	LPtest	HK			FCR			PER		
		sep	non	inc	sep	non	inc	sep	non	inc
separable	452	272	31	149	449	0	3	428	0	24
nonseparable	392	0	1	391	0	4	388	0	0	392

Table 1: Conclusion on linear separability by each algorithm (entries are number of problems; sep: separable, non: nonseparable, inc: inconclusive).

results that there is no known proof of convergence for HK and FCR in a predictable number of steps, and that linear programming has known, predictable time bounds for convergence on both linearly separable and nonseparable cases. The implementational difficulties bring doubts on the practicality of these adaptive algorithms.

The only reservation we have about this conclusion is that for linear programming we used the MINOS solver [13] through the AMPL interface [4], which was highly optimized commercial code, whereas the adaptive procedures were run under the simplest implementation by ourselves in C, so affordable (elapsed) time may not mean the same thing for the two groups. Also, we have not tested the dependence of the run time on the order in which the input vectors were presented, and we have not investigated dual problems that can be formulated for a given problem and solved by any of these procedures [15]. Nevertheless, we advocate that linear programming methods deserve more serious attention in classification studies. Without more sophisticated derivatives such as simultaneous primal-dual algorithms, the only apparent advantages of the adaptive procedures such as HK and FCR rules seem to be that (1) they can be implemented on very simple machines; and (2) their adaptive nature permits easier inclusion of new input that may become available during the training process, and thus they are better suited for on-line learning.

Acknowledgements

We thank Ken Clarkson, David Gay, Margaret Wright, and George Nagy for helpful discussions and suggestions of references. Mitra Basu thanks Bell Labs for the summer support in 1998.

References

[1] Basu, M. and Liang, Q., The Fractional Correction Rule : A New Perspective. *Neural Network,* **11**, 1998, 1027-1039.

[2] Blake, C., Keogh, E. Merz, C.J. UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science, 1998.

[3] Bennett, K.P., Mangasarian, O.L., Robust Linear Programming Discrimination of Two Linearly Inseparable Sets, *Optimization Methods and Software*, **1**, 1992, 23-24.

[4] Fourer, R., Gay, D.M., Kernighan, B.W., *AMPL: A Modeling Language for Mathematical Programming*, The Scientific Press, 1993.

[5] Glover, F., Improved Linear Programming Models for Discriminant Analysis, *Decision Sciences*, **21**, 4, 1990, 771-785.

[6] Grinold, R.G., Comment on "Pattern Classification Design by Linear Programming", *IEEE Transactions on Computers*, **C-18**, 4, April 1969, 378-379.

[7] Grinold, R.G., Mathematical Programming Methods of Pattern Classification, *Management Science*, **19**, 3, 1972, 272-289.

[8] Hassoun, M.H. and Song, J., Adaptive Ho-Kashyap Rules for Perceptron Training. *IEEE Trans. on Neural Networks*, **3**, 1992, 51-61.

[9] Ho, Y.C. and Kashyap, R.L., An Algorithm for Linear Inequalities and its Applications. *IEEE Trans. on Electronic Computers*, **14**, 1965, 683-688.

[10] Mangasarian, O.L., Linear and Nonlinear Separation of Patterns by Linear Programming, *Operations Research*, **13**, 1965, 444-452.

[11] Mays, C.H., **Adaptive Threshold Logic**. Ph.D. thesis, Stanford Electron. Labs., Stanford, CA., 1963.

[12] , Minsky, M., Papert, S., *Perceptrons*, expanded edition, the MIT press, 1988.

[13] Murtagh, B.A., Saunders, M.A., Large-Scale Linearly Constrained Optimization, *Mathematical Programming*, **14**, 1978, 41-72.

[14] Rosenblatt, F., **Principles of Neurodynamics: Perceptron and the Theory of Brain Mechanism**. Washington: Spartan Press, 1962.

[15] Roychowdhury, V.P., Siu, K.Y., Kailath, T., Classification of Linearly Nonseparable Patterns by Linear Threshold Elements, *IEEE Transactions on Neural Networks*, **6**, 2, March 1995, 318-331.

[16] Siu, K.Y., Roychowdhury, V. and Kailath, T., **Discrete Neural Computation**. Englewood Cliffs, NJ: Prentice Hall, 1995.

[17] Smith, F.W., Pattern Classifier Design by Linear Programming, *IEEE Transactions on Computers*, **C-17**, 4, April 1968, 367-372.

[18] Tou, J.T., Gonzalez, R.C., *Pattern Recognition Principles*, Addison-Wesley, 1974.

[19] Widrow, B. and Hoff, M.E. Jr., **Adaptive Switching Circuits**. Tech. Report 1553-1, Stanford Electron. Labs. Stanford, CA., 1960.

[20] Widrow, B. and Stearns, S.D., **Adaptive Signal Processing**. Englewood Cliffs, NJ: Prentice Hall, 1985.

[21] Widrow, B. and Lehr, M.A., 30 Years of Adaptive Neural Networks: Perceptron, Madeline, and Backpropagation. *Proceedings of the IEEE*, **78**, 1990, 1415-1442.

[22] Wright, M.H., Interior Methods for Constrained Optimization, in A. Iserles (ed.), *Acta Numerica*, 1992, 341-407.

for large problems. So the adaptive algorithm (7) was used instead. With LP, we used the simple formulation (8) to test for linear separability (referred to as LPtest), and also Smith's formulation (9) to derive a minimum error separating hyperplane (referred to as LPme). We included the perceptron training rule too for comparison purpose (PER), although it is understood that it does not converge for nonseparable input.

The problems were discrimination between all pairs of classes in 14 data sets from the UC-Irvine Machine Learning Depository [2]. The data sets were so chosen that each set has at least 500 input vectors and no missing values in the features. The names of the dataset are as follows: abalone, car, german, kr-vs-kp, letter, lrs, nursery, pima, segmentation, splice, tic-tac-toe, vehicle, wdbc, and yeast. For those sets containing categorical features, the values were numerically coded. There are a total of 844 two-class discrimination problems. Outcomes from the algorithms may be a conclusion of whether the problem is linearly separable or not, or inconclusive after a chosen number of iterations. We compared such outcomes from all the three procedures and the relative time it took for them to derive the results.

Table 1 shows the number of problems reported by each algorithm as separable, nonseparable, or inconclusive. Since LPtest always gives a definite answer, conclusions of other algorithms are compared to its results.

We found that within similar, affordable run time limits, linear programming always arrived at a definite conclusion, while HK and FCR often ended the runs inconclusively (100,000 iterations were used for HK, FCR, and PER). Of the 844 problems, LPtest reported that 452 are linearly separable and 392 are not. For 54% (453/844) of these problems, FCR arrived at a conclusion but the fraction is only 36% (304/844) for HK. For the separable problems, FCR arrived at the conclusion generally sooner than HK. For the nonseparable problems, both algorithms have problem fulfilling the claims of giving an indication: within the affordable time, only 1 problem was reported to be nonseparable by HK, and only 4 by FCR.

We also found that the results were very sensitive to the choices of the learning coefficient and convergence thresholds for the Ho-Kashyap rule. Other than affecting the speed of convergence, they can change the conclusion on separability: for 31 separable problems HK reported that they were nonseparable. An improper learning coefficient may cause overly large correction steps. The tolerance threshold determines when a number is considered zero which leads to a conclusion. Of those 31 problems falsely reported to be nonseparable by HK, 26 contain only 1 vector in the smaller class. The other 5 problems have 2 vectors in the smaller class. With a change in the learning coefficient (to 10% of original value), 6 were reported separable, 9 remained nonseparable, and the other 16 became inconclusive.

	LPme	HK	FCR	PER
LPtest	0.6974	0.0164	0.1930	0.1256
LPme		-0.0275	0.1586	0.1341
HK			0.7306	0.5723
FCR				0.5421

Table 2: Correlation coefficients of number of iterations for each pairs of algorithms to converge to a separating hyperplane.

	LPme	LPtest	HK	FCR	PER
n	0.1970	0.5749	-0.0928	-0.0746	-0.0555
m	0.8856	0.4056	-0.0067	-0.0704	-0.0768
nm	0.4778	0.7636	-0.0429	-0.0388	-0.0287

Table 3: Correlation coefficients of problem size measures and number of iterations for each algorithm to converge to a separating hyperplane. n: no. of dimensions; m: no. of input vectors.

Table 2 shows, for the separable problems, the correlation coefficients of the number of iterations it took for each pair of algorithms to converge to a hyperplane, computed over only those cases where both algorithms converged. In LP each iteration involves an input vector, but in the adaptive algorithm each iteration involves a loop through all relevant vectors in the entire set. For this reason, significant correlation exists only between the adaptive algorithms or the two LP formulations, but not between any of the adaptive algorithms and LP. The correlation is stronger between HK and FCR than between each of them and PER.

Correlation coefficients were also calculated between measures of the problem size and the number of iterations before convergence for each algorithm. (Table 3). For the adaptive procedures, the absence of significant correlation suggests that the difficulty of a problem does not necessarily depend on the problem size.

5 Conclusions

Our experiments show that linear programming, although long neglected in classification studies, generally yields more affordable and dependable results. On the contrary, the Ho-Kashyap and the fractional correction rules frequently do not converge within affordable time limits. The Ho-Kashyap rule may even lead to wrong conclusions. It is very difficult to choose the learning coefficients and tolerance thresholds to get all the conclusions right. This reinforces the theoretical

- Suppose the error vector has no positive component for some finite j then $|\epsilon^j| + \epsilon^j = 0$. In that case the correction will cease and neither the weight vector nor the margin vector will change (see (6)). Thus an error vector with no positive components conclusively points to the nonlinear nature of the data[2].

- Suppose ${\epsilon_j}^+ = (\epsilon^j + |\epsilon^j|)$ is never zero for finite j. We can derive from **Fact 2** that $|{\epsilon_{j+1}}^+|^2 < |{\epsilon_j}^+|^2$. Therefore $|{\epsilon_j}^+|$ must converge to zero. However, its distance from zero is unknown for any fixed j.

In summary, Ho-Kashyap algorithm indicates nonseparability but there is no bound on the number of steps. Hassoun and Song [8] propose a variation of the Ho-Kashyap algorithm equipped to produce an *optimal* separating surface for linearly nonseparable data. The authors claim that it can identify and discard nonseparable samples to make the data linearly separable with increased convergence speed. Again we notice similarity with the heuristic approach proposed in [16]. However, the authors do not provide any theoretical basis to their claim. The algorithm is only tested on simple toy problems. We have no knowledge of any extensive testing on real world problems.

3 Linear Programming

In searching for a linear classifier, the input vectors give a system of linear inequalities constraining the location and orientation of the optimal separating hyperplane. With a properly defined objective function, a separating hyperplane can be obtained by solving a linear programming problem. Several alternative formulations have been proposed in the past ([3], [5], [10], [15], [17]) employing different objective functions. An early survey of these methods is given in [7]. Here we mention a few representative formulations.

In a very simple formulation described in [15], the objective function is trivial, so that it is simply a test of linear separability by finding a feasible solution to the LP problem

$$\begin{array}{ll} \text{minimize} & \mathbf{0}^t\mathbf{w} \\ \text{subject to} & \mathbf{Z}^t\mathbf{w} \geq \mathbf{1} \end{array} \qquad (8)$$

where $\mathbf{w}$ is the weight vector of a separating hyperplane, $\mathbf{Z} = [\mathbf{z^1}, \mathbf{z^2}, ..., \mathbf{z^m}]$ is a matrix of column vectors $\mathbf{z}^j$ $(j = 1, ..., m)$, the m augmented input vectors as defined before, $\mathbf{0}$ and $\mathbf{1}$ are vectors of zero's and one's respectively.

[2]It can be shown that for linearly separable data, it is impossible for all components of the error vector to be negative at any given iteration.

This formulation gives only a test for linear separability but does not lead to any useful solution if the data are not linearly separable. Another formulation suggested by Smith ([6], [17]) minimizes an error function:

$$\begin{array}{ll} \text{minimize} & \mathbf{a}^t\mathbf{t} \\ \text{subject to} & \mathbf{Z}^t\mathbf{w} + \mathbf{t} \geq \mathbf{b} \\ & \mathbf{t} \geq \mathbf{0} \end{array} \qquad (9)$$

where $\mathbf{Z}$ is the augmented data matrix as before, $\mathbf{a}$ is a positive vector of weights, $\mathbf{b}$ is a positive margin vector chosen arbitrarily (e.g. $\mathbf{b} = \mathbf{1}$), and $\mathbf{t}$ and $\mathbf{w}$ are the error and weight vectors which are also decision variables for the LP problem. In [17] each component of $\mathbf{a}$ was set to be $1/m$.

More recently, Bennett and Mangasarian [3] modified this formulation to use different weights for input vectors belong to each of the two classes. Let m_1, m_{-1} be the number of vectors belonging to the two classes respectively, $\mathbf{a} = (a_1, a_2, ..., a_m)^t$, for $j = 1, ..., m$,

$$\begin{array}{lll} a_j & = 1/m_1 & \text{if } d^j = 1, \\ a_j & = 1/m_{-1} & \text{if } d^j = -1. \end{array}$$

It is argued that this formulation is more robust in the sense that it can guarantee a nontrivial solution $\mathbf{w}$ even if the centroids of the two classes happen to coincide.

Though arguably algorithms for solving LP problems are more sophisticated than the previously discussed iterative procedures, an important advantage of finding $\mathbf{w}$ by solving an LP problem is that if the feasible region is nonempty, solution can be determined in a finite number of steps.

The number of steps it takes to arrive at the solution, however, is dependent on the geometrical configuration of the data points. If the LP is solved by the simplex method, in the worst case, the algorithm may have to visit every vertex formed by the intersections of the constraining hyperplanes before reaching an optimum. Empirical evidence shows that in practice this rarely happens. More recently, interior-point methods [22], such as Karmarkar's, are shown to have better worst case time complexity. Still, the comparative advantages of such methods for an arbitrary problem remain unclear, partly because there has not been a good way to characterize the structure of a particular problem and relate that to the detailed operations of the algorithms.

4 Experimental Results

We applied the three procedures (HK, FCR, LP) that are claimed to indicate linear nonseparability to a collection of two-class discrimination problems. The original Ho-Kashyap rule involves computing a pseudoinverse matrix which turned out to be overly expensive

Group B: Error minimization procedures

The error correction procedure focuses on misclassified samples. Other procedures modify the weight vector using all samples at each iteration. Moreover, thus far a weight vector $\mathbf{w}$ is sought such that $\mathbf{w}^t\mathbf{z}^j \ \forall j$ is positive. Next, we discuss attempts to reformulate the problem of finding the solution to a set of linear inequalities as a problem of finding solution to a set of linear equations. Let us construct a matrix $\mathbf{Z} = [\mathbf{z}^1, \mathbf{z}^2, ..., \mathbf{z}^m]$. Let $\mathbf{b} = (b^1, b^2, ..., b^m)^t$ be a column vector. The decision equation (2) can be restated as

$$\mathbf{Z}^t\mathbf{w} = \mathbf{b} \tag{4}$$

The solution vector $\mathbf{w}$ is overdetermined since $\mathbf{Z}^t$ is rectangular with more rows than columns, assuming $m > n$. The idea is to search for a weight vector that minimizes some function of the error between the left and the right hand side of (4). Usual choice of a function to be minimized is one that represents sum-of-squared error

$$J = |\mathbf{Z}^t\mathbf{w} - \mathbf{b}|^2.$$

The central theme in both Widrow-Hoff and Ho-Kashyap procedures is to minimize J. Though, the difference in the details of the algorithms leads to drastically different results.

The α-LMS algorithm or Widrow-Hoff delta rule embodies the minimal disturbance principle [1][19]. It is designed to handle both linear and nonlinear input. The criterion function is minimized with respect to the weight vector $\mathbf{w}$ using gradient descent method. The unknown vector $\mathbf{b}$ is chosen arbitrarily and held constant throughout the computation. See [20] for derivation of the weight update equation using the gradient descent method. The iterative version of the weight update equation can be written as [20] :

$$\begin{aligned} \mathbf{w}^0 &\quad \text{arbitrary} \\ \mathbf{w}^{j+1} &= \mathbf{w}^j + \alpha\frac{(b^j - \mathbf{w}^{j\,t}\mathbf{z}^j)\mathbf{z}^j}{|\mathbf{z}^j|^2} \end{aligned} \tag{5}$$

It has been shown that this rule converges in the mean square sense to the solution $\mathbf{w}^*$ that corresponds to the least mean square output error if all input vectors are of same length in both linearly-separable as well as linearly non-separable cases [20]. It is known that in some cases this rule may fail to separate training vectors that are linearly separable [11]. This is not surprising, since the mean square error (MSE) solution depends on the margin vector $\mathbf{b}$. Different choices for $\mathbf{b}$ gives the solution different properties. Hence, when one does not have any clue about the distribution of the input data and arbitrarily fixes a margin vector it is possible that the resulting weight vector may not classify all vectors correctly even for a linearly separable problem.

[1] The rule aims at making minimum possible change in the weight vector during the update process such that the output for as many of the previously correctly classified samples as possible remain unperturbed.

Group C: Constrained error minimization procedures

Ho and Kashyap [9] modified the Widrow-Hoff procedure to obtain a weight vector $\mathbf{w}$ as well as a margin vector $\mathbf{b}$. They imposed the restriction that the m-dimensional margin vector must be positive-valued, i.e., $\mathbf{b} > 0$ ($b_k > 0, \ \forall\ k$). The problem is equivalent to finding $\mathbf{w}$ and $\mathbf{b} > 0$ such that $J = |\mathbf{Z}^t\mathbf{w} - \mathbf{b}|^2$ is minimized with respect to both $\mathbf{w}$ and $\mathbf{b}$. Note that since both $\mathbf{w}$ and $\mathbf{b}$ (subject to imposed constraint) are allowed to play a role in the minimization process, the minimum value (i.e., 0) for J can be achieved in this case. Hence the $\mathbf{w}$ that achieves that minimum is the separating vector in the linearly separable case. The weight updating rule is (for detailed derivation see [9])

$$\begin{aligned} \mathbf{b}^0 &> 0 \quad \text{otherwise arbitrary} \\ \mathbf{w}^0 &= (\mathbf{Z}^t)^\dagger\mathbf{b}^0 \\ \epsilon^j &= \mathbf{Z}^t\mathbf{w}^j - \mathbf{b}^j \\ \mathbf{b}^{j+1} &= \mathbf{b}^j + \alpha(|\epsilon^j| + \epsilon^j) \\ \mathbf{w}^{j+1} &= \mathbf{w}^j + \alpha(\mathbf{Z}^t)^\dagger(|\epsilon^j| + \epsilon^j) \end{aligned} \tag{6}$$

where $(\mathbf{Z}^t)^\dagger = (\mathbf{Z}\mathbf{Z}^t)^{-1}\mathbf{Z}$ is the pseudoinverse of $\mathbf{Z}^t$. Computation of the pseudoinverse may be avoided by using the following alternate procedure [8]

$$\begin{aligned} \epsilon_k^j &= \mathbf{z}^{k\,t}\mathbf{w}^j - {b_k}^j \\ {b_k}^{j+1} &= {b_k}^j + \rho_1/2(|\epsilon_k^j| + \epsilon_k^j) \\ \mathbf{w}^{j+1} &= \mathbf{w}^j - \rho_2\mathbf{z}^k(\mathbf{z}^{kt}\mathbf{w}^j - b_k^{j+1}) \end{aligned} \tag{7}$$

This algorithm yields a solution vector in case of linearly separable samples in finite number of steps if $0 < \rho_1 < 2$, and $0 < \rho_2 < 2/\|\mathbf{z}^k\|^2$. Since the focus of this paper is on data that are not separable, let us examine the behavior of this algorithm under nonseparable situation.

Note the following two facts for nonseparable case : **Fact 1.** $\epsilon^j \neq 0$ for any j and **Fact 2.** $|\epsilon^{j+1}|^2 < |\epsilon^j|^2$ i.e., the sequence $|\epsilon^1|^2, |\epsilon^2|^2, ...$ is a strictly monotonically decreasing sequence and must converge to the limiting value $|\epsilon|^2$, though the limiting value can not be zero. It can be shown that $(\epsilon^j + |\epsilon^j|)$ converges to zero suggesting a termination of the procedure. Now consider the following two cases.

values $\{d^1, ..., d^m\}$, $d^i \in \{1, -1\}$, *find a weight vector* $\mathbf{w} \in R^n$ *such that* $d^j = y^j = sgn(\mathbf{w}^t\mathbf{x}^j)$ *for* $j = 1, ..., m$.

The goal is to determine a weight vector $\mathbf{w}$ such that the following conditions are satisfied:

$$\begin{aligned} \mathbf{w}^t\mathbf{x}^j &> 0 \quad \text{if } d^j = +1 \\ \mathbf{w}^t\mathbf{x}^j &< 0 \quad \text{if } d^j = -1 \end{aligned} \tag{1}$$

The equation ${\mathbf{w}^*}^t\mathbf{x} = 0$ defines a hyperplane in R^n. Therefore finding a solution vector $\mathbf{w}^*$ to this equation is equivalent to finding a separating hyperplane that correctly classifies all vectors $\mathbf{x}^j$, j=1,2,...,m. In other words, an algorithm must be designed to find a hyperplane ${\mathbf{w}^*}^t\mathbf{x} = 0$ that partitions the input space into two distinct regions, one containing all vectors $\mathbf{x}^j$ for which the desired output is +1 and the other region containing all vectors $\mathbf{x}^j$ for which the desired output is -1. We reformulate this condition (1) to adopt the convention usually followed in the literature.

Remark 2: Define a vector $\mathbf{z}^j$:

$$\begin{cases} \mathbf{z}^j = +\mathbf{x}^j & \text{if } d^j = +1 \\ \mathbf{z}^j = -\mathbf{x}^j & \text{if } d^j = -1 \end{cases}$$

The output y^j for the modified input vector $\mathbf{z}^j$ is computed as $y^j = sgn(\mathbf{w}^t\mathbf{z}^j)$. The goal is to determine a weight vector $\mathbf{w}$ such that the following condition is satisfied:

$$\mathbf{w}^t\mathbf{z}^j > 0 \quad \text{if } d^j = +1 \; or \; -1 \tag{2}$$

Note that this simplification aids only in theoretical analysis. As far as the implementation is concerned, this does not alter the actual computation in a significant manner.

Descent procedures are those that modify the weight vector as the algorithms examine the input vectors one by one. There are non-descent based methods for obtaining linear classifiers such as Fisher's linear discriminant analysis. In this paper we focus on descent procedures that are categorized broadly into four groups, with the first three being adaptive procedures:

- **Group A:** Error correction procedures
- **Group B:** Error minimization procedures
- **Group C:** Constrained error minimization procedures, and
- **Group D:** Linear programming

It should be noted that the term "error" is defined differently in each context. In Group A it refers to misclassification, and in Groups B and C it refers to a measure of distance of a point from a hyperplane. The remainder of this section discusses Groups A, B, and C, whereas Group D will be covered in the next section.

Group A: Error correction procedures

Among the adaptive procedures, the fixed-increment perceptron training rule is the most well known. It can be shown that [14] if the input vectors are linearly separable this rule will produce a solution vector $\mathbf{w}^*$ in a finite number of steps. However, for input vectors that are not linearly separable, the perceptron algorithm does not converge. Since, if the input vectors are nonseparable, then for any set of weights $\mathbf{w}$, there will exist at least one input vector, $\mathbf{z}$, such that $\mathbf{w}$ misclassifies $\mathbf{z}$. In other words, the algorithm will continue to make weight changes indefinitely. In the cases where the input vectors are integer-valued, the weight vectors cycle through a finite set of values [12]. An observation of such cycling is an indication of that the input is linearly nonseparable. Though, there are no known time bounds for this to become observable. The projection learning rule [1] (more commonly known as the fractional correction rule [18]), a variation of the perceptron rule, is also based on error correcting principle. However, its behavior with nonlinear input is quite different from that of the perceptron rule. The weights are updated in the following manner:

$$\begin{aligned} \mathbf{w}^0 &\quad \text{arbitrary} \\ \mathbf{w}^{j+1} &= \mathbf{w}^j - \alpha\frac{({\mathbf{w}^j}^t\mathbf{z}^j)\mathbf{z}^j}{|\mathbf{z}^j|^2} \quad \text{if } {\mathbf{z}^j}^t\mathbf{w}^j \leq 0 \end{aligned} \tag{3}$$

It can be shown that this algorithm converges for linearly separable input. The result pertaining to nonlinear input can be stated as follows [1].

Proposition 2.2 *If the input is not linearly separable then the projection learning rule converges to a* $\mathbf{0}$ *solution for* $0 < \alpha < 2$.

Proof: Since no hyperplane can separate the input, the weight vector has to be updated at least once in each cycle. From 3 one can derive that

$$\|\mathbf{w}^{j+1}\|^2 - \|\mathbf{w}^j\|^2 = \frac{\alpha(\alpha - 2)({\mathbf{z}^j}^t\mathbf{w}^j)^2}{\|\mathbf{z}^j\|^2}$$

Note that $\forall\, \alpha$, $0 < \alpha < 2$, $\|\mathbf{w}^{j+1}\|^2 < \|\mathbf{w}^j\|^2$. This indicates that the sequence $\|\mathbf{w}^0\|$, $\|\mathbf{w}^1\|$, ... is a strictly monotonically decreasing sequence with lower bound 0. Therefore $\|\mathbf{w}^j\|$ approaches zero as j approaches infinity. This proves that the projection learning rule converges to the only solution (i.e., $\mathbf{w} = \mathbf{0}$) in the linearly nonseparable case for $0 < \alpha < 2$.

For one-dimensional linearly nonseparable input one can show that $\|w\|$ falls within a small distance ϵ from zero in a number of steps that can be expressed as a function of the initial weight vector, α, ϵ, and the angles between the input vectors [1]. However, no similar expression has been known for higher dimensional input.

The Learning Behavior of Single Neuron Classifiers on Linearly Separable or Nonseparable Input

Mitra Basu

Department of Electrical Engineering
The City College, CUNY
140th Street & Convent Ave.,
New York, NY 10031, USA
eemb@ee-mail.engr.ccny.cuny.edu

Tin Kam Ho

Bell Laboratories
Lucent Technologies
700 Mountain Avenue
Murray Hill, NJ 07974, USA
tkh@bell-labs.com

Abstract

Determining linear separability is an important way of understanding structures present in data. We explore the behavior of several classical descent procedures for determining linear separability and training linear classifiers in the presence of linearly nonseparable input. We compare the adaptive procedures to linear programming methods using many pairwise discrimination problems from a public database. We found that the adaptive procedures have serious implementational problems which make them less preferable than linear programming.

1 Introduction

Discovery and understanding of the structures in data is crucial in constructing classifiers. In practice, a classification problem often begins with a finite training set assumed to be representative of the expected input. Important clues about the complexity of the problem can be obtained by studying the geometrical structures present in such a training set. A geometrical property that is of fundamental importance is linear separability of the classes. Based on this, a number of descriptors of the point set geometry can be constructed. Whether the input is linearly separable or nonseparable, in constructing a classifier, deriving a linear discriminant that is optimal in a certain sense is a useful first step. Linear discriminants can serve as building blocks for piecewise linear classifiers, or be used to separate points projected to a higher dimensional space where they may become linearly separable.

In this paper we study a set of learning algorithms each of which yields a linear classifier: (1) fractional correction rule (FCR), (2) Ho-Kashyap rule (HK), and (3) linear programming (LP). If the classes are linearly separable then any of these algorithms will produce a correct classifier. However, if the classes are linearly nonseparable, then each claims to give indication about the possible non-linear nature of the data. We believe that a study of how these three linear-discriminant functions with their simple yet elegant procedures discover structures in the data that are not necessarily linear will be a step forward in understanding the data. Based on this we designed experiments to test the theoretical claims and observe the behavior of the algorithms on real world data which often contain linearly nonseparable classes.

2 The Single-Neuron Adaptive Classifiers

Consider a 2-class (ω_1, ω_2) problem. Let us assume that there are m sets of training pairs namely, $(\mathbf{x}^1, d^1)$, $(\mathbf{x}^2, d^2)$,..., where $\mathbf{x}^j \in R^n$ is the j^{th} input vector and $d^j \in$ {-1, +1}, j=1, 2, ..., m is the desired output for the j^{th} input vector. In a single unit neural network, the output y^j for an input vector $\mathbf{x}^j$ is computed as $y^j = sgn(\mathbf{w}^t\mathbf{x}^j - \theta)$. Such a network is referred to as a *single-layer perceptron* or a linear classifier.

Remark 1: Without loss of generality we include the threshold value θ in the weight vector as its last element and increase the dimension of every input vector by augmenting it with an entry that equals 1. Thus the weight vector is $\mathbf{w}^t = [w_1, ..., w_n, \theta]$ and the input vector is $\mathbf{x}^{j\,t} = [x_1{}^j, ..., x_n{}^j, 1]$.

Then the output of the perceptron can be written as $y^j = sgn(\sum_{k=1}^{n+1} w_k x_k^j) = sgn(\mathbf{w}^t\mathbf{x}^j)$. Let us assume that the input and the weight vectors are already augmented, then the problem of learning a linear classifier can be defined as follows

Proposition 2.1 *Given a set of input vectors* $(\mathbf{x}^1, ..., \mathbf{x}^m)$, $\mathbf{x}^i \in R^n$, *and a set of desired output*