Pham Nguyen, A.H., and E. Triantaphyllou, (2007), "*The Impact of Overfitting and Overgeneralization on the Classification Accuracy in Data Mining,*" in *Soft Computing for Knowledge Discovery and Data Mining*, (O. Maimon and L. Rokach, Editors), Springer, Part 4, Chapter 5, pp. 391-431.

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

Huy Nguyen Anh Pham and Evangelos Triantaphyllou*
Department of Computer Science
298 Coates Hall, Louisiana State University
Baton Rouge, LA 70803
Email: hpham15@lsu.edu and trianta@lsu.edu
Webpage: http://www.csc.lsu.edu/trianta

Many classification studies often times conclude with a summary table which presents performance results of applying various data mining approaches on different datasets. No single method outperforms all methods all the time. Furthermore, the performance of a classification method in terms of its false-positive and false-negative rates may be totally unpredictable. Attempts to minimize any of the previous two rates, may lead to an increase on the other rate. If the model allows for new data to be deemed as unclassifiable when there is not adequate information to classify them, then it is possible for the previous two error rates to be very low but, at the same time, the rate of having unclassifiable new examples to be very high. The root to the above critical problem is the overfitting and overgeneralization behaviors of a given classification approach when it is processing a particular dataset. Although the above situation is of fundamental importance to data mining, it has not been studied from a comprehensive point of view. Thus, this chapter analyzes the above issues in depth. It also proposes a new approach called the Homogeneity-Based Algorithm (or HBA) for optimally controlling the previous three error rates. This is done by first formulating an optimization problem. The key development in this chapter is based on a special way for analyzing the space of the training data and then partitioning it according to the data density of different regions of this space. Next, the classification task is pursued based on the previous partitioning of the training space. In this way, the previous three error rates can be controlled in a comprehensive manner. Some preliminary computational results seem to indicate that the proposed approach has a significant potential to fill in a critical gap in current data mining methodologies.

Keywords: Data mining, classification, prediction, overfitting, overgeneralization, false-positive, false-negative, homogenous set, homogeneity degree, optimization.

* Corresponding Author

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 large 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 good use of such massive datasets and trying to learn new knowledge of the system or phenomenon that created these data. 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 large datasets for the purpose of extracting new knowledge from the data. Such new knowledge can be derived in the form of 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 to the analyst. Such a system could be a 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.

For instance, a bank administrator could be interested in knowing whether a loan application should be approved or not based on some characteristics of applicants for credit. Here the two classes are: “approve” or “do not approve”. Attributes in this hypothetical scenario could be the age of the applicant, the income level of the applicant, the education level, whether he/she has a permanent job, etc. Then, the goal of the data mining process might be to extract any patterns that might be present in the data of successful credit applicants and also patterns that might be present in the data of non-successful applicants. By “successful applicants” we mean here those who can repay their

loans without any negative complications, while with "non-successful applicants" we mean those who default their loans.

There could be many questions to be asked, but only a few of them would be important for the decision. With the abundance of the data available in this area, a careful analysis could provide a *pattern* that exposes the main characteristics of reliable loan applicants. Then, the data mining analyst would like to identify such patterns from past data for which we know the final outcome and use 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 data for which we know the class value of each data point. We wish to infer some patterns from these data which in turn could be used to infer the class value of new points for which we do not know their class value. These patterns may be defined on the attributes used to describe the available data (also known as the *training data*). For instance, for the previous bank example the patterns may be defined on the level of education, years on the same job, 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 is 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.

There have been many theoretical and practical developments in the last two decades in this field. Most recent methods include the Statistical Learning Theory [Vapnik, 1998], Artificial Neural Networks (ANNs) [Hecht-Nielsen, 1989] and [Abdi, 2003], Decision Trees (DTs) [Quinlan, 1993], logic-based methods [Hammer and Boros, 1994], [Triantaphyllou, 1994; and 2007], and Support Vector Machines (SVMs) [Vapnik, 1979; and 1998] and [Cristianini and John, 2003].

In many real-life or experimental studies of data mining, some classification approaches work better with some datasets, while they work poorly with other datasets for no apparent reason. For instance, DTs had some success in the medical domain [Zavrsnik *et. al.*, 1995]. However, they also have had some certain limitations when they were used in this domain, as described for instance in [Kokol *et. al.*, 1998] and [Podgorelec, 2002]. Furthermore, the success of SVMs has been shown in bioinformatics, such as in [Byvatov, 2003] and [Huzefa *et. al.*, 2005]. At the same time, SVMs also did poorly in this field [Spizer *et. al.*, 2006]. If the data mining approach is accurate, then the people praise the mathematical model and claim that it is a good model. However, there is no good understanding of why such models are accurate or not. Their performance is often times coincidental.

A growing belief is that overfitting and overgeneralization problems may cause poor performance of the classification/class prediction model. Overfitting means that the extracted model describes the behavior of known data very well but does poorly on new data points. Overgeneralization occurs when the system uses the available data and then attempts to analyze vast amounts of data that has not seen yet.

Assume that there are two classes which, arbitrarily, we will call the positive and the negative class. Then, one may infer the following two classification models. The first model (which we will call the "positive" model) describes patterns embedded in the positive examples and which do not exist in the negative examples. In a similar manner, we define the "negative" model. For instance, when developing a diagnosis system for some types of cancer one may want to derive two models that can classify a new patient to either, positive (which means has cancer) or negative (which means does not have cancer) cases.

The analyst may want one of the previous two models to be more "conservative" while the other model to be more "liberal". If both models are ultra "conservative" then the implication is that they would only classify new cases that are very closely related to cases they already have seen in the training data. In this situation, the net effect would be many cases to be left as *unclassifiable* by both systems. Similarly, if both systems are classifying new data in a "liberal" manner, then they may contradict each other too often when they are presented with new cases. Again, this situation might be undesirable. Thus,

a “liberal” behavior by a classification model means that the model has a tendency for overgeneralization. A similar relationship exists between the concept of “conservative” and overfitting

This chapter aims at finding a way to balance both fitting and generalization in order to minimize the total misclassification cost of the final system. By doing so, it is hoped that the classification/prediction accuracy of the inferred models will be very high or at least as high as it can be achieved with the available training data. We plan to achieve this by balancing the previous two conflicting behaviors of the extracted systems.

The next section provides a preliminary description of the main research problem. The third section gives a summary of the main developments in the related literature. The proposed methodology is highlighted in the fourth section. That section shows how a balance between fitting and generalization has the potential to improve many existing classification algorithms. The fifth section discusses some promising preliminary results. These pilot results give an early indication of how this methodology may improve the performance of existing classification algorithms. Finally, this chapter ends with some conclusions.

2. Formal Problem Description

2.1 Some Basic Definitions

In order to help fix ideas, we first consider the hypothetical sample 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-D.

In general, a *data point* is a vector defined on *n* variables along with their values. In the above figure, *n* is equal to 2 and the two variables are indicated by the X and Y axis. Not all values may be known for a given data point. Data points describe the behavior of the system of interest to the analyst. For instance, in the earlier bank application a given data point may describe the level of education, years on the same job, level of income of a particular applicant, etc. The variables may be continuous, binary, or categorical, etc. All data are assumed to be deterministic and numeric at this point. 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.

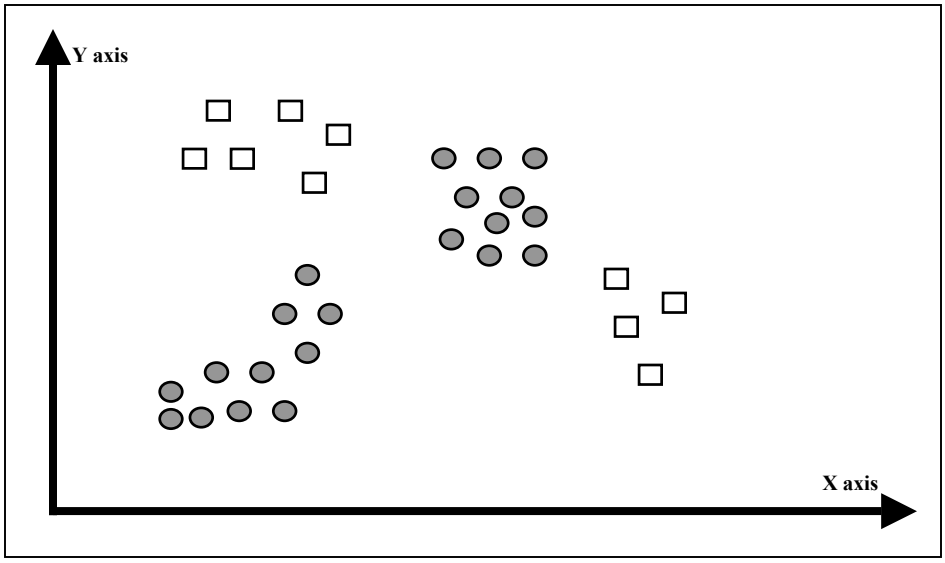


Figure 1: Sample data from two classes in 2-D.

We assume that there are only two classes. Arbitrarily, we will call one of them the *positive* class while the other 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* (examples) or the *classified examples*. The remaining of the data from the state space is called the *unclassified data* (examples).

2.2 Problem Description

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

In general, one classification system describes the positive data (and thus we will call it the *positive system*) while the other system describes the negative data (and thus we

will call it the *negative system*). In Figure 2.(a) the positive system corresponds to sets A and B (which define the positive pattern) while sets C and D correspond to the negative system (and thus they define the negative pattern).

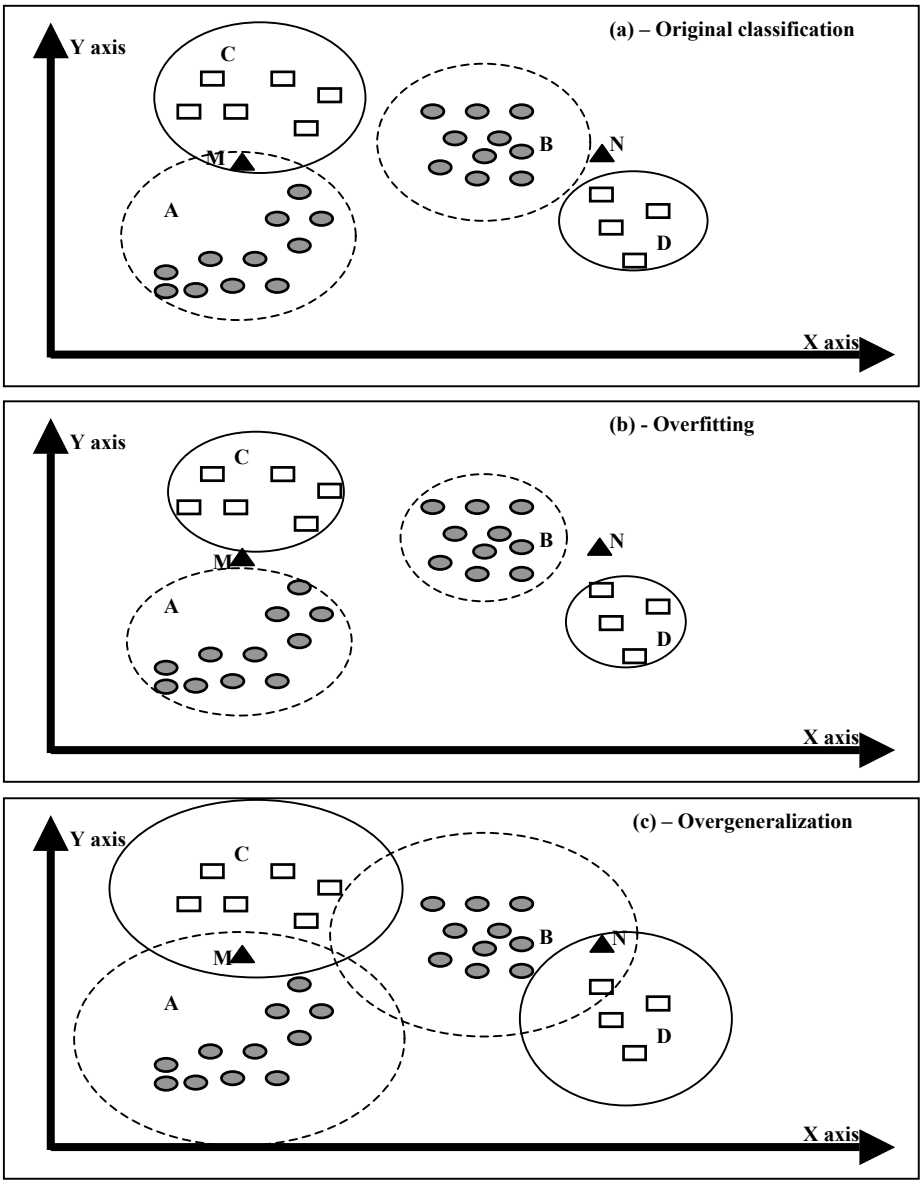


Figure 2: An illustrative example of overfitting and overgeneralization.

In many real-life applications, there are two different penalty costs 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-positive, while the second case is

known as false-negative. Furthermore, a closer examination of Figure 2.(a) indicates that there are some unclassifiable points which either are not covered by any of the patterns or are covered by patterns that belong to both classes. For instance, point N (indicated as a triangle) is not covered by any of the patterns, while point M (also a triangle) is covered by sets A and C which belong to the positive and the negative patterns, respectively.

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

Next consider Figure 2.(b). Suppose that all patterns 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 both points M and N are not covered by any of the inferred patterns. In other words, these points and several additional points which were classified before by the inferred systems now become unclassifiable.

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

On the other hand, suppose that the original patterns depicted as sets 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 patterns of both classes. Also, more points are now covered simultaneously by patterns of both classes. Thus, under this scenario we also have lots of

unclassifiable points because this scenario creates lots of cases of disagreement between the two classification systems (i.e., the positive and the negative systems). This realization means that the total penalty cost due to unclassifiable points will also be significantly higher than under the scenario depicted in Figure 2.(a). This scenario is known as *overgeneralization* of the training data.

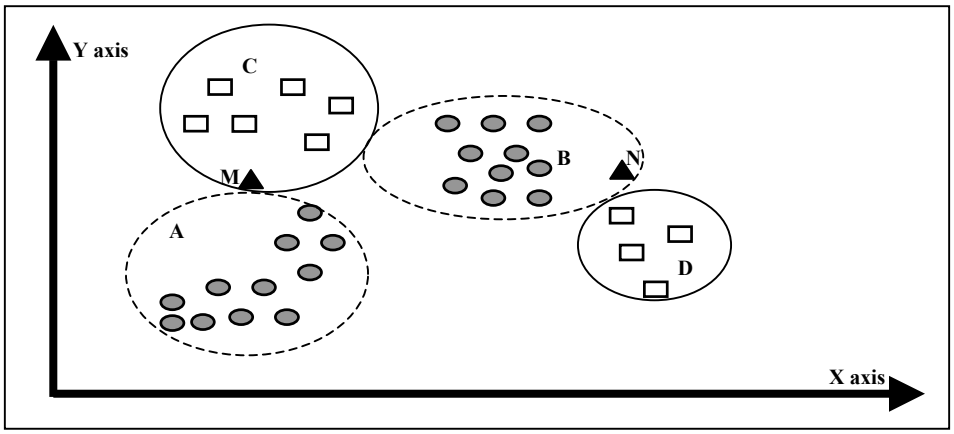


Figure 3: **An illustrative example of a better classification.**

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 systems (i.e., the positive and the negative systems) obtained from a classification algorithm. The balance of the two systems will target at minimizing the total misclassification costs of the final system.

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 *RATE_FP*, *RATE_FN*, and *RATE_UC* 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 denoted as *TC*. The problem is defined in the following expression:

$$TC = \min(C_{FP} \times RATE\_FP + C_{FN} \times RATE\_FN + C_{UC} \times RATE\_UC). \quad (1)$$

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

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 systems. For the sake of illustration, sets C and D are kept the same as in the original situation (i.e., as depicted in Figure 2.(a)) while set A has been reduced (i.e., it fits the training data more closely) and now it does not cover point M. On the other hand, set B is expanded (i.e., it generalizes the training data more) to cover point N. This new situation may correspond to a total misclassification cost that is smaller than those by any of the previous three scenarios. The following section will give a summary of the main developments in the related literature.

3. Literature Review

Most of the classification algorithms have focused on the minimization of the classification error of the training points. In this way, it is expected that new points will be classified with higher prediction accuracy. This section is a summary of the literature about ways that classification algorithms deal with the overfitting and the overgeneralization problems.

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 we trim some partitions of the tree.

There was recently a lot of effort which has focused on improving the pre-pruning methods. [Kohavi, 1996] proposed the NBTree (a hybrid of decision-tree 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 approach for building a DT by using a homogeneity criterion for splitting the space. However, the above approaches have a difficulty in choosing the threshold value for early termination. A value which is too high 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 some partitions of the tree. The reduction on the number of partitions makes the remaining tree more general. In order to help fix 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 ...*"

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 overgeneralization since pruning conditions are based on localized information.

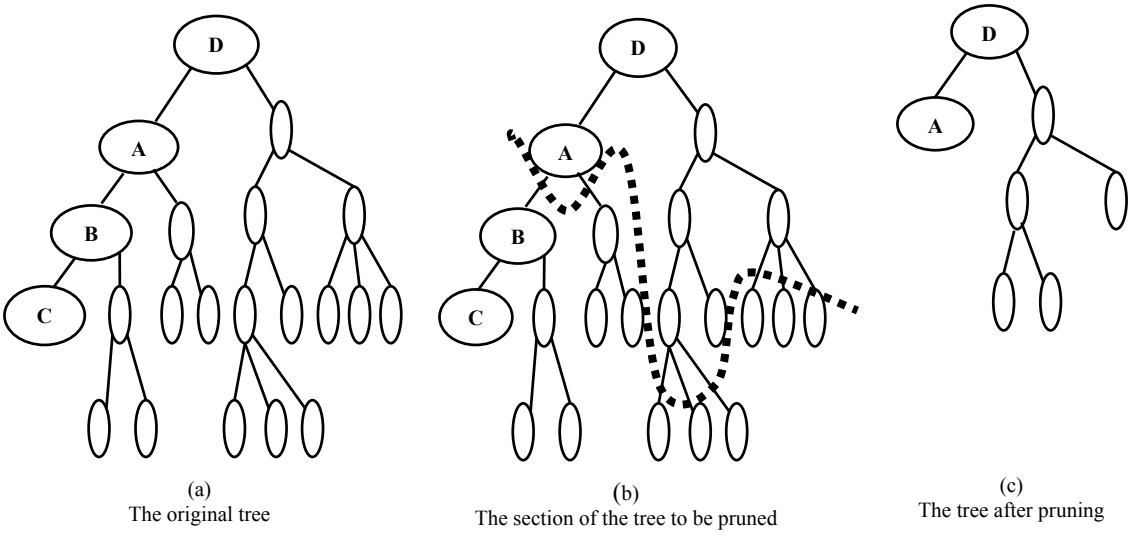


Figure 4: **An illustrative example of the DT pruning.**

Instead of the pruning methods, there have been some 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. This method is 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 between 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 dataset. There are two methods which infer classification rules: direct methods which infer classification rules directly from the data, and indirect methods which infer classification rules from other classification methods such as DTs, SVMs, or ANNs and then they 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, 2007].

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 two strategies for growing the classification rules: general-to-specific or specific-to-general. Then, the rules are refined by using 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 use a greedy approach to choose the new conjuncts to be added into the

rule antecedent part 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 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 when certain stopping criteria are met. A similar way exists for the negative points.

There are some related developments regarding these strategies. Such developments include a beam search approach [Clark and Boswell, 1991] which avoids the overgrowing of rules as result of the greedy behavior, the RIPPER algorithm [Cohen, 1995] which uses a rule induction algorithm. However, the use of the two strategies for growing classification rules has their drawbacks. The complexity for finding optimal rules is of exponential size of the search space. Although some rule pruning methods are used to improve their generalization error, they also leave drawbacks 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, 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 the overfitting or the overgeneralization problems [Tan *et. al.*, 2005]. One way to reduce the impact of *K* is to weight the influence of the nearest neighbors according to their distance to 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 has their drawbacks. Classifying a test example can be quite expensive since we need to compute a similarity degree between the test point and each training point. They are unstable since they are based on localized information only. Finally, it is difficult to find an appropriate value for *K* to avoid model overfitting or overgeneralization.

3.4 Bayes Classifiers

This approach uses 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 independent of each other and then they estimate by using the class conditional probability. This independence assumption, however, is obviously problematic because often times in many real applications there are strong conditional dependencies between the attributes. Furthermore, when using the independence assumption, NBs may suffer of overfitting since they are based on localized information.

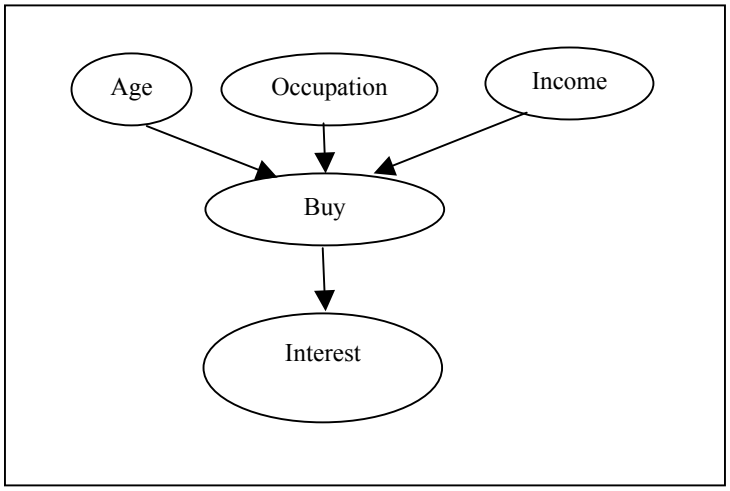


Figure 5: An illustrative example of a BBN [Rada, 2004].

Instead of requiring all attributes to be conditionally independent given a class, a BBN [Duda and Hart, 1973] allows only for pairs of attributes to be conditionally independent. We introduce this approach by discussing an illustrative example. Suppose that we have a training dataset consisting of the attributes: age, occupation, income, buy (i.e., buy some product X), and interest (i.e., "interest in purchasing insurance for this product"). 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).

There was a lot of effort which has focused on improving BBNs. This effort follows two general approaches: selecting a feature subset [Langley and Sage, 1994], [Pazzani, 1995], and [Kohavi and John, 1997] and relaxing the independence assumptions [Kononenko, 1991] and [Friedman *et. al.*, 1997]. However, these developments have the following drawbacks:

- They require a large amount of effort when constructing the network.
- They quietly degrade to overfitting 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 $Y = \text{sign}\sum_i (X_i w_i - t)$, 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.

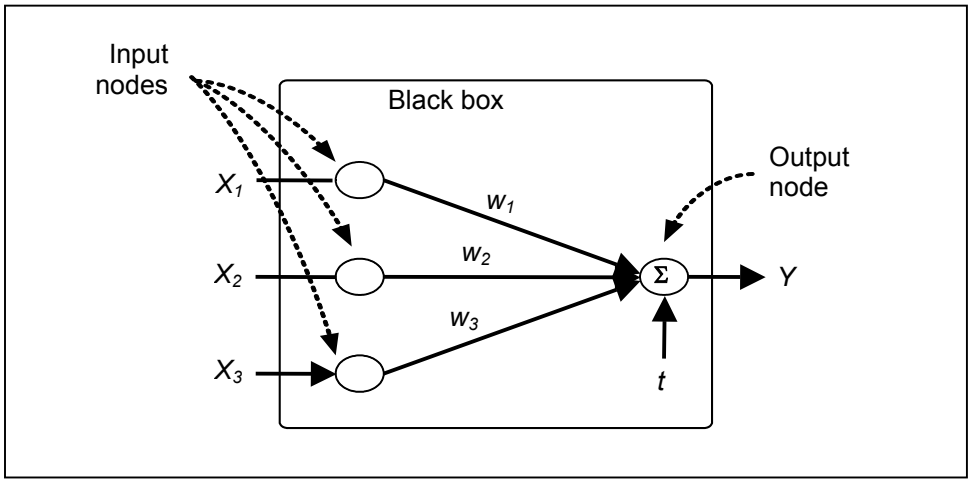


Figure 6: **An illustrative example of an ANN [Tan *et. al.*, 2005].**

In order to avoid overgeneralization or overfitting, 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 overgeneralization. On the other hand, a network that is too complex may not only fit the input but also the noisy points, thus leading to overfitting. According to [Geman, Bienenstock, and Doursat, 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 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 drawbacks:

- It is difficult to find an appropriate network topology for a given problem in order to avoid model overfitting and overgeneralization.
- It takes lots of 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 [Vapnik, 1995]. The basic idea behind SVMs is to find a maximal margin hyperplane, θ, that will separate points considered as vectors in an m-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.

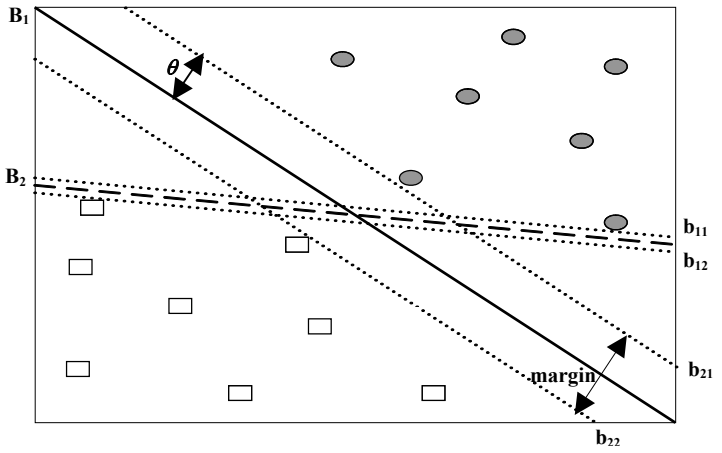


Figure 7: **An illustrative example of an SVM [Tan *et. al.*, 2005].**

In order to help fix ideas, we consider the simple illustrative 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 it is in the middle of the distance between 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 it may be the cause of too many misclassifications. Thus, the attempts for formulating the learning problem may lead to overgeneralization.

4. Proposed Methodology – The Homogeneity-Based Algorithm (HBA)

4.1 Some Key Observations

In order to help motivate the proposed methodology, we first consider the situation depicted in Figure 8.(a). This figure presents two inferred patterns. 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 (please recall that the circles in Figure 1 represent positive points). Moreover, in Figure 8.(a) there are two additional data points shown as small triangles and are denoted as points P and Q. At this situation, 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 data and inferred patterns to classify these two points. Because points P and Q are covered by patterns A and B, respectively, both of these points may be assumed to be positive examples.

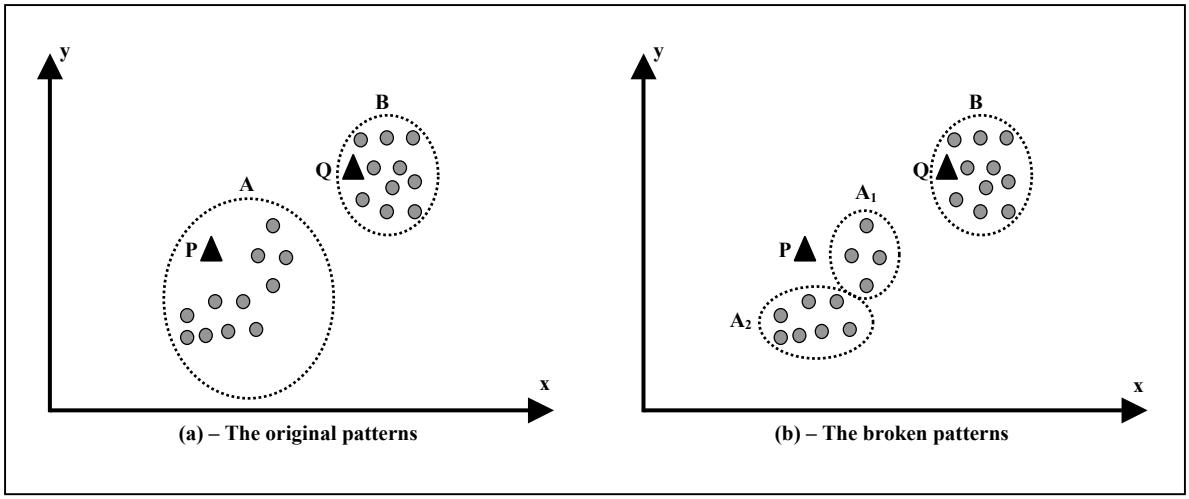


Figure 8: **Pattern B is a homogenous set while pattern A is a non homogenous set. Pattern A can be replaced by the two homogenous sets A_1 and A_2 as shown in part (b).**

Let us look more closely at pattern A. This pattern covers regions of the state space that are not adequately populated by positive training points. Such regions, for instance, exist in the upper left corner and the lower part of pattern A (see also Figure

8.(a)). It is possible that the unclassified points which belong to such regions are erroneously assumed to be of the same class as the positive training points covered by pattern A. Point P is in one of these sparely covered regions under pattern A. Thus, the assumption that point P is a positive point may not be very accurate.

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

The above simple observations lead one to surmise that the accuracy of the inferred systems can be increased if the derived patterns are, somehow, more compact and homogenous.

According to the Wikipedia Dictionary [2007], given a certain class (i.e., positive or negative), a homogenous set describes a steady or uniform distribution of a set of distinct points. That is, within the pattern there are no regions (also known as *bins*) with unequal concentrations of classifiable (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 homogenous set. An axiom and a theorem are derived from the definition of a homogenous set as follows:

Axiom 1: *Given is an inferred pattern C of size one. Then, C is a homogenous set.* This axiom is used later in Section 4.4.

Theorem 1: *Let us consider a homogenous set C. If C is divided into two parts, C_1 and C_2, then the two parts are also homogenous sets.*

Proof: We prove Theorem 1 by using contradiction. Since *C* is a homogenous set, there is a uniform random variable Z that represents the distribution of points in *C*. Similarly, Z_1 and Z_2 are the two random variables that represent the distribution of points in C_1 and C_2, respectively. Obviously, Z is the sum of Z_1 and Z_2. Assume that either Z_1 or Z_2 is a non homogenous set. Thus, $Z_1 + Z_2$ is not a uniform random variable. This contradicts the fact that Z is a uniform random variable□.

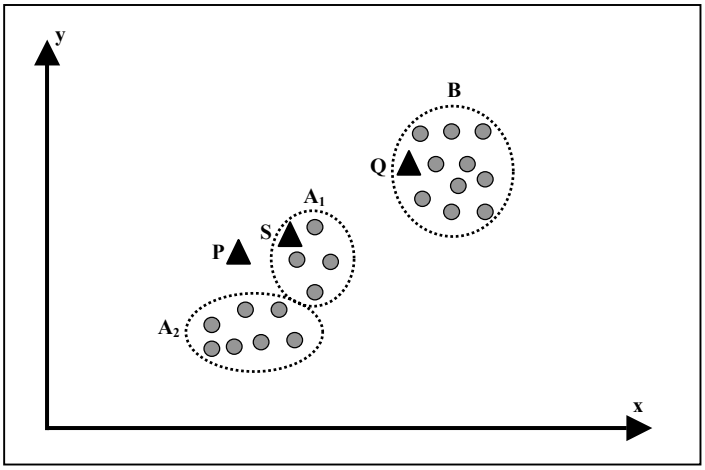


Figure 9: An illustrative example of homogenous sets.

The pattern which is represented by the non homogenous A can be replaced by two more homogenous sets denoted as A_1 and A_2 as in Figure 8.(b). Now the regions covered by the two new smaller patterns A_1 and A_2 are more homogenous than the area covered by the original pattern A. Given these considerations, 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 homogenous property of patterns may influence the number of misclassification cases of the inferred classification systems. Furthermore, if a pattern is a homogenous set, then the number of training points covered by this pattern may be another factor which affects the accuracy of the overall inferred systems. For instance, Figure 9 shows the case discussed in Figure 8.(b) (i.e., pattern A has been replaced by two more homogenous sets denoted as A_1 and A_2). Suppose that all patterns A_1, A_2 and B are homogenous sets and a new point S (indicated as a triangle) is covered by pattern A_1.

A closer examination of this figure shows that the number of points in B is higher than those in A_1. Although both points Q and S are covered by homogenous sets, the assumption that point Q is a positive point may be more accurate than the assumption that point S is a positive point. The above simple observation leads one to surmise that the accuracy of the inferred systems may also be affected by a *density* measure. Such a density could be defined as the number of points in each inferred pattern per unit of area or volume. Therefore, this density will be called the *homogeneity degree.*

In summary, a fundamental assumption here is as follows: if an unclassified point is covered by a pattern that is a homogenous set which also happens to have a high homogeneity degree, then it may be more accurately assumed to be of the same class as the points covered by that pattern. On the other hand, the accuracy of the inferred systems may be increased when their patterns are more homogenous and have high homogeneity degrees.

4.2 Non Parametric Density Estimation

Please recall that a pattern C of size n is a homogenous set if the pattern can be partitioned into smaller bins of the same unit size h and the density of these bins is almost equal to each other (or, equivalently, the standard deviation is small enough). In other words, if C is superimposed by a hypergrid of unit size h and the density of the bins inside C is almost equal to each other, then C is a homogenous set.

As seen in the above, the density estimation of a typical bin plays an important role in determining whether a set is a homogenous set or not. According to [Duda *et. al.*, 2001], the density estimation is the construction of an estimate, based on the observed data and on an unobservable underlying probability density function. There are two basic approaches to the density estimation:

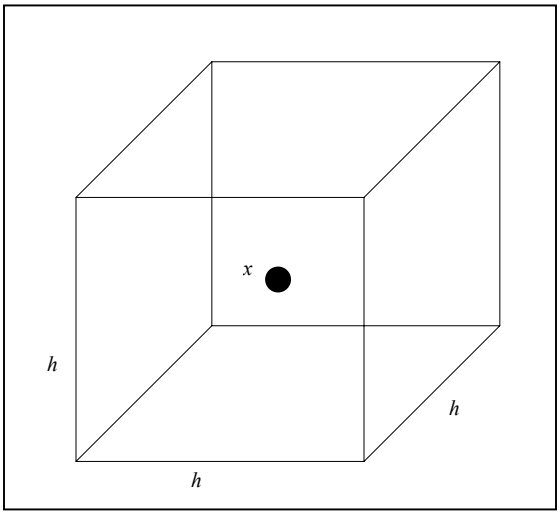


Figure 10: **A bin of unit size *h* and the center *x* in 3-D.**

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

The following sections will use the non parametric density estimation. That is, the approach divides pattern *C* into a number of small bins of unit size *h*. The density at the center *x* of each bin can be approximated by the fraction of points in *C* that fall into the corresponding bin and the volume of the bin. For instance, a bin in 3-D can be a cube of unit size *h* as depicted in Figure 10. Let *n* be the number of points in *C* and *d*(x) denote the *x*'s density, then:

$$d(x) = \frac{1}{n}[\frac{the\,number\,of\ \ examples\ \ falling\ in\ the\ bin\ with\ center\ x}{volume\ of\ \ the\ bin}]. \qquad (2)$$

The basic idea behind computing *d(x)* relies on the probability *p* that a data point *x*, drawn from a distribution function, will fall in bin *R*. By using this idea we arrive at the following obvious estimate for *d(x)*:

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

where *x* is a point within *R; k* is the number of points which fall in *R;* and *V* is the volume enclosed by *R*.

The Parzen Windows approach [Duda and Hart, 1973] was introduced as the most appropriate approach for the density estimation. That is, it temporarily assumes that the region *R* is a *D*-dimensional hypercube of unit size *h*. To find the number of points that fall within this region, the Parzen Windows approach defines a kernel function $\varphi(u)$ as follows:

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

It follows that the quantity $\varphi(\frac{x - x_i}{h})$ is equal to unity if the point x_i is inside the hypercube of unit size *h* and centered at *x*, and zero otherwise. Therefore, *k*, the number of points in the hypercube is given by:

$$k = \sum_{i=1}^{n} \varphi(\frac{x - x_i}{h}). \quad (5)$$

In the *D*-dimensional space, the kernel function can be presented as follows:

$$\varphi(\frac{x - x_i}{h}) = \prod_{m=1}^{D} \varphi(\frac{x^m - x_i^m}{h}) . \quad (6)$$

By using (6) in Equation (3), one gets:

$$d(x) \approx \frac{1}{n \times h^D} \sum_{i=1}^{n} \prod_{m=1}^{D} \varphi(\frac{x^m - x_i^m}{h}) . \quad (7)$$

Usually, but not always, $\varphi(u)$ will be radically symmetric. Thus, the unimodal probability density function, for instance the multivariate Gaussian density function, may be used to compute $\varphi(u)$:

$$\varphi(u) = \frac{1}{(2 \times \pi)^{\frac{D}{2}}} \exp(-\frac{1}{2} u^t u) . \quad (8)$$

Choosing a value for *h* plays the role of a smoothing parameter in the density estimation. That is, if $h \to \infty$, then the density at point *x* in *C*, *d(x)*, approaches a false density. As $h \to 0$, 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 pattern *C*. For easy illustration, assume that for pattern *C* which contains 5 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 illustrative example, we have set *S* equal to {2, 5} as both distances 2 and 5 occur with frequency equal to 3. By using the concept of the previous set *S*, Heuristic Rule 1 proposes a way for finding an appropriate value for *h* when we estimate the density *d(x)*. In particular, it uses the minimum value in *S* (which is equal to 2 in this illustration) as follows:

Heuristic Rule 1: *If h is set equal to the minimum value in set S and this value is used to compute d(x) by using Equation (7), then d(x) approaches to a true density.*

This heuristic rule is reasonable for the following reason. In practice, since pattern *C* 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 pattern C. If the value for h is the maximum distance, then C 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 C. This situation also leads to a false density.

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 simultaneously the bins would not degenerate to the single points in C.

4.3 The Proposed Approach

Recall that in optimizing the total misclassification cost as defined in Equation (1) for classification algorithms, one cannot separate the control of fitting and generalization into two independent studies. Instead of this, the key idea of the proposed methodology is to simultaneously balance both fitting and generalization by adjusting the inferred systems through the use of the concept of homogenous sets and the homogeneity degree. The proposed methodology can be summarized in terms of the following three phases:

- **Phase #1**: Apply a classification approach (such as a DT, ANN, or SVM) to infer the two classification systems (i.e., the positive and the negative classification systems). Suppose that each classification system consists of a set of patterns. Next, break the inferred patterns into hyperspheres.
- **Phase #2**: Determine whether the hyperspheres derived in Phase #1 are homogenous sets or not. If so, then go to Phase #3. Otherwise, break a non homogenous set into smaller hyperspheres. Repeat Phase #2 until all of the hyperspheres are homogenous sets.
- **Phase #3**: For each homogenous set, if its homogeneity degree is greater than a certain breaking threshold value, then expand it. Otherwise, break it into smaller homogenous sets. The approach stops when all of the homogenous sets have been processed.

Suppose that given is a homogenous set C. Let $HD(C)$ denote its homogeneity degree. There are five parameters which are used in the proposed methodology:

- Two expansion threshold values α^+ and α^- to be used for expanding the positive and the negative homogenous sets, respectively.
- Two breaking threshold values β^+ and β^- to be used for breaking the positive and the negative patterns, respectively.
- A density threshold value γ to be used for determining whether either a positive or a negative hypersphere is approximately a homogenous set or not.

Input:

- The training sets with the positive and the negative points.
- A given classification algorithm.
- Values of the control parameters α^+, α^-, β^+, β^-, and γ.

1. Call **Sub-Problem #1**. — Phase #1
2. Call **Sub-Problem #2**. — Phase #1
3. For each hypersphere C, do: — Phase #2
   Call **Sub-Problem #3** with inputs C and γ. — Phase #2
   If C is a non homogenous set, then call **Sub-Problem #4** and go to Step 3**.** — Phase #2
   End — Phase #2
4. Sort the homogeneity degrees in decreasing order.
5. For each homogenous set C, do: — Phase #3
   If $HD(C) \geq \beta^+$ (for positive sets) or $HD(C) \geq \beta^-$ (for negative sets), then — Phase #3
   Call **Sub-Problem #5** with inputs $HD(C)$ and α^+ or α^-. — Phase #3
   Else, Call **Sub-Problem #6.** — Phase #3
   End — Phase #3

Output: New positive and negative classification systems.

Figure 11: **The main algorithm.**

These three phases are also described in Figure 11 where they lead to the formulation of six sub-problems as follows:

- **Sub-Problem #1:** Apply a data mining approach (such as a DT, ANN, SVM) to infer the two classification systems.
- **Sub-Problem #2:** Break the inferred patterns into hyperspheres.

- **Sub-Problem #3:** Determine whether a hypersphere is a homogenous set or not. If so, then its homogeneity degree is estimated.
- **Sub-Problem #4:** If a hypersphere is not a homogenous set, then break it into smaller hyperspheres.
- **Sub-Problem #5:** Expand a homogenous set *C* by using the notion of its homogeneity degree *HD(C)* and the corresponding expansion threshold value plus some stopping conditions.
- **Sub-Problem #6:** Break a homogenous set *C* into smaller homogenous sets.

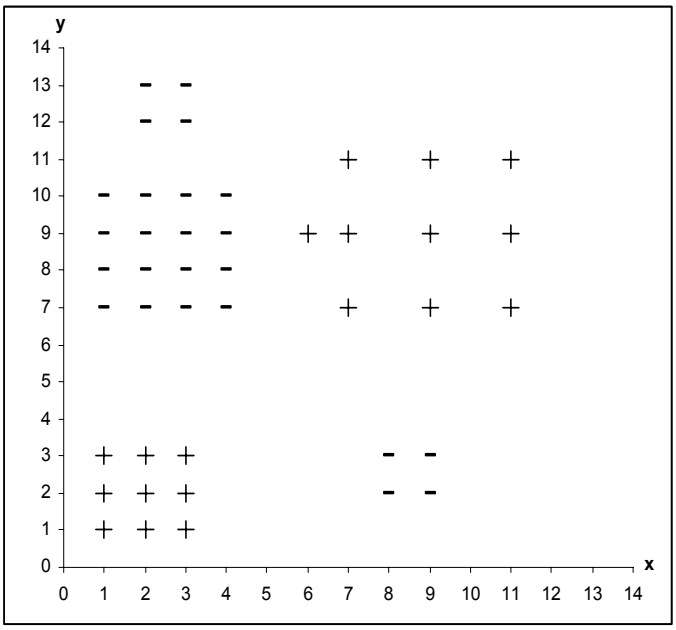
To solve Sub-Problem #1, one simply applies a classification algorithm and then derives the classification patterns. Furthermore, a solution to Sub-Problem #2 is similar to solutions for Sub-Problem #4. Therefore, the following sections present some procedures for solving Sub-Problems #2, #3, #5, and #6.

4.4 Solving Sub-Problem # 2

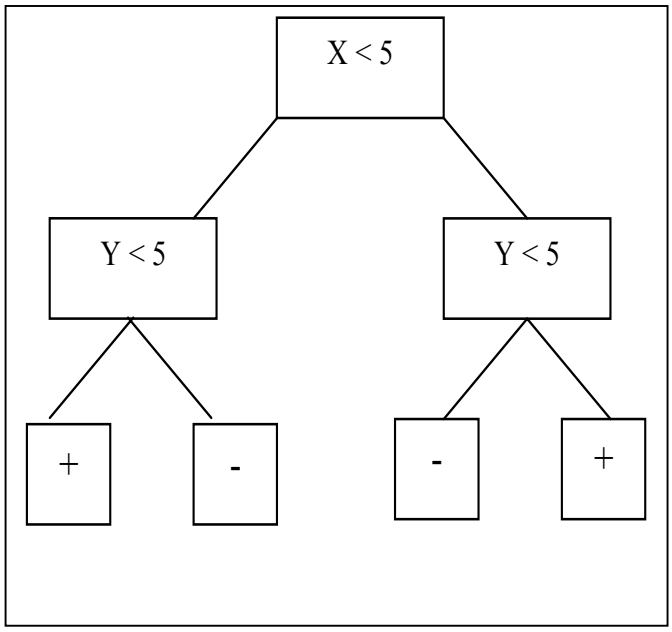
In order to help motivate the solution to Sub-Problem #2, we first consider the situation depicted in Figure 12.(a). This figure presents a set of positive training points and a set of negative training points in 2-D. Suppose that Sub-Problem #1 has applied a DT algorithm on these sample data to infer a decision tree as depicted in Figure 12.(b). This decision tree separates the training data into the four groups described by the two solid lines depicted in Figure 12.(c).

Next for each pattern somehow Sub-Problem #2 finds the minimum number of hyperspheres which cover all the points in the original patterns. For instance, the above situation is depicted in Figure 12.(d) in which the positive patterns and the bottom negative pattern are covered by the circles (please note that in 2-D hyperspheres are circles): B, D and C, respectively. The top negative pattern is covered by the two circles A and E.

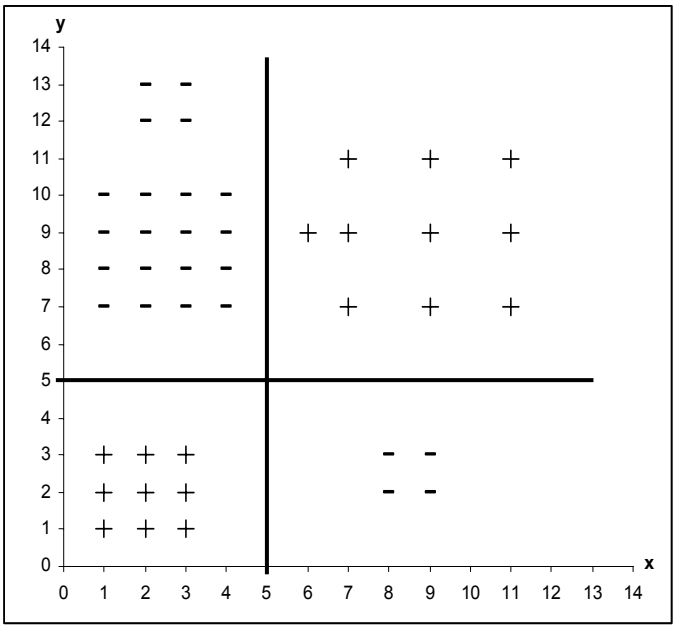
The problem of finding the minimum number of hyperspheres that can cover a pattern *C* of size *N* is similar to a form of the s*et cover problem,* an NP-complete problem [Karp, 1972]. In this research, a heuristic algorithm is proposed as depicted in Figure 13.



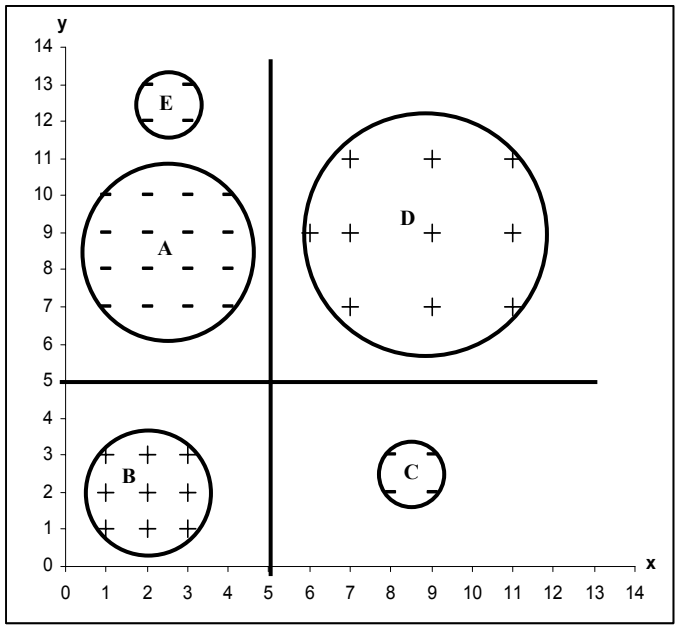
(a) – The original training data.



(a) – An inferred DT.



(c) – The inferred patterns from the DT.



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

Figure 12: **An Illustrative example of Phase #1.**

The algorithm starts by first estimating the densities of the N points by using Equation (7). Assume that the value for K is going from 1 to N. The algorithm will pick K points in C with the highest densities. Next, it uses these K points as centroids in the K-means clustering approach. If the K hyperspheres which are obtained from the clustering

approach cover *C*, then the algorithm will stop. Otherwise, we repeat the algorithm with the value for *K* increased by one. Obviously, the algorithm will stop after some iterations because of Axiom 1. For instance, in Figure 12.(d) the algorithm determines at least two circles which can cover the two positive patterns while it uses three circles for the two negative patterns.

Input:

- Pattern *C* of size *N*.

1. Estimate the densities of the *N* points by using Equation (7).
2. For *K*=1 to *N* do
   Pick *K* points in *C* with the highest densities.
   Use the *K*-means clustering approach to find *K* hyperspheres.
   If the *K* hyperspheres cover *C*, then STOP.
   Else, *K* = *K* + 1.
   End.

Output: *K* hyperspheres.

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

Recall that Sub-Problem #4 is to decompose a non homogenous set *C* into smaller hyperspheres in order to minimize the number of the hyperspheres which cover pattern *C*. We can use a similar algorithm as the one depicted in Figure 13.

4.5 Solving Sub-Problem #3

Let consider some hypersphere *C*. Sub-Problem #3 determines whether or not hypersphere *C* is a homogenous set. By using the idea of the non parametric density estimation described in Section 4.2, hypersphere *C* is divided into a number of small bins of unit size *h* and approximates the density at the center *x* of each bin. If the densities at the centers are approximately equal to each other, then *C* is a homogenous set.

In order to help motivate the algorithm for Sub-Problem #3, 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.

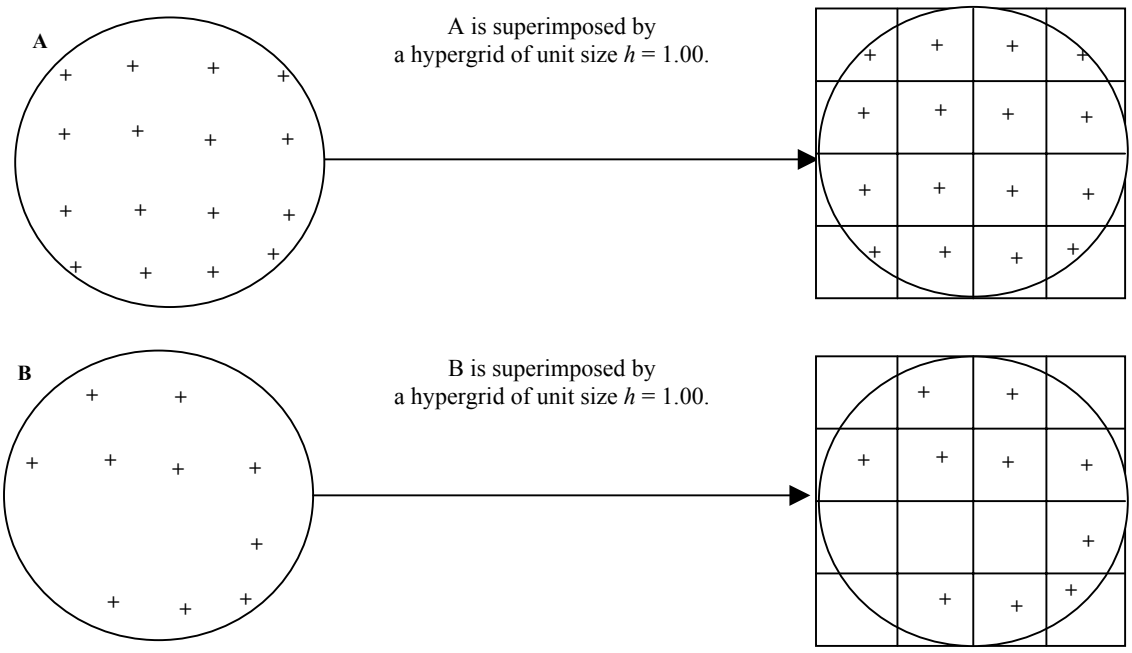


Figure 14: Illustrative examples of the homogenous set (at the top part) and the non homogenous set (at the bottom part).

Input: Hypersphere C and density threshold value γ.

1. Compute the distances between all pairs of points in C.
2. Let h be the distance mentioned in Heuristic Rule 1.
3. Superimpose C 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 γ, then

   C is a homogenous set and its homogeneity degree $HD(C)$ is computed by using Equation (9).

   Else, C is not a homogenous set.

Output: Decide whether or not hypersphere C is a homogenous set.

Figure 15: **The algorithm for Sub-Problem #3.**

Suppose that both circles A and B are superimposed by the same hypergrid V of unit size h equal to one. This situation is depicted in the right side of Figure 14. By using Equation (7), the right figures show 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 homogenous set while circle B is not.

Furthermore, 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 γ, say for $\gamma = 0.01$, then hypersphere *C* may be considered to be a homogenous set. The algorithm for Sub-Problem #3 is given in Figure 15.

As mentioned in Section 4.1, the homogeneity degree *HD(C)* is a factor that may affect the total misclassification cost of the inferred classification systems. If an unclassified point is covered by a homogenous set *C* which has a higher homogeneity degree, then it may more accurately be assumed to be of the same class as the points covered by the homogenous set *C*. Thus, a definition for *HD(C)* is an important step in improving the accuracy of the classification systems.

As discussed in Section 4.1, the concept of the homogeneity degree *HD(C)* is defined as the number of points inside the homogenous set *C* per unit of *C*'s volume. This definition, however, has its drawbacks. For instance, let us look at circles A and E as the one depicted in Figure 12. 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 pattern E is denser than pattern A. This is an apparent contradiction since in reality pattern A has more points and covers a wider region than pattern E. Thus, we need to find an appropriate definition for the homogeneity degree.

Intuitively, *HD(C)* depends on the value *h* defined in Heuristic Rule 1 and the number of points in *C*, denoted by n_C. If n_C increases, then *HD(C)* would slightly increase since the volume of *C* does not change and *C* has more points. Furthermore, if *h* increases, then the average distance between pairs of points in homogenous set *C* increases. Obviously, this leads to *HD(C)* decreases. Hence, *HD(C)* is inversely proportional to *h* while is directly proportional to n_C. We use the function $\ln(n_C)$ to show the slight effect of n_C to *HD(C)*.

$$HD(C) = \frac{\ln(n_C)}{h}. \quad (9)$$

For instance, *HD(*A) as depicted in Figure 14 is equal to $\frac{\ln(16)}{1} \approx 2.77$. Let us consider the illustrative example depicted in Figure 12. Now we have *HD(*A) equal to $\frac{\ln(16)}{1} \approx 2.77$, *HD(*B) equal to $\frac{\ln(9)}{1} \approx 2.19$, *HD(*C*) = HD(*E) equal to $\frac{\ln(4)}{1} \approx 1.38$, and *HD(*D) equal to $\frac{\ln(10)}{2} \approx 1.151$.

4.6 Solving Sub-Problem #5

Recall that the control of fitting and generalization for classification systems may be achieved by expanding or breaking the inferred homogenous sets by using their homogeneity degrees. Suppose that we are given a positive homogenous set *F* with its homogeneity degree *HD(F)*, the breaking threshold value β^+, and the expansion threshold value α^+. A similar definition exists for a negative homogenous set. According to the main algorithm depicted in Figure 11, if *HD(F)* is greater than or equal to β^+, then the homogenous set *F* will be expanded by using the expansion threshold value α^+. Otherwise, we will break the homogenous set *F* into smaller hyperspheres.

In order to help motivate this stage, we consider the example depicted in Figure 12. Please recall that the homogeneity degrees of circles A, B, C, D, and E are *HD(*A)=2.77, *HD(*B)=2.19, *HD*(C)=1.38, and *HD(*D)=1.15, and *HD(*E)=1.38, respectively. Suppose that the two breaking threshold values β^+ and β^- are equal to 1.00 and 1.50, respectively. Furthermore, let the two expansion threshold values α^+ and α^- be equal to 2.00. As depicted in Figure 16, the homogenous sets A, B, and D are expanded (the expanded regions are indicated by the solid line circles), while C and E are broken into four smaller circles (the broken regions are indicated by the small solid line circles). Please note that the breaking approach, i.e., Sub-Problem #6, is described in Section 4.7.

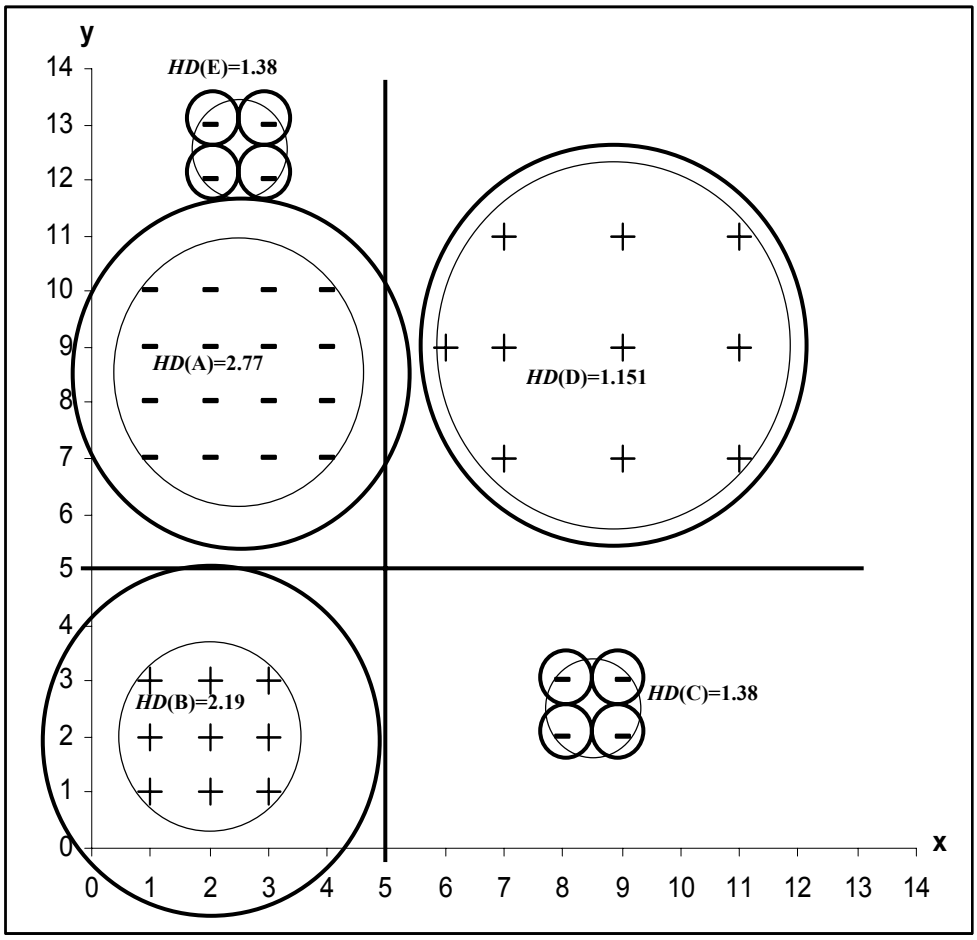


Figure 16: **An illustrative example of Sub-Problem #5.**

There are two types of expansion: a radial expansion in which a homogenous set *F* is expanded in all directions and a linear expansion in which a homogenous set *F* is expanded in a certain direction. For instance, in Figure 16 the homogenous sets A, B, and D have used the radial expansion approach. The following sections discuss in detail these two expansion types.

4.6.1 Radial Expansion

In the radial type, a homogenous set *F* is expanded in all directions. Let *M* be a region expanded from *F*. Let R_F and R_M denote the radiuses of *F* and *M*, respectively. In the radial expansion approach R_F is increased by a certain amount denoted as *T*, called a *step-size increase.* Thus, one gets:

$$R_M = R_F + T. \quad (10)$$

Following a dichotomous search methodology, we assume that there exists a hypersphere G which covers the homogenous set F. Furthermore, without loss of generality, let us assume that the radius R_G may be computed by:

$$R_G = 2 \times R_F. \tag{11}$$

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

$$T = \frac{R_G - R_F}{2}. \tag{12}$$

At the same time, T should depend on *HD(F*) because of the dichotomous search methodology. That is, if *HD(F)* gets higher, then T should get smaller. This means that *HD(F)* is inversely proportional to T. We may use a threshold value L to ensure that *HD(F*) is always greater than one. Thus, the value for T may be defined as follows:

$$T = \frac{R_G - R_F}{2} \times \frac{1}{L \times HD(F)}. \tag{13}$$

If we substitute back into Equation (10), R_M becomes:

$$R_M = R_F + \frac{R_G - R_F}{2} \times \frac{1}{L \times HD(F)}. \tag{14}$$

In order to help motivate the radial expansion algorithm, we consider the example indicated in Figure 17. This example uses the same hypothetical data as the ones depicted in Figure 12.(d). Assume that L is equal to one. A closer examination of Figure 17 indicates that the hypersphere A (i.e., the one-line circle with R_A=2.121) is covered by the three circles: a double-line circle which depicts circle G with $R_G = 2.121 \times 2 = 4.242$, a solid line circle which shows the final expanded region, and a dotted line circle which presents the hypersphere M whose radius is computed as follows:

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

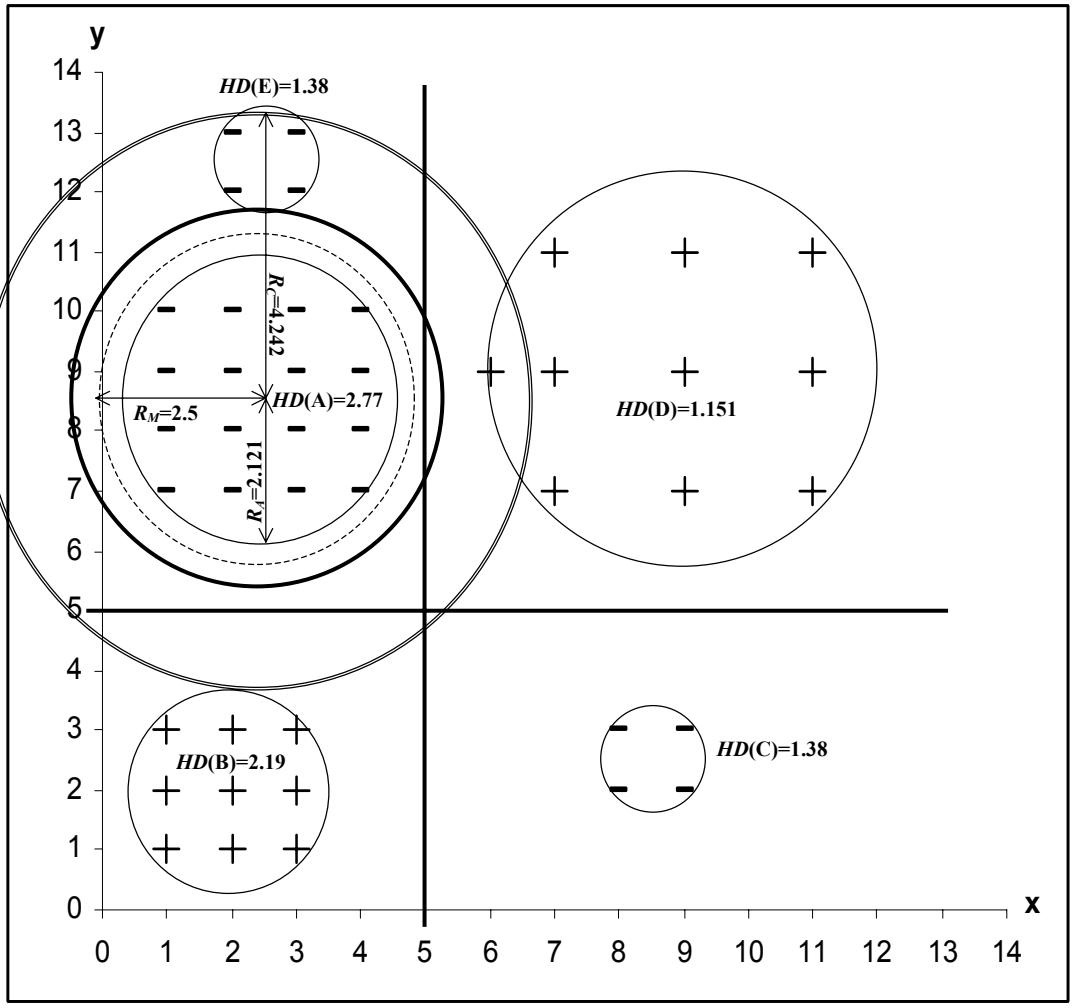


Figure 17: **An illustrative example of the radial expansion.**

Similarly, Equation (14) 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 next in Section 4.6.3. The final expanded region is the solid line circle depicted in Figure 17. Furthermore, this figure also shows that a part of the state space which has been inferred as a positive region by the DT algorithm. However, now it is derived as a negative region after using the HBA. This illustration indicates that the HBA may derive better classification systems. The radial expansion algorithm is depicted in Figure 18.

Input: Homogenous set F with $HD(F)$, R_F, and α^+

1. Set $M = F$ (i.e., $R_F = R_M$).
2. Set hypersphere G covering M with radius $R_G = 2 \times R_M$.
3. Repeat

   Set $E = M$ (i.e., $R_E = R_M$).

   Expand M by using Equation (14).

   Until (R_M satisfies stopping conditions discussed in Section 4.6.3 or $R_M = R_G$).
4. If R_M satisfies stopping conditions, then STOP.

   Else, go to Step 2.

Output: An expanded region E.

Figure 18: **The algorithm for the radial expansion.**

4.6.2 Linear Expansion

The linear approach expands a homogenous set F in a certain direction. There is a difference between the method presented in the previous section and the one presented in this section (i.e., linear vs. radial). That is, now the homogenous set F is first expanded to hypersphere M by using the radial expansion. Then, hypersphere M is expanded in a given direction by using the radial approach until it satisfies the stopping conditions mentioned next in Section 4.6.3. The final region is the union of all the expanded regions.

In order to help motivate the linear expansion approach, we consider the homogenous set A depicted in Figure 19. Suppose that by using the radial expansion for the homogenous set A with the expansion threshold value equal to 2.00, we get the hypersphere U (i.e., the two-line circle depicted in Figure 19.(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 solid lines depicted in Figure 19.(b). A similar approach exists for the Y axis depicted in Figure 19.(c). The final expanded region is the region which is defined by the union of the solid lines depicted in Figure 19.(d).

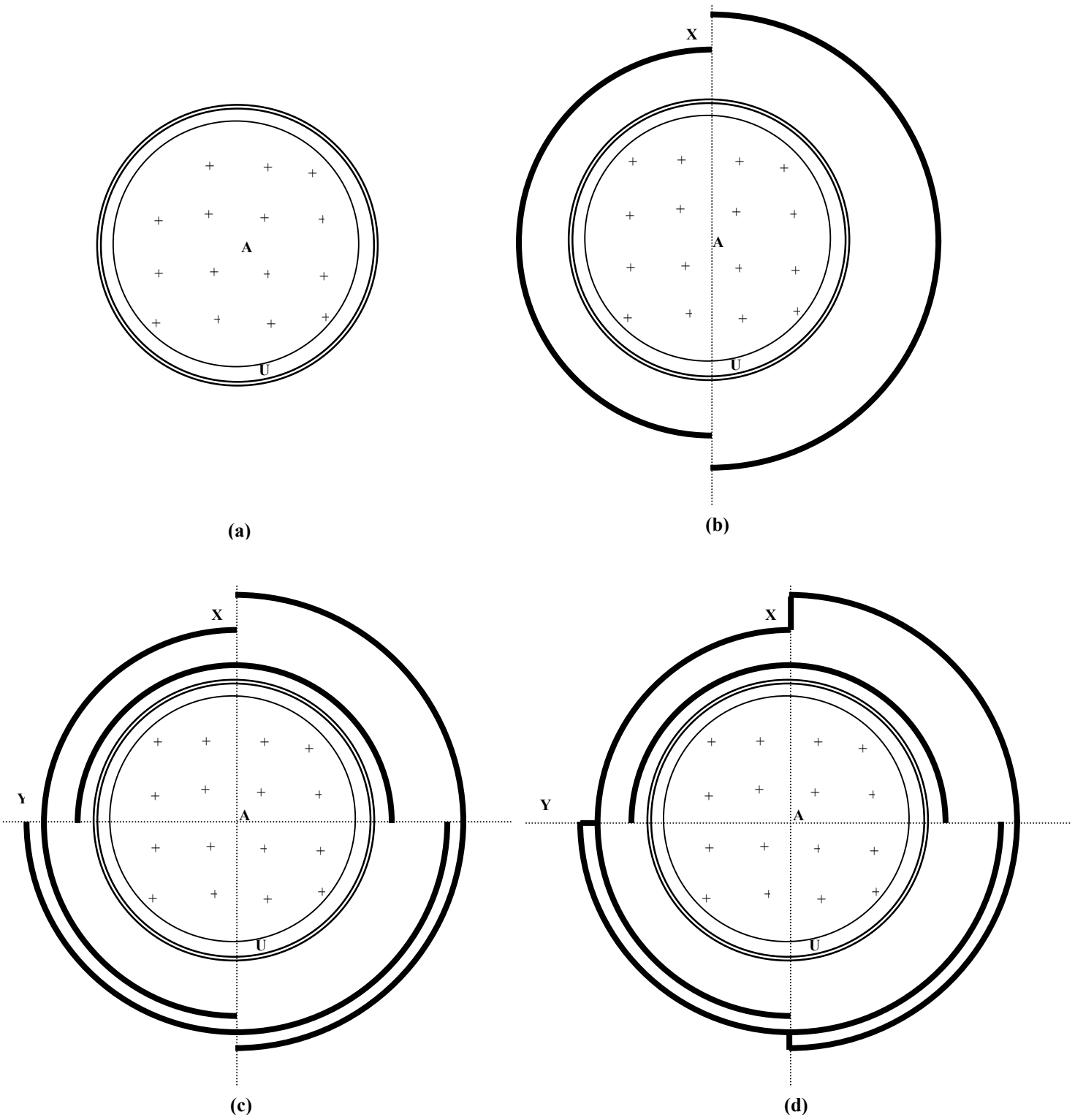


Figure 19: **An illustrative example of the linear expansion.**

4.6.3 Description of the Stopping Conditions

This section presents the stopping conditions for the radial expansion approach for expanding a homogenous set *F*. That is, the stopping conditions must satisfy the following requirements:

- Depend on the homogeneity degree. This has been mentioned in the fundamental assumption of the proposed approach.
- Stop when an expanded region reaches other patterns. We can use a softer condition in which the expanded region can accept several noisy data points. If the homogeneity degree is high, then the expanded region can accept more noisy data.

To address the first stopping condition, an upper bound for R_M should be directly proportional to the homogeneity degree *HD(F)*, the expansion threshold value α^+, and the original radius R_F. The second stopping condition can be determined while expanding. Furthermore, an upper bound on the number of noisy points should be directly proportional to *HD(F)* and the size of *F*, which is denoted as n_F. The stopping conditions are summarized as follows (a similar way exists for the expansion threshold value α^-):

$$R_M \le HD(F) \times R_F \times \alpha^+ \text{ and } \textit{the number of noisy points} \le \frac{HD(F) \times \alpha^+}{n_F}. \qquad (15)$$

4.7 Solving Sub-Problem #6

Suppose that given is a positive homogenous set *F*. Recall that if its homogeneity degree *HD(F)* is less than β^+, then the homogenous set *F* is broken into sub-patterns. According to Theorem 1, the sub-patterns are also homogenous sets. Thus, they can be expanded or broken down even more.

In order to help motivate this problem, we consider the example depicted in Figure 16. In this figure the two threshold values β^+ and β^- are equal to 1.00 and 1.50, respectively. Therefore, the homogenous sets C and E are broken down into four smaller circles for each set. Then, these smaller circles are considered to be homogenous sets with their homogeneity degrees equal to zero. Thus, they should not be expanded.

5. Some Computational Results

5.1 Datasets and Parametric Analysis

Please recall that this chapter aims at better understanding the performance of the HBA in balancing both fitting and generalization by adjusting the inferred systems through the use of the concept of homogenous sets and the homogeneity degree. The balance will target at minimizing the total misclassification costs, *TC*, of the final system:

$$TC = \min(C_{FP} \times RATE\_FP + C_{FN} \times RATE\_FN + C_{UC} \times RATE\_UC).$$

Please note that the penalty costs: C_{FP}, C_{FN}, and C_{UC} depend on each individual application. In the following experiments, we used some 2-D synthetic datasets which were divided into a training set and a testing set as described in Table 1.These data points were determined as follows. At first the map of VietNam was considered. Next some data points were generated randomly in 2-D. A data point would be a positive point, if it fell inside the map of VietNam. Otherwise, that point was defined as a negative point. The HBA attempted to use the training set to infer the map of VietNam (i.e., the positive and the negative systems). Then, we used the inferred map to test the testing set. The four parameters used in the HBA are as follows:

- Two expansion threshold values α^+ and α^- to be used for expanding the positive and the negative homogenous sets, respectively.
- Two breaking threshold values β^+ and β^- to be used for breaking the positive and the negative patterns, respectively.

Table 1: Characteristics of the 2-D synthetic datasets.

Name	Number of training points	Number of testing points
D_1	63	16
D_2	89	28
$D_3 = D_1 \cup D_2$	144	44

Furthermore, it was also assumed that β^+ and β^- were in [0, 2] while α^+ and α^- were in [0, 10]. Given is a certain 3-tuple of the penalty costs (C_{FP}, C_{FN}, C_{UC}). By using exhaustive search in the above ranges the HBA found the optimal combinations of α^+, α^-,

β^+, and β^- in order to minimize the *TC* value. On the other hand, given are different values for the 3-tuple (C_{FP}, C_{FN}, C_{UC}). We expect that the value for *TC* after controlling the fitting and generalization problems would be less than or at most equal to what was achieved by the original algorithms.

5.2 Experimental Results

The experiments were ran on a PC with 2.8GHZ speed and 1GB RAM under the Windows XP operating system. The original classification algorithms used in these experiments are based on SVMs, ANNs, and DTs. There were thirteen experiments done on the three datasets D_1, D_2, and D_3 with different values for the 3-tuple (C_{FP}, C_{FN}, C_{UC}). Furthermore, we used the libraries in Neural Network Toolbox 6.0 and Statistics Toolbox 6.0 [Matlab, 2004] for implementing the classification algorithms, the *K*-means clustering algorithm, and the density estimation approach. The experimental details are as follows:

Case 1: At first we studied the case of a 3-tuple (C_{FP}, C_{FN}, C_{UC}) in which the application would penalize much more for the false-positive cases than for the other types of error. Thus, the objective function in this case was assumed to be:

$$TC = 6 \times RATE\_FP + 3 \times RATE\_FN + RATE\_UC.$$

Table 2: Results for minimizing $TC = 6 \times RATE\_FP + 3 \times RATE\_FN + RATE\_UC$ on D_1.

Algorithm	*RATE_FP*	*RATE_FN*	*RATE_UC*	*TC*
SVM	1	0	7	13
DT	3	0	5	23
ANN	1	0	7	13
SVM-HBA	0	1	5	8
DT-HBA	0	1	5	8
ANN-HBA	0	1	5	8

Next, we ran the HBA on D_1 with β^+ and β^- divided into {0, 1, 2} and α^+ and α^- divided into {0, 2, 4, 6, 8, 10}. Recall that *RATE_FP*, *RATE_FN*, and *RATE_UC* are the false-positive, the false-negative, and the unclassifiable rates, respectively. Table 2 shows these three rates and the value of *TC* obtained from the algorithms. The notation "SVM-HBA" means that the HBA used the classification models first obtained by using the SVM algorithm before controlling the fitting and generalization problems. The two

similar notations exist for DT-HBA (the Decision Tree algorithm and the HBA) and ANN-HBA (the Artificial Neural Network algorithm and the HBA). Table 2 presents that SVM-HBA, DT-HBA, and ANN-HBA found the optimal *TC* to be equal to 8. This value was less than the value of *TC* achieved by the original algorithms (i.e., the SVM, DT, and ANN) by about 39%.

Table 3 presents information for the four specific parameter values when SVM-HBA found the optimal *TC*. The execution time in this case was approximately equal to 1 hour and 3 minutes.

Table 3: Values for the four parameters when the SVM-HBA ran on D_1 and found the optimal *TC*.

β^+	α^-	β^-	α^+
1	10	2	4
1	10	2	6
1	10	2	8
1	10	2	10

An even lower *TC* was found once we divided β^+ and β^- into {0, 1, 2} and divided α^+ and α^- into {0 to10}. These results are presented in Table 4.

Table 4: Results for minimizing $TC = 6 \times RATE\_FP + 3 \times RATE\_FN + RATE\_UC$ on D_1 with the smaller ranges.

Algorithm	*RATE_FP*	*RATE_FN*	*RATE_UC*	*TC*
SVM	1	0	7	13
DT	3	0	5	23
ANN	1	0	7	13
SVM-HBA	0	0	7	7
DT-HBA	0	0	7	7
ANN-HBA	0	0	7	7

Table 4 shows that if we split the four parameters into smaller ranges, then the HBA could find a lower *TC*. This may lead to a new strategy in which one can develop an approach for determining optimal combinations of the four parameter values by successively considering higher resolution.

Case 2: Now we consider a case in which the application would penalize much more for the unclassifiable cases than for the other types of error. Thus, the objective function in this case was assumed to be:

$$TC = RATE\_FP + 3 \times RATE\_FN + 6 \times RATE\_UC.$$

We ran the HBA on D_1 with β^+ and β^- divided into {0, 1, 2} and α^+ and α^- divided into {0 to 10}. Table 5 shows that SVM-HBA, DT-HBA, and ANN-HBA found an optimal *TC* which was less than the value of *TC* achieved by the original algorithms by about 53%.

Table 5: Results for minimizing $TC = RATE\_FP + 3 \times RATE\_FN + 6 \times RATE\_UC$ on D_1.

Algorithm	*RATE_FP*	*RATE_FN*	*RATE_UC*	*TC*
SVM	1	0	7	43
DT	3	0	5	33
ANN	1	0	7	43

Case 3: Now we consider a case in which the application would penalize the same way for the false-positive, the false-negative, and the unclassifiable cases. Thus, the objective function in this case was assumed to be:

$$TC = 3.3 \times RATE\_FP + 3.3 \times RATE\_FN + 3.3 \times RATE\_UC.$$

Table 6: Results for minimizing $TC = 3.3 \times RATE\_FP + 3.3 \times RATE\_FN + 3.3 \times RATE\_UC$ on D_1.

Algorithm	*RATE_FP*	*RATE_FN*	*RATE_UC*	*TC*
SVM	1	0	7	26.4
DT	3	0	5	26.4
ANN	1	0	7	26.4
SVM-HBA	0	1	5	19.8
	1	1	4	19.8
DT-HBA	1	1	3	16.5
	2	1	2	16.5
ANN-HBA	0	1	5	19.8
	1	1	4	19.8

We ran the HBA on D_1 with β^+ and β^- divided into {0, 1, 2} and α^+ and α^- divided into {0, 2, 4, 6, 8, 10}. Table 6 shows that SVM-HBA, DT-HBA, and ANN-HBA found two possible cases for each algorithm where the optimal value for *TC* was less than the value of *TC* achieved by the original algorithms by about 33%.

A similar result for *TC* once we ran the HBA on D_3, which had more training points, also divided β^+ and β^- into {0, 1, 2}, and α^+ and α^- into {0, 2, 4, 6, 8, 10}. These results are presented in Table 7.

Table 7: Results for minimizing $TC = 3.3 \times RATE\_FP + 3.3 \times RATE\_FN + 3.3 RATE\_UC$ on D_3.

Algorithm	*RATE_FP*	*RATE_FN*	*RATE_UC*	*TC*
SVM	5	3	26	112.2
DT	8	3	24	115.5
ANN	5	2	27	112.2
SVM-HBA	4	7	7	59.40
DT-HBA	7	7	9	75.90
ANN-HBA	4	7	7	59.40

Table 7 shows that SVM-HBA, DT-HBA, and ANN-HBA found the optimal *TC* which was less than the value of *TC* achieved by the original algorithms by about 47%. The execution time in this case was approximately equal to 12 hours and 50 minutes.

Case 4: Now we consider a case in which the application would penalize much more for the false-negative cases than for the other types of error. Furthermore, the penalty cost for unclassifiable cases was equal to zero Thus, the objective function in this case was assumed to be:

$$TC = 2 \times RATE\_FP + 20 \times RATE\_FN + 0 \times RATE\_UC.$$

We ran the HBA on D_2 with β^+ and β^- divided into {0, 1, 2} and α^+ and α^- divided into {0, 2, 4, 6, 8, 10}. Table 8 shows that SVM-HBA, DT-HBA, and ANN-HBA found an optimal *TC* equal to 0. This value was equal to the value of *TC* achieved by the original algorithms. However, SVM-HBA achieved an unclassifiable rate of 21 versus 28 for the original algorithms. A similar result existed for DT-HBA and ANN-HBA. The execution time in this case was approximately equal to 5 hours and 24 minutes.

Table 8: Results for minimizing $TC = 2\times RATE\_FP + 20\times RATE\_FN$ on D_2.

Algorithm	*RATE_FP*	*RATE_FN*	*RATE_UC*	*TC*
SVM	0	0	28	0
DT	0	0	28	0
ANN	0	0	28	0
SVM-HBA	0	0	21	0
DT-HBA	0	0	24	0
ANN-HBA	0	0	22	0

We also experimented with the following different objective functions on the dataset D_1:

$$TC = 6\times RATE\_FP + 2\times RATE\_FN + 2\times RATE\_UC,$$

$$TC = 4\times RATE\_FP + 2\times RATE\_FN + 4\times RATE\_UC, \text{ and}$$

$$TC = 3\times RATE\_FP + 6\times RATE\_FN + 1\times RATE\_UC.$$

Similarly, we experimented with the following different objective functions on the dataset D_2:

$$TC = 2\times RATE\_FP + 20\times RATE\_FN + 0\times RATE\_UC,$$

$$TC = 6\times RATE\_FP + 3\times RATE\_FN + 1\times RATE\_UC, \text{ and}$$

$$TC = 50\times RATE\_FP + 60\times RATE\_FN + 1\times RATE\_UC.$$

We also experimented with the following different objective functions on the dataset D_3:

$$TC = RATE\_FP + 3\times RATE\_FN + 6\times RATE\_UC, \text{ and}$$

$$TC = 20\times RATE\_FP + 2\times RATE\_FN + 0\times RATE\_UC.$$

In all these tests we concluded that the HBA always found the optimal combinations of α^+, α^-, β^+, and β^- in order to minimize the value of *TC*. Furthermore, the value for *TC* in all these cases was significantly less than or at most equal to what was achieved by the original algorithms.

6. Conclusions

The performance of a classification method in terms of the false-positive, the false-negative, and the unclassifiable rates may be totally unpredictable and depend on the application at hand. Attempts to minimize one of the previous rates, lead to increases on the other two rates. The root to the above critical problems is the overfitting and

overgeneralization behaviors of a given classification approach when it is processing a particular dataset. This chapter identified a gap between fitting and generalization with current algorithms and also defined the desired goal as an optimization problem. Next, it provided a new approach, called the Homogeneity-Based Algorithm (HBA), which appears to be very promising. There are some future research goals. For example, the HBA needs to be tested with higher dimensions and more data. This is ongoing research by our group. Currently we are implementing a GA (Genetic Algorithm) for finding the optimal values of the controlling threshold values α^+, α^-, β^+, and β^-. Some preliminary results seem to suggest that by using the GA one can achieve even better values for the various objectives functions at a fraction of the original CPU time (often times by spending between 50% to 80%).

REFERENCES

1. Abdi, H., (2003), "*A neural network primer,*" Journal of Biological Systems, vol. 2, pp. 247-281.
2. Ali, K., C. Brunk, and M. Pazzani, (1994), "*On learning multiple descriptions of a concept,*" Proceedings of Tools with Artificial Intelligence, New Orleans, LA, USA, pp. 476-483.
3. Artificial Neural Network Toolbox 6.0 and Statistics Toolbox 6.0, Matlab Version 7.0, website: http://www.mathworks.com/products/
4. Boros, E., P. L. Hammer, and J. N. Hooker, (1994), "*Predicting Cause-Effect Relationships from Incomplete Discrete Observations,*" Journal on Discrete Mathematics, vol. 7, no. 4, pp. 531-543.
5. Bracewell, R., (1999), "*The Impulse Symbol,*" Chapter 5 in The Fourier Transform and Its Applications, 3rd ed. New York: McGraw-Hill, pp. 69-97.
6. Breiman, L., (1996), "*Bagging predictors,*" Journal of Machine Learning, vol. 24, pp. 123-140.
7. Breiman, L., (2001), "*Random forests,*" Journal of Machine Learning, vol. 45, no. 1, pp. 5–32.
8. Breiman, L., J. H. Friedman, R. A. Olshen, and C. J. Stone, (1984), "*Classification and Regression Trees,*" Chapman & Hall/CRC Publisher, pp. 279-293.
9. Byvatov, E., and G. Schneider, (2003), "*Support vector machine applications in bioinformatics,*" Journal of Application Bioinformatics, vol. 2, no.2, pp. 67-77.
10. Clark, P., and R. Boswell, (1991), "*Rule induction with CN2: Some recent improvements,*" Y. Kodratoff, editor, Machine Learning - EWSL-91, Berlin, Springer-Verlag, pp. 151-163.
11. Clark, P., and T. Niblett, (1989), "*The CN2 Algorithm,"* Journal of Machine Learning, vol. 3, pp. 261-283.
12. Cohen S., L. Rokach, O. Maimon, (2007), "*Decision-tree instance-space decomposition with grouped gain-ratio,*" to appear in Journal of Information Science.
13. Cohen, W. W., (1995), "*Fast effective rule induction,*" Machine Learning:

Proceedings of the Twelfth International Conference, Tahoe City, CA., USA, pp. 115-123.

14. Cortes, C., and V. Vapnik, (1995), "*Support-vector networks,*" Journal of Machine Learning, vol. 20, no. 3, pp. 273-297.
15. Cover, T. M., and P. E. Hart, (1967), "*Nearest Neighbor Pattern Classification,*" Institute of Electrical and Electronics Engineers Transactions on Information Theory, vol. 13, no. 1, pp. 21-27.
16. Cristianini, N., and S. T. John, (2000), "*An Introduction to Support Vector Machines and other kernel-based learning methods,*" Cambridge University Press.
17. Dasarathy, B. V., and B. V. Sheela, (1979), "*A Composite Classifier System Design: Concepts and Methodology,*" Proceedings of the IEEE, vol. 67, no. 5, pp. 708-713.
18. Dietterich, T. G., and G. Bakiri, (1994), "*Solving multiclass learning problems via error-correcting output codes,*" Journal of Artificial Intelligence Research, vol. 2, pp. 263-286.
19. Duda, R. O., and P. E. Hart, (1973), "*Pattern Classification and Scene Analysis,*" Wiley Publisher, pp. 56-64.
20. Duda. O. R., E. H. Peter, G. S. David , (2001), "*Pattern Classification,*" Chapter 4: Nonparametric Techniques in Wiley Interscience Publisher, pp. 161-199.
21. Dudani, S., (1976), "*The Distance-Weighted k-Nearest-Neighbor Rule,*" IEEE Transactions on Systems, Man and Cybernetics, vol. 6, no. 4, pp. 325-327.
22. Friedman, N., D. Geiger, and M. Goldszmidt, (1997), "*Bayesian Network Classifiers,*" Journal of Machine Learning, vol. 29, pp. 131-161.
23. Geman, S., E. Bienenstock, and R. Doursat, (1992), "*Neural Networks and the Bias/Variance Dilemma,*" Journal of Neural Computation, vol. 4, pp. 1-58.
24. Hecht-Nielsen, R., (1989), "*Theory of the Backpropagation neural Network,*" International Joint Conference on neural networks, Washington, DC, USA, pp. 593-605.
25. Huzefa, R., and G. Karypis, (2005), "*Profile Based Direct Kernels for Remote Homology Detection and Fold Recognition,*" Journal of Bioinformatics, vol. 31, no. 23, pp. 4239-4247.
26. Karp, R. M., (1972), "*Reducibility Among Combinatorial Problems,*" Proceedings of

Sympos. IBM Thomas J. Watson Res. Center, Yorktown Heights, New York: Plenum, pp. 85-103.

27. Keller, J. M., M. R. Gray, and J. A. Givens, Jr, (1985), "*A Fuzzy K-Nearest Neighbor Algorithm,*" Journal of IEEE Transactions on Systems, Man, and Cybernetics, vol. 15, no. 4, pp. 580-585.
28. Kohavi R., (1996), "*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, pp. 202-207.
29. Kohavi, R., and G. John, (1997), "*Wrappers for Feature Subset Selection,*" Journal of Artificial Intelligence: special issue on relevance, vol. 97, no. 1-2, pp. 273-324.
30. Kokol, P., M. Zorman, M. M. Stiglic, and I. Malcic, (1998), "*The limitations of decision trees and automatic learning in real world medical decision making,*" Proceedings of the 9th World Congress on Medical Informatics MEDINFO'98, vol. 52, pp. 529-533.
31. Kononenko, I., (1991), "*Semi-naïve Bayesian classifier,*" Y. Kodratoff Editor, Proceedings of sixth European working session on learning, Springer-Verlag, pp. 206-219.
32. Kwok, S., and C. Carter, (1990), "*Multiple decision trees: uncertainty,*" Journal of Artificial Intelligence, vol.4, pp. 327-335.
33. Langley, P., and S. Sage, (1994), "*Induction of Selective Bayesian Classifiers,*" Proceedings of UAI-94, Seattle, WA, USA, pp. 399-406.
34. Mansour, Y., D. McAllester, (2000), "*Generalization Bounds for Decision Trees,*" Proceedings of the 13th Annual Conference on Computer Learning Theory, San Francisco, Morgan Kaufmann, USA, pp. 69–80.
35. Moody, J. E., (1992), "*The Effective Number of Parameters: An Analysis of Generalization and Regularization in Nonlinear Learning Systems,*" Journal of Advances in Neural Information Processing Systems, vol. 4, pp. 847-854.
36. Nock, R., and O. Gascuel, (1995), "*On learning decision committees,*" Proceedings of the Twelfth International Conference on Machine Learning, Morgan Kaufmann, Taho City, CA., USA, pp. 413-420.
37. Oliver, J. J., and D. J.Hand, (1995), "*On pruning and averaging decision trees,*"

Proceedings of the Twelfth International Conference on Machine Learning, Morgan Kaufmann, Taho City, CA., USA, pp. 430-437.

38. Pazzani, M.J., (1995), "*Searching for dependencies in Bayesian classifiers,*" Proceedings of AI & STAT'95, pp. 239-248.
39. Podgorelec, V., P. Kokol, B. Stiglic, and I. Rozman, (2002), "*Decision trees: an overview and their use in medicine,*" Journal of Medical Systems, Kluwer Academic/Plenum Press, vol. 26, no. 5, pp. 445-463
40. Quinlan, J. R., (1987), "*Simplifying decision trees,*" International Journal of Man-Machine Studies, vol. 27, pp. 221-234.
41. Quinlan, J. R., (1993), "*C4.5: Programs for Machine Learning,*" Morgan Kaufmann Publisher San Mateo, CA., USA, pp. 35-42.
42. Rada, M., (2004), "*Seminar on Machine Learning,*" a presentation of a course taught at University of North Texas.
43. Rokach L., O. Maimon, O. Arad, (2005), "*Improving Supervised Learning by Sample Decomposition,*" Journal of Computational Intelligence and Applications, vol. 5, no. 1, pp. 37-54.
44. Sands D., (1998), "*Improvement theory and its applications,*" Gordon A. D., and A. M. Pitts Editors, Higher Order Operational Techniques in Semantics, Publications of the Newton Institute, Cambridge University Press, pp. 275-306.
45. Schapire, R. E, (1990), "*The strength of weak learnability,*" Journal of Machine Learning, vol. 5, pp. 197-227.
46. Shawe-Taylor. J., and C. Nello, (1999), "*Further results on the margin distribution,*" Proceedings of COLT99, Santa Cruz, CA., USA, pp. 278-285.
47. Smith, M., (1996), "*Neural Networks for Statistical Modeling,*" Itp New Media Publisher, ISBN 1-850-32842-0, pp. 117–129.
48. Spizer, M., L. Stefan, C. Paul, S. Alexander, and F. George, (2006), "*IsoSVM – Distinguishing isoforms and paralogs on the protein level,*" Journal of BMC Bioinformatics, vol. 7:110,
    website: http://www.biomedcentral.com/content/pdf/1471-2105-7-110.pdf.
49. Tan, P. N., S. Michael, and K. Vipin, (2005), "*Introduction to Data Mining,*" Chapters 4 and 5, Addison-Wesley Publisher, pp. 145-315.

50. Triantaphyllou, E., (2007), "*Data Mining and Knowledge Discovery Via a Novel Logic-Based Approach,***"** A monograph, Springer, Massive Computing Series, 420 pages, (in print).
51. Triantaphyllou, E., and G. Felici, (Editors), (2006), "*Data Mining and Knowledge Discovery Approaches Based on Rule Induction Techniques,*" Springer, Massive Computing Series, 796 pages.
52. Triantaphyllou, E., L. Allen, L. Soyster, and S. R. T. Kumara, (1994), "*Generating Logical Expressions From Positive and Negative Examples via a Branch-and-Bound approach,*" Journal of Computers and Operations Research, vol. 21, pp. 783-799.
53. Vapnik, V., (1998), "*Statistical Learning Theory,*" Wiley Publisher, pp. 375-567.
54. Webb, G. I., (1996), "*Further experimental evidence against the utility of Occam's razor,*" Journal of Artificial Intelligence Research, vol. 4, pp. 397-417.
55. Webb, G. I., (1997), "*Decision Tree Grafting,*" Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI'97), vol. 2, pp. 23-29.
56. Weigend, A., (1994), "*On overfitting and the effective number of hidden units,*" Proceedings of the 1993 Connectionist Models Summer School, pp. 335-342.
57. Wikipedia Dictionary, (2007), website: http://en.wikipedia.org/wiki/Homogenous.
58. Wolpert, D. H, (1992), "*Stacked generalization,*" Journal of Neural Networks, vol. 5, pp. 241-259.
59. Zavrsnik, J., P. Kokol, I. Maleiae, K. Kancler, M. Mernik, and M. Bigec, (1995), "*ROSE: decision trees, automatic learning and their applications in cardiac medicine,*" MEDINFO'95, Vancouver, Canada, pp. 201-206.
60. Zhou Z. and C. Chen, (2002), "*Hybrid decision tree,*" Journal of Knowledge-Based Systems, vol. 15, pp. 515 - 528.