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

Note: Please check Dr. Triantaphyllou's web page (under the section on Conference Proceedings papers) for the complete reference to this paper (URL: http://www.csc.lsu.edu/trianta)

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 diagnose 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], to optimally control the overfitting and overgeneralization behaviors of classification on this dataset. The HBA is used in conjunction with traditional classification approaches 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 into 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 [3] in November 2007, 20.8 million children and adults in the United States (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 [4] 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 [5] used an early neural network to diagnose the onset of diabetes mellitus. Their approach yielded 76.0% accuracy. Similarly, Jankowski and Kadirkamanathan in [6] 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 [7] 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 [8] 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 [9] 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 [10] 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], 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. 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 paper ends with some conclusions.

2. Description of the HBA

2.1 Problem Description

As described in [2], 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] 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], 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* 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.

The following section provides the key details of the HBA.

2.2 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 Support Vector Machines (SVMs), Artificial Neural Networks (ANNs), or Decision Trees (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.
- **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 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.

```
Input:
  ▪ The training dataset $T$ with the positive and the negative points.
  ▪ A given classification algorithm.
  ▪ The density threshold value $\gamma$.
1. Divide $T$ into $T_1$ and $T_2$ as described in Phase #1.
2. Randomly initialize the values of the control parameters $\alpha^{+}$, $\alpha^{-}$, $\beta^{+}$, and $\beta^{-}$.
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 $\gamma$ 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 $\alpha^{+}$ or $\alpha^{-}$ 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 ($\alpha^{+}_{*}$, $\alpha^{-}_{*}$, $\beta^{+}_{*}$, $\beta^{-}_{*}$).
  • For the unclassifiable points by $S_1$ in $T_2$, we use Steps 3 to 7 with the optimal threshold values ($\alpha^{+}_{*}$, $\alpha^{-}_{*}$, $\beta^{+}_{*}$, $\beta^{-}_{*}$) 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.

The six phases described earlier 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.

The algorithms and the illustrative examples for Sub-Problems #1 to #6 are described in more detail in [2]. The following section presents the GA approach.

2.3 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. Therefore, 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. Furthermore, each chromosome consists of four genes corresponding to the four threshold values (α^+, α^-, β^+, β^-) as depicted in Figure 2. The initial population size is 20 (this size was determined empirically).

α^+	α^-	β^+	β^-

Figure 2: 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.

In order to help motivate the crossover function, we consider the two chromosomes A and B depicted in Figure 3. 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 ($\alpha_1^+, \alpha_2^-, \beta_2^+, \beta_1^-$).

Chromosome A	α_1^+	α_1^-	β_1^+	β_1^-
Chromosome B	α_2^+	α_2^-	β_2^+	β_2^-
Chromosome C	α_1^+	α_2^-	β_2^+	β_1^-

Figure 3: An illustrative example of the crossover function.

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 (2) or (3) to create the corresponding gene for the mutation child:

g_1 = ((α^+ or t_1) or a) and b, g_2 = ((α^- or t_2) or a) and b (2)

g_3 = ((β^+ or t_3) or c) and d, g_4 = ((β^- or t_4) or c) and d (3)

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 4. 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 4: 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 which had the same number of points from each class.
- 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 [11] 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:

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

Table 2: Results for minimizing *TC = RATE_FP + RATE_FN* on the PID dataset.

Algorithm	*RATE_FP*	*RATE_FN*	*RATE_UC*	*TC*
SVM	0	74	0	74
DT	27	36	0	63
ANN	22	39	0	61
SVM-HBA	0	10	0	10
DT-HBA	0	16	0	16
ANN-HBA	0	10	0	10

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 13 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 76%. 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
[5]	76.0%
[6]	77.6%
[7]	77.6%
[8]	78.6%
[9]	81.8%
[10]	77.7%
SVM-HBA	94.79%
ANN-HBA	94.79%
DT-HBA	91.67%

Table 3 presents a comparison between the achieved classification percentages of the different classification algorithms. Clearly, the HBA based on the 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%.

Table 4: Results for minimizing *TC = 3×RATE_FP + 3×RATE_FN +3×RATE_UC* on the PID dataset.

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

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:

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

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 57%.

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

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

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

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

The performance of a classification method in terms of the false-positive, the false-negative, and the unclassifiable rates may be totally unpredictable and depends on the application at hand. Attempts to minimize one of the previous rates lead to increases on the other rates. The root to the above critical problems is the overfitting and overgeneralization behaviors of a given classification approach when it is processing a particular dataset. This paper identified a gap between fitting and generalization with current algorithms and also defined the desired goal as an optimization problem. Next, it applied a new approach, called the Homogeneity-Based Algorithm (HBA). A GA approach was used to find optimal (or near optimal) values for the four parameters. The HBA is used in conjunction with traditional classification approaches (such as SVMs, DTs, ANNs, etc) to enhance their classification accuracy. The HBA was evaluated on the Pima Indian diabetes dataset. The obtained results appear to be very promising.

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

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

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

[10] 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.

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