韓國經營科學會誌
第25卷 第4號
2000年 12月

2차 함수를 이용한 비 선형 패턴인식 알고리즘 구축

김 락 상*

Construction of A Nonlinear Classification Algorithm Using Quadratic Functions

Lark Sang Kim*

Abstract

This paper presents a linear programming based algorithm for pattern classifications. Pattern classification is being considered to be critical in the area of artificial intelligence and business applications. Previous methods employing linear programming have been aimed at two-group discrimination with one or more linear discriminant functions. Therefore, there are some limitations in applying available linear programming formulations directly to general multi-class classification problems. The algorithm proposed in this manuscript is based on quadratic or polynomial discriminant functions, which allows more flexibility in covering the class regions in the N-dimensional space. The proposed algorithm is compared with other competitive methods of pattern classification in experimental results and is shown to be competitive enough for a general purpose classifier.

1. Introduction

This paper presents a new algorithm based on linear programming for pattern classification. The linear programming methods were introduced as a nonparametric alternative to Fisher's linear discrimination method. Several linear programming models using linear discriminant function have been tried by Freed and Glover(1986), Koehler & Erenguc(1990), and Rubin(1991). However, these methods have some limitations in applying directly to general multi-class classification problems.

The following notation is used in this paper. An input pattern is represented by the N-di-

* 청주대학교 경영정보학과 조교수

mensional vector $\mathbf{x}$, $\mathbf{x} = (x_1, x_2, \cdots, x_N)$. The pattern space, which is the set of all possibles that $\mathbf{x}$ may assume, is represented by Ωx . K denotes the total number of classes. The method is for supervised learning where the training set $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n$ is a set of sample patterns with known classification.

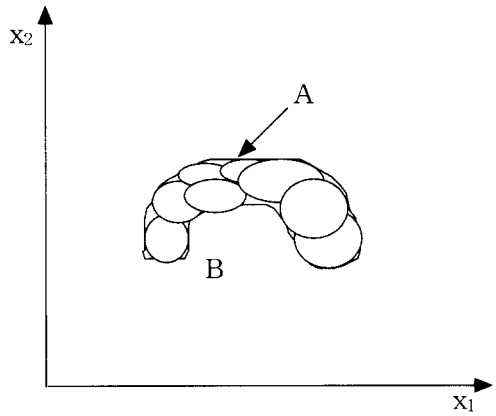
The paper is organized as follows. Section 2 briefly discusses the basic linear programming formulation ideas. In Section 3, the new outlier detection procedure is presented. The two-phase algorithm is summarized in Section 4. Section 5 has experimental results on several well-known applications.

2. Basic ideas of linear programming method for covering class regions

Linear programming models have been used in many classification algorithms. Significant contributions include those by Glover, Keene, and Duea(1988), Freed and Glover(1986), and Rubin (1991). Most of them derive hyperplanes to separate classes.

One of the fundamental ideas in pattern classification is to draw proper boundaries to separate the class regions. This method uses the idea of masking or covering a class region. Any complex nonconvex region can be covered by a set of elementary convex forms of varying size, such as hyperspheres and hyperellipsoids in the N-dimensional space (see [Figure 1]). The idea of using elementary convex covers is not new in pattern classification.

As with hypersphere and other classifiers, there may exist overlap among the elementary convex covers in order to provide complete and adequate coverage of a class region.



[Figure 1] Covering a nonconvex class region such as A by elementary masks

2.1 Classification of Input Patterns Using Masks(Discriminant Functions)

Let p elementary covers or masks (henceforth generally referred to as masks) be required to cover a certain class P region. To classify an input pattern as being in class P, it is necessary to determine if it falls within the area covered by one of the p masks. if the pattern space is two-dimensional and one of the p masks is a circle centered at (a, b) with a radius r, a simple discriminant (masking) function can be created to determine if an input pattern falls in the territory of the designated mask. Let $f(x_1, x_2) = r^2 - [(x_1-a)^2 + (x_2-b)^2]$ be the masking function for this circular mask. If (x_1, x_2) is an input pattern, and

if $f(x_1, x_2) >= 0$, then (x_1, x_2) is inside this mask and (x_1, x_2) belongs to class P;
if $f(x_1, x_2) < 0$, then (x_1, x_2) is not inside this mask and other masks will have

to be tested before a conclusion can be reached as to the classification of (x_1, x_2). A similar masking function for an elliptical mask will be $f(x_1, x_2) = 1 - [(x_1-a)^2/c^2 + (x_2-b)^2/d^2]$ in the usual notation.

In constructing a masking function $f(\mathbf{x})$, the procedure requires $f(\mathbf{x})$ to be at least slightly positive($f(\mathbf{x}) >= \varepsilon$) for input patterns covered by the mask and at least slightly negative ($f(\mathbf{x}) <= -\varepsilon$) for those not covered. So, in general, let p_k be the number of masking functions required to cover class k, k = 1 ··· K. Let $f_1^k(\mathbf{x}), \cdots, f_{pk}^k(\mathbf{x})$ denote these masking functions for class k. Then an input pattern $\mathbf{x}$ will belong to class j if and only if one or more of its masks is at least slightly positive, i.e., equal to or above a small positive threshold value ε, and the masks for all other classes are at least slightly negative, i.e., below a threshold value $-\varepsilon$. Each mask will have its own threshold value as determined during its construction. Expressed in mathematical notation, an input pattern $\mathbf{x}$ is in class j, if and only if

$$f_i^j(\mathbf{x}) >= \varepsilon_i^j \text{ for at least one mask } i,\ i=1 \cdots p_j, \text{ and}$$
$$f_i^k(\mathbf{x}) < -\varepsilon_i^k \text{ for all } k=j \text{ and } i=1 \cdots p_k. \quad (1)$$

If all masks are at least slightly negative (i. e., below their individual $-\varepsilon$ thresholds), the input pattern can not be classified, unless one uses a nearest mask notion. If masks from two or more classes are at least slightly positive, then also the input pattern can not be classified, unless once again one uses a "interior most" in a mask or similar notion. In the second situation, an indication can be obtained, however, about the possible contenders. Such cases would possibly arise when the set of input patterns or the nature of the class regions does not prevent the formation of overlapped masks belonging to different classes.

2.2 Construction of Masking Functions

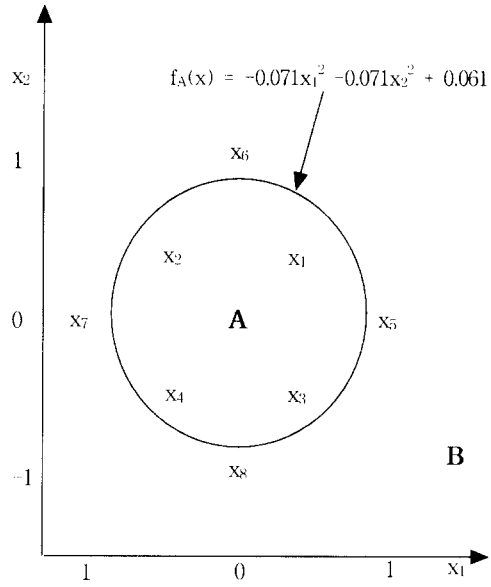
A variety of elementary convex masks can be used to cover a class region. A quadratic function

$$f(\mathbf{x}) = \sum_{i=1}^{N} a_i x_i + \sum_{i=1}^{N}\sum_{j=1}^{N} b_{ij} x_i x_j + c, \quad \text{where } \mathbf{x}=(x_1, x_2, \cdots, x_N) \quad (2)$$

can generate hyperellipsoids, hyperspheres, etc., as masks in the N-dimensional case. It can also generate nonconvex shapes (masks) which is acceptable as long as they help to cover a class region properly. A quadratic function is used here as the standard masking function. If N, the dimension of the pattern vector $\mathbf{x}$, is large, the cross-product terms are usually dropped or only a few used. The algorithm basically determines the coefficients a_i, b_{ij}, c (i=1 ··· N, j=1 ··· N) of these masks. In constructing masks, it is ensured that the generated mask covers only sample patterns of its designated class and not of others. Next is explained the construction of these masks ; that is, how to determine the number of masks required and how to solve for the parameters (coefficients) of a masking function.

Consider a simple two class problem shown in [Figure 2] in which class A is bounded by a circle centered at the origin and class B is the rest of the two-dimensional space. A total of 8 sample patterns have been taken as the training set - 4 patterns from class A and 4 patterns from class B - as shown in the figure.

A priori, it is not known how many elementary masks will suffice for any of the class regions. Thus, an attempt is first made to define a single elementary mask which will cover the whole class region. If this fails, the sample patterns in that



[Figure 2] Two classes, A and B, of input patterns

class are generally split into two or more clusters (using a clustering procedure that can produce a prespecified number of clusters) and then attempts are made to define separate masks for each of these clusters. If that should fail, or if only some are masked, then the unmasked clusters are further split for separate masking until masks are provided for each ultimate cluster. The general idea is to define as large a mask as possible to include as many of the sample patterns within a class in a given mask as is feasibly possible, thereby minimizing the total number of masks required to cover a given region. When it is not feasible to cover with a certain number of masks, the region is subdivided into smaller pieces and masking is attempted for each piece. That is, the unmasked sample patterns are successively subdivided into smaller clusters for masking. At any stage of this iterative procedure, there will be a number of clusters to be masked. It might be feasible to mask some of them, thereby necessitating the breakup only of the remaining unmasked clusters. This "divide and conquer" procedure is a heuristic procedure. One can explore many variations of it, some of which are discussed later.

Going back to the example, one first tries to mask class A with a single masking function. Let a mask of the form

$$f_A(\mathbf{x}) = a_1x_1 + a_2(2)x_2 + b_{11}x_1^2 + b_{12}x_1*x_2 + b_{22}x_2^2 + c \qquad (3)$$

be tried such that for input patterns in class A, $f_A(\mathbf{x})$ is at least slightly positive. As noted before, the masks are constructed so that they are slightly positive for the boundary points in the mask. This ensures a finite separation between the classes and prevents the formation of common boundaries. To determine the parameters, a_1, a_2, b_{11}, b_{12}, b_{22}, and c of the masking function in eqn (3), a linear program is set up which essentially states the following : "construct a masking function such that sample patterns from class A are slightly positive and those from class B are at least slightly negative." A linear programming model can be used because polynomials are linear functions of their parameters. The LP set up in this case is as follows :

Minimize ε

s. t.

$$f_A(\mathbf{x}_1) >= \varepsilon$$
$$f_A(\mathbf{x}_2) >= \varepsilon$$
$$f_A(\mathbf{x}_3) >= \varepsilon$$
$$f_A(\mathbf{x}_4) >= \varepsilon$$
$$f_A(\mathbf{x}_5) <= -\varepsilon$$
$$f_A(\mathbf{x}_6) <= -\varepsilon$$
$$f_A(\mathbf{x}_7) <= -\varepsilon$$

$f_A(\mathbf{x}_8) <= -\varepsilon$

$\varepsilon >=$ a small positive constant. (4)

Generally, a lower bound of 0.01 is used for ε. In terms of the masking function parameters, the LP will be as follows :

Minimize ε

s. t.

$0.4a_1 + 0.4a_2 + 0.16b_{11} + 0.16b_{12} + 0.16b_{22} + c >= \varepsilon$ for pattern x_1

$-0.4a_1 + 0.4a_2 + 0.16b_{11} - 0.16b_{12} + 0.16b_{22} + c >= \varepsilon$ for pattern x_2

$0.4a_1 - 0.4a_2 + 0.16b_{11} - 0.16b_{12} + 0.16b_{22} + c >= \varepsilon$ for pattern x_3

$-0.4a_1 - 0.4a_2 + 0.16b_{11} + 0.16b_{12} + 0.16b_{22} + c >= \varepsilon$ for pattern x_4

$a_1 + b_{11} + c <= -\varepsilon$ for pattern x_5

$a_2 + b_{22} + c <= -\varepsilon$ for pattern x_6

$-a_1 + b_{11} + c <= -\varepsilon$ for pattern x_7

$-a_2 + b_{22} + c <= -\varepsilon$ for pattern x_8

$\varepsilon >= 0.01$ (5)

The solution to this LP is $a_1 = 0$, $a_2 = 0$, $b_{11} = -0.071429$, $b_{12} = 0$, $b_{22} = -0.071429$, $c = 0.061429$, and $\varepsilon = 0.01$. The single masking function for class A, therefore, is $f_A(\mathbf{x}) = -0.071x_{12} - 0.071x_{22} + 0.061$. The plot of this masking function is shown in [Figure 3]. For any pattern in class A, $f_A(\mathbf{x}) >= \varepsilon$, and for any pattern in class B, $f_A(\mathbf{x}) <= -\varepsilon$. Therefore the masking function for class B, $f_B(\mathbf{x}) = -f_A(\mathbf{x})$.

The proposed method uses the idea of "masking" or "covering" a class region, which is different from most other LP methods in pattern classification. Any complex nonconvex or disjoint region can be covered by a set of elementary convex forms of varying size such as hyperspheres and hyperellipsoids in the N-dimensional space.

Consider the classification problem shown in [Figure 3], where the class A territory consists of two disjoint regions(A_1, A_2) and class B comprises the rest of space. According to the masking procedure, the first effort should be to define a single mask for each of the two classes. To define a single mask for class A, an LP is set up as follows :

Minimize ε

s. t.

$f_A(\mathbf{x}_i) >= \varepsilon$ for all sample patterns $\mathbf{x}_i$ belonging class A,

$f_A(\mathbf{x}_i) <= -\varepsilon$ for all sample patterns $\mathbf{x}_i$ belonging to class B

$\varepsilon >=$ a small positive constant.

where $f_A(\mathbf{x})$ is a quadratic. This LP has no feasible solution when B sample patterns occur between the A regions. This indicates that a single mask cannot be constructed for class A. So, in the next step, the class A sample patterns are divided, suppose, into two clusters (A_1 and A_2) for further masking attempts. Now, an attempt is made to mask A_1 and A_2 clusters separately. The LP for the A_1 cluster is :

Minimize ε

s. t.

$f_{A1}(\mathbf{x}_i) >= \varepsilon$ for all sample patterns $\mathbf{x}_i$ belonging to cluster A_1

$f_{A1}(\mathbf{x}_i) <= -\varepsilon$ for all sample patterns $\mathbf{x}_i$ belonging to class B

$\varepsilon >=$ a small positive constant.

Similarly, the LP for cluster A_2 is :

Minimize ε

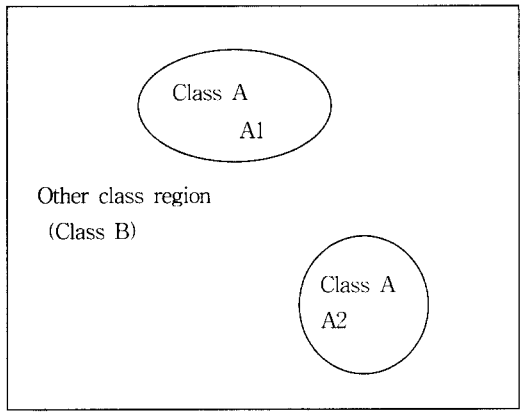
s. t.

$f_{A2}(\mathbf{x}_i) >= \varepsilon$ for all sample patterns $\mathbf{x}_i$ belonging to cluster A_2

$f_{A2}(\mathbf{x}_i) <= -\varepsilon$ for all sample patterns $\mathbf{x}_i$ belonging to class B

$\varepsilon >=$ a small positive constant.

Two LP solutions generated by the above two LPs will form a decision region by the "OR" condition. That is, an input pattern will be classified into class A if it is in either of two or in both masking functions of class A. The process of splitting subsets (or clustering) continues until all LP solutions of subsets (or clusters) are feasible. Thus, any arbitrary nonconvex or disjoint class region can be covered by several masking functions. If the class regions are overlapped two much, then the above procedure does not work. Thus, a heuristic to delete outlier patterns for applying the above procedure is needed.



[Figure 3] A classification problem with disjoint class regions

3. Identifying and Deleting Outliers in Classification Problems

General classification problems have in nature pattern vectors that fall outside their core class regions. Let these pattern vectors be called "outliers". A basic concept underlying many classification systems is to extract the core regions of a class from the information provided and ignore its outliers. The procedure is to divide the pattern space spanned by the training patterns into small neighborhoods and assigns each neighborhood or cluster to the class with the majority of its members. The remaining minority sample patterns in these neighborhoods (clusters) are treated as outliers and discarded.

This territory assignment idea is implemented in a three-step procedure in this method. In the first step, territories are allocated on the basis of significant majority (e.g., more than two-thirds majority in the cluster). More mixed clusters are dealt with in the next two steps by defining smaller clusters and using a simpler majority rule (e.g., more than half). One can view this procedure as doing the easy, noncontroversial allocations first and taking on harder cases later. The last two steps examine gradually smaller neighborhoods (clusters) for assignment decisions. The procedure also performs a two-point sensitivity analysis in the first step. The sensitivity analysis verifies the consistency of the assignments under two-different neighborhood sizes (i.e., two different average cluster sizes). All inconsistent assignments are carried forward to the next step. For example, when a sample pattern is found to be an outlier under one average cluster size but not the other, it is carried forward as "unassigned" to the next step for a closer examination. The procedure is outlined in more detail next.

Let m_1, m_2, m_3 (where $m_1 > m_2 >= m_3$) be the average cluster sizes used in the three steps. For instance, one can set $m_1 = 7$, $m_2 = 5$, $m_3 = 3$. The number of sample patterns remaining unassigned in any step (n_i) is divided by the corresponding

average cluster size (m_i) to determine the number of clusters to form (P_i) in that step using a k-means clustering algorithm. So $P_i = n_i / m_i$ and $n_1 = n$, n being the total number of training samples. In the first step, the training set is broken up into $(1+c)P_1$ and $(1-c)P_1$ clusters for sensitivity analysis, where c is a factor usually between 0.2 and 0.3. In this step, sample patterns in a cluster are labeled as "outlier" if their class has less than one-third of the members and they are labeled as "core patterns" if their class has at least two-thirds of the members. Otherwise, they are simply labeled "unassigned" for further processing in the next step. Consistency of these categorizations is checked across the two different breakup schemes($(1+c)P_1$ and $(1-c)P_1$ clusters). If a pattern's category is not consistent, it is labeled as "unassigned". Only unassigned patterns are carried forward to the next step-they comprise the n_2 patterns for step 2. The "core patterns" are kept aside to be masked and the "outliers" discarded.

In steps 2 and 3, the essential scheme of breaking up the remaining unassigned sample patterns into small clusters and labeling them using a majority rule remains the same. In these steps, an attempt is made to quickly resolve the remaining cases by relaxing the majority rule.

4. The Algorithm

Some notation, introduced in the last section, is defined more formally here. n_i is the number of sample patterns remaining unassigned at the start of the i-th step of the outlier detection phase (phase I). m_i is the average breakup cluster size and P_i, where $P_i = n_i / m_i$, is the number of clusters to form in the same i-th step of phase I. c is a factor used in sensitivity analysis in the first step of phase I. The training set is broken up into $(1+c)P_1$ and $(1-c)P_1$ clusters for this sensitivity analysis. q is the minimum cluster size for masking.

4.1 Phase I - Discard Outlier Patterns

[Step 1]

1. Breakup the training set into $(1+c)P_1$ and $(1-c)P_1$ small clusters using a k-means (average) clustering procedure.
2. For each set of clusters, label a pattern as "outlier" if its class has less than one-third of the members in its cluster ; label as "core pattern" if its class has at least two-third of the members and as "unassigned" otherwise.
3. Check consistency of these labels across the two sets of clusters. Any inconsistently labeled pattern is relabeled as "unassigned".
4. "Unassigned" patterns are carried over to step 2. the "outlier" patterns are discarded and the "core patterns" retained for masking.

[Step 2]

1. Breakup the n_2 remaining "unassigned" patterns into P_2 ($= n_2 / m_2$) small clusters using a clustering procedure that can produce a prespecified number of clusters.
2. Discard a pattern as "outlier" if its class has less than 50% of the members in its cluster, save it as "core pattern" if its class has more than 50% of the members and carry it over to step 3 as "unassigned" otherwise.

[Step 3]

1. Breakup the n_3 remaining "unassigned" patterns into P_3 ($= n_3 / m_3$) small clusters using a clustering procedure that can produce a pre-

specified number of clusters.

2. Discard a pattern as "outlier" if its class has less than 50% of the members in its cluster, save it as "core pattern" otherwise.

4.2 Phase II - Generate Masking Functions

1. Initialize class index i=0.
2. Let i=i+1. If i>K, where K is the total number of classes, stop. Otherwise, set j=1 and KL_j = 1, where KL_j is the number of unmasked class i clusters at the j-th stage of breaking up unmasked patterns. Let these unmasked clusters be indexed as C_{j1}, C_{j2}, ···, C_{jKLj}.
3. Using a quadratic masking function f(**x**), set up an LP as follows for each unmasked cluster C_{jk}, k=1, ···, KL_j :

Minimize ε

s. t.

$f^i_{Cjk}(\mathbf{x}) >= \varepsilon$ for all pattern vectors **x** in cluster C_{jk} of class i,

$f^i_{Cjk}(\mathbf{x}) < -\varepsilon$ for all pattern vectors **x** in classes other than class i,

ε >= a small positive constant.

where $f^i_{Cjk}(\mathbf{x})$ is the masking function for cluster C_{jk} of class i

Solve the LP for each unmasked cluster. If all LP solutions are feasible and optimal, the masking of class i is complete. Go to step 2 to mask next class. Otherwise, when some or all LPs are feasible, save all feasible masking functions obtained and go to step 4.

4. Let KL_j' be the number of clusters with infeasible LP solutions at the j-th stage. Subdivide (breakup) the sample patterns in these infeasible clusters into KL_{j+1} (where KL_{j+1} > KL_j') small clusters using a clustering procedure that can produce a prespecified number of clusters. Discard as "outliers" all sample patterns that are from clusters of size q or less.
5. Set j=j+1. KL_j is the number of unmasked class i clusters at this new stage. These unmasked clusters are indexed as before as C_{j1}, C_{j2}, ···, C_{jKLj}. Go back to step 3.

5. Computational Results

All experimental results have been obtained by implementing this algorithm on SAS. The linear programming algorithm implemented in SAS OR is based on simplex method. For clustering, the average linkage method of hierarchical clustering was used.

5.1 Vowel Classification

The vowel classification problem is described in Lippmann (1988) and has been used to compare different classifiers. It is based on the vowel format data of Peterson and Barney (1952). The data was generated from the spectrographic analysis of vowels in words formed by "h", followed by a vowel, followed by a "d" and consists of two-dimensional patterns. The words were spoken by 67 persons, including men, women, and children. The data on 10 vowels was split into two sets - 338 examples each for the training and test sets. Four classifiers - KNN, Gaussian, two-layer neural networks, and feature map - were compared on this dataset (Lippmann, 1988). All classifiers had similar error rates ranging from 18 to 22.8%. The two-layer back propagation network with 50 hidden units took at least 50,000 iterations to obtain an error rate of 20 percent.

<Table 1> shows the results of using different

<Table 1> Results of using different cluster sizes in phase I for the Vowel Classification problem and other classifiers

Training and test results of Vowel Classification problem				Performance of other classifiers (Error rates in the same test sets		
Cluster sizes $M=m_1,m_2,m_3$ phase I	# of Outliers found in phase I	# of Outliers found in phase II	Error rates in test sets	Two-Layer neural net. with 50 hidden units	KNN	Bayesian Classifier
			LP method			
M = 6, 4, 3	57	17	22.6	19.8	21	21
M = 6, 4, 4	52	27	22.5			
M = 6, 5, 3	61	13	22.8			
M = 6, 5, 4	58	20	20.4			
M = 7, 4, 3	58	17	21			
M = 7, 4, 4	53	24	21.5			
M = 7, 5, 3	61	14	21.3			
M = 7, 5, 4	58	21	19.8			
M = 7, 6, 3	64	11	21.3			
M = 7, 6, 4	63	11	21.3			
M = 8, 4, 3	58	16	21.9			
M = 8, 4, 4	53	26	19.2			
M = 8, 5, 3	61	13	19.4			
M = 8, 5, 4	58	20	20.7			

phase I cluster sizes in this procedure. The overall error rate is quite stable across the variety of phase I cluster sizes and is in the 19.2 to 22.8% range with one being 24.6%. This fact is often evident in other problems too, as will be shown, where the same cluster size combinations are used. The total number of outliers found (phases I and II combined) is also very consistent across the cluster size combinations. Because of outliers found in phase II, clean-up phase II runs were made. Test results for other classifiers are reported in <Table 1> as well. The reported error rates of other classifiers are the best ones of several trials. Compared to other classifiers reported, competitive error rate is generally achieved across all cases.

5.2. Noised Normal Distributions

An obvious question about this method is, how well would the outlier detection heuristic work on classes with dense overlap? To test the heuristic under those circumstances, the following two problems were tested :

Problem 1 - A simple 2-class problem where both classes are described by Normal distributions with different means, and with covariance matrices equal to identity I. A 4-dimensional problem with mean vectors [0 0 0 0] and [1 1 1 1] was tried. The optimal Bayes error rate is 15.2% in this case.

Problem 2 - A 2-class problem where both classes are described by Normal distributions with zero mean vectors and covariance matrices equal to I and 4I. The optimal Bayes error rate for a 4-dimensional problem is 17.64% and for a 8-dimensional problem is 9%

Both problems were simulated on randomly generated training sets of different sizes. <Tables 2> and <Tables 3> show the results. The results

show that for both problems, the error rate generally decreases as the training set size is increased and tends to the theoretical optimal Bayes error rate. A randomly generated test set of 400 examples was used for each problem. The table entries M = 7, 5, 3 etc. indicate the average cluster sizes used in each step of the three-step phase I procedure. The <Tables 2> and <Table 3> show that the error rates are fairly stable for different combinations of these parameters.

〈Table 2〉 Results for Noised Normal Distribution - Problem 1

Cluster sizes $M=m_1,m_2$ phase I	# of outliers found in phase I	# of outliers found in phase II	Error rates (%)	
			LP method	Neural net. with 8 hidden units
Number of sample patterns (n = 180)				
M = 6, 5	23	0	17.3	18.7
M = 7, 5	27	0	17.6	
M = 8, 5	27	0	19	

〈Table 3〉 Results for Noised Normal Distribution - Problem 2

Cluster sizes	# of outliers	# of outliers	Error rates (%)	
$M = m_1,m_2$ phase I	found in phase I	found in phase II	LP method	Neural net with 8, 15 hidden units each
Number of sample patterns (n = 180), 4-dimension				
M = 6, 5	29	0	19	19.5
M = 7, 5	29	0	19.2	
M = 8, 5	29	0	19	
Number of sample patterns (n = 400), 8-dimension				
M = 6, 5	38	15	10.5	11.8
M = 7, 5	49	20	11.2	
M = 8, 5	63	3	12	

6. Conclusion

The method in this paper has overcome some limitations of current LP formulations by integrating nonlinear masking(discriminant) functions and heuristic to delete outlier patterns and has been able to construct a solid classifier for general purpose classification problems with relatively good classification performance.

The discriminant functions used in this research are quadratic. Although these restricted forms of discriminant functions could achieve good results, more general forms of discriminant functions can be considered. Further research on developing the general masking functions for cover some arbitrary class regions is now being investigated. Also the generalization capabilities of the LP methods in data poor regions of high dimensional input space should be explored further.

References

[1] Abe, S., "Fuzzy Function Approximators with Ellipsoidal Regions", *IEEE Transactions on Systems, Man, and Cybernetics-Part B : Cybernetics*, Vol.29, No.4 (August 1999).

[2] Duda, R.O. and P.E. Hart, *Pattern classification and scene analysis. New York : John Wiley & Sons*, 1973.

[3] Everitt, B.S., *Cluster analysis* (2nd ed.). London : Heinemann Educational Books Ltd., 1980.

[4] Freed, N. and F. Glover, A linear programming approach to the discriminant problem. *Decision Sciences*, No.12(1981), pp.68-74.

[5] Freed, N. and F. Glover, Evaluating alternative linear programming models to solve the two-group discriminant problem. *Decision Sciences*, No.17(1986), pp.151-162.

[6] Glover, F., S. Keene and B. Duea, A new

class of models for the discriminant problem. *Decision Sciences*, No.19(1988), pp. 269-280.

[7] Hartigan, J.A., *Clustering algorithms*. New York : John Wiley & Sons, 1975.

[8] Huang, W.Y. and R.P. Lippmann, "Neural net and traditional classifiers. In D. Anderson (Ed.)", *Neural Information Processing Systems*, New York : American Institute of Physics, (1988) pp.387-396.

[9] Koehler, G.J. and S. Erenguc, " Misclassifications in Linear Discriminant Analysis", *Decision Sciences*, No.21(1990), pp.23-85.

[10] Lippmann, R.P., "Neural network classifiers for speech recognition", *The Lincoln Laboratory Journal*, No.1(1)(1988), pp.107-128.

[11] Lippmann, R.P., "Pattern classification using neural networks", *IEEE Communications Magazine*, No.27(1989), pp.47-64.

[12] Peterson, G.E. and H.L. Barney, "Control methods used in a study of vowels", *Journal of the Accoustic Society of America*, No.24 (1952), p.175.

[13] Rubin, P.A., "A comparision of linear programming and parametric approaches to the two- group discriminant problem", *Decision Sciences.*, No.21(1990), pp.373-385.

[14] Rubin, P.A., "Separation failure in linear programming discriminant models", *Decision Sciences*, No.22(1991), pp.519-535.

[15] Rumelhart, D.E., G.E. Hinton and R.J. Williams, "Learning Internal Representations by Error Propagation", Rumelhart and McCleland (eds.), Parallel Distributed Processing : Explorations in Microstructure of Cognition, Vol.1 : Foundations, Cambridge, Massachusetts : The MIT Press, 1986.