CENTER FOR
PARALLEL OPTIMIZATION

MACHINE LEARNING VIA MATHEMATICAL PROGRAMMING

by

Kristin P. Bennett

Computer Sciences Technical Report #1167

July 1993

MACHINE LEARNING VIA MATHEMATICAL PROGRAMMING

By

Kristin P. Bennett

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy
(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN – MADISON

1993

Abstract

The purpose of this research is to create novel algorithms for inductive machine learning based on mathematical programming. Given k sets, the problem is to create a function which can be used to classify a future point as a member of one of the k sets. We propose using a general function called a multisurface. A multisurface consists of a set of surfaces (usually linear) in R^n that partition the input space into disjoint regions that are assigned an output classification. We show that a multisurface corresponds to a neural network as well as to a decision tree. Our general approach to creating a multisurface is to model each component of the multisurface as a system of linear inequalities and to minimize the errors in these inequalities. We consider a number of problems, including two-category discrimination, multicategory discrimination, and bilinear separation.

For two-class problems, we investigate the multisurface method of pattern recognition proposed by Mangasarian. This linear programming method is equivalent to neural network training and compares favorably with the standard back-propagation algorithm of neural networks. However, the method is sensitive to noise. In this thesis we propose a new robust linear programming formulation for creating linear discriminants. Computationally, the proposed approach is superior to other linear programs. A decision-tree algorithm using the new robust linear program is competitive with other decision-tree methods of machine learning.

We extend the two-class results to k-class problems. We propose a single linear program to create a piecewise-linear separator for k classes. To improve

computational speed, we reformulate the problem as a piecewise-quadratic minimization problem. We develop a parallel gradient distribution algorithm to solve the problem in parallel and test the algorithm on a parallel machine. The resulting piecewise-linear separators can also be used for decision-tree construction.

Finally, we address bilinear separability: Can two classes be completely separated using only two planes? This problem is NP-complete. We formulate the problem as a system of disjunctive linear inequalities that can be solved by bilinear programming. We develop a Frank-Wolfe-type algorithm for solving the bilinear program. Over 140 consecutive instances of the NP-complete problem were solved correctly using our algorithm.

Acknowledgements

My greatest thanks go to my advisor, Professor Olvi Mangasarian, without whom this thesis would not exist. It was a privilege and an inspiration to work with him. He will always be my model for how to do things right.

I would like to thank my committee, Professors Jude Shavlik, Robert Meyer, Michael Ferris, and Wei-Yin Loh, for their guidance on my thesis and the many hours I spent in their classes. Additional special thanks go to Jude Shavlik for his extensive contribution to my understanding of machine learning and for passing many suggestions my way during the last five years.

I thank my family and friends for their steadfast support, encouragement, and reality checks. My mother, Marjorie Bennett, set an outstanding example for me to follow by getting her own Ph.D. My husband, Scott Vandenberg, provided generous and indispensable moral and practical support.

This work was partially supported by an Air Force Laboratory Graduate Fellowship.

Contents

Chapter 1

Multisurface representation of the machine learning problem

A fundamental problem of machine learning is that of discriminating between the elements of two or more sets. Given k sets, $\mathcal{A}^i$, $i = 1, \ldots, k \geq 2$, in the n-dimensional real space R^n, the problem is to construct a function that maps the input space to the output classification space, i.e. a mapping from R^n into the set $\{1, 2, \ldots, k\}$. Furthermore, this function should perform well on the training set (the set of examples used to create the function) and should "generalize" well to the testing set (future points unseen in training). A model is proposed based on the concept of a multisurface (MS) for representing this problem. A multisurface consists of a set of surfaces (usually linear) in R^n that partition the input space into disjoint regions that are assigned an output classification. In order to construct the classification function, the multisurface is characterized by a set of inequalities whose solutions are obtained by minimizing a weighted sum of infeasibilities. Both neural networks and decision trees can be constructed using this approach. To construct the multisurfaces, we develop a family of algorithms that minimize the error of linear, piecewise-linear, and bilinear classification functions using mathematical programming. The resulting mathematical programs are used

as subproblems in greedy multisurface construction algorithms. The principal differences between this work and related greedy algorithms for decision trees and neural networks are the formulations and the algorithms used in the subproblems.

This chapter begins with a discussion of simple linear surfaces. Then multisurfaces are presented along with their relation to neural networks and decision trees. We briefly discuss some related work and then provide a summary of the problems investigated in this thesis. An explanation of our experimental methods and notation conclude the chapter.

1.1 Single linear surface

We begin with the case of a single linear surface. Given the sets $\mathcal{A}^1$ and $\mathcal{A}^2$ in R^n, the object is to construct a plane $wx = \gamma$ that separates the two sets. The vector $w \in R^n$ is the normal to the plane. The scalar γ determines the distance of the plane from the origin. The problem reduces to the following set of linear inequalities:

$$\begin{aligned} A_i^1 w - \gamma &> 0 \quad i = 1, \ldots, m \\ -A_j^2 w + \gamma &> 0 \quad j = 1, \ldots, k \end{aligned} \tag{1.1}$$

where A_i^1, $i = 1, \ldots, m$ are the points in $\mathcal{A}^1$, and A_j^2, $j = 1, \ldots, k$ are all the points in $\mathcal{A}^2$. If this set of inequalities is feasible then the sets $\mathcal{A}^1$ and $\mathcal{A}^2$ are linearly separable. Figure 1.1 depicts two linearly separable sets and the corresponding separating plane. If equation (1.1) is not feasible, the sets are linearly inseparable. The goal in this case is to find a plane which minimizes some measure of the infeasibilities or classification errors.

For the linearly separable case, there are many methods for constructing a plane which satisfies equation (1.1). However, problems arise for the linearly inseparable case. Nilsson [60], Duda-Fossum [19], Duda-Hart [20], and Fukunaga [28] considered iterative methods that are extensions of the perceptron algorithm [55] or the Motzkin-Schoenberg algorithm [56]. However, convergence of these methods is not known if a separating linear surface does not exist [28, p. 374].

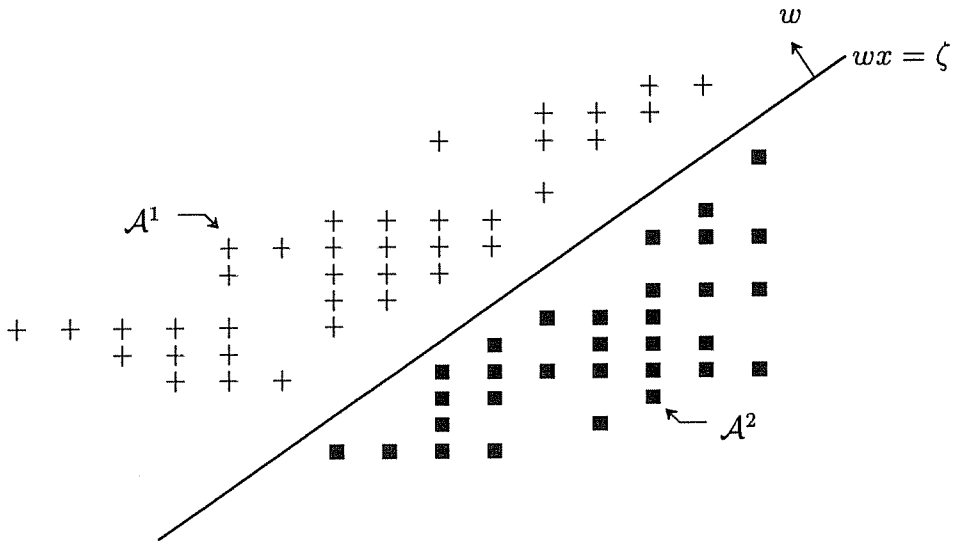


Figure 1.1: Two linearly separable sets and a separating plane

In fact, the perceptron algorithm may cycle and all that can be said is that the iterates remain bounded [12, 55]. Heuristic methods [29, 15] have been developed to get around this deficiency, but convergence is still not known. Several linear programming approaches which are similar to phase-1 in the simplex method exist [44, 32, 73, 31]. These approaches either fail in some cases or include ad hoc constraints which preclude some solutions. In Chapter 3, we propose a new robust linear program which avoids the pitfalls of previous approaches. Chapters 5 and 6 describe algorithms for the more general piecewise-linear case for multicategory classification.

1.2 Multisurface (MS) representation

Linear and piecewise-linear surfaces are frequently inadequate for many practical problems. Thus a more general separator is required. We introduce the idea of

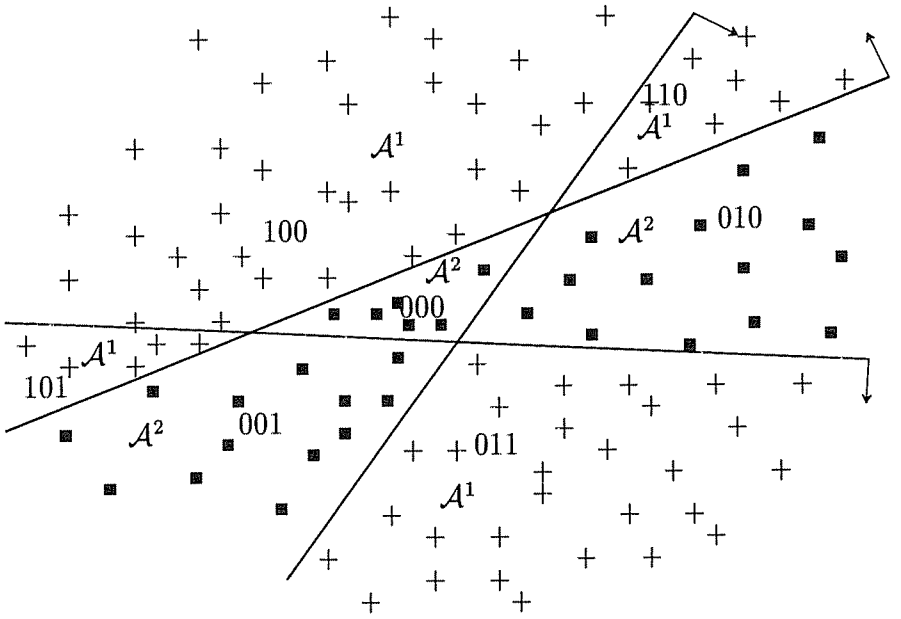


Figure 1.2: A multisurface which partitions R^2 into seven regions representing the sets $\mathcal{A}^1$ and $\mathcal{A}^2$

a multisurface which consists of many linear separators. Consider a multisurface consisting of h planes. The set of h planes divides the space into p polyhedral regions, $p \leq \sum_{i=0}^{n} \binom{h}{i}$ [30]. Each region can be represented by an h-bit binary number where each bit is determined by one of the planes. In particular, the i^{th} bit is 1 if the region lies on the positive side of the i^{th} plane, and 0 if the region is on the negative side of the i^{th} plane. Figure 1.2 shows an MS consisting of 3 linear surfaces which divide the space into 7 regions. Each region is assigned to one of 2 classes, $\mathcal{A}^1$ or $\mathcal{A}^2$.

For the purpose of this thesis, we limit our discussion to linear surfaces, namely planes. This is not very restrictive since nonlinear surfaces that are linear in their parameters (e.g. polynomials) can easily be accommodated by augmenting the

input space to contain polynomial combinations of the variables.

1.3 Relation to neural networks

We will first consider the simple case of the linear threshold unit, or perceptron [52, 68, 67], and then examine more general neural networks.

Each linear surface or plane represents a linear threshold unit (LTU) or perceptron, the building block of neural networks. An LTU is specified by a weight $w \in R^n$ (the normal of the plane) and a threshold $\zeta \in R$ (the location of the plane). For any vector x in the input space, the output of the LTU is 1 if $xw > \zeta$ or 0 if $xw \leq \zeta$, that is $step(wx - \zeta) := \begin{cases} 1 & if\ wx > \zeta \\ 0 & otherwise \end{cases}$. Thus the LTU indicates which of the two half spaces determined by the plane contains the point. A class is associated with each half space. Figure 1.3 depicts an LTU that represents the two half-spaces shown in Figure 1.1. A single LTU can only correctly recognize two linearly separable sets. Hence a neural network with many LTUs (called hidden units) is needed for more complex mappings.

The fundamental feedforward neural network (multilayer perceptron) [70, 78] is intimately related to multisurfaces. To demonstrate this relationship, we will construct a neural network with three hidden units and one output unit that is equivalent to the multisurface in Figure 1.2. The three LTUs (planes) in the figure become the three hidden units of the neural network. These three hidden units map each input of the network to a vertex of a 3-dimensional unit cube. Figure 1.4 illustrates this cube. Note that each vertex represents one of the polyhedral regions generated by the MS and is labelled with the appropriate class. From this intermediate representation the output units can be constructed. A single additional LTU, together with its weighted incoming arcs, maps the vertices of the cube to the final classification. This LTU and its incoming arcs are represented by the plane $3r_1+r_2+r_3 = 1.5$ in Figure 1.4. This same LTU becomes the output unit of the neural network representing the multisurface of Figure 1.2. This complete

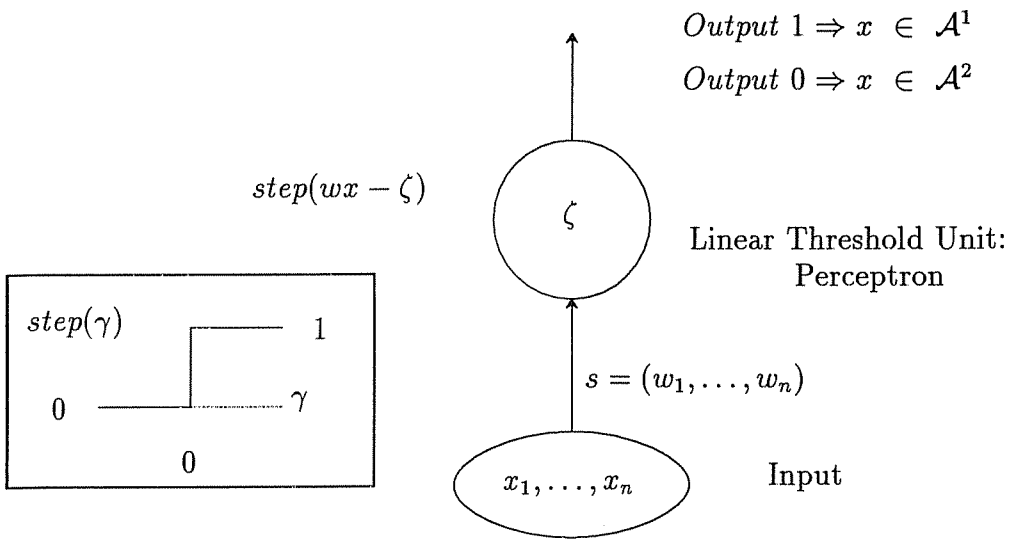


Figure 1.3: A linear threshold unit representing the separation in Figure 1.1

neural network is shown in Figure 1.5.

In general, multisurfaces with h surfaces correspond to neural networks with h units in the first hidden layer. The first hidden layer maps the input space R^n to an intermediate representation consisting of h-bit binary numbers or vertices of a h-dimensional cube. Each binary number or vertex is assigned to a class. Any additional hidden layers and the output unit(s) map the vertices of the cube or intermediate representation to the output classification. If the points in the intermediate representation are linearly separable by class, then only one output unit is required. If they are not linearly separable, then at most one additional layer of hidden units must be constructed. Typically if there are more than two classes, then there is an output unit for each class.

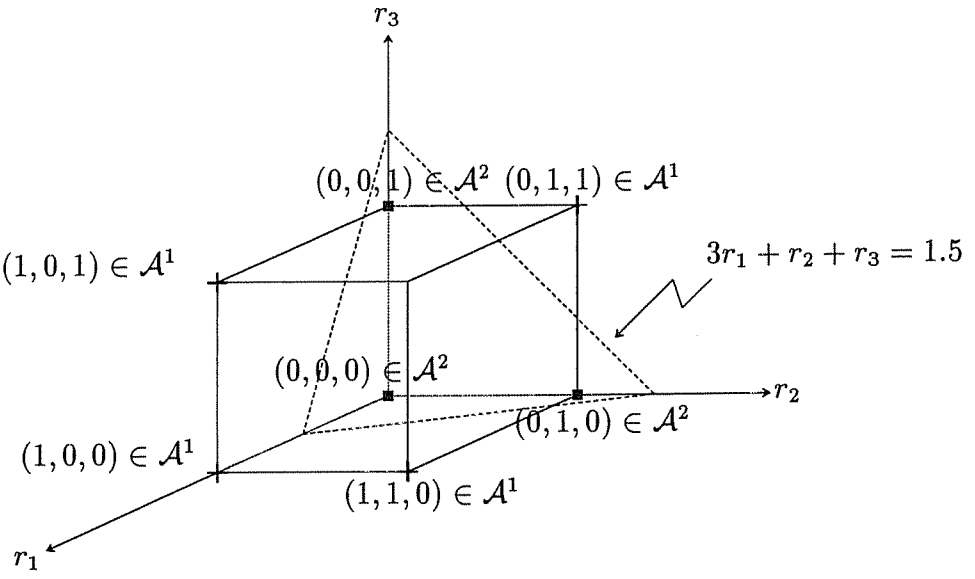


Figure 1.4: The vertices of the unit cube into which the sets $\mathcal{A}^1$ and $\mathcal{A}^2$ of Figure 1.3 are mapped, together with the plane separating them

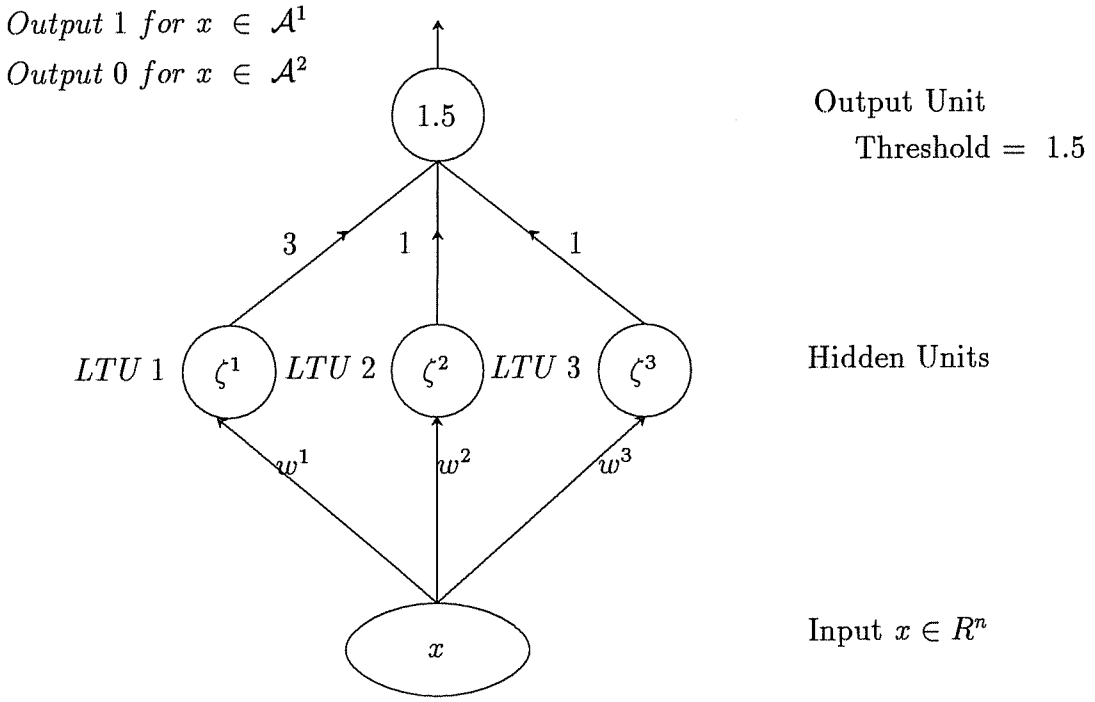


Figure 1.5: The feedforward neural network corresponding to Figures 1.2 and 1.4

1.4 Relation to decision trees

Multisurfaces and decision trees are also intimately related. An MS can be represented logically as a decision tree with linear threshold nodes. The nodes of the decision tree consist of a linear or other separator that has branches for each of the possible outcomes (classes). The leaves of the tree are the output classes. Conversely an MS is a geometric representation of a decision tree with linear threshold units. Each leaf of a decision tree corresponds to one or more regions of an MS, and every region of the MS is expressed as part of a leaf of the tree. A decision tree can be constructed from an MS by using the surfaces as the nodes within the decision tree. To create an MS from a decision tree, each decision node is represented as a surface and the leaves of the decision tree determine a class labelling of the resulting polyhedral regions. For example, the decision tree produced by the ID3 algorithm [63] can also be represented as an MS where the weights of the linear threshold units are vectors with only one non-zero element. Figure 1.6 depicts the MS shown in Figure 1.2 as a decision tree.

1.5 Greedy algorithms for constructing an MS

The optimization methods for linear and piecewise-linear separators can be extended to handle more complex multisurfaces by using greedy algorithms. There are many existing related greedy algorithms. Decision tree algorithms such as CART[14], FACT[42], and ID3[63] work by recursively dividing the input space into smaller regions which contain all or almost all points of a single class. Cascade Correlation [22] and other network construction algorithms [27, 54, 50] add hidden units to the neural network until a desired correctness is achieved. Neural tree algorithms [72, 76, 15, 71] use neural networks or perceptrons as nodes in decision trees. Roy *et al.* combines clustering and linear programming to construct neural networks one unit at a time[69]. With some minor variations, these all correspond to progressively adding surfaces to a multisurface until sufficient

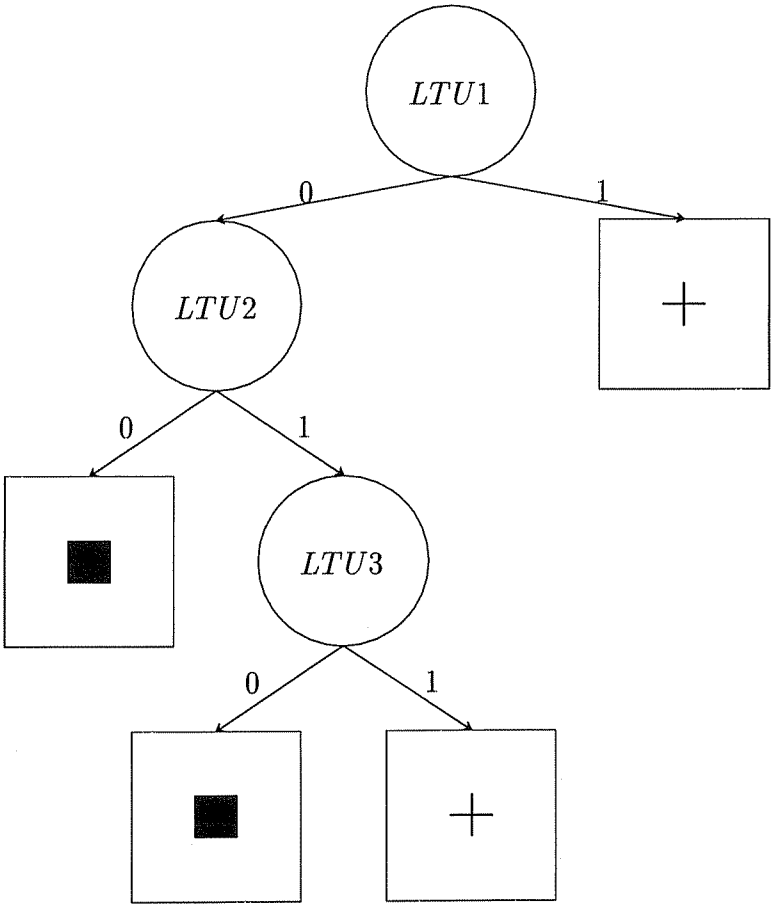


Figure 1.6: The 3-node decision tree corresponding to the MS depicted in Figure 1.2

accuracy is reached. The difference between our approach and this prior work is that we utilize novel algorithms based on mathematical programming to solve the subproblems required in greedy algorithms. The performance of our proposed subproblem algorithms has been proven theoretically and demonstrated computationally.

1.6 Problems investigated

In this thesis, we investigate a series of related alogorithms for constucting multisurfaces. We begin with multisurfaces for 2-class problems. These results are then extended to k-class problems (multicategory discrimination). Finally, we investigate an NP-complete 2-class bilinear separation problem.

1.6.1 Two-class discrimination

In Chapter 2, we investigate the Multisurface Method (MSM) of Pattern Recognition prosposed by Mangasarian [44]. MSM uses a series of linear programs in a greedy algorithm. Each linear program minimizes the maximum error in a set of inequalities. Thus it can be described as an ∞-norm approach. We show that the MSM procedure corresponds to constructing a feedforward neural network with partially fixed weights. The MSM separation can also be characterized as a decision tree. We compare MSM with backpropagation on a variety of different neural networks to evaluate the suitability of the architecture selected by the MSM greedy algorithm. We discuss the limitations of this algorithm, namely that it is intolerant to noise and that $2n$ linear programs (where n is the number of attributes) need to be solved at each iteration. Chapter 3 addresses these limitations.

A robust linear program which can create a linear discriminant by a single LP is proposed in Chapter 3. This LP minimizes the sum of the average infeasibilities of a set of highly structured inequalities. Careful choice of the objective function

ensures that the null solution results if and only if the means of the two sets are equal, and even in this case alternative nonzero solutions exist. Computational results show that this algorithm is superior to other LP approaches.

In Chapter 4, we use the robust LP proposed in Chapter 3 in a decision tree algorithm. The proposed method compares favorably with the CART [14] and ID3 [63] decision tree algorithms when applied to practical problems.

1.6.2 Multicategory discrimination

Chapter 5 extends the ideas of Chapters 3 and 4 to k-class problems. We define a piecewise-linear function that is the maximum of k affine functions. The function corresponds to a set of structured inequalities. If the inequalities are feasible then the k classes are piecewise-linear separable. If the classes are not piecewise-linear separable, a linear program is used to minimize the sum of the average errors or infeasibilities. Computational results show that this approach is very effective but also computationally costly for large problems.

In Chapter 6, we concentrate on minimizing the cost of constructing a piecewise-linear separator. By switching the error function to the average square error instead of the average sum of the absolute errors, the problem becomes a piecewise-quadratic minimization problem. The piecewise-quadratic minimization approach is considerably faster than the linear programming approach discussed in Chapter 5 and has the additional advantage that it is easily parallelizable. We adapt the parallel gradient distribution [47] method to this problem and implement the resulting algorithm on the CM5.

1.6.3 Bilinear separation

In the final chapter, we examine a 2-class NP-complete problem. The bilinear separability problem is that of determing whether two disjoints sets in R^n are separable by two planes. This problem can be formulated as two sets of disjunctive linear inequalities. We minimize the infeasiblilities using bilinear programming

and propose a simple algorithm similar to the Franke-Wolfe algorithm [26] to solve the bilinear programs. Computational results indicate that the algorithm is very effective for solving bilinearly separable problems.

1.7 Experimental methods

In this thesis, we describe experiments with several real world problems. Appendix A provides a brief description of each dataset, including the numbers of attributes, classes, and points. Recall that the goal is to create a function that generalizes well, i.e. classifies future points correctly. Thus we reserve points in a testing set, not used in training, to estimate generalization. Over repeated trials on each dataset, we measure training set accuracy, testing set accuracy, and training time. Two different methods were used for creating training and testing sets. In the first, random sampling, 67% of the points are randomly selected as the training set and 33% are reserved for testing. This is repeated for the desired amount of trials and the results are averaged over the trials. For the second, n-fold cross validation [40, 28], $\frac{1}{n}$ of the points are held out for testing and the remaining $\frac{n-1}{n}$ are used for training. The held-out $\frac{1}{n}$ points are tested on the resulting classifier. When n equals the total number of points, we refer to n-fold cross-validation as the leave-one-out method. Details of the experimental methods are provided with each experiment.

1.8 Mathematical notation

The following notation is used throughout this thesis. For a vector x in the n-dimensional real space R^n, x_+ will denote the vector in R^n with components $(x_+)_i := \max\{x_i, 0\}$, $i = 1, \ldots, n$. The notation $A \in R^{m \times n}$ will signify a real $m \times n$ matrix. For such a matrix, A^T will denote the transpose while A_i will denote the ith row. The 1-norm of x, $\sum_{i=1}^{n} |x_i|$, will be denoted by $\|x\|_1$. The 2-norm of x, $\sqrt{\sum_{i=1}^{n} x_i^2}$, will be denoted by $\|x\|_2$. A vector of ones in a space of

arbitrary dimension will be denoted by e. Additional notation used in individual chapters will be defined therein.

Chapter 2

Equivalence of neural networks and the multisurface method

2.1 Overview

In this chapter we examine the Multisurface Method (MSM) proposed by Mangasarian [44] for the separation of two disjoint pattern sets. MSM uses linear-programming-based subproblems which minimize the sum of the maximum classification error in each class. We show that the MSM algorithm produces a multisurface which can be modeled as either a decision tree or as a neural network. Our computational experiments indicate that training neural networks by MSM has the following advantages over the backpropagation algorithm (BP) [70]:

(a) Automatic determination of the number of hidden units.

(b) Achievement of 100% correctness on the training set, if desired.

(c) Faster training.

(d) Elimination of parameters from the training algorithm.

We begin with a review of the MSM linear programming approach for training a neural network for the binary classification of two disjoint point sets. In the

following two sections MSM's relationships with neural networks and decision trees are examined. The results of a computational comparison of MSM and BP are reported. The chapter concludes with an analysis of the strengths and weaknesses of the MSM approach. Note that many of the results of this chapter have been published [9].

2.2 The MSM classifier

Let the finite pattern sets $\mathcal{A}^1$ and $\mathcal{A}^2$ be two given disjoint training sets in the n-dimensional real feature space R^n. Let the cardinality of $\mathcal{A}^1$ and $\mathcal{A}^2$ be m^1 and m^2 respectively. The sets $\mathcal{A}^1$ and $\mathcal{A}^2$ are represented by the $m^1 \times n$ and $m^2 \times n$ matrices A^1 and A^2. If the convex hulls of the sets $\mathcal{A}^1$ and $\mathcal{A}^2$ do not intersect, or (equivalently) if they are linearly separable, a single linear program can generate a separating plane in polynomial time [36, 37] by solving the following problem [43]:

$$\begin{array}{ll} \underset{\alpha,\beta,w}{maximize} & \alpha - \beta \\ \text{subject to} & A^1 w \geq e\alpha, A^2 w \leq e\beta, -e \leq w \leq e \end{array} \tag{2.1}$$

where $w \in R^n$ is the weight vector associated with the separating plane, $\frac{\alpha+\beta}{2}$ is the threshold that locates the separating plane, and e is a vector of ones in a real space of arbitrary finite dimension. Note that this linear program generates the weight vector w and threshold $\frac{\alpha+\beta}{2}$ for a linear threshold unit which discriminates between two linearly separable sets.

When the sets $\mathcal{A}^1$ and $\mathcal{A}^2$ are not linearly separable, the null solution, $w = 0$, $\alpha = \beta = 0$, is optimal for (2.1), and no useful information is derived from the problem. Hence problem (2.1) must be modified to generate a sequence of different pairs of separating planes which constitute the MSM classifier. Each pair of parallel planes distinguishes a subset of $\mathcal{A}^1$ from a subset of $\mathcal{A}^2$. Total separation can be achieved by the MSM classifier which we now describe.

The multisurface consists of p pairs of planes with weight vectors $w^1, ..., w^p \in$

R^n, and thresholds $\alpha^1, ..., \alpha^{p-1}, \frac{\alpha^p+\beta^p}{2}$, and $\beta^1, ..., \beta^{p-1}, \frac{\alpha^p+\beta^p}{2}$, with $\alpha^i < \beta^i, i = 1, ..., p-1$ and $\alpha^p > \beta^p$. The last pair of planes coalesces into a single plane in order not to leave any part of the space unclassified. Classification is achieved as follows:

MSM Classifier for $x \in R^n$

for $i = 1$ to $p-1$
begin
    if $xw^i > \beta^i$ then $x \in A^1$; stop
    if $-xw^i > -\alpha^i$ then $x \in A^2$; stop
end
if $xw^p > \frac{\alpha^p+\beta^p}{2}$ then $x \in A^1$
else $x \in A^2$

Geometrically the MSM classifier corresponds to separation by a multisurface as illustrated in Figure 2.1 for a hypothetical case of two pairs of planes in two dimensions. Figure 2.2 depicts separation by four pairs of parallel planes of the Wisconsin Breast Cancer Data (WBCD), the actual clinical data described in Section 2.6 and Appendix A. This figure shows which points are separated by a given pair of planes and which points remain to be separated by a succeeding pair of planes. The top part of Figure 2.2 is the projection of 369 nine-dimensional data points on the two-dimensional space spanned by the normals, w^1 and w^2, to the first two pairs of separating planes. Note that w^1 and w^2 are not orthogonal to each other in general. In the bottom part of Figure 2.2, the points which were not classified by the first two pairs of parallel planes (i.e. the points in the parallelogram formed in the top graph) are projected on the two-dimensional space spanned by the normals, w^3 and w^4, to the last two pairs of separating planes.

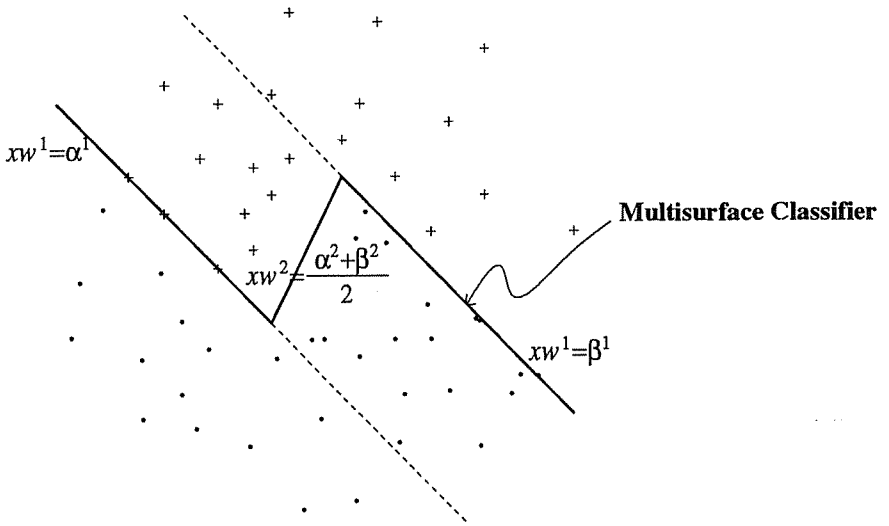


Figure 2.1: Geometric depiction of MSM classifier

2.3 MSM classifier as a neural network

We give now a novel representation of the MSM classifier as a trained feedforward neural network, which is depicted in Figure 2.3 and which can be trained efficiently by our linear programming approach. This network is composed of n input units, $2p-1$ hidden units, and 1 output unit. For $i = 1, ..., p-1$, w^i and $-w^i$ are the incoming weights to the $(2i-1)^{th}$ and the $(2i)^{th}$ hidden units with thresholds β^i and $-\alpha^i$ respectively. For $i = p-1$, w^p is the incoming weight to the $(2p-1)^{th}$ hidden unit with threshold $\frac{\alpha^p+\beta^p}{2}$. The weights on the arcs connecting the $2p-1$ hidden units to the output unit (see Figure 2.3) are predetermined such that the activation of the output unit is caused by the firing hidden unit with the lowest index. The threshold of the output unit is 0. To reduce clutter in Figure 2.3, the n arcs connecting the n input units to a hidden unit are consolidated into one arc.

Note that predetermined weights are used between the hidden units and the output unit. This may restrict the representational power of the network for a fixed number of units (i.e. the MSM classifier cannot represent all possible neural

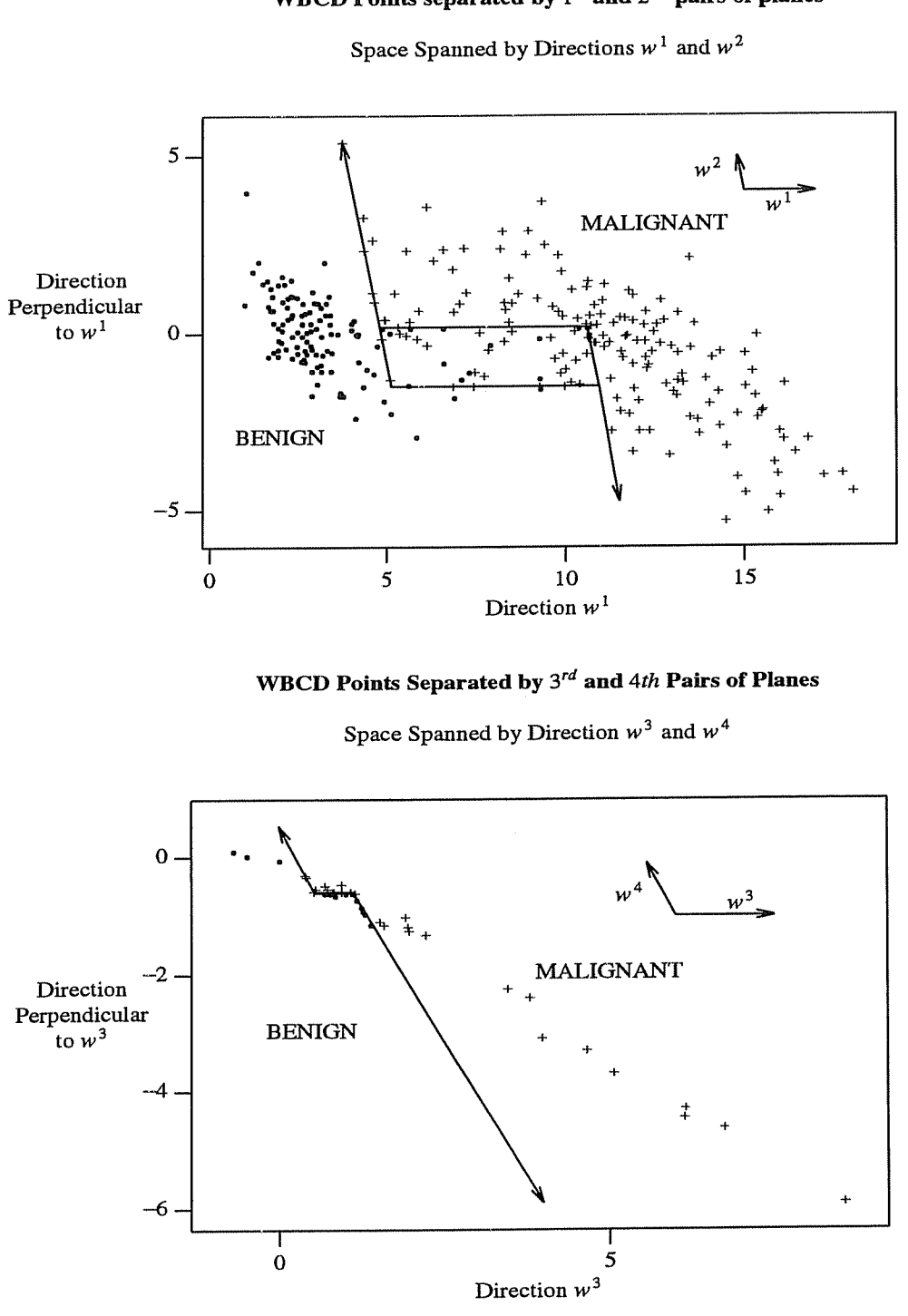


Figure 2.2: MSM classifier for Wisconsin Breast Cancer data

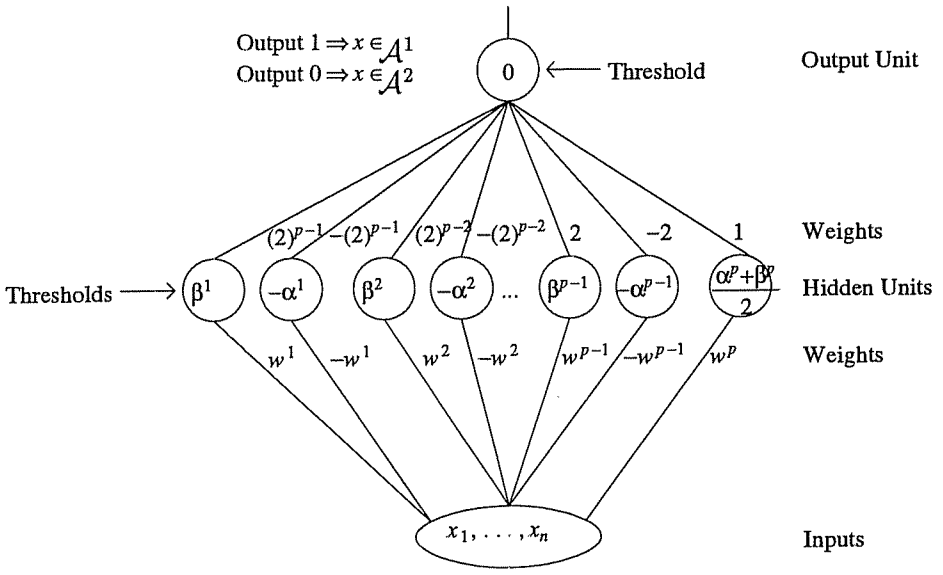


Figure 2.3: The MSM classifier as a neural network

networks with a given number of hidden units.) However, this does not seem to degrade the performance of the MSM neural network, nor does it prevent MSM from discriminating between any two disjoint point sets given a sufficient number of hidden units. MSM, a greedy algorithm, dynamically determines the number of hidden units required in order to correctly classify all the training examples. The performance of MSM is comparable to unrestricted networks trained with BP. In fact, the number of hidden units determined by MSM is a good estimate of the number required for a given classification task by an unconstrained neural network using BP. For example, consider the n-bit parity function, which takes an n-bit binary vector as its argument and returns 1 if the number of ones is odd, and 0 otherwise. This problem requires n hidden units for a feedforward network [70]. In our tests the MSM topology required n hidden units when n is odd, and $n+1$ units when n is even, and the network was trained in considerably less time using MSM than with BP. Figure 2.4 plots the execution time of MSM and BP

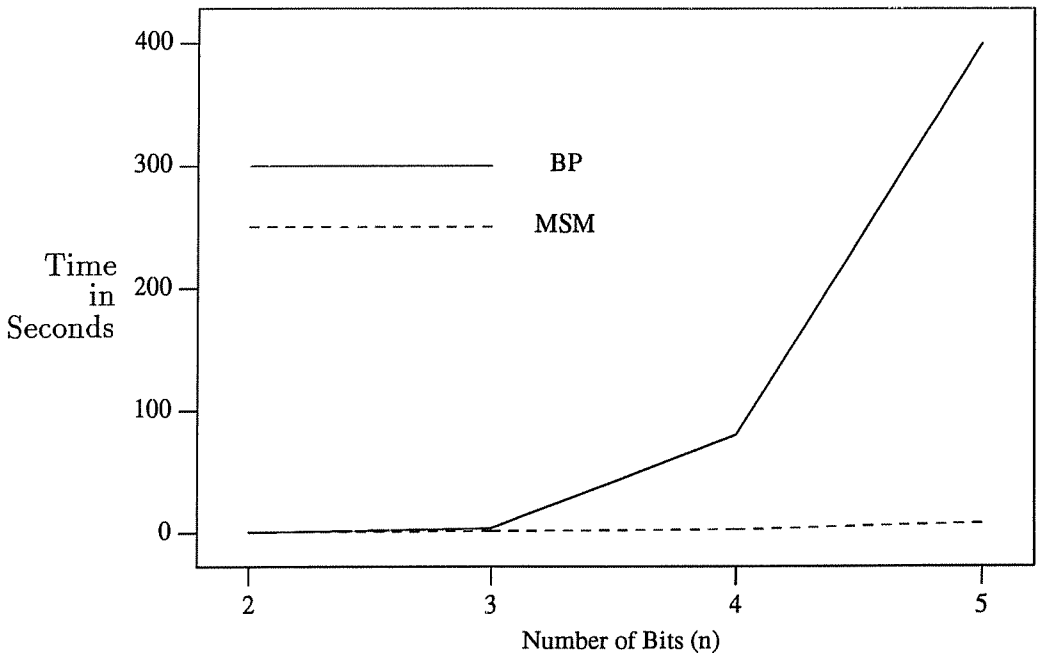


Figure 2.4: Execution times for BP and MSM on n-bit parity problems

on several n-bit parity problems solved on a DECstation 3100.

2.4 The MSM classifier as a decision tree

Logically the MSM classifier can be represented as a decision tree. The binary tree in Figure 2.5 depicts an MSM classifier with $p = 2$ pairs of planes (or three total planes) such as the one in Figure 2.1. Each node of the tree corresponds to one plane of the MSM classifier. Thus there are $2p - 1$ decision nodes where p is the number of parallel planes. The tree in Figure 2.5 can be extended to handle $p > 2$ pairs by replacing the dotted line with two decision nodes for each additional pair of planes. A decision tree is a convenient way to represent the MSM classifier.

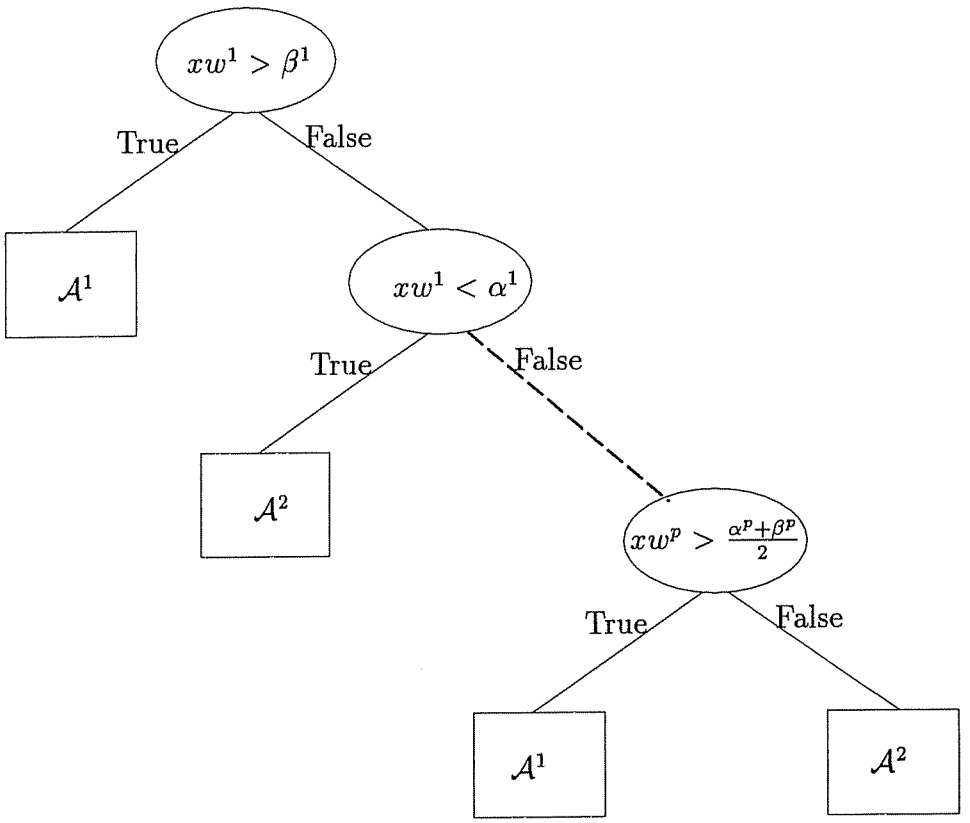


Figure 2.5: Decision tree representing MSM classifier with p pairs of planes

2.5 Training the MSM classifier

In order to generate the MSM classifier for the general case of linearly inseparable pattern sets, Problem (2.1) is modified as follows to ensure that w is nonzero:

$$\max_{1\leq i\leq n}\max_{w,\alpha,\beta}\{\alpha-\beta|A^1w\geq e\alpha,\ A^2w\leq e\beta,\ -e\leq w\leq e, w_i=\pm 1\}. \tag{2.2}$$

Problem (2.2) can be solved by solving $2n$ linear programs. Each Problem (2.2) is solved by a finite sequence of linear programs. For the i^{th} linear program(LP), the $w\neq 0$ constraint is replaced by $w_i=1$ if i is odd, or by $w_i=-1$ if i is even [44, 48]. Note that problem (2.2) is equivalent[48] to

$$\max_{w,\alpha,\beta}\{\alpha-\beta|A^1w\geq e\alpha,\ A^2w\leq e\beta,\ \|w\|_\infty=1\} \tag{2.3}$$

which in turn is easily seen to be the following problem

$$\max_{\|w\|_\infty=1}\left(\min_{1\leq i\leq m^1}A^1_iw-\max_{1\leq i\leq m^2}A^2_iw\right) \tag{2.4}$$

The plane found by solving Problem (2.2) separates portions of the sets $\mathcal{A}^1$ and $\mathcal{A}^2$ from each other. The points thus separated are removed from $\mathcal{A}^1$ and $\mathcal{A}^2$. This process is repeated until no points remain. Each time Problem (2.2) is solved, the sum of the maximum remaining errors within each set with respect to the plane $xw=\frac{\alpha+\beta}{2}$ is $\alpha-\beta$. Thus Problem (2.2) is an ∞-norm approach since it minimizes the maximum error within each class. The algorithm with an antidegeneracy procedure ensures that total separation of any two disjoint point sets can be achieved in polynomial time.

Note that unlike BP, there are no parameters in the learning algorithm that must be determined experimentally in MSM. For example, the number of hidden units within the MSM neural net is determined automatically by the program.

2.6 Computational comparison of BP and MSM on medical diagnosis problems

MSM has been used at the University of Wisconsin Hospitals for the diagnosis of breast cancer [49, 80, 82, 81]. We briefly discuss this application first and then give comparisons with BP.

The Wisconsin Breast Cancer Data (WBCD) set, developed by Dr. W. H. Wolberg, consists of nine measurements taken from fine needle aspirates from human breast tissue. On this data set, MSM was trained originally on 369 samples and was tested subsequently on 70 newly acquired samples all of which were classified correctly except one. At that time it was retrained and since then it has correctly classified all 48 subsequent samples. For reliability in medical diagnosis applications, 100% correctness on the training and testing sets is very desirable for noise-free data. MSM achieved 100% training set correctness on this data set unlike other approaches such as BP, statistical pattern separation and other decision tree approaches [83]. It is important to emphasize that in medical diagnosis, classification is performed on all available data and the classifier is then used on incoming data. Table 2.6 compares the best results obtained by BP as implemented in [51] with the results from MSM on the WBCD. It is interesting to note that BP does not achieve the correctness rate on either training or testing sets achieved by MSM.

Additional experiments have been conducted comparing the performance of BP and MSM on the WBCD and the Cleveland Heart Disease Data Set [18] described in Appendix A. Random splitting into 67% training and 33% testing sets was used. The results were averaged over 10 trials. These results are depicted in Figures 2.6 and 2.7.

From these figures we draw the following conclusions:

(a) 100% correctness was always achieved by MSM on the training set but not by BP.

Table 2.1: Comparison of MSM and BP on the Wisconsin Breast Cancer data

	MSM	BP
Training Time (seconds)	108.0	469.5
Training Set Correctness (%)	100.0	98.9
Testing Set Correctness (%)	98.3	94.9
Number of Hidden Units	7	6

Training Set Size = 369 Testing Set Size = 118

(b) On the testing sets, MSM achieved correctness rates which are within 4% of the correctness rates of BP. The higher discrepancies occurred on the relatively noisy Cleveland Heart Disease Data set.

(c) The number of hidden units, which is determined automatically by MSM, is a good estimate for the number of hidden units required using BP to achieve optimal training and testing set correctness with minimal training time.

(d) The time to train MSM is consistently much less than for BP. If we take into account that BP requires experimentation to determine the optimal values of learning parameters and the number of hidden units, the difference becomes more pronounced.

2.7 Strengths of MSM

To sum up, MSM is capable of training a neural network or decision tree and determining an appropriate number of hidden units or decision nodes. For any

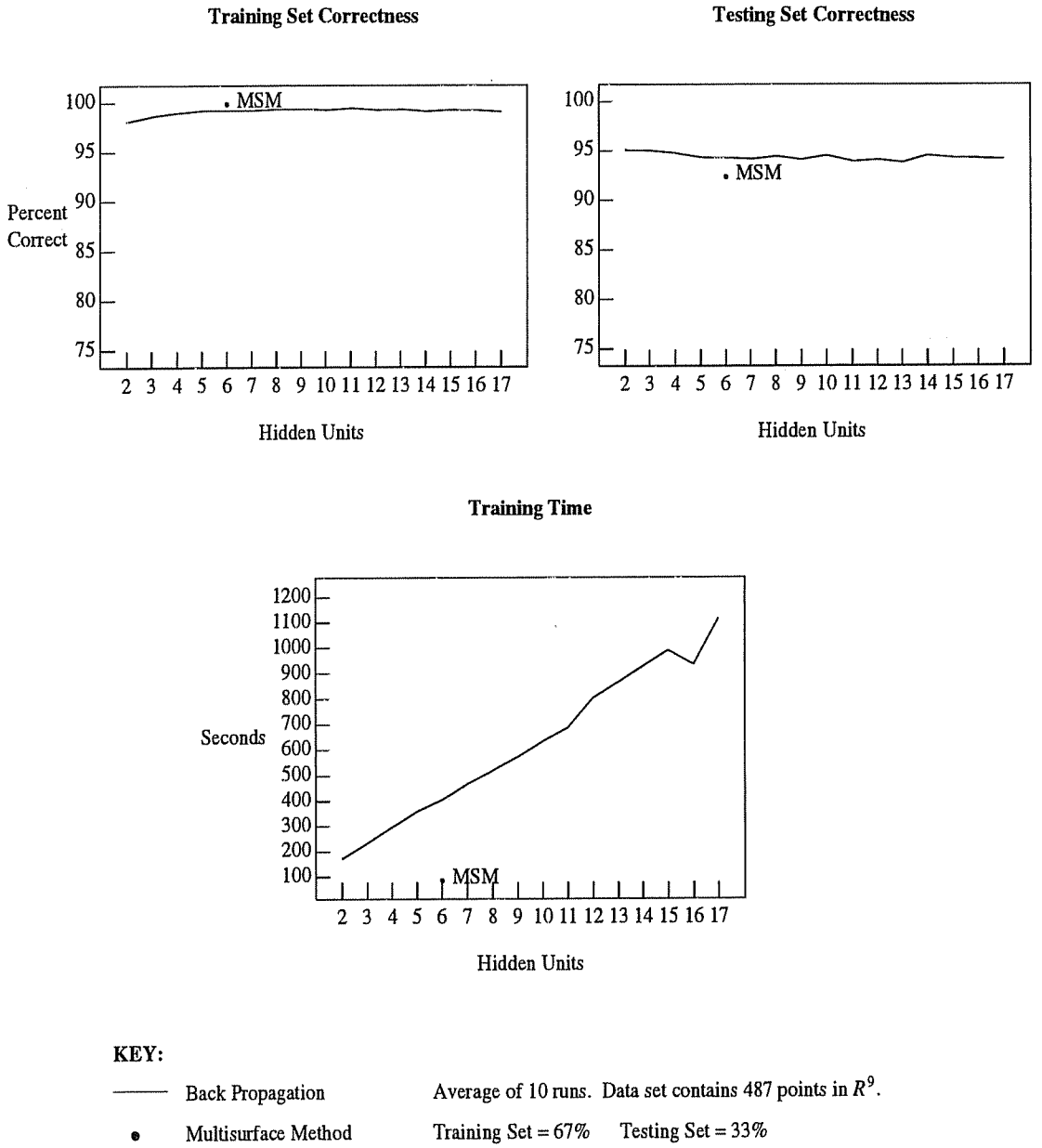


Figure 2.6: Results of MSM and backpropagation on the Wisconsin Breast Cancer data

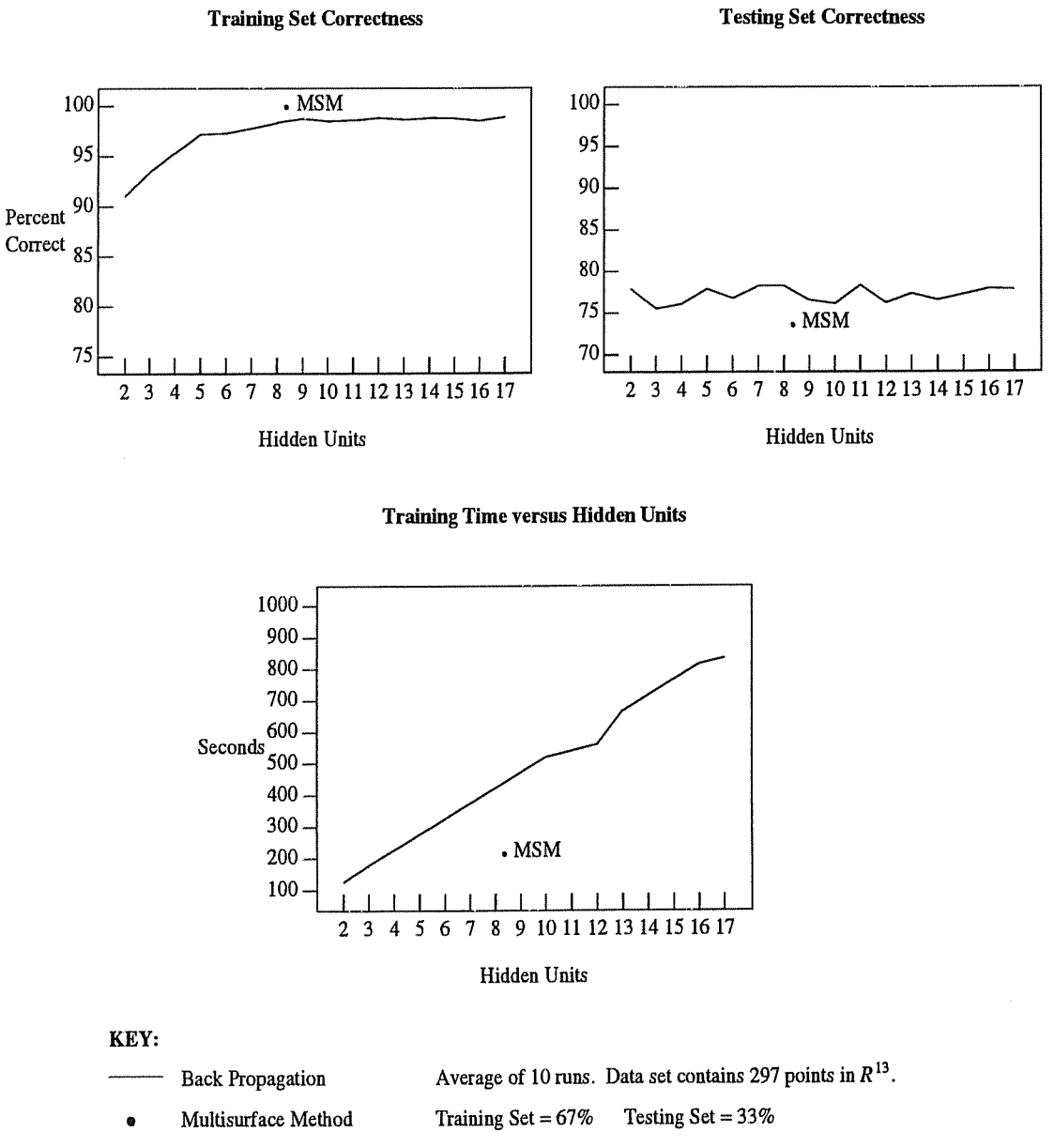


Figure 2.7: Results of MSM and backpropagation on the Cleveland Heart Disease data

two disjoint sets, the algorithm achieves 100% correctness on the training set in polynomial time. Unlike backpropagation, the algorithm requires no choices of parameters. On the problems tested, it was also faster than back propagation. A very successful diagnosis program based on this method has been used at University of Wisconsin Hospitals for breast cancer diagnosis [49, 80, 82, 81].

2.8 Extensions

The computational time and generalization accuracy of MSM could be improved. For large problems requiring many planes, the computational cost is relatively high, since for each pair of planes the algorithm solves $2n$ linear programs with $m+2n+1$ constraints in $n+2$ variables, where m is the total number of points and n is the dimension of the feature space. In the second experiment, MSM did not generalize as well as BP on both of the datasets. Looking ahead to Chapter 3, a single linear discriminant generalizes quite well on both the WBCD and Cleveland Heart Disease datasets. MSM uses too many planes and this causes a reduction in the training set accuracy. This is called *overfitting* [34, p. 147].

As in approximation theory, the simplest function possible should be used to fit a given set of points. The algorithm as written adds planes until 100% correctness is achieved on the training set. However, relaxing this requirement does not appreciably improve the performance of the algorithm. The main underlying difficulty is that the subproblem (2.2) is very sensitive to outliers and noisy points. For the linearly-inseparable case, the MSM linear programs work by "squeezing" together the planes $xw = \alpha$ and $xw = \beta$, while maintaining $A^1w \geq e\alpha$ and $A^2w \leq e\beta$. A few spurious points can stop the planes, and lead to fitting noisy data. Hence outliers will determine the MSM classifier. This is especially a problem with noisier data sets. See Figure 2.8 for an example. A new method for determining a linear separator which is faster and more robust in the presence of noise is needed. This is the subject of the next chapter.

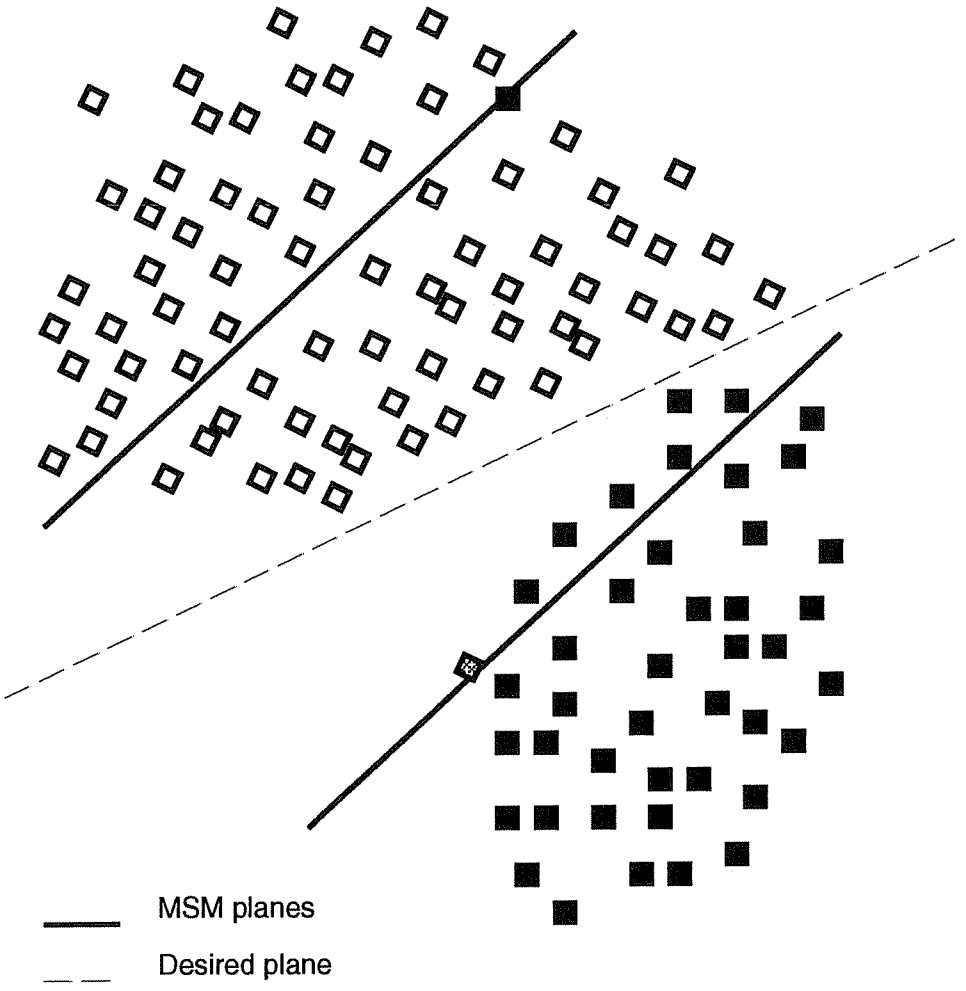


Figure 2.8: Two outlying points determining an MSM classifier

Chapter 3

Robust linear programming separation

3.1 Overview

We now propose a new linear program for linear discrimination of the sets $\mathcal{A}^1$ and $\mathcal{A}^2$. This linear program minimizes the average of the misclassified points in sets $\mathcal{A}^1$ and $\mathcal{A}^2$ (a 1-norm approach). Our principal objective was to formulate a single linear program with the following properties:

(i) If the convex hulls of $\mathcal{A}^1$ and $\mathcal{A}^2$ are disjoint, a strictly separating plane is obtained.

(ii) If the convex hulls of $\mathcal{A}^1$ and $\mathcal{A}^2$ intersect, a plane is obtained that minimizes some measure of the misclassified points, for **all** possible cases.

(iii) No extraneous constraints are imposed on the linear program that rule out any specific case from consideration.

(iv) The resulting linear discriminant generalizes well in practice.

Most linear programming formulations [43, 32, 73, 31] have property (i), however, no previous ones (to our knowledge) have both properties (ii) and (iii). For

example, the linear programs 2.1 and 2.2 [43], described in Chapter 2, fail to satisfy property (ii) for all linearly inseparable cases, while Smith's linear program [73] fails to satisfy (ii) when uniform weights are used in its objective function as originally proposed. The linear programs[32, 31] fail in satisfying both (ii) and (iii). Our linear programming formulation, on the other hand, has all three properties (i-iii). Computational results establish that our linear program also satisfies property (iv).

It is interesting to note that our proposed linear program (3.15) will always generate some error-minimizing plane even in the usually troublesome case when the means of the two sets are identical. For this case, among the possible solutions to our linear program is the null solution. However, this null solution is never unique for our linear program and thus a useful alternative solution is always available. For example, such an alternative, the 45° line, is obtained computationally by our linear program for the classical counterexample of linear inseparability: the Exclusive-Or example[70]. (See Example 3.9 below.)

We outline our results now. In Section 3.2 we state our linear program (3.15) and establish that it possesses properties (i)-(iii) above in Theorems 3.7 and 3.8. This contrasts with other linear program formulations. In Example 3.10 we show that ($\bar{w} = 0,\ \bar{\gamma} = 1$) uniquely solves Smith's linear program ((3.14) with $\delta_1 = \delta_2$) and hence property (ii) is violated. Similarly, in Remark 3.11 we give an example which violates property (ii) for Grinold's linear program[32] (3.27) and give conditions under which this is always true. In Section 3.3 we report on some computational results using our proposed linear program on the Wisconsin Breast Cancer Database and the Cleveland Heart Disease Database. Note that this material has appeared previously [10].

3.2 A robust linear programming separation

Our linear program is based on the following error-minimizing optimization problem:

$$\min_{w,\gamma} \frac{1}{m^1}\|(-A^1w + e\gamma + e)_+\|_1 + \frac{1}{m^2}\|(A^2w - e\gamma + e)_+\|_1 \tag{3.1}$$

where $A^1 \in R^{m^1 \times n}$ represents the m^1 points of the set $\mathcal{A}^1$, $A^2 \in R^{m^2 \times n}$ represents the m^2 points of the set $\mathcal{A}^2$, w is the n-dimensional "weight" vector representing the normal to the optimal "separating" plane, and the real number γ is a threshold that gives the location of the separating plane: $wx = \gamma$. The choice of the weights $\frac{1}{m^1}$ and $\frac{1}{m^2}$ in (3.1) is critical (as we shall demonstrate below in Theorems 3.7 and 3.8) in that it sets our approach apart from Smith's linear program[73] where equal weights were proposed, and from other linear programming formulations[43, 32, 31]. Our choice is a "natural" one in that the useless null solution $w = 0$ is not encountered computationally for linearly inseparable sets. This is theoretically justified (Theorem 3.7 below) because $w = 0$ cannot be a solution unless the following equality between the arithmetic means of $\mathcal{A}^1$ and $\mathcal{A}^2$ holds

$$\frac{eA^1}{m^1} = \frac{eA^2}{m^2} \tag{3.2}$$

However in this case, it is guaranteed that a nonzero optimal w exists in addition to $w = 0$ (Theorem 3.8 below).

We begin our analysis by justifying the use of the optimization problem (3.1) which minimizes the average of the misclassified points of $\mathcal{A}^1$ and $\mathcal{A}^2$ by the separating plane $xw = \gamma$. We now define linear separability for concreteness.

Definition 3.3 ***(Linear Separability)*** *The point sets $\mathcal{A}^1$ and $\mathcal{A}^2$, represented by the matrices $A^1 \in R^{m^1 \times n}$ and $A^2 \in R^{m^2 \times n}$ respectively, are linearly separable if and only if*

$$\min_{1 \le i \le m^1} A_i^1 v > \max_{1 \le i \le m^2} A_i^2 v \quad \textit{for some} \quad v \in R^n \tag{3.3}$$

or equivalently

$$A^1w \ge e\gamma + e, \quad e\gamma - e \ge A^2w \quad \textit{for some} \quad w \in R^n, \ \gamma \in R \tag{3.4}$$

That (3.4) implies (3.3) is evident. To see the converse just note the relations

$$w := 2v/\nu,\ \nu := \min_{1\le i\le m^1} A_i^1 v - \max_{1\le i\le m^2} A^2 > 0,\ \gamma := \min_{1\le i\le m^1} \frac{A_i^1 v}{\nu} + \max_{1\le i\le m^2} \frac{A^2 v}{\nu} \quad (3.5)$$

Note also that when the sets $\mathcal{A}^1$ and $\mathcal{A}^2$ are linearly separable, as defined by (3.4), the plane

$$\{x|wx = \gamma\} \quad (3.6)$$

is a strict separating plane with

$$A^1 w > e\gamma \quad and \quad e\gamma > A^2 w \quad (3.7)$$

With the above definitions the following lemma becomes evident.

Lemma 3.4 *The sets $\mathcal{A}^1$ and $\mathcal{A}^2$ represented by $A^1 \in R^{m^1 \times n}$ and $A^2 \in R^{m^2 \times n}$ respectively are linearly separable if and only if the minimum value of (3.1) is zero, in which case $(w = 0,\ \gamma)$ cannot be optimal.*

Proof. Note that the minimum of (3.1) is zero if and only if

$$-A^1 w + e\gamma + e \le 0 \ \ and \ \ A^2 w - e\gamma + e \le 0$$

which is equivalent to the linear separability definition (3.3). To see that $(w = 0,\ \gamma)$ cannot be optimal for (3.1), note that if we set $w = 0$ in (3.1) we get

$$\min_{\gamma} (1+\gamma)_+ + (1-\gamma)_+ = 2 > 0 \quad (3.8)$$

which contradicts the requirement that the minimum of (3.1) be zero for linearly separable $\mathcal{A}^1$ and $\mathcal{A}^2$. □

The import of Lemma 3.4 is that the optimization problem (3.1), which is equivalent to the linear program (3.15) below, will always generate a separating plane $wx = \gamma$ for linearly separable sets $\mathcal{A}^1$ and $\mathcal{A}^2$. For linearly inseparable sets $\mathcal{A}^1$ and $\mathcal{A}^2$, the optimization problem (3.1) will generate an optimal separating plane $wx = \gamma$ which minimizes the average violations

$$\frac{1}{m^1}\sum_{i=1}^{m^1} (-A_i^1 w + \gamma + 1)_+ + \frac{1}{m^2}\sum_{i=1}^{m^2} (A_i^2 w - \gamma + 1)_+ \quad (3.9)$$

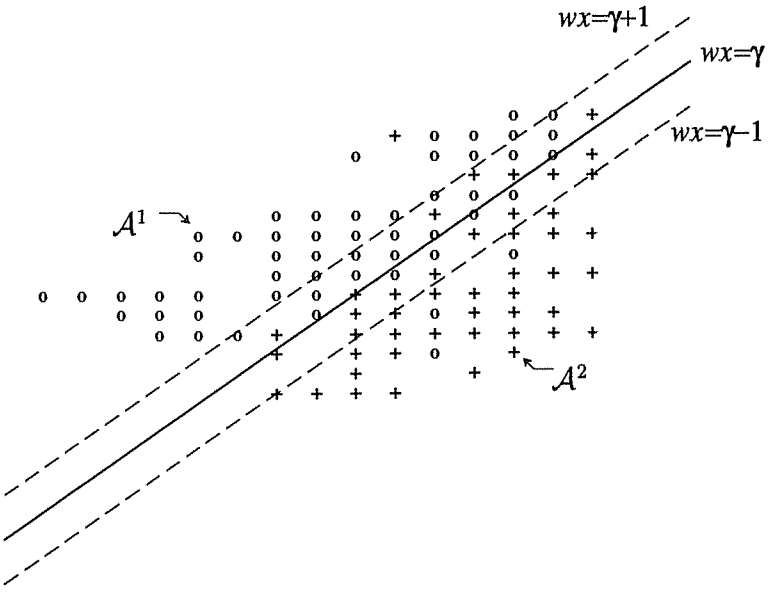


Figure 3.1: An optimal separator $wx = \gamma$ for linearly inseparable sets: $\mathcal{A}^1$ (o) and $\mathcal{A}^2$ $(+)$

of points of $\mathcal{A}^1$ which lie on the wrong side of the plane $wx = \gamma + 1$, that is in $\{x|wx < \gamma + 1\}$, and of points of $\mathcal{A}^2$ which lie on the wrong side of the plane $wx = \gamma - 1$, that is in $\{x|wx > \gamma - 1\}$. See Figure 3.1.

Note also that the location of the plane $wx = \gamma$ obtained by minimizing the average violations (3.9) can be further optimized by holding w fixed at the optimal value and solving the one-dimensional optimization problem in γ

$$\min_{\min_i A_i^1 w \le \gamma \le \max_j A^2 w} \frac{1}{m^1} \sum_{i=1}^{m^1} (-A_i^1 w + \gamma)_+ + \frac{1}{m^2} \sum_{i=1}^{m^2} (A^2 w - \gamma)_+ \tag{3.10}$$

This "secondary" optimization is not necessary in general, but for some problems it improves the location of the optimal separator for a fixed orientation of the planes. The objective of the one-dimensional problem (3.10) is a piecewise-linear convex function which can be easily minimized by evaluating the function at the breakpoints $\gamma = A_1^1 w, ..., A_{m^1}^1 w, A_1^2 w, ..., A_{m^2}^2 w$.

In order to set up the equivalent linear programming formulation to (3.1) we state first a simple lemma that relates a norm minimization problem such as (3.1) to a constrained optimization problem devoid of norms of plus-functions.

Lemma 3.5 *Let $g: R^n \rightarrow R^{m^1}$, $h: R^n \rightarrow R^{m^2}$ and let S be a subset of R^n. The problems*

$$\min_{x \in S} \|g(x)_+\|_1 + \|h(x)_+\|_1 \tag{3.11}$$

$$\min_{x \in S} \{ey + ez | y \geq g(x),\ y \geq 0,\ z \geq h(x),\ z \geq 0\} \tag{3.12}$$

have identical solution sets.

Proof. The equivalence follows by noting that for the minimization problem (3.12), the optimal y, z *and* x must be related through the equalities $y = g(x)_+$, $z = h(x)_+$. □

Using this lemma we can state an equivalent linear programming formulation to (3.1) as follows.

Proposition 3.6 *For $\delta_1 > 0$, $\delta_2 > 0$, the error-minimizing problem*

$$\min_{w,\gamma} \delta_1 \|(-A^1 w + e\gamma + e)_+\|_1 + \delta_2 \|(A^2 w - e\gamma + e)_+\|_1 \tag{3.13}$$

is equivalent to the linear program

$$\min_{w,\gamma,y,z} \{\delta_1 ey + \delta_2 ez | A^1 w - e\gamma + y \geq e,\ -A^2 w + e\gamma + z \geq e,\ y \geq 0,\ z \geq 0\} \tag{3.14}$$

The linear program (3.14) originally proposed by Smith[73] with equal weights $\delta_1 = \delta_2 = \dfrac{1}{m^1 + m^2}$ does not possess all the properties found in our linear program with $\delta_1 = \dfrac{1}{m^1}$ and $\delta_2 = \dfrac{1}{m^2}$:

$$\min_{w,\gamma,y,z} \left\{ \frac{ey}{m^1} + \frac{ez}{m^2} \middle| A^1 w - e\gamma + y \geq e,\ -A^2 w + e\gamma + z \geq e,\ y \geq 0,\ z \geq 0 \right\} \tag{3.15}$$

The principal advantage that (3.15) has over other linear programs, including Smith's, is that for the linearly inseparable case it will *always* generate a nontrivial

w *without* an extraneous constraint. To our knowledge no other linear programming formulation has this property for linearly inseparable sets. We establish this property by first considering the linear program (3.14) for arbitrary positive weights δ_1 and δ_2 and then showing under what conditions $w = 0$ constitutes a solution of the problem.

Theorem 3.7 ***(Occurrence of the null solution*** w ***in LP (3.14))*** *Let* $\delta_2 m^2 \geq \delta_1 m^1$. *The linear program (3.14) has a solution* $(w = 0,\ \gamma, y, z)$ *if and only if*

$$\frac{eA^1}{m^1} = vA^2,\ v \geq 0,\ ev = 1,\ v \leq \frac{\delta_2 e}{\delta_1 m^1}, \tag{3.16}$$

that is, if the arithmetic mean of the points in $\mathcal{A}^1$ *equals a convex combination of some points of* $\mathcal{A}^2$. *When* $\delta_2 m^2 = \delta_1 m^1$, *(3.16) degenerates to*

$$\frac{eA^1}{m^1} = \frac{eA^2}{m^2}, \tag{3.17}$$

that is, the arithmetic mean of the points in $\mathcal{A}^1$ *equals the arithmetic mean of the points in* $\mathcal{A}^2$.

Proof. We note first that $\delta_2 m^1 \geq \delta_1 m^2$ does not result in any loss of generality because the roles of the sets $\mathcal{A}^1$ and $\mathcal{A}^2$ can be switched to obtain this inequality. Consider now the dual to the linear program (3.14)

$$\max_{u,v} \{eu + ev | {A^1}^T u - {A^2}^T v = 0,\ -eu + ev = 0,\ 0 \leq u \leq \delta_1 e,\ 0 \leq v \leq \delta_2 e\} \tag{3.18}$$

The point $(w = 0, \delta, y, z)$ is optimal for the primal problem (3.14) if and only if

$$\begin{aligned} 2\delta_1 m^1 &= \min_\gamma\ \delta_1 m^1 (1+\gamma)_+ + \delta_2 m^2 (1-\gamma)_+ \\ &= \min_{\gamma,y,z} \{\delta_1 ey + \delta_2 ez | -e\gamma + y \geq e,\ e\gamma + z \geq e, (y,z) \geq 0\} \end{aligned} \tag{3.19}$$

$$= \max_{u,v} \{eu + ev | {A^1}^T u - {A^2}^T v = 0,\ -eu + ev = 0,\ 0 \leq u \leq \delta_1 e,\ 0 \leq v \leq \delta_2 e\} \tag{3.20}$$

Since $eu = ev$ and $eu + ev = 2\delta_1 m^1$, it follows that $eu = ev = \delta_1 m^1$. Since $0 \leq u \leq \delta_1 e$, and so if any $u_i < \delta_1$, then $eu < \delta_1 m^1$, contradicting $eu = \delta_1 m^1$.

Hence $u = \delta_1 e$ and $ev = eu = \delta_1 m^1$. By normalizing u and v by dividing by $\delta_1 m^1$ we obtain (3.16). When $\delta_2 m^2 = \delta_1 m^1$, then from (3.16) we have $0 \leq v \leq \frac{e}{m^2}$. Since $ev = 1$, it follows that $v = \frac{e}{m^2}$ and (3.17) follows from (3.16). □

This theorem gives a theoretical explanation to some observed computational experience, namely that Smith's linear program (3.14) with $\delta_1 = \delta_2$, ended sometimes with the useless null w for real-world linearly inseparable problems, whereas our linear program (3.15) never did. The reason for that is the rarity of the satisfaction of (3.17) by real problems in contrast to the possibly frequent satisfaction of (3.16).

We now proceed to our next results which show when the null vector $w = 0$ constitutes a solution to the linear program (3.14). Except for our proposed choice of $\delta_1 = \frac{1}{m^1}$ and $\delta_2 = \frac{1}{m^2}$, such $w = 0$ solutions can be unique and nothing can be done to alter them. (See Example 3.10 below.) However, for our linear program (3.15), even when the null w occurs in the rare case of (3.17), e.g. in the contrived but classical Exclusive-Or example[70], there always exists an alternate non-null optimal w. (see Example 3.9 below). These results are presented in the following theorem, examples, and remarks.

Theorem 3.8 ***(Nonuniqueness of the null w solution to the linear program (3.15))*** *The solution $(w = 0, \gamma, y, z)$ to (3.15) is not unique in w.*

Proof. Note from the first equality of (3.19) with $\delta_1 m = \delta_2 k = 1$ that when $(\bar{w} = 0, \gamma, y, z)$ is a solution to (3.15), then $\bar{\gamma}$ can be any point in [-1, 1]. In particular, take $\bar{\gamma} = 0$. Then for this choice of $\bar{w} = 0$, $\bar{\gamma} = 0$, the corresponding optimal y, z for (2.11) are $\bar{y} = e$, $\bar{z} = e$ and the active constraints are the first two constraints of (3.15). Hence $(\bar{w}, \bar{\gamma}, \bar{y}, \bar{z})$ is unique in $\bar{w}$ if and only if the following system of linear inequalities has *no* solution (w, γ, y, z)

$$\begin{array}{rcl} \frac{e}{m^1} y + \frac{e}{m^2} z & \leq & \frac{e}{m^1} \bar{y} + \frac{\bar{z}}{m^2} \bar{z} \\ A^1 w - e\gamma + y & \geq & A^1 \bar{w} - e\bar{\gamma} + \bar{y} = e \\ -A^2 w + e\gamma + z & \geq & -A^2 \bar{w} + e\bar{\gamma} + \bar{z} = e \\ w & \neq & \bar{w} \end{array} \tag{3.21}$$

This is equivalent to the following system of linear inequalities having no solution (w, γ, y, z) for *each* h in R^n :

$$\begin{array}{rl} -\frac{e}{m^1}(y-\bar{y}) - \frac{e}{m^2}(z-\bar{z}) & \geq 0 \\ A^1(w-\bar{w}) - e(\gamma-\bar{\gamma}) + (y-\bar{y}) & \geq 0 \\ -A^2(w-\bar{w}) + e(\gamma-\bar{\gamma}) + (z-\bar{z}) & \geq 0 \\ -h(w-\bar{w}) & > 0 \end{array} \tag{3.22}$$

By Motzkin's theorem of the alternative[45], (3.22) has no solution for a given h in R^n if and only if the following system of linear inequalities *does* have a solution (ζ, u, v) for that h in R^n :

$$\begin{array}{rl} {A^1}^T u - {A^2}^T v & = h \\ -eu + ev & = 0 \\ -\frac{1}{m^1} e\zeta + u & = 0 \\ -\frac{1}{m^2} e\zeta + v & = 0 \\ \zeta, u, v & \geq 0 \end{array} \tag{3.23}$$

Obviously it is possible to choose h in R^n such that (3.23) has no solution, since there are h in R^n that cannot be written as:

$$h = \frac{{A^1}^T e\zeta}{m^1} - \frac{{A^2}^T e\zeta}{m^2}, \ \zeta \geq 0. \tag{3.24}$$

Hence (3.22) has a solution for some h in R^n. Consequently (3.21) has a solution and $\bar{w} = 0$ is not unique. □

We now apply this theorem to the classical Exclusive-Or example[70] for which condition (3.17) is satisfied. In this case $(\bar{w} = 0, \bar{\gamma}, \bar{y}, \bar{z})$ is a solution to (3.15), however, it is not unique in $\bar{w} = 0$.

Example 3.9 ***(Exclusive-Or)***

$$A^1 = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}; \quad A^2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

For this example $\frac{eA^1}{2} = \frac{eA^2}{2}$ and $(\bar{w} = 0,\ \bar{\gamma} = 0,\ \bar{y} = e,\ \bar{z} = e)$ is a solution to the linear program (3.15) which can also be written in the equivalent (3.1) formulation of

$$\min_{w,\gamma} \frac{1}{2}[(1+\gamma)_+ + (1+\gamma-w_1-w_2)_+ + (1-\gamma+w_1)_+ + (1-\gamma+w_2)_+] = 2 \quad (3.25)$$

However, the point $\bar{w} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, $\bar{\gamma} = 1$ is also optimal, because it gives the same minimum value of 2. The $-45°$ direction in the $w-$space associated with this solution is useful in the multisurface method of pattern separation[44, 9] since it can be used to generate the first part of a piecewise-linear separator. In practice the linear program package returned the optimal point $\bar{w} = \begin{pmatrix} 2 \\ -2 \end{pmatrix}$, $\bar{\gamma} = -1$, which generates another $45°$ direction that can also be used for piecewise-linear separation.

We turn now to the case when $\delta_1 m^1 \neq \delta_2 m^2$. In particular, consider Smith's case of $\delta_1 = \delta_2 = \frac{1}{m^1+m^2}$. A similar analysis to that of Theorem 3.8 does **not** give guaranteed nonuniqueness of the solution $(\bar{w} = 0, \bar{\gamma}, \bar{y}, \bar{z})$ in $\bar{w} = 0$ for the linear program (3.14) with $\delta_1 = \delta_1 = \frac{1}{m^1+m^2}$. In fact, to the contrary, the analysis shows that indeed $\bar{w} = 0$ is unique under certain conditions which are satisfied by the following counterexample from Mangasarian *et al* [48] to Smith's claim[73] that his linear program (3.14) with $\delta_1 = \delta_2 = \frac{1}{m^1+m^2}$ always generates a nonzero $\bar{w}$. In reality $\bar{w} = 0$ is unique for this example.

Example 3.10 ***(Unique $\bar{w} = 0$ for Smith's LP (2.10b), $\delta_1 = \delta_2 = \frac{1}{m^1+m^2}$)***

$$A^1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad A^2 = \begin{bmatrix} -1 \\ 0 \\ 4 \end{bmatrix}, \quad m = 2, \quad k = 3.$$

For this problem the equivalent norm-minimization problem (3.1) to Smith's LP

(3.14) with $\delta_1 = \delta_2 = \dfrac{1}{m^1 + m^2}$ is:

$$\min_{w,\gamma} f(w,\alpha) := \min_{w,\gamma} \quad \tfrac{1}{5}[(-w+\gamma+1)_+ + (-2w+\gamma+1)_+ + (-w-\gamma+1)_+ + (-\gamma+1)_+ + (4w-\gamma+1)_+] = \tfrac{4}{5} \tag{3.26}$$

and is achieved at $\bar{w} = 0$, $\bar{\gamma} = 1$. The uniqueness of this solution can be established by considering the subdifferential (see Section 5.1.4, p.127 of Polyak[62] and Equation 14.1.4, p.363 of Fletcher[25]) of the function $f(w,\gamma)$ at (0,1) which is given by

$$\partial f(0,1) = \frac{1}{5}\begin{bmatrix} -1-2-\ell_1+0\cdot\ell_2+4\ell_3 \\ 1+1-\ell_1-\ell_2-\ell_3 \end{bmatrix} = \frac{1}{5}\begin{bmatrix} -3-\ell_1+4\ell_3 \\ 2-\ell_1-\ell_2-\ell_3 \end{bmatrix}$$

with $0 \le \ell \le e$. Since $0 \in \partial f(0,1)$ with

$$1 \ge \ell_3 \ge 0.8, \quad \ell_2 = 5(1-\ell_3), \quad \ell_1 = 4\ell_3 - 3,$$

it follows that $0 \in$ *interior* $(\partial f(0,1))$ and hence by Lemma 3, p.137 of Polyak[62], (0,1) is a unique solution of (3.26). (The uniqueness of (0,1) can also be shown by considering the linear program (3.14).) A similar analysis of the unique null can be performed for Grinold's linear program (3.27).

Remark 3.11 *Grinold[32] proposed the following linear program*

$$\min_{w,\gamma,\rho} \left\{ -\rho \;\middle|\; \begin{matrix} A^1 w - e\gamma - e\rho \ge 0, \; -A^2 w + e\gamma - e\rho \ge 0, \\ (eA^1 - eA^2)w + (m^2 - m^1)\gamma = m^2 + m^1 \end{matrix} \right\} \tag{3.27}$$

In Mangasarian et al [48] the example $A^1 = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$, $A^2 = \begin{bmatrix} \frac{1}{2} \\ 2 \\ 4 \end{bmatrix}$ *was given to show that the solution* $(\bar{w} = 0, \; \bar{\gamma} = 5, \; \bar{\rho} = -5)$ *is unique in* $\bar{w}$. *In fact, it can be shown that* $(\bar{w} = 0, \bar{\gamma}, \bar{\rho})$ *is always a solution of (3.27) whenever there exists* u *such that*

$$uA^1 + \frac{eA^1 - eA^2}{m^2 - m^1} = 0, \quad eu = 1, \quad u \ge 0, \quad m^2 > m^1 \tag{3.28}$$

Furthermore, $\bar{w} = 0$ *is unique if for each* h *in* R^n

$$\Big(A^{1^T} + \frac{eA^1 - eA^2}{m^2 - m^1} e\Big)u = h, \quad \textit{has a solution} \quad u \ge 0 \tag{3.29}$$

3.3 Computational comparisons

In this section we give computational comparisons on two real-world databases: the Wisconsin Breast Cancer Database[83, 82] and the Cleveland Heart Disease Database[18], using our proposed linear program (3.15), Smith's LP (3.14) with $\delta_1 = \delta_2 = \frac{1}{m^1 + m^2}$, and the MSM linear programming formulation[48], Problem (2.2). Note that (2.2) can be solved by solving $2n$ linear programs. The corresponding linear separation obtained from (2.2) is

$$\bar{w}x = \frac{\bar{\alpha} + \bar{\beta}}{2} \tag{3.30}$$

where $(\bar{w}, \bar{\alpha}, \bar{\beta})$ is a solution of (2.2).

Figure 3.2 summarizes the results obtained for two linearly inseparable databases: Wisconsin Breast Cancer, and Cleveland Heart Disease. See Appendix A for a description of the datasets. Our testing methodology consisted of dividing each set randomly into a training set consisting of 67% of the data and a testing set consisting of the remaining 33%, and averaging the results over 10 trials. No "secondary" optimization using (3.10) was performed for any method. For each database our linear program (3.15) (referred to as MSM1, multisurface method 1-norm) outperformed both Smith's linear program (3.14) with $\delta_1 = \delta_2$ and the MSM linear programming method (2.2). The average run times for MSM1 and Smith are very close: 3.84 and 3.89 seconds respectively on a DEC station 5000/125 for the Wisconsin Breast Cancer Database, while the corresponding time for MSM was 53.54 seconds. For the Cleveland Heart Disease Database the corresponding times are: 2.82, 2.89, and 53.82 seconds respectively. Note that the percent error of MSM1 on the testing set was better than that of Smith and considerably better than that of MSM on both databases.

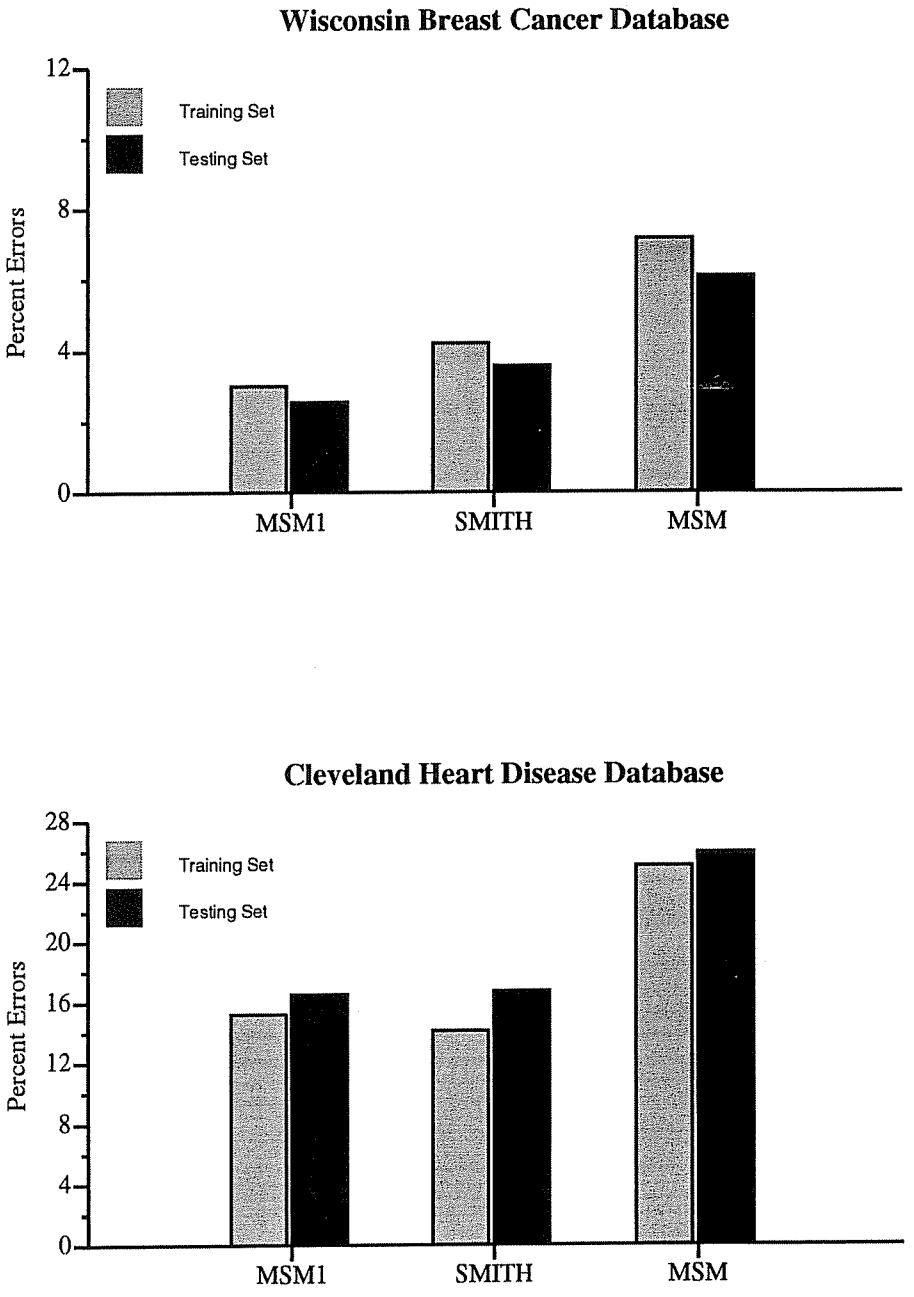


Figure 3.2: Comparison of three linear programming discriminators for linearly inseparable sets

3.4 Strengths of robust linear program

We have presented a robust linear program which always generates a linear surface as an "optimal" separator for two linearly inseparable sets. The "optimality" of the separator consists of minimizing the sum of the average misclassification in each class. By using an appropriately weighted sum, we have overcome the problem of the null solution, which has plagued previous linear programming approaches. These approaches either left this difficulty unresolved or imposed an extraneous linear constraint that never resolved the problem completely. This new 1-norm error minimization problem is considerably better with respect to testing accuracy, training accuracy, and training time than the ∞-norm formulation (2.2) discussed in the previous chapter. The fact that computational results on real-world problems give an edge to our linear program (3.15) over other approaches makes it, in our opinion, the preferred linear program for the linearly inseparable case.

3.5 Extensions

Linear program (3.15) is well suited for use in greedy decision-tree algorithms. As a subproblem in a decision-tree algorithm, the linear program can be used to generate multisurfaces. Fortunately, the new LP is an order of magnitude faster than the linear programming approach discussed in the previous chapter. Fast execution of subproblems is very desirable in decision trees since many subproblems may need to be solved for a single decision tree. In the next chapter, we propose a decision-tree algorithm that uses linear-program (3.15).

Chapter 4

The MSM-Tree algorithm

We propose a decision-tree algorithm based on linear programming. The linear program (3.1) introduced in Chapter 2 is applied recursively to create a multisurface to discriminate between two sets. Unlike popular decision tree algorithms such as CART [14] and ID3 [63], the proposed algorithm relies on linear-combination splits. Computational results show that the proposed algorithm produces smaller trees in less time that are as good in quality as those produced by the CART and C4.5 [65] decision tree algorithms.

4.1 Linear combination splits

Typically, tree-structured classification algorithms such as CART [14] and ID3 [63] use univariate splits, i.e. splits based on a single variable. While univariate trees are easy to interpret logically, complex trees may be required to express multivariate relations. Linear-combination (LC) splits allow multivariate splits to be expressed more succinctly, potentially resulting in simpler trees with fewer nodes. A recent study of multivariate decision trees found that allowing multivariate or linear-combination splits improved accuracy of the resulting decision trees over decision trees with only univariate splits. The perceptron trees [76] and neural tree networks [71] utilize LC splits. CART also also offers an optional LC-splitting

capability. The potential difficulties with these splitting algorithms are discussed in Section 4.2. Finding the best LC split can be posed as a linear program (LP) that minimizes a weighted sum of the misclassification errors. The LP can be solved efficiently using fast algorithms that avoid local minima.

This chapter is organized as follows. Section 4.2 discusses the LP formulation. Comparisons of our LP approach with other LC splitting methods are made. Section 4.3 describes the LP decision tree approach. Section 4.4 contains results of experiments comparing the LP approach with the CART and C4.5 [63, 65] decision-tree approaches. This work was previously reported in [6].

4.2 LP versus other multivariate methods

The optimal LC split consists of a separating plane that minimizes some measure of misclassification error. We propose using linear program (3.1) in Chapter 3 to find such a discriminant. Other decision-tree algorithms have used variants of back propagation [70], variants of the perceptron algorithm, heuristic searches, and simulated annealing. CART uses a heuristic search algorithm that is computationally costly and is prone to local minima [14]. Utgoff [76, 15] explores several different variations of the perceptron algorithm, which address the cycling and convergence problems [55, 29]. Since the perceptron algorithm fails to converge for the linearly *inseparable* case, stopping conditions are more difficult to determine and there is no guarantee that an optimal solution will be found. Sankar and Mamonne's neural tree network [71] uses back propagation [70] modified to use the sum of the absolute value of the errors to train each unit. It suffers from the usual difficulties of back propagation: choice of parameters, local minima, and stopping conditions. Heath proves that the problem of constructing an LC split, which minimizes the total number of misclassifications, is NP-Complete [33]. The SADT algorithm proposed by Heath uses a randomized algorithm (simulated annealing) to attempt to avoid the many local minima. The advantage of the LP approach used with the simplex method [17] is that there are no parameters, no

problems with local minima or convergence, and it has well-defined, easy-to-check stopping conditions.

4.3 MSM tree algorithm

We call the LP-based tree algorithm multisurface method - tree (MSMT) because it is an extension of the multisurface method of pattern recognition [44, 48] to decision trees. For each node in the tree, the best split of the points reaching that node is found by solving LP (3.1) using the simplex method [17]. The node is split into two branches, and the same procedure is applied until there are mostly points of one class at the node or there are too few points at the node. In practice, we split the most impure nodes first, as measured by the information function popularized by ID3, and limit the tree to at most 10 splits. The leaf nodes are assigned the class of the majority of points at that node. We adopted the pessimistic pruning strategy used in C4.5 [64, 65]. This strategy attempts to avoid overfitting of the training data by removing subtrees to reduce future classification errors.

4.4 Computational results

In this section we give computational comparisons on several real-world databases: the Wisconsin Breast Cancer Database [48, 49], the Cleveland Heart Disease Database [18], and the Bank Failure Database [3]. These datasets are described in Appendix A. We use MSMT, CART, and C4.5 (the new and improved ID3). Our original experimental design called for the linear-combination feature of CART. Unfortunately, our commercial CART package crashes after extensive computation time whenever the linear-combination feature is invoked. Thus CART used only univariate splits in conjunction with a cost-complexity pruning procedure. C4.5 used univariate splits with pessimistic pruning. The windowing feature of C4.5 was disabled because windowing did not seem to improve the C4.5 results significantly. Also, windowing could be used with any of the three algorithms if

desired.

Table 4.1 summarizes the results on the three databases. Note that categorical features within the Cleveland Heart data were converted to ordered integers for MSMT but not for C4.5 and CART. The Bank Failure Database exceeded the space limitations for the CART program so there are no results for CART.

Ten-fold cross validation was used to measure generalization. The times reported are the CPU time on a DECStation 5000/125 required to construct and prune one tree averaged over the 10 folds. The CART program performs additional computations and was executed on a different machine. Thus no times are reported for the CART algorithm. The percent training set error and the number of leaf nodes reported are the results from using the entire database one time.

MSMT quickly produced trees with fewer nodes and better generalization than the other two methods. The cross-validation error for MSMT was less than that for C4.5 and CART on all three databases. MSMT produced smaller trees in terms of leaf nodes than did C4.5 and CART. Dramatic reduction in tree size makes the tree easier to interpret and thus compensates for the slightly more complex LC splits. CART also had smaller trees than C4.5 probably because of its better but more expensive pruning algorithm. MSMT and C4.5 were very fast on the Breast Cancer data and the Heart Disease data. C4.5 is slightly faster, especially on the Heart Disease Database which has categorical variables. C4.5 handles categorical variables very efficiently. MSMT, like other LC methods, requires that the attributes be either linearized or encoded as binary attributes in a higher dimensional space. Thus MSMT works best on numerical attributes. On the Bank Failure Database, MSMT was much faster than C4.5, indicating that MSMT works well on larger data sets.

4.5 Strengths of MSMT

We have presented an LP method for constructing two-class decision trees. Unlike previous linear-combination or multivariate splitting methods, the LP approach

Table 4.1: Comparison of MSMT, C4.5, and CART on Three Databases

WISCONSIN BREAST CANCER

Method	Train Error	CV Error	Leaf Nodes	Time (secs)
MSMT	2.4%	3.0%	2	6.8
C4.5	2.8%	3.8%	11	3.7
CART	5.3%	5.3%	3	-

CLEVELAND HEART DISEASE

Method	Train Error	CV Error	Leaf Nodes	Time (secs)
MSMT	15.5%	18.2%	2	9.2
C4.5	9.4%	25.9%	28	1.0
CART	16.8%	20.5%	6	-

BANK FAILURE

Method	Train Error	CV Error	Leaf Nodes	Time (secs)
MSMT	6.4%	6.5%	3	156.3
C4.5	5.0%	7.2%	67	261.0

Train Error := % error on entire data set

CV Error := % cross-validation error (10-fold)

has no problems with local minima, choice of parameters, and convergence criteria. The MSMT algorithm compares favorably with classical decision tree methods in terms of accuracy, training time, and size of trees. The LP described is limited to two-class problems. We have demonstrated that LP-based decision tree algorithms compare very favorably with other approaches and warrant further investigation and application.

4.6 Extensions

To make MSMT a fully functional decision tree method, some enhancements are needed. The first is variable elimination. Ideally, we would like for each node of the tree to use as few variables as possible. Thus we would like to modify our algorithm to eliminate unused variables. Methods must be developed which exploit the structure of the problem to indicate which variables are nonessential. One method would solve a secondary optimization problem after creating an initial split at the node which searches for an alternative optimal or nearly optimal solution with fewer variables. This a difficult problem since the underlying problem of which variables to keep or throw out is combinatorial. Also, methods need to be developed to handle missing data. Many datasets do not have all the attributes completed for all types of data. It is not clear how to handle this type of data with linear programming. A related question arises in how best to map non-numeric data such as color = {red, blue, green} so that it can be best used in the linear program. A fourth issue is what type of pruning should be used when linear programming is employed to create the nodes of the decision tree. The last enhancement is to make the method handle problems with many classes. In the next chapter we show how this linear programming approach can be generalized to $k \geq 2$ multicategory discriminant problems.

Chapter 5

Multicategory separation via linear programming

In this chapter we extend the linear-programming approach proposed in Chapter 2 for two classes to k classes. A single linear program is proposed for discriminating between the elements of k disjoint point sets in R^n. When the conical hulls of the k sets are $(k-1)$-point disjoint in R^{n+1}, a k-piece piecewise-linear surface generated by the linear program completely separates the k sets. This improves on a previous linear programming approach that required that each set be linearly separable from the remaining $k-1$ sets. When the conical hulls of the k sets are not $(k-1)$-point disjoint, the proposed linear program generates an error-minimizing piecewise-linear separator for the k sets. For this case it is shown that the null solution is never a unique solver of the linear program and occurs only under the rather rare condition when the mean of each point set equals the mean of the means of the other $k-1$ sets. This makes the proposed linear computational programming formulation useful for approximately discriminating between k sets that are not piecewise-linear separable.

5.1 Introduction

We consider the k disjoint sets $\mathcal{A}^i$, $i = 1, \ldots, k$, in the n-dimensional real space R^n represented by the $m^i \times n$ matrices, A^i, $i = 1, \ldots, k$. Our objective here is to discriminate between these sets by a piecewise-linear convex function which is the maximum of k linear (affine) functions. The linear pieces of one such typical piecewise-linear surface projected on R^2 are depicted in Figure 5.1 together with the four sets in R^2 that are separated from each other. Many authors have considered this problem. Nilsson [60], Duda-Fossum [19], Duda-Hart [20], and Fukunaga [28], considered iterative methods which are extensions of the perceptron algorithm or the Motzkin-Schoenberg algorithm [56] for determining a piecewise-linear separator when it exists. Convergence of these methods is not known if such a piecewise-linear surface does not exist [28, page 374]. Smith [74] on the other hand considered solving k systems of linear inequalities by solving k linear programs to obtain a piecewise-linear separator. Unfortunately, this may not be possible for many simple piecewise-linear separable problems as we shall demonstrate below. By contrast our linear programming approach works for all piecewise-linear separable sets, and for those that are not some approximate separation will be achieved.

We note that piecewise-linear separation of k disjoint sets in R^n (see Definition 5.2.1) is a natural extension of the classical separation of two disjoint point sets $\mathcal{A}^1$ and $\mathcal{A}^2$ in R^n with nonintersecting convex hulls by using the piecewise-linear maximum of two linear functions. This is equivalent to separation by a single plane [35, 43, 10]. See Figure 5.2.

One of the first questions to resolve is: When is it indeed possible to discriminate between k sets by a piecewise-linear separator which is the maximum of k linear functions? Condition (5.6) of Theorem 5.3 gives a necessary and sufficient condition for such piecewise-linear separability of k sets. Geometrically, this condition can be interpreted as follows. For each $i = 1, \ldots, k$, choose $k - 1$ points $\tilde{A}^i_j, j = 1, \ldots, k,\ j \neq i$, in the conical hull of $\bar{\mathcal{A}}^i \subset R^{n+1}$, where $\bar{\mathcal{A}}^i$ is the set of m^i

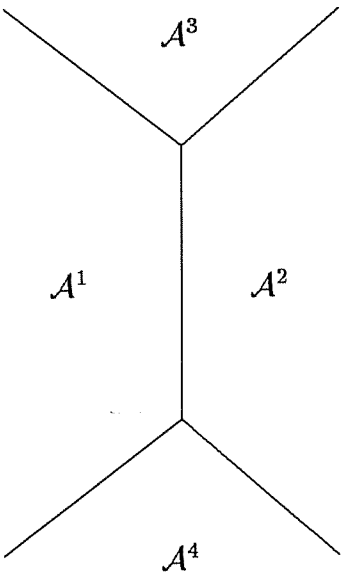


Figure 5.1: Projection of the linear pieces of a piecewise-linear surface on R^2 and the sets $\mathcal{A}^i$, $i = 1, \ldots, 4$, that it separates

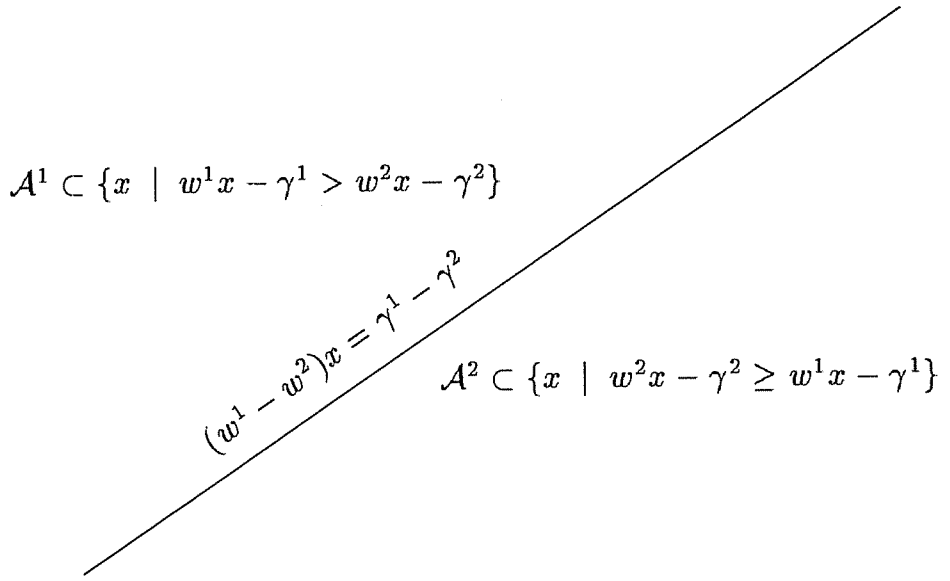


Figure 5.2: Separation of two sets $\mathcal{A}^1$ and $\mathcal{A}^2$ by a plane, or equivalently by the maximum of two linear functions

points in R^{n+1} made up of the rows of $\bar{A}^i := [A^i \;\; e]$, where e is an $m^i \times 1$ column of ones. Condition (5.6) says that there are no points other than $0 \in R^{n+1}$ in these conical hulls of $\bar{\mathcal{A}}^i$, $i = 1, \ldots, k$, satisfying

$$\sum_{\substack{j=1 \\ j \neq i}}^{k} \tilde{A}^i_j = \sum_{\substack{j=1 \\ j \neq i}}^{k} \tilde{A}^j_i, \quad i = 1, \ldots, k \tag{5.1}$$

We refer to this condition as the conical hulls in R^{n+1} of the k sets $\mathcal{A}^i$, $i = 1, \ldots, k$, being $(k-1)$-**point disjoint**. This is a considerably more relaxed requirement than that of Smith [74]. Smith proposed solving k systems of linear inequalities that are equivalent to the linear separation of each of the k sets from the remaining $k-1$ sets. This is far too restrictive an assumption and does not in general hold for simple piecewise-linear separable sets. Figure 5.3 depicts three sets separable by a 3-piece piecewise-linear function, but for which no one set is linearly separable from the remaining two. An even simpler example in R^1 is $\mathcal{A}^1 = \{-1\}$, $\mathcal{A}^2 = \{0\}$ and $\mathcal{A}^3 = \{1\}$. These three sets are piecewise-linear separable by $\max\left\{-x - \frac{1}{2}, \frac{1}{4}, x - \frac{1}{2}\right\}$ but $\mathcal{A}^2$ is not linearly separable from $\mathcal{A}^1 \cup \mathcal{A}^3$.

In Section 5.2 we begin with Definition 5.2.1 of piecewise-linear separability of k sets. Then, as indicated above, we characterize this separability in Theorem 5.3. In Remark 5.3.1 we note that piecewise-linear separation implies pairwise-linear separation, but not conversely. A computationally constructive characterization of piecewise-linear separability of k sets is given in Theorem 5.4 by obtaining a zero minimum for the linear program (5.9) and a corresponding piecewise-linear separation (5.3). Since such piecewise-linear separation (5.3) for the k sets $\mathcal{A}^i$, $i = 1, \ldots, k$, is determined by the quantities $(w^i - w^j, \gamma^i - \gamma^j)$, $i, j = 1, \ldots, k$, $j \neq i$, it is important to ensure the nonzeroness of $(w^i - w^j)$, $i, j = 1, \ldots, k$, $j \neq i$. This is done precisely in Theorem 5.5 where it is shown that the null solution $w^i - w^j = 0, i, j = 1, \ldots, k$, $j \neq i$, occurs under the rather rare condition that the mean of each point set equals the mean of the means of the other $k-1$ sets. Theorem 5.6, however, shows that the null solution, even in this case, is never unique. Section 5.7 contains some computational results employing the proposed

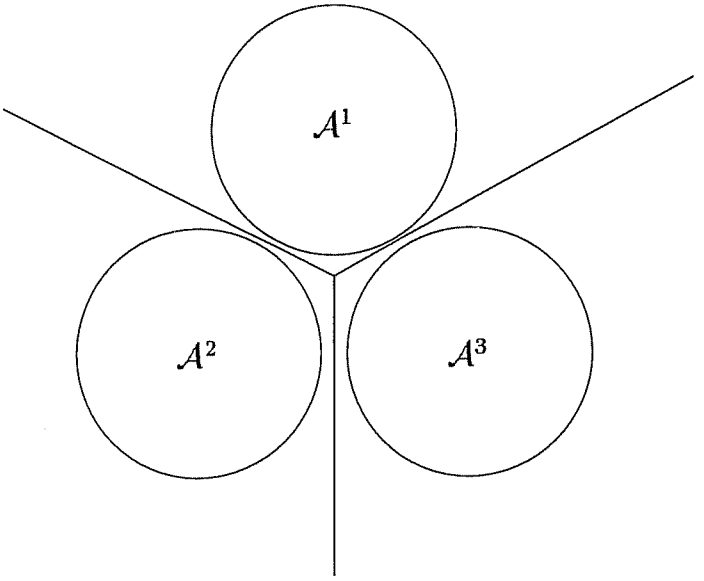


Figure 5.3: Three sets separable by a piecewise-linear function, but no one of which is linearly separable from the other two

linear programming formulation (5.9). Note this work will be published soon[8].

5.2 Multicategory separation by a piecewise-linear surface

We begin by defining the concept of piecewise-linear separation of k sets in R^n [60].

Definition 5.2.1 (Piecewise-linear Separability) *The k sets $\mathcal{A}^i$, $i = 1, \ldots, k$, each consisting of m^i, $i = 1, \ldots, k$, points in R^n and represented by the $m^i \times n$ matrices, A^i, $i = 1, \ldots, k$, are piecewise-linear separable if there exist $w^i \in R^n$, $\gamma^i \in R$, $i = 1, \ldots, k$ such that*

$$A^i w^i - e\gamma^i > A^i w^j - e\gamma^j, \; i, \; j = 1, \ldots, k, \; i \neq j \tag{5.2}$$

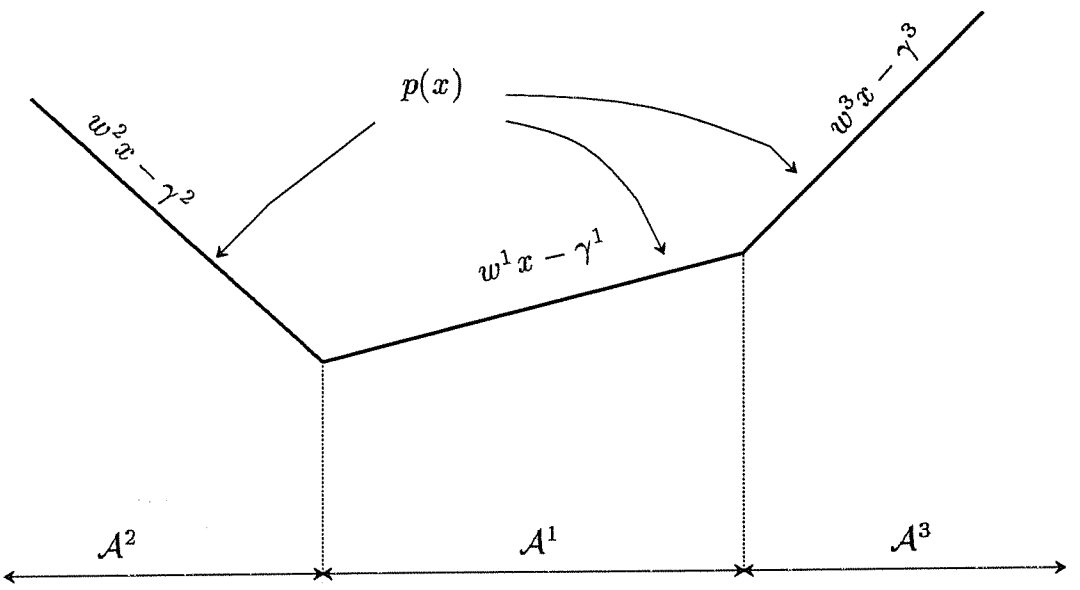


Figure 5.4: Piecewise-linear separation of sets $\mathcal{A}^1$, $\mathcal{A}^2$ and $\mathcal{A}^3$ by the convex piecewise-linear function $p(x) = \max_{i=1,2,3} w^i x - \gamma^i$

or equivalently

$$A^i w^i - e\gamma^i \geq A^i w^j - e\gamma^j + e,\ i, j = 1, \ldots, k,\ i \neq j \tag{5.3}$$

Remark 5.2.1 *The piecewise-linear separability can be interpreted geometrically by the existence of a piecewise-linear convex function determined from* (w^i, γ^i), $i = 1, \ldots, k$, *by*

$$p(x) = \max_{1 \leq \ell \leq k} xw^\ell - \gamma^\ell \tag{5.4}$$

and such that

$$\left\langle \begin{array}{l} p(x) = xw^i - \gamma^i \\ p(x) > xw^j - \gamma^j \end{array} \right\rangle \begin{array}{l} for\ \ x \in \mathcal{A}^i,\ i = 1, \ldots, k \\ for\ \ j \neq i \end{array} \tag{5.5}$$

Figure 5.4 depicts a simple piecewise-linear $p(x)$ on R that separates three sets.

Our first objective is to characterize piecewise-linear separability. This is done in the following theorem.

Theorem 5.3 (Characterization of Piecewise-Linear Separability) *Let $\mathcal{A}^i$, $i = 1, \ldots, k$ be nonempty point sets in R^n. The following are equivalent:*

(a) $\mathcal{A}^i$, $i = 1, \ldots, k$ are piecewise-linear separable; that is, there exist $w^i \in R^n$, $\gamma^i \in R$, $i = 1, \ldots k$, satisfying (5.2) or (5.3).

(b) The conical hulls $K(\bar{\mathcal{A}}^i)$ of the k sets $\bar{\mathcal{A}}^i$, where $\bar{A}^i := [A^i \;\; e]$, $i = 1, \ldots, k$, are $(k-1)$-point disjoint in R^{n+1}, that is

$$\sum_{\substack{j=1 \\ j \neq i}}^{k} u^{ij}[A^i \;\; e] = \sum_{\substack{j=1 \\ j \neq i}}^{k} u^{ji}[A^j \;\; e],\; u^{ij} \geq 0,\; i, j = 1, \ldots, k,\; j \neq i \implies u^{ij} = 0,\; i, j = 1, \ldots, k,\; j \neq i \tag{5.6}$$

Proof. Throughout the following arguments, i, $j = 1, \ldots, k$ and $j \neq i$.

$$\begin{aligned}
(a) \;\Leftrightarrow\; & A^i(w^i - w^j) - e(\gamma^i - \gamma^j) > 0, \qquad (5.7)\\
& \text{have solution } (w^i,\; w^j,\; \gamma^i\; \gamma^j) \in R^n \times R^n \times R \times R\\
\Leftrightarrow\; & A^i(w^i - w^j) - e(\gamma^i - \gamma^j) - e\zeta \geq 0,\; \zeta > 0,\\
& \text{have solution } (w^i,\; w^j,\; \gamma^i, \gamma^j,\; \zeta) \in R^n \times R^n \times R \times R \times R\\
\Leftrightarrow\; & -\sum_{\substack{j=1 \\ j \neq i}}^{k} u^{ji}A^j + \sum_{\substack{j=1 \\ j \neq i}}^{k} u^{ij}A^i = 0,\; \sum_{\substack{j=1 \\ j \neq i}}^{k} u^{ji}e - \sum_{\substack{j=1 \\ j \neq i}}^{k} u^{ij}e = 0\\
& \sum_{i=1}^{k} \sum_{\substack{j=1 \\ j \neq i}}^{k} u^{ij}e > 0, \text{ have no solution } u^{ij} \geq 0,\; u^{ij} \in R^{m^i}\\
& \qquad\qquad \text{(By Motzkin's Theorem of the Alternative[45])}\\
\Leftrightarrow\; & (b).
\end{aligned}$$

□

Remark 5.3.1 *It is evident from Definition 5.2.1 that piecewise-linear separability of the sets $\mathcal{A}^i$, $i = 1, \ldots, k$ implies pairwise linear separability of the same*

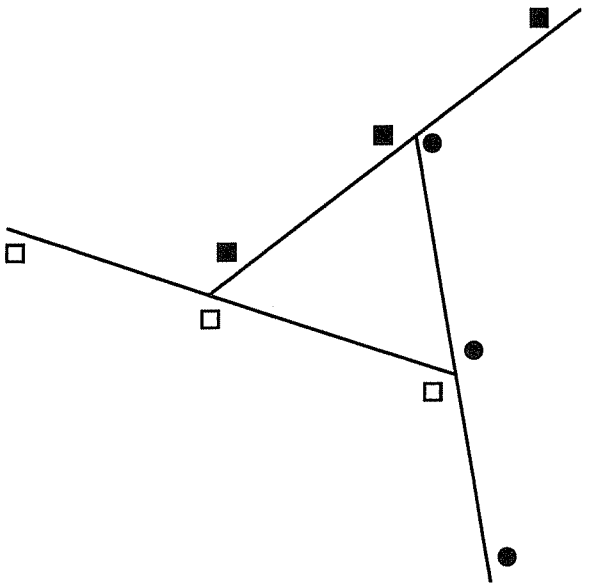


Figure 5.5: The whirlwind counterexample: Three sets that are pairwise linearly separable, but not piecewise-linear separable

sets. However the converse is not true as evidenced by the "whirlwind" counterexample depicted in Figure 5.5 for which three sets are pairwise linearly separable, but are not piecewise-linear separable. The latter fact, which may not be immediately evident from the figure, can be computationally verified by showing that the implication of (5.6) does not hold by solving the dual linear program (5.12) and showing that it has a positive maximum.

We can now specify a linear program that will generate a piecewise-linear separation between the sets $\mathcal{A}^i$, $i = 1, \ldots, k$, if one exists, otherwise it will generate an error-minimizing separation. The linear program will generate $(w^i,\ \gamma^i) \in R^n \times R,\ i = 1, \ldots, k$, that will satisfy (5.3) by minimizing the 1-norm of the averaged violations of (5.3), that is

$$\min_{w^i,\gamma^i} \sum_{i=1}^{k} \sum_{\substack{j=1 \\ j\neq i}}^{k} \frac{e}{m^i} \left(-A^i(w^i - w^j) + e(\gamma^i - \gamma^j) + e\right)_+ \tag{5.8}$$

This minimization problem can be written as the following linear program (LP):

$$\min_{w^i,\gamma^i,y^{ij}} \left\{ \sum_{i=1}^{k} \sum_{\substack{j=1 \\ j\neq i}}^{k} \frac{ey^{ij}}{m^i} \;\middle|\; \begin{array}{c} y^{ij} \geq -A^i(w^i - w^j) + e(\gamma^i - \gamma^j) + e,\ y^{ij} \geq 0, \\ i \neq j,\ i,\ j = 1,\ldots,k \end{array} \right\} \tag{5.9}$$

The purpose of the weights $\frac{1}{m^i}$, which are analogous to the weights in the objective [10] for the two-category case, is to avoid the null $w^i - w^j = 0$ solution. See Theorem 5.5 below. Note that for any solution $(w^i,\ \gamma^j,\ y^{ij})$, of the LP (5.9), we have

$$y^{ij} = (-A^i(w^i - w^j) + e(\gamma^i - \gamma^j) + e)_+,\ i \neq j,\ i,\ j = 1,\ldots,k \tag{5.10}$$

Since the inequalities (5.3) of the piecewise-linear separator are satisfied if and only if the minimum of (5.8) or equivalently of (5.9) is zero, we have the following result.

Theorem 5.4 (Multicategory Piecewise-linear Separation via LP)
The sets $\mathcal{A}^i$, $i = 1,\ldots,k$, represented by the $m^i \times n$ matrices A^i, $i = 1,\ldots,k$, are piecewise-linear separable if and only if the solvable linear program (5.9) has a zero minimum, in which case any solution $(w^i,\ \gamma^i,\ y^{ij})$, $i,\ j = 1,\ldots,k,\ j \neq k$, provides a piecewise-linear separation as defined by (5.3).

As was the case for linear separation of two sets by linear programming [9, 10], it is important here also to rule out the null solution in case the sets $\mathcal{A}^i$, $i = 1,\ldots,k$ are not piecewise-linear separable. Since the piecewise-linear separation (5.3) is in effect achieved by a special pairwise linear separation between the sets $\mathcal{A}^i$, $i = 1,\ldots,k$, which is determined by $(w^i - w^j,\ \gamma^i - \gamma^i) \in R^n \times R^1$, $i \neq j$, $i,j = 1,\ldots,k$, it is the nonzeroness of $w^i - w^j$, $i \neq j$, i, $j = 1,\ldots,k$ that matters. In [10] it was shown for the two-category case that $w^1 - w^2$ can be zero if and only if the $\mathcal{A}^1$ and $\mathcal{A}^2$ have equal means, in which case the null solution $w^1 - w^2 = 0$ is not a unique solution of the linear program. We shall derive generalizations of these two results to the multicategory case. Nonzeroness of $w^i - w^j$, $i \neq j$, i, $j = 1,\ldots,k$,

is an important issue when one is trying to generate an approximate piecewise-linear separation (that is, allow some errors in the separation) for sets which are not piecewise-linear separable. Zero $w^i - w^j,\ i \neq j,\ i, j = 1, \ldots, k$ will yield no information and no approximate separation for this case.

We give now a result that provides a necessary and sufficient condition for the occurrence of null $w^i - w^j,\ i \neq j,\ i, j = 1, \ldots, k$.

Theorem 5.5 (Null Solution Occurrence) *The linear program (5.9) has the null solution, $w^i - w^j = 0,\ i \neq j,\ i, j = 1, \ldots, k$ if and only if*

$$\frac{eA^i}{m^i} = \frac{1}{k-1} \sum_{\substack{j=1 \\ j \neq i}}^{k} \frac{eA^j}{m^j},\ i = 1, \ldots, k \tag{5.11}$$

Proof. The dual of the linear program (5.9) is

$$\max_{u^{ij}} \left\{ \sum_{i=1}^{k} \sum_{\substack{j=1 \\ j \neq i}}^{k} eu^{ij} \;\middle|\; \begin{array}{l} \displaystyle\sum_{\substack{j=1 \\ j \neq i}}^{k} (u^{ij}A^i - u^{ji}A^j) = 0,\ \ i = 1, \ldots, k \\ \displaystyle\sum_{\substack{j=1 \\ j \neq i}}^{k} (-eu^{ij} + eu^{ji}) = 0,\ \ i = 1, \ldots, k \\ 0 \leq u^{ij} \leq \dfrac{e}{m^i}\ \ i \neq j,\ i = 1, \ldots, k \end{array} \right\} \tag{5.12}$$

The vectors $w^i - w^j = 0,\ i \neq j,\ i, j = 1, \ldots, k$ constitute an optimal solution for the primal LP (5.9) if and only if the equivalent minimization problem (5.8) is solved by setting $w^i - w^j = 0$ and $\gamma^i - \gamma^j = 0,\ i \neq j,\ i, j = 1, \ldots, k$, which gives $y^{ij} = e,\ i \neq j,\ i, j = 1, \ldots, k$ and a primal minimum value of $k(k-1)$. Since $0 \leq u^{ij} \leq \dfrac{e}{m^i},\ i \neq j,\ i = 1, \ldots, k$, and the dual optimal objective must equal the primal optimal objective of $k(k-1)$ we have

$$\sum_{i=1}^{k} \sum_{\substack{j=1 \\ j \neq i}}^{k} eu^{ij} = k(k-1)$$

It follows that

$$u^{ij} = \frac{e}{m^i},\ j \neq i,\ i = 1, \ldots, k$$

and

$$\frac{(k-1)}{m^i}\, eA^i = \sum_{\substack{j=1 \\ j\neq i}}^{k} \frac{eA^j}{m^j}, \quad i = 1, \ldots, k,$$

which is (5.11). □

We now show even in the unlikely event that (5.11) is satisfied, the null solution $w^i - w^j = 0,\ i \neq j,\ i, j = 1, \ldots, k$, is not unique, and hence some nonzero solution will also be optimal.

Theorem 5.6 (Nonuniqueness of the Null Solution) *If condition (5.11) is satisfied, the null solution $w^i - w^j = 0,\ i \neq j,\ i, j = 1, \ldots, k$, to the linear program (5.9) is not unique.*

Proof. Let the primal solution to (5.9) be such that

$$\bar{w}^i - \bar{w}^j = 0,\ \bar{\gamma}^i - \bar{\gamma}^j = 0,\ \bar{y}^{ij} = e,\ i \neq j,\ i = 1, \ldots, k$$

Hence only the constraints $y^{ij} \geq 0$ of the linear program are inactive. It follows that this solution is unique in $\bar{w}^i - \bar{w}^j$ if and only if the following **has no solution for all** $h^{ij} \in R^n$, in the variables $(y^{ij} - \bar{y}^{ij},\ w^i - \bar{w}^i,\ \gamma^i - \bar{\gamma}^i),\ i \neq j,\ i, j = 1, \ldots, k$:

$$\sum_{i=1}^{k} \sum_{\substack{j=1 \\ j\neq i}}^{k} -\frac{e}{m^i}\,(y^{ij} - \bar{y}^{ij}) \geq 0$$

$$(y^{ij} - \bar{y}^{ij}) + A^i((w^i - w^j) - (\bar{w}^i - \bar{w}^j)) - e((\gamma^i - \gamma^j) - (\bar{\gamma}^i - \bar{\gamma}^j)) \geq 0$$

$$\sum_{\substack{i,j=1 \\ i\neq j}}^{k} -h^{ij}((w^i - w^j) - (\bar{w}^i - \bar{w}^j)) > 0$$

By Motzkin's Theorem [45] this is equivalent to the following system **having a solution for all** $h^{ij} \in R^n,\ i \neq j,\ i, j = 1, \ldots, k$:

$$\sum_{\substack{j=1 \\ j\neq i}}^{k} (u^{ij}A^i - u^{ji}A^j) = h^{ij},\ i = 1, \ldots, k$$
$$\sum_{\substack{j=1 \\ j\neq i}}^{k} (-eu^{ij} + eu^{ji}) = 0,\ i = 1, \ldots, k$$
$$-\frac{1}{m^i}\, e\zeta + u^{ij} = 0,\ i \neq j,\ i = 1, \ldots, k$$
$$(\zeta,\ u^{ij}) \geq 0,\ i \neq j,\ i = 1, \ldots, k$$

This is obviously not true because there are $h^{ij},\ i \neq j,\ i, j = 1, ..., k$ in R^n that cannot be written as:

$$h^{ij} = \zeta(k-1)\left(\frac{eA^i}{m^i} - \frac{1}{k-1}\sum_{\substack{j=1 \\ j\neq i}}^{k} \frac{eA^j}{m^j}\right),\ \zeta \geq 0$$

Hence $\bar{w}^i - \bar{w}^j = 0$ is not unique. □

5.7 Computational results

We present now computational results that utilize the linear programming formulation (5.9) for discriminating between k sets for both the piecewise-linear separable and inseparable cases. Three different problems were considered: wine cultivar discrimination [1], iris classification [24], and breast cancer prognosis [84]. All three databases are available via anonymous file transfer protocol (ftp) from the University of California Irvine UCI Repository Of Machine Learning Databases [57]. Table 5.1 gives the number of classes, dimension, and number of points (size) of each database as well as the results of using linear program (5.9) to discriminate between the classes. Cross validation testing was done by using the leave-one-out method [40, 28] to estimate the training set correctness and the testing set correctness (correctness on unseen examples). The average training and testing correctness over all the trials is given in Tables 5.1 and 5.2. The MINOS

Table 5.1: Performance of multicategory linear program on three problems. Correctness estimated using "leave-one-out" cross validation.

Problem	Classes (k)	Dimension (n)	Size	Percent Correct	
				Training	Testing
Wine Cultivars	3	13	178	100.0	91.0
Iris Plants	3	4	150	98.7	96.7
Breast Cancer	3	11	122	66.3	56.6

linear programming package [58] was used to solve LP (5.9). The null solution was never encountered in these experiments. A brief discussion of the numerical results follows.

The Italian wine cultivar database is piecewise-linear separable. A single linear program was able to correctly separate the training set. The testing set performance was comparable to previously published results [1]. Fisher's classical iris discrimination problem is almost piecewise-linear separable and once again the multicategory linear programming approach performed quite well on both the training and testing sets. These problems illustrate that a single multicategory linear program can effectively discriminate sets that are totally (or almost totally) piecewise-linear separable.

The breast cancer prognosis problem is inherently more difficult. The breast cancer prognosis problem was created by dividing the Wisconsin breast cancer database into three classes: cancer which recurred (developed distant metastasis) in less than 3 months, cancer which recurred in between 3 and 24 months, and cancers which did not recur in 24 months. The fact that the sets are not piecewise-linear separable is evidenced by the relatively poor training set accuracy shown in Table 5.1. A single linear program is insufficient for solving such problems. However, the results of several multicategory linear programs can be combined by

Table 5.2: Comparison of MSMT-MC and single multicategory linear program on breast cancer prognosis problem. Correctness estimated using "leave-one-out" cross validation.

	Single LP	MSMT-MC	Change
Training Correctness (%)	66.3	93.0	+40.3%
Testing Correctness (%)	56.6	66.4	+17.3%
Average Number of LP's	1	5.1	+4.1

using multisurface methods [44, 9, 6]. To demonstrate this, we used the multicategory linear program to create the multivariate splits in the multisurface method tree algorithm (MSMT) [6, 84].

MSMT-multicategory works by applying the linear program (5.9) to a k-class classification problem. The resulting piecewise-linear surface divides the space into k regions. If each of these k regions contains mostly points of one class, then we are done. If any region contains an unacceptable mixture of points then the linear program (5.9) is used again to divide that region into k or fewer regions. The resulting discriminant function can be thought of as a decision tree, thus the name **multisurface method tree - multicategory** (MSMT-MC).

The results using MSMT-MC are given in Table 5.2. These results show that using a multisurface approach enables the linear program (5.9) to be used for solving problems that are not piecewise-linear separable. By applying this approach to the breast cancer prognosis problem the training set accuracy improved over 40 percent and the testing set accuracy improved over 17 percent.

5.8 Strengths

An effective algorithm for discriminating between k disjoint point sets in R^n has been proposed and tested. The algorithm is based on the solution of a single linear

program when the k sets are either piecewise-linear separable or approximately so. A characterization of piecewise-linear separability was given. It was also shown that, for piecewise-linear inseparable sets, the null solution of our linear program is never unique and occurs only under unusual conditions. This makes the proposed linear programming approach useful both as an approximate discriminator for piecewise-linear inseparable sets and as an exact discriminator for separable ones. Computational results demonstrate the effectiveness of the approach.

5.9 Enhancements

The obvious next step with the proposed linear programming approach is to incorporate it into a fully functional decision-tree method. The desirable enhancements for decision trees discussed in the last chapter (variable elimination, handing symbolic variables, etc.) are still needed with this formulation. The only undesirable feature of this method is that the problem size grows very large when the numbers of points and classes is large, increasing the computation time. We would like to develop faster parallel algorithms to speed training without decreasing generalization performance. This is the topic of the next chapter.

Chapter 6

Parallel multicategory discrimination

In this chapter, we formulate multicategory piecewise-linear separation as a piecewise-quadratic minimization problem which minimizes the 2-norm of the average errors within each class. Computationally, a serial implementation of this new method is up to an order of magnitude faster than the linear programming formulation in Chapter 5. To take advantage of the structure of the program, we used the parallel gradient distribution method [47] to solve this problem. We implemented the parallel gradient distribution algorithm on the Thinking Machines CM-5 parallel processor and decreased computation time by an average factor of 4.6 over a quasi-Newton algorithm [58].

We give now an outline of the chapter. In Section 6.1 we review the definition of piecewise-linear separability and the 1-norm error formulation, and then give the 2-norm formulation (6.6). We establish a new simple condition (6.9) for the occurrence of the null solution for the 2-norm problem (6.6), which turns out to be equivalent to that for the 1-norm formulation (6.5) [8]. In Section 6.2, we discuss both serial and parallel algorithms for solving the optimization problem and compare this formulation with previous approaches. The pertinent theorem of the parallel gradient distribution method [47] are given. Details of the algorithms and

implementation are given in Section 6.2. Section 6.3 gives a serial computational comparison of the linear-programming 1-norm approach and the new 2-norm approach, as well as the results of the proposed parallel algorithm implemented serially and in parallel on the Thinking Machines CM-5 parallel processor.

We adopt some special notation within this chapter. The sequence $\{x_i\}$, $i = 0, 1, \ldots$, will represent iterates in the h-dimensional real space R^h generated by some algorithm. For $\ell = 1, \ldots, k$, $x_i^\ell \in R^{h^\ell}$ will represent an h^ℓ-dimensional subset of components of x_i, where $\sum_{\ell=1}^{k} h^\ell = h$. The complement of ℓ in $\{1, \ldots, k\}$ will be denoted by $\bar{\ell}$ and we write $x_i = (x_i^\ell,\ x_i^{\bar{\ell}})$, $\ell = 1, \ldots, k$. For a differentiable function $f: R^h \rightarrow R$, ∇f will denote the h-dimensional vector of partial derivatives with respect to x, and $\nabla_\ell f$ will denote the h^ℓ-dimensional vector of partial derivatives with respect to $x^\ell \in R^{h^\ell}$, $\ell = 1, \ldots, k$. For k points y in R^h, $\sum_{\ell=1}^{k} \lambda_j y_j$, such that $\lambda_j > 0$ and $\sum_{j=1}^{k} \lambda_j = 1$, is said to be a strong convex combination of the points y_j, $j = 1, \ldots, k$. If f has continuous first partial derivatives on R^h, we say $f \in C^1(R^h)$.

6.1 Multicategory separation by piecewise-linear surfaces

We begin by defining the concept of piecewise-linear separation of k sets in R^n [60, 8] and formulating the problem of minimizing the 1-norm error as a linear-program.

Definition 6.1.1 (Piecewise-linear Separability) *The k sets $\mathcal{A}^i$, $i = 1, \ldots, k$, each consisting of m^i, $i = 1, \ldots, k$, points in R^n and represented by the $m^i \times n$ matrices, A^i, $i = 1, \ldots, k$, are piecewise-linear separable if there exist $w^i \in R^n$, $\gamma^i \in R$, $i = 1, \ldots, k$ such that*

$$A^i w^i - e^i \gamma^i \geq A^i w^j - e^i \gamma^j + e^i,\ i, j = 1, \ldots, k,\ i \neq j \tag{6.1}$$

Equivalently, there exists a piecewise-linear convex function determined by $(w^i,\ \gamma^i)$, $i = 1, \ldots, k$, *such that*

$$p(x) = \max_{1 \le \ell \le k} xw^\ell - \gamma^\ell, \tag{6.2}$$

and

$$\left\langle \begin{array}{l} p(x) = xw^i - \gamma^i \\ p(x) > xw^j - \gamma^j \end{array} \right\rangle \begin{array}{l} for\ \ x \in \mathcal{A}^i,\ i = 1, \ldots, k \\ for\ \ j \neq i \end{array} \tag{6.3}$$

In [8], it was shown that the inequalities of the piecewise-linear separator are satisfied if and only if the minimum of the 1-norm of the average violations of the inequalities (6.1) is zero, namely

$$0 \ = \ \min_{w^i, \gamma^i} \sum_{i=1}^{k} \sum_{\substack{j=1 \\ j \neq i}}^{k} \frac{e^i}{m^i} \left(-A^i(w^i - w^j) + e^i(\gamma^i - \gamma^j) + e^i\right)_+ \tag{6.4}$$

This minimization problem can be written as the following linear program (LP) :

$$\min_{w^i, \gamma^i, y^{ij}} \left\{ \sum_{i=1}^{k} \sum_{\substack{j=1 \\ j \neq i}}^{k} \frac{e^i y^{ij}}{m^i} \ \middle| \ \begin{array}{c} y^{ij} \geq -A^i(w^i - w^j) + e^i(\gamma^i - \gamma^j) + e^i,\ y^{ij} \geq 0, \\ i \neq j,\ i,\ j = 1, \ldots, k \end{array} \right\} \tag{6.5}$$

Figure 6.1 depicts the piecewise-linear classifier found by the LP (6.5) for a typical piecewise-linear separable case with $k = 4$ and $n = 2$.

As shown in [8], the linear program (6.5) is quite effective on real-world problems. In practice such problems are rarely piecewise-linear separable and thus a multivariate decision tree must be used. A multivariate decision tree works by applying the linear program or another algorithm to a k-class classification problem. The resulting piecewise-linear surface divides the space into k regions. If each of these k regions contains mostly points of one class, then we are done. If any region contains an unacceptable mixture of points then the linear program (6.5) or the other algorithm is used again to divide that region into k or fewer regions. The resulting discriminant function can be thought of as a decision tree. Figure 6.2 illustrates a decision surface found by a multivariate decision tree algorithm

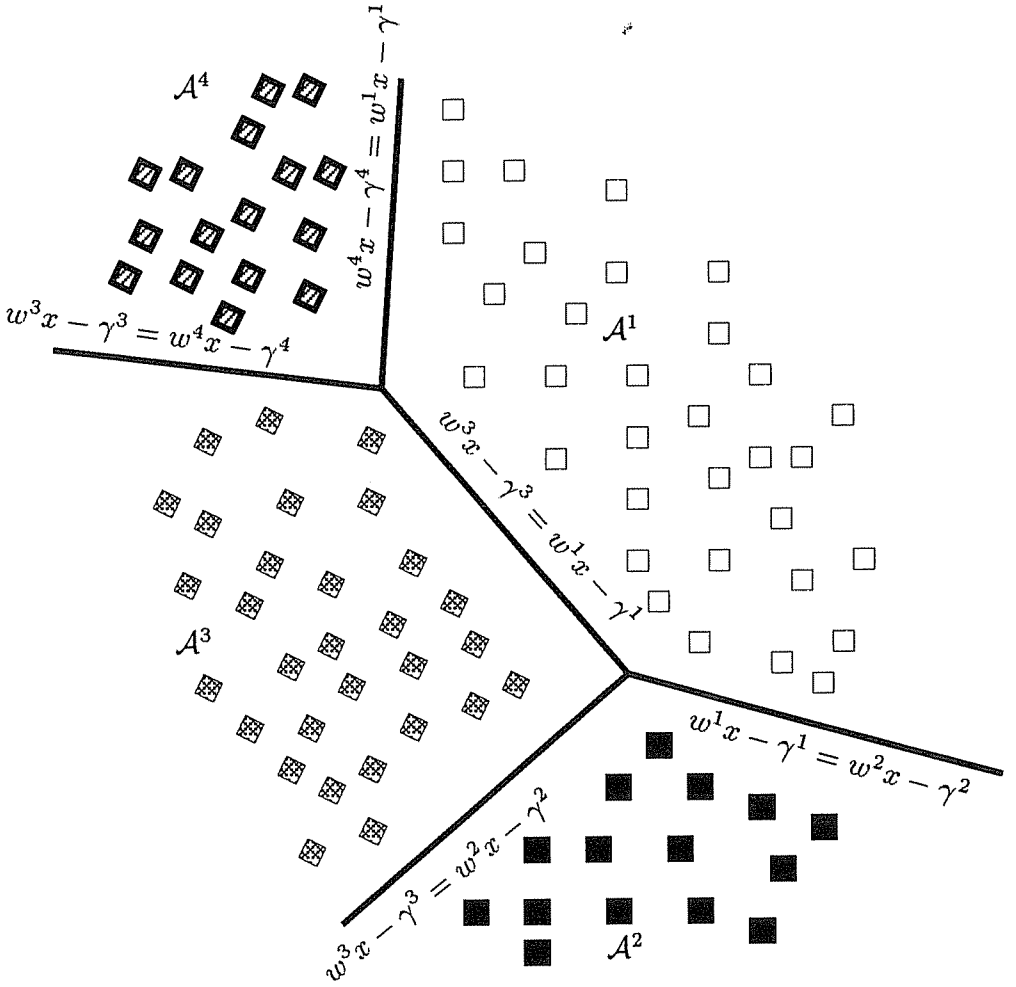


Figure 6.1: Piecewise-linear separator of 4 classes in R^2

and Figure 6.3 depicts the corresponding decision tree. Although the LP (6.5) is effective for use in such an algorithm, it can be very slow because the LP problem size can get quite large. Specifically for a problem with m points in R^n that belong to k classes, there are $m \times (k-1)$ constraints and $m \times (k-1) + k \times (n+1)$ variables (not counting slacks). Since many such LPs may be needed to find a single decision tree, a fast method is desired. Ideally, an iterative method is also desirable in case new points are added and the tree needs to be adjusted [77]. Previous iterative approaches based on extensions to the perceptron algorithm [60, 19, 20] do not have stable performance for the inseparable case and as a result heuristic methods [29, 15] have been developed to get around this deficiency. Ideally we would like to have a fast parallelizable algorithm that can be shown to converge for both separable and the more common inseparable problems. By starting from Definition 6.1.1 and reformulating the problem we can accomplish this.

Consider the 2-norm formulation of minimizing of the average violation:

$$\min_{w^i,\gamma^i} f(w,\gamma) \;=\; \frac{1}{2}\sum_{i=1}^{k} \sum_{\substack{j=1 \\ j\neq i}}^{k} \frac{1}{m^i} \left\| (-A^i(w^i - w^j) + e^i(\gamma^i - \gamma^j) + e^i)_+ \right\|_2^2 \qquad (6.6)$$

Squaring the plus function results in a piecewise-quadratic function that is differentiable. We refer to (6.6) as the piecewise-quadratic minimization (PQM) problem. The first partial derivative of the function (below) can be shown to be Lipschitz-continuous. In particular we have

$$\begin{aligned} \nabla_{w^\ell} f(w,\gamma) \;=\; & \sum_{\substack{j=1 \\ j\neq \ell}}^{k} \frac{-1}{m^\ell} A^{\ell T}(-A^\ell(w^\ell - w^j) + e^\ell(\gamma^\ell - \gamma^j) + e^\ell)_+ + \\ & \sum_{\substack{i=1 \\ i\neq l}}^{k} \frac{1}{m^i} A^{iT}(-A^i(w^i - w^\ell) + e^i(\gamma^i - \gamma^\ell) + e^i)_+ \end{aligned} \qquad (6.7)$$

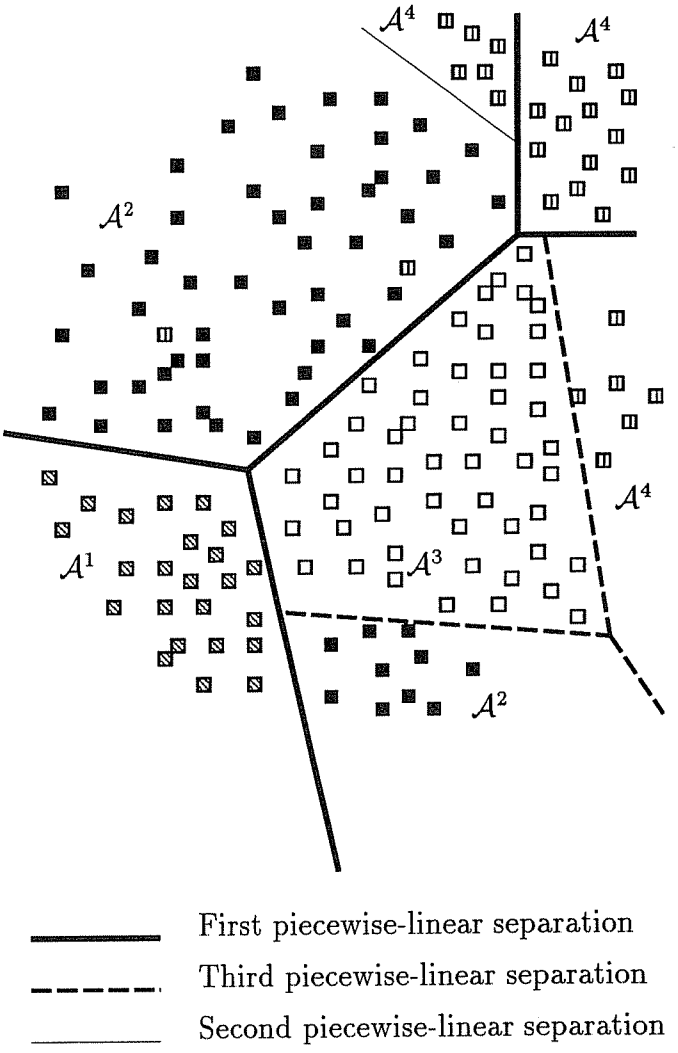


Figure 6.2: Geometric depiction of decision tree consisting of 3 piecewise-linear separators distinguishing 4 classes in R^2

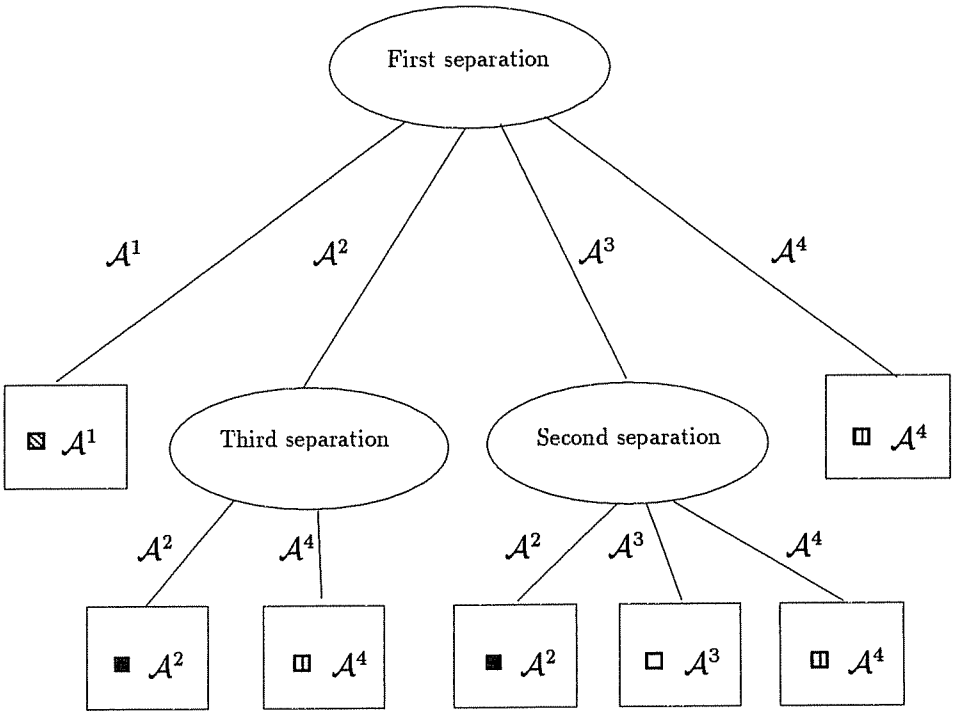


Figure 6.3: Decision tree representing the 3 piecewise-linear separators depicted in Figure 6.2

$$\nabla_{\gamma^\ell} f(w,\gamma) = \sum_{\substack{j=1 \\ j\neq\ell}}^{k} \frac{1}{m^\ell} e^\ell(-A^\ell(w^\ell - w^j) + e^\ell(\gamma^\ell - \gamma^j) + e^\ell)_+ + \sum_{\substack{i=1 \\ i\neq\ell}}^{k} \frac{-1}{m^i} e^i(-A^i(w^i - w^\ell) + e^i(\gamma^i - \gamma^\ell) + e^i)_+ \tag{6.8}$$

As in the 1-norm formulation, the inequalities (6.1) for piecewise-linear separation hold if and only if the minimum of (6.6) is zero. Consequently the following theorem holds.

Theorem 6.1.1 ***(Multicategory Separation via Piecewise Quadratic Minimization (PQM))*** *The sets $\mathcal{A}^i$, $i = 1,\ldots,k$, represented by the $m^i \times n$ matrices A^i, $i = 1,\ldots,k$, are piecewise-linear separable if and only if the solvable piecewise quadratic minimization (6.6) has a zero minimum, in which case any solution $(w^i,\ \gamma^i)$, $i = 1,\ldots,k$, provides a piecewise-linear separation as characterized in Definition 6.1.1.*

As was the case for linear and piecewise-linear separation of two sets by linear programming [9, 10], it is important to determine when the useless null solution occurs for sets $\mathcal{A}^i$, $i = 1,\ldots,k$ that are not piecewise-linear separable. Note that the piecewise-linear separation (6.1) is achieved by a special pairwise linear separation between the sets $\mathcal{A}^i$, $i = 1,\ldots,k$, that is determined by $(w^i - w^j,\ \gamma^i - \gamma^i) \in R^n \times R^1$, $i \neq j$, $i,j = 1,\ldots,k$,. It is therefore the nonzeroness of $w^i - w^j$, $i \neq j$, i, $j = 1,\ldots,k$ that matters. Nonzeroness of $w^i - w^j$, $i \neq j$, i, $j = 1,\ldots,k$, is an important issue when one is trying to generate an approximate piecewise-linear separation (i.e. allow some errors in the separation) for sets that are not piecewise-linear separable. Zero $w^i - w^j$, $i \neq j$, i, $j = 1,\ldots,k$ will yield no information and hence no approximate separation for this case is obtained.

We now give a result that provides a necessary and sufficient condition for the occurrence of the null solution: $w^i - w^j = 0$, $i \neq j$, i, $j = 1,\ldots,k$.

Theorem 6.1.2 (Null Solution Occurrence) *The piecewise quadratic minimization (PQM) (6.5) has the null solution, $w^i - w^j = 0$, $i \neq j$, i, $j = 1,\ldots,k$ if and only if all class means are equal, that is*

$$\frac{e^i A^i}{m^i} = \frac{e^j A^j}{m^j},\ i = 1, \ldots, k,\ j = 1, \ldots, k \tag{6.9}$$

Proof. The vectors $w^i - w^j = 0,\ i \neq j,\ i,\ j = 1, \ldots, k$ constitute an optimal solution of problem (6.6) if and only if $\gamma^i = \gamma^j = 0,\ i \neq j,\ i, j = 1, \ldots, k$. For these values of $w^i, \gamma^i, i = 1, \ldots, k$,

$$\nabla_{w^\ell} f(w, \gamma) \ = \ 0,\ \nabla_{\gamma^\ell} f(w, \gamma) \ = \ 0,\ \ell = 1, \ldots, k. \tag{6.10}$$

Evaluating the partial gradients at such an optimal point gives

$$\begin{aligned} \nabla_{w^\ell} f(w, \gamma) \ &= \ \sum_{\substack{j=1 \\ j \neq \ell}}^{k} -\frac{1}{m^\ell} A^{\ell T} e^\ell \ + \ \sum_{\substack{i=1 \\ i \neq \ell}}^{k} \frac{1}{m^i} A^{iT} e^i \\ &= \ (1 - k)\tfrac{A^{\ell T} e^\ell}{m^\ell} \ + \ \textstyle\sum_{\substack{i=1 \\ i \neq \ell}}^{k} \frac{1}{m^i} A^{iT} e^i \ = \ 0 \end{aligned} \tag{6.11}$$

$$\nabla_{\gamma^\ell} f(w, \gamma) \ = \ \sum_{\substack{j=1 \\ j \neq \ell}}^{k} \frac{1}{m^\ell} e^\ell e^\ell \ - \ \sum_{\substack{i=1 \\ i \neq \ell}}^{k} \frac{1}{m^i} e^i e^i \ = \ (k-1) \ - \ (k-1) \ = \ 0 \tag{6.12}$$

Equation (6.12) is automatically satisfied. Obviously, (6.9) implies (6.11). To show the converse, suppose that condition (6.9) does not hold. Without loss of generality let $\frac{e^1 A^1}{m^1} > \frac{e^2 A^2}{m^2}$. If condition (6.11) holds, then we have the contradiction

$$(k-1)\frac{e^1 A^1}{m^1} > (k-1)\frac{e^2 A^2}{m^2} = \frac{e^1 A^1}{m^1} + \sum_{j=3}^{k} \frac{e^j A^j}{m^j} > \frac{e^2 A^2}{m^2} + \sum_{j=3}^{k} \frac{e^j A^j}{m^j} = (k-1)\frac{e^1 A^1}{m^1}. \tag{6.13}$$

Hence (6.11) does not hold and the proof is complete.

□

It is also true for the LP (6.5), that the null solution occurs if and only if condition (6.9) holds [8]. However, condition (6.9) was written in a slightly more complex form in [8, Equation (10)]. For real-world classification problems, all k classes rarely have the same mean. Thus the null solution does not pose a computational difficulty from a practical standpoint.

6.2 PQM and partial gradient distribution

In this section we examine serial and parallel methods for solving the piecewise quadratic minimization (6.6). This problem may be solved serially by any unconstrained first-order optimization method. Our computational results, presented in Section 6.3, indicate that a quasi-Newton method [58, p.2] was considerably faster than solving the corresponding linear program (6.5). Parallel approaches are attractive for machine learning problems because the problem size may be quite large and the problem may need to be solved many times in the course of a decision-tree construction. We took advantage of the structure of the problem, and applied the parallel gradient distribution (MCD-PGD) method [47] that is described below. We refer the reader to [47] for more details of MCD-PGD.

The parallel gradient distribution algorithm theorem is based on forcing function arguments. The definition of a forcing function is provided below. Some typical forcing functions are $\alpha\zeta$, $\alpha\zeta^2$, $\max\{\sigma_1(\zeta),\ \sigma_2(\zeta)\}$, $\min\{\sigma_1(\zeta),\ and\ \sigma_2(\zeta)\}$ where $\sigma_1(\zeta)$ and $\sigma_2(\zeta)$ are forcing functions.

Definition 6.2.1 ***Forcing function*** *A continuous function σ from the nonnegative real line R_+ into itself such that $\sigma(0) = 0$, $\sigma(\zeta) > 0$ for $\zeta > 0$ and such that for the sequence of nonnegative real numbers $\{\zeta_i\}$:*

$$\{\sigma(\zeta_i)\} \to 0 \ \ implies \ \ \{\zeta_i\} \to 0.$$

is said to be a forcing function on the sequence $\{\zeta_i\}$.

The following theorem describes the PGD used for this work. See [47] for the convergence proof of this theorem and other related algorithms.

Theorem 6.2.1 ***Parallel gradient distribution algorithm theorem 1 [47, Corollary 3.2]*** *Let $f \in C^1(R^h)$. Start with any $x_0 \in R^h$. Having x_i stop if $\nabla f(x_i) = 0$, else compute x_{i+1} from directions $d_i^\ell \in R^{h^\ell}$, and stepsizes $\lambda_i^\ell \in R$, $\ell = 1, \ldots, k$, $\sum_{\ell=1}^{k} h^\ell = h$, as follows:*

Direction d_i^ℓ:

$$-\nabla_\ell f(x_i) d_i^\ell \geq \tau_\ell(\|\nabla_\ell f(x_i)\|),\ \ell = 1, \ldots k \tag{6.14}$$

where τ_ℓ is a forcing function on $\{\|\nabla_\ell f(x_i)\|\}$, $\ell = 1, \ldots, k$.
Asynchronous Stepsize: *Choose y_i^ℓ, $\ell = 1, \ldots, k$, such that for $\bar{\ell}$, the complement of ℓ in $\{1, \ldots, k, \}$:*

$$f(x_i) - f(y_i^\ell,\ x_i^{\bar{\ell}}) \geq \mu_\ell(-\nabla_\ell f(x_i) d_i^\ell) \geq 0,\ \ell = 1, \ldots, k \tag{6.15}$$

where μ_ℓ is a forcing function on the sequence of nonnegative real numbers $\{-\nabla_\ell f(x_i) d_i^\ell\}$ for bounded $\{d_i^\ell\}$, $\ell = 1, \ldots, k$.
Synchronization: *Find x_{i+1} such that*

$$f(x_{i+1}) \leq \min_{1 \leq \ell \leq k} f(y_i^\ell,\ x_i^{\bar{\ell}}) \tag{6.16}$$

Then, either $\{x_i\}$ terminates at a stationary point $x_{\bar{i}}$ of $\min_x f(x)$, or for each accumulation point $(\bar{x},\ \bar{d})$ of $\{x_i,\ d_i\}$, $\bar{x}$ is a stationary point of $\min_x f(x)$.

In [47] a number of implementations of Theorem 6.2.1 were proposed including gradient descent and quasi-Newton directions and stepsizes such as the Armijo and minimization stepsizes. We shall use another implementation of Theorem 6.2.1 based on the following simple remarks. Instead of choosing y^ℓ, $\ell = 1, \ldots, k$, so as to satisfy the realizable inequalities (6.14) and (6.15), we take the best possible y_i^ℓ, that is:

$$f(y_i^\ell, x_i^{\bar{\ell}}) \ = \ \min_{x^\ell} f(x^\ell, x_i^{\bar{\ell}}), \quad \ell = 1, \ldots, k. \tag{6.17}$$

Hence conditions (6.14) and (6.15) are satisfied for some τ_ℓ and μ_ℓ, $\ell = 1, \ldots, k$. Similarly for the synchronization step (6.16), take the best possible x_{i+1} over the affine hull of x_i and $(y_i^\ell, x^{\bar{\ell}})$, $\ell = 1, \ldots, k$. Hence (6.16) is satisfied. Consequently we have the following parallel algorithm that we propose for our multicategory discrimination problem. This algorithm can also be considered as a parallel variable distribution algorithm [23].

Theorem 6.2.2 ***Parallel gradient distribution algorithm theorem 2(PGD)*** *Let $f \in C^1(R^h)$. Start with any $x_0 \in R^h$. Having x_i stop if $\nabla f(x_i) = 0$, else compute x_{i+1} as follows.*

Parallelization: *Find $y_i^\ell \in R^{h^\ell}$, $\ell = 1, \ldots, k$ such that*

$$(y_i^\ell, x^{\bar{\ell}}) \in arg \min_{x^\ell} f(x^\ell, x_i^{\bar{\ell}}) \tag{6.18}$$

Synchronization: *Find $x_{i+1} \in R^h$ such that*

$$\begin{aligned} x_{i+1} &= \lambda_i^0 x_i + \lambda_i^1(y_i^1, x_i^{\bar{1}}) + \ldots + \lambda_i^k(y_i^k, x_i^{\bar{k}}) \\ &\in arg \min_{\lambda^0, \lambda^1, \ldots, \lambda^k} f(\lambda^0 x_i + \lambda^1(y_i^1, x_i^{\bar{1}}) + \ldots + \lambda^k(y_i^k, x_i^{\bar{k}})) \end{aligned} \tag{6.19}$$

Then, either the algorithm terminates at a stationary solution $x_{\bar{i}}$ of $\min_x f(x)$, or for each accumulation point $\bar{x}$ of $\{x_i\}$, $\bar{x}$ is a stationary point of $\min_x f(x)$.

A natural way to apply Theorem 3.2 to PQM (6.6) is to let $x_i^\ell = (w_i^\ell, \gamma_i^\ell)$. This results in the following algorithm.

Algorithm 6.2.1 ***MCD-PGD(Multicategory Discrimination via PGD)*** *Start with $x_0 = (w_0, \gamma_0) \in R^{kn+k}$ and define $f(x) = f(w, \gamma)$ as in (6.6).*

- *Stop if $\nabla f(x_i) = 0$.*
- ***Parallelization:*** *For each class $\ell = 1, \ldots, k$ find $y_i^\ell \in R^{n+1}$:*

$$(y_i^\ell, x_i^{\bar{\ell}}) \in arg \min_{x^\ell} f(x^\ell, x_i^{\bar{\ell}}) \tag{6.20}$$

Stop if $\nabla f(y_i^\ell, x_i^{\bar{\ell}}) = 0$.

- ***Synchronization:***

$$\begin{aligned} x_{i+1} &= \lambda_i^0 x_i + \lambda_i^1(y_i^1, x_i^{\bar{1}}) + \ldots + \lambda_i^k(y_i^k, x_i^{\bar{k}}) \\ &\in arg \min_{\lambda^0, \lambda^1, \ldots, \lambda^k} f(\lambda^0 x_i + \lambda^1(y_i^1, x_i^{\bar{1}}) + \ldots + \lambda^k(y_i^k, x_i^{\bar{k}})) \end{aligned} \tag{6.21}$$

- *Repeat*

We used the following simple heuristic for choosing a starting point $x_0 = (w_0, \gamma_0)$, which consists of taking w_0^ℓ as the difference between the mean of class ℓ and the mean of all the points:

$$w_0^\ell = \frac{e^\ell A^\ell}{m^\ell} - \frac{\sum_{j=1}^{k} e^j A^j}{\sum_{j=1}^{k} m^j}, \qquad \gamma_0^l = 0, \qquad l = 1, \ldots, k. \tag{6.22}$$

The unconstrained convex minimization subproblems, (6.20) and (6.21), were solved by using the quasi-Newton algorithm in the MINOS [58] optimization package.

Many variations of the direction and synchronization steps of Algorithm 6.2.1 are possible under Theorem 6.2.1. The algorithm presented was the best we found computationally. The algorithm is easily parallelized by distributing each of the subproblems (6.20) among k processors. The processors then synchronize once to share the results of the ℓ subproblems and the result of the synchronization step. We limited the number of iterations within the subproblems to the number of variables in the problem. This prevented one processor from spending too much time on one subproblem thus causing the other processors to be idle. We also relaxed the termination criteria slightly. The algorithm was halted if the gradient was sufficiently small (10^{-3}) or if the change in the objective function between major iterations was too small. Computational results in Section 7.4.2 show that the relaxation of the optimality condition did not adversely effect the quality of the solution found in terms of the number of points misclassified in the training and test sets.

We experimented with variations of the direction and synchronization steps. For example, the synchronization step (6.21) was replaced with a strong convex combination of the k points found in step (6.20) such as the average of the k points. This synchronization step was too conservative. The time per iteration was reduced but the number of iterations greatly increased. The final approach described above was adopted after a number of trials.

6.3 Computational results

We conducted a series of computational experiments to investigate three questions: How does the LP formulation (6.5) compare with the PQM formulation (6.6)? How does the serial MCD-PGD algorithm compare with a purely serial quasi-Newton algorithm? And how well does the MCD-PGD algorithm perform on a parallel machine? The serial experiments were performed on a DECstation 5000/125. The parallel experiments were performed on a Thinking Machines CM-5 parallel processor. The linear programming and quadratic subproblems were solved using the MINOS [58] package. The following discrimination problems were used to compare the algorithms. The wine recognition data [1], referred to as wine, is piecewise-linear separable. Fisher's classical Iris identification problem [24], referred to as Iris, is almost piecewise-linear separable. In the forensic glass identification data [21], referred to as glass, is not piecewise-linear separable. In the image segmentation problem [15], the data is divided into two parts: a training set consisting of 210 points and a testing set consisting of 2310 points. We refer to the set of 210 points as image-s since it is piecewise-linear separable, and the set of 2310 points as image-n since it is not piecewise-linear separable. Appendix A lists the number of points, attributes, and classes contained in each of the data sets.

6.3.1 Comparison of serial implementation of LP and PQM

We compared the linear programming formulation (6.5) and the new piecewise-quadratic minimization formulation (6.6) on the machine learning problems: wine, Iris, glass, and image-s described above. The LP (6.5) was solved serially using MINOS [58], and PQM (6.6) was solved serially by a quasi-Newton method employed by MINOS. Note that the goal of these problems is to construct a function for classifying future unseen points. Thus we used three criteria to evaluate the algorithms: the time to construct the function (the training time), the percent correctness on the training set, and the percent correctness on unseen points.

We used 10-fold cross-validation [39] to estimate these criteria. In 10-fold cross-validation, $\frac{9}{10}$ of the points were used for training and $\frac{1}{10}$ of the points were held out and tested on the resulting function. This is repeated 10 times, once for each $\frac{1}{10}$ used as the testing set. The results of the training time, testing set accuracy, and training set accuracy were averaged over the 10 trials. This was performed on each of the above data sets. The training set accuracies, testing set accuracies, and training times are given in Table 7.2 together with their standard deviations. The accuracies are given in terms of percent correctness. Times are seconds of CPU time on a DECstation 5000/125. The p-value gives the significance of a paired t-test between the results for the LP (6.5) and the results of PQM (6.6). Low p values indicate a significant difference between the means of the results.

The results indicate that the PQM formulation is considerably superior with respect to training time, sometimes by an much as an order of magnitude. The training set accuracies for the LP and PQM formulations were virtually the same. However, the testing set accuracy for PQM was better than the LP results. Further investigation is needed to determine the best choice of error formulation for good generalization (testing set accuracy). The training time was clearly faster for PQM. Thus PQM achieved a significant improvement in run-time performance even before parallelization was introduced.

6.3.2 Comparison of serially implemented MCD-PGD and quasi-Newton

In the second set of experiments, we compared the computational results of solving PQM (6.6) with a quasi-Newton method versus solving PQM with the MCD-PGD Algorithm 6.2.1 implemented on the DECstation/125 serial machine. In addition to the wine, Iris, glass, and image-s problems, the large image-n problem was added. Table 7.3 gives the training set accuracies, the testing set accuracies, and the training times for both algorithms on the five datasets.

Table 6.1: Comparison of Serial Implementation of Linear Program (6.5) and Piecewise-Quadratic Minimization(6.6)

Training Set Accuracy

Dataset	Average Accuracy (%)		t-test
	LP	PQM	p
Wine	100.0 ± 0.0	100.0 ± 0.0	1.0
Iris	98.8 ± 0.6	98.8 ± 0.6	1.0
Glass	76.3 ± 1.2	74.1± 2.0	0.0005
Image-s	100.0 ± 0.0	99.9 ± 0.0	0.34

Testing Set Accuracy

Dataset	Average Accuracy (%)		t-test
	LP	PQM	p
Wine	89.4 ± 7.6	93.9 ± 7.1	0.02
Iris	94.7 ± 6.8	97.3 ± 4.7	0.10
Glass	60.8 ± 11.4	61.3 ± 13.1	0.74
Image-s	79.1 ± 0.1	85.3 ± 7.4	0.08

Training Time

Dataset	Training Time (secs)		t-test
	LP	PQM	p
Wine	14.3 ± 1.2	5.2 ± 1.5	< 0.00001
Iris	5.8 ± 0.7	0.4 ± 0.1	< 0.00001
Glass	231.2 ± 25.2	12.4 ± 25.1	< 0.00001
Image-s	610.5 ± 107.0	96.9 ± 13.1	< 0.00001

The average training set and testing set accuracies were not significantly different on any dataset. The relaxation of the optimality criterion discussed in Section 6.2 does not adversely affect the testing set accuracy of the solution on these problems. In practice, stopping before the objective function is exactly optimal (i.e. $\nabla f(x_i) = 0$) may improve generalization as well as as training time. In machine learning applications, requiring exact optimality can cause over-fitting. For the backpropagation algorithm [70], one successful stopping criteria is to reserve part of the training set as a tuning set, and to stop the algorithm when the accuracy on the tuning set decreases [41, p. 41-42]. We plan to investigate in the future the use of such tuning sets to halt the algorithm.

The training times for the MCD-PGD and quasi-Newton algorithms were competitive. For small problems the quasi-Newton algorithm is clearly a better choice. However, for larger problems such as glass, image-s and image-n, MCD-PGD did as well as and even better than quasi-Newton. Ignoring communication costs and idle time, this indicates that for large problems 100% speedup efficiency **may** be achieved using parallel computation. The next section investigates the actual speedup efficiency achieved by MCD-PGD on the CM-5 parallel machine.

6.3.3 Comparison of parallel implementation of MCD-PGD and quasi-Newton

For the final set of comparisons, we implemented Algorithm 6.2.1 on the Wisconsin 32-node CM-5 parallel processor. For a k-class discrimination problem, we used a parallel version of MCD-PGD on k nodes. For comparison we ran the quasi-Newton method on 1 node. We limited the investigation to the three datasets (glass, image-s, and image-n), that exhibited promising theoretical speedup in the above serial experiment. The average computation time over the 10 cross-validation runs is reported in Table 7.4. There was a significant decrease in computation time using MCD-PGD over quasi-Newton. The speedup efficiency, that is the ratio of time on 1-node divided by k times the time on k nodes, was 50-91%.

Table 6.2: Comparison of serial implementation of MCD-PGD and quasi-Newton algorithms on DECstation 5000/125

Training Set Accuracy

Dataset	Average Accuracy (%)		t-test
	PGD	Quasi-Newton	p
Wine	100.0 ± 0.0	100.0 ± 0.0	1.00
Iris	98.7 ± 0.5	98.9 ± 0.6	0.34
Glass	73.5 ± 1.0	73.6 ± 1.5	0.60
Image-s	100.0 ± 0.0	100.0 ± 0.0	1.00
Image-n	96.0 ± 2.9	96.5 ± 1.3	0.003

Testing Set Accuracy

Dataset	Average Accuracy (%)		t-test
	PGD	Quasi-Newton	p
Wine	91.6 ± 3.9	91.1 ± 8.3	0.84
Iris	96.0 ± 4.5	96.7 ± 4.7	0.34
Glass	63.1 ± 11.1	63.1 ± 10.5	1.00
Image-s	86.7 ± 6.5	86.7 ± 7.0	1.00
Image-n	95.5 ± 1.2	95.4 ± 1.3	0.68

Training Time

Dataset	Training Time (secs)		t-test
	PGD	Quasi-Newton	p
Wine	9.4 ± 2.0	4.5 ± 1.5	0.00025
Iris	1.6 ± 0.5	0.43 ± 0.1	0.00002
Glass	33.4 ± 2.4	32.4 ± 9.9	0.75
Image-s	111.5 ± 24.9	138.2 ± 58.1	0.09
Image-n	1294.0 ± 132.1	1748.2 ± 1130.6	0.20

Table 6.3: Comparison of MCD-PGD and quasi-Newton algorithms on three 7-class problems implemented on the CM-5

Training Time

Dataset	Training Time (secs)		t-test p	time reduction	efficiency (%)
	Quasi-Newton	PGD			
Glass	63.3 ± 24.0	17.7 ± 3.0	< 0.00001	3.6	51
Image-s	203.6 ± 81.7	32.1 ± 9.1	< 0.00001	6.3	91
Image-n	2470.0 ± 1113.1	704.0 ± 137.0	< 0.00001	3.6	50

The lower efficiency is primarily caused by segments of the algorithm that create idle time. The subproblems (6.20) solved in the parallelization step may take different amounts of computational time. The other processors remain idle until the last processor finishes. We tried to minimize this effect by limiting the number of iterations in the subproblems. The synchronization step (6.21) in Algorithm 6.2.1 causes processors to remain idle thus decreasing the efficiency. Two possible approaches to improve efficiency are: use of a cheaper synchronization step, and allowing each processor to do its own synchronization as soon as it finishes. The latter approach is suitable for a shared memory machine and would result in an asynchronous algorithm. These are directions for future work.

6.4 Strengths

We have proposed an easily parallelizable formulation for the multicategory discrimination problem that consists of minimizing a piecewise-quadratic function. This formulation is comparable in accuracy to previous linear programming formulations, but is considerably faster when implemented serially. We developed

a parallel gradient distribution algorithm to minimize a piecewise-quadratic error function on both serial and parallel machines. The serial implementation holds the promise of a fast parallel implementation once idle and communication costs are minimized. The parallel implementation efficiencies of 50% to 91% are good and can be further improved via an asynchronous algorithm. Actual computation time was reduced on average by a factor of 4.5.

6.5 Extensions

Several extensions to PGD are planned: improved efficiency via asynchronous processing, alternate stopping criteria, alternate partitioning of variables, and utilization in parallel decision-tree algorithms.

The first step is to improve the efficiency of the parallel implementation. A totally asynchronous algorithm is needed. One possibility is to have each processor perform the "synchronization" using the most recent data available as soon as its subproblem solution is complete. This would eliminate the idle time caused by waiting for other processors to complete. The ramifications of this algorithmic change on the convergence theory will need to be determined.

Another extension is to examine alternative stopping criteria. As discussed in Section 6.3, tuning sets could be used in order to stop the subproblems and major iterations when generalization decreases. This might lead to improved generalization and improved training times. Alternate partitioning refers to the way the variables are divided into subproblems. Currently the variables are divided by class. An alternative would be to divide the subproblems by attributes. For most problems, there are more attributes than classes so greater usage of the parallel machine could be achieved.

We plan to incorporate PGD into a parallel decision-tree algorithm. The parallel decision-tree algorithm could exploit parallelism by constructing many nodes in the decision tree at once, and by applying PGD to the construction of each node. The algorithm could also evaluate other types of univariate or

multivariate splits such as those found in CART [14], ID3 [63], and FACT [42], and then choose the best split out of the ones generated. This might result in increased generalization.

Chapter 7

Bilinear separation of two sets

In this chapter, we examine the NP-complete problem of determining whether two disjoint point sets in the n-dimensional real space R^n can be separated by two planes. Analogously to the approach to linear separability followed in Chapter 3, we define bilinear separability as two sets of disjunctive linear inequalities. To minimize the infeasibilities of these inequalities, we propose a bilinear program that minimizes the scalar product of two linear functions on a polyhedral set. This bilinear program, which has a vertex solution, is processed by an iterative linear programming algorithm that terminates in a finite number of steps at a point satisfying a necessary optimality condition or at a global minimum. This algorithm can be applied to other bilinear programming programs. Computational experience shows that our approach is very effective on bilinear separable problems. Much of this chapter will appear in [7].

We begin with background on the bilinear separation problem. In Section 7.2 we prove some basic results on the existence of vertex solutions to bilinear programs as well as some linear-programming-based finite algorithms for their solution. Because of the special property of a zero minimum for bilinearly separable problems, we have opted for the simpler Frank-Wolfe type algorithms [26] (see Appendix B), rather than the more complex algorithms that have been given for bilinear programs [2, 85, 75, 79]. In Section 7.3, we define bilinear separability

and show the equivalence of this definition to bilinear programming. Our computational results, summarized in Section 7.4, indicate that the proposed algorithms are quite effective, especially in view of the fact that the underlying problem is NP-complete. More precisely, only the bilinear separability problem corresponding to Figure 7.1(a)-7.1(b) has been shown to be NP-complete [53, 13]. That the corresponding problem to Figure 7.1(c) is NP-complete as well can be deduced from Theorem 7.3.1 below by noting that the bilinear program (7.14) is a special instance of the bilinear program (7.16).

7.1 Bilinear separation

The problem we wish to consider is the following: Given two disjoint point sets $\mathcal{A}^1$ and $\mathcal{A}^2$ in the n-dimensional real space R^n, can they be (strictly) separated by two planes? This is a fundamental NP-complete problem [53, 13] that is depicted in Figure 7.1 for the 2-dimensional real space R^2. The configurations (a) and (b) of Figure 7.1 are equivalent as can easily be seen if the roles of $\mathcal{A}^1$ and $\mathcal{A}^2$ are interchanged. Bilinear separation is a natural extension of linear separation, which has long been known to be equivalent to the polynomial-time solution of a single linear program [16, 43, 73, 9]. Linear separation is also equivalent to separation by Rosenblatt's perceptron or linear threshold unit (LTU) [67, 70, 34] (see Figure 7.2). However, most problems are not linearly separable. For example, the simple Minsky-Papert Exclusive-Or classical problem [55] is not linearly separable, but is bilinearly separable. It can be solved by a neural network with 2 layers of LTUs [70, 51](see Figure 7.3).

It is interesting to note that the bilinear separation depicted in Figure 7.2(a), which corresponds to the topology of the bilinear separation of Figure 7.1(a) and 7.1(b), can be represented by a single hidden layer neural network with two hidden units and one output unit. However, this is not the case for a bilinear separation with the topology of Figure 7.1(c). In fact, if we separated the Exclusive-Or example by planes using this topology, as shown in Figure 7.4(a), the corresponding

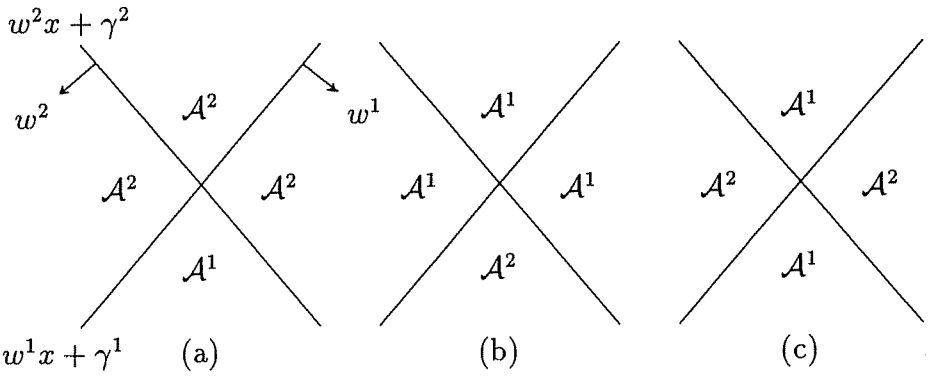
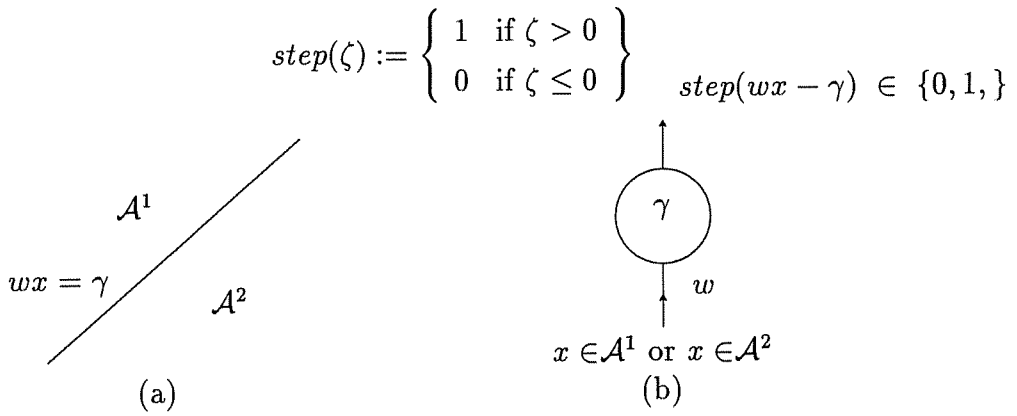


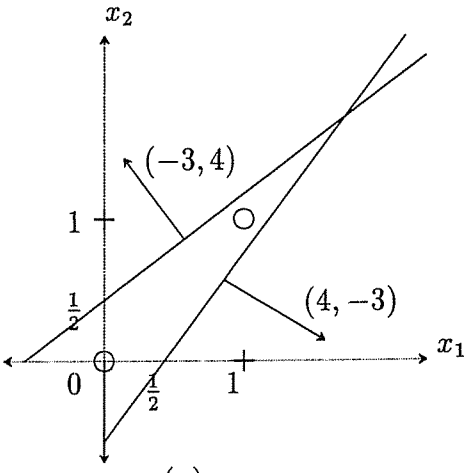
Figure 7.1: Bilinearly separable sets $\mathcal{A}^1$ and $\mathcal{A}^2$ in R^n



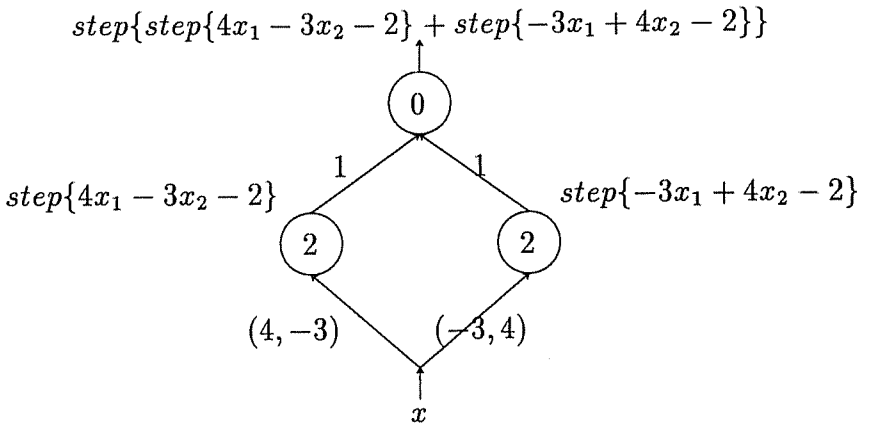
(*a*) *Linearly separable sets*

(*b*) *Equivalent linear threshold unit with incoming weights* w *and threshold* γ

Figure 7.2: Linear separator and equivalent linear threshold unit



(a)

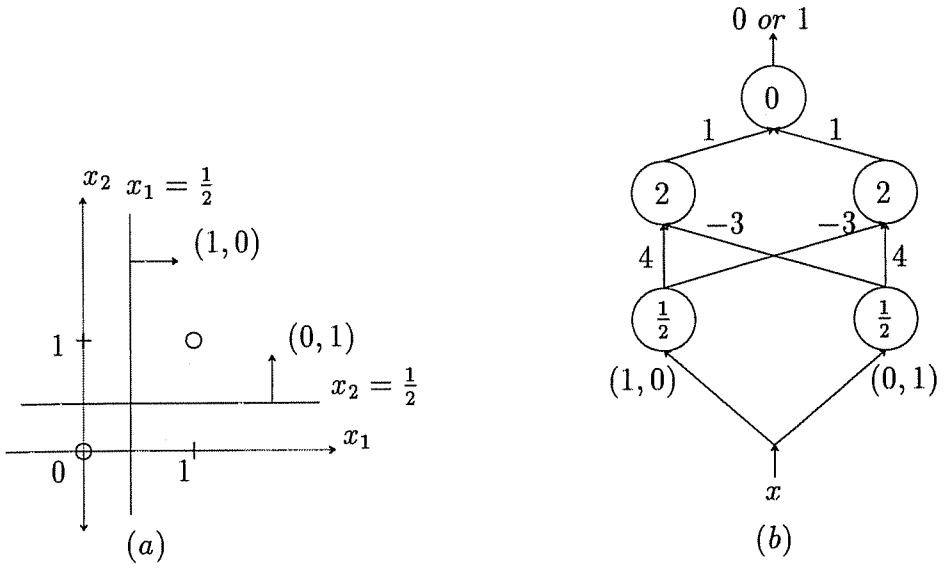


(b)

(a) *Bilinearly separable exclusive – or example* : $\mathcal{A}^1 = \{(0,0),(1,1)\}$, $\mathcal{A}^2 = \{(1,0),(0,1)\}$

(b) *Equivalent neural network*

Figure 7.3: Bilinear separator for exclusive-or example and equivalent neural network



(*a*) *Bilinear separation using the topology of Figure* 7.1(*c*)

(*b*) *Equivalent neural network*

Figure 7.4: Alternative bilinear separator for exclusive-or example and equivalent neural network

neural network depicted in Figure 7.4(b), requires two hidden layers each with 2 units and one output unit. This indicates that this is a more complex separation from a neural network point of view . It will turn out that this separation leads also to a more difficult bilinear program with coupled constraints: Program (7.16) instead of Program (7.14) below.

7.2 Vertex solutions and finite algorithms for bilinear programs

We present here some simple basic results for bilinear programs which show under what conditions these problems have vertex solutions. These results are then used to generate finite linear-programming-based algorithms for solving bilinear separability problems as bilinear programs (see Section 7.3). We shall consider two categories of bilinear programs corresponding to cases (a)-(b) and to case (c) of Figure 7.1. The first case will have uncoupled constraints while the second case will have coupled constraints. We note that there are many papers on bilinear programs with uncoupled constraints such as [75, 79, 61, 38], and with coupled constraints [2, 85]. However, none of the papers on coupled constraints appear to exploit zeroness of the minimum as we do here for the case of bilinearly separable sets. This fact allows us to conclude that a vertex solution exists for such problems. This in turn leads to finite termination for the proposed algorithms. For uncoupled constraints, the existence of a vertex solution is known and has been exploited algorithmically [75, 79]. For completeness and contrast with the proof for the case of coupled bilinear programs, we begin with simple proof of the existence of a vertex solution for uncoupled bilinear programs.

Proposition 7.2.1 ***(Existence of vertex solutions for uncoupled bilinear programs)*** *If the uncoupled bilinear program*

$$\min_{x,y,r,s} \{\, xy \mid Cx + Er \geq g,\ Dy + Fs \geq h,\ (x,y,r,s) \geq 0\} \tag{7.1}$$

is feasible, then it has a vertex solution.

Proof. Since the quadratic objective function is bounded below by zero on the polyhedral feasible region, it must have a solution $(\bar{x}, \bar{y}, \bar{r}, \bar{s})$ [26]. Hence the linear program

$$\min_{(x,r)\in\mathcal{X}} x\bar{y},\ \ \mathcal{X} = \{\, Cx + Er \geq g,\ (x,r) \geq 0\}$$

has a vertex $(\hat{x}, \hat{r})$ of its feasible region $\mathcal{X}$ as solution [59], and such that

$$\hat{x}\bar{y} = \bar{x}\bar{y}.$$

Similarly, the linear program

$$\min_{(y,s)\in\mathcal{Y}} \hat{x}y, \quad \mathcal{Y} = \{ Dy + Fs \geq h, \ (y,s) \geq 0\}$$

has a vertex $(\hat{y}, \hat{s})$ of its feasible region $\mathcal{Y}$ as solution, and such that

$$\hat{x}\hat{y} = \hat{x}\bar{y} = \bar{x}\bar{y}.$$

Hence $((\hat{x}, \hat{r}), (\hat{y}, \hat{s}))$ is a vertex of $\mathcal{X} \times \mathcal{Y}$ and a vertex solution of (7.1). □

To establish the existence of a vertex solution to a coupled bilinear program we require the additional assumption that the minimum value of the bilinear objective be zero. This does not affect the application to bilinear separation, since the objective function does indeed become zero for bilinearly separable sets.

Proposition 7.2.2 ***(Existence of vertex solution for coupled bilinear programs)*** *If the coupled bilinear program*

$$\min_{x,y,r} \{ xy \mid Cx + Dy + Er \geq g, \ (x,y,r) \geq 0\} \tag{7.2}$$

has a zero minimum, then it has a vertex solution.

Proof. Let $\mathcal{S}$ denote the feasible region of (7.2). Note first that

$$xy = 0, \ (x,y) \geq 0 \ \Leftrightarrow \ x - (x-y)_+ = 0$$

and for $x, y \in R^n$ and $i = 1, \ldots, n$

$$\begin{aligned} (x,y,r) \in \mathcal{S} \ &\Rightarrow \ (x,y) \geq 0 \\ &\Rightarrow \ ((x-(x-y)_+)_i = \left\langle \begin{array}{ll} y_i & \text{if } x_i - y_i \geq 0 \\ x_i & \text{if } x_i - y_i < 0 \end{array} \right\rangle \geq 0 \end{aligned}$$

Hence

$$0 = \min_{(x,y,r)\in\mathcal{S}} xy \;\Leftrightarrow\; 0 = \min_{(x,y,r)\in\mathcal{S}} e(x-(x-y)_+)$$

However, $e(x-(x-y)_+)$ is a concave function because $e(x-y)_+$ is convex. Since we are minimizing a concave function on a polyhedral set not containing lines going to infinity in both directions, it must have a vertex solution [66, Corollary 32.3.4]. □

The above proof is based on the proofs of Lemmas 1 and 2 of [46] which show that every solvable linear complementarity problem, monotonic or not, has a vertex solution. With the above theorems we can formulate finite algorithms for each of the uncoupled and coupled bilinear programs: algorithms UBPA 7.2.1, UBPA1 7.2.2 and BPA 7.2.3.

Algorithm 7.2.1 (Uncoupled bilinear program algorithm (UBPA)) *Start with any feasible point* (x^0, y^0, r^0, s^0) *for (7.1). Determine* $(x^{i+1}, y^{i+1}, r^{i+1}, s^{i+1})$ *from* (x^i, y^i, r^i, s^i) *as follows:*

$$(x^{i+1}, r^{i+1}) \in \textit{ arg vertex partial } \min_{x,r} \{\, xy^i \mid Cx + Er \geq g,\ (x,r) \geq 0\}$$

$$(y^{i+1}, s^{i+1}) \in \textit{ arg vertex partial } \min_{y,s} \{\, x^{i+1}y \mid Dy + Fs \geq h,\ (y,s) \geq 0\}$$

and *such that* $x^{i+1}y^{i+1} < x^iy^i$. *Stop when impossible.*

In the above algorithm, "*arg vertex partial* min" denotes a vertex in the solution set of the indicated linear program, or any vertex along the path of the simplex or other pivotal method that attempts to decrease the objective function. The combined decrease of both steps must be such that $x^{i+1}y^{i+1} < x^iy^i$. We now establish finiteness of the above algorithm.

Theorem 7.2.1 (Finite termination of UBPA) *Let the uncoupled bilinear program (7.1) be feasible. Then UBPA 7.2.1 terminates in a finite number of steps at*

a global solution or a point $(x^{i+1}, y^i, r^{i+1}, s^i)$ *that satisfies the minimum principle necessary optimality condition [45]*

$$y^i(x - x^{i+1}) + x^{i+1}(y - y^i) \geq 0 \quad \forall\, (x,r) \in \mathcal{X},\ \forall\, (y,s) \in \mathcal{Y}. \tag{7.3}$$

Proof. If for some i, $x^{i+1}y^{i+1} \not< x^iy^i$, then each of the linear programs of UBPA must have been solved to optimality and

$$xy^i \geq x^iy^i = x^{i+1}y^i = x^{i+1}y^{i+1} \leq x^{i+1}y, \quad \forall\, (x,r) \in \mathcal{X},\ \forall\, (y,s) \in \mathcal{Y}$$

from which the minimum principle condition (7.3) follows. Since there are a finite number of vertices of $\mathcal{X} \times \mathcal{Y}$, and since each vertex visited by UBPA is less than the previous one in xy-value, no vertex is repeated. Thus UBPA must terminate at either a global minimum or a point satisfying the minimum principle (7.3). □

Note that UBPA is serial in nature in that (y^{i+1}, s^{i+1}) is computed **after** (x^{i+1}, r^{i+1}) is computed. A parallel version, Algorithm 7.2.2 below, where both (x^{i+1}, r^{i+1}) and (y^{i+1}, s^{i+1}) are computed simultaneously, can be shown to possess the finite termination property. We omit the proof of finite termination at a global solution or a stationary point for Algorithm 7.2.2, since it is completely analogous to the proof of Theorem 7.2.1.

Algorithm 7.2.2 (Uncoupled bilinear program algorithm 1 (UBPA1)) *Start with any feasible point* (x^0, y^0, r^0, s^0) *for (7.1). Determine* $(x^{i+1}, y^{i+1}, r^{i+1}, s^{i+1})$ *from* (x^i, y^i, r^i, s^i) *as follows:*

$$(\tilde{x}^{i+1}, \tilde{r}^{i+1}) \in \textit{ arg vertex partial} \min_{x,r} \{\, xy^i \mid Cx + Er \geq g,\ (x,r) \geq 0\}$$

$$(\tilde{y}^{i+1}, \tilde{s}^{i+1}) \in \textit{ arg vertex partial} \min_{y,s} \{\, x^iy \mid Dy + Fs \geq h,\ (y,s) \geq 0\}$$

$$(x^{i+1}, y^{i+1}) \in \{(\tilde{x}^{i+1}, y^i), (x^i, \tilde{y}^{i+1}), (\tilde{x}^{i+1}, \tilde{y}^{i+1})\}$$

and *such that* $x^{i+1}y^{i+1} < x^iy^i$. *Stop when impossible.*

We turn our attention now to the more general case and give a finite Frank-Wolfe algorithm for the coupled bilinear program (7.2) when its objective function has a zero minimum.

Algorithm 7.2.3 (Bilinear program algorithm (BPA)) *Start with any feasible point* (x^0, y^0, r^0) *for (7.2). Determine* $(x^{i+1}, y^{i+1}, r^{i+1})$ *from* (x^i, y^i, r^i) *as follows:*

- $(u^i, v^i, w^i) \in$ *arg vertex* $\min_{(x,y,r)} \{ xy^i + x^i y \mid Cx + Dy + Er \geq g, (x, y, r) \geq 0 \}$
- *Stop if* $u^i y^i + x^i v^i = 2x^i y^i$
- $(x^{i+1}, y^{i+1}, r^{i+1}) = (1 - \lambda^i)(x^i, y^i, r^i) + \lambda^i (u^i, v^i, w^i)$ *where*
  $\lambda^i \in$ *arg* $\min_{0 \leq \lambda \leq 1} (x^i + \lambda(u^i - x^i))(y^i + \lambda(u^i - y^i))$

Theorem 7.2.2 (Convergence and finite termination theorem for BPA) *Either the sequence* $\{(x^i, y^i, r^i)\}$ *of BPA terminates at some* $\{(x^j, y^j, r^j)\}$ *satisfying the minimum principle necessary optimality condition*

$$y^j(x - x^j) + x^j(y - y^j) \geq 0 \quad \forall (x, y, r) \in \mathcal{S}, \tag{7.4}$$

where $\mathcal{S}$ *is the feasible region of (7.2), or each accumulation point* $(\bar{x}, \bar{y}, \bar{z})$ *of* $\{(x^i, y^i, r^i)\}$ *satisfies the minimum principle. If* $\bar{x}\bar{y} = 0$, *then one of the vertices of the sequence* $\{(u^i, v^i, w^i)\}$ *solves the bilinear program (7.2) and* $u^i v^i = 0$.

Proof. Follows from Theorems B.0.1 and B.0.2 of Appendix B. □

In practice, all bilinearly separable examples attempted terminated at a zero minimum in a finite number of steps. For bilinearly inseparable problems, such termination may not be possible, and in fact the sequence may be unbounded. The latter could be an indication of bilinear inseparability. Should the problem accumulate to a stationary point for which $\bar{x}\bar{y} > 0$, then this would also be an indication of bilinear inseparability. We do not claim that we can always determine bilinear inseparability in all cases, since this is an NP-complete problem. However our computational results are such that we feel confident that our approach will easily determine bilinear separability whenever it exists.

7.3 Equivalence of bilinear separability and bilinear programming

In this section we will show how to reduce the bilinear separability problem to one of three bilinear programs. Consider the disjoint point sets $\mathcal{A}^1$ and $\mathcal{A}^2$ in R^n made up of m^1 and m^2 points and represented by the matrices $A^1 \in R^{m^1 \times n}$ and $A^2 \in R^{m^2 \times n}$ respectively. Recall from Chapter 2 that that when the convex hulls of $\mathcal{A}^1$ and $\mathcal{A}^2$ are disjoint then $\mathcal{A}^1$ and $\mathcal{A}^2$ are said to be **linearly separable**. By using the duality theory of linear programming, this can be shown to be equivalent to the existence of a plane $wx = \gamma$ strictly separating $\mathcal{A}^1$ from $\mathcal{A}^2$ (see Figure 7.2(a)) which is equivalent to [35, 43, 73, 9]

$$-A^1 w + e\gamma + e \leq 0, \quad A^2 w - e\gamma + e \leq 0, \quad \text{for some } w \in R^n, \ \gamma \in R. \tag{7.5}$$

Based on the above definition of linear separability, we now define bilinear separability as follows:

Definition 7.3.1 (Bilinear separability definition (See Figure 7.5))
The sets $\mathcal{A}^1$ and $\mathcal{A}^2$ are bilinearly separable if and only if at least one of the following systems of disjunctive linear inequalities is solvable for $(w^1, w^2, \gamma^1, \gamma^2)$ thus giving the separating planes $w^1 x = \gamma^1$ and $w^2 x = \gamma^2$:

$$\left\langle \begin{array}{lll} -A^1 w^1 + \gamma^1 e + e \leq 0 & \textbf{\textit{and}} & -A^1 w^2 + \gamma^2 e + e \leq 0 \\ A_i^2 w^1 - \gamma^1 + 1 \leq 0 & \textbf{\textit{or}} & A_i^2 w^2 - \gamma^2 + 1 \leq 0, \ i = 1, \ldots, , k \end{array} \right\rangle \tag{7.6}$$

$$\left\langle \begin{array}{lll} -A^2 w^1 + \gamma^1 e + e \leq 0 & \textbf{\textit{and}} & -A^2 w^2 + \gamma^2 e + e \leq 0 \\ A_i^1 w^1 - \gamma^1 + 1 \leq 0 & \textbf{\textit{or}} & A_i^1 w^2 - \gamma^2 + 1 \leq 0, \ i = 1, \ldots, , m \end{array} \right\rangle \tag{7.7}$$

$$\left\langle \begin{array}{c} \left\langle \begin{array}{c} (-A_i^1 w^1 + \gamma^1 + 1 \leq 0 \ \textbf{\textit{and}} \ -A_i^1 w^2 + \gamma^2 + 1 \leq 0) \\ \textbf{\textit{or}} \\ (A_i^1 w^2 - \gamma^2 + 1 \leq 0 \ \textbf{\textit{and}} \ A_i^1 w^1 - \gamma^1 + 1 \leq 0), \ i = 1, \ldots, m \end{array} \right\rangle \\ \\ \left\langle \begin{array}{c} (A_i^2 w^1 - \gamma^1 + 1 \leq 0 \ \textbf{\textit{and}} \ -A_i^2 w^2 + \gamma^2 + 1 \leq 0) \\ \textbf{\textit{or}} \\ (A_i^2 w^2 - \gamma^2 + 1 \leq 0 \ \textbf{\textit{and}} \ -A_i^2 w^1 + \gamma^1 + 1 \leq 0), \ i = 1, \ldots, k \end{array} \right\rangle \end{array} \right\rangle \tag{7.8}$$

It is easy to see that the above definition can be stated in the following alternative form.

Definition 7.3.2 (Alternative bilinear separability definition)
The sets $\mathcal{A}^1$ and $\mathcal{A}^2$ are bilinearly separable if and only if at least one of the following systems of inequalities is solvable for $(w^1, w^2, \gamma^1, \gamma^2)$:

$$\begin{array}{l} -A^1 w^1 + \gamma^1 e + e \leq 0 \ and \ -A^1 w^2 + \gamma^2 e + e \leq 0 \\ (A^2 w^1 - \gamma^1 e + e)_+ (A^2 w^2 - \gamma^2 e + e)_+ \ = \ 0 \end{array} \tag{7.9}$$

$$\begin{array}{l} -A^2 w^1 + \gamma^1 e + e \leq 0 \ and \ -A^2 w^2 + \gamma^2 e + e \leq 0 \\ (A^1 w^1 - \gamma^1 e + e)_+ (A^1 w^2 - \gamma^2 e + e)_+ \ = \ 0 \end{array} \tag{7.10}$$

$$\begin{array}{l} ((-A^1 w^1 + \gamma^1 e + e)_+ + (-A^1 w^2 + \gamma^2 e + e)_+) \times \\ \quad ((A^1 w^2 - \gamma^2 e + e)_+ + (A^1 w^1 - \gamma^1 e + e)_+) \ = \ 0 \end{array} \tag{7.11}$$

$$\begin{array}{l} ((A^2 w^1 - \gamma^1 e + e)_+ + (-A^2 w^2 + \gamma^2 e + e)_+) \times \\ \quad ((A^2 w^2 - \gamma^2 e + e)_+ + (-A^2 w^1 + \gamma^1 e + e)_+) \ = \ 0 \end{array} \tag{7.12}$$

To reduce the above definition of bilinear separability to a bilinear program, we make use of the following simple lemma.

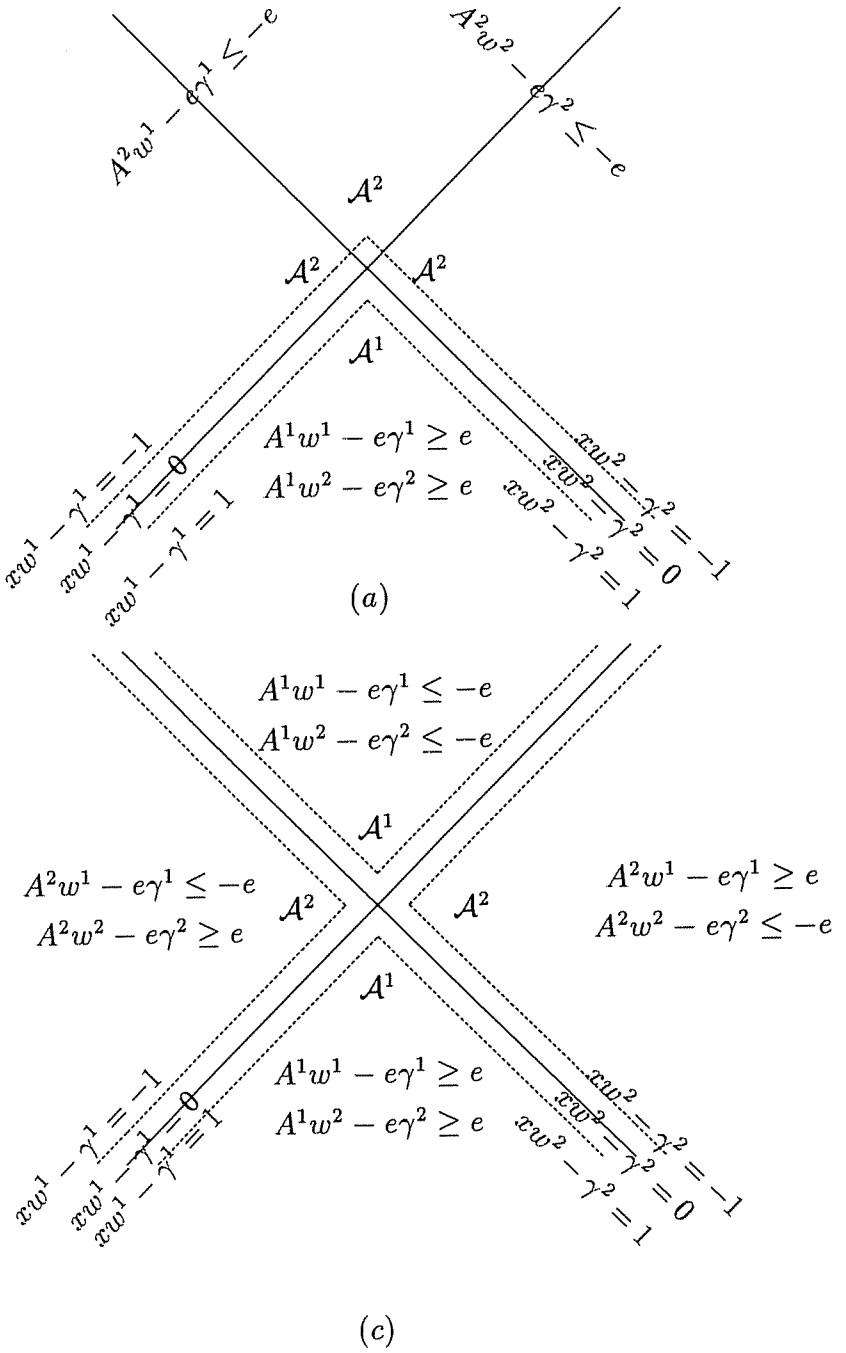


Figure 7.5: Geometric representation of bilinear separability definition

Lemma 7.3.1 *Let* $\mathcal{T}^i \subset R^m$, $i = 1, \ldots, 4$. *Then*

$$\left\langle \begin{array}{c} t^i \in \mathcal{T}^i,\ i = 1, \ldots, 4 \\ (t^1_+ + t^2_+)(t^4_+ + t^3_+) = 0 \end{array} \right\rangle \Leftrightarrow \left\langle \begin{array}{ccc} 0 = & \min\limits_{y^i, t^i, i=1,\ldots,4} & (y^1 + y^2)(y^4 + y^3) \\ & such\ that & t^i \in \mathcal{T}^i,\ t^i \leq y^i,\ 0 \leq y^i \\ & & i = 1, \ldots, 4 \end{array} \right\rangle \tag{7.13}$$

Proof. ($\Rightarrow$) Let $\hat{t}^i$, $i = 1, \ldots, 4$ solve the first problem. Define $\hat{y}^i := \hat{t}^i_+ \geq \hat{t}^i$, $i = 1, \ldots, 4$. Then $\hat{y}^i \geq \hat{t}^i$, $\hat{y}^i \geq 0$, $\hat{t}^i \in \mathcal{T}^i$, for $i = 1, \ldots, 4$. Hence $(\hat{y}^i, \hat{t}^i)$, $i = 1, \ldots, 4$, is feasible for the second problem, and is optimal because

$$(\hat{y}^1_+ + \hat{y}^2_+)(\hat{y}^4_+ + \hat{y}^3_+) = (\hat{t}^1_+ + \hat{t}^2_+)(\hat{t}^4_+ + \hat{t}^3_+) = 0.$$

($\Leftarrow$) Let $(\hat{y}^i, \hat{t}^i)$, $i = 1, \ldots, 4$, solve the second problem. Then

$$(\hat{y}^1_j + \hat{y}^2_j) = 0 \text{ or } (\hat{y}^4_j + \hat{y}^3_j) = 0,\ j = 1, \ldots, m$$

Consequently

$$(\hat{t}^1_j \leq 0 \text{ and } \hat{t}^2_j \leq 0) \text{ or } (\hat{t}^4_j \leq 0 \text{ and } \hat{t}^3_j \leq 0),\ j = 1, \ldots, m$$

Hence

$$(\hat{t}^1_{j+} + \hat{t}^2_{j+}) = 0 \text{ or } (\hat{t}^4_{j+} + \hat{t}^3_{j+}) = 0,\ j = 1, \ldots, m$$

and

$$(\hat{t}^1_+ + \hat{t}^2_+)(\hat{t}^4_+ + \hat{t}^3_+) = 0.$$

□

We now formulate the alternate bilinear separability definition as a bilinear program with the help of this lemma.

Theorem 7.3.1 ***(Equivalence of bilinear separability to bilinear programming)*** *The sets* $\mathcal{A}^1$ *and* $\mathcal{A}^2$ *are bilinearly separable if and only if at least one of the following bilinear programs has a zero minimum in which case a minimum solution* $(\hat{w}^1, \hat{w}^2, \hat{\gamma}^1, \hat{\gamma}^2)$ *determines the separating planes* $\hat{w}^1 x = \hat{\gamma}^1$ *and* $\hat{w}^2 x = \hat{\gamma}^2$*:*

$$
\begin{array}{lll}
\textit{minimize} & z^1 z^2 & \\
z^1, z^2, w^1, w^2, \gamma^1, \gamma^2 & & \\
\textit{such that} & -A^1 w^1 + \gamma^1 e + e \leq 0, & -A^1 w^2 + \gamma^2 e + e \leq 0 \\
& A^2 w^1 - \gamma^1 e + e \leq z^1, & A^2 w^2 - \gamma^2 e + e \leq z^2 \\
& 0 \leq z^1 & 0 \leq z^2
\end{array} \tag{7.14}
$$

$$
\begin{array}{lll}
\textit{minimize} & y^1 y^2 & \\
y^1, y^2, w^1, w^2, \gamma^1, \gamma^2 & & \\
\textit{such that} & -A^2 w^1 + \gamma^1 e + e \leq 0, & -A^2 w^2 + \gamma^2 e + e \leq 0 \\
& A^1 w^1 - \gamma^1 e + e \leq y^1, & A^1 w^2 - \gamma^2 e + e \leq y^2 \\
& 0 \leq y^1 & 0 \leq y^2
\end{array} \tag{7.15}
$$

$$
\begin{array}{lll}
\textit{minimize} & (y^1 + y^2)(y^4 + y^3) \; + \; (z^1 + z^2)(z^4 + z^3) & \\
y^1, y^2, y^3, y^4, z^1, z^2, z^3, z^4, & & \\
w^1, w^2, \gamma^1, \gamma^2 & & \\
\textit{such that} & -A^1 w^1 + \gamma^1 e + e \leq y^1, & -A^1 w^2 + \gamma^2 e + e \leq y^2 \\
& 0 \leq y^1 & 0 \leq y^2 \\
& A^1 w^1 - \gamma^1 e + e \leq y^3, & A^1 w^2 - \gamma^2 e + e \leq y^4 \\
& 0 \leq y^3 & 0 \leq y^4 \\
& A^2 w^1 - \gamma^1 e + e \leq z^1, & -A^2 w^2 + \gamma^2 e + e \leq z^2 \\
& 0 \leq z^1 & 0 \leq z^2 \\
& -A^2 w^1 + \gamma^1 e + e \leq z^3, & A^2 w^2 - \gamma^2 e + e \leq z^4 \\
& 0 \leq z^3 & 0 \leq z^4
\end{array} \tag{7.16}
$$

Proof. Use Lemma 7.3.1 on Definition 7.3.2. □

Remark 7.3.1 *Note that there is a key difference between problems (7.14)-(7.15) and (7.16) above. The former have uncoupled constraints in* (z^1, w^1, γ^1) *and*

(z^2, w^2, γ^2), *and in* (y^1, w^1, γ^1) *and* (y^2, w^2, γ^2), *while (7.16) does not. This allows us to use the simplex method and the finitely terminating UBPA 7.2.1 and UBPA1 7.2.2 on (7.14)-(7.15).*

Remark 7.3.2 *In order to apply the results of Section 7.2, all the variables in the above bilinear programs need to be nonnegative. This is easily achieved by adding two nonnegative one-dimensional variables* ζ^1, ζ^2 *to the problem and defining* $w^1 = \hat{w}^1 - e\zeta^1$, $w^2 = \hat{w}^2 - e\zeta^2$, $\gamma^1 = \hat{\gamma}^1 - \zeta^1$, $\gamma^2 = \hat{\gamma}^2 - \zeta^2$, $(\hat{w}^1, \hat{w}^2, \hat{\gamma}^1, \hat{\gamma}^2, \zeta^1, \zeta^2) \geq 0$.

7.4 Computational results

Both the uncoupled and coupled bilinear programming algorithms were implemented and tested on a suite of problems. The linear programming package MINOS 5.4 [58] was used to solve the linear subproblems. Initial experimentation with UBPA 7.2.1, UBPA1 7.2.2, and BPA 7.2.3 showed that they were prone to halting at solutions with $w^1 = 0$, $w^2 = 0$, or $w^1 = w^2$ which satisfied the minimum principle necessary optimality condition. These are clearly undesirable solutions and are easily detected in practice. The $w^1 = w^2$ case was avoided by starting with a good initial solution found by using the first two planes of the MSMT algorithm [6, 10]. This consists of obtaining the best linear programming split [10] using a single plane and then splitting one of the two resulting half-spaces (the one with the greatest percentage of misclassified points) with a second plane. The problem of $w^1 = 0$ and $w^2 = 0$ was solved by detecting when this occurred and then adding a linear constraint to the problem which made the zero solution infeasible. This did not affect the convergence proof of the algorithm. Care must be taken to avoid excluding the optimal solution.

We tested the algorithms on two-dimensional toy problems and on randomly generated problems in high dimensions. Figure 7.6 shows a solution found by the UBPA1 7.2.2 on a two-dimensional problem with 65 points in $\mathcal{A}^1$ and 102 points $\mathcal{A}^2$. The algorithm was able to completely separate the two sets in 157 simplex

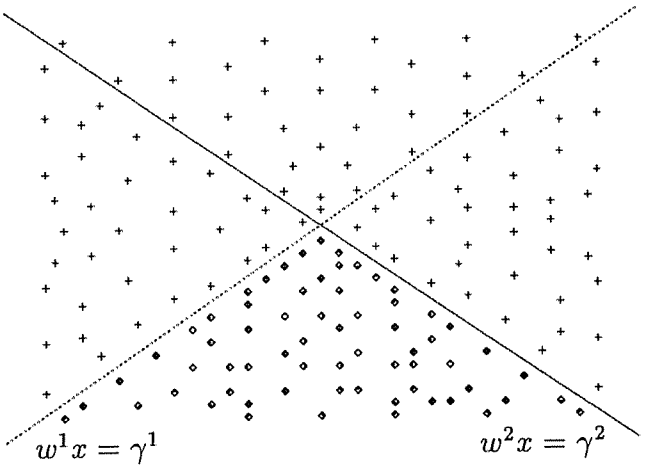


Figure 7.6: Bilinear separation found by UBPA1

pivots. The second two-dimensional problem, consisting of 130 points in $\mathcal{A}^1$ and 74 points in $\mathcal{A}^2$, is depicted in Figure 7.7. BPA 7.2.3 found the bilinear separation in 829 simplex pivots.

In order to investigate the effectiveness of the linear programming approach on high-dimensional bilinearly separable problems, we randomly generated test problems as follows [4]. We generated two random planes and determined two sets that are bilinearly separable by these two planes. We generated points in both sets. A subset of these points, the "training set", was used in either Algorithm 2.1 or 2.3 to obtain two planes (not necessarily the ones originally generated) that bilinearly separate the points in the training set. The remaining points were used as a "testing set" to determine how close the two planes were to the original two planes which determine the bilinearly separable sets.

More specifically, two points were randomly generated on an n-dimensional unit sphere. These points were used as the normals w^1 and w^2 for the two separating planes $w^1x = 0$ and $w^2x = 0$ with the topology of either of Figure 7.1(a)

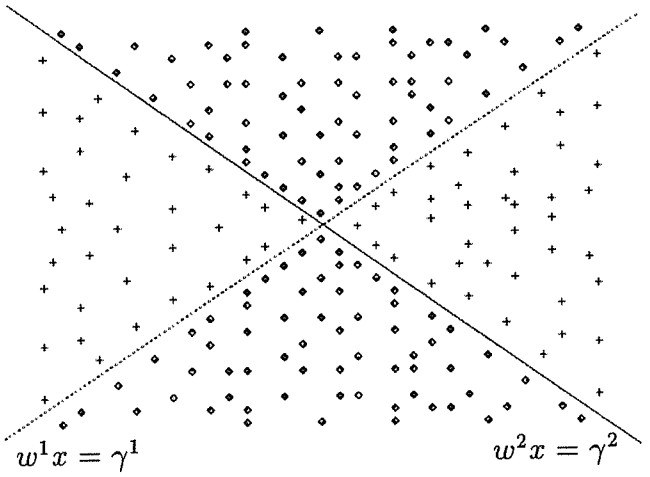


Figure 7.7: Bilinear Separation Found by BPA

or Figure 7.1(c). Then a training set of l randomly generated points on the n-dimensional unit sphere was generated and classified using the two planes. Similarly a testing set of consisting of 5000 points was also generated. The bilinear algorithm appropriate for the topology was run using the points in the training set, and then used to classify the points in the testing set. For the uncoupled problems, we performed these experiments for 10, 25, 50, and 100 dimensional problems using 500 and 1000 training points. For the coupled problems, we performed these experiments for 5, 10, and 25 dimensional problems using 500 and 1000 training points. The results are summarized in Tables 7.1 and 7.2 and in Figures 7.8 and 7.9.

Table 7.1 summarizes the training set performance on the uncoupled problems using UBPA 7.2.1. Table 7.2 summarizes the training set performance on the coupled problems using BPA 7.2.3. The algorithms correctly determined that the sets were bilinearly separable in every case. Note that we used the fact that $\gamma^1 = \gamma^2 = 0$ at the solution to constrain the algorithm when $w^1 = 0$ or $w^2 = 0$

Table 7.1: Performance of uncoupled bilinear programming algorithm UBPA on training set

Dimension n	Training Set Size	100% Bilinear Separation
10	500	10 of 10
10	1000	10 of 10
25	500	10 of 10
25	1000	10 of 10
50	500	10 of 10
50	1000	10 of 10
100	500	10 of 10
100	1000	10 of 10
100	1000	10 of 10

Table 7.2: Performance of coupled bilinear programming algorithm BPA on training set

Dimension n	Training Set Size	100% Bilinear Separation
5	500	10 of 10
10	500	10 of 10
10	1000	10 of 10
25	500	10 of 10
25	1000	10 of 10

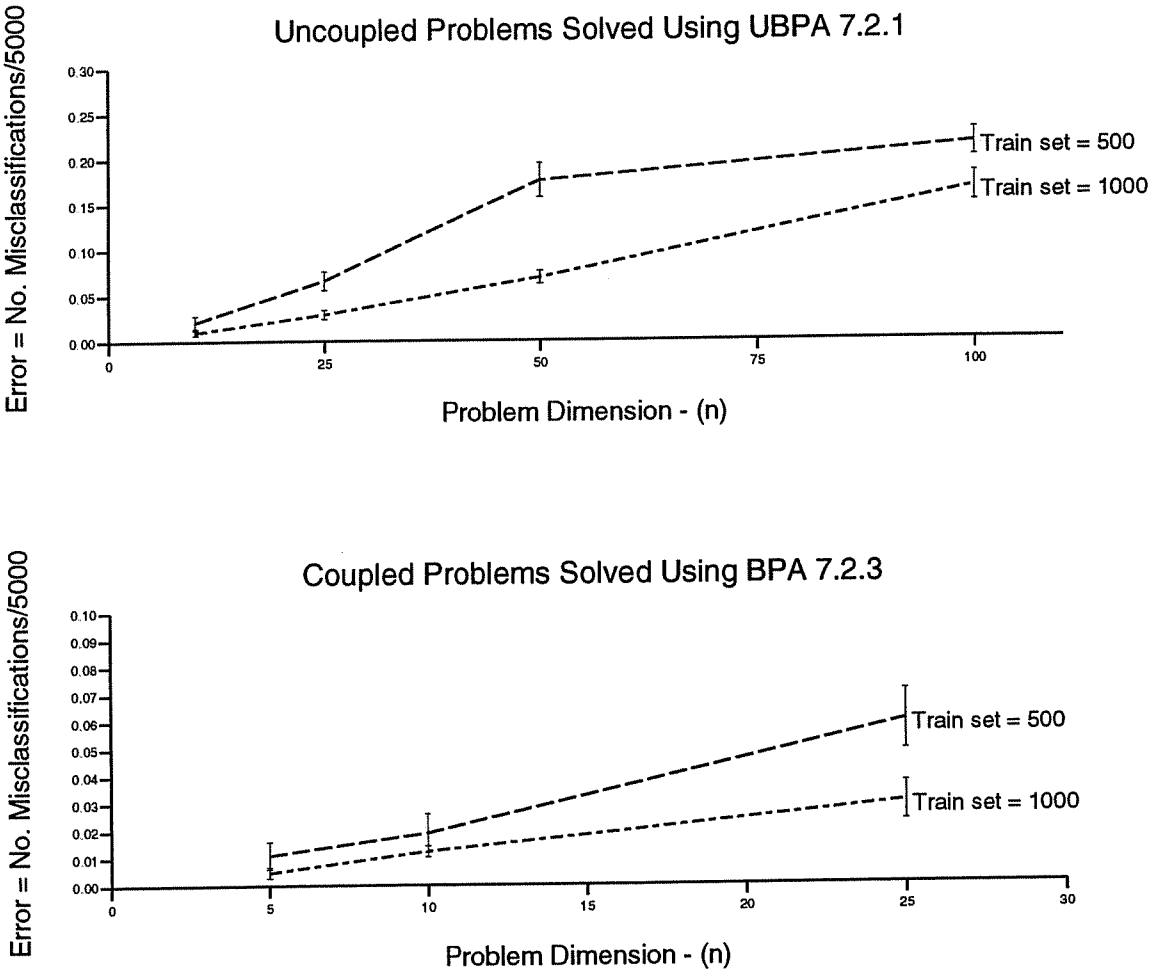


Figure 7.8: Average testing set error (testing set = 5000, 10 trials)

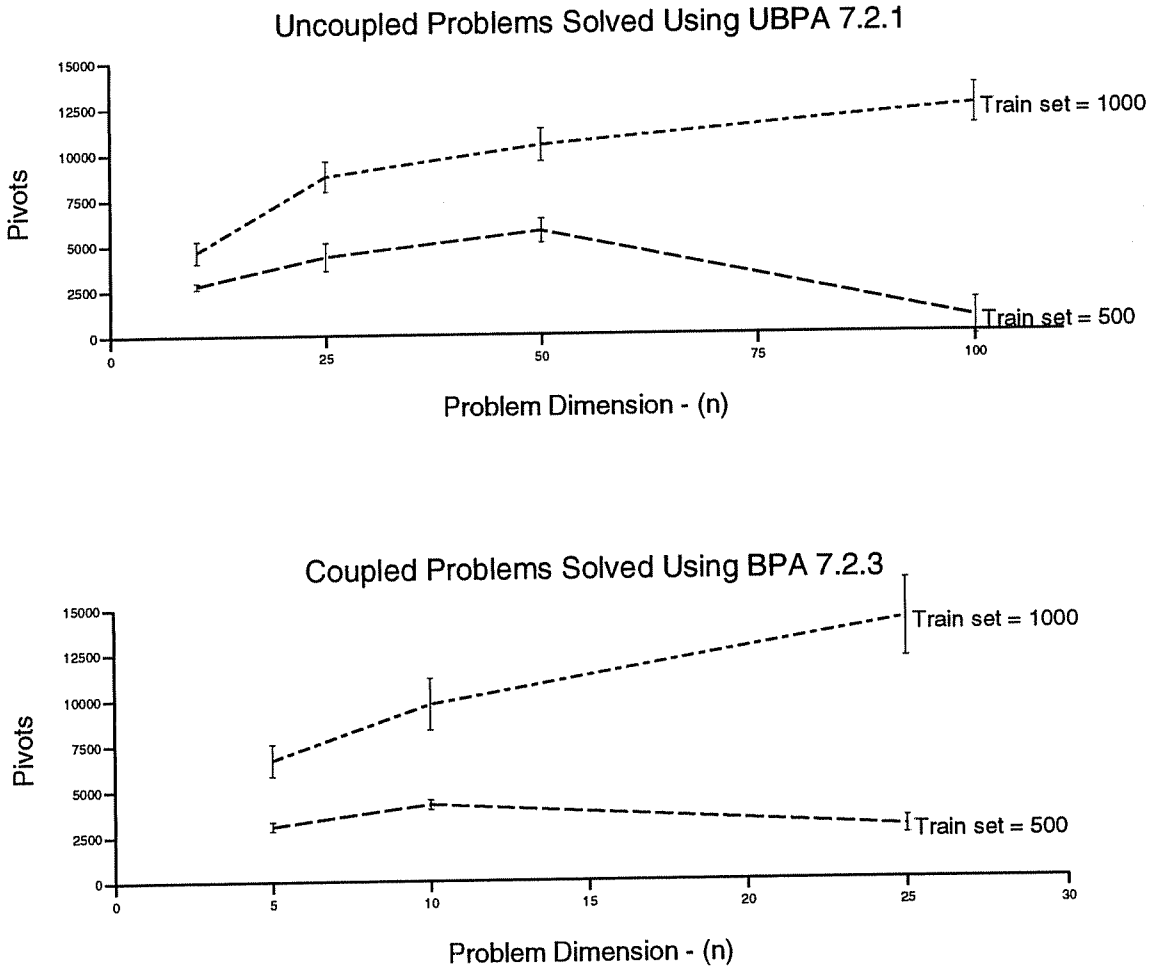


Figure 7.9: Average computational cost (10 trials)

was encountered. However, even with $\gamma^1 = \gamma^2 = 0$, the underlying problem is still NP-complete.

Figure 7.8 depicts the performance of the bilinear separation on an unseen testing set made up of 5000 points. The curves in this figure (as well as those in Figure 7.9) are the average of 10 runs, and are marked with 95% confidence intervals. Curves for two bilinear separations are given: one trained on a 500-point set and one on a 1000-point set. These curves indicate that the trained bilinear separation is able to learn the underlying structure quite accurately for small dimensional problems, as seen from the small errors for these problems. But as the problem dimension increases, the error increases. However, if more training examples are used the error decreases. This is indicated by lower curves in Figure 7.8 for the 1000-point training set compared to the curves for the 500-point training set. These trends agree with computational learning theory results [5] that provide necessary and sufficient conditions for valid generalization (i.e. correctness on testing sets) which are dependent on the problem dimension and the number of points in the training set.

To measure how well the algorithms scale as the dimension grows we examined the average number of total simplex pivots to solve each problem. The results are reported in Figure 7.9 for the uncoupled and coupled algorithms UBPA 7.2.1 and BPA 7.2.3. The computational cost of the coupled algorithm is considerably higher than that of the uncoupled algorithm. Not only does the coupled algorithm require more pivots, but each pivot takes longer since the number of rows in Problem (7.16) to which the coupled algorithm BPA 7.2.3 is applied is twice as many as the number of rows in Problem (7.14) to which the uncoupled algorithm UBPA 7.2.1 is applied. Also the number of columns in problem (7.16) is approximately four times the number of columns in Problem (7.14). Thus the more demanding problems solved by the coupled algorithm BPA 7.2.3 were of smaller maximum dimension, 25 instead of 100, than the problems solved by the uncoupled algorithm UBPA 7.2.1. We did successfully solve with the coupled algorithm problems with as many 1000 points in R^{100}, but did not run 10 such cases to average and report.

Note that in Figure 7.9 the number of pivots required for the uncoupled algorithm UBPA 7.2.1 for a training set size of 500 in 100 dimensions is considerably lower than the corresponding number for the 50-dimensional case. This is because 500 is not a sufficient number of training set examples to adequately represent a problem in a 100-dimensional space. Nine out of ten generated training sets were linearly separable, and were quickly separated linearly by the MSMT algorithm [6, 10].

We also compared the performance of UBPA 7.2.1 and BPA 7.2.3 with that of the back-propagation (BP) algorithm [70], the standard algorithm for training neural networks which in general can only guarantee a stationary point. BP was used to train neural networks configured as in Figure 7.3(b) and Figure 7.4(b) to solve the bilinear separability problem. We found that BP could consistently solve only small dimensional problems with the topology of Figure 7.1(a). Specifically BP completely separated five out of five such 5-dimensional problems. However BP solved only one out of five problems in 5-dimensional space with the topology of Figure 7.1(c). Also BP failed to solve higher dimensional problems for both topologies within 40,000 epochs. Unlike UBPA 7.2.1 and BPA 7.2.3, BP has no well defined stopping criterion and hence may have failed because it was not given sufficient processing time. We found BP to be computationally more costly than the bilinear programming approach for all but small problems. For example, in the 10-dimensional case with the the topology of Figure 7.1(c) and 500 training examples, BP failed after an average of 8204 seconds, while BPA correctly solved this problem in 691 seconds on average.

7.5 Strengths

The NP-complete bilinear separation problem was posed as two bilinear programs. These bilinear programs were shown to have vertex solutions for bilinearly separable problems. This property led to linear-programming-based algorithms with finite termination. Computational experiments indicated the viability of these algorithms for problems of up to 100 dimensions and 1000 points. An important

computational achievement to point out is the fact that all of the 140 cases of the NP-complete problems, represented in Tables 7.1 and 7.2, were solved correctly without a single failure.

7.6 Extensions

We suggest two natural extensions to the bilinear programming approach to bilinear separation proposed in this chapter. The first would be to adapt this method to k-class discrimination problems. The problem of whether k disjoint classes in R^n can be strictly separated by h planes is also NP-complete since it contains bilinear separability as a special case. There are up to $k\sum_{i=0}^{n}\binom{h}{i}$ [30] different possible class labelings of the multisurface determined by the h planes. Unlike the bilinear separability case, it is probably not tractable to consider all possible permutations of the problem. It is possible to formulate the problem for a given permutation of the separate k classes with h planes problem. For example, consider the problem of the given sets $\mathcal{A}^1$ and $\mathcal{A}^2$. Does there exist a multisurface with the structure shown in Figure 1.2 which strictly separates the sets? This last problem can be formulated as a minimization problem. We conjecture that the new minimization problem, which is multilinear, will have vertex solutions in the separable case, and results similar to those of Appendix B will apply. The details and proofs of this last approach are left for future work.

The second extension would be to use the bilinear separability algorithm and its extension to k-class problems as a subproblem in a decision-tree algorithm. First, we would assess the performance of the algorithm on inseparable problems to see if this is worthwhile. Second, we would investigate parallelizing the algorithm because the bilinear algorithm is not fast computationally and extensions to more complex k-class problems are bound to be slower. The problem does have special structure so this could lead to some interesting parallel algorithms.

Chapter 8

Conclusions

We have proposed a family of algorithms for inductive machine learning based on mathematical programming. These algorithms produce multisurfaces that can be viewed as neural networks as well as decision trees. In each case we formalized the desired relationships as a system of linear inequalities, and then minimized the errors in these inequalities. For two-category discrimination, we proposed a new robust linear program for constructing linear discriminants. The proposed approach was computationally superior to other linear programming approaches. The multisurface method tree (MSMT) algorithm, which utilized the robust linear programming as a subproblem, compared favorably with other decision tree algorithms. We then generalized the two-class linear programming to k-class problems. The resulting error-minimization problem can be formulated as a linear program or a piecewise-quadratic minimization problem. We were able to decrease computational time without sacrificing quality of the solution by parallelizing the piecewise-linear separability problem using a new parallel gradient distribution algorithm. For bilinear separation, we successfully applied a novel bilinear programming algorithm that exploits the structure of the problem to 140 consecutive NP-complete problems. The new bilinear programming and parallel gradient distribution algorithms may be useful for solving other mathematical programming

problems. We have outlined possible extensions to these approaches throughout the thesis. Computational results demonstrate the viability of the proposed methods.

Bibliography

[1] S. Aeberhard, D. Coomans, and O. de Vel. Comparison of classifiers in high dimensional settings. Technical Report 92-02, Departments of Computer Science and of Mathematics and Statistics, James Cook University of North Queensland, 1992.

[2] F.A. Al-Khayyal and J.E. Falk. Jointly constrained biconvex programming. *Mathematics of Operations Research*, 8(2):273–286, 1983.

[3] R. S. Barr and T. F. Siems. Private communication to O. L. Mangasarian, 1990.

[4] E.B. Baum. Private communication to O. L. Mangasarian. NEC Research Institute, Princeton, NJ 08540, 1992.

[5] E.B. Baum and D. Haussler. What size net gives valid generalization. In D.S. Touretzky, editor, *Advances in Neural Information Processing Systems I*, pages 81–90, San Mateo, California, 1989. Morgan Kaufmann.

[6] K. P. Bennett. Decision tree construction via linear programming. In M. Evans, editor, *Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society Conference*, pages 97–101, Illinois Institute of Technology, Chicago, 1992. Midwest Artificial and Cognitive Science Society.

[7] K. P. Bennett and O. L. Mangasarian. Bilinear separation of two sets in n-space. Computer Sciences Department Technical Report 1109, University

of Wisconsin, Madison, Wisconsin, 1992. To appear in *Computational Optimization and Applications.*

[8] K. P. Bennett and O. L. Mangasarian. Multicategory separation via linear programming. Computer Sciences Department Technical Report 1127, University of Wisconsin, Madison, Wisconsin, 1992. To appear in *Optimization Methods and Software.*

[9] K. P. Bennett and O. L. Mangasarian. Neural network training via linear programming. In P. M. Pardalos, editor, *Advances in Optimization and Parallel Computing*, pages 56–67, Amsterdam, 1992. North Holland.

[10] K. P. Bennett and O. L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1:23–34, 1992.

[11] C. Berge and A. Ghouila-Houri. *Programming, Games and Transportation Networks.* Wiley, New York, 1965.

[12] H.D. Block and S.A. Levin. On the boundedness of an iterative procedure for solving a system of linear inequalities. *Proceedings of the American Mathematical Society*, 26:229–235, 1970.

[13] A. Blum and R.L. Rivest. Training a 3-node neural network is NP-complete. In D.S. Touretzky, editor, *Advances in Neural Information Processing Systems I*, pages 494–501, San Mateo, California, 1989. Morgan Kaufmann.

[14] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees.* Wadsworth International, Califormna, 1984.

[15] C. E. Brodley and P. E. Utgoff. Multivariate decision trees. COINS Technical Report 92-83, University of Massachussets, Amherst, Massachusetts, 1992. To appear in *Machine Learning.*

[16] A. Charnes. Some fundamental theorems of perceptron theory and their geometry. In J. T. Lou and R. H. Wilcox, editors, *Computer and Information Sciences*, pages 67–74, Washington, 1964. Spartan Books.

[17] G.B. Dantzig. *Linear Programming and Extensions.* Princeton University Press, Princeton, New Jersey, 1963.

[18] R. Detrano, A. Janosi, W. Steinbrunn, M. Pfisterer, J. Schmid, S. Sandhu, K. Guppy, S. Lee, and V. Froelicher. International application of a new probability algorithm for the diagnosis of coronary artery disease. *American Journal of Cardiology*, 64:304–310, 1989.

[19] R. O. Duda and H. Fossum. Pattern classification by iteratively determined linear and piecewise linear discriminant functions. *IEEE Transactions on Electronic Computers*, 15:220–232, 1966.

[20] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* John Wiley & Sons, New York, 1973.

[21] I. W. Evett and E.J. Spiehler. Rule induction in forensic science. Technical report, Central Research Establishment, Home Office Forensic Science Service, Aldermaston, Reading, Berkshire RG7 4PN, 1987.

[22] S.E. Fahlman and C. Lebiere. The cascade-correlation learning architecture. In D.S. Touretzky, editor, *Advances in Neural Information Processing Systems II*, pages 524–532, San Mateo, California, 1990. Morgan Kaufmann.

[23] M. C. Ferris and O. L. Mangasarian. Parallel variable distribution. Symposium on Parallel Optimization 3, Madison, Wisconsin, "July 7-9," 1993.

[24] R.A. Fisher. The use of multiple measurements in taxonomic problems. *Annual Eugenics*, 7(Part II):179–188, 1936.

[25] R. Fletcher. *Practical Methods of Optimization*, volume 2. John Wiley & Sons, Chichester, 1981.

[26] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3:95–110, 1956.

[27] M. Frean. The upstart algorithm: A method for constructing and training feedforward neural networks. *Neural Computation*, 2:198–209, 1990.

[28] K. Fukunaga. *Statistical Pattern Recognition.* Academic Press, New York, 1990.

[29] S. Gallant. Optimal linear discriminants. In *Proceedings of the International Conference on Pattern Recognition*, pages 849–852. IEEE Computer Society Press, 1986.

[30] G.M. Georgiou. Comments on hidden nodes in neural nets. *IEEE Transactions on Circuits and Systems*, 38:1410, 1991.

[31] F. Glover. Improved linear programming models for discriminant analysis. *Decision Sciences*, 21:771–785, 1990.

[32] R.C. Grinold. Mathematical methods for pattern classification. *Management Science*, 19:272–289, 1972.

[33] David Heath. *A geometric framework for machine learning.* PhD thesis, The John Hopkins University, 1992.

[34] J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley, Redwood City, California, 1991.

[35] W. H. Highleyman. A note on linear separation. *IEEE Transactions on Electronic Computers*, 10:777–778, 1961.

[36] N. Karmarkar. A new polynomial time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[37] L.G. Khachijan. A polynomial algorithm for linear programming. *Soviet Mathematics Doklady*, 20:191–194, 1979.

[38] H. Konno and T. Kuno. Linear multiplicative programming. *Mathematical Programming*, 56:51–64, 1992.

[39] P. A. Lachenbruch and R. M. Mickey. Estimation of error rates in discriminant analysis. *Technometrics*, 10:1–11, 1968.

[40] P.A. Lahenbruch and R.M. Mickey. Estimation of error rates in discriminant analysis. *Technometrics*, 10:1–11, 1968.

[41] K. Lang, A. Waibel, and G. Hinton. A time-delay neural network architecture for isolated word recognition. *Neural Networks*, 3:23–43, 1990.

[42] W.-Y. Loh and N. Vanichsetakul. Tree-structured classification via generalized discriminant analysis (with discussion). *Journal of the American Statistical Association*, 83:715–728, 1988.

[43] O. L. Mangasarian. Linear and nonlinear separation of patterns by linear programming. *Operations Research*, 13:444–452, 1965.

[44] O. L. Mangasarian. Multi-surface method of pattern separation. *IEEE Transactions on Information Theory*, IT-14:801–807, 1968.

[45] O. L. Mangasarian. *Nonlinear Programming.* McGraw–Hill, New York, 1969.

[46] O. L. Mangasarian. Characterization of linear complementarity problems as linear programs. *Mathematical Programming Study*, 7:74–87, 1978.

[47] O. L. Mangasarian. Parallel gradient distribution in unconstrained optimization. Computer Sciences Technical Report 1145, University of Wisconsin, Madison, Wisconsin 53706, 1993.

[48] O.L. Mangasarian, R. Setiono, and W.H. Wolberg. Pattern recognition via linear programming: Theory and application to medical diagnosis. In T. F. Coleman and Y. Li, editors, *Proceedings of the Workshop on Large-Scale*

Numerical Optimization, Cornell University, Ithaca, New York, October 19-20, 1989, pages 22–31, Philadelphia, Pennsylvania, 1990. SIAM.

[49] O.L. Mangasarian and W. H. Wolberg. Cancer diagnosis via linear programming. *SIAM News*, 23:1 & 18, 1990.

[50] M. Marchand, M. Golean, and P. Rujan. A convergence theorem for sequential learning in two layer perceptrons. *Europhysics Letters*, 11:487–492, 1990.

[51] J.L. McClelland and D.E. Rummelhart. *Explorations in Parallel Distributed Processing: A Handbook of Models, Programs, and Exercises.* MIT Press, Cambridge, Massachusetts, 1987.

[52] W. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 7:115–133, 1943.

[53] N. Megiddo. On the complexity of polyhedral separability. *Discrete and Computational Geometry*, 3:325–337, 1988.

[54] M. Mezard and J.P. Nadal. Learning in feedforward layered networks: The tiling algorithm. *Journal of Physics*, 22:2191–2294, 1989.

[55] M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry.* MIT Press, Cambridge, Massachusetts, 1969.

[56] T. S. Motzkin and I. J. Schoenberg. The relaxation method for linear inequalities. *Canadian Journal of Mathematics*, 6:393–404, 1954.

[57] P.M. Murphy and D.W. Aha. UCI repository of machine learning databases. Department of Information and Computer Science, University of California, Irvine, California, 1992.

[58] B.A. Murtagh and M.A. Saunders. MINOS 5.0 user's guide. Technical Report SOL 83.20, Stanford University, December 1983.

[59] K.G. Murty. *Linear Programming.* John Wiley & Sons, New York, New York, 1983.

[60] N. J. Nilsson. *Learning Machines.* McGraw Hill, New York, 1965.

[61] P. M. Pardalos. Polynomial time algorithm for some classes of constrained non-convex quadratic problems. *Optimization*, 21:843–853, 1990.

[62] B.T. Polyak. *Introduction to Optimization.* Optimization Software, Inc., Publications Division, New York, 1987.

[63] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1984.

[64] J. R. Quinlan. Decision trees as probabilistic classifiers. In *Proceedings of Fourth International Workshop on Machine Learning*, Los Altos, CA, 1987. Morgan Kaufmann.

[65] J. R. Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27(1), 1987.

[66] R. T. Rockafellar. *Convex Analysis.* Princeton University Press, Princeton, New Jersey, 1970.

[67] F. Rosenblatt. Two theorems of statistical separability in the perceptron. In *Proceedings of Symposium Held at National Physical Laboratory, November 1958*, volume 1, London, 1959. HMS Stationery Office.

[68] F. Rosenblatt. The perceptron–a perceiving and recognizing automaton. Technical Report 85-460-1, Cornell Aeronautical Laboratory, Itahca, New York, January 1957.

[69] A. Roy, L. S. Kim, and S. Mukhopadhyay. A polynomial time algorithm for the construction and training of a class of multilayer perceptrons. *Neural Networks*, 6:535–545, 1993.

[70] D.E. Rumelhart, G.E. Hinton, and J.L. McClelland. Learning internal representations. In D.E. Rumelhart and J.L. McClelland, editors, *Parallel Distributed Processing*, pages 318–362, Cambridge, Massachusetts, 1986. MIT Press.

[71] A. Sankar and R. J. Mammone. Speaker independent vowel recognition using neural tree networks. In *Proceedings of IJCNN*, 1992.

[72] J. A. Sirat and J. P. Nadal. Neural trees: A new tool for classification. Preprint, Laboratoires d'Electronique Philips, Limeil-Brevannes, France, 1990.

[73] F. W. Smith. Pattern classifier design by linear programming. *IEEE Transactions on Computers*, C-17:367–372, 1968.

[74] F. W. Smith. Design of multicategory pattern classifiers with two-category classifier design procedures. *IEEE Transactions on Computers*, 18:367–372, 1969.

[75] T. V. Thieu. A note on the solution of bilinear programming problems by reduction to concave minimization. *Mathematical Programming*, 41:249–260, 1988.

[76] P. E. Utgoff. Perceptron trees: A case study in hybrid concept representations. *Connection Science*, 1(4):377–391, 1989.

[77] P. E. Utgoff and C. E. Brodley. An incremental method for finding multivariate splits for decision trees. In *Proceedings of the Seventh International Conference on Machine Learning*, pages 58–65, Los Altos, CA, 1990. Morgan Kaufmann.

[78] P.J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behaviorial Sciences.* PhD thesis, Harvard University, 1974.

[79] D. J. White. A linear programming approach to solving bilinear programs. *Mathematical Programming*, 56:45–50, 1992.

[80] W.N.Street, W.H. Wolberg, and O.L. Mangasarian. Nuclear feature extraction for breast tumor diagnosis. In *IS&T/SPIE 1993 International Symposium on Electronic Imaging: Science and Technology*, volume 105, San Jose, California, 1993.

[81] W. H. Wolberg and O. L. Mangasarian. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. *Proceedings of the National Academy of Sciences U.S.A.*, 87:9193–9196, 1990.

[82] W. H. Wolberg and O.L. Mangasarian. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. *Proceedings of the National Academy of Sciences, U.S.A.*, 87:9193–9196, 1990.

[83] W. H. Wolberg, M. S. Tanner, and W. Y. Loh. Diagnostic schemes for fine needle aspirates of breast masses. *Analytical and Quantitative Cytology and Histology*, 10:225–228, 1988.

[84] W.H. Wolberg, K.P. Bennett, and O.L. Mangasarian. Breast cancer diagnosis and prognostic determination from cell analysis. Manuscript, Departments of Surgery and Human Oncology and Computer Sciences, University of Wisconsin, Madison, WI 53706, 1992.

[85] Y. Yajima and H. Konno. Efficient algorithms for solving rank two and rank three bilinear programming problems. *Journal of Global Optimization*, 1:155–171, 1991.

Appendix A

Descriptions of datasets

This appendix describes the real-world datasets used for the computational experiments in this thesis. For quick reference, Table A.1 provides the numbers of dimensions, classes, and total points for each dataset. These are listed in alphabetical order and described briefly. All of these datasets except the Bank data are available via anonymous ftp (file transfer protocol) from the University of California Irvine (UCI) Repository of Machine Learning Databases [57].

- **Bank Failure:** This data, which was collected by Richard S. Barr of Southern Methodist University and Thomas F. Siems of the Federal Reserve Bank of Dallas [3], has 9 numeric features which range from 0 to 1. The data represent 4311 successful banks and 441 failed banks.

- **Cleveland Heart Disease:** The Cleveland Heart Disease Database [18] consists of 197 points in a 13-dimensional real space, of which 137 are negative and 60 are positive. Categorical features within this database were converted to ordered integers.

- **Glass Identification Database:** In the glass identification data [21], the chemical analysis of forensic glass is used to determine the origin of the glass. It consists of 214 points in R^9 which represent six different types of

Dataset	Dimension	Classes	Total Points
Bank	9	2	4752
Cleveland	13	2	197
Glass	9	7	214
Image-s	19	7	210
Image-u	19	7	2310
Iris	4	3	150
WBCD	9	2	566
Wine	13	3	178

Table A.1: Description of datasets used in experiments

glass. The distribution of examples in classes is as follows: float-processed-building-windows 87 points, float-processed-vehicle-windows 17, non-float-processed-building-windows 76, containers 13, tableware 9, and headlamps 29. Note that this dataset is poorly scaled so the data was normalized.

- **Image-s and Image-u:** In the image segmentation problem [15], nineteen low-level real-valued image features are used to determine the image segment: sky, cement, window, brick, grass, foliage, or path. This image data was generated by the Vision Group at the University of Massachusetts. The image data is divided into two parts: a training set consisting of 210 points and a testing set consisting 2310 points. We refer to the set of 210 points as Image-s since it is piecewise-linear separable, and the set of 2310 points as Image-u since it is not piecewise-linear separable.

- **Iris Identification:** Fisher's classical Iris identification problem [24], referred to as Iris, uses physical attributes of Iris blossoms to determine the type of Iris. It consists of 150 points in R^4 which belong to three classes (50 points in each of the classes).

- **WBCD: Wisconsin Breast Cancer Data:** This database [48, 49, 82] consists of 566 points of which 354 are benign and 212 are malignant, all in a 9-dimensional real space. All of the attributes take on a value of 0 to 10 where 0 indicates a missing value. The results in Chapter 2 utilize only 487 of the points because these were all that were available at the time the experiments were conducted.

- **Wine recognition data:** The wine recognition data [1], referred to as wine, uses the chemical analysis of wine to determine the cultivar. Of the 178 points in R^{13}, 59 points are in class 1, 71 points are in class 2, and 48 points are in class 3.

Appendix B

Frank-Wolfe algorithm for nonconvex and bilinear programs

For convenience and completeness, we give simple convergence proofs for a Frank-Wolfe [26] algorithm without any convexity assumptions. We also establish finite termination when the sequence of points generated by the algorithm tends to a zero minimum. These results, which do not seem to be readily available in the stated form, are needed for our bilinear program algorithm BPA 7.2.3. Our proofs are based on those of [11] for the convex case.

We consider the following problem and assumptions.

Problem B.0.1

$$\min_{x \in \mathcal{X}} f(x)$$

where $f : R^n \rightarrow R$, $\mathcal{X}$ *is a polyhedral set in* R^n *that does not contain straight lines that go to infinity in both directions (e.g. it contains the constraint* $x \geq 0$*),* f *has continuous first partial derivatives on* $\mathcal{X}$ *and* f *is bounded below on* $\mathcal{X}$*.*

Algorithm B.0.1 (Frank-Wolfe algorithm) *Start with any* $x^0 \in \mathcal{X}$*. Compute* x^{i+1} *from* x^i *as follows.*

- $$v^i \in arg\ vertex \min_{x \in \mathcal{X}} \bigtriangledown f(x^i)x$$

- $$Stop\ if\ \ \bigtriangledown f(x^i)v^i = \bigtriangledown f(x^i)x^i$$

- $$x^{i+1} = (1-\lambda^i)x^i + \lambda^i v^i \ where$$

$$\lambda^i \in arg\min_{0\leq\lambda\leq 1} f((1-\lambda)x^i + \lambda v^i)$$

Theorem B.0.1 (Convergence of Frank-Wolfe algorithm) *The algorithm terminates at some x^j that satisfies the minimum principle necessary optimality condition: $\bigtriangledown f(x^j)(x-x^j) \geq 0$, for all $x \in \mathcal{X}$, or each accumulation point $\bar{x}$ of the sequence $\{x^i\}$ satisfies the minimum principle.*

Proof. If the algorithm stops at x^j then

$$\bigtriangledown f(x^j)x^j = \bigtriangledown f(x^j)v^j = \min_{x\in\mathcal{X}} \bigtriangledown f(x^j)x$$

and the minimum principle is satisfied at x^j. If the algorithm does not terminate, let $\{x^{i_j}\} \rightarrow \bar{x}$ and without loss of generality, let $v^{i_j} = \bar{v}$, some fixed vertex of $\mathcal{X}$. Then for $\lambda \in [0,1]$

$$f((1-\lambda)x^{i_j} + \lambda\bar{v}) - f(x^{i_j}) \geq \min_{x\in[x^{i_j},\bar{v}]} f(x) - f(x^{i_j}) = f(x^{i_j+1}) - f(x^{i_j}).$$

Since $\{f(x^i)\}$ is a nonincreasing sequence bounded below it converges. Hence the limit of the last difference above is zero, and we have in the limit

$$f((1-\lambda)\bar{x} + \lambda\bar{v}) - f(\bar{x}) \geq 0 \quad \forall\ \lambda \in [0,1]$$

Letting $\lambda \rightarrow 0$ and invoking the differentiability of f gives

$$\bigtriangledown f(\bar{x})(\bar{v} - \bar{x}) \geq 0.$$

But by algorithm construction

$$\bigtriangledown f(x^{i_j})(x - x^{i_j}) \geq \bigtriangledown f(x^{i_j})(\bar{v} - x^{i_j}) \quad \forall\ x\ \in\ \mathcal{X}$$

and in the limit

$$\bigtriangledown f(\bar{x})(x - \bar{x}) \geq \bigtriangledown f(\bar{x})(\bar{v} - \bar{x}) \geq 0\ \ \forall\, x \in \mathcal{X}$$

□

We establish now finite termination of the Frank-Wolfe algorithm when $f(x)$ is a bilinear nonnegative function with a zero minimum. This is precisely our case of bilinear separability.

Theorem B.0.2 (Finite termination theorem for Frank-Wolfe algorithm) *In Problem A.1 let*

$$f(x) := (Gx + p)(Hx + q)$$

$$Gx + p \geq 0,\ Hx + q \geq 0\ \ \forall x \in \mathcal{X}$$

where $G, H \in R^{k \times n}$ *and* $\mathcal{X}$ *is a polyhedral set with no straight lines going to infinity in both directions. If the sequence* x^i *of the Frank-Wolfe algorithm accumulates to an* $\bar{x}$ *such that* $f(\bar{x}) = 0$*, then one of the vertices* $\{v^i\}$ *of* $\mathcal{X}$ *generated by the algorithm is a solution. Else* $\bar{x}$ *satisfies the minimum principle necessary optimality condition.*

Proof. Let $\mathcal{V}$ be the finite subset of vertices of $\mathcal{X}$ that constitutes the sequence of vertices $\{v^i\}$ generated by the algorithm. Then

$$\{x^i\} \subset \textit{ convex hull } \{x^0 \cup \mathcal{V}\} \textit{ and } \bar{x} \in \textit{ convex hull } \{x^0 \cup \mathcal{V}\}.$$

where $\bar{x}$ is an accumulation point of $\{x^i\}$ such that $f(\bar{x}) = 0$. If $\bar{x} \in \mathcal{V}$, then we are done. If not then

$$\bar{x} = (1 - \lambda)x + \lambda v, \textit{for some } x \in \mathcal{X},\ v \in \mathcal{V} \textit{ and } \lambda \in (0, 1)$$

Since for $j = 1, \ldots, k$, one of the linear functions $G_j\bar{x} + p_j$ or $H_j\bar{x} + q_j$ is zero and both are nonnegative at x and v, that same linear function must vanish at both x and v. Hence

$$G_j v + p_j = 0 \textit{ or } H_j v + q_j = 0,\ j = 1, \ldots, k$$

Hence $f(v) = 0$. The last statement of the theorem follows from Theorem B.0.1.

□