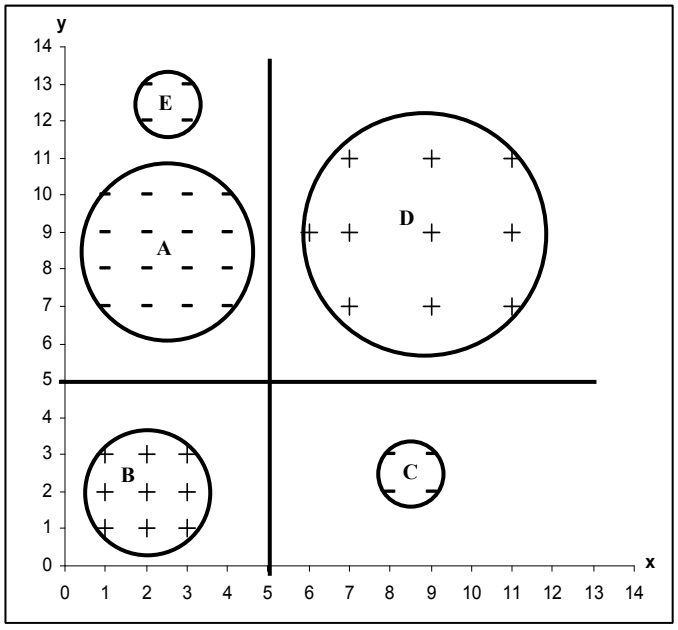

(a) – The original training data.




(b) – An inferred DT.




(c) – The inferred patterns from the DT.




(d) – The hyperspheres (i.e., circles in 2-D) from the DT.

**Figure 11:** An example for Phase 2.

clustering approach [Seber, 1984]. In particular, the *K*-means approach uses a two-phase iterative algorithm to minimize the sum of point-to-centroid distances, summed over all *K* clusters. Each iteration in the first phase consists of reassigning data points to their nearest cluster centroid, all at once, followed by recalculation of cluster centroids. In the second phase, data points are individually reassigned if doing so will reduce the sum of distances and cluster centroids are recomputed after each reassignment. If the *K* hyperspheres which are obtained from the clustering approach cover *P*, then the algorithm will stop. Otherwise, we repeat the algorithm with the value for *K* increased by one.

**Input:** Region *P* of size *n*.

1. Estimate the densities of the *n* points by using Equation (6).
2. For *K* = 1 to *n* do
3. Begin {for}
4. Pick *K* points in *P* with the highest densities.
5. Use the *K*-means clustering approach to find *K* hyperspheres.
6. If the *K* hyperspheres cover *P*, then STOP.
7. End {for}.

**Output**: *K* hyperspheres.

**Figure 12:** The algorithm for Sub-Problem 1.

For instance, in Figure 11(d) the algorithm determines two circles which can cover the two positive regions, while it uses three circles for the two negative regions. Please note that the algorithm depicted in Figure 12 is also applied to Sub-Problems 3 and 4.

### 4.3.2 Sub-Problem 2

Let us consider some hypersphere $P$. Sub-Problem 2 determines whether hypersphere $P$ is homogeneous. By using the idea of the Quadrat analysis approach described in [Greig-Smith, 1952], hypersphere $P$ is first divided into a number of small bins of unit size $h$. Next, we approximate the density at the center $x$ of each bin. The value $h$ is defined as in Heuristic Rule 1 in Section 4.2. If the densities at the centers are approximately equal to each other, then $P$ is a homogeneous region. The algorithm for Sub-Problem 2 is given in Figure 13.

In order to explain the main idea of the algorithm for Sub-Problem 2, we first consider the situation depicted in Figure 14. The left side of this figure presents two positive circles, called A and B in 2-D. Suppose that both circles A and B are superimposed by the same hypergrid $V$ of unit size $h$ equal to 1.00. This situation is depicted in the right side of Figure 14. By using Equation (6), the right side of Figure 14 shows that all bins in circle A are of the same density, equal to $\frac{1}{16 \times 1^2} = 0.0625$. In contrast, the density of some of the bins in circle B is equal to $\frac{0}{16 \times 1^2} = 0$. Thus, circle A is a homogeneous region, while circle B is not.

Instead of the strict condition which requires the same density at the centers of the bins, we may apply a softer condition. That is, if the standard deviation of the densities at the centers of the bins is approximately less or equal to $\gamma$, say when $\gamma = 0.01$, then hypersphere $P$ may be considered to be a homogeneous region.

**Input**: Hypersphere $P$ of size $n$ and the density threshold value $\gamma$.

1. Compute the distances between all pairs of points in $P$.
2. Let $h$ be the distance described in Heuristic Rule 1 in Section 4.2.
3. Superimpose $P$ into hypergrid $V$ of unit size $h$.
4. Approximate the density at the center $x$ of each bin.
5. Compute the standard deviation of the densities at the centers of the bins.
6. If the standard deviation is less than or equal to $\gamma$, then

   $P$ is a homogeneous region and its homogeneity degree $HD(P)$ is computed by using Equation (7).
7. Else, $P$ is not a homogeneous region.

**Output**: The decision whether or not hypersphere $P$ is homogeneous.

**Figure 13:** The algorithm for Sub-Problem 2.

As mentioned earlier in Section 4.1, $HD(P)$ is a factor which may affect the total misclassification cost of the inferred classification models. If an unclassified point is covered by a homogeneous region $P$ which has a higher homogeneity degree, then that point may more accurately be assumed to be of the same class as the points covered by the homogeneous region $P$. Thus, a suitable definition for $HD(P)$ is an important step in improving the accuracy of the derived classification models.

The concept of the homogeneity degree $HD(P)$ can be defined as the number of points inside the homogeneous region $P$ per unit of $P$'s volume. This definition, however, has its weaknesses. For instance, let us look at circles A and E as depicted in Figure 11(d). According to the above

definition, *HD(*A) is equal to $\frac{16}{2 \times 1.5^2 \times \pi} \approx 1.1318$ , while *HD(*E) is equal to $\frac{4}{2 \times 0.5^2 \times \pi} \approx 2.5465$ . This means that region E is denser than region A. This is an apparent contradiction since in reality region A has more points and covers a wider region than region E. Thus, we need to find an appropriate definition for the concept of the homogeneity degree.

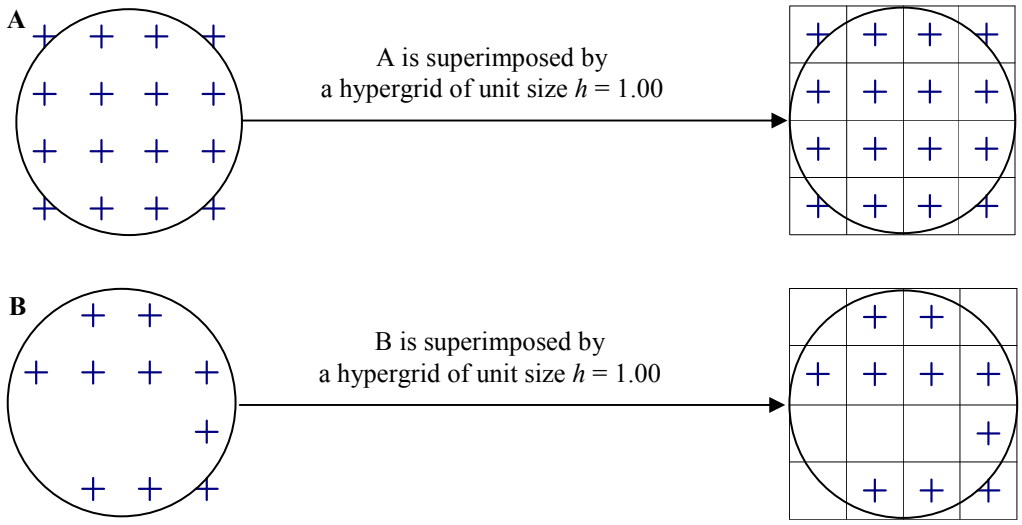



**Figure 14:** Examples of the homogeneous region (at the top part) and the non-homogeneous region (at the bottom part).

Intuitively, *HD*(*P*) depends on the value *h* defined in Heuristic Rule 1 and *P*'s size (*n*). If *n* increases, then *HD*(*P*) would slightly increase since the volume of *P* does not change and *P* has more points. Furthermore, if *h* increases, then the average distance between pairs of points in homogeneous region *P* increases. Obviously, this leads to *HD*(*P*) decreases. Hence, *HD*(*P*) is inversely proportional to *h*, while *HD*(*P*) is directly proportional to *n*. We use the function ln(*n*) to show the slight effect of *n* to *HD*(*P*):

$$HD(P) = \frac{\ln(n)}{h}. \tag{7}$$

For instance, *HD(*A) as depicted in Figure 14 is equal to $\frac{\ln(16)}{1} \approx 2.77$ . Let us consider the example depicted in Figure 11(d). Now *HD(*A) is equal to $\frac{\ln(16)}{1} \approx 2.77$ , *HD(*B) is equal to $\frac{\ln(9)}{1} \approx 2.19$ , *HD(*C*)* = *HD(*E) is equal to $\frac{\ln(4)}{1} \approx 1.38$ , and *HD(*D) is equal to $\frac{\ln(10)}{2} \approx 1.151$. Intuitively, the *HD*s for the circles illustrated in Figure 11(d) make better sense than those derived from the concept of the volume definition.

### 4.3.3 Sub-Problem 5

Recall that the control of fitting and generalization for classification models may be achieved by expanding or breaking the inferred homogeneous regions by using their homogeneity degrees. Suppose that we are given a positive homogeneous region *F* with its homogeneity degree *HD(F)*, the breaking threshold value $\beta^+$, and the expansion threshold value $\alpha^+$. A similar approach can be developed for a negative homogeneous region. According to the main algorithm depicted in Figure 10, if *HD(F)* is greater than or equal to $\beta^+$, then *F* will be expanded by using the expansion threshold value $\alpha^+$. Otherwise, we will break *F* into smaller hyperspheres.

In order to explain Sub-Problem 5, we consider the example depicted in Figure 11(d). Please recall that the homogeneity degrees of the homogenous regions A, B, C, D, and E are *HD(*A) = 2.77, *HD(*B) = 2.19, *HD*(C) = 1.38, *HD(*D) = 1.15, and *HD(*E) = 1.38, respectively. Suppose that the two breaking threshold values $\beta^+$ and $\beta^-$ are equal to 1.00 and 1.50, respectively. Furthermore, let the two expansion threshold values $\alpha^+$ and $\alpha^-$ be equal to 2.00 (both equal). In decreasing order of the *HD* values, the regions A, B, C, E, and D are expanded or broken as shown in Figure 15. As depicted in this figure, the homogeneous regions A, B, and D are expanded (the expanded regions are indicated by the bold-line outer circles), while C and E are

broken into four smaller circles (the broken regions are indicated again by the bold-line circles). The algorithm for breaking a homogeneous region was described in Section 4.3.1 (i.e., Sub-Problem 1).

Two types of expansion exist for a homogeneous region *F*: radial and linear expansions. In radial expansion, a homogeneous region *F* is enlarged in all directions. In linear expansion, a homogeneous region *F* is extended in a certain direction. For instance, in Figure 15 circles A, B, and D have used the radial expansion approach. The following sections discuss in detail these two expansion types.

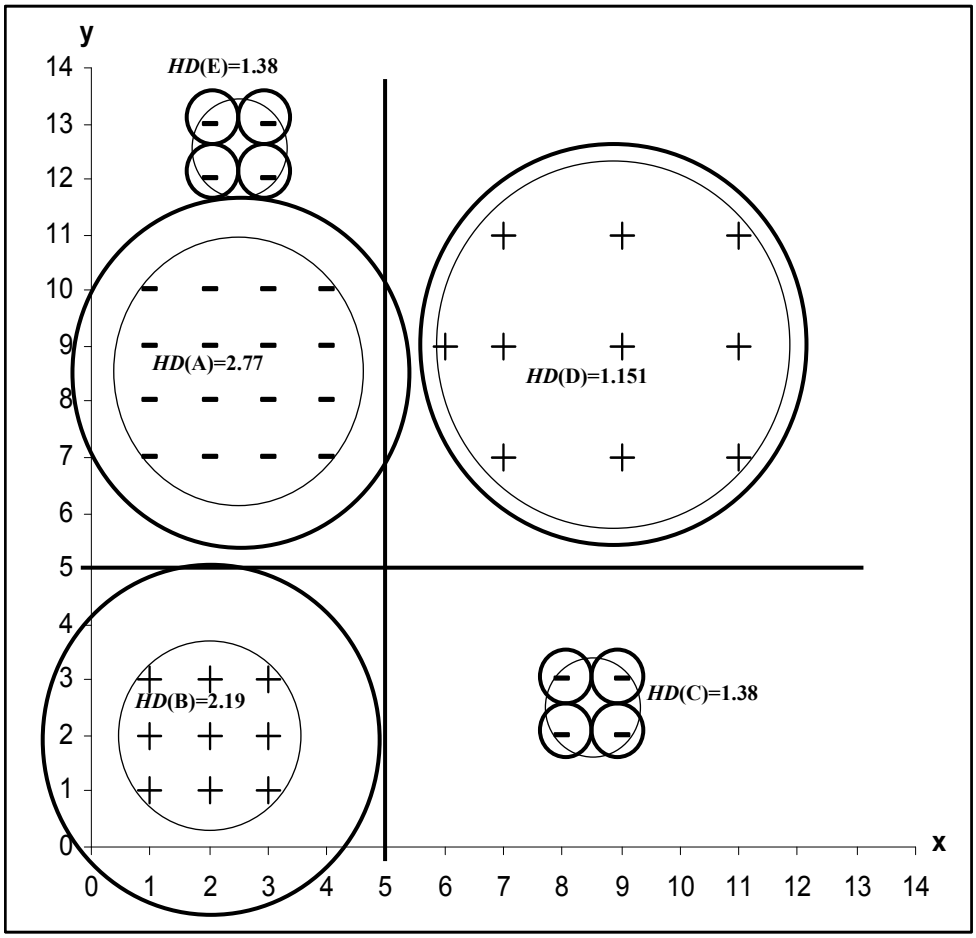



**Figure 15:** An example for Sub-Problem 5.

#### 4.3.3.1 Radial Expansion

In the radial type, a homogeneous region $F$ is expanded in all directions. Let $M$ be a region expanded from $F$. Let $R_F$ and $R_M$ denote the radii of $F$ and $M$, respectively. In the radial expansion approach, $R_F$ is increased by a certain amount denoted as $ST$, called a step-size increment. Thus, one gets:

$$R_M = R_F + ST. \tag{8}$$

By using a dichotomous search methodology, we assume that there exists a hypersphere $G$ which covers the homogeneous region $F$. Furthermore, without loss of generality, let us assume that the radius $R_G$ may be computed by:

$$R_G = 2 \times R_F. \tag{9}$$

By using $R_G$ and $R_F$, we can derive the step-size increment $ST$. That is, $ST$ must depend on the difference between $R_G$ and $R_F$. One of the ways that $ST$ may be determined is as follows:

$$ST = \frac{R_G - R_F}{2}. \tag{10}$$

At the same time, $ST$ should depend on $HD(F)$ because of the dichotomous search methodology. That is, if $HD(F)$ gets higher, then $ST$ should get smaller. This means that $HD(F)$ is inversely proportional to $ST$. If $HD(F)$ is less than one, then an additional parameter $L$ is applied. This parameter ensures that the step-size increment is not too fast. Furthermore, if $HD(F)$ is too small, $F$ is not expanded. Thus, the value for $ST$ may be defined as follows:

$$ST = \frac{R_G - R_F}{2} \times \frac{1}{L \times HD(F)}. \tag{11}$$

If we substitute back Equation (11) into Equation (8), $R_M$ becomes:

$$R_M = R_F + \frac{R_G - R_F}{2} \times \frac{1}{L \times HD(F)}. \tag{12}$$

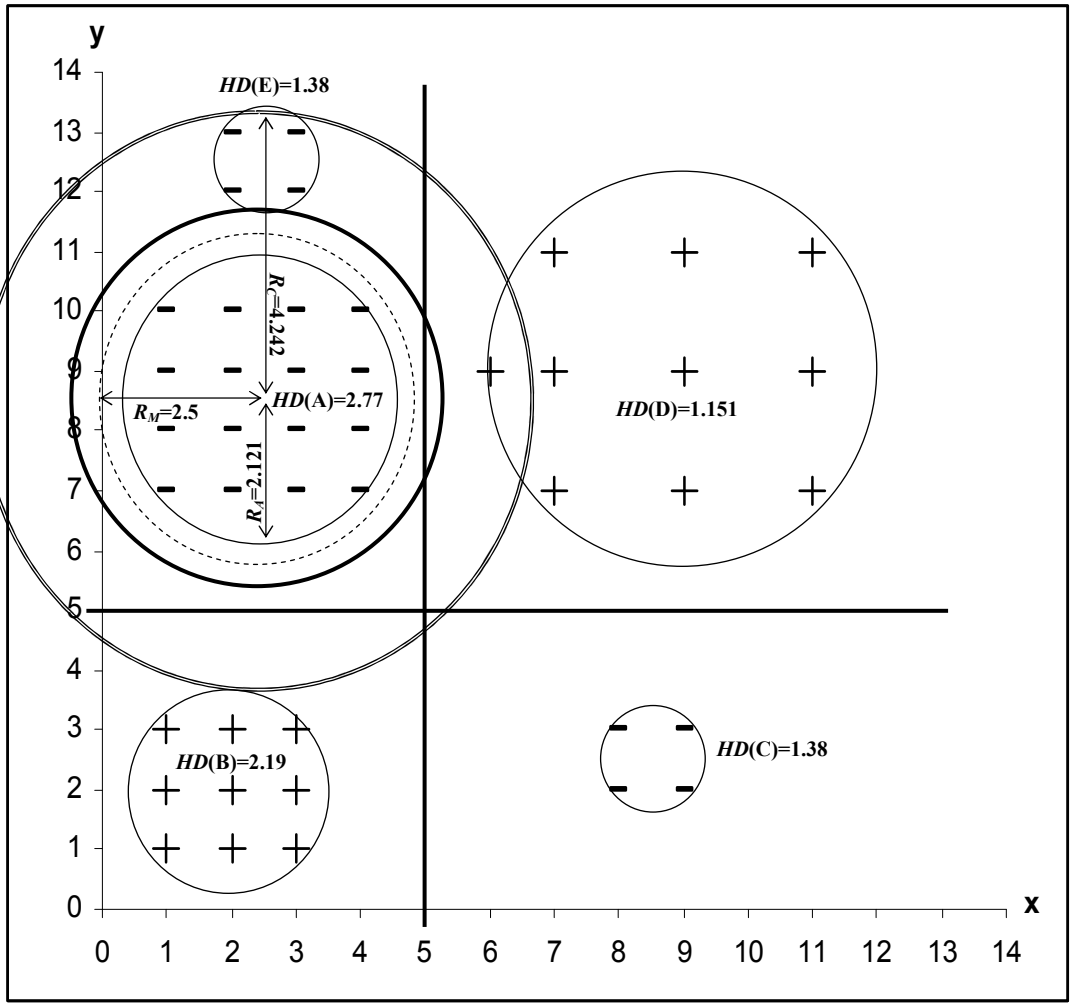



**Figure 16:** An example of radial expansion.

In order to illustrate the radial expansion algorithm depicted in Figure 17, we consider the example indicated in Figure 16. This example uses the same hypothetical data depicted in Figure 15. Assume that the value for $L$ is equal to 1.00. A closer examination of Figure 16 indicates that circle A (i.e., the single-line inner circle with $R_{\mathrm{A}} = 2.121$) is surrounded by the three outer circles: a double-line circle which depicts circle $G$ with $R_G = 2.121 \times 2 = 4.242$, a bold-line circle which shows the final expanded region, and a dotted-line circle which presents the hypersphere $M$ whose radius is computed as follows:

$$R_M = R_F + \frac{R_G - R_M}{2} \times \frac{1}{L \times HD(\mathrm{A})} = 2.121 + \frac{4.242 - 2.121}{2} \times \frac{1}{1 \times 2.77} \approx 2.5.$$

**Input**: Hypersphere $F$ of density $HD(F)$ and radius $R_F$, and the expansion coefficient $\alpha^+$ or $\alpha^-$ (for positive or negative regions, respectively).

1. Set $M = F$ (i.e., $R_F = R_M$). /*initialization*/
2. Set hypersphere $G$ covering $M$ with radius $R_G = 2 \times R_M$.
3. Repeat
4. Set $E = M$ (i.e., $R_E = R_M$).
5. Expand $M$ by using Equation (12).
6. Until ($R_M$ satisfies stopping conditions discussed in Section 4.3.3.3 or $R_M = R_G$).
7. If $R_M$ satisfies stopping conditions, then STOP.
8. Else, go to Step 2.

**Output**: An expanded region $E$.

**Figure 17:** The algorithm for radial expansion.

Similarly, Equation (12) computes the following values for $R_M$ in four iterations: 2.8, 3.06, 3.23, and 3.25, respectively, until $R_M$ satisfies the stopping conditions mentioned later in Section 4.3.3.3. The final expanded region is the bold-line circle depicted in Figure 16. Furthermore, this figure also shows that A's expanded region has a small region (in between A's expanded region and the vertical line at X = 5). This sub-region is now predicted as a positive region, while the DT approach has predicted it as a negative region. This illustration indicates that the HBA could yield more accurate results than the traditional classification approaches.

#### 4.3.3.2 Linear Expansion

The linear approach expands a homogeneous region $F$ in a certain direction. A difference exists between the method presented in the previous section and the one presented in this section (i.e., linear vs. radial). That is, now the homogeneous region $F$ is first expanded to hypersphere $E$ by using the radial expansion. Then, hypersphere $E$ is expanded in a given direction by using the radial approach until it satisfies the stopping conditions mentioned in Section 4.3.3.3. The final region is the union of all the expanded regions.

In order to clarify the linear expansion approach, we consider the homogeneous region A depicted in Figure 18. Suppose that by using radial expansion for the homogeneous region A with the expansion threshold value equal to 2.00, we get the hypersphere U (i.e., the two-line circle depicted in Figure 18(a)). Next, we divide the hypersphere U in the X axis into two parts. The radial expansion approach would expand each one of the parts as the bold-lines depicted in Figure 18(b). A similar approach exists for the Y axis depicted in Figure 18(c). The final expanded region is the region which is defined by the union of the bold-lines depicted in Figure 18(d).

#### 4.3.3.3 The Stopping Conditions

The stopping conditions for expanding a homogeneous region $F$ of size $N_F$ into $M$ depend on $HD(F)$ and the expansion coefficient $\alpha^+$ or $\alpha^-$ (for positive or negative regions, respectively). If $HD(F)$ is high, then a hypersphere $F$ is expanded wider. However, the expanded region should not be too wide as this may lead to the overgeneralization problem. [Pham and Triantaphyllou, 2008a, 2008b, and 2009a] proposed that the radius $R_M$ should not be greater than the product of the following terms: $HD(F)$, $\alpha^+$, and $R_F$. The stopping conditions also include that $M$ should not intersect other regions. This can be determined while expanding $M$. The expanded region $M$ can

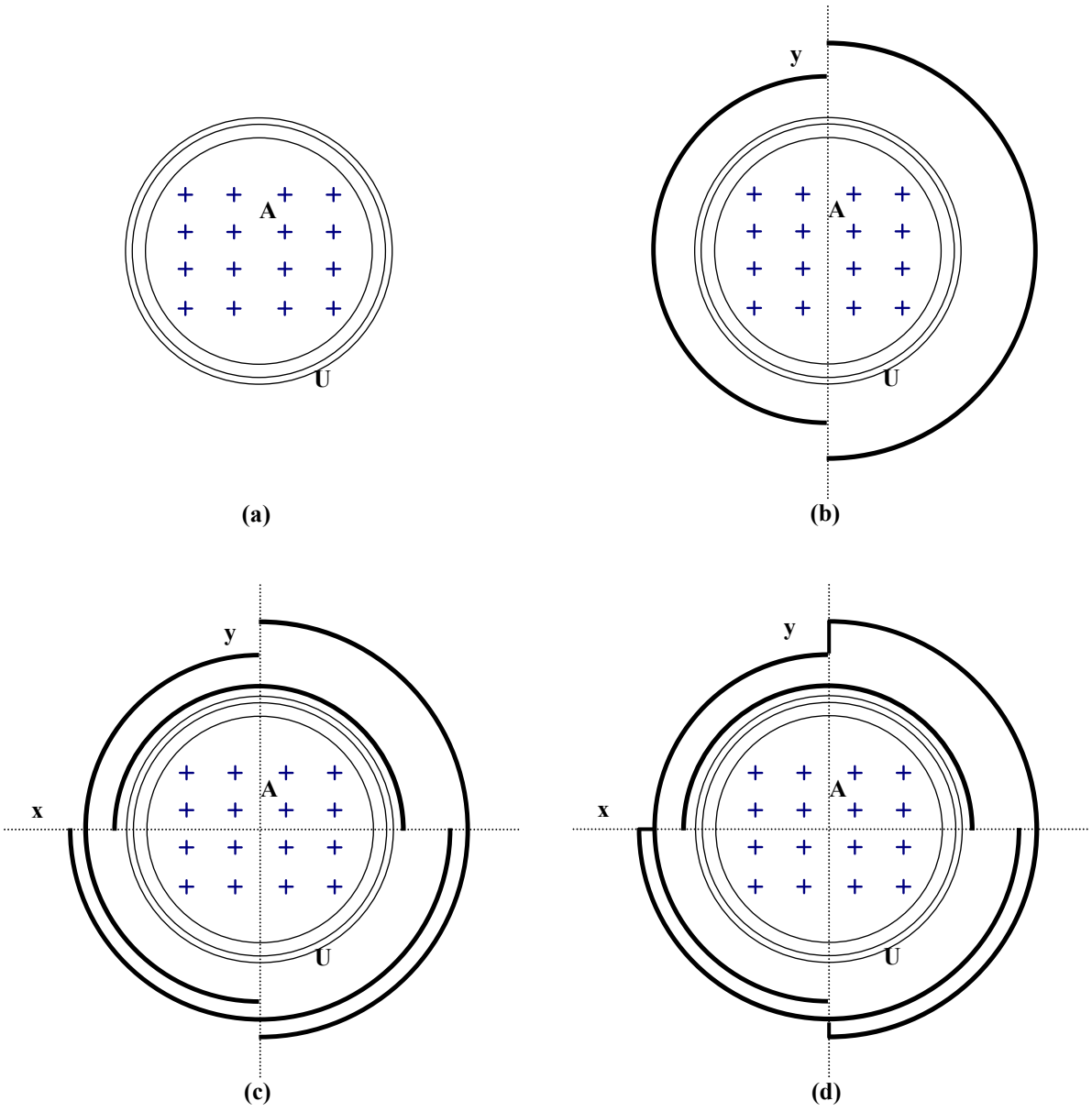



**Figure 18:** An example of linear expansion.

accept several noisy points if $HD(F)$ is high enough. Thus, for positive regions the radial expansion approach is stopped if any one of the following two conditions given as Equation (13) is not satisfied:

$R_M \leq HD(F) \times R_F \times \alpha^+$ and the number of noisy points within $M$ is less than or equal to $HD(F) \times \alpha^+$. (13)

A similar equation exists for negative regions:

$R_M \leq HD(F) \times R_F \times \alpha^-$ and the number of noisy points within $M$ is less than or equal to $HD(F) \times \alpha^-$. (14)

### 4.3.4 A Genetic Algorithm (GA) for Finding the Threshold Values

Recall that the main algorithm depicted in Figure 10 uses the four threshold values ($\alpha^+$, $\alpha^-$, $\beta^+$, $\beta^-$) to derive new classification models. The appropriate values for these parameters control whether or not the HBA provides better classification models. If the breaking threshold values (i.e., $\beta^+$ and $\beta^-$) are too high, then this would result in the overfitting problem. On the other hand, too low breaking threshold values may not be sufficient to overcome the overgeneralization problem. The opposite situation is true with the expansion threshold values (i.e., $\alpha^+$ and $\alpha^-$).

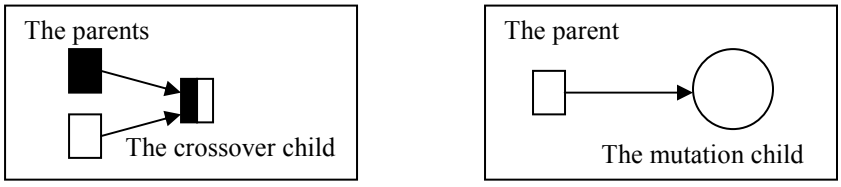
Since the ranges for the threshold values depend on each individual application, the search space may be too large. An exhaustive search would be impractical. Thus, a GA approach is proposed to find approximate optimal threshold values denoted by $(\alpha_*^+, \alpha_*^-, \beta_*^+, \beta_*^-)$. According to [Goldberg, 1989] and [De Jong, 2005], the structure of a general GA approach is as depicted in Figure 19.

A population in the GA approach consists of a set of chromosomes. Each chromosome is a series of genes. At each step, the GA uses the current population to create the children which make up the next generation. The algorithm first selects a group of individual chromosomes, called parents, which have better fitness values (according to the evaluative / fitness function) than the average of the current population. The GA then creates two types of children for the next generation. The first type is formed by using the crossover function. That is, children are created by combining the genes of a pair of parents. The second type is created according to the mutation function. Thus, children are created by introducing random changes, or mutations, to a single parent. Figure 20(a) represents the crossover function, while Figure 20(b) shows the mutation function.

1. Create a random initial population.
2. Create a sequence of new populations. At each step, the algorithm uses the individuals in the current generation to create the next population. To create the new population, the algorithm performs the following steps:
   2.1 Score each chromosome of the current population by computing its fitness value.
   2.1 Select chromosomes, called parents, based on their fitness.
   2.3 Produce children from the parents. Children are produced either by making random changes to a single parent, which is called mutation, or by combining the chromosomes of a pair of parents, which is called crossover.
   2.4 Replace the current population with the children to form the next generation.
3. The algorithm stops when any one of the stopping criteria is met.

**Figure 19:** The structure of a general GA approach.

In the HBA, the GA approach uses Equation (1) as the fitness function and the set $T_2$ for calibration. Each chromosome in this GA approach consists of four genes corresponding to the four threshold values ($\alpha^+$, $\alpha^-$, $\beta^+$, $\beta^-$) as depicted in Figure 21. The use of the GA approach for the HBA was decided because the objective function (i.e., Equation (1)) may have multiple extreme points as the error rates may vary across the input data space in ways which cannot be expressed by usual mathematical functions.




(a). The crossover function. (b). The mutation function.

**Figure 20:** The two types of the children in a general GA approach.

| $\alpha^+$ | $\alpha^-$ | $\beta^+$ | $\beta^-$ |
|---|---|---|---|

**Figure 21:** An example of a chromosome consisting of the four genes.

The initial population size was 20 (this size was determined empirically). One of the 20 chromosomes was initialized to values (1, 1, 0, 0). When the value of $\alpha^+$ or $\alpha^-$ is equal to 1, this indicates no expansion of a region is required. Similarly, when the value of $\beta^+$ or $\beta^-$ is equal to 0, this indicates no breaking of a region is required. Over the given number of generations, the population evolves toward an optimal chromosome.

The crossover function creates the crossover children by combining pairs of parents in the current population. At each coordinate of the child's chromosome, the crossover function randomly picks the gene at the same coordinate from one of the two parents and then assigns it to the child. In order to explain the crossover function, we consider the two chromosomes A ($\alpha_1^+, \alpha_1^-, \beta_1^+, \beta_1^-$) and B ($\alpha_2^+, \alpha_2^-, \beta_2^+, \beta_2^-$) depicted in Figure 22. The function randomly selects the gene at the same coordinate from one of the parents A and B and then assigns it to child C. Thus, chromosome C could be ($\alpha_1^+, \alpha_2^-, \beta_2^+, \beta_1^-$).

| | | | | |
|---|---|---|---|---|
| Chromosome A | $\alpha_1^+$ | $\alpha_1^-$ | $\beta_1^+$ | $\beta_1^-$ |
| Chromosome B | $\alpha_2^+$ | $\alpha_2^-$ | $\beta_2^+$ | $\beta_2^-$ |
| Chromosome C | $\alpha_1^+$ | $\alpha_2^-$ | $\beta_2^+$ | $\beta_1^-$ |

**Figure 22:** An example of the crossover function.

The mutation function creates a child ($g_1$, $g_2$, $g_3$, $g_4$) by changing the genes of the parent chromosome ($\alpha^+$, $\alpha^-$, $\beta^+$, $\beta^-$). Let us consider the first two genes $\alpha^+$ and $\alpha^-$ which are in the range [*a, b*], while the last two genes $\beta^+$ and $\beta^-$ are in the range [*c, d*]. The mutation function first

randomizes a chromosome ($t_1$, $t_2$, $t_3$, $t_4$) by using the Gaussian distribution (this distribution was selected empirically by trial and error). Next, the genes of the mutation child are created by using Equations (15) and (16). These equations attempt to generate genes which are quite different from the parent genes. Furthermore, these equations try to prevent the mutated genes from assuming values outside the valid ranges of the four key control parameters described above.

$$g_1 = ((\alpha^+ \text{ OR } t_1) \text{ OR } a) \text{ AND } b, \text{ and } g_2 = ((\alpha^- \text{ OR } t_2) \text{ OR } a) \text{ AND } b. \quad (15)$$

$$g_3 = ((\beta^+ \text{ OR } t_3) \text{ OR } c) \text{ AND } d, \text{ and } g_4 = ((\beta^- \text{ OR } t_4) \text{ OR } c) \text{ AND } d. \quad (16)$$

| $g_1$ | $g_2$ | $g_3$ | $g_4$ |
|---|---|---|---|
| (((2 OR 1) OR 0) AND 3) = 3 | (((1 OR 1) OR 0) AND 3) = 1 | (((5 OR 3) OR 0) AND 10) = 2 | (((7 OR 7) OR 0) AND 10) = 2 |

**Figure 23:** An example of the mutation function.

In explaining the use of the mutation function, we consider the parent chromosome ($\alpha^+$, $\alpha^-$, $\beta^+$, $\beta^-$) to be (2, 1, 5, 7). Assume that ($\alpha^+$, $\alpha^-$) are in the range [0, 3], while ($\beta^+$, $\beta^-$) are in the range [0, 10]. Suppose that the chromosome ($t_1$, $t_2$, $t_3$, $t_4$), which is created by using a Gaussian distribution, is (1, 1, 3, 7). Figure 23 represents the mutation child. In this figure, ($g_1$, $g_2$, $g_3$, $g_4$) is equal to (3, 1, 2, 2). Under the above settings, the GA approach runs during successive iterations and stops if there is no successive improvement for the fitness function.

### 4.4 The Limitation of the HBA

A problem with the HBA is that it may require excessive computing time. This is due to the way homogeneous regions are determined. In the HBA, Greig's approach of quadrants [Greig-Smith, 1952] was used to determine whether a hypersphere is homogeneous or not. In particular, we need to determine the centers of the bins to compute the densities. This step strongly depends

on the dimensionality of the training data. This is the reason why the HBA may be impractical for data of high dimensionality.

The next chapter describes a new approach called the convexity-based algorithm (CBA). This new approach bypasses HBA's problem by considering the classification of some strategically selected points outside the training dataset. Now the partitioning procedure does not depend on the dimensionality of the training data. The new method is much faster, but the derived systems may not be as accurate as the ones achieved by the HBA, yet still more accurate than the stand-alone traditional classification approaches. However, the CBA may derive systems in situations where the HBA cannot do it due to excessive computing time.

# CHAPTER 5. THE CONVEXITY-BASED ALGORITHM (CBA)[4]

## 5.1 Fundamental Assumptions in the Development of the CBA

As it was described in the previous chapter, the HBA identifies regions in the space of the training dataset where training points are located in a homogeneous manner. A region of the data is considered as homogenous if it can be divided into sub-regions of the same size whose densities are approximately equal with each other. It is this step of the HBA that makes it computationally expensive. Instead of the concept of homogeneous regions, the new approach (i.e., the CBA described in [Pham and Triantaphyllou, 2010]) uses the concept of convex regions as defined in [Melnik, 2002]. Notice that regions derived from a given classification approach are also called decision regions. A region is convex if and only if, for every pair of points within the region, every point along the line segment between them is also within the region. A concave region is complementary to the concept of a convex region. The CBA also assumes that all attributes in a dataset are numerical.

Melnik's approach represents the training dataset as a graph. Any node of this graph uniquely corresponds to a training data point from the training dataset and vice-versa. Thus, we have positive nodes and negative nodes depending on whether they correspond to positive or negative training points, respectively. Two positive nodes are considered to be connected in this graph if and only if the following conditions are satisfied. First, we consider the line segment in the data space which connects these two nodes. Next, we sample along this line segment according to a predefined step length. The sample points defined in this way are classified according to the

[4]A partial part of this chapter was in Computers & Operations Research, Vol. 38, No. 1, 2010, pp. 174-189.

given classification approach. If all of these points are of the same class value as the two nodes, then we say that these two nodes are connected with each other. Otherwise, they are not connected. A similar graph can be defined for the negative training points.

A clique in this graph corresponds to a set of training points which can be connected with each other in any possible pair. Therefore, according to the way provided in [Melnik, 2002], such a clique corresponds to a convex region, while remaining parts of this graph not forming such cliques are considered as concave regions. The CBA approach uses convex regions to substitute for the homogeneous regions applied in the more time-consuming HBA approach. However, there are a number of differences between the convex regions described above and the homogeneous regions used in the HBA.

The ways in which a convex region and a homogeneous region are determined are different. That is, each homogeneous region is determined by training points located in a homogeneous manner. However, each convex region includes training points not necessarily located in such a manner.

The convex region depends on two other factors that do not apply to the homogeneous region. The first factor is the number of sample points used to examine a connection between two positive (or two negative) nodes. If many points along the line segment between the two nodes are sampled, then the assumption that there is a connection between the two nodes is more accurate. Hence, we can have more confidence in the results when we work with the derived convex regions. This kind of sampling may, however, result in excessive computing time. Conversely, if too few points are sampled, then points belonging to different class values may be overlooked, thereby decreasing the accuracy of the assumption. A heuristic way to determine sample points is discussed in Section 5.2.1. The second factor is the accuracy of the original model obtained from a given classification approach. If the original model is inaccurate, then the

sample points used to examine a connection between two nodes are classified less accurately, thereby decreasing the validity of the assumption. This could make the derived convex regions inaccurate in some cases.

The above observations indicate that the CBA may not always be superior to the HBA when it comes to improving accuracy. Thus, classification models derived from the CBA may not be as accurate as those of the HBA. However, because the convex regions do not depend on the dimensionality of the training points, the CBA is much faster than the HBA.

Similar to the concept of the homogeneity degree used in the HBA, each convex region in the CBA is associated with a density measure, called a convex-density (*CD*) [Pham and Triantaphyllou, 2010]. This value measures the density of the training points within a given convex region. An appropriate definition of the *CD* value is discussed in Section 5.2.2.

The key assumption for the CBA is summarized as follows. If an unclassified point is covered by a convex region which has a high *CD* value, then that point may be more accurately assumed to be of the same class value as the nodes (or training points) within the convex region. In other words, the performance of a given classification approach may be enhanced when regions derived from the original approach are convex regions which have high *CD* values. This could be done if such regions are used to find an optimal balance between fitting and generalization.

### 5.2 The CBA

Similar to the input of the HBA, we assume that two classification models denoted as $M_1$ (one for the positive data and the other for the negative data) and a training dataset $T$ are given in the CBA. The desired goal of the CBA is to enhance models $M_1$ in order to obtain an optimal *TC* as described in Equation (1). The key difference between the HBA and CBA is the CBA's usage

of convex regions instead of homogeneous regions. The CBA also uses the same five parameters as those of the HBA:

- Two expansion coefficients $\alpha^+$ and $\alpha^-$ to be used when expanding positive and negative convex regions, respectively.
- A breaking threshold value $\beta^+$ to be used when determining whether a positive convex region should be broken. A similar concept is applied on the breaking threshold value $\beta^-$ for a negative convex region.
- A density threshold value $\gamma$ to be used when determining whether a region is convex.

The main steps of the CBA are depicted in Figure 24 and are summarized in terms of the following phases:

- **Phase 1** (Steps 1 and 2): As with Phase 1 in the HBA, the first phase of the CBA begins with the normalization of the values of the attributes in $T$ because the CBA uses the Euclidean distance in its implementation. This distance requires that $T$ 's attributes should be defined in the same unit (i.e., as percentages). This phase continues to follow by dividing $T$ into two random sub-datasets: $T_1$ whose size is, say, 90% of $T$ 's size and $T_2$ whose size, in this case, is the remaining 10% of $T$ 's size. These percentages can be determined empirically with trial and error. The phase ends with the creation of the four random parameters ($\alpha^+$, $\alpha^-$, $\beta^+$, $\beta^-$).
- **Phase 2** (Step 3): Define the graphs and represent them as two connectivity matrices for the positive and the negative points in $T_1$ by using the method described in Section 5.1. Next, we use clustering to identify concave and convex regions from the two graphs. These regions are represented by smaller connectivity matrices extracted from the above matrices.
- **Phase 3** (Steps 4 to 7): Break the concave regions obtained in Phase 2 into convex regions. The breaking task is used because concave regions do not describe the interrelationship of training points as do convex regions [Melnik, 2002]. A heuristic algorithm for this task is

shown in Section 5.2.3. Step 5 starts with the computation of the *CD* values of the convex regions. Some convex regions are broken again if their *CD* values are less than either $\beta^+$ (for positive regions) or $\beta^-$ (for negative regions). For example, we may have a convex region whose nodes are separated into two groups which are far away from each other. Under this consideration, the *CD* value of the convex region might be too small. Thus, it should be broken into smaller convex regions. Step 7 covers the convex regions by hyperspheres by using the algorithm discussed in Section 5.2.4. The covering task is based on the fact that each hypersphere is determined by a center and a radius. Hyperspheres do not depend on the dimensionality of the training dataset.

- **Phase 4** (Step 8): Expand the hyperspheres obtained in Phase 3 in decreasing order of their *CD* values. The CBA uses the same expansion algorithms as those of the HBA. These expansion algorithms, as described in Section 4.3.3, contain a four-step process which involves the expansion of the radius resulting in the creation of a new hypersphere. Please note that instead of the homogeneity-degrees the expansion algorithms in the CBA use the convexity-densities.
- **Phase 5** (Step 9): Apply the genetic algorithm (GA) to Phases 3 and 4 with Equation (1) as the fitness function and $T_2$ as the calibration dataset. The CBA uses the same GA approach as that of the HBA described in Section 4.3.4. The GA approach seeks the four optimal parameters ( $\alpha_*^+$ , $\alpha_*^-$ , $\beta_*^+$ , $\beta_*^-$ ) , for the three given cost coefficients $C_{FP}$, $C_{FN}$, and $C_{UC}$.
- **Phase 6** (Step 10): The final phase of the CBA mirrors the HBA by repeating Phases 2 to 4 with the optimal parameters ( $\alpha_*^+$ , $\alpha_*^-$ , $\beta_*^+$ , $\beta_*^-$ ) on the entire training dataset *T* in order to infer the final pair of models $M_2$.

The following sections provide the details of the above phases.

**Input**: The positive and negative classification models $M_1$, the training dataset $T$, the density threshold value $\gamma$, and the three cost coefficients $C_{FP}$, $C_{FN}$, and $C_{UC}$.

1. Normalize $T$ and divide $T$ into two random sub-datasets: the actual training dataset $T_1$ and the calibration dataset $T_2$.
2. Randomly initialize the values of the four parameters ($\alpha^+$, $\alpha^-$, $\beta^+$, $\beta^-$).
3. Define the positive and negative graphs for $T_1$ and use clustering to identify concave and convex regions from these graphs.
4. Break concave regions (if any) into convex regions.
5. Compute the *CD* values of the convex regions.
6. Break the convex regions into sub-convex regions, if their *CD* values are less than $\beta^+$ or $\beta^-$ for the positive and negative data, respectively.
7. Cover the convex regions obtained in Steps 4 and 6 with hyperspheres.
8. In decreasing order of the *CD* values, expand hypersphere $P$ by using *CD(P)* and $\alpha^+$ or $\alpha^-$ for the positive and negative data, respectively.
9. Apply the GA approach to Steps 6 to 8 with Equation (1) as the fitness function and use the calibration dataset $T_2$. The GA approach finds the optimal values of the four parameters denoted as ($\alpha_*^+$, $\alpha_*^-$, $\beta_*^+$, $\beta_*^-$).
10. Use the optimal values ($\alpha_*^+$, $\alpha_*^-$, $\beta_*^+$, $\beta_*^-$) on the entire training dataset $T$ to repeat Steps 3 to 8 and infer the final pair of models $M_2$.

**Output**: The positive and negative classification pair of models $M_2$.

**Figure 24:** The CBA [Pham and Triantaphyllou, 2010].

### 5.2.1 A Heuristic for Determining Sample Points

As described in Section 5.1, whether two positive (or two negative) nodes (training points) are connected depends on the classifications of the sample points between the two nodes. Assume that we are given a training dataset $T_1$ and the positive and negative models $M_1$. This section shows a heuristic way to determine these sample points. Assume that a value for $d$ is defined as in Equation (17):

$$d = \min\{\text{the Euclidean distance of all pairs of training points in } T_1\}. \quad (17)$$

Two positive (or two negative) nodes are assumed to be connected if and only if the following test is satisfied. We divide the line segment between these two nodes into smaller segments of size $d$. Sample points derived from the above segmentation are classified by using $M_1$. If all of these points are of the same class value as the two end nodes, then it is assumed that these two nodes are connected with each other. A distance equal to $d$, defined as above, worked well on our tests when used to sample the line segment between a pair of end nodes. If the sampling distance (step) is too large, then we have under-sampled. Conversely, if the distance is too small, then we have over-sampled.

### 5.2.2 Determining Whether a Region Is Convex

By using a sampling step of size $d$ as defined in Section 5.2.1, the CBA derives the positive and negative graphs. Next, the CBA uses clustering to break a graph into regions. This section describes an algorithm which is used to determine whether a region is convex and is depicted in Figure 25. As discussed in Section 5.1, a convex region corresponds to a clique whose connectivity matrix includes all ‘1’s. The connectivity matrix is symmetric and all elements of the first diagonal are equal to 0 (‘0’ is used here in accordance with Matlab convention). Thus, whether a region $P$ of size $n$ is convex is determined by checking whether $P$’s connectivity

matrix includes all '1's (with the exception of the elements of the first diagonal). A softer condition for *P*'s connectivity matrix can be applied. That is, if the percentage of the number of '1's in *P*'s connectivity matrix is greater than or equal to $\gamma$, say for $\gamma$ equal to 0.9, then *P* is considered to be convex. The value of 0.90 was used for the parameter γ as result of some pilot tests. If a value closer to 1.0 is used, then more and smaller regions will qualify to be designated as convex. This would result in more computing time for the rest of the steps of the CBA and also the derived models may be affected by overfitting of the training data. The reverse would happen if the value of $\gamma$ is a rather small number.

**Input**: Region *P* of size *n* and the density threshold value $\gamma$.

1. Consider *P*'s connectivity matrix denoted as *W*.

2. Set *num* = the number of '1's in *W*.

3. If $\frac{num}{n \times (n-1)} \geq \gamma$, then *P* is convex and *CD*(*P*) is computed by Equation (19).

**Output**: Whether *P* is convex.

**Figure 25:** An algorithm for determining whether a region is convex.

As seen in Step 3 in Figure 25, each convex region is associated with a *CD* value. Tichy in [Tichy, 1973] proposed *CD*(*P*) to be the proportion of *P*'s training points over the total number of points in $T_1$ for a given class value. This definition, however, has its drawbacks. For instance, Figure 26 presents some positive and negative training points in 2-D of a given dataset $T_1$. Suppose that $M_1$ is applied on $T_1$ and the convex regions are derived. The derived convex regions

are assumed to be the circles A, B, C, D, and E as depicted in Figure 26. Such regions do not always have to be circles but this is assumed here for simplicity. According to Tichy's definition, *CD*(B) and *CD*(D) have the same value equal to $\frac{9}{18} = 0.5$. This is apparently a contradiction since circle B is denser than D.

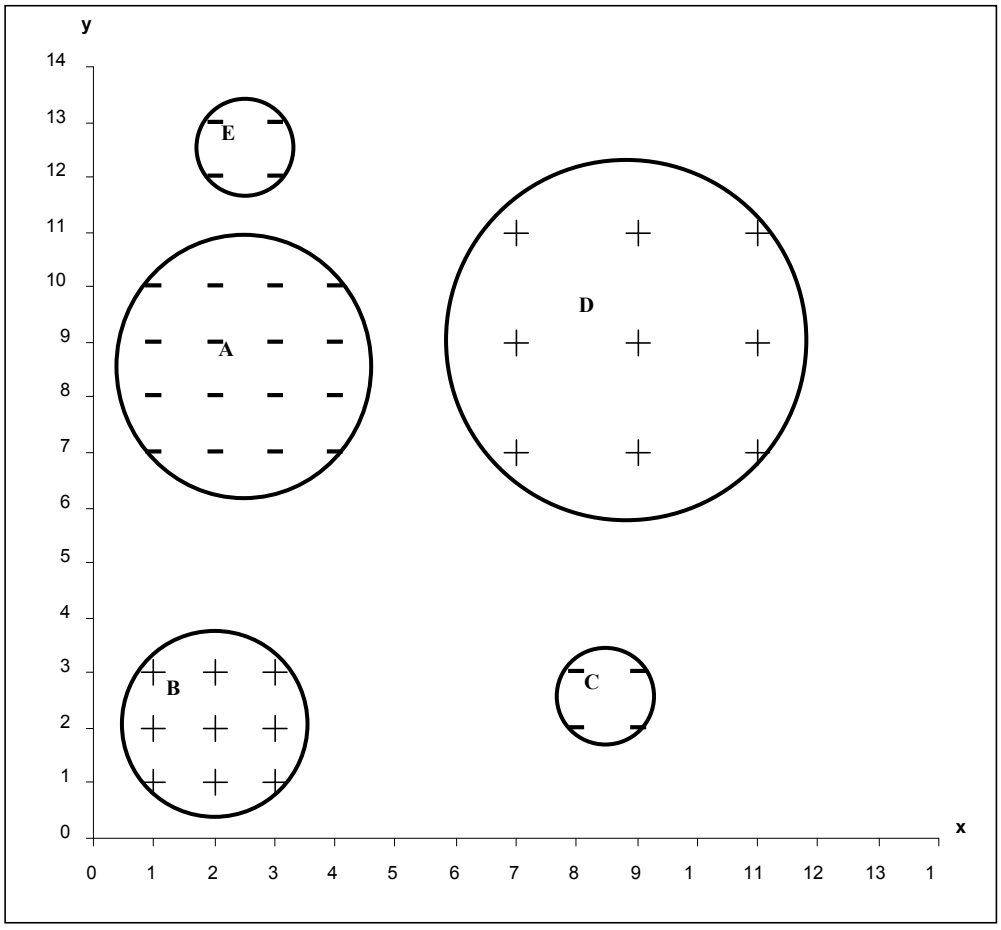



**Figure 26:** An example of convex regions in $T_1$.

The value for *CD*(*P*) is directly related to *n* and to the average minimum distance to neighbors denoted as *h* which is defined as follows:

$$h = \sum_{i}^{n} \frac{\min(d_i)}{n}. \tag{18}$$

The notation $\min(d_i)$ in Equation (18) is the Euclidean distance between each node and its nearest neighbor. If $P$ has a higher $n$ value and a smaller $h$ value, then $P$ is denser. The authors in [Pham and Triantaphyllou, 2010] propose $CD(P)$ to be calculated as in Equation (19) where $D$ is the number of dimensions of the training points in $T_1$:

$$CD(P) = \frac{\ln(n)}{h^D}. \tag{19}$$

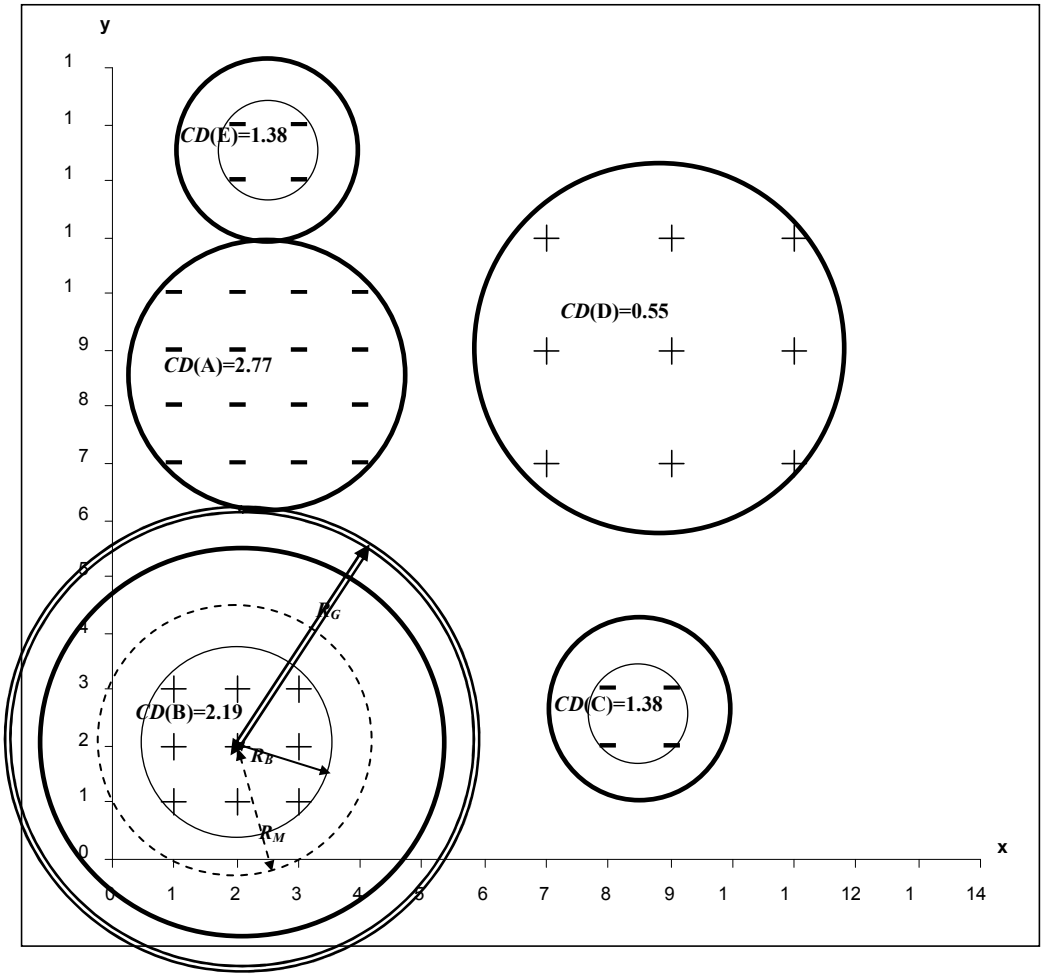



**Figure 27:** An example for the radial expansion algorithm.

Equation (19) shows that if $n$ increases, then $CD(P)$ slightly increases since $P$ has more nodes. Furthermore, if $h$ decreases, then $P$'s nodes are closer together. This leads to increasing

the value of $CD(P)$. Hence, $CD(P)$ is inversely proportional to $h$, while $CD(P)$ is directly proportional to $\ln(n)$. The function $\ln(n)$ achieves a slighter effect of $n$ on $CD(P)$. For instance, the $CD$(A), $CD$(B), $CD$(C), $CD$(D), and $CD$(E) values for Figure 26 now are equal to $\frac{\ln(16)}{1^2} \approx 2.77$, $\frac{\ln(9)}{1^2} \approx 2.19$, $\frac{\ln(4)}{1^2} \approx 1.38$, $\frac{\ln(9)}{2^2} \approx 0.55$, and $\frac{\ln(4)}{1^2} \approx 1.38$, respectively. Intuitively, these density values make better sense than the ones derived from Tichy's definition.

By using the radial expansion algorithm described in Section 4.3.3.1, let us consider the example indicated in Figure 27 which is based on the one shown in Figure 26. Assume that the value for threshold $L$ in Equation (12) is 1.00. Let the two breaking threshold values $\beta^+$ and $\beta^-$ be 0.5 (both equal). No convex regions are broken because their $CD$ values are greater than the breaking threshold values. Let the two expansion coefficients $\alpha^+$ and $\alpha^-$ be 2.00 (both equal). In decreasing order of the $CD$ values, the regions A, B, C, E, and D are expanded as shown in Figure 27. The expanded regions are covered by the bold-line circles (or hyperspheres) as discussed in Section 5.2.4. Regions A and D are portrayed as they are initially. This is due to the fact that at the first expansion step region A intersects region E. Similarly, region D intersects region A.

A closer examination of Figure 27 reveals that circle B (i.e., the single-line inner circle with $R_B = 1.8019$) is surrounded by three outer circles: a double-line circle $G$ with $R_G = 1.8019 \times 2 = 3.6038$, a bold-line circle which shows the final expanded region, and a dashed-line circle $M$ of radius $R_M$ given as follows:

$$R_M = R_B + \frac{R_G - R_M}{2} \times \frac{1}{L \times CD(B)} = 1.8019 + \frac{3.6038 - 1.8019}{2} \times \frac{1}{1 \times 2.19} \approx 2.2133 .$$

In particular, region B is expanded by computing the following successive values for $R_M$: 2.5308, 2.7758, 2.9648, and so on, until $R_M$ is equal to 3.6036. At that moment, the iterations are terminated because of the stopping conditions described in Section 4.3.3.3. Please note that these

conditions specialize on homogeneous regions with the concept of the homogeneity-degrees. However, the stopping conditions of the CBA employ the convex regions with the concept of the convex-densities.

### 5.2.3 A Heuristic Algorithm for Breaking a Region into Convex Regions

The problem of finding the minimum number of convex regions that cover a region *P* of size *n* is similar to a form of the set cover problem, an NP-complete problem [Karp, 1972]. Melnik in [Melnik, 2002] proposed the Hamming distance for breaking *P* into smaller convex regions. The Hamming distance between two strings is defined by the number of positions for which the corresponding attributes are different. If the Hamming distance between a pair of rows in *P*'s connectivity matrix is small enough, then Melnik groups the nodes together. However, this method does not take into account the distances between *P*'s nodes as constraints for grouping them. Thus, Melnik's approach may derive convex regions that overlap each other.

A heuristic approach for breaking a region into a set of convex regions is proposed in Figure 28. This approach is based on Hierarchical Clustering (HC) as described in [Seo, et. al., 2002], [Karypis, et. al., 1999], and [Moore, 2010]. HC is a deterministic approach and produces a hierarchy of clusters represented by a dendrogram. This dendrogram may be used to partition a dataset (of size *n*) into *K* desired clusters, where $K = 1, 2, \ldots,$ or *n* (see also [Moore, 2010]).

The algorithm depicted in Figure 28 may end up with *n* clusters (in the degenerative case), each defined on a single data point only. This would happen if no partition of the dataset with convex regions was produced or the derived convex regions had *CD* values that were too small because the current breaking threshold values were inadequate (i.e., they were too large). If this is the case, then the GA approach described in Section 4.6 will run again as the fitting function given as equation (1) would lead to a gross overfitting of the data. Successive runs of the GA

approach should converge to an optimal (or near optimal) set of the threshold values and thus the final partition would represent a balance between overfitting and overgeneralization. Or in other words, the misclassification total cost would be minimum or very small.

**Input:** A concave or convex region $P$ of size $n$ (in the form of a graph) and the breaking threshold values $\beta^+$ or $\beta^-$ (for positive or negative data, respectively).

1. Use the Hierarchical Clustering (HC) approach to produce a dendrogram with a hierarchy of all possible clusters (the maximum number of which is equal to $n$).
2. For $K = 1$ to $n$ do
3. Begin {for}
4. Use the dendrogram produced by the HC approach to partition the region $P$ into $K$ clusters.
5. If all of these regions are convex, then
6. If their $CD$ values are greater than or equal to the breaking threshold value ($\beta^+$ or $\beta^-$), then these regions are returned and we stop the iterations.
7. End {for}

**Output**: A set of convex regions.

**Figure 28:** The heuristic algorithm for breaking a region.

### 5.2.4 A Method for Covering a Convex Region by a Hypersphere

A hypersphere can cover a convex region by using the linear algorithm developed by Ritter in [Ritter, 1990] and depicted in Figure 29. Let us consider a convex region $P$ of size $n$. A hypersphere which covers $P$ is determined by a center $C_P$ and a radius $R_P$. The algorithm is

initialized by an educated guess for creating a hypersphere $B_1$ for $P$. This is done by finding two nodes of $P$ which are far from each other and using the line between them as the initial diameter. Let $C_P$ be located at the center of the diameter and $R_P$ be half the length of the diameter.

Next, each node $S_{i+1}$ of $P$ is tested for inclusion in the current hypersphere. This is done by checking whether its distance from $C_P$ is less than or equal to $R_P$. If $S_{i+1}$ is in $B_i$, then $B_{i+1} = B_i$, and the algorithm goes to the next node. Otherwise, $B_i$ is expanded just enough to include itself as well as $S_{i+1}$. This is done by drawing a line as depicted in Figure 30 from $S_{i+1}$ to the current center $C_P$ of $B_i$ and extending it further to intersect the far side of $B_i$. This segment is then used as the new diameter for an expanded hypersphere $B_{i+1}$ and $C_P$ is relocated at the center of the new diameter.

**Input**: Convex decision region $P$ of size $n$.

1. Initialize a diameter.
2. Set hypersphere $B_1$ with $C_P$ as the center of the diameter and $R_P$ as half the length of the diameter.
3. For each point $S_{i+1}$ of $P$ do
4. Begin {for}
5. If the distance from $C_P$ to $S_{i+1}$ is less than or equal to $R_P$, then $B_{i+1} = B_i$.
6. Else
   $B_i$ is expanded just enough to include itself as well as the point $S_{i+1}$.
7. End {for}

**Output**: $B_n$.

**Figure 29:** The algorithm for covering a convex region by a hypersphere.

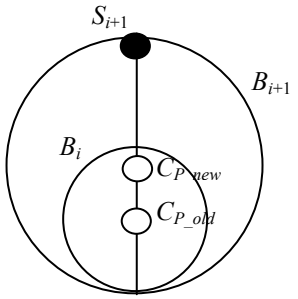



**Figure 30:** An example for expanding a hypersphere of a convex region.

### 5.3 Complexity Analysis

Next, we analyze the time complexity of the CBA depicted in Figure 24 for the general case. Let $n$ be the number of the training points in $T$. Step 1 normalizes $T$ and then divides $T$ into $T_1$ and $T_2$. This takes linear time on the size of $T$ or O($n$). Step 2 takes O(1). Step 3 defines the positive and negative graphs for $T_1$ and takes O($n^3$) time [Melnik, 2002].

Steps 4 and 6 operate in the same manner. These steps break either a concave or a convex region into smaller convex regions. The breaking algorithm depicted in Figure 28 is related to the algorithm depicted in Figure 25. Each iteration of the breaking algorithm includes two steps: breaking a region into smaller regions by using the hierarchical clustering approach and determining whether a region is convex. The time complexity of the hierarchical clustering approach is O($n^2$) [Seo, et. al., 2002] and [Karypis, et. al., 1999]. The time complexity for determining whether a region is convex is O($n^2$). Thus, the time complexity of Steps 4 and 6 is O($n^3$). The time complexity for Step 5 includes the time complexity of the breaking algorithm.

Each convex region is covered by a hypersphere by using Ritter's linear algorithm. Thus, Step 7 takes O($n^2$) time. Step 8 expands convex regions by using the algorithm depicted in Figure 17. There are two loops on Lines 3 and 4 of the algorithm. The loop on Line 3 is inside the loop on Line 4. The quadratic time applies to the loop on Line 3. Thus, the total time complexity of Step 8 is O($n^3$). Step 9 applies the GA approach on Steps 6 to 8. We assume that

the GA approach is executed for 100 generations. Thus, the time complexity of the GA approach is $O(100 \times n^3) = O(n^3)$ [Goldberg, 1989]. Step 10 repeats Steps 3 to 8 with the four optimal parameters $(\alpha_*^+, \alpha_*^-, \beta_*^+, \beta_*^-)$ on $T$. This step makes the total time complexity of Steps 3 to 8 equal to $O(n^3) + O(n^2) + O(n^2) = O(n^3)$. Therefore, the overall time complexity of the CBA is equal to $O(n^3)$. The following chapter presents some results obtained from the HBA and CBA. It also provides a discussion on those results.

# CHAPTER 6. RESULTS AND DISCUSSION[5]

## 6.1 Datasets and Environment for the Experiments

The original classification approaches used in the experiments were SVMs, ANNs, and DTs. For a given 3-tuple of error penalty costs ($C_{FP}$, $C_{FN}$, $C_{UC}$), each original algorithm was run alone and also in conjunction with the HBA and with the CBA. The following two methods were used to analyze the performance of the HBA and CBA:

- The **four-fold cross-validation** method in which each dataset was divided into four parts where one part corresponded to the testing dataset and the three remaining parts formed the training dataset *T*. The performance of an approach was the average of the performances of the four folds. The authors in [Pham and Triantaphyllou, 2010] employed this method to analyze the performance of the HBA and CBA.
- The **test-set** method in which we randomly chose 25% of the dataset to be in the testing dataset and the remaining 75% to be the training dataset *T*. The authors in [Pham and Triantaphyllou, 2008a, 2008b, and 2009a] applied this method to analyze the performance of the HBA.

Under the above methods, the *TC*s obtained from the original algorithms and those found in conjunction with the HBA and with the CBA were compared when applied on the same dataset. The experiments used the twelve datasets found in [Asuncion, et. al., 2009], [Tin, et. al., 1996], and [Weiss, et. al., 1989]. The results of some previous studies on these datasets are shown in Tables 1(a) and 1(b). These datasets were selected because current classification approaches

[5] A partial part of this chapter was in Computers & Operations Research, Vol. 38, No. 1, 2010, pp. 174-189 and in "Studies in Computation Intelligence," (Roger Yin Lee, Editor), Chapter 2, Vol. 131, Springer, Berlin, Germany, 2008, pp. 11-26..

have analyzed them with varying success and also because they are considered benchmark datasets.

**Table 1(a):** The datasets used in the experiments.

| Dataset | No. of Points | No. of Attributes | Previous studies | Accuracy(%) |
|---|---|---|---|---|
| Pima Indians Diabetes (PID) [Asuncion, et. al., 2009] | 768 | 8 | Early Neural Networks in [Smith, et. al., 1988] | 76.0 |
| | | | IncNet in [Jankowski, et. al., 1997] | 77.6 |
| | | | Fuzzy approach in [Au, et. al., 2001] | 77.6 |
| | | | Flexible Neural-Fuzzy Inference System in [Rutkowski, et. al., 2003] | 78.6 |
| | | | Fuzzy Neural Networks in [Leon, 2006] | 81.8 |
| | | | Statlog Project in [Michie, et. al., 1994] | 78.0 |
| Haberman Surgery Survival (HSS) [Asuncion, et. al., 2009] | 306 | 3 | SVMs using linear terms in the objective function in [Kecman, et. al., 2001] | 71.2 |
| | | | Proximal SVMs in [Fung, et. al., 2001] | 72.5 |
| | | | Integer SVMs in [Domm, et. al., 2005] | 62.7 |
| | | | Logical functions in [Shevked, et. al., 2007] | 66.2 |
| Wisconsin Breast Cancer (WBC) [Asuncion, et. al., 2009] | 286 | 10 | C4.5 in [Quinlan, 1996] | 94.7 |
| | | | Rule Induction approach in [Hamilton, et. al., 1996] | 96.0 |
| | | | Linear Discriminant Analysis approach in [Ster, et. al., 1996] | 96.8 |
| | | | SVMs in [Bennet, et. al., 1997] | 97.2 |
| | | | Neuro-Fuzzy approach in [Nauck, et. al., 1999] | 95.1 |
| | | | Fuzzy-GA approach in [Pena-Reyes, et. al., 1999] | 97.5 |
| | | | Neuro-Rule approach in [Setiono, 2000] | 98.1 |
| | | | Supervised Fuzzy Clustering in [Abonyi, et. al., 2003] | 95.6 |
| | | | Fuzzy Artificial Immune Recognition System (FAIRS) in [Polat, et. al., 2007] | 98.5 |
| | | | Classification through ELECTRE and Data Mining in [Mastrogiannis, et. al., 2009] | 94.4 |
| Liver-Disorders (LD) [Asuncion et. al., 2009] | 345 | 6 | RULES-4 algorithm [Pham, et. al., 2000] | 55.9 |
| | | | C4.5 in [Cheung, 2001] | 65.5 |
| | | | Reduced SVMs in [Lee, et. al., 2001a and 2001b] | 74.9 |
| | | | SVMs in [Van, et. al., 2002] | 69.2 |
| | | | Least Squares SVMs in [Çomaka, et. al., 2007] | 94.3 |
| | | | FAIRS in [Polat, et. al., 2007] | 83.4 |

**Table 1(b):** Continued.

| Dataset | No. of Points | No. of Attributes | Previous studies | Accuracy(%) |
|---|---|---|---|---|
| Statlog Heart (SH) [Asuncion, et. al., 2009] | 270 | 13 | Different approaches in Statlog Project in [Statlog Heart, 2009] | 76.7 |
| | | | Attribute weighted artificial immune system in [Özşen, et. al., 2009] | 87.4 |
| Australian Credit Approval (ACA) [Asuncion, et. al., 2009] | 690 | 14 | C4.5 in [Friedman, et. al., 1997] | 85.7 |
| | | | Eight genetic programming approaches in [Sakprasat, et. al., 2007] | 83.0 |
| | | | Different approaches in Statlog Project in [Statlog Australia Credit Approval, 2009] | 86.9 |
| | | | Extend Naive Bayes in [Hsu, et. al., 2008] | 76.7 |
| | | | SVMs in [Huang, et. al., 2007] | 85.5 |
| Appendicitis (AP) [Weiss, et. al., 1989] | 106 | 7 | Predictive Value Maximization approach in [Weiss, et. al., 1989] | 89.6 |
| | | | Fuzzy Rule-Based Classification System in [Nakashima, et. al., 2003] | 84.0 |
| | | | Nefclass in [Blachnik, et. al., 2006] | 87.7 |
| FourClass [Tin, et. al., 1996] | 862 | 2 | Fuzzy Kernel Multiple Hyperspheres in [Lei, et. al., 2007] | 99.8 |
| | | | KNN-SVM in [Setiono, 2000] | 100.0 |
| German Credit Data (GCD) [Asuncion, et. al., 2009] | 1,000 | 24 | Graph-based relational concept learner in [Gonzalez, et. al., 2001] | 71.5 |
| | | | DTs in [Eggermont, et. al., 2004] | 72.9 |
| | | | SVMs in [Huang, et. al., 2007] | 77.9 |
| Ionosphere (INS) [Asuncion, et. al., 2009] | 351 | 34 | Features Selection in conjunction with ANNs and KNNs in [Gavrilis, et. al., 2008] | 90.6 |
| | | | Integration between fuzzy class association rules and SVMs [Kianmehr, et. al., 2008] | 89.2 |
| Parkinsons (PA) [Asuncion, et. al., 2009] | 195 | 23 | ANNs in [Ene, 2008] | 81.3 |
| | | | SVMs in [Little, et. al., 2009] | 91.4 |
| SPECTF [Asuncion, et. al., 2009] | 267 | 45 | CLIP3 in [Kurgan, et. al., 2001] | 77.0 |
| | | | Rough set-base multiple criteria linear programming in [Zhanga, et. al., 2009] | 68.0 |

In the experiments, a given dataset $T$ was normalized by using Equation (20) so that all units would be dimensionless. The term $T_{i,j}$ in this equation shows the value of the data point $i$ in terms

**Table 2(a):** $TC = \min(1\times RateFP + 1\times RateFN)$ under the four-fold cross-validation method [Pham and Triantaphyllou, 2010].

| Dataset | Algo. | Original algorithm | | | | Original algorithm in conjunction with the HBA | | | | | Original algorithm in conjunction with the CBA | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | *RateFP* | *RateFN* | *RateUC* | *TC* | *RateFP* | *RateFN* | *RateUC* | *TC* | Improvement 1 | *RateFP* | *RateFN* | *RateUC* | *TC* | Improvement 2 | Improvement 3 |
| PID | SVM | 0.5 | 31.3 | 1.0 | 31.8 | 7.3 | 0.0 | 59.4 | 7.3 | 77.0 | 26.0 | 5.2 | 1.0 | 31.3 | 1.6 | -75.4 |
| | DT | 5.2 | 29.7 | 0.0 | 34.9 | 2.6 | 0.0 | 71.4 | 2.6 | 92.5 | 18.2 | 4.2 | 15.6 | 22.4 | 35.8 | -56.7 |
| | ANN | 3.1 | 22.9 | 11.5 | 26.0 | 3.1 | 0.0 | 71.4 | 3.1 | 88.0 | 8.3 | 7.8 | 22.9 | 16.1 | 38.0 | -50.0 |
| HSS | SVM | 26.0 | 5.2 | 5.2 | 31.2 | 0.0 | 10.4 | 44.2 | 10.4 | 66.7 | 27.3 | 1.3 | 0.0 | 28.6 | 8.3 | -58.3 |
| | DT | 19.5 | 15.6 | 0.0 | 35.1 | 0.0 | 15.6 | 26.0 | 15.6 | 55.6 | 26.0 | 1.3 | 2.6 | 27.3 | 22.2 | -33.3 |
| | ANN | 18.2 | 10.4 | 11.7 | 28.6 | 0.0 | 14.3 | 27.3 | 14.3 | 50.0 | 22.1 | 3.9 | 16.9 | 26.0 | 9.1 | -40.9 |
| WBC | SVM | 0.0 | 27.8 | 8.3 | 27.8 | 11.1 | 1.4 | 47.2 | 12.5 | 55.0 | 0.0 | 15.3 | 11.1 | 15.3 | 45.0 | -10.0 |
| | DT | 6.9 | 18.1 | 16.7 | 25.0 | 8.3 | 1.4 | 48.6 | 9.7 | 61.1 | 5.6 | 13.9 | 22.2 | 19.4 | 22.2 | -38.9 |
| | ANN | 1.4 | 18.1 | 22.2 | 19.4 | 8.3 | 1.4 | 50.0 | 9.7 | 50.0 | 4.2 | 8.3 | 31.9 | 12.5 | 35.7 | -14.3 |
| LD | SVM | 0.0 | 37.2 | 1.2 | 37.2 | 0.0 | 0.0 | 90.7 | 0.0 | 100.0 | 11.6 | 17.4 | 12.8 | 29.1 | 21.9 | -78.1 |
| | DT | 18.6 | 15.1 | 2.3 | 33.7 | 0.0 | 8.1 | 79.1 | 8.1 | 75.9 | 10.5 | 9.3 | 12.8 | 19.8 | 41.4 | -34.5 |
| | ANN | 10.5 | 22.1 | 11.6 | 32.6 | 0.0 | 4.7 | 80.2 | 4.7 | 85.7 | 7.0 | 11.6 | 19.8 | 18.6 | 42.9 | -42.9 |
| AP | SVM | 11.1 | 0.0 | 14.8 | 11.1 | 0.0 | 0.0 | 81.5 | 0.0 | 100.0 | 11.1 | 0.0 | 14.8 | 11.1 | 0.0 | -100.0 |
| | DT | 7.4 | 0.0 | 22.2 | 7.4 | 0.0 | 0.0 | 81.5 | 0.0 | 100.0 | 7.4 | 0.0 | 22.2 | 7.4 | 0.0 | -100.0 |
| | ANN | 7.4 | 0.0 | 25.9 | 7.4 | 0.0 | 0.0 | 81.5 | 0.0 | 100.0 | 7.4 | 0.0 | 25.9 | 7.4 | 0.0 | -100.0 |
| FourClass | SVM | 4.6 | 1.9 | 1.9 | 6.5 | 0.5 | 4.2 | 41.7 | 4.6 | 28.6 | 4.6 | 1.9 | 1.9 | 6.5 | 0.0 | -28.6 |
| | DT | 2.8 | 3.7 | 1.4 | 6.5 | 0.5 | 2.8 | 35.6 | 3.2 | 50.0 | 2.8 | 3.7 | 1.4 | 6.5 | 0.0 | -50.0 |
| | ANN | 0.0 | 3.2 | 2.8 | 3.2 | 0.5 | 2.8 | 31.0 | 3.2 | 0.0 | 0.0 | 3.2 | 2.8 | 3.2 | 0.0 | 0.0 |

**Table 2(b)**: Continued.

| Dataset | Algo. | Original algorithm | | | | Original algorithm in conjunction with the HBA | | | | | Original algorithm in conjunction with the CBA | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | *RateFP* | *RateFN* | *RateUC* | *TC* | *RateFP* | *RateFN* | *RateUC* | *TC* | Improvement 1 | *RateFP* | *RateFN* | *RateUC* | *TC* | Improvement 2 | Improvement 3 |
| SH | SVM | 0.0 | 44.1 | 2.9 | 44.1 | | | | | | 13.2 | 16.2 | 14.7 | 29.4 | 33.3 | |
| | DT | 8.8 | 23.5 | 1.5 | 32.4 | | | N/A | | | 7.4 | 14.7 | 16.2 | 22.1 | 31.8 | N/A |
| | ANN | 2.9 | 22.1 | 5.9 | 25.0 | | | | | | 2.9 | 11.8 | 26.5 | 14.7 | 41.2 | |
| ACA | SVM | 0.0 | 35.8 | 6.9 | 35.8 | | | | | | 0.0 | 35.8 | 6.9 | 35.8 | 0.0 | |
| | DT | 1.2 | 28.3 | 2.3 | 29.5 | | | N/A | | | 1.2 | 28.3 | 2.3 | 29.5 | 0.0 | N/A |
| | ANN | 0.6 | 24.3 | 1.2 | 24.9 | | | | | | 0.6 | 24.3 | 1.2 | 24.9 | 0.0 | |
| GCD | SVM | 28.0 | 0.0 | 0.4 | 28.0 | | | | | | 6.0 | 8.0 | 21.6 | 14.0 | 50.0 | |
| | DT | 23.6 | 5.6 | 1.2 | 29.2 | | | N/A | | | 5.6 | 8.8 | 12.4 | 14.4 | 50.7 | N/A |
| | ANN | 21.2 | 8.4 | 1.2 | 29.6 | | | | | | 4.0 | 8.4 | 10.0 | 12.4 | 58.1 | |
| INS | SVM | 0.0 | 30.7 | 5.7 | 30.7 | | | | | | 6.8 | 19.3 | 17.0 | 26.1 | 14.8 | |
| | DT | 0.0 | 31.8 | 2.3 | 31.8 | | | N/A | | | 6.8 | 15.9 | 21.6 | 22.7 | 28.6 | N/A |
| | ANN | 5.7 | 17.0 | 10.2 | 22.7 | | | | | | 6.8 | 9.1 | 18.2 | 15.9 | 30.0 | |
| PA | SVM | 18.4 | 0.0 | 2.0 | 18.4 | | | | | | 6.1 | 12.2 | 2.0 | 18.4 | 0.0 | |
| | DT | 2.0 | 18.4 | 0.0 | 20.4 | | | N/A | | | 10.2 | 0.0 | 10.2 | 10.2 | 50.0 | N/A |
| | ANN | 4.1 | 16.3 | 4.1 | 20.4 | | | | | | 14.3 | 0.0 | 6.1 | 14.3 | 30.0 | |
| SPECTF | SVM | 20.9 | 0.0 | 1.5 | 20.9 | | | | | | 6.0 | 9.0 | 13.4 | 14.9 | 28.6 | |
| | DT | 1.5 | 22.4 | 3.0 | 23.9 | | | N/A | | | 20.9 | 0.0 | 3.0 | 20.9 | 12.5 | N/A |
| | ANN | 20.9 | 0.0 | 1.5 | 20.9 | | | | | | 13.4 | 0.0 | 10.4 | 13.4 | 35.7 | |

of attribute $j$ in $T$. A coefficient equal to 10,000 units was applied in Equation (20) instead of 100 units. This was due to the fact that if the percentage normalization was applied directly, then the value normalized this way might be too small. Such small values might lead to difficulties in comparisons and computations.

$$T_{i,j} = \frac{T_{i,j} \times 10{,}000}{\sum_i T_{i,j}} \tag{20}$$

The HBA and CBA assumed that $\beta^+$ and $\beta^-$ were in the range [0, 1], while $\alpha^+$ and $\alpha^-$ were in the range [0, 30] with both ranges having been determined empirically. The experiments were run on a PC with a CPU of 2.8GHZ speed and 3GB RAM under the Windows XP operating system. We used the libraries associated with the default settings in [Matlab, 2007] to implement the original algorithms. The following section shows the results on four families of tests that were based on four different settings of penalty costs.

### 6.2 Experimental Results

Family #1 of tests: The first family of tests did not penalize for the unclassifiable type (i.e., its cost is 0), but penalized at the same level, say of one unit, for the false-positive and false-negative types. This family of tests was equivalent to the analyses of previous studies. Thus, the objective function in this family was assumed to be:

$$TC = \min(1 \times RateFP + 1 \times RateFN).$$

Under the **four-fold cross-validation** method, the results for this family of tests are presented in Tables 2(a) and 2(b) [Pham and Triantaphyllou, 2010]. These tables show the three failure rates (as percentages) and the *TC*s obtained under the various algorithms. The column "Improvement 1" shows any improvements (as a percentage) of the *TC* achieved by the HBA

when compared to that of the original algorithm. The column "Improvement 2" shows any improvements (as a percentage) of the *TC* achieved by the CBA when compared to that of the original algorithm. Values in the columns "Improvement 1" and "Improvement 2" are defined by the percent decrease between the *TC* achieved by the original algorithm and the *TC* obtained from the HBA and CBA, respectively. Similarly, the column "Improvement 3" shows any improvements (as a percentage) of the *TC* achieved by the CBA when compared to that of the HBA. Each value in this column is defined by the difference between the corresponding values in the columns "Improvement 1" and "Improvement 2." A negative or zero value in the column "Improvement 3" shows that the CBA in terms of *TC* values had no improvement when compared to that of the HBA. The above notation is also used in Tables 3 to 12.

In particular, Tables 2(a) and 2(b) show that the HBA when applied on the PID, HSS, WBC, LD, AP, and FourClass datasets found the average *TC*s to be less than those of the original algorithms by about 85.9, 57.4, 55.4, 87.2, 100.0, and 26.2%, respectively. The results show that the HBA could not run (i.e., it is N/A) with the SH, ACA, GCD, INS, PA, and SPECTF datasets because of its excessive computing time. Furthermore, Tables 2(a) and 2(b) show that the CBA when applied on the PID, HSS, WBC, LD, AP, FourClass, SH, ACA, GCD, INS, PA, and SPECTF datasets found the average *TC*s to be less than or equal to those of the original algorithms by about 25.2, 13.2, 34.3, 35.4, 0.0, 0.0, 35.4, 0.0, 52.9, 24.5, 26.7, and 25.6%, respectively. All the results obtained by using the CBA in terms of the *TC* values had no improvement when compared to those from the HBA.

Tables 2(a) and 2(b) also show that some *RateUC* values achieved by the HBA and CBA were typically dominated by the sum of *RateFP* and *RateFN*. This was due to the fact that we did not penalize (i.e., that cost was equal to 0) for the unclassifiable type. Thus, the HBA and CBA

attempted to minimize the *TC* by classifying as many cases as possible in the unclassifiable type. This situation might lead to degenerative results with the extreme example of all cases being classified as the unclassifiable type. The degenerative results occurred when the HBA was applied on the LD and AP datasets. In such situations, the *TC* is equal to zero, but this is misleading.

**Table 3:** The accuracy percentages of the HBA and CBA under the four-fold cross-validation method [Pham and Triantaphyllou, 2010].

| Dataset | Original algorithm | HBA | CBA | Improvement 1 | Improvement 2 | Improvement 3 |
|---|---|---|---|---|---|---|
| PID | 69.1 | 95.7 | 76.7 | 26.6 | 7.6 | -19.0 |
| HSS | 68.4 | 86.6 | 72.7 | 18.2 | 4.3 | -13.9 |
| WBC | 75.9 | 89.4 | 84.3 | 13.5 | 8.4 | -5.1 |
| LD | 65.5 | 95.7 | 77.5 | 30.2 | 12 | -18.2 |
| AP | 91.4 | 100.0 | 91.4 | 8.6 | 0.0 | -8.6 |
| FourClass | 94.6 | 96.3 | 94.6 | 1.7 | 0.0 | -1.7 |
| SH | 66.2 | N/A | 77.9 | N/A | 11.7 | N/A |
| ACA | 66.9 | N/A | 69.9 | N/A | 3.0 | N/A |
| GCD | 71.1 | N/A | 86.4 | N/A | 15.3 | N/A |
| INS | 71.6 | N/A | 78.4 | N/A | 6.8 | N/A |
| PA | 80.3 | N/A | 85.7 | N/A | 5.4 | N/A |
| SPECTF | 78.1 | N/A | 83.6 | N/A | 5.5 | N/A |

Under the **four-fold cross-validation** method, Table 3 shows the accuracy percentages of the HBA and CBA in terms of the sum of *RateFP* and *RateFN* [Pham and Triantaphyllou, 2010]. This table shows that the HBA (when it was possible to use) was more accurate than the results obtained from the CBA. There were six datasets (HSS, LD, AP, FourClass, GCD, and SPECTF) for which the HBA and CBA were more accurate than the results obtained from the previous studies. However, there were six datasets (PID, WBC, SH, ACA, INS, and PA) for which the CBA was less accurate than (but not by much) the results obtained from previous studies.

**Table 4:** $TC = \min(1\times RateFP + 1\times RateFN)$ under the test-set method [Pham and Triantaphyllou, 2008b and 2009a].

| Dataset | Algo. | Original algorithm | | | | Original algorithm in conjunction with the HBA | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | *RateFP* | *RateFN* | *RateUC* | *TC* | *RateFP* | *RateFN* | *RateUC* | *TC* | Improvement 1 |
| PID | SVM | 0.0 | 38.5 | 56.8 | 38.5 | 0.0 | 5.2 | 74.5 | 5.2 | 86.5 |
| | DT | 14.1 | 18.8 | 61.5 | 32.9 | 0.0 | 8.3 | 58.9 | 8.3 | 74.8 |
| | ANN | 11.5 | 20.3 | 61.5 | 31.8 | 0.0 | 5.2 | 74.5 | 5.2 | 83.6 |
| HSS | SVM | 7.9 | 22.4 | 35.5 | 30.3 | 1.3 | 7.9 | 27.6 | 9.2 | 69.6 |
| | DT | 21.1 | 14.5 | 32.9 | 35.6 | 1.3 | 7.9 | 27.6 | 9.2 | 74.2 |
| | ANN | 11.8 | 15.8 | 36.8 | 27.6 | 1.3 | 9.2 | 18.4 | 10.5 | 62.0 |
| WBC | SVM | 12.5 | 15.3 | 52.8 | 27.8 | 1.4 | 0.0 | 29.2 | 1.4 | 95.0 |
| | DT | 16.7 | 12.5 | 51.4 | 29.2 | 2.8 | 0.0 | 12.5 | 2.8 | 90.4 |
| | ANN | 18.1 | 5.6 | 56.9 | 23.7 | 1.4 | 0.0 | 15.3 | 1.4 | 94.1 |
| LD | SVM | 43.5 | 0.0 | 55.1 | 43.5 | 0.0 | 0.0 | 97.1 | 0.0 | 100.0 |
| | DT | 23.2 | 17.4 | 52.2 | 40.6 | 1.4 | 0.0 | 89.9 | 1.4 | 96.6 |
| | ANN | 23.2 | 17.4 | 50.7 | 40.6 | 4.3 | 0.0 | 87.0 | 4.3 | 89.4 |
| AP | SVM | 0.0 | 4.8 | 95.2 | 4.8 | 0.0 | 0.0 | 100.0 | 0.0 | 100.0 |
| | DT | 19.0 | 4.8 | 76.2 | 23.8 | 0.0 | 0.0 | 100.0 | 0.0 | 100.0 |
| | ANN | 0.0 | 4.8 | 95.2 | 4.8 | 0.0 | 0.0 | 100.0 | 0.0 | 100.0 |

Under the **test-set** method, the results for this family of tests are presented in Table 4 [Pham and Triantaphyllou, 2008b and 2009a]. This table shows that the HBA when applied on the PID, HSS, WBC, LD, and AP datasets found the average *TC*s to be less than those of the original algorithms by about 81.6, 68.6, 93.2, 95.3, and 100.0%, respectively. Table 4 also shows that extreme degenerative results occurred when the HBA was applied on the LD and AP datasets.

Analyses of the impact of *RateFP*, *RateFN*, and *RateUC* focus on the corresponding penalty costs. A higher penalty cost for a given type of error implies that the system will result in fewer cases of that type. The following families of tests describe such situations.

Family #2 of tests: This family of tests analyzed the nine medical datasets: PID, HSS, WBC, LD, AP, FourClass, SH, PA, and SPECTF. In these tests, we penalized considerably more for the false-negative type than for the false-positive and unclassifiable types. For the medical datasets,

**Table 5:** $TC = \min(1\times RateFP + 20\times RateFN + 3\times RateUC)$ under the four-fold cross-validation method [Pham and Triantaphyllou, 2010]

| Dataset | Algo. | Original algorithm | | | | Original algorithm in conjunction with the HBA | | | | | Original algorithm in conjunction with the CBA | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | *RateFP* | *RateFN* | *RateUC* | *TC* | *RateFP* | *RateFN* | *RateUC* | *TC* | Improvement 1 | *RateFP* | *RateFN* | *RateUC* | *TC* | Improvement 2 | Improvement 3 |
| PID | SVM | 0.5 | 31.3 | 1.0 | 628.6 | 12.5 | 7.8 | 31.3 | 262.5 | 58.2 | 2.1 | 0.0 | 96.4 | 291.1 | 53.7 | -4.6 |
| | DT | 5.2 | 29.7 | 0.0 | 599.0 | 8.9 | 0.0 | 53.6 | 169.8 | 71.7 | 0.0 | 0.0 | 99.5 | 298.4 | 50.2 | -21.5 |
| | ANN | 3.1 | 22.9 | 11.5 | 495.8 | 8.3 | 2.6 | 52.6 | 218.2 | 56.0 | 0.0 | 0.0 | 100.0 | 300.0 | 39.5 | -16.5 |
| HSS | SVM | 26.0 | 5.2 | 5.2 | 145.5 | 20.8 | 1.3 | 1.3 | 50.6 | 65.2 | 27.3 | 1.3 | 0.0 | 53.2 | 63.4 | -1.8 |
| | DT | 19.5 | 15.6 | 0.0 | 331.2 | 15.6 | 1.3 | 3.9 | 53.2 | 83.9 | 26.0 | 1.3 | 2.6 | 59.7 | 82.0 | -2.0 |
| | ANN | 18.2 | 10.4 | 11.7 | 261.0 | 15.6 | 1.3 | 16.9 | 92.2 | 64.7 | 22.1 | 3.9 | 16.9 | 150.6 | 42.3 | -22.4 |
| WBC | SVM | 0.0 | 27.8 | 8.3 | 580.6 | 8.3 | 1.4 | 20.8 | 98.6 | 83.0 | 1.4 | 0.0 | 86.1 | 259.7 | 55.3 | -27.8 |
| | DT | 6.9 | 18.1 | 16.7 | 418.1 | 8.3 | 1.4 | 23.6 | 106.9 | 74.4 | 22.2 | 0.0 | 58.3 | 197.2 | 52.8 | -21.6 |
| | ANN | 1.4 | 18.1 | 22.2 | 429.2 | 8.3 | 1.4 | 22.2 | 102.8 | 76.1 | 1.4 | 0.0 | 86.1 | 259.7 | 39.5 | -36.6 |
| LD | SVM | 0.0 | 37.2 | 1.2 | 747.7 | 4.7 | 1.2 | 30.2 | 118.6 | 84.1 | 0.0 | 0.0 | 98.8 | 296.5 | 60.3 | -23.8 |
| | DT | 18.6 | 15.1 | 2.3 | 327.9 | 2.3 | 0.0 | 24.4 | 75.6 | 77.0 | 0.0 | 0.0 | 95.3 | 286.0 | 12.8 | -64.2 |
| | ANN | 10.5 | 22.1 | 11.6 | 487.2 | 4.7 | 0.0 | 54.7 | 168.6 | 65.4 | 26.7 | 0.0 | 53.5 | 187.2 | 61.6 | -3.8 |
| AP | SVM | 11.1 | 0.0 | 14.8 | 55.6 | 11.1 | 0.0 | 14.8 | 55.6 | 0.0 | 11.1 | 0.0 | 14.8 | 55.6 | 0.0 | 0.0 |
| | DT | 7.4 | 0.0 | 22.2 | 74.1 | 7.4 | 0.0 | 22.2 | 74.1 | 0.0 | 7.4 | 0.0 | 22.2 | 74.1 | 0.0 | 0.0 |
| | ANN | 7.4 | 0.0 | 25.9 | 85.2 | 7.4 | 0.0 | 25.9 | 85.2 | 0.0 | 7.4 | 0.0 | 25.9 | 85.2 | 0.0 | 0.0 |
| FourClass | SVM | 4.6 | 1.9 | 1.9 | 47.2 | 29.6 | 0.5 | 0.9 | 41.7 | 11.8 | 4.6 | 1.9 | 1.9 | 47.2 | 0.0 | -11.8 |
| | DT | 2.8 | 3.7 | 1.4 | 81.0 | 2.8 | 0.5 | 4.6 | 25.9 | 68.0 | 2.8 | 3.7 | 1.4 | 81.0 | 0.0 | -68.0 |
| | ANN | 0.0 | 3.2 | 2.8 | 73.1 | 2.8 | 0.5 | 7.4 | 34.3 | 53.2 | 0.0 | 3.2 | 2.8 | 73.1 | 0.0 | -53.2 |
| SH | SVM | 0.0 | 44.1 | 2.9 | 891.2 | | | | | | 0.0 | 0.0 | 100.0 | 300.0 | 66.3 | |
| | DT | 8.8 | 23.5 | 1.5 | 483.8 | | | N/A | | | 0.0 | 0.0 | 100.0 | 300.0 | 38.0 | N/A |
| | ANN | 2.9 | 22.1 | 5.9 | 461.8 | | | | | | 13.2 | 0.0 | 48.5 | 158.8 | 65.6 | |
| PA | SVM | 18.4 | 0.0 | 2.0 | 24.5 | | | | | | 18.4 | 0.0 | 2.0 | 24.5 | 0.0 | |
| | DT | 2.0 | 18.4 | 0.0 | 369.4 | | | N/A | | | 10.2 | 0.0 | 10.2 | 40.8 | 89.0 | N/A |
| | ANN | 4.1 | 16.3 | 4.1 | 342.9 | | | | | | 14.3 | 0.0 | 6.1 | 32.7 | 90.5 | |
| SPECTF | SVM | 20.9 | 0.0 | 1.5 | 25.4 | | | | | | 20.9 | 0.0 | 1.5 | 25.4 | 0.0 | |
| | DT | 1.5 | 22.4 | 3.0 | 458.2 | | | N/A | | | 20.9 | 0.0 | 3.0 | 29.9 | 93.5 | N/A |
| | ANN | 20.9 | 0.0 | 1.5 | 25.4 | | | | | | 20.9 | 0.0 | 1.5 | 25.4 | 0.0 | |

the false-negative type means that a patient who in reality has a disease is diagnosed as disease-free. The false-positive type means that a patient who in reality is disease-free is diagnosed as having the disease. It was hoped that the higher penalty cost for the false-negative type would result in fewer false-negative cases. We assumed that the penalty cost for the false-negative type was 20 units. The penalty costs for the false-positive and unclassifiable types were 1 and 3 units, respectively. Thus, the objective function for this family of tests was assumed to be:

$$TC = \min(1 \times RateFP + 20 \times RateFN + 3 \times RateUC).$$

Under the **four-fold cross-validation** method, the results for this family of tests are presented in Table 5 [Pham and Triantaphyllou, 2010]. This table shows that the HBA when applied on the PID, HSS, WBC, LD, AP, and FourClass datasets found the average *TC*s to be less than those of the original algorithms by about 62.0, 71.3, 77.8, 75.5, 0.0, and 44.3%, respectively. The HBA could not run with the SH, PA, and SPECTF datasets because of its excessive computing time. Furthermore, Table 5 shows that the CBA when applied on the PID, HSS, WBC, LD, AP, FourClass, SH, PA, and SPECTF datasets found the average *TC*s to be less than or equal to those of the original algorithms by about 47.8, 62.5, 49.2, 44.9, 0.0, 0.0, 56.6, 59.8, and 31.2%, respectively. The results support the ability of the CBA to run with all nine medical datasets. All the results from the CBA in terms of the *TC* values had no improvement when compared to those from the HBA. Since we penalized more for the false-negative type, the average *RateFN* achieved by the CBA in Table 5 was less than that of the CBA in Tables 2(a) and 2(b) by about 90.6%.

**Table 6:** $TC = \min(1\times RateFP + 20\times RateFN + 3\times RateUC)$ under the test-set method [Pham and Triantaphyllou, 2008b and 2009a].

| Dataset | Algo. | Original algorithm | | | | Original algorithm in conjunction with the HBA | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | *RateFP* | *RateFN* | *RateUC* | *TC* | *RateFP* | *RateFN* | *RateUC* | *TC* | Improvement 1 |
| PID | SVM | 0.0 | 38.5 | 56.8 | 940.4 | 0.0 | 8.3 | 54.7 | 330.1 | 64.9 |
| | DT | 14.1 | 18.8 | 61.5 | 574.6 | 2.6 | 5.2 | 70.8 | 319.0 | 44.5 |
| | ANN | 11.5 | 20.3 | 61.5 | 602.0 | 0.0 | 5.2 | 74.5 | 327.5 | 45.6 |
| HSS | SVM | 7.9 | 22.4 | 35.5 | 562.4 | 1.3 | 13.2 | 11.8 | 300.7 | 46.5 |
| | DT | 21.1 | 14.5 | 32.9 | 409.8 | 1.3 | 9.2 | 18.4 | 240.5 | 41.3 |
| | ANN | 11.8 | 15.8 | 36.8 | 438.2 | 1.3 | 13.2 | 10.5 | 296.8 | 32.3 |
| WBC | SVM | 12.5 | 15.3 | 52.8 | 476.9 | 15.3 | 0.0 | 11.1 | 48.6 | 89.8 |
| | DT | 16.7 | 12.5 | 51.4 | 420.9 | 15.3 | 0.0 | 12.5 | 52.8 | 87.5 |
| | ANN | 18.1 | 5.6 | 56.9 | 300.8 | 15.3 | 0.0 | 11.1 | 48.6 | 83.8 |
| LD | SVM | 43.5 | 0.0 | 55.1 | 208.8 | 43.5 | 0.0 | 55.1 | 208.8 | 0.0 |
| | DT | 23.2 | 17.4 | 52.2 | 527.8 | 0.0 | 8.7 | 84.1 | 426.3 | 19.2 |
| | ANN | 23.2 | 17.4 | 50.7 | 523.3 | 0.0 | 0.0 | 94.2 | 282.6 | 46.0 |
| AP | SVM | 0.0 | 4.8 | 95.2 | 381.6 | 0.0 | 0.0 | 100.0 | 300.0 | 21.4 |
| | DT | 19.0 | 4.8 | 76.2 | 343.6 | 0.0 | 0.0 | 100.0 | 300.0 | 12.7 |
| | ANN | 0.0 | 4.8 | 95.2 | 381.6 | 0.0 | 0.0 | 100.0 | 300.0 | 21.4 |

Under the **test-set** method, the results for this family of tests are presented in Table 6 [Pham and Triantaphyllou, 2008b and 2009a]. This table shows that the HBA when applied on the PID, HSS, WBC, LD, and AP datasets found the average *TC*s to be less than those of the original algorithms by about 51.7, 40.0, 87.0, 21.7, and 18.5%, respectively. Table 6 also shows that extreme degenerative results occurred when the HBA was applied on the LD and AP datasets.

Furthermore, we penalized even more for the false-negative type. In particular, we considered the following objective function:

$$TC = \min(1\times RateFP + 100\times RateFN + 3\times RateUC).$$

**Table 7:** *TC* = min(1×*RateFP* + 100×RateFN + 3×RateUC) under the four-fold cross-validation method [Pham and Triantaphyllou, 2010].

| Dataset | Algo. | Original algorithm | | | | Original algorithm in conjunction with the HBA | | | | | Original algorithm in conjunction with the CBA | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | *RateFP* | *RateFN* | *RateUC* | *TC* | *RateFP* | *RateFN* | *RateUC* | *TC* | Improvement 1 | *RateFP* | *RateFN* | *RateUC* | *TC* | Improvement 2 | Improvement 3 |
| PID | SVM | 0.5 | 31.3 | 1.0 | 3,128.6 | 12.5 | 0.5 | 40.6 | 186.5 | 94.0 | 0.0 | 0.0 | 100.0 | 300.0 | 90.4 | -3.6 |
| | DT | 5.2 | 29.7 | 0.0 | 2,974.0 | 53.6 | 0.0 | 1.6 | 58.3 | 98.0 | 67.2 | 0.0 | 0.0 | 67.2 | 97.7 | -0.3 |
| | ANN | 3.1 | 22.9 | 11.5 | 2,329.2 | 12.5 | 0.0 | 50.5 | 164.1 | 93.0 | 2.1 | 0.0 | 95.3 | 288.0 | 87.6 | -5.3 |
| HSS | SVM | 26.0 | 5.2 | 5.2 | 561.0 | 18.2 | 1.3 | 2.6 | 155.8 | 72.2 | 27.3 | 1.3 | 0.0 | 157.1 | 72.0 | -0.2 |
| | DT | 19.5 | 15.6 | 0.0 | 1,577.9 | 15.6 | 1.3 | 2.6 | 153.2 | 90.3 | 26.0 | 1.3 | 2.6 | 163.6 | 89.6 | -0.7 |
| | ANN | 18.2 | 10.4 | 11.7 | 1,092.2 | 15.6 | 1.3 | 22.1 | 211.7 | 80.6 | 22.1 | 2.6 | 16.9 | 332.5 | 69.6 | -11.1 |
| WBC | SVM | 0.0 | 27.8 | 8.3 | 2,802.8 | 8.3 | 1.4 | 20.8 | 209.7 | 92.5 | 1.4 | 0.0 | 86.1 | 259.7 | 90.7 | -1.8 |
| | DT | 6.9 | 18.1 | 16.7 | 1,862.5 | 8.3 | 1.4 | 15.3 | 193.1 | 89.6 | 22.2 | 0.0 | 58.3 | 197.2 | 89.4 | -0.2 |
| | ANN | 1.4 | 18.1 | 22.2 | 1,873.6 | 8.3 | 1.4 | 23.6 | 218.1 | 88.4 | 1.4 | 0.0 | 86.1 | 259.7 | 86.1 | -2.2 |
| LD | SVM | 0.0 | 37.2 | 1.2 | 3,724.4 | 7.0 | 0.0 | 80.2 | 247.7 | 93.3 | 0.0 | 0.0 | 98.8 | 296.5 | 92.0 | -1.3 |
| | DT | 18.6 | 15.1 | 2.3 | 1,537.2 | 2.3 | 0.0 | 24.4 | 75.6 | 95.1 | 50.0 | 0.0 | 16.3 | 98.8 | 93.6 | -1.5 |
| | ANN | 10.5 | 22.1 | 11.6 | 2,254.7 | 4.7 | 0.0 | 54.7 | 168.6 | 92.5 | 26.7 | 0.0 | 53.5 | 187.2 | 91.7 | -0.8 |
| AP | SVM | 11.1 | 0.0 | 14.8 | 55.6 | 11.1 | 0.0 | 14.8 | 55.6 | 0.0 | 11.1 | 0.0 | 14.8 | 55.6 | 0.0 | 0.0 |
| | DT | 7.4 | 0.0 | 22.2 | 74.1 | 7.4 | 0.0 | 22.2 | 74.1 | 0.0 | 7.4 | 0.0 | 22.2 | 74.1 | 0.0 | 0.0 |
| | ANN | 7.4 | 0.0 | 25.9 | 85.2 | 7.4 | 0.0 | 25.9 | 85.2 | 0.0 | 7.4 | 0.0 | 25.9 | 85.2 | 0.0 | 0.0 |
| FourClass | SVM | 4.6 | 1.9 | 1.9 | 195.4 | 3.2 | 0.9 | 29.6 | 184.7 | 5.5 | 4.6 | 1.9 | 1.9 | 195.4 | 0.0 | -5.5 |
| | DT | 2.8 | 3.7 | 1.4 | 377.3 | 2.3 | 1.4 | 43.1 | 270.4 | 28.3 | 2.8 | 3.7 | 1.4 | 377.3 | 0.0 | -28.3 |
| | ANN | 0.0 | 3.2 | 2.8 | 332.4 | 4.6 | 0.0 | 50.5 | 156.0 | 53.1 | 0.0 | 3.2 | 2.8 | 332.4 | 0.0 | -53.1 |
| SH | SVM | 0.0 | 44.1 | 2.9 | 4,420.6 | | | | | | 0.0 | 0.0 | 100.0 | 300.0 | 93.2 | |
| | DT | 8.8 | 23.5 | 1.5 | 2,366.2 | | | N/A | | | 0.0 | 0.0 | 100.0 | 300.0 | 87.3 | N/A |
| | ANN | 2.9 | 22.1 | 5.9 | 2,226.5 | | | | | | 13.2 | 0.0 | 48.5 | 158.8 | 92.9 | |
| PA | SVM | 18.4 | 0.0 | 2.0 | 24.5 | | | | | | 18.4 | 0.0 | 2.0 | 24.5 | 0.0 | |
| | DT | 2.0 | 18.4 | 0.0 | 1,838.8 | | | N/A | | | 10.2 | 0.0 | 10.2 | 40.8 | 97.8 | N/A |
| | ANN | 4.1 | 16.3 | 4.1 | 1,649.0 | | | | | | 14.3 | 0.0 | 6.1 | 32.7 | 98.0 | |
| SPECTF | SVM | 20.9 | 0.0 | 1.5 | 25.4 | | | | | | 20.9 | 0.0 | 1.5 | 25.4 | 0.0 | |
| | DT | 1.5 | 22.4 | 3.0 | 2,249.3 | | | N/A | | | 20.9 | 0.0 | 3.0 | 29.9 | 98.7 | N/A |
| | ANN | 20.9 | 0.0 | 1.5 | 25.4 | | | | | | 20.9 | 0.0 | 1.5 | 25.4 | 0.0 | |

**Table 8:** $TC = \min(1\times RateFP + 100\times RateFN + 3\times RateUC)$ under the test-set method [Pham and Triantaphyllou, 2008b and 2009a].

| Dataset | Algo. | Original algorithm | | | | Original algorithm in conjunction with the HBA | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | *RateFP* | *RateFN* | *RateUC* | *TC* | *RateFP* | *RateFN* | *RateUC* | *TC* | Improvement 1 |
| PID | SVM | 0.0 | 38.5 | 56.8 | 4,020.4 | 31.8 | 0.5 | 12.5 | 119.3 | 97.0 |
| | DT | 14.1 | 18.8 | 61.5 | 2,078.6 | 29.2 | 0.5 | 16.7 | 129.3 | 93.8 |
| | ANN | 11.5 | 20.3 | 61.5 | 2,226.0 | 30.7 | 0.0 | 15.6 | 77.5 | 96.5 |
| HSS | SVM | 7.9 | 22.4 | 35.5 | 2,354.4 | 14.5 | 0.0 | 13.2 | 54.1 | 97.7 |
| | DT | 21.1 | 14.5 | 32.9 | 1,569.8 | 14.5 | 0.0 | 9.2 | 42.1 | 97.3 |
| | ANN | 11.8 | 15.8 | 36.8 | 1,702.2 | 14.5 | 0.0 | 13.2 | 54.1 | 96.8 |
| WBC | SVM | 12.5 | 15.3 | 52.8 | 1,700.9 | 2.8 | 8.3 | 19.4 | 891.0 | 47.6 |
| | DT | 16.7 | 12.5 | 51.4 | 1,420.9 | 16.7 | 12.5 | 51.4 | 1,420.9 | 0.0 |
| | ANN | 18.1 | 5.6 | 56.9 | 748.8 | 18.1 | 5.6 | 56.9 | 748.8 | 0.0 |
| LD | SVM | 43.5 | 0.0 | 55.1 | 208.8 | 43.5 | 0.0 | 55.1 | 208.8 | 0.0 |
| | DT | 23.2 | 17.4 | 52.2 | 1,919.8 | 0.0 | 0.0 | 97.1 | 291.3 | 84.8 |
| | ANN | 23.2 | 17.4 | 50.7 | 1,915.3 | 0.0 | 0.0 | 95.7 | 287.1 | 85.0 |
| AP | SVM | 0.0 | 4.8 | 95.2 | 765.6 | 0.0 | 0.0 | 100.0 | 300.0 | 60.8 |
| | DT | 19.0 | 4.8 | 76.2 | 727.6 | 0.0 | 0.0 | 100.0 | 300.0 | 58.8 |
| | ANN | 0.0 | 4.8 | 95.2 | 765.6 | 0.0 | 0.0 | 100.0 | 300.0 | 60.8 |

Under the **four-fold cross-validation** method, the results for this family of tests are presented in Table 7 [Pham and Triantaphyllou, 2010]. This table shows that the HBA when applied on the PID, HSS, WBC, LD, AP, and FourClass datasets found the average *TC*s to be less than those of the original algorithms by about 95.0, 81.0, 90.2, 93.7, 0.0, and 29.0%, respectively. The HBA was not applicable to the SH, PA, and SPECTF datasets because of its excessive computing time. Furthermore, Table 7 shows that the CBA when applied on the PID, HSS, WBC, LD, AP, FourClass, SH, PA, and SPECTF datasets found the average *TC*s to be less than those of the original algorithms by about 91.9, 77.1, 88.8, 92.4, 0.0, 0.0, 91.1, 65.3, and 32.9%, respectively. The results prove that the CBA could again run with all nine medical datasets. All the results from the CBA in terms of the *TC* values had no improvement when compared to those from the HBA. Since we penalized considerably more for the false-negative

type, the average *RateFN* achieved by the CBA in Table 7 was less than that of the CBA in Tables 2(a), 2(b), and 5 by about 92.2 and 16.7%, respectively. These results demonstrate that as the penalty cost for the false-negative type became higher, even fewer false-negative cases were found.

Some results achieved by the CBA when applied on the medical datasets in Table 7 mirrored those on Table 5. This was due to the fact that the CBA optimized the *TC* by attempting to minimize *RateFN*. This rate was almost equal to zero when the CBA was applied on these datasets with the penalty cost equal to 20 units. Thus, the higher penalty costs for the false-negative type could not result in an even smaller value for *RateFN*. Hence, the *TC* was the same. Similarly, the results achieved by the CBA when applied on the AP and FourClass datasets in Tables 2(a), 2(b), 5, and 7 were identical to those under the original algorithms. The reason was that the CBA optimized the *TC* by attempting to minimize *RateFN*. This rate was almost equal to zero when the original algorithms were applied on these datasets. Tables 5 and 7 also show that some degenerative results occurred when the HBA and CBA were applied on the PID, WBC, LD and SH datasets.

Under the **test-set** method (see also Section 6.1), the results for this family of tests are presented in Table 8 [Pham and Triantaphyllou, 2008b and 2009a]. This table shows that the HBA when applied on the PID, HSS, WBC, LD, and AP datasets found the average *TC*s to be less than those of the original algorithms by about 95.8, 97.3, 15.9, 56.6, and 60.1%, respectively. Table 8 also shows that degenerative results occurred when the HBA was applied on the LD and AP datasets. Since we penalized considerably more for the false-negative type, the average *RateFN* achieved by the HBA in Table 8 was less than that of the HBA in Table 4 by about 37.9%. These results also show that as the penalty cost for the false-negative type became

higher, even fewer false-negative cases were found. Tables 2(a), 2(b), 4, 5, 6, 7, and 8 show that the **four-fold cross-validation** method provided slightly better results than those of the **test-set** method. Thus, the following families of tests would not apply the **test-set** method to analyze the performance of the algorithms.

Family #3 of tests: This family of tests analyzed the ACA, GCD, and INS datasets which are related to applications of credit approval and physics. The HBA was not applicable to these datasets because of its excessive computing time. We created a greater penalty for the false-positive type than for the false-negative and unclassifiable types with the hope that the higher penalty cost for false-positive type would result in fewer false-positive cases. We assumed that the penalty cost for the false-positive type was 20 units. The penalty costs for the false-negative and unclassifiable types were 1 and 3 units, respectively. Thus, the objective function in this family of tests was assumed to be:

$$TC = \min(20 \times RateFP + 1 \times RateFN + 3 \times RateUC).$$

Under the **four-fold cross-validation** method, Table 9 presents the results for this family of tests [Pham and Triantaphyllou, 2010]. This table shows that the CBA when applied on the ACA, GCD, and INS datasets found the average *TC*s to be less than or equal to those of the original algorithms by about 0.0, 54.9, and 16.7%, respectively. Since we penalized more for the false-positive type, the average *RateFP* achieved by the CBA in Table 9 was less than that of the CBA in Tables 2(a) and 2(b) by about 95.2%.

**Table 9:** $TC = \min(20\times RateFP + 1\times RateFN + 3\times RateUC)$ under the four-fold cross-validation method [Pham and Triantaphyllou, 2010].

| Dataset | Algo. | Original algorithm | | | | Original algorithm in conjunction with the CBA | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | *RateFP* | *RateFN* | *RateUC* | *TC* | *RateFP* | *RateFN* | *RateUC* | *TC* | Improvement 2 |
| ACA | SVM | 0.0 | 35.8 | 6.9 | 56.6 | 0.0 | 35.8 | 6.9 | 56.6 | 0.0 |
| | DT | 1.2 | 28.3 | 2.3 | 58.4 | 1.2 | 28.3 | 2.3 | 58.4 | 0.0 |
| | ANN | 0.6 | 24.3 | 1.2 | 39.3 | 0.6 | 24.3 | 1.2 | 39.3 | 0.0 |
| GCD | SVM | 28.0 | 0.0 | 0.4 | 561.2 | 0.0 | 0.0 | 100.0 | 300.0 | 46.5 |
| | DT | 23.6 | 5.6 | 1.2 | 481.2 | 0.0 | 65.2 | 8.4 | 90.4 | 81.2 |
| | ANN | 21.2 | 8.4 | 1.2 | 436.0 | 0.0 | 6.8 | 89.2 | 274.4 | 37.1 |
| INS | SVM | 0.0 | 30.7 | 5.7 | 47.7 | 0.0 | 30.7 | 5.7 | 47.7 | 0.0 |
| | DT | 0.0 | 31.8 | 2.3 | 38.6 | 0.0 | 31.8 | 2.3 | 38.6 | 0.0 |
| | ANN | 5.7 | 17.0 | 10.2 | 161.4 | 0.0 | 19.3 | 20.5 | 80.7 | 50.0 |

Furthermore, we penalized even more for the false-positive type. In particular, we considered the following objective function:

$$TC = \min(100\times RateFP + 1\times RateFN + 3\times RateUC).$$

**Table 10:** $TC = \min(100\times RateFP + 1\times RateFN + 3\times RateUC)$ under the four-fold cross-validation method [Pham and Triantaphyllou, 2010].

| Dataset | Algo. | Original algorithm | | | | Original algorithm in conjunction with the CBA | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | *RateFP* | *RateFN* | *RateUC* | *TC* | *RateFP* | *RateFN* | *RateUC* | *TC* | Improvement 2 |
| ACA | SVM | 0.0 | 35.8 | 6.9 | 56.6 | 0.0 | 35.8 | 6.9 | 56.6 | 0.0 |
| | DT | 1.2 | 28.3 | 2.3 | 150.9 | 1.2 | 28.3 | 2.3 | 150.9 | 0.0 |
| | ANN | 0.6 | 24.3 | 1.2 | 85.5 | 0.0 | 24.3 | 19.7 | 83.2 | 2.7 |
| GCD | SVM | 28.0 | 0.0 | 0.4 | 2,801.2 | 0.0 | 0.0 | 100.0 | 300.0 | 89.3 |
| | DT | 23.6 | 5.6 | 1.2 | 2,369.2 | 0.0 | 65.2 | 8.4 | 90.4 | 96.2 |
| | ANN | 21.2 | 8.4 | 1.2 | 2,132.0 | 0.0 | 10.8 | 80.4 | 252.0 | 88.2 |
| INS | SVM | 0.0 | 30.7 | 5.7 | 47.7 | 0.0 | 30.7 | 5.7 | 47.7 | 0.0 |
| | DT | 0.0 | 31.8 | 2.3 | 38.6 | 0.0 | 31.8 | 2.3 | 38.6 | 0.0 |
| | ANN | 5.7 | 17.0 | 10.2 | 615.9 | 0.0 | 19.3 | 20.5 | 80.7 | 86.9 |

Under the **four-fold cross-validation** method, the results for the third family of tests are presented in Table 10 [Pham and Triantaphyllou, 2010]. This table shows that the CBA when applied on the ACA, GCD, and INS datasets found the average *TC*s to be less than or equal to those of the original algorithms by about 0.9, 91.2, and 29.0%, respectively. Since we penalized considerably more for the false-positive type, the average *RateFP* achieved by the CBA in Table 10 was less than that of the CBA in Tables 2(a), 2(b), and 9 by about 97.6 and 50.0%, respectively. The above results show that as the penalty cost for the false-positive type became higher, fewer false-positive cases were found.

Some results achieved by the CBA when applied on the ACA, GCD, and INS datasets in Table 10 were the same as those of Table 9. As before, this was due to the fact that the CBA optimized the *TC* by attempting to minimize *RateFP*. This rate was almost equal to zero when the CBA was applied on these datasets with the penalty cost equal to 20 units. Thus, the higher penalty costs for the false-negative type could not result in a smaller value for *RateFP*. Hence, the *TC* was the same. In Tables 9 and 10, some degenerative results also occurred when the CBA was applied on the GCD dataset.

As seen in Tables 2(a), 2(b), 4, 5, 6, 7, 8, 9, and 10, the HBA and CBA achieved *TC*s less than or equal to (i.e., better) those of the original algorithms. Some identical *TC*s occurred because the GA approach used in the HBA and CBA was initialized by one of the 20 chromosomes whose values were (1, 1, 0, 0). This setting was discussed in Section 4.3.4. This is equivalent to the setting of the original algorithms. If the GA approach did not find a better *TC* value, then the HBA and CBA would return the *TC* value achieved by the original algorithms.

Tables 2(a), 2(b), 5, 7, 9, and 10 show that the *TC*s obtained by the CBA were typically greater than (i.e., worse) those obtained by the HBA. The number of cases in which the *TC*s

obtained by the CBA were identical to those achieved by the original algorithms occurred more frequently than those achieved by the HBA. In fact, only ten cases under the HBA were identical to those under the original algorithms compared to thirty-seven cases under the CBA. The above results show that the classification models derived under the CBA might not be as accurate as those under the HBA.

Family #4 of tests: The final family of tests analyzed the change of only one of the penalty costs (say for the false-negative type) from very low to very high values, while we kept the same level for the other two types. The purpose of this family of tests was to demonstrate that indeed the CBA can be very versatile with changes on the penalty costs. The HSS and PA datasets were used in this family as demonstrations. The SVM and DT in conjunction with the CBA were applied on the HSS and PA datasets, respectively. We assumed the same penalty cost for the false-positive and unclassifiable types and that was equal to one unit each. The penalty cost for the false-negative type was set at different values from 1 to 100 units.

Under the **four-fold cross-validation** method, the results for the HSS and PA datasets in the fourth family of tests are presented in Figures 31 and 32, respectively. In these figures, the X-axis represents the different penalty costs for *RateFN*, while the Y-axis shows *RateFN* (as a percentage) obtained under different penalty costs.

The plots in these two figures show that the CBA determined different levels of *RateFN* at costs equal to 1, 5 or 75 units for the HSS dataset and at 1 or 15 units for the PA dataset. However, *RateFN* was at the same level for costs in the intervals [1, 1], [5, 70] or [75, 100] with the HSS dataset and costs in the intervals [1, 10] or [15, 100] with the PA dataset. These plots also show that the level of *RateFN* was inversely proportional with the penalty cost.

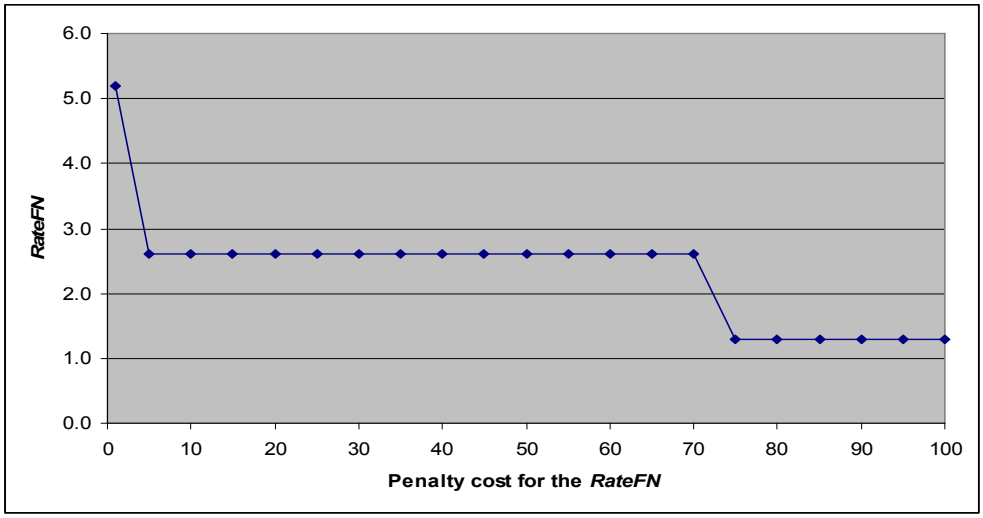



**Figure 31:** *RateFN* (in %) under different values of $C_{FN}$ when the CBA in conjunction with an SVM was applied on the HSS dataset [Pham and Triantaphyllou, 2010].

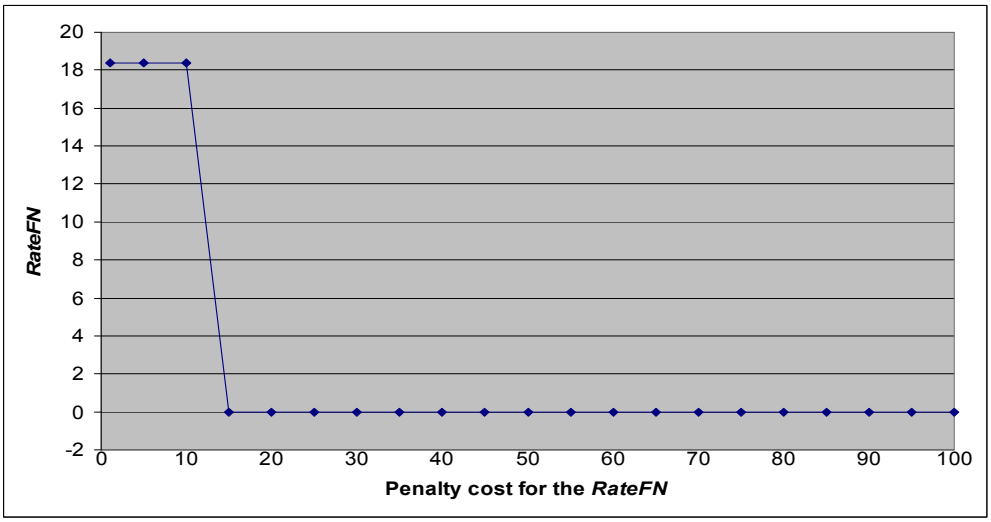



**Figure 32**: *RateFN* (in %) under different values of $C_{FN}$ when the CBA in conjunction with a DT was applied on the PA dataset [Pham and Triantaphyllou, 2010].

The above observations show that the plots followed stepwise patterns. There are two issues which can be seen from the stepwise patterns. First, a higher penalty cost for one type of error could indeed result in fewer errors of the corresponding type. Secondly, suppose that the CBA finds two different penalty costs whose error rates are at the same level. For any costs in the

interval between the two costs, the CBA also derives the same level of the error rate. Thus, we may use a binary search that will be discussed in the following chapter to determine all such intervals and the points of change.

# CHAPTER 7. A BINARY SEARCH APPROACH TO DETERMINE TRENDS OF CLASSIFICATION ERROR RATES[6]

## 7.1 Motivation

Some theoretical and practical developments have been made in the past two decades regarding the determination of penalties for classification errors. Some approaches depended on experts to assess penalties for classification errors. However, the approaches, in many instances, amounted to little more than guesswork. Instead of fixed penalties, [Hollmen, et. al., 2000] formulated the penalties as functions. Using a Bayesian formulation, these functions were allowed to operate on training data and were recalculated for each training point. This approach was employed in the detection of telecommunication fraud. By looking at the training dataset, [Zadronzy, et. al., 2001] applied a least-square multiple linear regression to predict the penalties, and such penalties were then used for testing instances. [Ikizler and Güvenir, 2003] proposed an approach, the Benefit-Maximizing Classifier with Feature Intervals (BMFI), which applied five different voting methods to determine penalties. However, none of the above approaches attempted to find general patterns of penalties for classification errors.

The family #4 of tests in the previous chapter shows that rates of classification errors might be arranged in stepwise patterns. The finding of stepwise patterns of classification errors can have important implications for data mining researchers in determining penalties for classification errors. In fact, for given penalties, a classification approach will exhibit some error rates. An interesting question is to find out whether an error rate stays the same if the

[6] A partial part of this chapter was in a pending paper written by Pham, H. N. A. and Triantaphyllou, E.

corresponding penalty is increased or decreased. If the rate is not the same, it should show the penalties for which the level of the rate is changed. Stepwise patterns can answer such questions. Each step of a stepwise pattern shows an interval of penalties for which the error rates are identical. The different steps of the pattern depict penalties where the error rates vary across levels.

The next section of this chapter explores patterns found in the family #4 of tests. Then, a binary search approach (BSA) is proposed to determine such patterns. LD, PA, WBC, and HSS datasets located at UCI [Asuncion, et. al., 2009] are used as test materials. Finally, some computational results are reported for the BSA.

### 7.2 An Exploration of Patterns

Assume that we are given a training dataset $T$ and a traditional classification approach $A$. Let $C$ denote the HBA, CBA, or any other similar approach. Assume that $C$ is the CBA. Recall that $C_{FP}$, $C_{FN}$, and $C_{UC}$ are the unit penalties for the false-positive, the false-negative, and the unclassifiable types, respectively. *RateFP*, *RateFN*, and *RateUC* denote the false-positive, the false-negative, and the unclassifiable rates, respectively. We assume that the values of $C_{FP}$ are initially in the interval $[a_1, a_2]$, while the values of $C_{FP}$ are in the interval $[b_1, b_2]$. The value of the *TC* (i.e., the total misclassification cost) is defined by:

$$TC = \min\ (C_{FP} \times RateFP + C_{FN} \times RateFN + C_{UC} \times RateUC\ ) . \qquad (1)$$

We also consider $C$ as a function. The $C$'s input is comprised of $T$, $C_{FP}$, $C_{FN}$, $C_{UC}$, and $A$. The function $C$ returns a minimum value of *TC* associated with the three error rates: *RateFP*, *RateFN*, and *RateUC*. Furthermore, suppose that the sum of penalties for the false-positive, false-negative, and unclassifiable types is equal to 100 units (i.e., $C_{FP} + C_{FN} + C_{UC} = 100$). Thus, when the penalties of $C_{FP}$ and $C_{FN}$ are available, we can use those penalties to determine $C_{UC}$. In the

following experiments, the values of $C_{FP}$ and $C_{FN}$ need to be considered. The BSA also employs all of the above assumptions and notation for its development.

In the following experiments, *C* in conjunction with the SVM approach was employed for HSS, LD, and WBC. For PA, the traditional classification approach *A* was a DT approach. As discussed in the previous chapter, the values of $C_{FN}$ and *RateFN* are usually considered in medical applications. For each given value of $C_{FP}$, *C* was run with different values of $C_{FN}$, ranging from low to high. In particular, the values of $C_{FP}$ were 10, 20, 30, 40, and 50 units. The values of $C_{FN}$ were assigned from 5 units and were sequentially increased by a step-size of 5 units. The sum of $C_{FP}$ and $C_{FN}$ has to be less than or equal to 100 units. The small step-size of $C_{FN}$ (i.e., 5 units) was adopted in order to find patterns for *RateFN* under different values of $C_{FP}$. The following tests were run on a PC with a CPU of 2.8GHZ speed and 3GB RAM under the Windows XP operating system.

In terms of the input: $C_{FP}$ and $C_{FN}$ and the output: *RateFN*, the results for LD, PA, WBC, and HSS are presented in Figures 33 to 36, respectively. As seen in these figures, the X-axis represents the different values of $C_{FN}$, while the Y-axis shows the *RateFN* (as a percentage) obtained for different values of $C_{FN}$. Each plot represents the relationship between $C_{FN}$ and *RateFN* for a given value of $C_{FP}$.

Figure 33 shows that *C*, when applied to LD, determined two levels for *RateFN*, 38.4 and 0.0%, under the different values of $C_{FP}$ and $C_{FN}$. All plots start with the same level (i.e., 38.4%) for *RateFN* because of the following reasons: First, the computational results show that the levels for *RateFP*, *RateFN*, and *RateUC* derived from the traditional classification approach *A* (i.e., the SVM approach) were equal to 0, 38.4, and 1.2%, respectively. Second, the levels for *RateFP* and *RateUC* were low. For given low values of $C_{FN}$, higher values of $C_{FP}$ and $C_{UC}$, when applied to

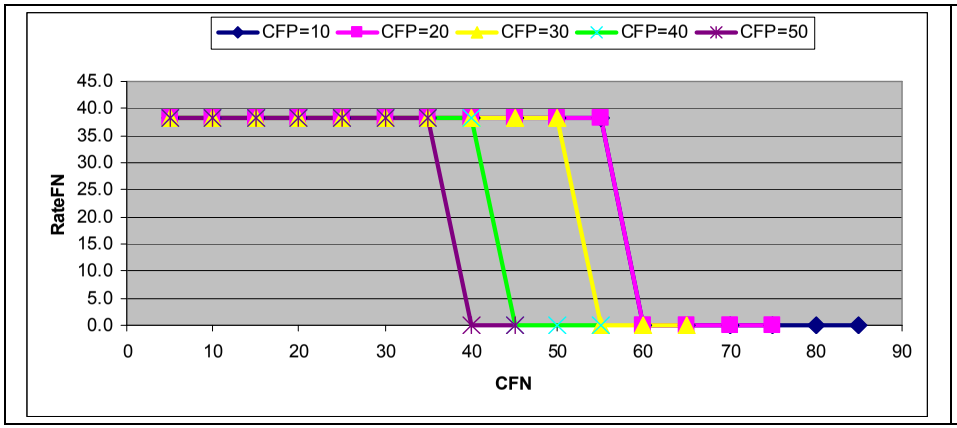



**Figure 33:** The results in terms of $C_{FP}$, $C_{FN}$, and *RateFN* when $C$ in conjunction with the SVM was applied to the LD dataset.

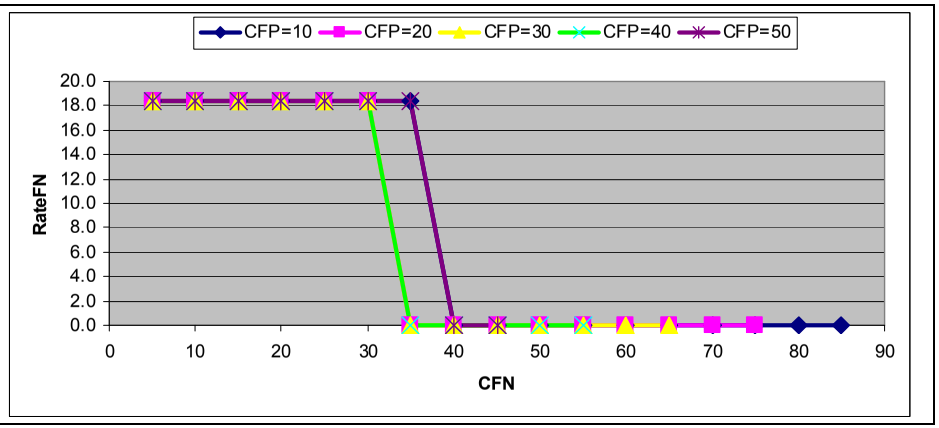



**Figure 34:** The results in terms of $C_{FP}$, $C_{FN}$, and *RateFN* when $C$ in conjunction with the DT was applied to the PA dataset.

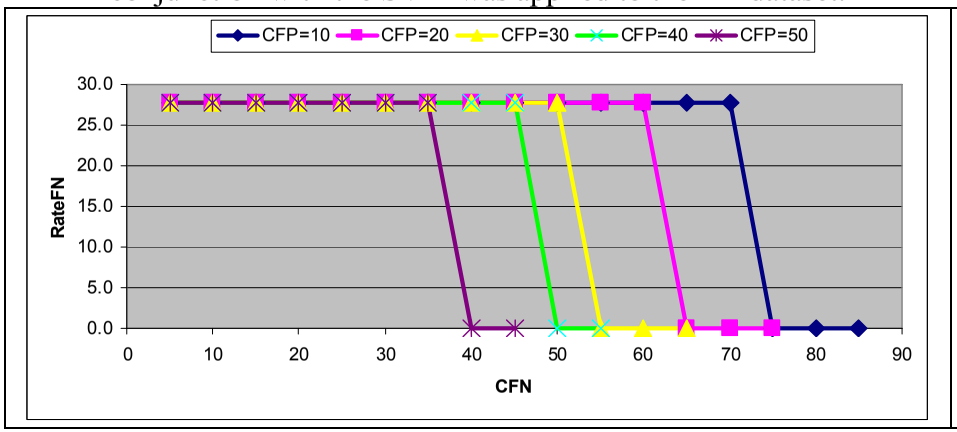



**Figure 35:** The results in terms of $C_{FP}$, $C_{FN}$, and *RateFN* when $C$ in conjunction with the SVM was applied to the WBC dataset.

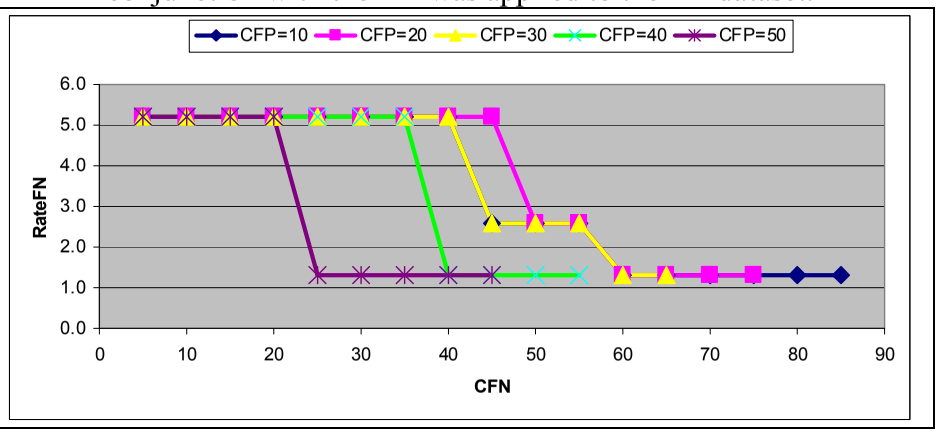



**Figure 36:** The results in terms of $C_{FP}$, $C_{FN}$, and *RateFN* when $C$ in conjunction with the SVM was applied to the HSS dataset.

*C*, did not result in lower levels for *RateFP* and *RateUC*. Hence, the levels for *RateFP*, *RateFN*, and *RateUC* were retained. In other words, all of the plots retained *RateFN* at 38.4% for low values of $C_{FN}$. All of the plots end at the second level (i.e., 0.0%) for *RateFN*. This is due to the fact that *C* used much higher values of $C_{FN}$.

Figure 34 shows that *C*, when applied to PA, also derived two levels for *RateFN*, 18.4 and 0.0%, under the different values of $C_{FP}$ and $C_{FN}$. The levels for *RateFP*, *RateFN*, and *RateUC* derived from the traditional classification approach *A* (i.e., the DT approach) were equal to 2.0, 18.4, and 0%, respectively. The levels for *RateFP* and *RateUC* were low. This situation is similar to the scenario depicted in Figure 33. Thus, all of the plots begin at the same level (i.e., 18.4%) for *RateFN*. The second level (i.e., 0.0%) for all of the plots was produced by greatly higher values of $C_{FN}$. In the same way, Figure 35 shows that *C*, when applied to WBC, derived two levels for *RateFN*, 27.8 and 0.0%, under the different values of $C_{FP}$ and $C_{FN}$. Because the levels for *RateFP*, *RateFN*, and *RateUC* derived from *A* (i.e., the SVM approach) were equal to 0.0, 27.8, and 8.3%, respectively, all of the plots begin at the same level (i.e., 27.8%) for *RateFN*. Upper values of $C_{FN}$ resulted in the second level (i.e., 0.0%) for all of the plots.

Figure 36 indicates that *C*, when applied to HSS, determined three levels for *RateFN*, 5.2, 2.6, and 1.3%, under the various values of $C_{FP}$ and $C_{FN}$. All of the plots in this figure begin at the level of 5.2%. This is due to the fact that the levels for *RateFP*, *RateFN*, and *RateUC* derived from the traditional classification approach *A* (i.e., the SVM approach) were equal to 26.0, 5.2, and 5.2%, respectively. Under the current situation, the levels for *RateFP* and *RateUC* were low. For given low values of $C_{FN}$, higher values of $C_{FP}$ and $C_{UC}$, when applied to *C*, did not result in lower levels for *RateFP* and *RateUC*. Hence, the levels for *RateFP*, *RateFN*, and *RateUC* were retained. In other words, all plots retained *RateFN* at 5.2% for low values of $C_{FN}$. For the values

of $C_{FP}$ which range from 10 to 30 units, the plots have two more levels (i.e., 2.6 and 1.3%). At the level of 2.6%, because the values of $C_{FN}$ which range from 45 to 55 units are higher than those of $C_{FP}$ and $C_{UC}$, $C$ optimized the $TC$ by somewhat minimizing $RateFN$. This resulted in the level 2.6%. Much higher values of $C_{FN}$ derived the level of 1.3%. The plot in which the value of $C_{FP}$ is 40 units and in which the values of $C_{FN}$ range from 40 units only has the level of 1.3%, because the values of $C_{FN}$ are always greater than those of $C_{FP}$ and $C_{UC}$. The plot in which the value of $C_{FP}$ is 50 units and in which the values of $C_{FN}$ range from 25 units has the level of 1.3%. Because the value of $C_{FP}$ is greater than the values of $C_{FN}$ and $C_{UC}$, $C$ optimized the $TC$ by minimizing $RateFP$. The computational results show that the minimum level of $RateFP$ was 14.3% where the corresponding level of $RateFN$ was 1.3%.

The stepwise patterns depicted in Figures 33 to 36 may be explained as follows. For a given training dataset $T$ and the traditional classification approach $A$, suppose that $C$ employs two different 3-tuples of the penalties ($C_{FP}$, $C_{FN}^1$, $C_{UC}$) and ($C_{FP}$, $C_{FN}^2$, $C_{UC}$). The values of $C_{FN}^1$ and $C_{FN}^2$ are much higher than those of $C_{FP}$ and $C_{UC}$. We also assume that $C_{FN}^1$ is less than $C_{FN}^2$. Under such input, $C$ returns $TC_1$ and $TC_2$, respectively. The values of $TC_1$ and $TC_2$ are associated with ($RateFP_1$, $RateFN_1$, $RateUC_1$) and with ($RateFP_2$, $RateFN_2$, $RateUC_2$), respectively. In order to minimize the values of the $TC$, the value of $RateFN_2$ are certainly less than or equal to $RateFN_1$. Furthermore, assume that we are in the case where the values of $RateFN_1$ and $RateFN_2$ are the same. Under this consideration, the third 3-tuple ($C_{FP}$, $C_{FN}^3$, $C_{UC}$) is applied to $C$. The value of $C_{FN}^3$ is in the interval $[C_{FN}^1, C_{FN}^2]$. Next, the value of $TC_3$ associated with ($RateFP_3$, $RateFN_3$, $RateUC_3$) is achieved by $C$. The value of $RateFN_3$ must be equal to $RateFN_1$ and $RateFN_2$. In fact, we assume that the values of $RateFN_3$ is either greater or less than those of $RateFN_1$ and $RateFN_2$. Such cases contradict the fact that when the higher value of the penalty is

given, the lower level of the corresponding error rate can be achieved by $C$. The above results will be used now to develop the BSA.

### 7. 3 The BSA

The plots seen in Section 7.2 exhibit stepwise patterns. Because the patterns show a declining trend, the BSA depicted in Figure 37 can be applied in order to reproduce them. The BSA's input is comprised of the training dataset $T$, the intervals $[a_1, a_2]$ and $[b_1, b_2]$ for $C_{FP}$ and $C_{FN}$, respectively, and the traditional classification approach $A$. With these input, the search space becomes a rectangle created by the intervals $[a_1, a_2]$ and $[b_1, b_2]$.

Because of the assumption in which the sum of $C_{FP}$, $C_{FN}$, and $C_{UC}$ is equal to 100 units, Steps 2 and 3 of the BSA show that if the initial search space does not satisfy the assumption, then the BSA returns the empty intervals. Otherwise, Steps 4 to 15 of the BSA apply $C$ to find the optimal values for $TC$ at the vertices of the search space. At each vertex, $C$ is only applied when the sum of $C_{FP}$ and $C_{FN}$ is less than or equal to 100 units. Otherwise, the optimal value of $TC$ and the associated rates are assigned to infinite values (denoted by " $\infty$ "s).

Furthermore, without loss of generality, we can restrict the consideration to the levels for *RateFN*. If the differences between the *RateFN*s (which are not infinite values ) are less than a threshold value, say 0.5 units, then the BSA is stopped and returns the current intervals $[a_1, a_2]$ and $[b_1, b_2]$ as seen at Step 17. Otherwise, the BSA eliminates three-quarters of the search space from the consideration, and then performs the search on the remaining space. Such tasks are seen in Steps 20 to 23.

The above procedure is due to the fact that the search space is a rectangle created by the intervals $[a_1, a_2]$ and $[b_1, b_2]$. In a similar methodology of a traditional binary search, the rectangle is divided into four sub-spaces. Each recursion is then applied to a corresponding sub-

space. By doing this repeatedly, the BSA will eventually return the non-overlapping areas in the search space where all of the levels for *RateFN* in each area are almost the same.

BinarySearch($T$, $a_1$, $a_2$, $b_1$, $b_2$, $A$ )

1. Begin
2. If ( $a_1 + b_1 > 100$ ) and ( $a_2 + b_2 > 100$ ) and ( $a_1 + b_2 > 100$ ) and ( $a_2 + b_1 > 100$ )
3. return the empty intervals
4. If ( $a_1 + b_1 \leq 100$ )
5. Set [$TC_1$, $RateFP_1$, $RateFN_1$, $RateUC_1$] = $C(T, a_1, b_1, 100 - a_1 - b_1, A)$
6. Else Set [$TC_1$, $RateFP_1$, $RateFN_1$, $RateUC_1$] = [ $\infty$ , $\infty$ , $\infty$ , $\infty$ ]
7. If ( $a_2 + b_2 \leq 100$ )
8. Set [$TC_2$, $RateFP_2$, $RateFN_2$, $RateUC_2$] = $C(T, a_2, b_2, 100 - a_2 - b_2, A)$
9. Else Set [$TC_2$, $RateFP_2$, $RateFN_2$, $RateUC_2$] = [ $\infty$ , $\infty$ , $\infty$ , $\infty$ ]
10. If ( $a_1 + b_2 \leq 100$ )
11. Set [$TC_3$, $RateFP_3$, $RateFN_3$, $RateUC_3$] = $C(T, a_1, b_2, 100 - a_1 - b_2, A)$
12. Else [$TC_3$, $RateFP_3$, $RateFN_3$, $RateUC_3$] = [ $\infty$ , $\infty$ , $\infty$ , $\infty$ ]
13. If ( $a_2 + b_1 \leq 100$ )
14. Set [$TC_4$, $RateFP_4$, $RateFN_4$, $RateUC_4$] = $C(T, a_2, b_1, 100 - a_2 - b_1, A)$
15. Else [$TC_4$, $RateFP_4$, $RateFN_4$, $RateUC_4$] = [ $\infty$ , $\infty$ , $\infty$ , $\infty$ ]
16. If ($RateFN_1$, $RateFN_2$, $RateFN_3$, and $RateFN_4$ are not $\infty$ ) and $|RateFN_1 - RateFN_2| \leq 0.5$ and $|RateFN_3 - RateFN_4| \leq 0.5$ and $|RateFN_1 - RateFN_3| \leq 0.5$
17. Return the intervals [$a_1$, $a_2$] and [$b_1$, $b_2$]
18. Else
19. Begin
20. BinarySearch($T$, $a_1$, $\frac{a_2 + a_1}{2}$, $b_1$, $\frac{b_2 + b_1}{2}$, $A$)
21. BinarySearch($T$, $a_1$, $\frac{a_2 + a_1}{2}$, $\frac{b_2 + b_1}{2}$, $b_2$, $A$)
22. BinarySearch($T$, $\frac{a_2 + a_1}{2}$, $a_2$, $b_1$, $\frac{b_2 + b_1}{2}$, $A$)
23. BinarySearch($T$, $\frac{a_2 + a_1}{2}$, $a_2$, $\frac{b_2 + b_1}{2}$, $b_2$, $A$)
24. End
25. End

**Figure 37:** The BSA.

### 7.4 Computational Results

We employed the BSA on the medical datasets introduced in Section 7.2. We assumed that the intervals [$a_1$, $a_2$] and [$b_1$, $b_2$] for $C_{FP}$ and $C_{FN}$ were [10, 50] and [5, 85], respectively. For LD,

PA, WBC, and HSS, the BSA divided the search spaces (i.e., the rectangles) into the areas as shown in Figures 38 to 41, respectively. In each search space, the horizontal-axis represents the $C_{FN}$ interval, while the vertical-axis shows the $C_{FP}$ interval. In each search space, the areas that are in different gray levels represent the different levels for *RateFN*. For examples, Figures 38 to 41 have 2, 2, 2, and 3 gray levels, respectively.

The above rectangular representation can be shown in the XYZ-coordinate system (or 3-D) as follows. That is, the X-axis represents the $C_{FN}$ interval; the Y-axis shows the $C_{FP}$ interval; and the Z-axis indicates the levels for *RateFN*. In this representation, the levels for *RateFN* follow a stair. When the steps of that stair are projected onto the XY-plane, they completely cover the search space and do not overlap as seen in Figures 38 to 41.

Furthermore, the search spaces depicted in Figures 38 to 41 can be represented in the same manner as the plots shown in Figures 33 to 36. The plots are shown in Figures 42 to 45. As it can be seen in these figures, each plot corresponds to a stepwise pattern whose levels and points of change are determined. Because the BSA divides a search space into non-overlapping areas, the levels of each plot are connected by straight lines.

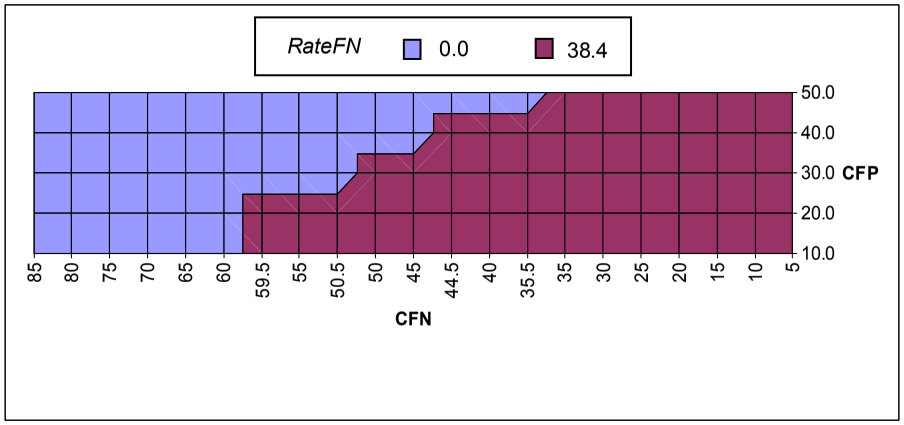



**Figure 38:** The results obtained from the BSA on the LD dataset.

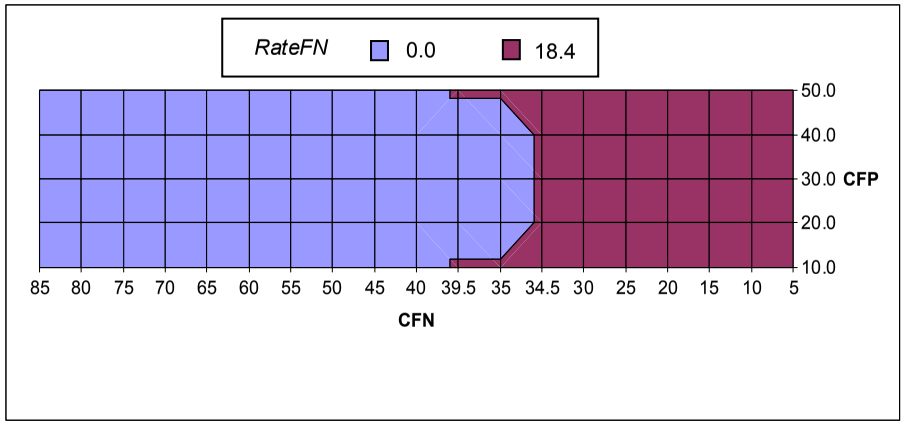



**Figure 39:** The results obtained from the BSA on the PA dataset.

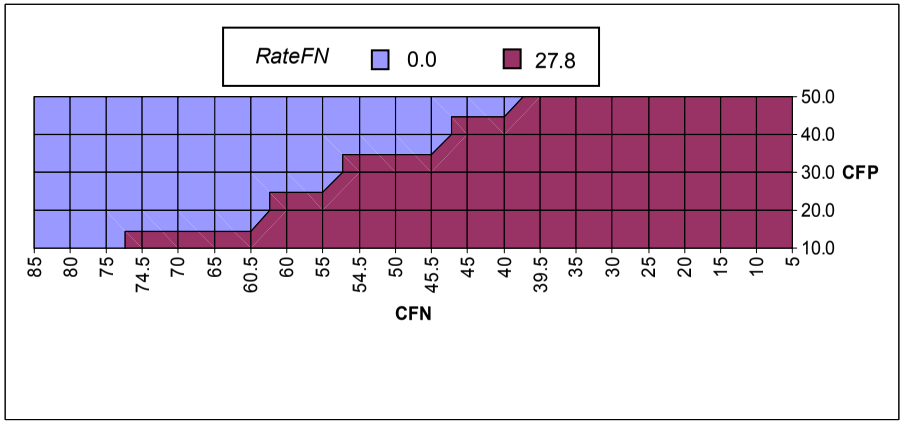



**Figure 40:** The results obtained from the BSA on the WBC dataset.

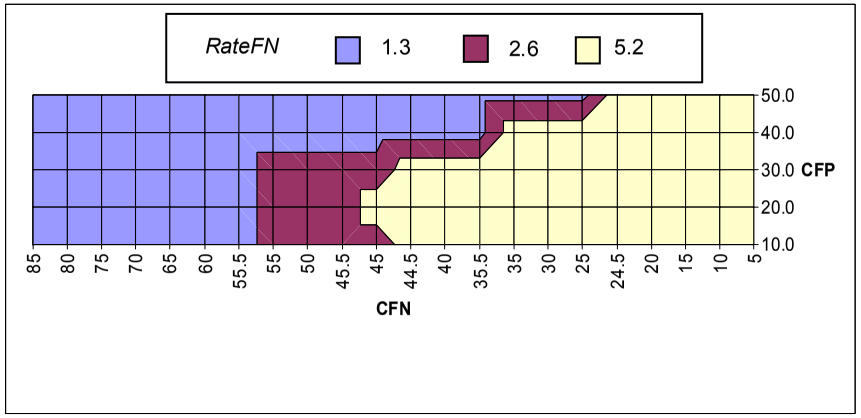



**Figure 41:** The results obtained from the BSA on the HSS dataset.

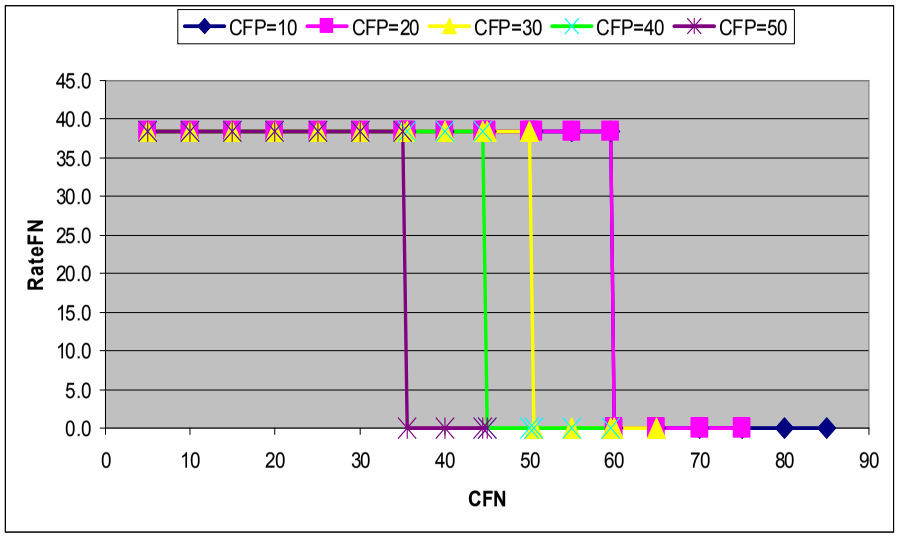



**Figure 42:** A different representation for the results obtained from the BSA on the LD dataset.

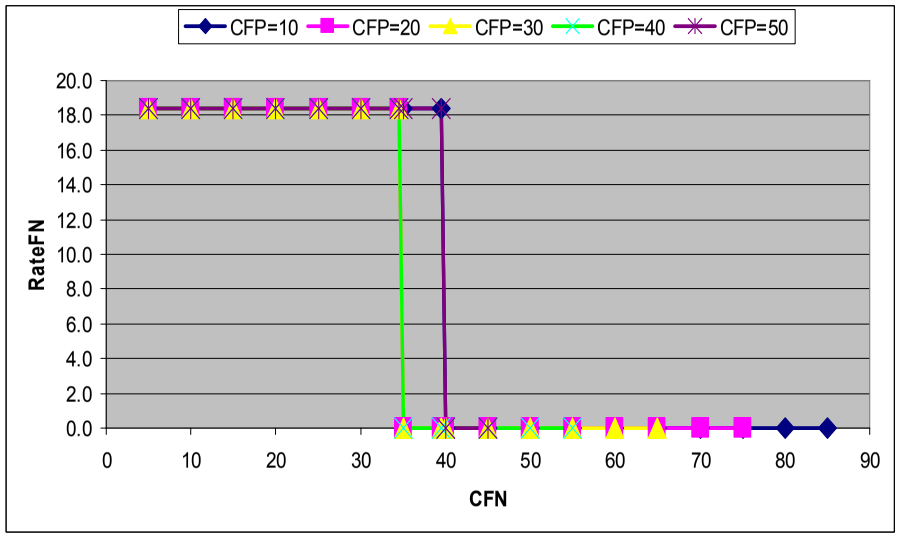



**Figure 43:** A different representation for the results obtained from the BSA on the PA dataset.

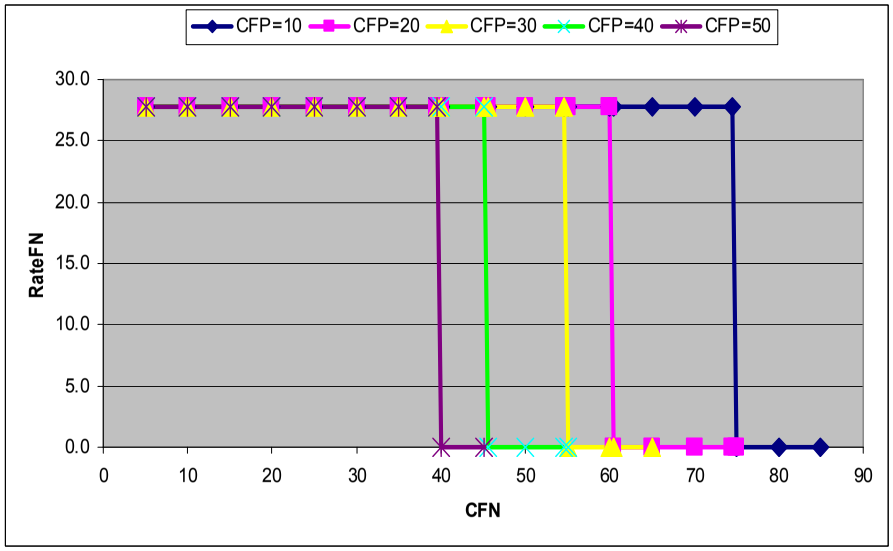



**Figure 44:** A different representation for the results obtained from the BSA on the WBC dataset.

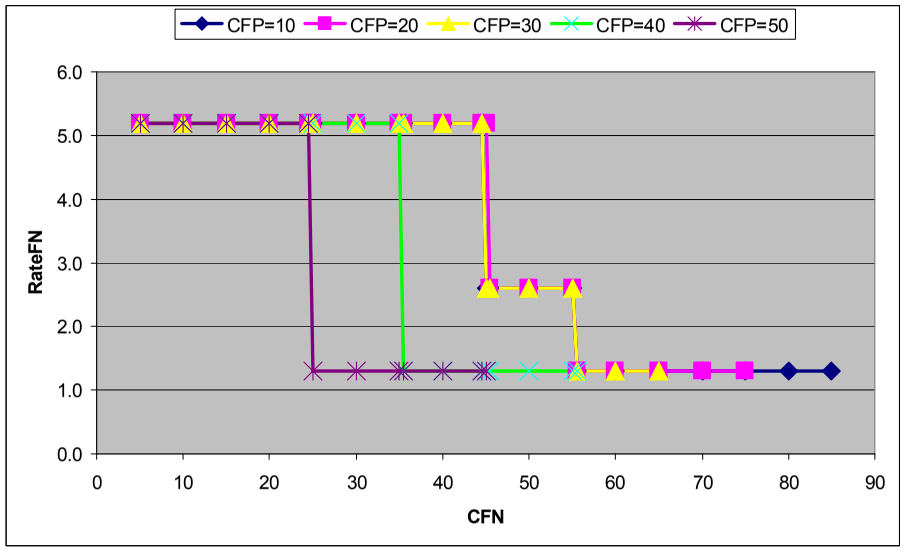



**Figure 45:** A different representation for the results obtained from the BSA on the HSS dataset.

# CHAPTER 8. CONCLUSIONS AND FUTURE WORK[7]

When current classification approaches derive models from training data, a balance between fitting and generalization is not considered in a systematic manner. Moreover, the error rates in conjunction with the penalty costs are not used in such approaches.

This dissertation recognized the above problems and proposed a method, called the Homogeneity-Based Algorithm (HBA) [Pham and Triantaphyllou, 2008a, 2008b, and 2009a] to deal with them. When the HBA is combined with traditional classification approaches (such as Support Vector Machines, Decision Trees, and Artificial Neural Networks), it may significantly enhance their prediction accuracy by using the concept of homogeneous regions. However, the HBA may be impractical for large datasets because of the excessive computing time required in the way the HBA determines homogeneous regions.

This dissertation also proposed a meta-heuristic approach, called the Convexity-Based Algorithm (CBA) [Pham and Triantaphyllou, 2010], to alleviate the problem of excessive computing time with the HBA. Instead of the concept of homogeneous regions, the CBA uses the concept of convex regions that do not depend on the dimensionality of the training data.

The HBA and CBA were studied on twelve datasets, found at the machine learning data repository at University of California, Irvine (UCI), USA, with different penalty costs for classification errors assigned in a systematic way. The computational results obtained from the CBA were compared with those of (when they were possible to use) the HBA. Those results are summarized in Tables 11 and 12.

---

[7] A partial part of this chapter was in Computers & Operations Research, Vol. 38, No. 1, 2010, pp. 174-189.

**Table 11:** The computing times (in hours) of the original algorithm, HBA, and CBA when applied on the datasets.

| Dataset | Original Algorithm | HBA | CBA | Times the CBA was faster than the HBA |
|---|---|---|---|---|
| PID | 0.76 | 14.95 | 4.14 | 3.61 |
| HSS | 0.26 | 2.05 | 0.65 | 3.15 |
| WBC | 0.06 | 14.81 | 0.45 | 32.91 |
| LD | 0.06 | 10.17 | 0.89 | 11.43 |
| AP | 0.01 | 12.09 | 0.20 | 60.45 |
| FourClass | 0.97 | 4.33 | 4.31 | 1.00 |
| SH | 0.13 | N/A | 0.38 | N/A |
| ACA | 1.11 | N/A | 3.34 | N/A |
| GCD | 3.94 | N/A | 6.55 | N/A |
| INS | 0.53 | N/A | 1.05 | N/A |
| PA | 0.11 | N/A | 0.22 | N/A |
| SPECTF | 0.13 | N/A | 0.36 | N/A |

**Table 12:** The TCs achieved by the original algorithms, HBA, and CBA when applied on the datasets.

| Dataset | Original-*TC* | HBA-*TC* | CBA-*TC* | Improvement 1 | Improvement 2 | Improvement 3 |
|---|---|---|---|---|---|---|
| PID | 30.9 | 4.3 | 23.3 | 86.0 | 24.7 | -61.3 |
| HSS | 31.6 | 13.4 | 27.3 | 57.5 | 13.7 | -43.8 |
| WBC | 24.1 | 10.6 | 15.7 | 55.8 | 34.6 | -21.2 |
| LD | 34.5 | 4.3 | 22.5 | 87.6 | 34.8 | -52.8 |
| AP | 8.6 | 0.0 | 8.6 | 100.0 | 0.0 | -100.0 |
| FourClass | 5.4 | 3.7 | 5.4 | 31.4 | 0.0 | -31.4 |
| SH | 33.8 | N/A | 22.1 | N/A | 34.8 | N/A |
| ACA | 30.1 | N/A | 30.1 | N/A | 0.0 | N/A |
| GCD | 28.9 | N/A | 13.6 | N/A | 53.0 | N/A |
| INS | 28.4 | N/A | 21.6 | N/A | 24.0 | N/A |
| PA | 19.7 | N/A | 14.3 | N/A | 27.6 | N/A |
| SPECTF | 21.9 | N/A | 16.4 | N/A | 25.0 | N/A |

Table 11 presents the computing times of the HBA and CBA when applied on the twelve datasets. The CBA computing time on each dataset was defined as the average of the computing time of the SVM, DT, and ANN in conjunction with the CBA. A similar definition was used for the original algorithm and the HBA computing time. Table 11 shows that the CBA was applicable to all twelve datasets, while the HBA was applicable to only six datasets.

Furthermore, the CBA computing time was always less than that of the HBA (when it was possible to use).

Table 12 is a summary of the results derived from Tables 2(a) and 2(b). In this table, the value "CBA-*TC*" for each given dataset was defined as the average of the *TC* achieved by the SVM, DT, and ANN in conjunction with the CBA. A similar definition was used for the "Original-*TC*" and "HBA-*TC*" cases. Table 12 shows that "CBA-*TC*" was always less than or equal to that of the original algorithms. However, "CBA-*TC*" was not less than that of the HBA (when it was possible to use). The results in Tables 11 and 12 indicate that the CBA was more efficient than the HBA, but less accurate than the HBA. However, the CBA was able to derive systems in situations where the HBA was not applicable due to excessive computing time.

The HBA and CBA also indicated that for different penalties of a classification error the corresponding error rate exhibited a stepwise pattern. Thus, this dissertation proposed a binary search approach (BSA) to derive such patterns. With the assumption that the search space is created by two penalty intervals of two classification errors, the BSA recursively splits it into sub-spaces to locate the steps of the patterns. The well-known medical datasets: Liver-Disorders, Parkinsons, Wisconsin Breast Cancer, and Haberman Surgery Survival were employed in the experiments. The BSA determined the stepwise patterns of the false-positive type for the above datasets. Such patterns appeared as stairs in 3-D. When we project the steps of a stair onto the search space, their projections do not intersect and totally cover the search space. The results contribute to filling a critical gap in the data mining literature. That is, the determination of stepwise patterns of classification errors can assist researchers in determining their applicable penalties. Finally, the website http://csc.lsu.edu/~huypham/HBA_CBA/index.html gives the HBA and CBA approaches along with some related computational tools.

This research can be extended in several ways. For instance, a new approach in the CBA may derive convex regions more accurately. One way may analyze the obtained results in terms of the theoretical model proposed in [Thomas, et. al., 1999]. Finally, one may wish to study a standard tool which is similar to the Receiver Operating Characteristic [Fawcett, 2006]. This standard tool may assess the accuracy of different classification approaches with the three types of error associated with different penalty costs. The world just begins from now.

# REFERENCES


[1] Abdi, H., "*A Neural Network Primer,*" Journal of Biological Systems, Vol. 2, 2003, pp. 247-281.

[2] Abonyi, J. and H. Szeifert, "*Supervised Fuzzy Clustering for the Identification of Fuzzy Classifiers,*" Pattern Recognition Letters, Vol. 24, 2003, pp. 2195-2207.

[3] Artificial Neural Network Toolbox 6.0 and Statistics Toolbox 6.0, Matlab Version 7.0, Website: www.mathworks.com/products/.

[4] Asuncion, A. and D. J. Newman, "*UCI-Machine Learning Repository,*" Website archive.ics.uci.edu/ml/, University of California, Irvine, School of Information and Computer Sciences, CA, USA, 2010.

[5] Au, W. H. and K. C. C. Chan, "*Classification with Degree of Membership: A Fuzzy Approach,*" Proceedings of the first IEEE International Conference on Data Mining, San Jose, CA, USA, 2001, pp. 35-42.

[6] Bennet, K. P. and J. A. Blue, "*A Support Vector Machine Approach to Decision Trees,*" Math Report, No. 97-100, Rensselaer Polytechnic Institute, Troy, NY, USA, 1997.

[7] Blachnik, M. and W. Duch, "*Prototype-Based Threshold Rules,*" LNCS of Neural Information Processing, Vol. 4234, Springer, Berlin, German, 2006, pp. 1028-1037.

[8] Breiman, L., "*Random Forests,*" Journal of Machine Learning, Vol. 45, No. 1, 2001, pp. 5-32.

[9] Breiman, L., "*Bagging Predictors,*" Journal of Machine Learning, Vol. 24, 1996, pp. 123-140.

[10] Breiman, L., J. H. Friedman, R. A. Olshen, and C. J. Stone, "*Classification and Regression Trees,*" Chapman & Hall/CRC Publisher, CA, USA, 1984, pp. 279-293.

[11] Cheung, N., "*Machine Learning Techniques for Medical Analysis,*" BSc thesis, School of Information Technology and Electrical Engineering, University of Queensland, Australia, 2001.

[12] Clark, P. and R. Boswell, "*Rule Induction with CN2: Some Recent Improvements,*" (Y. Kodratoff, editor), Machine Learning - EWSL-91, Springer, Berlin, Germany, 1991, pp. 151-163.

[13] Clark, P. and T. Niblett, "*The CN2 Algorithm,"* Journal of Machine Learning, Vol. 3, 1989, pp. 261-283.

[14] Cohen, S., L. Rokach, and O. Maimon, "*Decision-Tree Instance-Space Decomposition with Grouped Gain-Ratio,*" Journal of Information Science, Vol. 177, No. 17, 2007, pp. 3592-3612.

[15] Cohen, W. W., "*Fast Effective Rule Induction,*" Machine Learning: Proceedings of the twelfth International Conference, Tahoe City, CA, USA, 1995, pp. 115-123.

[16] Çomaka, E., K. Polatb, S. Güneşb, and A. Arslana, "*A New Medical Decision Making System: Least Square Support Vector Machine (LSSVM) with Fuzzy Weighting Pre-Processing*," Expert Systems with Applications, Vol. 32, No. 2, 2007, pp. 409-414.

[17] Cortes, C. and V. Vapnik, "*Support-Vector Networks,*" Journal of Machine Learning, Vol. 20, No. 3, 1995, pp. 273-297.

[18] Cover, T. M. and P. E. Hart, "*Nearest Neighbor Pattern Classification,*" Institute of Electrical and Electronics Engineers Transactions on Information Theory, Vol. 13, No. 1, 1967, pp. 21-27.

[19] Crammer, K. and Y. Singer, "*On the Learnability and Design of Output Codes for Multiclass Problems*," Machine Learning, Vol. 47, No. 2-3, Springer, The Netherlands, May 2002, pp. 201-233.

[20] Cristianini, N. and S. T. John, "*An Introduction to Support Vector Machines and other Kernel-Based Learning Methods,*" Cambridge University Press, UK, 2000.

[21] Dasarathy, B. V. and B. V. Sheela, "*A Composite Classifier System Design: Concepts and*

*Methodology,*" Proceedings of the IEEE, Vol. 67, No. 5, 1979, pp. 708-713.

[22] De Jong, K., "*Genetic Algorithms: a 30 Year Perspective*," Perspectives on Adaptation in Natural and Artificial Systems (Booker L., S. Forrest, M. Mitchell, and R. Riolo, Editors), Oxford University Press, New York, NY, USA, 2005.

[23] De Vaus, D., "*Analyzing Social Science Data: 50 Key Problems in Data Analysis*," Sage Publications Ltd., CA, USA, first edition, 2002.

[24] Dietterich, T. G. and G. Bakiri, "*Solving Multiclass Learning Problems via Error-Correcting Output Codes*," Journal of Artificial Intelligence Research, Vol. 2, 1995, pp. 263–286.

[25] Domm, M., A. Engel, P. P. Louis, and J. Goldberg, "*An Integer Support Vector Machine*," Proceedings of the sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing-2005, Towson, MD, USA, 2005, pp. 144-149.

[26] Duda, R. O. and P. E. Hart, "*Pattern Classification and Scene Analysis,*" Wiley-Interscience, New York, NY, USA, 1973, pp. 56-64.

[27] Dudani, S., "*The Distance-Weighted k-Nearest-Neighbor Rule,*" IEEE Transactions on Systems, Man, and Cybernetics, Vol. 6, No. 4, 1976, pp. 325-327.

[28] Eggermont, J., J. N. Kok, and W. A. Kosters, "*Genetic Programming for Data Classification: Partitioning the Search Space*," Proceedings of the 2004 Symposium on applied computing, 2004, pp. 1001-1005.

[29] Ene, M., "*Neural Network-Based Approach to Discriminate Healthy People from those with Parkinson's Disease*," Annals of the University of Craiova, Math. Comp. Sci. Ser., Vol. 35, 2008, pp. 112-116.

[30] Fawcett, T., "*An introduction to ROC analysis*," Pattern Recognition Letters, Vol. 27, 2006, pp. 861-874.

[31] Friedman, N., D. Geiger, and M. Goldszmidt, "*Bayesian Network Classifiers,*" Journal of Machine Learning, Vol. 29, 1997, pp. 131-161.

[32] Fung, G. and O. L. Mangasarian, "*Proximal Support Vector Machine Classifiers,*" Proceedings of the seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press, San Francisco, CA, USA, 2001, pp. 77-86.

[33] Gavrilis, D., L. G. Tsoulos, and E. Dermatas, "*Selecting and Constructing Features Using Grammatical Evolution*," Pattern Recognition Letters, Vol. 29, 2008, pp. 1358-1365.

[34] Geman, S., E. Bienenstock, and R. Doursat, "*Neural Networks and the Bias/Variance Dilemma,*" Journal of Neural Computation, Vol. 4, 1992, pp. 1-58.

[35] Goldberg, D. E., "*Genetic Algorithms in Search, Optimization, and Machine Learning*," Addison-Wesley Publishers, Boston, MA, USA, 1989.

[36] Gonzalez, J. A., L. B. Holder, and D. J. Cook, "*Graph-Based Concept Learning*," Proceedings of the fourteenth International FAIRS Conference, 2001, FL, USA, pp. 377-381.

[37] Greig-Smith, P., "*The Use of Random and Contiguous Quadrats in the Study of the Structure of Plant Communities,*" Ann. Bot. 16, 1952, pp. 293-316.

[38] Hamilton, H. J., N. Shan, and N. Cercone, "*RIAC: A Rule Induction Algorithm Based on Approximate Classification,*" Tech. Report, No. CS 96-06, University of Regina, Regina, Canada, 1996.

[39] Hecht-Nielsen, R., "*Theory of the Backpropagation Neural Network,*" International Joint Conference on Neural Networks, Washington, DC, USA, 1989, pp. 593-605.

[40] Hollmen, H., M. Skubacz, and M. Taniguchi, "*Input Dependent Misclassification Costs for Cost-Sensitive Classifiers*," Proceedings of the Second International Conference on Data Mining, Cambridge, UK, 2000, pp. 495-503.

[41] Hsu, C. C., Y. P. Huang, and K. W. Chang, "*Extended Naive Bayes Classifier for Mixed Data*," Expert Systems with Applications, Vol. 35, 2008, pp. 1080-1083.

[42] Huang, C. L., M. C. Chen, and C. J. Wang, "*Credit Scoring with a Data Mining Approach Based on Support Vector Machines*," Expert Systems with Applications, Vol. 33, No. 4, 2007, pp. 847-856.

[43] Ikizler, N. and H. A. Güvenir, "*Maximizing Benefit of Classifications Using Feature Intervals*," LNAI "*Knowledge-Based Intelligent Information and Engineering Systems*," Vol. 2773, Springer, Berlin, Germany, 2003, pp. 339-345.

[44] Jankowski, N. and V. Kadirkamanathan, "*Statistical Control of RBF-like Networks for Classification,*" Proceedings of the seventh International Conference on Artificial Neural Networks (ICANN), Lausanne, Switzerland, 1997, pp. 385-390.

[45] Karp, R. M., "*Reducibility Among Combinatorial Problems,*" Proceedings of Sympos. IBM Thomas J. Watson Res. Center, Yorktown Heights, NY, USA, 1972, pp. 85-103.

[46] Karypis, G., E. H. Han, and V. Kumar, "*CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling*," IEEE Computer, Vol. 32, No. 8, August 1999, pp. 68-75.

[47] Kecman, V. and T. Arthanari, "*Comparisons of QP and LP Based Learning From Empirical Data*," LNCS and LNAI (L. Monostori, J. V´ancza, and M. Ali, editors), Springer, New York, NY, USA, June 2001, pp. 326-332.

[48] Keller, J. M., M. R. Gray, and J. A. Givens Jr, "*A Fuzzy K-Nearest Neighbor Algorithm,*" Journal of IEEE Transactions on Systems, Man, and Cybernetics, Vol. 15, No. 4, 1985, pp. 580-585.

[49] Kianmehr, K., M. Alshalalfa, and R. Alhajj, "*Effectiveness of Fuzzy Discretization for Class Association Rule-Based Classification*," LNCS, Vol. 4994, Springer, Berlin, Germany, 2008, pp. 298-308.

[50] Kohavi, R. and J. H. George, "*Wrappers for Feature Subset Selection,*" Journal of Artificial Intelligence: Special Issue on Relevance, Vol. 97, No. 1-2, 1997, pp. 273-324.

[51] Kohavi, R., "*Scaling up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid,*" Proceedings of the second International Conference on Knowledge Discovery and Data Mining, Portland, OR, USA, 1996, pp. 202-207.

[52] Kononenko, I., "*Semi-Naïve Bayesian Classifier,*" (Y. Kodratoff Editor), Proceedings of the sixth European Working Session on Learning, Springer, Berlin, Germany, 1991, pp. 206-219.

[53] Kurgan, L. A., K. J. Cios, R. Tadeusiewicz, M. Ogiela, and L. S. Goodenday, "*Knowledge Discovery Approach to Automated Cardiac SPECT Diagnosis*," Artificial Intelligence in Medicine, Vol. 23, No. 2, 2001, pp 149-169.

[54] Langley, P. and S. Sage, "*Induction of Selective Bayesian Classifiers,*" Proceedings of UAI-94, Seattle, WA, USA, 1994, pp. 399-406.

[55] Lee, Y. J. and O. L. Mangasarian, "*RSVM: Reduced Support Vector Machines,*" Proceedings of the first SIAM International Conference on Data Mining, Chicago, IL, USA, 2001a.

[56] Lee, Y. J. and O.L. Mangasarian, "*SSVM: A Smooth Support Vector Machine for Classification,*" Computational Optimization and Applications, Vol. 20, No. 1, 2001b, pp. 5-22.

[57] Lei, G., W. Hui-Zhong, and X. Liang, "*A Novel Classification Algorithm Based on Fuzzy Kernel Multiple Hyperspheres,"* Proceedings of the fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007), Vol. 2, 2007, Haikou, Hainan, China, pp. 114-118.

[58] Leon, W. D. IV, "*Enhancing Pattern Classification with Relational Fuzzy Neural Networks and Square BK-Products*," PhD Dissertation in Computer Science, Florida State University, FL, USA, 2006, pp. 71-74.

[59] Little, M. A., P. E. McSharry, E. J. Hunter, J. Spielman, and L. O. Ramig, "*Suitability of Dysphonia Measurements for Telemonitoring of Parkinson's Disease*," IEEE Transactions on Biomedical Engineering, 2009.

[60] Mansour, Y. and D. McAllester, "*Generalization Bounds for Decision Trees,*" Proceedings of the thirteenth Annual Conference on Computer Learning Theory, San Francisco, CA, USA, 2000, pp. 69-80.

[61] Mastrogiannis, N., B. Boutsinas, and I. Giannikos, "*A Method for Improving the Accuracy of Data Mining Classification Algorithms*," Computers and Operations Research, Vol. 36, No. 10, 2009, pp. 2829-2839.

[62] Melnik, O., "*Decision Region Connectivity Analysis: A Method for Analyzing High-Dimensional Classifiers*," Machine Learning, Vol. 48, 2002, pp. 321-351.

[63] Michie, D., D. J. Spiegelhalter, and C. C. Taylor, "*Machine Learning, Neural and Statistical Classification*," Series Artificial Intelligence, Chapter 9, Prentice Hall, Englewood Cliffs, NJ, USA, 1994, pp. 157-160.

[64] Moody, J. E., "*The Effective Number of Parameters: An Analysis of Generalization and Regularization in Non-linear Learning Systems,*" Journal of Advances in Neural Information Processing Systems, Vol. 4, 1992, pp. 847-854.

[65] Moore, A. W., "*K-means and Hierarchical Clustering*," Carnegie Mellon University, Pittsburgh, PA, USA, online tutorial at the following URL: http://www.autonlab.org/tutorials/kmeans.html, 2010.

[66] Nakashima, T., G. Nakai, and H. Ishibuchi, "*Constructing Fuzzy Ensembles for Pattern Classification Problems,*" Proceedings of the International Conference on Systems, Man and Cybernetics, Washington, D.C., USA Vol. 4, October 2003, pp. 3200-3205.

[67] Nauck, D. and R. Kruse, "*Obtaining Interpretable Fuzzy Classification Rules from Medical Data,*" Artificial Intelligence in Medicine, Vol. 16, 1999, pp. 149-169.

[68] Özşen, S. and S. Güneş, "*Attribute Weighting via Genetic Algorithms for Attribute Weighted Artificial Immune System (AWAIS) and Its Application to Heart Disease and Liver Disorders Problems*," Expert Systems with Applications, Vol. 36, No. 1, 2009, pp. 386-392.

[69] Pazzani, M. J., "*Searching for Dependencies in Bayesian Classifiers,*" Proceedings of AI & STAT'95, 1995, pp. 239-248.

[70] Pena-Reyes, C. A. and M. Sipper, "*A Fuzzy-Genetic Approach to Breast Cancer Diagnosis,*" Artificial Intelligence in Medicine, Vol. 17, 1999, pp. 131-155.

[71] Pham, D. T., S. S. Dimov, and Z. Salem, "*Technique for Selecting Examples in Inductive Learning,*" Proceedings of the European symposium on intelligent techniques (ESIT 2000), Aachen, Germany, 2000, pp. 119-127.

[72] Pham, H. N. A. and E. Triantaphyllou, "*A Meta-Heuristic Approach for Improving the Accuracy in Some Classification Algorithms,*" Computers & Operations Research, Vol. 38, No. 1, 2010, pp. 174-189.

[73] Pham, H. N. A. and E. Triantaphyllou, "*An Application of a New Meta-Heuristic for Optimizing the Classification Accuracy When Analyzing Some Medical Datasets*," Expert Systems with Applications, Vol. 36, No. 5, 2009a, pp. 9240-9249.

[74] Pham, H. N. A. and E. Triantaphyllou, "*Prediction of Diabetes by Employing a New Data Mining Approach Which Balances Fitting and Generalization*," the Series Book "*Studies in Computation Intelligence,"* (Roger Yin Lee, Editor), Chapter 2, Vol. 131, Springer, Berlin, Germany, 2008b, pp. 11-26.

[75] Pham, H. N. A. and E. Triantaphyllou, "*The Impact of Overfitting and Overgeneralization on the Classification Accuracy in Data Mining,*" the book "*Soft Computing for Knowledge Discovery and Data Mining*" (O. Maimon and L. Rokach, Editors), Part 4, Chapter 5, Springer, New York, NY, USA, 2008a, pp. 391-431.

[76] Polat, K. A., S. Sahan, H. Kodaz, and S. Gunes, "*Breast Cancer and Liver Disorders Classification Using Artificial Immune Recognition System (AIRS) with Performance Evaluation by Fuzzy Resource Allocation Mechanism,*" Expert Systems with Applications, Vol. 32, 2007, pp. 172-183.

[77] Pujol, O., P. Radeva, and J. Vitrià, "*Discriminant ECOC: A Heuristic Method for Application Dependent Design of Error Correcting Output Codes*," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 28, No. 6, June 2006, pp. 1001-1007.

[78] Quinlan, J. R., "*Improved Use of Continuous Attributes in C4.5*," Artificial Intelligence Research, Vol. 4, 1996, pp. 77-90.

[79] Quinlan, J. R., "*C4.5: Programs for Machine Learning,*" Morgan Kaufmann Publisher, San Mateo, CA, USA, 1993, pp. 35-42.

[80] Quinlan, J. R., "*Simplifying Decision Trees,*" International Journal of Man-Machine Studies, Vol. 27, 1987, pp. 221-234.

[81] Ritter, J., "*An Efficient Bounding Sphere,*" Graphics Gems, San Diego, CA, USA, 1990, pp.301-303.

[82] Rokach, L., O. Maimon, and O. Arad, "*Improving Supervised Learning by Sample Decomposition,*" Journal of Computational Intelligence and Applications, Vol. 5, No. 1, 2005, pp. 37-54.

[83] Rutkowski, L. and K. Cpalka, "*Flexible Neuro-Fuzzy Systems,*" IEEE Transactions on Neural Networks, Vol. 14, 2003, pp. 554-574.

[84] Sakprasat, S. and M. C Sinclair, "*Classification Rule Mining for Automatic Credit Approval Using Genetic Programming*," Proceedings of the IEEE Congress on Evolutionary Computation, 2007, Singapore, pp. 548-555.

[85] Seber, G. A. F., "*Multivariate Observations*," Wiley-Interscience, New York, NY, USA, 1984.

[86] Segata, N. and E. Blanzieri, "*Empirical Assessment of Classification Accuracy of Local SVM*," Technical Report # DISI-08-014, University of Trento, Italy, March 2008.

[87] Seo, J. and B. Shneiderman, "*Interactively Exploring Hierarchical Clustering Results*," Computer, Vol. 35, No. 7, July 2002, pp. 80-86.

[88] Setiono, R., "*Generating Concise and Accurate Classification Rules for Breast Cancer Diagnosis,*" Artificial Intelligence in Medicine, Vol. 18, 2000, pp. 205-219.

[89] Shevked, Z. and L. Dakovski, "*Learning and Classification with Prime Implicants Applied To Medical Data Diagnosis,*" Proceedings of the 2007 International Conference on Computer Systems and Technologies, Rousse, Bulgaria, June 2007.

[90] Smith, J. W., J. E. Everhart, W. C. Dickson, W. C. Knowler, and R. S. Johannes, "*Using the ADAP Learning Algorithm to Forecast the Onset of Diabetes Mellitus,*" Proceedings of the twelfth Symposium on Computer Applications and Medical Care, Los Angeles, CA, USA, 1988, pp. 261-265.

[91] Smith, M., "*Neural Networks for Statistical Modeling,*" ITP New Media Publisher, ISBN 1-850-32842-0, Boston, MA, USA, 1996, pp. 117-129.

[92] Statlog Australia Credit Approval, Website: www.is.umk.pl/projects/datasets-stat.html#Australian, 8/2009.

[93] Statlog Heart, Website: www.is.umk.pl/projects/datasets-stat.html#Heart, August 2008.

[94] Ster, B. and A. Dobnikar, "*Neural Networks in Medical Diagnosis Comparison with Other*

*Methods,*" Proceedings of the International Conference on Engineering Applications of Neural Networks (EANN'96), London, UK, 1996, pp. 427-430.

[95] Tan, P. N., S. Michael, and K. Vipin, "*Introduction to Data Mining,*" Chapters 4 and 5, Addison-Wesley Publishers, Boston, MA, USA, 2005, pp. 145-315.

[96] Thomas, J. W. and J. P. Hofer, "*Accuracy of Risk-Adjusted Mortality Rate as a Measure of Hospital Quality of Care*," Medical Care, Vol. 37, No.1, 1999, pp. 83-92.

[97] Tichy, N., "*An Analysis of Clique Formation and Structure in Organizations,*" Administrative Science Quarterly, Vol. 18, No. 2, 1973, pp. 194-208.

[98] Tin, K. H. and M. K. Eugene, "*Building Projectable Classifiers of Arbitrary Complexity,*" Proceedings of the thirteenth International Conference on Pattern Recognition, Vienna, Austria, August 1996, pp. 880-885.

[99] Triantaphyllou, E. and G. Felici, "*Data Mining and Knowledge Discovery Approaches Based on Rule Induction Techniques,*" Springer, New York, NY, USA, Massive Computing Series, 2006, 796 pages.

[100] Triantaphyllou, E., "*Data Mining and Knowledge Discovery Via Logic-Based Methods: Theory, Algorithms, and Applications*," Springer, New York, NY, USA, Optimization and Its Applications Series, Vol. 43, 2010, 350 pages.

[101] Van, G. T., J. A. K. Suykens, G. Lanckriet, A. Lambrechts, B. De Moor, and J. Vandewalle, "*Bayesian Framework for Least Squares Support Vector Machine Classifiers, Gaussian Processes and Kernel Fisher Discriminant Analysis,*" Neural Computation, Vol. 14, 2002, pp. 1115-1147.

[102] Vapnik, V., "*Statistical Learning Theory,*" Wiley-Interscience, New York, NY, USA, 1998, pp. 375-567.

[103] Webb, G. I., "*Further Experimental Evidence Against the Utility of Occam's Razor,*" Journal of Artificial Intelligence Research, Vol. 4, 1996, pp. 397-417.

[104] Webster Dictionary, 2010, http://www.merriam-webster.com/dictionary/homogeneous

[105] Weigend, A., "*On Overfitting and the Effective Number of Hidden Units,*" Proceedings of the 1993 Connectionist Models Summer School, 1993, pp. 335-342.

[106] Weiss, S. M. and I. Kapouleas "*An Empirical Comparison of Pattern Recognition, Neural Nets, and Machine Learning Classification Methods,*" Proceedings of the eleventh International Joint Conference on Artificial Intelligence, Detroit, MI, USA, 1989, pp. 781-787.

[107] Zadrozny, B. and C. Elkan, "*Learning and Making Decisions When Costs and Probabilities Are Both Unknown*," Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 2001, pp. 204-213.

[108] Zhanga, Z., Y. Shib, and G. Gao, "*A Rough Set-Based Multiple Criteria Linear Programming Approach for the Medical Diagnosis and Prognosis*," Expert Systems with Applications, Vol. 36, No. 5, 2009, pp. 8932-8937.

[109] Zhou, Z. and C. Chen, "*Hybrid Decision Tree,*" Journal of Knowledge-Based Systems, Vol. 15, 2002, pp. 515-528.

# APPENDIX. PERMISSIONS

**1. Elsevier**

"When is clearance of rights not required?

- ……..
- Fair Use/Fair Dealing (varies by country)
    - Includes copying on a limited basis for purposes such as education and research, known as "fair use" in the US or "fair dealing" in the UK.
- ……."

Website: http://www.elsevier.com/wps/find/authorsview.authors/non-elsevier_permissions

# VITA

Huy Pham received his bachelor and master degrees from the University of Sciences in Viet Nam. Before being a doctoral student of the Department of Computer Science at Louisiana State University in 2004, he worked as a lecturer at the University of Sciences in Viet Nam and also as an IT director at Saigon Institute of Technology in Viet Nam.