Please use the following reference regarding this paper:

Pham, N.A. Huy and Triantaphyllou, Evangelos, **"Prediction of Diabetes by Employing a New Data Mining Approach Which Balances Fitting and Generalization,"** Computer and Information Science, a book of the Springer series of books titled: "Studies in Computational Intelligence," (Roger Yin Lee, Editor), Springer, Heidelberg, Germany, Chapter xx, pp. xxx -xxx, 2008.

Prediction of Diabetes by Employing a New Data Mining Approach Which Balances Fitting and Generalization

Huy Nguyen Anh Pham and Evangelos Triantaphyllou
Department of Computer Science,
298 Coates Hall, Louisiana State University, Baton Rouge, LA 70803
Emails: hpham15@lsu.edu and trianta@lsu.edu

Abstract

The Pima Indian diabetes (PID) dataset [1], originally donated by Vincent Sigillito from the Applied Physics Laboratory at the Johns Hopkins University, is one of the most well-known datasets for testing classification algorithms. This dataset consists of records describing 786 female patients of Pima Indian heritage which are at least 21 years old living near Phoenix, Arizona, USA. The problem is to predict whether a new patient would test positive for diabetes. However, the correct classification percentage of current algorithms on this dataset is oftentimes coincidental. The root to the above critical problem is the overfitting and overgeneralization behaviors of a given classification algorithm when it is processing a dataset. Although the above situation is of fundamental importance in data mining, it has not been studied from a comprehensive point of view. Thus, this paper describes a new approach, called the Homogeneity-Based Algorithm (or HBA) as developed by Pham and Triantaphyllou in [2-3], to optimally control the overfitting and overgeneralization behaviors of classification on this dataset. The HBA is used in conjunction with traditional classification approaches (such as Support Vector Machines (SVMs), Artificial Neural Networks (ANNs), or Decision Trees (DTs)) to enhance their classification accuracy. Some computational results seem to indicate that the proposed approach significantly outperforms current approaches.

1. Introduction

Insulin is one of the most important hormones in the body. It aids the body in converting sugar, starches and other food items into the energy needed for daily life. However, if the body does not produce or properly use insulin, the redundant amount of sugar will be driven out by urination. This phenomenon (or disease) is called diabetes. The cause of diabetes is still a mystery, although obesity and lack of exercise appear to possibly play significant roles.

According to the American Diabetes Association [4] in November 2007, 20.8 million children and adults in the United States (i.e., approximately 7% of the population) were diagnosed with diabetes. Thus, the ability to diagnose diabetes early plays an important role for the patient's treatment process. The World Health Organization [5] proposed the eight attributes, depicted in Table 1, of physiological measurements and medical test results for the diabetes diagnosis.

Table 1: The eight attributes for the diabetes diagnosis.

No.	Attribute
1	Number of times pregnant
2	Plasma glucose concentration in an oral glucose tolerance test
3	Diastolic blood pressure (mm/Hg)
4	Triceps skin fold thickness (mm)
5	2-hour serum insulin (μU/ml)
6	Body mass index (kg/m^2)
7	Diabetes Pedigree function
8	Age (years)

Furthermore, one of the many applications of data mining involves the analysis of data for which we know the class value of each data point. We wish to infer some patterns from these data which in turn could be used to infer the class value of new points for which we do not know their class values. For instance, a doctor could be interested in knowing whether a patient would test positive for diabetes based on the above eight attributes. This kind of data mining analysis is called classification or class prediction of new data points.

The PID dataset [1], originally donated by Vincent Sigillito from the Applied Physics Laboratory at the Johns Hopkins University, is one of the most well-known datasets for testing classification algorithms. This dataset consists of records describing 768 female patients of Pima Indian heritage which are at least 21 years old living near Phoenix, Arizona, USA. From the 768 patients in the PID dataset, classification algorithms used a training set with 576 patients and a testing dataset with 192 patients. However, the correct classification percentage of current algorithms on this dataset is oftentimes coincidental.

For instance, Smith et al. in [6] used an early neural network to diagnose the onset of diabetes mellitus. Their approach yielded 76.0% accuracy. Similarly, Jankowski and Kadirkamanathan in [7] developed a radial basis function network suite called IncNet which used 100 neurons and trained for 5,000 iterations. This approach yielded 77.6% accuracy. Au and Chan in [8] attempted to improve the correct classification percentage on the PID dataset by using a fuzzy approach. Au and Chan first represented the revealed regularities and exceptions using linguistic terms, and then mined interesting rules for the classification based on membership degrees. Their approach yielded 77.6% accuracy. Rutkowski and Cpalka in [9] introduced a new neural-fuzzy structure called a flexible neural-fuzzy inference system (FLEXNFIS). Based on the input and output data, they proposed the parameters of the membership functions and the type of the neuron systems (Mamdani or logical). However, their correct classification percentage on the PID dataset was 78.6%. Davis in [10] developed a fuzzy neural network by using the BK-Square products. This fuzzy neural network was then tested on the PID dataset. The result of his approach yielded 81.8% accuracy. Furthermore, the results obtained from the StatLog project [11] when evaluating for many different classification algorithms on the PID dataset showed that their correct classification percentage was less than 78%.

The root to the low accuracies is the overfitting and overgeneralization behaviors of a given classification algorithm when it is processing this dataset. Although the above situation is of fundamental importance in data mining, it has not been studied from a comprehensive point of view. Thus, the main goal of this paper is to apply a new approach, called the Homogeneity-Based Algorithm (or HBA), as described in [2-3], to optimally control the overfitting and overgeneralization behaviors on the PID dataset. That is, the HBA would minimize the total misclassification cost in terms of the false-positive, false-negative, and unclassifiable rates. By doing so, it is hoped that the classification/prediction accuracy of the inferred models will be very high or at least as high as it can be achieved with the available training data.

The next section is a brief description of the HBA and it is adopted from [2-3]. That section shows how a balance between fitting and generalization has the potential to improve many existing classification algorithms. The third section discusses some promising results. These results give an indication of how this methodology may improve the classification/prediction accuracy. Finally, this chapter ends with some conclusions.

2. Description of the HBA

2.1 Problem Description

As described in [2-3], many real-life applications have the following three different penalty costs:

- A cost when a true-positive point is classified as negative.
- A cost when a true-negative point is classified as positive.
- A cost when a data point cannot be classified by any of the classification patterns.

The first case is known as *false-negative*, while the second case is known as *false-positive*. The last case is known as *unclassifiable*. Furthermore, [2-3] showed that attempts to minimize any of the previous rates might affect to the other rates. Thus, we cannot separate the control of fitting and generalization into two independent studies. That is, we need to find a way to simultaneously balance fitting and generalization by adjusting the inferred systems (i.e., the positive and the negative systems) obtained from a classification algorithm. The balance of the two systems will attempt to minimize the total misclassification cost of the final system.

In particular, let us denote C_{FP}, C_{FN}, and C_{UC} as the unit penalty costs for the false-positive, the false-negative, and the unclassifiable cases, respectively. Let *RATE_FP*, *RATE_FN*, and *RATE_UC* be the false-positive, the false-negative, and the unclassifiable

rates, respectively. Then, the problem is to achieve a balance between fitting and generalization that would minimize, or at least significantly reduce, the total misclassification cost denoted as *TC*. Thus, the problem is defined as in the following expression:

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

This methodology may assist the data mining analyst to create classification systems that would be optimal in the sense that their total misclassification cost would be minimized. As mentioned in [2-3], there are two key issues regarding the HBA:

- The accuracy of the inferred classification systems can be increased if the derived patterns are, somehow, more compact and *homogenous*. A pattern *C* of size n_C is a homogenous set if the pattern can be partitioned into smaller bins of the same unit size *h* and the density of these bins is almost equal to each other.
- The accuracy of the inferred classification systems may also be affected by a *density* measure. Such a density could be defined as the number of data points in each inferred pattern per unit of area or volume. Therefore, this density will be called the *homogeneity degree.* Suppose that a homogenous set *C* is given. Then, *HD(C)* will denote its homogeneity degree.

2.2 Non-parametric Density Estimation

As seen in Section 2.1, the density estimation of a typical bin plays an important role in determining whether a set is a homogenous set. One of the most appropriate approaches for the non-parametric density estimation is Parzen Windows [12]. That is, the Parzen Windows approach temporarily assumes that the bin *R* is a *D*-dimensional hypercube of unit size *h*. To find the number of points that fall within this bin, the Parzen Windows approach defines a kernel function $\varphi(u)$ as follows:

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

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

$$\varphi(\frac{x - x_i}{h}) = \prod_{m=1}^{D} \varphi(\frac{x^m - x_i^m}{h}) \quad (3)$$

Let n_C be the number of points in *C* and *d*(x) denote the *x*'s density, then:

$$d(x) \approx \frac{1}{n_C \times h^D} \sum_{i=1}^{n_C} \prod_{m=1}^{D} \varphi(\frac{x^m - x_i^m}{h}) \quad (4)$$

Choosing a value for *h* plays the role of a smoothing parameter in the Parzen Windows approach. We propose a way for finding an appropriate value for *h* as follows:

Heuristic Rule 1: *If h is set equal to the minimum value in set S and this value is used to compute d(x) by using Equation (4), then d(x) approaches to a true density.*

For instance, suppose that we determine all distances between all possible pairs formed by taking any two points from pattern *C* of size 5 (i.e., there are five points in *C*). Thus, there are 10 distances totally. For easy illustration, assume that these distances are as follows: 6, 1, 2, 2, 1, 5, 2, 3, 5, 5. Then, we define *S* as the set of the distances which have the highest frequency. For the previous illustrative example, we have set *S* equal to {2, 5} as both distances 2 and 5 occur with frequency equal to 3 (which is the highest frequency). By using the concept of the previous set *S*, Heuristic Rule 1 proposes an appropriate value for *h* which is equal to 2. The following section briefly provides the key details of the HBA [2-3].

2.3 The HBA

There are five parameters which are used in the HBA and are computed by using a Genetic Algorithm (GA) approach:

- Two expansion threshold values α^+ and α^- to be used for expanding the positive and the negative homogenous sets, respectively.
- Two breaking threshold values β^+ and β^- to be used for breaking the positive and the negative patterns, respectively.
- A density threshold value γ to be used for determining whether either a positive or a negative hypersphere is approximately a homogenous set.

The HBA depicted in Figure 1 is summarized in terms of the following six phases:

- **Phase # 1**: Randomly initialize the threshold values. Assume a training dataset *T* is given. We divide *T* into the two random sub-datasets: T_1 whose size is equal to, say 90%, of *T*'s size and T_2 whose size is equal to 10% of *T*'s size (these percentages are determined empirically).
- **Phase #2**: Apply a classification approach (such as SVMs, ANNs, or DTs) on the training dataset T_1 to infer the two classification systems (i.e., the positive and the negative classification systems). Suppose that each classification system consists of a set of

patterns. Next, break the inferred patterns into hyperspheres.

- **Phase #3**: Determine whether the hyperspheres derived in Phase #2 are homogenous sets or not. If so, then compute their homogeneity degree and go to Phase #4. Otherwise, break a non-homogenous set into smaller hyperspheres. Repeat Phase #3 until all of the hyperspheres are homogenous sets.
- **Phase #4**: For each homogenous set, if its homogeneity degree is greater than a certain breaking threshold value, then expand it. Otherwise, break it into smaller homogenous sets. Phase #4 stops when all of the homogenous sets have been processed.

Input:
- The training dataset T with the positive and the negative points.
- A given classification algorithm.
- The density threshold value γ.

1. Divide T into T_1 and T_2 as described in Phase #1.
2. Randomly initialize the values of the control parameters α^+, α^-, β^+, and β^-.
3. Call **Sub-Problem #1** with the training dataset T_1 to infer the two classification models.
4. Call **Sub-Problem #2** to form the hyperspheres from the inferred patterns.
5. For each hypersphere C, do:
   Call **Sub-Problem #3** with inputs C and γ to determine whether C is a homogenosu set.
   If C is a non-homogenous set, then call **Sub-Problem #4** to break it and go to Step 5.
6. Sort the homogeneity degrees in decreasing order.
7. For each homogenous set C, do:
   If $HD(C) \geq \beta^+$ (for positive sets) or $HD(C) \geq \beta^-$ (for negative sets), then
      Call **Sub-Problem #5** with inputs $HD(C)$ and α^+ or α^- to expand C.
   Else, Call **Sub-Problem #6** to break C.

Notes:
- Apply a GA approach on Steps 5 to 7 by using Equation (1) as the fitness function and T_2 as a calibration dataset to find the classification model S_1 and the optimal threshold values (α^+_*, α^-_*, β^+_*, β^-_*).
- For the unclassifiable points by S_1 in T_2, we use Steps 3 to 7 with the optimal threshold values (α^+_*, α^-_*, β^+_*, β^-_*) to infer the additional classification model S_2.

10. Let $S = S_1 \cup S_2$.

Output: A new classification system S.

Figure 1: The HBA.

- **Phase #5**: Evaluate the classification models (i.e., the homogenous sets processed in Phase #4) by using the dataset T_2 as a calibration dataset. The evaluation returns the value of Equation (1). Next, apply a genetic algorithm (GA) with the expression in Equation (1) as the fitness function to find the new threshold values (α^+, α^-, β^+, β^-) and then go to Phase #4. After a number of iterations, Phase #5 returns the optimal threshold values (α^+_*, α^-_*, β^+_*, β^-_*) and the classification model S_1 (i.e., the positive and the negative classification models) with the best value for Equation (1).

Phase #6: Suppose that the calibration dataset T_2 can be divided into the two sub-datasets: $T_{2,1}$, which consists of the points classified by S_1, and $T_{2,2}$, which includes the unclassifiable points by S_1. We apply Phases #2 to #4 on the sub-dataset $T_{2,2}$ with the optimal threshold values (α^+_*, α^-_*, β^+_*, β^-_*). This phase infers the additional classification model S_2. The final classification model is the union of S_1 and S_2.

The six phases described above lead to the formulation of six sub-problems as follows:

- **Sub-Problem #1:** Apply a data mining approach to infer the two classification systems.
- **Sub-Problem #2:** Break the inferred patterns into hyperspheres.
- **Sub-Problem #3:** Determine whether a hypersphere is a homogenous set or not. If so, then its homogeneity degree is estimated.
- **Sub-Problem #4:** If a hypersphere is not a homogenous set, then break it into smaller hyperspheres.
- **Sub-Problem #5:** Expand a homogenous set C by using $HD(C)$ and the corresponding expansion threshold value plus some stopping conditions.
- **Sub-Problem #6:** Break a homogenous set C into smaller homogenous sets.

To solve Sub-Problem #1, one simply applies a traditional classification algorithm and then derives the classification patterns. Furthermore, a solution to Sub-Problem #2 is similar to solutions for Sub-Problem #4. Therefore, the following sections present some summary procedures for solving Sub-Problems #2, #3, #5, and #6 and the GA approach. The explanation and the illustrative examples for Sub-Problems #1 to #6 are described in more detail in [2-3].

2.4 Solving Sub-Problem #2

The goal of Sub-Problem #2 is to find the minimum number of hyperspheres that can cover a pattern C of size n_C. A heuristic algorithm for Sub-Problem #2 is proposed as depicted in Figure 2.

The algorithm starts by first estimating the densities of the n_C points by using Equation (4). Assume that the value for K is going from 1 to n_C. The algorithm will

pick K points in C with the highest densities. Next, it uses these K points as centroids in the K-means clustering approach. If the K hyperspheres which are obtained from the clustering approach cover C, then the algorithm will stop. Otherwise, we repeat the algorithm with the value for K increased by one. Obviously, the algorithm will stop after some iterations because a hypersphere of size one is a homogenous set.

Input: Pattern C of size n_C.
1. Estimate the densities of the n_C points by using Equation (4).
2. For K=1 to n_C do
   Pick K points in C with the highest densities.
   Use the K-means clustering approach to find K hyperspheres.
   If the K hyperspheres cover C, then STOP.
   Else, $K = K + 1$.

Output: K hyperspheres.

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

2.5 Solving Sub-Problem #3

Let us consider hypersphere C of size n_C. Sub-Problem #3 determines whether or not hypersphere C is a homogenous set as follows. Hypersphere C is first divided into a number of small bins of unit size h and then approximates the density at the center x of each bin. If the densities at the centers are approximately equal to each other, then C is a homogenous set.

A softer condition can be applied instead of requiring exactly the same density at the centers of the bins. That is, if the standard deviation of the densities at the centers of the bins is approximately less or equal to γ, say for $\gamma = 0.01$, then hypersphere C may be considered to be a homogenous set. The algorithm for Sub-Problem #3 is given in Figure 3.

Input: Hypersphere C and density threshold value γ.
1. Compute the distances between all pairs of points in C.
2. Let h be the distance mentioned in Heuristic Rule 1.
3. Superimpose C into hypergrid V of unit size h.
4. Approximate the density at the center x of each bin.
5. Compute the standard deviation of the densities at the centers of the bins.
6. If the standard deviation is less than or equal to γ, then
   C is a homogenous set and its homogeneity degree *HD(C)* is computed by using Equation (5).
   Else, C is not a homogenous set.

Output: Decide whether C is a homogenous set.

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

As seen above, the homogeneity degree *HD(C)* is a factor that may affect the total misclassification cost of the inferred classification systems. If an unclassified point is covered by a homogenous set C which has a higher homogeneity degree, then it may more accurately be assumed to be of the same class as the points covered by the homogenous set C. Thus, a definition for *HD(C)* is an important step in improving the accuracy of the classification systems. Pham and Triantaphyllou in [2-3] have proposed a way for computing *HD(C)* as follows:

$$HD(C) = \frac{\ln(n_C)}{h}. \qquad (5)$$

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

2.6 Sub-Problem #5

Suppose that we are given a positive homogenous set F with its homogeneity degree *HD(F)*, the breaking threshold value β^+, and the expansion threshold value α^+. A similar definition exists for a negative homogenous set. According to the main algorithm depicted in Figure 1, if *HD(F)* is greater than or equal to β^+, then the homogenous set F will be expanded by using the expansion threshold value α^+. Otherwise, we will break the homogenous set F into smaller hyperspheres.

There are two types of expansion for F: a radial expansion in which a homogenous set F is expanded in all directions and a linear expansion in which a homogenous set F is expanded in a certain direction. The following section explains in detail these two expansion types [2-3].

2.6.1 Radial Expansion

Let M be a region expanded from F. Let R_F and R_M denote the radiuses of F and M, respectively. The radial expansion algorithm is depicted in Figure 4.

The idea of this algorithm is to expand a homogeneous set F as much as possible by using a dichotomous search methodology. That is, R_F is increased by a certain amount denoted as T, called a *step-size increase,* in each iteration. Thus, one gets:

$$R_M = R_F + T. \quad (6)$$

Input: Homogenous set F with $HD(F)$, R_F, and α^+
1. Set $M = F$ (i.e., $R_F = R_M$).
2. Set hypersphere G covering M with radius $R_G = 2 \times R_M$.
3. Repeat
   Set $E = M$ (i.e., $R_E = R_M$).
   Expand M by using Equation (10).
   Until (R_M satisfies stopping conditions discussed in Section 2.6.3 or $R_M = R_G$).
4. If R_M satisfies stopping conditions, then STOP.
   Else, go to Step 2.

Output: An expanded region E.

Figure 4: The algorithm for the radial expansion.

A value for T is determined as follows. We first assume that there exists a hypersphere G which covers the homogenous set F. Without loss of generality, let us assume that the radius R_G may be computed by:

$$R_G = 2 \times R_F \quad (7)$$

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

$$T = \frac{R_G - R_F}{2} \quad (8)$$

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

$$T = \frac{R_G - R_F}{2} \times \frac{1}{L \times HD(F)} \quad (9)$$

If we substitute back into Equation (6), R_M becomes:

$$R_M = R_F + \frac{R_G - R_F}{2} \times \frac{1}{L \times HD(F)} \quad (10)$$

2.6.2 Linear Expansion

The linear approach expands a homogenous set F in a certain direction. There is a difference between the method presented in the previous section and the one presented in this section (i.e., linear vs. radial). That is, now the homogenous set F is first expanded to hypersphere M by using the radial expansion. Then, hypersphere M is expanded in a given direction by using the radial approach until it satisfies the stopping conditions mentioned next in Section 2.6.3. The final region is the union of all the expanded regions.

2.6.3 Description of the Stopping Conditions

As described in [2-3] the stopping conditions of the radial expansion approach for homogenous set F of size n_F must satisfy the following requirements:

- Depend on the homogeneity degree $HD(F)$. This has been mentioned in Section 2.1.
- Stop when an expanded region reaches other patterns. However, this condition can be relaxed by accepting several noisy data points in the expanded region. If the homogeneity degree $HD(F)$ is high, then the expanded region can accept more noisy data.

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

$$R_M \le HD(F) \times R_F \times \alpha^+ \text{ and}$$
$$\textit{the number of noisy points} \le \frac{HD(F) \times \alpha^+}{n_F} \quad (11)$$

Similar conditions exist for the expansion threshold value α^-.

2.7 Sub-Problem #6

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

2.8 A Genetic Algorithm (GA) Approach for Finding the Threshold Values

Recall that the main algorithm depicted in Figure 1 uses the four threshold values α^+, α^-, β^+, and β^- to derive a new classification system. If the breaking threshold values (i.e., β^+, and β^-) are too high, then this would result in the overfitting problem. On the other hand, too low breaking threshold values may not be sufficient to overcome the overgeneralization problem. The opposite situation is true with the expansion threshold values (i.e., α^+ and α^-).

Since the ranges for the threshold values depend on each individual application, the search space may be large. In this investigation an exhaustive search would be impractical. Thus, we propose to use a GA

approach to find approximate optimal threshold values as follows. The HBA uses Equation (1) as the fitness function and the dataset T_2 as a calibration dataset. The GA approach has been applied here because Equation (1) is not unimodal. Furthermore, each chromosome consists of four genes corresponding to the four threshold values (α^+, α^-, β^+, β^-) as depicted in Figure 5. The initial population size is 20 (this size was determined empirically).

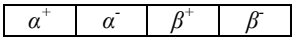

Figure 5: An illustrative example of a chromosome consisting of the four genes.

The algorithm creates the crossover children by combining pairs of parents in the current population. At each coordinate of the child's chromosome, the crossover function randomly selects the gene at the same coordinate from one of the two parents and assigns it to the child.

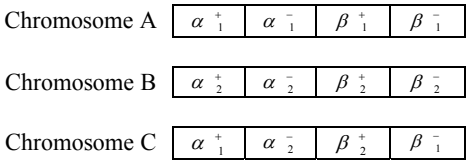

Figure 6: An illustrative example of the crossover function.

In order to help motivate the crossover function, we consider the two chromosomes A and B depicted in Figure 6. Assume that the chromosomes A and B consist of the four genes $(\alpha_1^+, \alpha_1^-, \beta_1^+, \beta_1^-)$ and $(\alpha_2^+, \alpha_2^-, \beta_2^+, \beta_2^-)$, respectively. The algorithm randomly selects the gene at the same coordinate from one of the chromosomes A and B and then assigns it to child C. Thus, the chromosome C may be $(\alpha_1^+, \alpha_2^-, \beta_2^+, \beta_1^-)$.

The algorithm creates the mutation child (g_1, g_2, g_3, g_4) by randomly changing the genes of the parent chromosome (α^+, α^-, β^+, β^-). Suppose that the first two genes α^+ and α^- are in the range [*a, b*], while the last two genes β^+ and β^- are in the range [*c, d*]. The algorithm first randomizes a chromosome (t_1, t_2, t_3, t_4) by using the Gaussian distribution. Next, one would prefer that the genes in the mutation child are also in the corresponding ranges. Thus, for each gene at the same coordinate from the parent, the algorithm uses either one of the following Equations (12) or (13) to create the corresponding gene for the mutation child:

$g_1 = ((\alpha^+ \text{ or } t_1) \text{ or } a)$ and b, $g_2 = ((\alpha^- \text{ or } t_2) \text{ or } a)$ and b (12)

$g_3 = ((\beta^+ \text{ or } t_3) \text{ or } c)$ and d, $g_4 = ((\beta^- \text{ or } t_4) \text{ or } c)$ and d (13)

In order to help motivate the mutation function, let us consider a parent chromosome, say, (2, 1, 5, 7). Assume that (α^+, α^-) are in the range [0, 3], while (β^+, β^-) are in the range [0, 10]. Also suppose that the chromosome, which is created by using the Gaussian distribution, is (10, 6, 3, 7). The mutation child is presented in Figure 7. The GA stops if there is no improvement in the fitness function during successive iterations.

g_1	g_2	g_3	g_4
((2 or 10) or 0) and 3 = 2	((1 or 6) or 0) and 3 = 3	((5 or 3) or 0) and 10 = 2	((7 or 7) or 0) and 10 = 2

Figure 7: An illustrative example of the mutation function.

3. A Computational Study

3.1 Experimental methodology

From the 768 patients, the HBA divided the PID dataset into a training dataset *T* with 576 patients and a testing dataset with 192 patients.

Suppose that we are given a certain 3-tuple of the unit penalty costs (C_{FP}, C_{FN}, C_{UC}). The experiments were done as follows:

Step 1: The original algorithm was first trained on the training dataset *T* and then derived the value for *TC* by using the testing dataset.

Step 2: The HBA was trained on the training dataset T_1 as described in Section 2.2 and then derived the value for *TC* by also using the testing dataset. It was assumed that β^+ and β^- were in [0, 2] while α^+ and α^- were in [0, 20].

Step 3: Compare the two values for *TC* returned in steps 1 and 2, respectively.

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

3.2 Experimental Results

The experiments were run on a PC with 2.8GHZ speed and 3GB RAM under the Windows XP operating system. The original classification algorithms used in these experiments are based on SVMs, ANNs, and DTs. There were more than 54 experiments done on the PID dataset with different values for the 3-tuple (C_{FP}, C_{FN}, C_{UC}). Furthermore, we used the libraries in Neural Network Toolbox 6.0, Genetic Algorithm and Direct Search Toolbox 2.1, and Statistics Toolbox 6.0 [13] for implementing the

classification algorithms, the GA approach, and the density estimation approach. The experimental details are as follows:

Case 1: At first we studied the case of a 3-tuple (C_{FP}, C_{FN}, C_{UC}) in which the application would not penalize for the unclassifiable cases while the application would penalize at the same cost, say one unit, for the other two types of error. Under this scenario, the problem is equivalent to the evaluation of the current classification algorithms which require either positive or negative outputs (see Table 4). Thus, the objective function in this case was assumed to be:

$$TC = 1 \times RATE\_FP + 1 \times RATE\_FN.$$

Table 2: Results for minimizing $TC = 1 \times RATE\_FP + 1 \times RATE\_FN$ on the PID dataset.

Algorithm	*RATE_FP*	*RATE_FN*	*RATE_UC*	*TC*	% of improvement
SVM	0	74	0	74	
DT	27	36	0	63	
ANN	22	39	0	61	
SVM-HBA	0	10	0	10	86.49%
DT-HBA	0	16	0	16	74.60%
ANN-HBA	0	10	0	10	83.61%

The results are presented in Table 2. In this case, Table 2 shows the three rates and the value of *TC* obtained from the algorithms. The notation "SVM-HBA" means that the HBA used the classification models first obtained by using the SVM algorithm before controlling the fitting and generalization problems. Two similar notations are used for DT-HBA (the Decision Tree algorithm and the HBA) and ANN-HBA (the Artificial Neural Network algorithm and the HBA). Table 2 presents that after 100 generations, SVM-HBA, DT-HBA, and ANN-HBA found the optimal *TC* to be equal to 10, 16, and 10 units, respectively. These values of *TC* were less than the average value of *TC* achieved by the original algorithms (i.e., the SVM, DT, and ANN) by about 81.57%. The values for α^+, α^-, β^+, and β^- when ANN-HBA found the optimal *TC* by using the GA approach are 0.39, 18, 0.23, and 0.35, respectively.

Table 3: Results for the PID dataset.

Algorithm	% Accuracy	% of improvement
[6]	76.0%	
[7]	77.6%	
[8]	77.6%	
[9]	78.6%	
[10]	81.8%	
[11]	77.7%	
SVM-HBA	94.79%	16.57%
ANN-HBA	94.79%	16.57%
DT-HBA	91.67%	13.45%

Table 3 presents a comparison between the achieved classification percentages of the different classification algorithms. Clearly, the results by the HBA when it was combined with the traditional approaches were more accurate than those by the stand alone algorithms.

Case 2: Now we consider a case in which the application would penalize the same way, say three units, for the false-positive, the false-negative, and the unclassifiable cases. Thus, the objective function in this case was assumed to be:

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

The results are presented in Table 4. In this case, Table 4 shows that after 100 generations, SVM-HBA, DT-HBA, and ANN-HBA found the optimal value for *TC* which was less than the value of *TC* achieved by the original algorithms by about 50.48%.

Table 4: Results for minimizing $TC = 3 \times RATE\_FP + 3 \times RATE\_FN + 3 \times RATE\_UC$ on the PID dataset.

Algorithm	*RATE_FP*	*RATE_FN*	*RATE_UC*	*TC*	% of improvement
SVM	0	74	109	549	
DT	27	36	118	543	
ANN	22	39	118	537	
SVM-HBA	2	40	54	288	47.54%
DT-HBA	1	61	24	258	52.49%
ANN-HBA	1	57	29	261	51.40%

Case 3: Now we consider a case in which the application would penalize much more for the false-negative cases than for the other types of error. Thus, the objective function in this case was assumed to be:

$$TC = 1 \times RATE\_FP + 20 \times RATE\_FN + 3 \times RATE\_UC.$$

The results are presented in Table 5. In this case, Table 5 shows that after 100 generations, SVM-HBA, DT-HBA, and ANN-HBA found the optimal value for *TC* which was less than the value of *TC* achieved by the original algorithms by about 51.59%.

Table 5: Results for minimizing $TC = 1 \times RATE\_FP + 20 \times RATE\_FN + 3 \times RATE\_UC$ on the PID dataset.

Algorithm	*RATE_FP*	*RATE_FN*	*RATE_UC*	*TC*	% of improvement
SVM	0	74	109	1,807	
DT	27	36	118	1,101	
ANN	22	39	118	1,156	
SVM-HBA	0	16	105	635	64.86%
DT-HBA	5	10	136	613	44.32%
ANN-HBA	0	10	143	629	45.59%

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

$TC = 20 \times RATE\_FP + 2 \times RATE\_FN + 1 \times RATE\_UC$,

$TC = 20 \times RATE\_FP + 20 \times RATE\_FN + 1 \times RATE\_UC$, $TC = 20 \times RATE\_FP + 1 \times RATE\_FN + 20 \times RATE\_UC$, $TC = 1 \times RATE\_FP + 20 \times RATE\_FN + 20 \times RATE\_UC$, and $TC = 3 \times RATE\_FP + 6 \times RATE\_FN$.

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

4. Conclusions

Millions of people in the United States and the world have diabetes. Many of these people do not even know they have it. The ability to predict diabetes early plays an important role for the patient's treatment process. However, the correct prediction percentage of current algorithms is oftentimes low. Thus, this chapter applied a new approach, called the Homogeneity-Based Algorithm (HBA), for enhancing the diabetes prediction. That is, the HBA is first used in conjunction with traditional classification approaches (such as SVMs, DTs, ANNs). A GA approach was then used to find optimal (or near optimal) values for the four parameters of the HBA. The Pima Indian diabetes dataset was used for evaluating the performance of the HBA. The obtained results appear to be very important both for accurately predicting diabetes and also for the data mining community, in general.

References

[1] Asuncion A. and D.J. Newman, "*UCI-Machine Learning Repository,*" University of California, Irvine, California, USA, School of Information and Computer Sciences, 2007.

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

[3] H. N. A. Pham, and E. Triantaphyllou, "*An Optimization Approach for Improving Accuracy by Balancing Overfitting and Overgeneralization in Data Mining*," submitted for publication, January 2008.

[4] American Diabetes Association, website: http://www.diabetes.org/home.jsp, 2007.

[5] World Health Organization, "*Diabetes Mellitus: Report of a WHO Study Group. Geneva: WHO,*" Technical Report Series 727, 1985.

[6] J. W. Smith, J. E. Everhart, W. C. Dickson, W. C. Knowler, and R. S. Johannes, "*Using the ADAP learning algorithm to forecast the onset of diabetes mellitus,*" Proceedings of 12th Symposium on Computer Applications and Medical Care, Los Angeles, California, USA, 1988, pp. 261 - 265.

[7] N. Jankowski and V. Kadirkamanathan, "*Statistical control of RBF-like networks for classification,*" Proceedings of the 7th International Conference on Artificial Neural Networks (ICANN), Lausanne, Switzerland, 1997, pp. 385 - 390.

[8] W. H. Au and K. C. C. Chan, "*Classification with degree of membership: A fuzzy approach,*" Proceedings of the 1st IEEE Int'l Conference on Data Mining, San Jose, California, USA, 2001, pp. 35 - 42.

[9] L. Rutkowski and K. Cpalka, "*Flexible neuro-fuzzy systems,*" IEEE Transactions on Neural Networks, vol. 14, 2003, pp. 554 - 574.

[10] W. L. Davis IV, "*Enhancing Pattern Classification with Relational Fuzzy Neural Networks and Square BK-Products*," PhD Dissertation in Computer Science, 2006, pp. 71 - 74.

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

[12] Duda, R. O., and P. E. Hart, "*Pattern Classification and Scene Analysis,*" Wiley Publisher, 1973, pp. 56-64.

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