CENTER FOR
PARALLEL OPTIMIZATION

SERIAL AND PARALLEL MULTICATEGORY DISCRIMINATION

by

Kristin P. Bennett and O. L. Mangasarian

Computer Sciences Technical Report #1165

July 1993

Serial and Parallel Multicategory Discrimination

Kristin P. Bennett & O. L. Mangasarian*

July 27, 1993

Abstract

A parallel algorithm is proposed for a fundamental problem of machine learning, that of multicategory discrimination. The algorithm is based on minimizing an error function associated with a set of highly structured linear inequalities. These inequalities characterize piecewise-linear separation of k sets by the maximum of k affine functions. The error function has a Lipschitz continuous gradient that allows the use of fast serial and parallel unconstrained minimization algorithms. A serial quasi-Newton algorithm is considerably faster than previous linear programming formulations. A parallel gradient distribution algorithm is used to parallelize the error-minimization problem. Preliminary computational results are given for both a DECstation 5000/125 and a Thinking Machines Corporation CM-5 multiprocessor.

1 Introduction

We consider a fundamental problem of machine learning and pattern recognition, that of discriminating between k sets. Given k disjoint sets, $\mathcal{A}^i$, $i = 1, \ldots, k$, in the n-dimensional real space R^n, the problem is to construct a function that discriminates between these k sets. The function can then be used to classify future points that belong to one of the sets. We propose a piecewise-linear convex function which is the maximum of k linear (affine) functions. This function has proven to be very useful in decision-tree learning methods [5]. In [2], a linear programming approach was proposed for constructing the function by minimizing the average classification error. In the present work we formulate a 2-norm approach that involves the minimization of an unconstrained piecewise-quadratic convex function with a Lipschitz-continuous gradient. The two principal advantages of the new formulation over the linear programming approach are that (i) the serial version of the new approach is much faster than the linear programming formulation, and (ii) the new approach is much easier to parallelize via an iterative parallel gradient distribution algorithm [15]. Nilsson [19], Duda-Fossum [6], Duda-Hart [7], and Fukunaga [11] considered iterative methods that are extensions of the perceptron algorithm or the Motzkin-Schoenberg algorithm [16] for determining a piecewise-linear separator provided one exists. Unlike our proposed approach, convergence of these iterative methods is not known if a separating piecewise-linear surface does not exist [11, p. 374].

We give now an outline of the paper. In Section 2 we review the linear-programming formulation (5) for piecewise-linear separation using the 1-norm error formulation, and then give the 2-norm formulation (6). We establish a new simple condition (9) for the occurrence of the null solution for the 2-norm problem (6), which turns out to be equivalent to that for the 1-norm formulation (5) [2].

*Computer Sciences Department, University of Wisconsin, 1210 West Dayton Street, Madison, WI 53706, email: *bennett@cs.wisc.edu, olvi@cs.wisc.edu*. This material is based on research supported by Air Force Office of Scientific Research Grant AFOSR-89-0410, National Science Foundation Grants CCR-9101801 and CDA-9024618.

In Section 3, we discuss both serial and parallel algorithms for solving the optimization problem and compare this formulation with previous approaches. The pertinent theorem (Theorem 3.1) of the parallel gradient distribution method [15] is given and an algorithm based on it is described (Theorem 3.2). Details of the algorithms and implementation are given in Section 3. Section 4 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.

Our notation is described now. For a vector x in the h-dimensional real space R^h, x_+ will denote the vector in R^h with components $(x_+)_i := \max\{x_i, 0\}$, $i = 1, \ldots, h$. The notation $A \in R^{m \times h}$ will signify a real $m \times h$ matrix. For such a matrix, A^T will denote the transpose while A_i will denote the ith row. A vector of ones in the real space R^{m^i}, $m = \sum_{i=1}^{k} m^i$, will be denoted by e^i. The 2-norm is denoted by $\|.\|_2$, while the 1-norm is denoted by $\|.\|_1$. 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 $f: R^h \to R$, ∇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_j, $j = 1, \ldots, k$ in R^h, the point $\sum_{j=1}^{k} \lambda_j y_j$, 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 write $f \in C^1(R^h)$.

2 Multicategory Separation by a Piecewise-Linear Surface

We begin by defining the concept of piecewise-linear separation of k sets in R^n [19, 2] and formulating the problem of minimizing the 1-norm error as a linear-program.

Definition 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*

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

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

$$p(x) = \max_{1 \leq \ell \leq k} x w^\ell - \gamma^\ell, \tag{2}$$

and

$$\left\langle \begin{array}{l} p(x) = x w^i - \gamma^i \\ p(x) > x w^j - \gamma^j \end{array} \right\rangle \begin{array}{l} for \;\; x \in \mathcal{A}^i, \; i = 1, \ldots, k \\ for \;\; j \neq i \end{array} \tag{3}$$

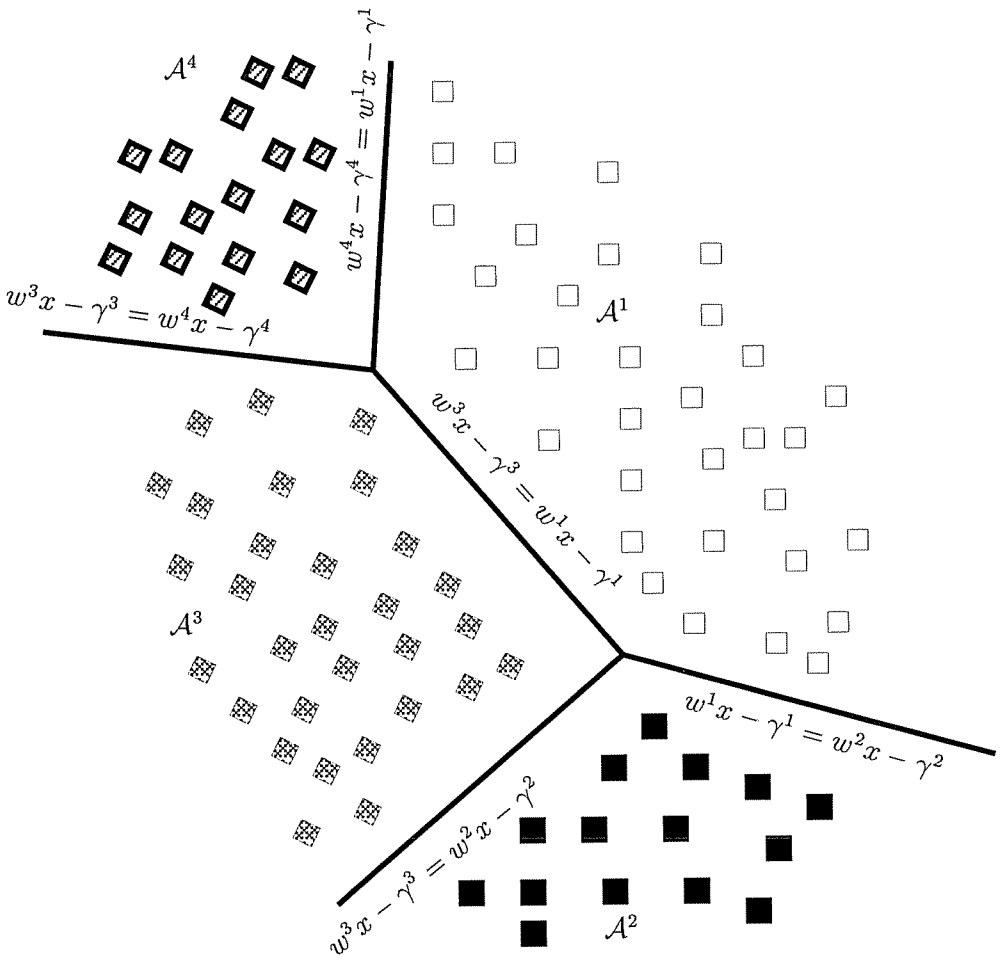

Figure 1: **Piecewise-linear separator of 4 classes in R^2**

In [2], 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 (1) is zero, namely

$$0 = \min_{w^i,\gamma^i} \sum_{i=1}^{k} \sum_{\substack{j=1 \\ j\neq i}}^{k} \frac{e^i}{m^i} \left(-A^i(w^i - w^j) + e^i(\gamma^i - \gamma^j) + e^i\right)_+ \tag{4}$$

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

$$\min_{w^i,\gamma^i,y^{ij}} \left\{ \sum_{i=1}^{k} \sum_{\substack{j=1 \\ j\neq i}}^{k} \frac{e^i y^{ij}}{m^i} \;\middle|\; \begin{array}{c} y^{ij} \geq -A^i(w^i - w^j) + e^i(\gamma^i - \gamma^j) + e^i,\ y^{ij} \geq 0, \\ i \neq j,\ i,\ j = 1, \ldots, k \end{array} \right\} \tag{5}$$

Figure 1 depicts the piecewise-linear classifier found by the LP (5) for a typical piecewise-linear separable case with $k = 4$ and $n = 2$.

As shown in [2], the linear program (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 (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 2 illustrates a decision surface found by a multivariate decision tree algorithm and Figure 3 depicts the corresponding decision tree. Although the LP (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 [21]. Previous iterative approaches based on extensions to the perceptron algorithm [19, 6, 7] do not have stable performance for the inseparable case and as a result heuristic methods [12, 5] 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 2.1 and reformulating the problem we can accomplish this.

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

$$\min_{w^i,\gamma^i} f(w,\gamma) \;=\; \frac{1}{2}\sum_{i=1}^{k}\sum_{\substack{j=1\\ j\neq i}}^{k}\frac{1}{m^i}\left\|(-A^i(w^i-w^j)+e^i(\gamma^i-\gamma^j)+e^i)_+\right\|_2^2 \tag{6}$$

Squaring the plus function results in a piecewise-quadratic function that is differentiable. We refer to (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

$$\begin{aligned}\nabla_{w^\ell} f(w,\gamma) \;=\; & \sum_{\substack{j=1\\ j\neq \ell}}^{k}\frac{-1}{m^\ell}\,A^{\ell T}(-A^\ell(w^\ell-w^j)+e^\ell(\gamma^\ell-\gamma^j)+e^\ell)_+ + \\ & \sum_{\substack{i=1\\ i\neq l}}^{k}\frac{1}{m^i}\,A^{iT}(-A^i(w^i-w^\ell)+e^i(\gamma^i-\gamma^\ell)+e^i)_+\end{aligned} \tag{7}$$

$$\begin{aligned}\nabla_{\gamma^\ell} f(w,\gamma) \;=\; & \sum_{\substack{j=1\\ j\neq \ell}}^{k}\frac{1}{m^\ell}\,e^\ell(-A^\ell(w^\ell-w^j)+e^\ell(\gamma^\ell-\gamma^j)+e^\ell)_+ + \\ & \sum_{\substack{i=1\\ i\neq \ell}}^{k}\frac{-1}{m^i}e^i(-A^i(w^i-w^\ell)+e^i(\gamma^i-\gamma^\ell)+e^i)_+\end{aligned} \tag{8}$$

As in the 1-norm formulation, the inequalities (1) for piecewise-linear separation hold if and only if the minimum of (6) is zero. Consequently the following theorem holds.

Theorem 2.1 (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) 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 2.1.*

As was the case for linear and piecewise-linear separation of two sets by linear programming [3, 4], it is important to determine when the useless null solution occurs for sets $\mathcal{A}^i$, $i = 1,\ldots,k$

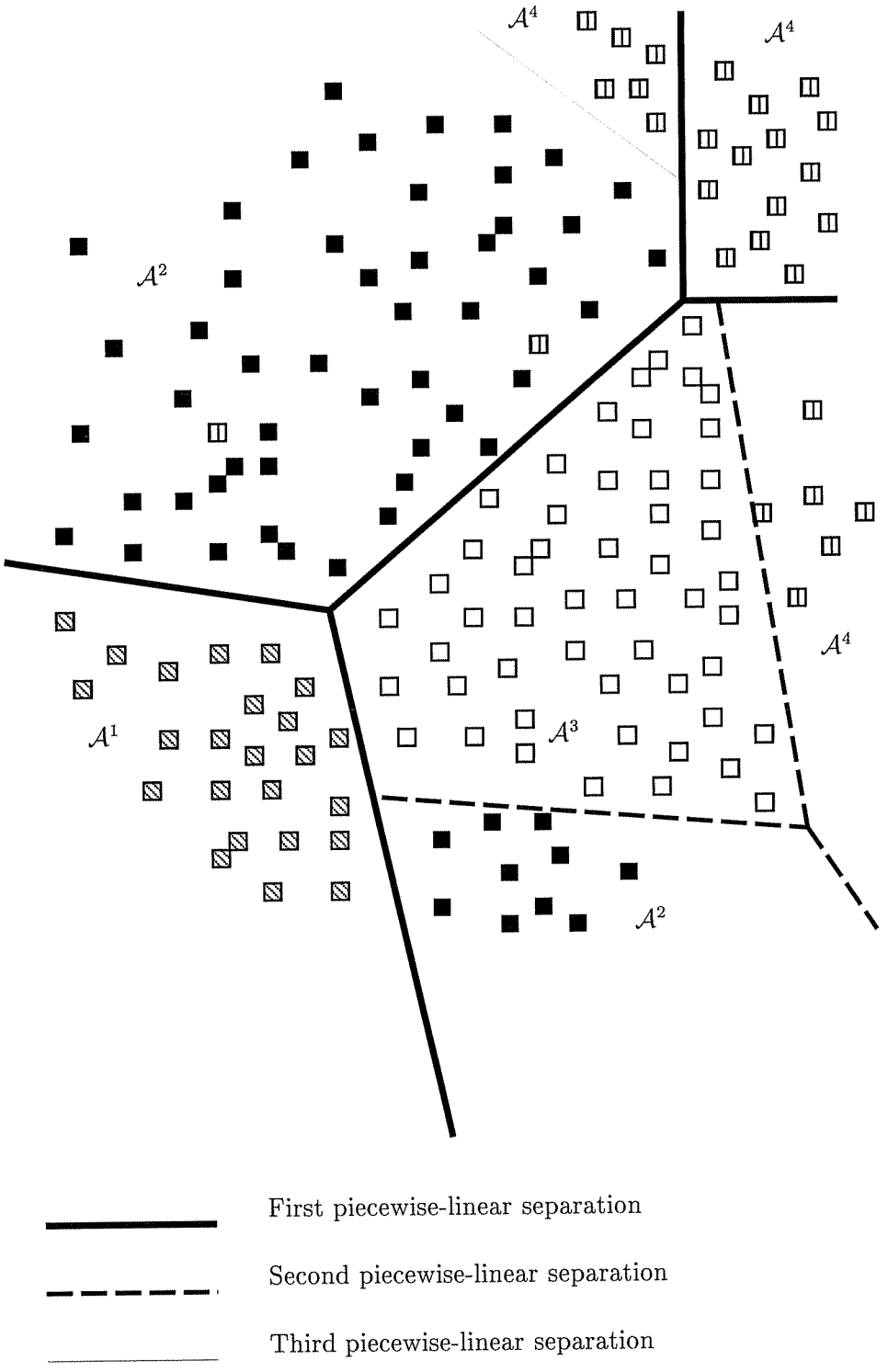

Figure 2: **Geometric depiction of decision tree consisting of 3 piecewise-linear separators distinguishing 4 classes in R^2**

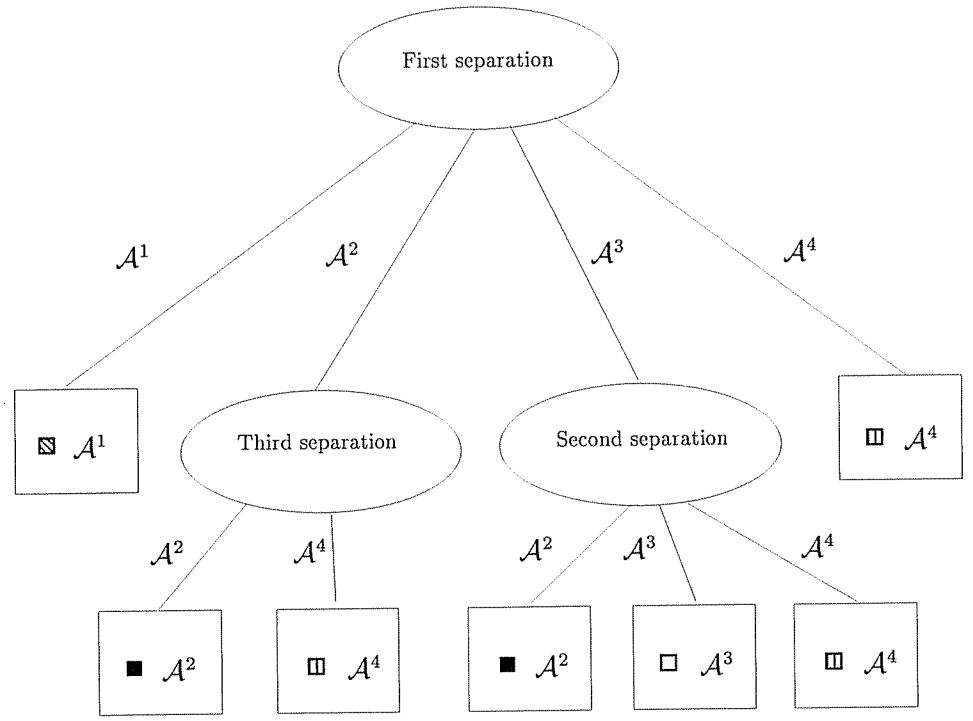

Figure 3: **Decision tree representing the 3 piecewise-linear separators depicted in Figure 2**

that are not piecewise-linear separable. Note that the piecewise-linear separation (1) is achieved by a special pairwise linear separation between the sets $A^i,\ i = 1,\ldots,k$, 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 2.2 (Null Solution Occurrence) *The piecewise quadratic minimization (PQM) (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*

$$\frac{e^i A^i}{m^i} = \frac{e^j A^j}{m^j},\ i = 1,\ldots,k,\ j = 1,\ldots,k \tag{9}$$

Proof. The vectors $w^i - w^j = 0,\ i \neq j,\ i, j = 1,\ldots,k$ constitute an optimal solution of problem (6) if and only if $\gamma^i = \gamma^j = 0,\ i \neq j,\ i,j = 1,\ldots,k$. For these values of $w^i, \gamma^i, i = 1,\ldots,k$,

$$\nabla_{w^\ell} f(w,\gamma) \ = \ 0,\ \nabla_{\gamma^\ell} f(w,\gamma) \ = \ 0,\ \ell = 1,\ldots,k. \tag{10}$$

Evaluating the partial gradients at such an optimal point gives

$$\nabla_{w^\ell} f(w,\gamma) \ = \ \sum_{\substack{j=1\\ j\neq\ell}}^{k} -\frac{1}{m^\ell} A^{\ell T} e^\ell \ + \ \sum_{\substack{i=1\\ i\neq\ell}}^{k} \frac{1}{m^i} A^{iT} e^i \ = \ (1-k)\frac{A^{\ell T} e^\ell}{m^\ell} \ + \ \sum_{\substack{i=1\\ i\neq\ell}}^{k} \frac{1}{m^i} A^{iT} e^i \ = \ 0 \tag{11}$$

$$\nabla_{\gamma^\ell} f(w,\gamma) \ = \ \sum_{\substack{j=1\\ j\neq\ell}}^{k} \frac{1}{m^\ell} e^\ell e^\ell \ - \ \sum_{\substack{i=1\\ i\neq\ell}}^{k} \frac{1}{m^i} e^i e^i \ = \ (k-1) \ - \ (k-1) \ = \ 0 \tag{12}$$

Equation (12) is automatically satisfied. Obviously, (9) implies (11). To show the converse, suppose that condition (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 (11) holds, then we have the contradiction

$$(k-1)\frac{e^1 A^1}{m^1} > (k-1)\frac{e^2 A^2}{m^2} = \frac{e^1 A^1}{m^1} + \sum_{j=3}^{k} \frac{e^j A^j}{m^j} > \frac{e^2 A^2}{m^2} + \sum_{j=3}^{k} \frac{e^j A^j}{m^j} = (k-1)\frac{e^1 A^1}{m^1}. \tag{13}$$

Hence (11) does not hold and the proof is complete.

□

It is also true for the LP (5), that the null solution occurs if and only if condition (9) holds [2]. However, condition (9) was written in a slightly more complex form in [2, 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.

3 PQM and Partial Gradient Distribution

In this section we examine serial and parallel methods for solving the piecewise quadratic minimization (6). This problem may be solved serially by any unconstrained first-order optimization method. Our computational results, presented in Section 4, indicate that a quasi-Newton method [18, p.2] was considerably faster than solving the corresponding linear program (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 [15] that is described below. We refer the reader to [15] 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 3.1 ***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 [15] for the convergence proof of this theorem and other related algorithms.

Theorem 3.1 ***Parallel gradient distribution algorithm theorem 1 [15, 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:

$$-\nabla_\ell f(x_i) d_i^\ell \geq \tau_\ell(\|\nabla_\ell f(x_i)\|),\ \ell = 1, \ldots k \tag{14}$$

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 $\{1, \ldots, k, \}$:*

$$f(x_i) - f(y_i^\ell,\ x_i^{\bar{\ell}}) \geq \mu_\ell(-\nabla_\ell f(x_i) d_i^\ell) \geq 0,\ \ell = 1, \ldots, k \tag{15}$$

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*

$$f(x_{i+1}) \leq \min_{1 \leq \ell \leq k} f(y_i^\ell,\ x_i^{\bar{\ell}}) \tag{16}$$

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 [15] a number of implementations of Theorem 3.1 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 3.1 based on the following simple remarks. Instead

of choosing y^ℓ, $\ell = 1, \ldots, k$, so as to satisfy the realizable inequalities (14) and (15), we take the best possible y_i^ℓ, that is:

$$f(y_i^\ell, x_i^{\bar{\ell}}) \;=\; \min_{x^\ell} f(x^\ell, x_i^{\bar{\ell}}), \quad \ell = 1, \ldots, k. \tag{17}$$

Hence conditions (14) and (15) are satisfied for some τ_ℓ and μ_ℓ, $\ell = 1, \ldots, k$. Similarly for the synchronization step (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 (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 [9].

Theorem 3.2 ***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*

$$(y_i^\ell, x^{\bar{\ell}}) \;\in arg \min_{x^\ell} f(x^\ell, x_i^{\bar{\ell}}) \tag{18}$$

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

$$x_{i+1} = \lambda_i^0 x_i + \lambda_i^1 (y_i^1, x_i^{\bar{1}}) + \ldots + \lambda_i^k (y_i^k, x_i^{\bar{k}}) \in arg \min_{\lambda^0, \lambda^1, \ldots, \lambda^k} f(\lambda^0 x_i + \lambda^1 (y_i^1, x_i^{\bar{1}}) + \ldots + \lambda^k (y_i^k, x_i^{\bar{k}})) \tag{19}$$

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) is to let $x_i^\ell = (w_i^\ell, \gamma_i^\ell)$. This results in the following algorithm.

Algorithm 3.1 ***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).*

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

$$(y_i^\ell, x_i^{\bar{\ell}}) \in arg \min_{x^\ell} f(x^\ell, x_i^{\bar{\ell}}) \tag{20}$$

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

- ***Synchronization:***

$$x_{i+1} = \lambda_i^0 x_i + \lambda_i^1 (y_i^1, x_i^{\bar{1}}) + \ldots + \lambda_i^k (y_i^k, x_i^{\bar{k}}) \in arg \min_{\lambda^0, \lambda^1, \ldots, \lambda^k} f(\lambda^0 x_i + \lambda^1 (y_i^1, x_i^{\bar{1}}) + \ldots + \lambda^k (y_i^k, x_i^{\bar{k}})) \tag{21}$$

- *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:

$$w_0^\ell \;=\; \frac{e^\ell A^\ell}{m^\ell} \;-\; \frac{\sum_{j=1}^{k} e^j A^j}{\sum_{j=1}^{k} m^j}, \qquad \gamma_0^l = 0, \qquad l = 1, \ldots, k. \tag{22}$$

The unconstrained convex minimization subproblems, (20) and (21), were solved by using the quasi-Newton algorithm in the MINOS [18] optimization package.

Many variations of the direction and synchronization steps of Algorithm 3.1 are possible under Theorem 3.1. The algorithm presented was the best we found computationally. The algorithm is easily parallelized by distributing each of the subproblems (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 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 (21) was replaced with a strong convex combination of the k points found in step (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.

4 Computational Results

We conducted a series of computational experiments to investigate three questions: How does the LP formulation (5) compare with the PQM formulation (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 [18] package. Actual discrimination problems were used to compare the algorithms. These data sets are available via anonymous ftp (file transfer protocol) from the University of California-Irvine Repository of Machine Learning Databases and Domain Theories [17]. The wine recognition data [1], referred to as wine, uses the chemical analysis of wine to determine the cultivar. This wine set is piecewise-linear separable. Fisher's classical Iris identification problem [10], referred to as Iris, used physical attributes of Iris blossoms to determine the type of Iris. The Iris data is almost piecewise-linear separable. In the forensic glass identification data [8], referred to as glass, the chemical analysis of forensic glass is used to determine the origin of the glass. The glass data is not piecewise-linear separable. In the image segmentation problem [5], 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-n since it is not piecewise-linear separable. Table 1 lists the number of points, attributes, and classes contained in each of the data sets.

4.1 Comparison of Serial Implementation of LP and PQM

We compared the linear programming formulation (5) and the new piecewise-quadratic minimization formulation (6) on the machine learning problems: wine, Iris, glass, and image-s described

Table 1: **Description of datasets used in experiments**

Dataset	Dimension	Classes	Total Points
Wine	13	3	178
Iris	4	3	150
Glass	9	7	214
Image-s	19	7	210
Image-n	19	7	2310

above. The LP (5) was solved serially using MINOS [18], and PQM (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 [13] 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 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 (5) and the results of PQM (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.

4.2 Comparison of Serially Implemented MCD-PGD and Quasi-Newton

In the second set of experiments, we compared the computational results of solving PQM (6) with a quasi-Newton method versus solving PQM with the MCD-PGD Algorithm 3.1 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 3 gives the training set accuracies, the testing set accuracies, and the training times for both algorithms on the five datasets.

The average training set and testing set accuracies were not significantly different on any dataset. The relaxation of the optimality criterion discussed in Section 3 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

Table 2: **Comparison of Serial Implementation of Linear Program (5) and Piecewise-Quadratic Minimization(6)**

Training Set Accuracy

	Average Accuracy (%)		t-test
Dataset	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

	Average Accuracy (%)		t-test
Dataset	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

	Training Time (secs)		t-test
Dataset	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

algorithm [20], 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 [14, 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.

4.3 Comparison of Parallel Implementation of MCD-PGD and Quasi-Newton

For the final set of comparisons, we implemented Algorithm 3.1 on the 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 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%. The lower efficiency is primarily caused by segments of the algorithm that create idle time. The subproblems (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 (21) in Algorithm 3.1 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.

5 Conclusion

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.

Table 3: **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 4: **Comparison of MCD-PGD and quasi-Newton algorithms on the CM-5**

Training Time

Dataset	Classes k	Quasi-Newton Training Time (secs)	MCD-PGD Training Time (secs)	t-test p	time reduction	efficiency (%)
Glass	7	63.3 ± 24.0	17.7 ± 3.0	< 0.00001	3.6	51
Image-s	7	203.6 ± 81.7	32.1 ± 9.1	< 0.00001	6.3	91
Image-n	7	2470.0 ± 1113.1	704.0 ± 137.0	< 0.00001	3.6	50

References

[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] 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.

[3] 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.

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

[5] 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.

[6] 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.

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

[8] 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.

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

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

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

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

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

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

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

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

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

[18] B.A. Murtagh and M.A. Saunders. MINOS 5.1 user's guide. Technical Report SOL 83.20R, Stanford University, January 1987.

[19] N. J. Nilsson. *Learning Machines.* MIT Press, Cambridge, Massachusetts, 1966.

[20] 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.

[21] 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.