Computers & Operations Research 38 (2011) 174–189

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

A meta-heuristic approach for improving the accuracy in some classification algorithms

Huy Nguyen Anh Pham, Evangelos Triantaphyllou *

Department of Computer Science, 298 Coates Hall, Louisiana State University, Baton Rouge, LA 70803, USA

ARTICLE INFO

Available online 28 April 2010

Keywords:
Classification
Fitting
Generalization
False positive
False negative
Unclassifiable
Convex region
Optimization
Genetic algorithms

ABSTRACT

Current classification algorithms usually do not try to achieve a balance between fitting and generalization when they infer models from training data. Furthermore, current algorithms ignore the fact that there may be different penalty costs for the false-positive, false-negative, and unclassifiable types. Thus, their performance may not be optimal or may even be coincidental. This paper proposes a meta-heuristic approach, called the Convexity Based Algorithm (CBA), to address these issues. The new approach aims at optimally balancing the data fitting and generalization behaviors of models when some traditional classification approaches are used. The CBA first defines the total misclassification cost (*TC*) as a weighted function of the three penalty costs and the corresponding error rates as mentioned above. Next it partitions the training data into regions. This is done according to some convexity properties derivable from the training data and the traditional classification method to be used in conjunction with the CBA. Next the CBA uses a genetic approach to determine the optimal levels of fitting and generalization. The *TC* is used as the fitness function in this genetic approach. Twelve real-life datasets from a wide spectrum of domains were used to better understand the effectiveness of the proposed approach. The computational results indicate that the CBA may potentially fill in a critical gap in the use of current or future classification algorithms.

1. Introduction

The capacity and capabilities of systems for data storage and analysis have increased dramatically in recent years. The typical setting of interest in this paper involves availability of some historic data. Such data describe observations about a system or phenomenon. Each observation 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 problem reverts to a number of two-class problems (see, for instance, [1–3]). Then the main problem is to use such historic data (also known as the training data) to infer a model that would accurately classify new observations of unknown class values.

Many theoretical and practical developments have been made in the last years regarding the development of approaches for inferring classification models from training data. The most recent approaches include the Statistical Learning Theory [4], Artificial Neural Networks (ANNs) [5,6], Decision Trees (DTs) [7–9], and Support Vector Machines (SVMs) [10,4]. In such classification approaches, there are three different types of possible errors:

- The *false-negative* type, where a data point that in reality is positive is predicted as negative.
- The *false-positive* type, where a data point that in reality is negative is predicted as positive.
- The *unclassifiable* type, where the classification approach cannot predict a new point. This happens due to insufficient information being 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. Pham and Triantaphyllou [11–13] argued that such approaches usually did not try to achieve a balance between fitting and generalization when they inferred models from datasets. Thus, the models they infer may suffer from overfitting and overgeneralization problems, and this causes their poor performance. Overfitting occurs when a model can accurately classify data points that are very closely related to the training data but performs poorly with data that 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 that are not closely related to the training data.

* Corresponding author. Tel.: +1 225 578 1348; fax: +1 225 578 1465.
E-mail addresses: hpham15@lsu.edu (H.N. Pham), trianta@lsu.edu, etriantaphyllou@yahoo.com (E. Triantaphyllou).

0305-0548/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2010.04.011

Usually, current classification approaches attempt to minimize the sum of 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 some serious diseases (say some kind of aggressive cancer). In a case like this a false-positive diagnosis would subject a patient to some emotional challenge and unnecessary medical tests and treatments. On the other hand, a false-negative diagnosis may cause loss of critical time, which in turn may turn out 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 type of unclassifiable cases is more subtle. Now 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 offer advice only when a new instance is of an obvious nature (i.e., either clearly positive or clearly negative) and avoid any challenging instance. This would result in high numbers of unclassifiable cases. Thus, this outcome should be related to a penalty value as well.

Pham and Triantaphyllou in [11–13] recognized the previous problems related to the above three error types and the need to have different penalty costs associated with them. Furthermore, given the three penalty costs, an optimal or near optimal total misclassification cost (*TC*) could be reached if the fitting and generalization behaviors of the inferred models were treated differently for the positive and negative training data. They achieved this by first identifying regions in the space of the training data where the training data are located in a rather homogenous 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. Next, they applied existing classification methods on the training data within such homogenous regions, and the final model was an aggregation of the models inferred from such regions. This method was called the Homogeneity Based Algorithm (HBA). In computational experiments reported in [12,13], the HBA performed rather well. That is, the *TC* was reduced quite significantly when compared with the application of some traditional approaches without the use of the HBA.

However, a problem with the HBA is that it may require excessive computing time. This is due to the way homogenous regions are determined. For the HBA, Greig's approach of quadrants [14] was used to determine whether a region is homogenous or not. This step strongly depends on the dimensionality (i.e., primarily the number of attributes) of the training data. This is the reason why the HBA may be impractical for data of high dimensionality.

The new approach (i.e., the CBA) bypasses this 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, but still more accurate than the stand alone traditional classification approaches. However, the CBA may be used to derive systems in situations where the HBA cannot work due to excessive computing time.

The rest of the paper is organized as follows. The next section summarizes some related developments regarding model fitting and generalization. The third section provides a description of the main research problem and the fundamental assumptions for the CBA. The proposed methodology is highlighted in the fourth section. This section shows how a balance between fitting and generalization has the potential to enhance the classification accuracy of existing classification algorithms. The fifth section discusses some promising computational experiments. Finally, the last section summarizes the main contribution and offers directions for future research.

2. Some related developments

The following classification approaches have typically focused on minimizing the sum of the false-positive and false-negative error rates without considering the two error rates in a weighted fashion. They do so by controlling either fitting or generalization. Decision Tree (DT) approaches use one of two methods: pre- and post-pruning methods to control the fitting or generalization problems. Under the pre-pruning approaches described in [15–19], the approach for growing a tree is halted by some early stopping rules before generating a fully grown tree. That is, pruning occurs during the tree growing phase. The post-pruning approaches described in [7–10,20–23] operate in the reverse direction, allowing a DT to first grow to its maximum size, and then they work from the bottom up by trimming branches deemed as weak according to some evaluative criteria until no such branches remain. However, DT approaches have met with difficulty in choosing a suitable threshold value for the stopping or trimming criteria. The threshold values in the above studies have been proposed for particular applications rather than as a general threshold value.

In order to control the fitting or generalization problems, Rule-Based Classifiers, which directly infer classification rules from a training dataset, use one of two strategies: general-to-specific or specific-to-general. Under these two strategies, as described in [24–26], a classification rule is initialized by either finding all possible candidates or randomly choosing some candidates. The rule is then refined until some stopping criteria are met. The derived rules are treated by an approach described in [27]. 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, as with the DT case, the above approaches have difficulty in choosing a suitable threshold value for the stopping criteria.

K-Nearest Neighbor Classifiers, as described in [28,29], find *K* training points that are relatively similar to a testing point in order to determine its class value. The way for choosing the value for *K* affects the accuracy of these approaches [30,31]. A non-suitable value for *K* can lead to either overfitting or over-generalization [32].

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. First, Naïve Bayes Classifiers (NBCs) assume that all attributes are conditionally independent, given the value of the class variable. Second, Bayesian Belief Networks (BBNs) allow for pairs of attributes to be conditionally dependent when the value of the class variable is known. The BBNs described in [33–38] improve their classification accuracy by relaxing some independence assumptions. However, these approaches may quickly degrade to overfitting when one probabilistically combines the data with prior knowledge.

The goal of an Artificial Neural Network (ANN) is to determine the weights of the network in order to minimize the total sum of squared differences between the expected and the predicted outputs. 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 datasets [5,6]. The importance of choosing an appropriate ANN topology for a given problem can affect the accuracy of such approaches [39–42]. An inappropriate ANN topology may lead to either overfitting or overgeneralization.

The basic concept behind Support Vector Machines (SVMs) is to find a maximal margin hyperplane or set of hyperplanes that can separate training points [4,43,44]. In general, SVMs attempt to formulate learning as a convex optimization problem for which efficient algorithms are available to find a global solution. For many datasets, however, an SVM cannot formulate the learning problem as a convex optimization problem because of excessive misclassifications. This may lead to overgeneralization.

3. Formal problem definition and fundamental assumptions

3.1. Problem description

Let us denote as C_{FP}, C_{FN}, and C_{UC} the penalty costs for the false-positive, the false-negative, and the unclassifiable types, respectively. The values for these parameters depend on the particular application under consideration. Let *RateFP*, *RateFN*, and *RateUC* be the false-positive, the false-negative, and the unclassifiable rates, respectively. The problem addressed in this paper is how to modify an existing classification model such that the following weighted *TC* (i.e., the total misclassification cost) will be minimized or significantly reduced:

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

Next, some key assumptions are described.

3.2. Fundamental assumptions in the development of the CBA

The CBA 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 [45] for a discussion on some related key problems in data analysis.

The HBA, as described in [11–13], identifies regions in the space of the training dataset where training points are located in a homogenous manner. It is this step of the HBA that makes it computationally expensive. Instead of the concept of homogenous regions, the new approach (i.e., the CBA) uses the concept of convex regions as defined by Melnik [46]. 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 a convex region.

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 data points, respectively. Two positive nodes are considered to be connected in this graph if and only if the following test is satisfied. First, we consider the line segment in the data space that 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 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 data points.

A clique in this graph corresponds to a set of training data points that can be connected with each other in any possible pair. Therefore, in the way defined by Melnik [46] 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 this type of convex regions as a substitute for the homogenous regions applied in the more time consuming HBA approach. However, there are a number of differences between the convex regions described above and the homogenous regions used in the HBA.

The ways in which a convex region and a homogenous region are determined are different. That is, each homogenous region is determined by training data points located in a homogeneous manner [11–13]. However, each convex region includes training data points not necessarily located in such a manner.

The convex regions depend on two other factors that do not apply to the homogenous regions. 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 4.1. 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 to be inaccurate in some cases.

The above observations indicate that the CBA may not always be superior to the HBA when it comes to improving the 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 *convex density* (*CD*). This value measures the density of the training points within a given convex region. An appropriate definition of *CD* value is discussed in Section 4.2.

The key assumptions for the CBA are summarized as follows. If an unclassified point is covered by a convex region that 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. On the other hand, the performance of a given classification approach may be enhanced when regions derived from the original approach are convex regions that have high *CD* values. This could be done if such regions are used to find an optimal balance between fitting and generalization.

4. Proposed approach—the CBA

Assume that two classification models denoted as M_1 (one for positive data and the other for negative data) and a training

dataset T are given. The desired goal of the CBA is to enhance models M_1 in order to obtain an optimal TC as described in Eq. (1). There are five control parameters used in the CBA.

- Two expansion coefficients α^+ and α^- to be used when expanding positive and negative convex regions, respectively.
- A breaking threshold value β^+ to be used when determining whether a positive convex region should be broken. A similar concept is applied on the breaking threshold value β^- for negative convex regions.
- A density threshold value γ to be used when determining whether a region is convex.
- The main steps of the CBA are depicted in Fig. 1 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 CBA. This distance requires that T's attributes should be defined in the same unit (i.e., as percentages). Next, the CBA divides T into two random sub-datasets: T_1 whose size is, says, 90% of T's size and T_2 whose size is the remaining (i.e., 10%) of T's size. These percentages can be determined empirically by trial and error. Finally, the CBA randomly initializes the four parameters (α^+, α^-, β^+, β).
- *Phase* 2 (Step 3): Define the graphs and represent them as two connectivity matrices for the positive and the negative points in T_1 using the method described in Section 3.2. 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 4–7): 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 convex regions [46]. A heuristic algorithm for this task is shown in Section 4.3. Step 5 starts with computation of *CD* values of the convex regions. Some convex regions are broken again if their *CD* values are less than either β^+ (for positive regions) or β^- (for negative regions). For example, we may have a convex region whose nodes are separated into two groups that 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 4.4. The covering task is based on the fact that each hypersphere is determined by a center and a radius. Hyperspheres do not depend on dimensionality of the training dataset.
- *Phase* 4 (Step 8): Expand the hyperspheres obtained in Phase 3 in decreasing order of their *CD* values. The expansion algorithms are shown in Section 4.5.
- *Phase* 5 (Step 9): Apply the genetic algorithm (GA) described in Section 4.6 to Phases 3 and 4 with Eq. (1) as the fitness function and T_2 as the calibration dataset. The GA approach finds the four optimal parameters (α_*^+,α_*^-,β_*^+,β_*^-), given the three cost coefficients C_{FP}, C_{FN}, and C_{UC}.
- *Phase* 6 (Step 10): Use the optimal parameters (α_*^+,α_*^-,β_*^+,β_*^-) on the entire training dataset T to repeat Phases 2–4 and infer the final pair of models M_2.

The following sections provide details about the above phases.

4.1. A heuristic to determine sample points

As described in Section 3.2, whether two positive (or two negative) nodes (training data 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 follows:

$$d = \min\{\textit{Euclidean distance of all pairs of training data points in } T_1\}. \quad (2)$$

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 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.

Input: The positive and negative classification models M_1, the training dataset T, the density threshold value γ, 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 (α^+, α^-, β^+, β^-).
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 β^+ or β^- for the positive and negative data, respectively.
7. Cover the convex regions obtained in Steps 4 and 6 with hyperspheres.
8. For each hypersphere C in decreasing order of the *CD* values do:
   Expand C by using *CD(C)* and α^+ or α^- 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 (α_*^+, α_*^-, β_*^+, β_*^-).
10. Use the optimal values (α_*^+, α_*^-, β_*^+, β_*^-) 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.

Fig. 1. CBA.

4.2. Determining whether a region is convex

By using a sampling step of size d as defined in Section 4.1, the CBA derives positive and negative graphs. Next, the CBA uses clustering to break a graph into regions. This section describes an algorithm that is used to determine whether a region is convex and is depicted in Fig. 2. As discussed in Section 3.2, 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_P is convex is determined by checking whether P's connectivity matrix includes all '1's. A softer condition for P's connectivity matrix can be applied. That is, if the percentage of number of '1's in P's connectivity matrix is greater than or equal to γ, say for γ equal to 0.9, then P is considered to be convex. The value of 0.90 was used for the parameter γ as a 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 γ is a rather small number.

As seen in Step 3 in Fig. 2, each convex region is associated with a CD value. Tichy [47] 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, Fig. 3 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–E as depicted in Fig. 3. 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 $9/18=0.5$. This is apparently a contradiction since circle B is denser than D. The value for $CD(P)$ is directly related to N_P and to the average minimum distance to neighbors denoted as h, which is defined as follows:

$$h = \sum_{i}^{N_P} \frac{\min(d_i)}{N_P}. \tag{3}$$

The notation $\min(d_i)$ in Eq. (3) is the Euclidean distance between each node and its nearest neighbor. If P has a higher N_P value and a smaller h value, then P is denser. We propose $CD(P)$ to be calculated as follows, where D is the number of dimensions of the training points in T_1:

$$CD(P) = \frac{\ln(N_P)}{h^D}. \tag{4}$$

Eq. (4) shows that if N_P 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 an increase in the value of $CD(P)$. Hence, $CD(P)$ is inversely proportional to h, while $CD(P)$ is directly proportional to $\ln(N_P)$. The function $\ln(N_P)$ achieves a slighter effect of N_P on $CD(P)$. For instance, the CD(A), CD(B), CD(C), CD(D), and CD(E) values for Fig. 3 now are equal to $\ln(16)/1^2 \approx 2.77$, $\ln(9)/1^2 \approx 2.19$, $\ln(4)/1^2 \approx 1.38$, $\ln(9)/2^2 \approx 0.55$, and $\ln(4)/1^2 \approx 1.38$, respectively. Intuitively, these density values make better sense than the ones derived from Tichy's definition.

4.3. A heuristic algorithm for breaking a region into convex regions

The problem of finding the minimum number of convex regions that cover a concave region P of size N_P is similar to a form of the *set cover problem*, an NP-complete problem [48]. Melnik [46] 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 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 with each other.

A heuristic approach for breaking a region into a set of convex regions is proposed in Fig. 4. This approach is based on Hierarchical Clustering (HC) as described in [49–51]. 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_p) into K desired clusters, where $K=1, 2, \ldots,$ or N_p (see also [51]).

The algorithm depicted in Fig. 4 may end up with N_p 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

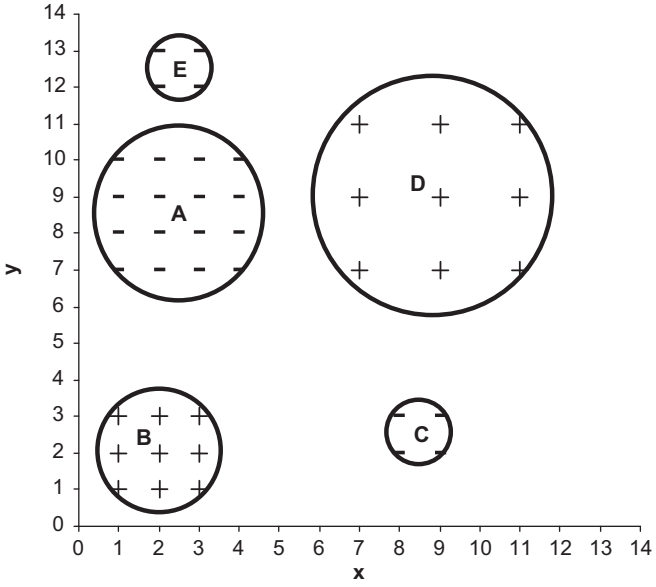

Fig. 3. Example of convex regions in T_1.

Input: Region P of size N_P and the density threshold value γ.

1. Consider P's connectivity matrix denoted as L.
2. Set num = the number of 1's in L.
3. If $\frac{num}{N_P \times (N_P - 1)} \geq \gamma$, then P is convex and $CD(P)$ is computed by Equation (4).

Output: Whether P is convex.

Fig. 2. Algorithm for determining whether a region is convex.

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

1. Use the HC approach to produce a dendrogram with a hierarchy of all possible clusters (the maximum number of which is equal to N_P).
2. For $K = 1$ to N_P 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 (β^+ or β^-), then these regions are returned and we stop the iterations.
7. End {for}

Output: A set of convex regions.

Fig. 4. Heuristic algorithm for breaking a region.

values were inadequate (i.e., they were too large). If this is the case, then the GA approach will run again as the fitting function given as Eq. (1) would suffer gross overfitting of the data. Successive runs of the GA approach should converge to an optimal (or near optimal) set of 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.

4.4. 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 [52]. Let us consider a convex region P of size N_P. A hypersphere that covers P is determined by a center C_P and radius R_P. The algorithm is initialized by a good guess for creating a hypersphere B_1 for P. This is done by finding two nodes of P that 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 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.

4.5. Methods for expanding a hypersphere

This section shows how to expand a hypersphere F of size N_F using the value of $CD(F)$ and the expansion coefficient. In order to illustrate the expansion approach, consider the example depicted in Fig. 5, which is based on the one shown in Fig. 3.

Recall that the CD values for the convex regions A–E are 2.77, 2.19, 1.38, 0.55, and 1.38, respectively. Let the two breaking threshold values β^+ and β^- 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 α^+ and α^- be 2.00 (both equal). In decreasing order of CD values, the regions A, B, C, E, and D are expanded as shown in Fig. 5. The expanded regions are covered by the solid line circles (or hyperspheres) as discussed in Section 4.4. 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.

There are two types of expansion: a radial expansion in which a hypersphere F is expanded in all directions and a linear

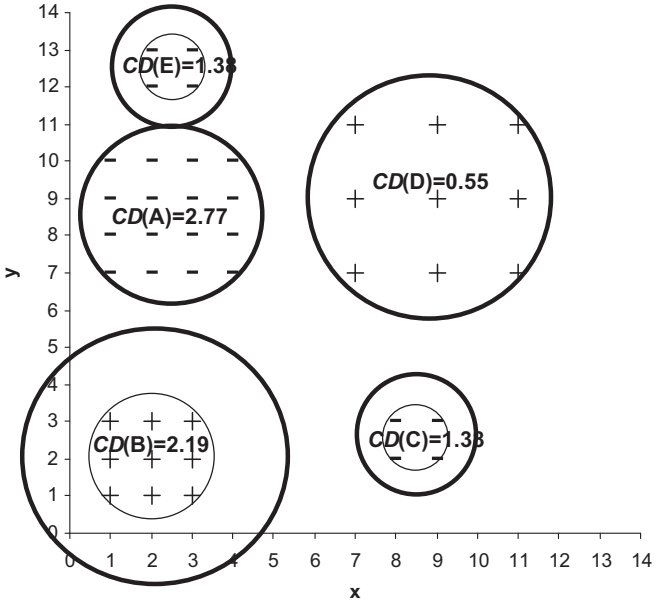

Fig. 5. Example of the expansion approach.

expansion in which a hypersphere F is expanded in a certain direction. The following section shows in detail these two expansion types.

4.5.1. Radial and linear expansions

Let M be a region that is expanded from F. Let R_F and R_M denote the radiuses of F and M, respectively. In the radial expansion algorithm depicted in Fig. 6, R_F is increased by a *step-size increment* denoted as W. One gets

$$R_M = R_F + W. \tag{5}$$

Using a dichotomous search methodology, Pham and Triantaphyllou [11–13] proposed a value for W as described in Eq. (6). If $CD(F)$ is less than one, then an additional parameter Q is applied. This parameter ensures that the step-size increase is not too fast. Furthermore, if $CD(F)$ is too small, F is not expanded:

$$W = \frac{R_G - R_F}{2}\frac{1}{Q \times CD(F)}. \tag{6}$$

Thus, when one substitutes Eq. (6) back into Eq. (5), R_M becomes

$$R_M = R_F + \frac{R_G - R_F}{2}\frac{1}{Q \times CD(F)}. \tag{7}$$

Let us consider the example indicated in Fig. 7 which is based on the one shown in Fig. 5. Assume that Q is 1.00. A closer

Input: Hypersphere F of density $CD(F)$ and radius R_F, and the expansion coefficient α^+ or α^- (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
   Set $E = M$ (i.e., $R_E = R_M$).
   Expand M by using Equation (7).
   Until (R_M satisfies the stopping conditions discussed in Section 4.5.2 or $R_M = R_G$).
4. If R_M satisfies the stopping conditions, then STOP.
   Else, go to Step 2.

Output: An expanded region E.

Fig. 6. Algorithm for radial expansion.

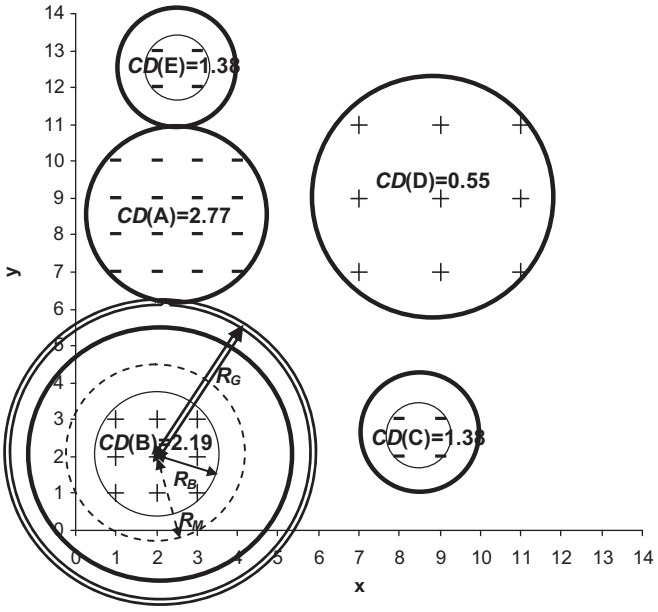

Fig. 7. Example for the radial expansion algorithm.

examination of Fig. 7 reveals that circle B (i.e., the one-line circle with $R_B = 1.8019$) is covered by three circles: a double-line circle G with $R_G = 1.8019 \times 2 = 3.6038$, a solid-line circle that shows the final expanded region, and a dashed-line circle M of radius R_M given as follows:

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

Eq. (7) computes the following successive values for R_M: 2.5308, 2.7758, 2.9648, and so on, until R_M is 3.6036. At that moment, the iterations are terminated because of the stopping conditions discussed in the next section.

The linear type first expands hypersphere F into hypersphere M using the radial expansion. Next, M is expanded in a given direction using the radial approach until it satisfies the stopping conditions (given next). The final region is the union of all expanded regions.

4.5.2. Description of the stopping conditions

The stopping conditions for expanding a hypersphere F of size N_F into M depend on $CD(F)$ and the expansion coefficient α^+ or α^- (for positive or negative regions, respectively). If $CD(F)$ 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. We propose that the radius R_M should not be greater than the product of the following terms: $CD(F)$, α^+, 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 accept several noisy points if $CD(F)$ is high enough. Thus, for positive regions the radial expansion approach is stopped if any one of the following two conditions is not satisfied:

$$R_M \le CD(F) \times R_F \times \alpha^+ \quad \text{and the number of noisy points within } M \text{ is less than or equal to } CD(F) \times \alpha^+. \tag{8}$$

A similar equation exists for negative regions:

$$R_M \le CD(F) \times R_F \times \alpha^- \quad \text{and the number of noisy points within } M \text{ is less than or equal to } CD(F) \times \alpha^-. \tag{9}$$

4.6. A genetic algorithm (GA) to find optimal values of the control parameters

The four parameters (α^+, α^-, β^+, β^-) control the number of misclassification cases for the final classification models. Since the ranges for the four parameters depend on each individual application, the search space might be too large. An exhaustive search could be impractical.

The GA approach is of the type described in [53,54] and has been applied to find the optimal parameters denoted as $(\alpha_*^+, \alpha_*^-, \beta_*^+, \beta_*^-)$. The use of the GA approach for the CBA was decided because the objective function (i.e., Eq. (1)) of the CBA may have multiple extreme points as the error rates may vary across the input data space in ways that cannot be expressed by usual mathematical functions.

At each step, the GA approach selects chromosomes from the current population as parents for the next population and uses them to produce (via crossover and mutation operations) children for the next generation. The selection of parents is done using Eq. (1) as the fitness function and the set T_2 for calibration. Each chromosome consists of four genes that correspond to the four control parameters $(\alpha^+, \alpha^-, \beta^+, \beta^-)$. 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 α^+ or α^- is equal to 1, this indicates no expansion of a region is required. Similarly, when the value of β^+ or β^- 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.

A crossover function creates children by combining pairs of parents from the population. At each coordinate of a child, the crossover function randomly picks the gene up at the same coordinate from one of the parents and then assigns it to the child. 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

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

Fig. 8. Example of the mutation function.

consider the first two genes α^+ and α^-, which are in the range $[a,b]$, while the last two genes β^+ and β^- are in the range $[c,d]$. The mutation function first randomizes a chromosome (t_1,t_2,t_3,t_4) using a Gaussian distribution (this distribution was selected empirically by trial and error). Next, genes of the mutation child are created using Eqs. (10) and (11). These equations attempt to generate genes that 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.

$$g_1 = ((\alpha^+ \text{ OR } t_1) \text{ OR } a) \text{ AND } b \text{ and } g_2 = ((\alpha^- \text{ OR } t_2) \text{ OR } a) \text{ AND } b. \tag{10}$$

$$g_3 = ((\beta^+ \text{ OR } t_3) \text{ OR } c) \text{ AND } d \text{ and } g_4 = ((\beta^- \text{ OR } t_4) \text{ OR } c) \text{ AND } d. \tag{11}$$

In order to help motivate the use of the mutation function, we consider the parent chromosome $(\alpha^+,\alpha^-,\beta^+,\beta^-)$ to be $(2,1,5,7)$. Assume that (α^+,α^-) is in the range $[0,3]$, while (β^+,β^-) is in the range $[0,10]$. Suppose that the chromosome (t_1,t_2,t_3,t_4) created using a Gaussian distribution is $(1,1,3,7)$. The mutation child is presented in Fig. 8. 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.

5. Complexity analysis

Next, we analyze time complexity of the CBA for the general case. Let N be the number of 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 [46].

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 Fig. 4 is related to the algorithm depicted in Fig. 2. Each iteration of the breaking algorithm includes two steps: breaking a region into smaller regions using the hierarchical clustering approach and determining whether a region is a convex region. The time complexity of the hierarchical clustering approach is $O(N^2)$ [49,50]. The time complexity for determining whether a region is a convex region 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 using Ritter's linear algorithm. Thus, Step 7 takes $O(N^2)$ time. Step 8 expands convex regions by using the algorithm depicted in Fig. 6. 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 6–8. We assume that the GA approach is executed for 100 generations. Thus, the time complexity of the GA approach is $O(100\times N^3)=O(N^3)$ [53]. Step 10 repeats Steps 3–8 with the four optimal parameters $(\alpha_*^+,\alpha_*^-,\beta_*^+,\beta_*^-)$ on T. This step makes the total time complexity of Steps 3–8 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 $O(N^3)$.

6. Some empirical studies

This section starts with a brief summary of the CBA. Please recall that for a given classification approach, the CBA first represents a given training dataset as two graphs: one for the positive data and one for the negative data. Next, the CBA analyzes these two graphs using clustering to identify concave and convex regions in the training data. Any concave regions are broken into convex ones. Densities of the convex regions (for the positive and the negative data points) are used to shrink or expand these regions. This is done using a GA approach in a way that attempts to minimize the total misclassification cost by determining an optimal balance between fitting and generalization. This is achieved by taking into consideration the false-positive, false-negative, and unclassifiable rates and penalty costs as expressed in Eq. (1).

6.1. Datasets and the environment for the experiments

The original classification algorithms 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 [11–13] and with the CBA. We used the four-fold cross-validation method to analyze their performance. The *TC*s obtained from the original algorithms and those in conjunction with the HBA and with the CBA when applied on the same dataset were compared.

The experiments used the 12 datasets found in [55–57]. The results of some previous studies on these datasets are shown in Table 1. These datasets were selected because current classification algorithms have analyzed them with variable success and are considered as a kind of benchmark datasets. In the experiments, a given dataset T was normalized using Eq. (12), so all units are dimensionless. The term $T_{i,j}$ in this equation shows the value of the data point i in terms of attribute j in T. A coefficient equal to 10,000 units was applied in Eq. (12) 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.

$$T_{i,j} = \frac{T_{i,j}\times 10{,}000}{\sum_i T_{i,j}} \tag{12}$$

The CBA assumed that β^+ and β^- were in the range $[0,1]$, while α^+ and α^- were in the range $[0,30]$. These ranges were determined empirically. The experiments were run on a PC with a CPU of 2.8 GHZ speed and 3 GB RAM under the Windows XP operating system. We used the libraries associated with the default settings in Matlab [58] 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

Table 1
Datasets used in the experiments.

Dataset	No. of points	No. of attributes	Previous studies	Accuracy (%)
Pima Indians Diabetes (PID) [55]	768	8	Early Neural Networks [59]	76.0
			IncNet [60]	77.6
			Fuzzy approach [61]	77.6
			Flexible Neural-Fuzzy Inference System [62]	78.6
			Fuzzy Neural Networks [63]	81.8
			Statlog Project [64]	78.0
Haberman Surgery Survival (HSS) [55]	306	3	SVMs using linear terms in the objective function [65]	71.2
			Proximal SVMs [66]	72.5
			Integer SVMs [67]	62.7
			Logical functions [68]	66.2
Wisconsin Breast Cancer (WBC) [55]	286	10	C4.5 [9]	94.7
			Rule Induction approach [69]	96.0
			Linear Discriminant Analysis approach [70]	96.8
			SVMs [71]	97.2
			Neuro-Fuzzy approach [72]	95.1
			Fuzzy-GA approach [73]	97.5
			Neuro-Rule approach [74]	98.1
			Supervised Fuzzy Clustering [75]	95.6
			Fuzzy Artificial Immune Recognition System [17]	98.5
			Classification through ELECTRE and data mining [27]	94.4
Liver-Disorder (LD) [55]	345	6	RULES-4 algorithm [76]	55.9
			C4.5 [77]	65.5
			Reduced SVMs [78,79]	74.9
			SVMs [80]	69.2
			Least Squares SVMs [81]	94.3
			FAIRS [17]	83.4
Statlog Heart (SH) [55]	270	13	Different approaches in Statlog Project [82]	76.7
			Attribute weighted artificial immune system [83]	87.4
Australian Credit Approval (ACA) [55]	690	14	C4.5 [34]	85.7
			Eight genetic programming approaches [84]	83.0
			Different approaches in Statlog Project [85]	86.9
			Extend Naive Bayes [86]	76.7
			SVMs [87]	85.5
Appendicitis (AP) [57]	106	7	Predictive Value Maximization approach [57]	89.6
			Fuzzy Rule-Based Classification System [88]	84.0
			Nefclass [89]	87.7
FourClass [56]	862	2	Fuzzy Kernel Multiple Hyperspheres [90]	99.8
			KNN-SVM [91]	100.0
German Credit Data (GCD) [55]	1000	24	Graph-based relational concept learner [92]	71.5
			DTs [93]	72.9
			SVMs [87]	77.9
Ionosphere (INS) [55]	351	34	Features selection in conjunction with ANNs and KNNs [94]	90.6
			Integration between fuzzy class association rules and SVMs [95]	89.2
Parkinsons (PA) [55]	195	23	ANNs [96]	81.3
			SVMs [97]	91.4
SPECTF [55]	267	45	CLIP 3 [98]	77.0
			Rough set-base multiple criteria linear programming [99]	68.0

previous studies. Thus, the objective function in this family was assumed to be

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

The results for this family of tests are presented in Table 2. This table shows the three failure rates (as percentages) and the *TC*s obtained under various algorithms. The column "Improvement 1" shows any improvements (as a percentage) of the *TC* achieved by the HBA when compared with 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 with 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 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 3–9.

In particular, Table 2 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 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

Table 2
$TC = \min(1 \times RateFP + 1 \times RateFN)$.

Dataset	Algorithm	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
SH	SVM	0.0	44.1	2.9	44.1	N/A					13.2	16.2	14.7	29.4	33.3	N/A
	DT	8.8	23.5	1.5	32.4						7.4	14.7	16.2	22.1	31.8	
	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	N/A					0.0	35.8	6.9	35.8	0.0	N/A
	DT	1.2	28.3	2.3	29.5						1.2	28.3	2.3	29.5	0.0	
	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	N/A					6.0	8.0	21.6	14.0	50.0	N/A
	DT	23.6	5.6	1.2	29.2						5.6	8.8	12.4	14.4	50.7	
	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	N/A					6.8	19.3	17.0	26.1	14.8	N/A
	DT	0.0	31.8	2.3	31.8						6.8	15.9	21.6	22.7	28.6	
	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	N/A					6.1	12.2	2.0	18.4	0.0	N/A
	DT	2.0	18.4	0.0	20.4						10.2	0.0	10.2	10.2	50.0	
	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	N/A					6.0	9.0	13.4	14.9	28.6	N/A
	DT	1.5	22.4	3.0	23.9						20.9	0.0	3.0	20.9	12.5	
	ANN	20.9	0.0	1.5	20.9						13.4	0.0	10.4	13.4	35.7	

Table 3
Accuracy percentages of the HBA and CBA.

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

computing time. Furthermore, Table 2 shows 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 results by the CBA in terms of the *TC* values had no improvement when compared to those by the HBA. Table 2 also shows that some *RateUC*s 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. An extreme situation might occur when all the cases are

Table 4
$TC = \min(1 \times RateFP + 20 \times RateFN + 3 \times RateUC)$.

Dataset	Algorithm	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	N/A					0.0	0.0	100.0	300.0	66.3	N/A
	DT	8.8	23.5	1.5	483.8						0.0	0.0	100.0	300.0	38.0	
	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	N/A					18.4	0.0	2.0	24.5	0.0	N/A
	DT	2.0	18.4	0.0	369.4						10.2	0.0	10.2	40.8	89.0	
	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	N/A					20.9	0.0	1.5	25.4	0.0	N/A
	DT	1.5	22.4	3.0	458.2						20.9	0.0	3.0	29.9	93.5	
	ANN	20.9	0.0	1.5	25.4						20.9	0.0	1.5	25.4	0.0	

classified as the unclassifiable type. An illustration of this situation is 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 shows the accuracy percentages of the HBA and CBA in terms of the sum of *RateFP* and *RateFN*. The negative numbers show the CBA had no improvement when compared to the HBA. The HBA (when it was possible to use) results were 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 the results obtained from previous studies.

Analyses of the impact of *RateFP*, *RateFN*, and *RateUC* are driven by 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 medical datasets: PID, HSS, WBC, LD, AP, FourClass, SH, PA, and SPECTF. Now we penalized considerably more for the false-negative type than for the false-positive and unclassifiable types. For the medical datasets, 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:

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

The results for this family of tests are presented in Table 4. 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 4 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 show that the CBA could run with all nine medical datasets. All results by the CBA in terms of the *TC* values had no improvement when compared to those by the HBA. Since we penalized more for the false-negative type, the average *RateFN* achieved by the CBA in Table 4 was less than that of the CBA in Table 2 by about 90.6%.

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

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

The results for this family of tests are presented in Table 5. 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

Table 5
$TC = \min(1 \times RateFP + 100 \times RateFN + 3 \times RateUC)$.

Dataset	Algorithm	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	3128.6	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	2974.0	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	2329.2	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	1577.9	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	1092.2	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	2802.8	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	1862.5	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	1873.6	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	3724.4	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	1537.2	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	2254.7	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	4420.6	N/A					0.0	0.0	100.0	300.0	93.2	N/A
	DT	8.8	23.5	1.5	2366.2						0.0	0.0	100.0	300.0	87.3	
	ANN	2.9	22.1	5.9	2226.5						13.2	0.0	48.5	158.8	92.9	
PA	SVM	18.4	0.0	2.0	24.5	N/A					18.4	0.0	2.0	24.5	0.0	N/A
	DT	2.0	18.4	0.0	1838.8						10.2	0.0	10.2	40.8	97.8	
	ANN	4.1	16.3	4.1	1649.0						14.3	0.0	6.1	32.7	98.0	
SPECTF	SVM	20.9	0.0	1.5	25.4	N/A					20.9	0.0	1.5	25.4	0.0	N/A
	DT	1.5	22.4	3.0	2249.3						20.9	0.0	3.0	29.9	98.7	
	ANN	20.9	0.0	1.5	25.4						20.9	0.0	1.5	25.4	0.0	

Table 6
$TC = \min(20 \times RateFP + 1 \times RateFN + 3 \times RateUC)$.

Dataset	Algorithm	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

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 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 show that the CBA could again run with all nine medical datasets. All results by the CBA in terms of the *TC* values had no improvement when compared to those by the HBA. Since we penalized considerably more for the false-negative type, the average *RateFN* achieved by the CBA in Table 5 was less than that of the CBA in Table 2 and 4 by about 92.2% and 16.7%, respectively. These results show 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 5 were the same as those in Table 4. This is 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, 4, and 5 were the same as those under the original algorithms. The reason is 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 4 and 5 also

Table 7
$TC = \min(100 \times RateFP + 1 \times RateFN + 3 \times RateUC)$.

Dataset	Algorithm	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	2801.2	0.0	0.0	100.0	300.0	89.3
	DT	23.6	5.6	1.2	2369.2	0.0	65.2	8.4	90.4	96.2
	ANN	21.2	8.4	1.2	2132.0	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

Table 8
Computing times (in hours) of the original algorithm, HBA, and CBA 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 9
*TC*s achieved by the original algorithms, HBA, and CBA when were 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

show that some degenerative results occurred when the HBA and CBA were applied on the PID, WBC, LD, and SH datasets.

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. It was hoped 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

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

Table 6 presents the results for this family of tests. 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 6 was less than that of the CBA in Table 2 by about 95.2%.

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

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

The results for this family of tests are presented in Table 7. 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 7 was less than that of the CBA in Table 2 and 6 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 7 were the same as those of Table 6. 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 6 and 7, some degenerative results also occurred when the CBA was applied on the GCD dataset.

As seen in Tables 2 and 4–7, the HBA and CBA achieved *TC*s less than or equal (i.e., better) to 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 4.6. 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 and 4–7 show that the *TC*s obtained by the CBA were typically greater (i.e., worse) than those 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 was more frequent than that achieved by the HBA. In fact, only ten cases under the HBA were identical to those under the original algorithms compared to 37 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—this 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 in 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.

The results for the HSS and PA datasets in this family of tests are presented in Figs. 9 and 10, 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

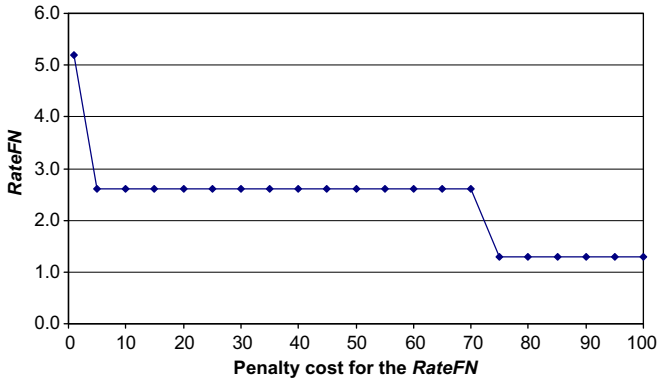

Fig. 9. For different costs of *RateFN*, the CBA in conjunction with an SVM was applied on the HSS dataset.

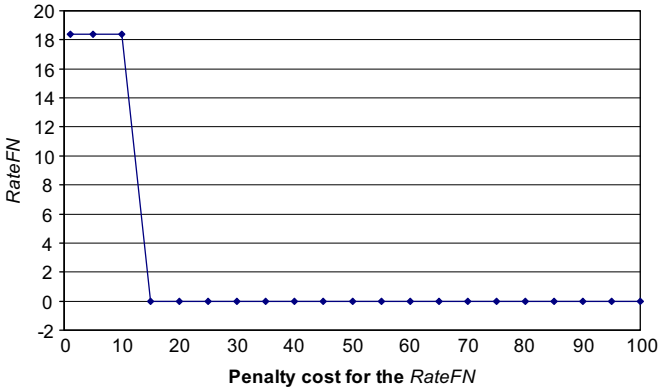

Fig. 10. For different costs of *RateFN*, the CBA in conjunction with a DT was applied on the PA dataset.

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 to the penalty cost. The above observations show that the plots followed stepwise patterns. There are two issues that 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. Second, 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 to determine all such intervals and the points of change.

7. Conclusions

When current classification approaches derive models from training data, a balance between fitting and generalization is not considered in a systematic manner. Moreover, error rates in conjunction with penalty costs are not used in such approaches. Some earlier studies (i.e., [11–13]) recognized the above problems and proposed a method called the Homogeneity Based Algorithm (HBA). However, the HBA may be impractical for large datasets because of the excessive computing time required due to the way the HBA determines homogenous regions.

A meta-heuristic approach, called the Convexity Based Algorithm (CBA), was proposed to alleviate the problem of excessive computing time with the HBA. Instead of the concept of homogenous sets, the CBA uses the concept of convex regions that do not depend on dimensionality of the training data. The CBA was studied on 12 datasets with different penalty costs for classification errors and was compared with (when it was possible to use) the HBA. The results are summarized in Tables 8 and 9.

Table 8 presents the computing times of the HBA and CBA when applied on the 12 datasets. Recall that the experiments were run on a PC with a CPU of 2.8 GHZ speed and 3 GB RAM under a Windows XP operating system. The CBA computing time on each dataset was defined as the average of 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 8 shows that the CBA was applicable to all 12 datasets, while the HBA was applicable to only 6 datasets. Furthermore, the CBA computing time was always less than that of the HBA (when it was possible to use).

Table 9 is a summary of the results derived from Table 2. In this table, the value "CBA-*TC*" for each given dataset was defined

as the average of the total misclassification cost (*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 9 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 8 and 9 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. Finally the website 〈www.csc.lsu.edu/~huypham/CBA_guide.html〉 gives the CBA and HBA approaches along with some related computational tools.

The CBA could be expanded in several ways. For instance, a new approach may derive convex regions more accurately. Additionally, the determination of suitable values for the density threshold value γ could also be considered in future work. One may also need to study an approach that can determine appropriate penalty costs for various types of error such as the theoretical model proposed in [100].

Acknowledgements

The authors are very appreciative for valuable comments made by the anonymous reviewers. These comments have helped the authors to improve the quality of this paper.

References

[1] Pujol O, Radeva P, Vitrià J. Discriminant ECOC: a heuristic method for application dependent design of error correcting output codes. IEEE Transactions on Pattern Analysis and Machine Intelligence 2006;28(6):1001–7.
[2] Dietterich TG, Bakiri G. Solving multiclass learning problems via error-correcting output codes. Journal of Artificial Intelligence Research 1995;2:263–86.
[3] Crammer K, Singer Y. On the learnability and design of output codes for multiclass problems. Machine Learning 2002;47(2–3):201–33.
[4] Vapnik V. Statistical learning theory. Wiley-Interscience; 1998. [p. 375–567].
[5] Abdi H. A Neural Network primer. Journal of Biological Systems 2003;2:247–81.
[6] Hecht-Nielsen R. Theory of the backpropagation neural network. In: Proceedings of the international joint conference on neural networks, Washington, DC, USA, 1989. p. 593–605.
[7] Quinlan JR. C4.5: programs for machine learning. San Mateo, CA, USA: Morgan Kaufmann Publisher; 1993. [p. 35–42].
[8] Quinlan JR. Simplifying decision trees. International Journal of Man–Machine Studies 1987;27:221–34.
[9] Quinlan JR. Improved use of continuous attributes in C4.5. Artificial Intelligence Research 1996;4:77–90.
[10] Breiman L. Bagging predictors. Journal of Machine Learning 1996;24:123–40.
[11] Pham HNA, Triantaphyllou E. Prediction of diabetes by employing a new data mining approach which balances fitting and generalization. In: Yin Lee R, editor. Studies in computation intelligence, vol. 131. Berlin, Germany: Springer; 2008. [p. 11–26, chapter 2].
[12] Pham HNA, Triantaphyllou E. An application of a new meta-heuristic for optimizing the classification accuracy when analyzing some medical datasets. Expert Systems with Applications 2009;36(5):9240–9.
[13] Pham HNA, Triantaphyllou E. The impact of overfitting and overgeneralization on the classification accuracy in data mining. In: Maimon O, Rokach L, editors. Soft computing for knowledge discovery and data mining. New York, NY, USA: Springer; 2007. [p. 391–431, part 4, chapter 5].
[14] Greig-Smith P. The use of random and contiguous quadrats in the study of the structure of plant communities. Annals of Botany 1952;16:293–316.
[15] Cohen S, Rokach L, Maimon O. Decision-tree instance-space decomposition with grouped gain-ratio. Journal of Information Science 2007;177(17):3592–612.
[16] Kohavi R. Scaling up the accuracy of naive-Bayes classifiers: a decision-tree hybrid. In: Proceedings of the second international conference on knowledge discovery and data mining, Portland, OR, USA, 1996. p. 202–7.
[17] Polat KA, Sahan S, Kodaz H, Gunes S. Breast cancer and liver disorders classification using artificial immune recognition system (airs) with performance evaluation by fuzzy resource allocation mechanism. Expert Systems with Applications 2007;32:172–83.
[18] Rokach L, Maimon O, Arad O. Improving supervised learning by sample decomposition. Journal of Computational Intelligence and Applications 2005;5(1):37–54.
[19] Zhou Z, Chen C. Hybrid decision tree. Journal of Knowledge-Based Systems 2002;15:515–28.
[20] Breiman L, Friedman JH, Olshen RA, Stone CJ. Classification and regression trees. Chapman & Hall/CRC Publisher; 1984. [p. 279–93].
[21] Breiman L. Random forests. Journal of Machine Learning 2001;45(1):5–32.
[22] Mansour Y, McAllester D. Generalization bounds for decision trees. In: Proceedings of the 13th annual conference on computer learning theory, San Francisco, CA, USA, 2000. p. 69–80.
[23] Webb GI. Further experimental evidence against the utility of Occam's razor. Journal of Artificial Intelligence Research 1996;4:397–417.
[24] Clark P, Boswell R. Rule induction with CN2: some recent improvements. In: Kodratoff Y, editor. Machine learning—EWSL-91. Berlin, Germany: Springer; 1991. [p. 151–63].
[25] Clark P, Niblett T. The CN2 algorithm. Journal of Machine Learning 1989;3:261–83.
[26] Cohen WW. Fast effective rule induction. In: Proceedings of the 12th international conference on machine learning, Tahoe City, CA, USA, 1995. p. 115–23.
[27] Mastrogiannis N, Boutsinas B, Giannikos I. A method for improving the accuracy of data mining classification algorithms. Computers and Operations Research 2009;36(10):2829–39.
[28] Cover TM, Hart PE. Nearest neighbor pattern classification. Institute of Electrical and Electronics Engineers Transactions on Information Theory 1967;13(1):21–7.
[29] Dasarathy BV, Sheela BV. A composite classifier system design: concepts and methodology. Proceedings of the IEEE 1979;67(5):708–13.
[30] Dudani S. The distance-weighted k-nearest-neighbor rule. IEEE Transactions on Systems, Man, and Cybernetics 1976;6(4):325–7.
[31] Keller JM, Gray MR, Givens Jr. JA. A fuzzy K-nearest neighbor algorithm. Journal of IEEE Transactions on Systems, Man, and Cybernetics 1985;15(4):580–5.
[32] Tan PN, Michael S, Vipin K. Introduction to data mining. Addison-Wesley Publishers; 2005. [p. 145–315, chapters 4 and 5].
[33] Duda RO, Hart PE. Pattern classification and scene analysis. Wiley-Interscience; 1973. [p. 56–64].
[34] Friedman N, Geiger D, Goldszmidt M. Bayesian network classifiers. Journal of Machine Learning 1997;29:131–61.
[35] Kohavi R, George JH. Wrappers for feature subset selection. Journal of Artificial Intelligence 1997;97(1–2):273–324. [special issue on relevance].
[36] Kononenko I. Semi-naïve Bayesian classifier. In: Kodratoff Y, editor. Proceedings of the sixth European working session on learning. Berlin, Germany: Springer; 1991. p. 206–19.
[37] Langley P, Sage S. Induction of selective Bayesian classifiers. In: Proceedings of UAI-94, Seattle, WA, USA, 1994. p. 399–406.
[38] Pazzani MJ. Searching for dependencies in Bayesian classifiers. In: Proceedings of AI&STAT'95, 1995. p. 239–48.
[39] Geman S, Bienenstock E, Doursat R. Neural networks and the bias/variance dilemma. Journal of Neural Computation 1992;4:1–58.
[40] Moody JE. The effective number of parameters: an analysis of generalization and regularization in non-linear learning systems. Journal of Advances in Neural Information Processing Systems 1992;4:847–54.
[41] Weigend A. On overfitting and the effective number of hidden units. In: Proceedings of the 1993 connectionist models summer school, 1993. p. 335–42.
[42] Smith M. Neural networks for statistical modeling. ITP New Media Publisher; 1-850-32842-0. [p. 117–29].
[43] Cortes C, Vapnik V. Support-vector networks. Journal of Machine Learning 1995;20(3):273–97.
[44] Cristianini N, John ST. An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press; 2000.
[45] De Vaus D. Analyzing social science data: 50 key problems in data analysis, 1st ed. Sage Publications Ltd.; 2002.
[46] Melnik O. Decision region connectivity analysis: a method for analyzing high-dimensional classifiers. Machine Learning 2002;48:321–51.
[47] Tichy N. An analysis of clique formation and structure in organizations. Administrative Science Quarterly 1973;18(2):194–208.
[48] Karp RM. Reducibility among combinatorial problems. In: Proceedings of the symposium, IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA, 1972. p. 85–103.
[49] Seo J, Shneiderman B. Interactively exploring hierarchical clustering results. Computer 2002;35(7):80–6. July.
[50] Karypis G, Han EH, Kumar V. CHAMELEON: a hierarchical clustering algorithm using dynamic modeling. IEEE Computer 1999;32(8):68–75. August.
[51] Moore AW. K-means and hierarchical clustering. Online tutorial at the following URL: 〈http://www.autonlab.org/tutorials/kmeans.html〉, Carnegie Mellon University, USA, 2010.
[52] Ritter J. An efficient bounding sphere. In: Graphics Gems, 1990. p. 301–3.
[53] Goldberg DE. Genetic algorithms in search, optimization, and machine learning. Addison-Wesley Publishers; 1989.
[54] De Jong K. Genetic algorithms: a 30 year perspective. In: Booker L, Forrest S, Mitchell M, Riolo R, editors. Perspectives on adaptation in natural and artificial systems. Oxford University Press; 2005.

[55] Asuncion A, Newman DJ. UCI-machine learning repository. Website 〈archive.ics.uci.edu/ml/〉, University of California, Irvine, School of Information and Computer Sciences, CA, USA, 2010.
[56] Tin KH, Eugene MK. Building projectable classifiers of arbitrary complexity. In: Proceedings of the 13th international conference on pattern recognition, Vienna, Austria, August, 1996. p. 880–5.
[57] Weiss SM, Kapouleas I. An empirical comparison of pattern recognition, neural nets, and machine learning classification methods. In: Proceedings of the 11th international joint conference on artificial intelligence, Detroit, MI, USA, 1989. p. 781–7.
[58] Artificial Neural Network Toolbox 6.0 and Statistics Toolbox 6.0. Matlab Version 7.0. Website: 〈www.mathworks.com/products/〉.
[59] Smith JW, Everhart JE, Dickson WC, Knowler WC, Johannes RS. Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In: Proceedings of the 12th symposium on computer applications and medical care, Los Angeles, CA, USA, 1988. p. 261–5.
[60] Jankowski N, Kadirkamanathan V. Statistical control of RBF-like networks for classification. In: Proceedings of the seventh international conference on artificial neural networks, ICANN, Lausanne, Switzerland, 1997. p. 385–90.
[61] Au WH, Chan KCC. Classification with degree of membership: a fuzzy approach. In: Proceedings of the first IEEE international conference on data mining, San Jose, CA, USA, 2001. p. 35–42.
[62] Rutkowski L, Cpalka K. Flexible neuro-fuzzy systems. IEEE Transactions on Neural Networks 2003;14:554–74.
[63] Leon IV WD. Enhancing pattern classification with relational fuzzy neural networks and square BK-products. PhD dissertation, Computer Science, Florida State University, FL, USA, 2006. p. 71–4.
[64] Michie D, Spiegelhalter DJ, Taylor CC. Machine learning, neural and statistical classification, series artificial intelligence. Englewood Cliffs, NJ, USA: Prentice Hall; 1994. [p. 157–60, chapter 9].
[65] Kecman V, Arthanari T. Comparisons of QP and LP based learning from empirical data. In: Monostori L, V′ancza J, Ali M, editors. LNCS and LNAI. New York, USA: Springer; 2001. p. 326–32.
[66] Fung G, Mangasarian OL. Proximal support vector machine classifiers. In: Proceedings of the seventh ACM SIGKDD international conference on knowledge discovery and data mining. San Francisco, CA, USA: ACM Press; 2001. p. 77–86.
[67] Domm M, Engel A, Louis PP, Goldberg J. An integer support vector machine. In: Proceedings of the sixth international conference on software engineering, artificial intelligence, networking and parallel/distributed computing, 2005, Towson, MD, USA. p. 144–9.
[68] Shevked Z, Dakovski L. Learning and classification with prime implicants applied to medical data diagnosis. In: Proceedings of the 2007 international conference on computer systems and technologies, Rousse, Bulgaria, June 2007.
[69] Hamilton HJ, Shan N, Cercone N. RIAC: a rule induction algorithm based on approximate classification. Technical report no. CS 96-06, University of Regina, Regina, Canada, 1996.
[70] Ster B, Dobnikar A. Neural Networks in medical diagnosis comparison with other methods. In: Proceedings of the international conference on engineering applications of neural networks, EANN'96, London, UK, 1996. p. 427–30.
[71] Bennet KP, Blue JA. A support vector machine approach to decision trees. Math report, no. 97-100, Rensselaer Polytechnic Institute, Troy, NY, USA, 1997.
[72] Nauck D, Kruse R. Obtaining interpretable fuzzy classification rules from medical data. Artificial Intelligence in Medicine 1999;16:149–69.
[73] Pena-Reyes CA, Sipper M. A fuzzy-genetic approach to breast cancer diagnosis. Artificial Intelligence in Medicine 1999;17:131–55.
[74] Setiono R. Generating concise and accurate classification rules for breast cancer diagnosis. Artificial Intelligence in Medicine 2000;18:205–19.
[75] Abonyi J, Szeifert H. Supervised fuzzy clustering for the identification of fuzzy classifiers. Pattern Recognition Letters 2003;24:2195–207.
[76] Pham DT, Dimov SS, Salem Z. Technique for selecting examples in inductive learning. In: Proceedings of the European symposium on intelligent techniques, ESIT 2000, Aachen, Germany, 2000. p. 119–27.
[77] Cheung N. Machine learning techniques for medical analysis. BSc thesis, School of Information Technology and Electrical Engineering, University of Queensland, Australia, 2001.
[78] Lee YJ, Mangasarian OL. RSVM: reduced support vector machines. In: Proceedings of the first SIAM international conference on data mining, Chicago, IL, USA, 2001.
[79] Lee YJ, Mangasarian OL. SSVM: a smooth support vector machine for classification. Computational Optimization and Applications 2001;20(1): 5–22.
[80] Van GT, Suykens JAK, Lanckriet G, Lambrechts A, De Moor B, Vandewalle J. Bayesian framework for least squares support vector machine classifiers, Gaussian processes and kernel Fisher discriminant analysis. Neural Computation 2002;14:1115–47.
[81] Comaka E, Polatb K, Gunesb S, Arslana A. A new medical decision making system: least square support vector machine (LSSVM) with fuzzy weighting pre-processing. Expert Systems with Applications 2007;32(2):409–14.
[82] Heart S. Website: 〈www.is.umk.pl/projects/datasets-stat.html#Heart〉, August 2008.
[83] Ozşen S, Gunes S. 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 2009;36(1):386–92.
[84] Sakprasat S, Sinclair MC. Classification rule mining for automatic credit approval using genetic programming. In: Proceedings of the IEEE congress on evolutionary computation, Singapore, 2007. p. 548–55.
[85] Statlog Australia Credit Approval. Website: 〈www.is.umk.pl/projects/datasets-stat.html#Australian〉, 8/2009.
[86] Hsu CC, Huang YP, Chang KW. Extended naive Bayes classifier for mixed data. Expert Systems with Applications 2008;35:1080–3.
[87] Huang CL, Chen MC, Wang CJ. Credit scoring with a data mining approach based on support vector machines. Expert Systems with Applications 2007;33(4):847–56.
[88] Nakashima T, Nakai G, Ishibuchi H. Constructing fuzzy ensembles for pattern classification problems. In: Proceedings of the international conference on systems, man and cybernetics, Washington, DC, USA, vol. 4, October 2003. p. 3200–5.
[89] Blachnik M, Duch W. Prototype-based threshold rules. In: Neural Information Processing, LNCS, vol. 4234. Berlin, German: Springer; 2006. [p. 1028–37].
[90] Lei G, Hui-Zhong W, Liang X. A novel classification algorithm based on fuzzy kernel multiple hyperspheres. In: Proceedings of the fourth international conference on fuzzy systems and knowledge discovery, FSKD 2007, Haikou, Hainan, China, vol. 2, 2007. p. 114–8.
[91] Segata N, Blanzieri E. Empirical assessment of classification accuracy of local SVM. Technical report # DISI-08-014, University of Trento, Italy, March 2008.
[92] Gonzalez JA, Holder LB, Cook DJ. Graph-based concept learning. In: Proceedings of the 14th international FAIRS conference, FL, USA, 2001. p. 377–81.
[93] Eggermont J, Kok JN, Kosters WA. Genetic programming for data classification: partitioning the search space. In: Proceedings of the 2004 symposium on applied computing, 2004. p. 1001–5.
[94] Gavrilis D, Tsoulos LG, Dermatas E. Selecting and constructing features using grammatical evolution. Pattern Recognition Letters 2008;29: 1358–65.
[95] Kianmehr K, Alshalalfa M, Alhajj R. Effectiveness of fuzzy discretization for class association rule-based classification. In: LNCS, vol. 4994. Berlin, Germany: Springer; 2008. [p. 298–308].
[96] Ene M. Neural network-based approach to discriminate healthy people from those with Parkinson's disease. Annals of the University of Craiova, Mathematics and Computer Science Series 2008;35:112–6.
[97] Little MA, McSharry PE, Hunter EJ, Spielman J, Ramig LO. Suitability of dysphonia measurements for telemonitoring of Parkinson's disease. IEEE Transactions on Biomedical Engineering 2009.
[98] Kurgan LA, Cios KJ, Tadeusiewicz R, Ogiela M, Goodenday LS. Knowledge discovery approach to automated cardiac SPECT diagnosis. Artificial Intelligence in Medicine 2001;23(2):149–69.
[99] Zhanga Z, Shib Y, Gao G. A rough set-based multiple criteria linear programming approach for the medical diagnosis and prognosis. Expert Systems with Applications 2009;36(5):8932–7.
[100] Thomas JW, Hofer JP. Accuracy of risk-adjusted mortality rate as a measure of hospital quality of care. Medical Care 1999;37(1):83–92.