# Mathematical Programming in Machine Learning *

O. L. Mangasarian†

Mathematical Programming Technical Report 95-06
April 1995–Revised July 1995

**Abstract**

**We describe in this work a number of central problems of machine learning and show how they can be modeled and solved as mathematical programs of various complexity.**

## 1 Introduction

Machine learning can be thought of as generalizing information gleaned from given data to new unseen data. As such it can be considered as determining a mapping between an input set and an output set in a robust manner that is amenable to generalization. In this work we shall concentrate on a number of fundamental problems of machine learning, and show how mathematical programming plays a significant role in their formulation and solution. In Section 2 we consider the classical problem of discriminating between two point sets in the n-dimensional real space $R^n$, and show that its complexity ranges from polynomial-time to NP-complete, depending on the measure of error employed. When the traditional distance of a misclassified point to a separating plane is used as an error, a single linear program [6, 15, 16, 4] usually solves the problem. Recently [10, 18, 2, 7] a more complex, and for certain applications more realistic, error measure has been considered, namely the number of misclassified points by a separating plane. This problem, even though shown to be NP-complete [7], can be effectively solved by a parametric [2] or a hybrid method [7]. In Section 3 we describe a central problem of machine learning, that of improving generalization [27]. We give a very simple model which justifies the often accepted rule-of-thumb of machine learning and approximation theory, that overfitting leads to poor generalization. In fact we go the opposite direction, and show that inexact fitting can lead to improved generalization. In Section 4 we use an equivalence between the step function and the complementarity problem to show that the problem of training a neural network can be represented as mathematical program with equilibrium constraints (MPEC) which has been studied recently in the literature [14].

A word about our notation now. For a vector $x$ in the $n$-dimensional real space $R^n$, $x_+$ will denote the vector in $R^n$ with components $(x_+)_i := \max\{x_i, 0\}$, $i = 1, \ldots, n$. Similarly $x_*$ will denote the vector in $R^n$ with components $(x_*)_i := (x_i)_*$, $i = 1, \ldots, n$ , where $()_*$ is the step function that maps a nonpositive number into zero and a positive number into one. The $p$-norm will be denoted by $\|\cdot\|_p$ for $p = 1, 2, \ldots, \infty$, while $\|\cdot\|$ will denote an arbitrary norm. We will

*This material is based on research supported by Air Force Office of Scientific Research Grant F49620-94-1-000036 and National Science Foundation Grant CCR-9322479.

†Computer Sciences Department, University of Wisconsin, 1210 West Dayton Street, Madison, WI 53706, email: *olvi@cs.wisc.edu.*

also make use of the function $x_-$ which will denote the vector in $R^n$ with components $(x_-)_i := \min\{x_i, 1\}$, $i = 1, \ldots, n$. 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 row $i$. For two vectors $x$ and $y$ in $R^n$, $x^Ty$ will denote the scalar product, while $x \perp y$ will denote $x^Ty = 0$. A vector of ones in a real space of arbitrary dimension will be denoted by $e$. The symbols ":=" and "=:" will denote a definition of a term adjacent to the colon by a term adjacent to the equality.

## 2 Linear Discrimination

We begin with the fundamental problem of constructing a linear discriminator between two given point sets $\mathcal{A}$ and $\mathcal{B}$ in $R^n$. That is we look for a plane

$$x^Tw = \theta, \tag{1}$$

such that

$$\begin{array}{ll} x^Tw > \theta & \text{for } x \in \mathcal{A} \\ x^Tw < \theta & \text{for } x \in \mathcal{B}. \end{array} \tag{2}$$

Here $w$ is the normal to the plane and $\dfrac{|\theta|}{\|w\|_2}$ is the Euclidean distance from the origin to the plane. In general it is not possible to satisfy (2) except in the special case when the convex hulls of $\mathcal{A}$ and $\mathcal{B}$ do not intersect. Thus, one resorts in the general case to optimizing some error criterion in the satisfaction of (2). The simplest such criterion is to use linear programming in order to construct a plane (1) that maximizes a weighted sum of the distances of correctly classified points to the plane [16, 4] as follows:

$$\max_{w,\theta,y,z} \left\{e^Ty + e^Tz \mid Aw \geq e\theta + y,\ Bw \leq e\theta - z,\ y \leq e,\ z \leq e\right\} \tag{3}$$

Here the rows of the matrices $A \in R^{m\times n}$ and $B \in R^{k\times n}$ represent the $m$ points in $\mathcal{A}$ and the $k$ points in $\mathcal{B}$ respectively, while $e$ is a vector of ones of appropriate dimension. The objective function of (3) represents the sum of distances, positive and truncated to one for correctly classified points and nonpositive for incorrectly classified points, to the plane $x^Tw = \theta$, multiplied by $\|w\|_2$. Although, apparently different from the robust linear program of [4, Proposition 2.4], it is equivalent to it if we set the weights $\delta_1 = \delta_2 = 1$ in the latter and make a simple change of variables. A principal advantage of the formulation (3), is that it ties more easily with the classification maximization formulation (7) below, once we make use of the function

$$(\zeta)_- := \min\{\zeta, 1\} = 1 - (1-\zeta)_+. \tag{4}$$

By using this nondecreasing piecewise-linear concave function, the linear program (3) can be written as the following unconstrained concave maximization problem:

$$\max_{w,\theta} e^T(Aw - e\theta)_- + e^T(-Bw + e\theta)_- \tag{5}$$

Another simplifying feature of the linear programming formulation (3) is the absence of a normalization vector $e$ from both terms of (5), and from the first two constraints of (3). The linear program (3) also maintains the non-nullity properties of $w$ [4, Theorems 2.5 & 2.6], which can be summarized as follows here. The point $(w = 0, \gamma, y, z)$ is a solution of (3) if and only if $e^TA = e^TB$ and $m = k$, in which case the solution is never unique in $w = 0$. If the convex hulls of $\mathcal{A}$ and $\mathcal{B}$ are disjoint, then all points in $\mathcal{A}$ and $\mathcal{B}$ are correctly classified, and the equivalent programs (3) and (5) yield a maximum value of $m + k$, equal to the total number of points in $\mathcal{A}$ and $\mathcal{B}$ that have been completely separated by the plane $x^T w = \theta$. However, in the general case of intersecting convex hulls, the linear program (3) obtains an approximate separating plane that maximizes a weighted sum of distances of points as described above. For this case the number of correctly classified points is given by:

$$e^T y_* + e^T z_* \quad \text{or equivalently} \quad e^T((Aw - e\theta)_-)_* + e^T((-Bw + e\theta)_-)_*, \tag{6}$$

where $(w, \theta, y, z)$ is a solution of (3) and, as indicated earlier, $(\cdot)_*$ is the step function. Although the linear programming formulation (3) is very effective for practical problems [21] and can be used in the construction of neural networks [3] as well multi-surface discriminators [1, 4], it does not minimize the number of misclassified points by the plane (1), which may be an important consideration in certain applications. In order to minimize the number of misclassified points, we need to *maximize* the number of satisfied components of the inequalities (2). This corresponds to solving the following problem:

$$\max_{w,\theta} e^T(Aw - e\theta)_* + e^T(-Bw + e\theta)_* \tag{7}$$

We refer to this problem as the *classification maximization problem*, that is the problem of maximizing the number of correctly classified points. Although this is an NP-complete problem [7, Proposition 2], effective methods for its solution have been proposed [18, 7] and successfully tested on real world problems [2, 7]. We outline two of these approaches briefly now.

We first describe a hybrid approach, recently proposed and tested successfully on ten publicly available databases [7]. The idea of this approach is to combine the two criteria described above as follows. For a fixed orientation $w$, translate the plane (1) by varying $\theta$, so as to minimize the number of misclassified points, that is solve (7) for a fixed $w$, which is then a one dimensional problem in $\theta$ with a finite number of objective function values. Then for a fixed $\theta$, rotate the plane (1) by varying $w$, so as to minimize the weighted average of the sum of the distances of the misclassified points to the plane (1), that is solve the linear program (3) in $w$ for a fixed value of $\theta$. The algorithm stops, when successive line searches in $\theta$ fail to decrease the number of misclassified points. Needless to say, there is no guarantee that this approach will give a global solution to the NP-complete problem (7). However it seems to have the best generalization [7] as determined by tenfold cross-validation [26] on the ten data sets employed.

We describe now another approach for solving the classification maximization problem (7), by reducing it to an LPEC, a linear program with equilibrium constraints [18]. We begin with a variation of a lemma of [18, Lemma 2.1] which ensures that the backward implication of the lemma holds also for zero components of $a$.

**Lemma 2.1 Equilibrium characterization of step function** $()_*$ *For* $r \in R^m,\ u \in R^m,\ a \in R^m$ *and* $e$*, a vector of ones in* $R^m$ *:*

$$r = (a)_*,\ u = (a - \epsilon e)_+ \iff \left( \begin{array}{c} 0 \le r \perp u - a + \epsilon e \ge 0 \\ 0 \le u \perp -r + e \ge 0 \end{array} \right), \tag{8}$$

*where* $\epsilon$ *is a sufficiently small positive number, that is*

$$0 < \epsilon < \inf_{a_i \neq 0} |a_i|. \tag{9}$$

**Proof** The points $r = (a)_*$ and $u = (a - \epsilon e)_+$ uniquely solve the dual linear programs

$$\max_r \ \{(a - \epsilon e)^T r | 0 \leq r \leq e\} \ \text{ and } \ \min_u \ \{e^T u | u \geq a - \epsilon e, \ u \geq 0\}. \tag{10}$$

The right hand side of the equivalence (8) is merely the Karush-Kuhn-Tucker necessary and sufficient optimality conditions for $r = (a)_*$ and $u = (a - \epsilon e)_+$ to solve (10). $\square$

We note that the use of $\epsilon$ is unnecessary in [18], because of the following equivalence:

$$r = (a)_*, \ u = (a)_+ \iff (r, u) \in \arg\min_{r,u} \ \{e^T r | 0 \leq r \perp u - a \geq 0, \ 0 \leq u \perp -r + e \geq 0\}, \tag{11}$$

and because the term $e^T r$ is minimized in [18], but is being maximized here (see (12) for example). With Lemma 2.1, we can reformulate the classification maximization problem (7) as the following LPEC with $\epsilon$ sufficiently small and positive:

$$\begin{array}{ll} \underset{w,\theta,r,u,s,v}{\text{maximize}} & e^T r + e^T s \\ & 0 \leq r \perp u - Aw + e\theta + \epsilon e \geq 0 \\ \text{subject to} & 0 \leq u \perp -r + e \geq 0 \\ & 0 \leq s \perp v + Bw - e\theta + \epsilon e \geq 0 \\ & 0 \leq v \perp -s + e \geq 0 \end{array} \tag{12}$$

Note that with the exception of the "*perp*" condition, all constraints and the objective function are linear. To overcome the nonlinear effect of the $\perp$-condition, an implicitly exact penalty function formulation has been proposed as well as a parametric approach [18]. The parametric approach is preferable, because (12) has infinitely many stationary points as was pointed out in [18]. The reason for this anomaly is that any $(w, \theta)$ determining a plane $x^T w = \theta$ that does not contain any points from either the sets $\mathcal{A}$ or $\mathcal{B}$ , is a stationary solution for problem (12). This is so because a slight perturbation of the plane does not change the number of misclassified points. To overcome this difficulty a parametric reformulation was proposed in [18] and implemented in [2]. For the classification maximization problem (7), the parametric reformulation of (12) is the following:

$$\begin{array}{ll} \underset{w,\theta,r,u,s,v}{\text{minimize}} & [r^T(-Aw + e\theta + \epsilon e) + e^T u] + [s^T(Bw - e\theta + \epsilon e) + e^T v] =: f(\mu) \\ & 0 \leq r, \ \ u - Aw + e\theta + \epsilon e \geq 0 \\ & 0 \leq u, \ \ -r + e \geq 0 \\ \text{subject to} & 0 \leq s, \ \ v + Bw - e\theta + \epsilon e \geq 0 \\ & 0 \leq v, \ \ -s + e \geq 0 \\ & e^T r + e^T s \geq \mu \\ & \mu \in [0, \infty) \end{array} \tag{13}$$

Here $\mu$ is a parameter that represents the number of points correctly classified. The largest value of $\mu$, such that the objective function has a minimum of zero, is the maximum number of points that can be correctly classified by a plane $x^T w = \theta$. Note that $f(\mu)$ is a nondecreasing function of $\mu$, and the largest value $\bar{\mu}$ for which $f(\bar{\mu}) = 0$, constitutes a maximum to the NP-complete classification maximization problem (7). The parametric approach consists of starting at some large $\mu > \bar{\mu}$, solving (13) by a Frank-Wolfe algorithm [8, 5], for decreasing values of $\mu$ until $\bar{\mu}$ is reached. Efficient estimation of successive values of $\mu$ can be achieved by a secant method applied to $f(\mu)$. The method seems to work quite well as evidenced by computational results given in [2, 7].

## 3 Improving Generalization

In this section we shall consider a fundamental problem of machine learning: How to train a system on a given training set so as to improve generalization on a new unseen testing set [13, 24, 28]. We shall concentrate on some very recent results [27] obtained for a simple linear model and which make critical use of mathematical programming ideas. These ideas, although rigorously established for a simple linear model only, seem to extend to much more complex systems, including neural networks [27].

The model that we shall consider here consists of the *training set* $\{A, a\}$ where $A$ is a given $m \times n$ real matrix and $a$ is a given $m \times 1$ real vector. A vector $x$ in $R^n$ is to be "learnt" such that the linear system

$$Ax = a, \tag{14}$$

which does not have an exact solution, is satisfied in some approximate fashion, and such that the error in satisfying

$$Cx = c, \tag{15}$$

for some unseen *testing set* $(C, c) \in R^{k \times n} \times R^k$, is minimized. Of course, if we disregard the testing set error (15), the problem becomes the standard least-norm problem:

$$\min_{x \in R^n} \|Ax - a\|, \tag{16}$$

where $\|\cdot\|$ is some norm on $R^m$. However with an eye to possible perturbations in the given training set $\{A, a\}$, we pose the following motivational question: If the vector $a$ of the training set is known only to an accuracy of $\tau$, where $\tau$ is some positive number, does it make sense to attempt to drive the error to zero as is done in (16), or is it not better to tolerate errors in the satisfaction of $Ax = a$ up to a magnitude of $\tau$? In other words, instead of (14), we should try to satisfy the following system of inequalities, in some best sense:

$$-e\tau \leq Ax - a \leq e\tau \tag{17}$$

To do that, we solve the following regularized quadratic program for some nonnegative $\tau$ and a small positive $\epsilon$:

$$\begin{array}{ll} \underset{x,y,z}{\text{minimize}} & \frac{1}{2}\|y\|_2^2 + \frac{1}{2}\|z\|_2^2 + \frac{\epsilon}{2}\|x\|_2^2 \\ \text{subject to} & -z - e\tau \leq Ax - a \leq e\tau + y \\ & y, z \geq 0. \end{array} \tag{18}$$

Here $\epsilon$ is a small fixed positive regularization constant that ensures the uniqueness of the $x$ component of the solution. We note immediately, that if $\tau = 0$, problem (18) degenerates to the regularized classical least squares problem:

$$\min_{x \in R^n} \frac{1}{2}\|Ax - a\|_2^2 + \frac{\epsilon}{2}\|x\|_2^2. \tag{19}$$

The key question to ask here, is this: Under what conditions does a solution $x(\tau)$ of (18), for some $\tau > 0$, give a smaller error on a testing set? We are able to give an answer to this question and corroborate it computationally [27], by considering a general testing set $(C, c) \in R^{k \times n} \times R^k$

as well as a simpler testing set, where only the right side of (14) is perturbed. We begin with the latter and simpler perturbation, that is:

$$Ax = a + t, \tag{20}$$

where $t$ is some arbitrary perturbation in $R^m$, and consider the following associated error function:

$$f(\tau) := \frac{1}{2} \|Ax(\tau) - a - t\|_2^2 . \tag{21}$$

In particular we would like to know when is $f(0)$ *not* a local minimum of $f(\tau)$ on the set $\{\tau \mid \tau \geq 0\}$. In fact we are only interested in the $\tau$-interval $[0, \hat{\tau}]$, where $\hat{\tau}$ is defined by

$$\hat{\tau} := \min_x \|Ax - a\|_\infty , \tag{22}$$

because the minimum value of (18) approaches zero, for $\tau \geq \hat{\tau}$, as $\epsilon$ approaches zero. Since $x(\tau)$ is continuous and piecewise-linear on $\tau \geq 0$ it follows that $f(\tau)$ defined by (21) is continuous piecewise-quadratic on $[0, \hat{\tau}]$, and hence attains a minimum at some $\tau$ in $[0, \hat{\tau}]$. Since $f(\tau)$ is directionally differentiable, it follows that *if* the directional derivative $f'(\tau; 1)$ at $\tau = 0$ in the positive direction is negative, then $\tau = 0$ is a strict local maximum of $f(\tau)$. Hence, as measured by the error criterion (21), $x(\tau)$ for some positive $\tau$ provides a better point. The following theorem gives a sufficient condition for $f'(0; 1) < 0$ and thus ensuring that solving (18) for some positive $\tau$ produces an $x(\tau)$ that generalizes better on the system (20) than that obtained by solving a plain regularized least squares problem (19), that is $f(\bar{\tau}) < f(0)$ for some $\bar{\tau} \in (0, \hat{\tau}]$.

**Theorem 3.1** *[27]* **Improved generalization on $Ax = a+t$ with positive training tolerance** *The testing set error function $f(\tau)$ of (21) has a strict local maximum at $0$ and a global minimum on $[0, \hat{\tau}]$, where $\hat{\tau}$ is defined by (22), at some $\bar{\tau} > 0$, whenever*

$$(\epsilon x(0) + A^T t)^T (x(\tau) - x(0)) > 0 \tag{23}$$

*for some $\tau \in (0, \tilde{\tau}]$, for some sufficiently small $\tilde{\tau}$.*

For the more general testing model given by $Cx = c$ of (15), we have the following result for improved generalization.

**Theorem 3.2** *[27]* **Improved generalization on $Cx = c$ with positive training tolerance** *Let $x(\tau)$ be defined by the tolerant training of $Ax = a$ by the quadratic program (18) with tolerance $\tau \geq 0$. Let $g(\tau)$ denote the error generated by $x(\tau)$ in the testing model $Cx = c$, defined by:*

$$g(\tau) := \frac{1}{2} \|Cx(\tau) - c\|_2^2 . \tag{24}$$

*The zero-tolerance error $g(0)$ generated by $x(0)$ is a strict local maximum over $\tau \geq 0$ whenever*

$$\|r(0)\|_2^2 > r(\tau)^T r(0) \text{ for some } \tau \in (0, \tilde{\tau}] \tag{25}$$

*for some sufficiently small $\tilde{\tau}$, where $r(\tau)$ is defined by*

$$r(\tau) := Cx(\tau) - c. \tag{26}$$

Computational results carried out in [27] have corroborated the improved generalization results of Theorem 3.1 above, as well as for more complex models such as neural networks, where a threshold tolerance in measuring the error in the backpropagation algorithm [23, 11, 20] is allowed.

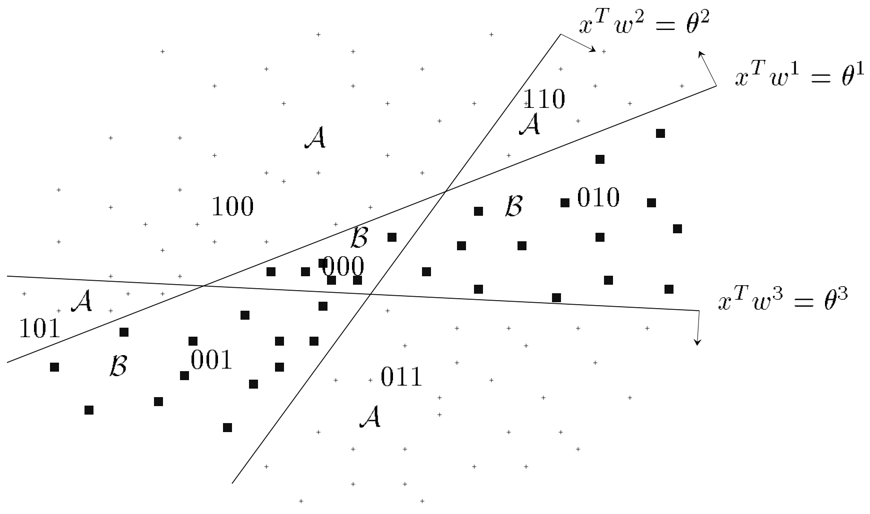


Figure 1: **Seven polyhedral regions in $R^2$ generated by three planes: $x^T w^1 = \theta^1$, $x^T w^2 = \theta^2$ and $x^T w^3 = \theta^3$. Each region contains elements of only one set $\mathcal{A}$ or $\mathcal{B}$ and is tagged by a binary number, the $i$th digit of which denotes whether the region is on the 1-side, $x^T w^i > \theta^i$, or 0-side, $x^T w^i < \theta^i$, of the $i$th plane.**

# 4 Neural Networks as Mathematical Programs with Equilibrium Constraints

A neural network, which is a generalization of a separating plane in $R^n$, can be defined as a nonlinear map from $R^n$ into some set, typically {0,1}. One intuitive way to generate such a map is to divide $R^n$ into various polyhedral regions, each of which containing elements of only one of two given disjoint point sets $\mathcal{A}$ and $\mathcal{B}$ . (See Figure 1.) In its general form, this problem is again an extremely difficult and nonconvex problem. However, various greedy sequential constructions of the planes determining the various polyhedral regions [16, 19, 1] have been quite successful in obtaining very effective algorithms for training neural networks. These algorithms are much faster than the classical online backpropagation (BP) gradient algorithm [23, 11, 20], where the training is done on one point at a time. Often online BP is erroneously referred to as a descent algorithm, which it is not.

In this section of the paper we relate the polyhedral regions into which $R^n$ is partitioned, to a classical neural network with one hidden layer of linear threshold units (LTUs) and one output LTU. (See Figure 2.) An LTU is an abstraction of a human neuron which fires if its input exceeds its threshold value. Thus the LTU, depicted by its threshold value of $\theta^1$ in Figure 2, will have the output $(x^T w^1 - \theta^1)_*$, where $(\cdot)_*$ is the step function defined earlier. An obvious representation of such an LTU would be by the plane $x^T w^1 = \theta^1$. It turns out that every neural network mapping $R^n$ into the set {0,1} can be related to a partitioning of $R^n$ into polyhedral regions, but not conversely. However, any two disjoint point sets in $R^n$ can be discriminated between by *some* polyhedral partition that corresponds to a neural network with one hidden layer with a sufficient number of hidden units [12, 19].

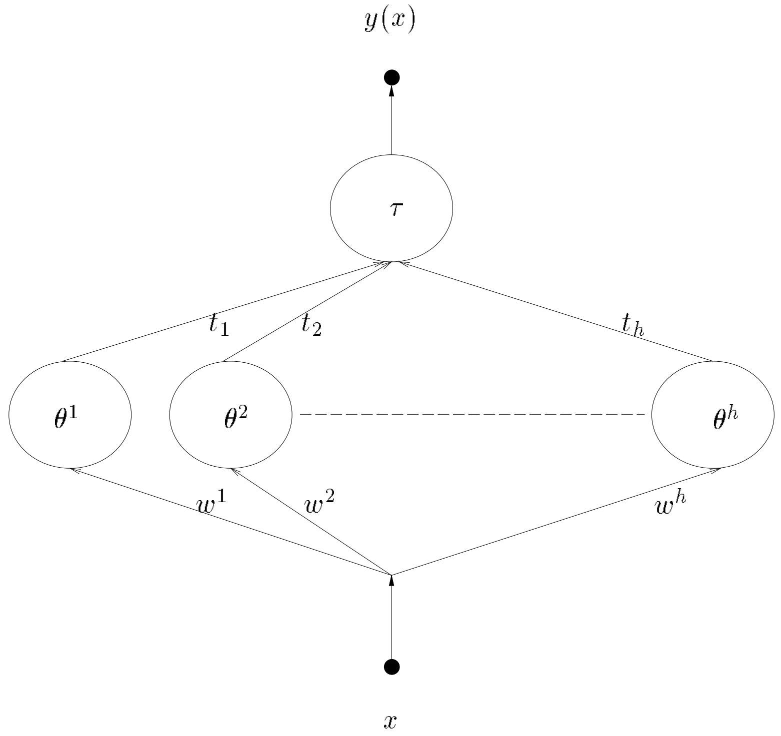


Figure 2: **A typical feedforward neural network with a single layer of $h$ hidden linear threshold units (LTUs), input $x \in R^n$, and output $y(x) \in \{0, 1\}$. The output of hidden unit $i$ is $(x^T w^i - \theta^i)_*$, $i = 1, \ldots, h$. The output $y(x)$ of the output LTU is $(\sum_{i=1}^{h}(x^T w^i - \theta^i)_* t_i - \tau)_*$.**

We describe now precisely when a specific partition of $R^n$ by $h$ separating planes

$$x^T w^i = \theta^i,\ i = 1, \ldots, h, \tag{27}$$

corresponds to a neural network with $h$ hidden units. (See Figures 1 and 2.) The $h$ separating planes (27) divide $R^n$ into at most $p$ polyhedral regions, where [9]

$$p := \sum_{i=0}^{n} \left( \begin{array}{c} h \\ i \end{array} \right). \tag{28}$$

We shall assume that $\mathcal{A}$ and $\mathcal{B}$ are contained in the interiors of two mutually exclusive subsets of these regions. (See Figure 1.) Each of these polyhedral regions can be mapped uniquely into a vertex of the unit cube in $R^h$,

$$\{z | z \in R^h,\ 0 \leq z \leq e\} \tag{29}$$

by using the map:

$$(x^T w^i - \theta^i)_*,\ i = 1, \ldots, h, \tag{30}$$

where $x$ is a point in $R^n$ belonging to some polyhedral region. If the $p$ polyhedral regions of $R^n$ constructed by the $h$ planes (27) are such that vertices of the cube (29) corresponding to points in

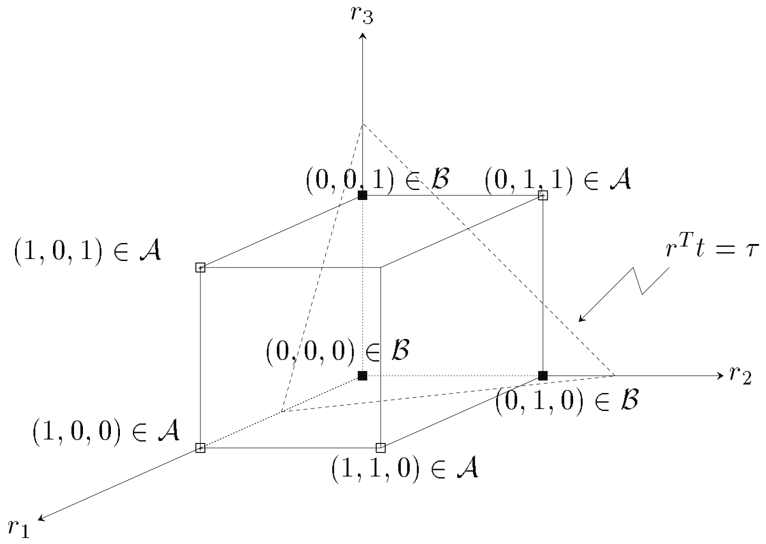


Figure 3: **The vertices of the unit cube into which the sets $\mathcal{A}$ and $\mathcal{B}$ of Figure 1 are mapped by the three planes shown in that figure, or equivalently by three hidden LTUs of a neural network. A plane, $r^T t = \tau$, separates the vertices associated with $\mathcal{A}$ from those associated with $\mathcal{B}$ . This plane corresponds to the output LTU of a neural network and the weights of its incoming arcs.**

$\mathcal{A}$ , are linearly separable in $R^h$ from the vertices of (29) corresponding to points in $\mathcal{B}$ , by a plane

$$r^T t = \tau, \tag{31}$$

as in the example of Figure 3, then the polyhedral partition of $R^n$ corresponds to a neural network with $h$ hidden linear threshold units (with thresholds $\theta^i$, incoming arc weights $w^i$, $i = 1, \ldots, h$) and output linear threshold unit (with threshold $\tau$ and incoming arc weights $t_i$, $i = 1, \ldots, h$) [17]. This condition is necessary and sufficient for the polyhedral partition of $R^n$ in order for it to correspond to a neural network with one layer of hidden units. For more details, see [17].

"Training" a neural network consists of determining $(w^i, \theta^i) \in R^{n+1}$, $i = 1, \ldots, h$, $(t, \tau) \in R^{h+1}$, such that the following nonlinear inequalities are satisfied as best as possible:

$$\begin{aligned} \sum_{i=1}^{h} (Aw^i - e\theta^i)_* t_i > e\tau \\ \sum_{i=1}^{h} (Bw^i - e\theta^i)_* t_i < e\tau \end{aligned} \tag{32}$$

This can be achieved by maximizing a weighted sum of correctly classified points in $R^h$ by solving the following unconstrained maximization problem (as in the equivalent programs (3) and (5)):

$$
\begin{array}{l}
\max\limits_{w^i,\theta^i,t_i,\tau} \; e^T(\sum\limits_{i=1}^{h}(Aw^i - e\theta^i)_* t_i - e\tau)_- \\
+e^T(\sum\limits_{i=1}^{h} -(Bw^i - e\theta^i)_* t_i + e\tau)_-,
\end{array}
\tag{33}
$$

where the function $(\cdot)_-$ is defined in (4). If instead of the step function $(\zeta)_*$ the sigmoid function $\sigma(\zeta)$ is used in (33), where $\sigma(\zeta) := \frac{1}{1+e^{-\alpha\zeta}}$, $\alpha > 0$, we obtain an error function similar to the error function that backpropagation attempts to find a stationary point for, and for which a convergence proof is given in [20], and stability analysis in [25]. We note that the classical exclusive-or (XOR) example [22] for which $A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $B = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}$, gives a maximum value of four for (33) with the following solution:

$$
\begin{array}{l}
(w^1, \theta^1) = ((2 \;\; -2),\ 1),\ (w^2, \theta^2) = ((-2 \;\; 2),\ 1) \\
(v, \tau) = ((2 \;\; 2),\ 1)
\end{array}
\tag{34}
$$

This corresponds to correctly separating the two points in $\mathcal{A}$ from the two points in $\mathcal{B}$ .

It is interesting to note that the same solution for the XOR example is given by the greedy multisurface method tree (MSMT) [1]. MSMT attempts to separate as many points of $\mathcal{A}$ and $\mathcal{B}$ as possible by a first plane obtained by solving (3), and then repeats the process for each of the ensuing halfspaces, until adequate separation is obtained. For this example, the first plane obtained [4] is $(w^1, \theta^1) = ((2 \;\; -2),\ 1)$, which separates $\{(1,0)\}$ from $\{(0,0),(0,1),(1,1)\}$. The second plane obtained is $(w^2, \theta^2) = ((-2 \;\; 2),\ 1)$, separates $\{(0,1)\}$ from $\{(0,0),(1,1)\}$, and the separation is complete between $\mathcal{A}$ and $\mathcal{B}$. These planes correspond to the same neural network obtained by solving (33), which of course is not always the case when using the greedy MSMT method. However MSMT frequently gives better solutions than those generated by BP and is much faster than BP.

We now set up the problem (33) as an MPEC. We first use the equivalence between the step function $(\cdot)_*$ and an equilibrium condition given by Lemma 2.1 and obtain the following problem, where $\epsilon$ is a sufficiently small positive number:

$$
\begin{array}{ll}
\underset{w^i,\theta^i,r^i,u^i,s^i,v^i,t_i,\tau}{\text{maximize}} & e^T(\sum\limits_{i=1}^{h} r^i t_i - e\tau)_- + e^T(\sum\limits_{i=1}^{h} -s^i t_i + e\tau)_- \\
& 0 \le r^i \perp u^i - Aw^i + e\theta^i + \epsilon e \ge 0 \\
& 0 \le u^i \perp -r^i + e \ge 0 \\
\text{subject to} & 0 \le s^i \perp v^i - Bw^i + e\theta^i + \epsilon e \ge 0 \\
& 0 \le v^i \perp -s^i + e \ge 0 \\
& i = 1, \ldots, h.
\end{array}
\tag{35}
$$

By using the equivalence between the formulations (3) and (5) we can formulate (35) as the following MPEC:

$$
\begin{array}{ll}
\underset{w^i,\theta^i,r^i,u^i,s^i,v^i,t_i,\tau,y^i,z^i}{\text{maximize}} & e^T\sum_{i=1}^h y^i + e^T\sum_{i=1}^h z^i \\
\text{subject to} & \begin{array}{l}
\sum_{i=1}^h r^i t_i - e\tau \geq y^i, \;\; y^i \leq e \\
\sum_{i=1}^h -s^i t_i + e\tau \geq z^i, \;\; z^i \leq e \\
0 \leq r^i \perp u^i - Aw^i + e\theta^i + \epsilon e \geq 0 \\
0 \leq u^i \perp -r^i + e \geq 0 \\
0 \leq s^i \perp v^i - Bw^i + e\theta^i + \epsilon e \geq 0 \\
0 \leq v^i \perp -s^i + e \geq 0 \\
i = 1, \ldots, h
\end{array}
\end{array}
\tag{36}
$$

In a manner similar to the parametric reformulation (13) of the LPEC (12) associated with the classification maximization problem (7), the above MPEC can be reformulated as the following parametric bilinear program:

$$
\begin{array}{ll}
\underset{w^i,\theta^i,r^i,u^i,s^i,v^i,t_i,\tau,y^i,z^i}{\text{minimize}} & \sum_{i=1}^h (r^i)^T(-Aw^i + e\theta^i + \epsilon e) + e^T u^i + \sum_{i=1}^h (s^i)^T(-Bw^i + e\theta^i + \epsilon e) + e^T v^i =: g(\mu) \\
\text{subject to} & \begin{array}{l}
\sum_{i=1}^h r^i t_i - e\tau \geq y^i, \;\; y^i \leq e \\
\sum_{i=1}^h -s^i t_i + e\tau \geq z^i, \;\; z^i \leq e \\
0 \leq r^i, u^i - Aw^i + e\theta^i + \epsilon e \geq 0 \\
0 \leq u^i, -r^i + e \geq 0 \\
0 \leq s^i, v^i - Bw^i + e\theta^i + \epsilon e \geq 0 \\
0 \leq v^i, -s^i + e \geq 0 \\
i = 1, \ldots, h \\
e^T\sum_{i=1}^h y^i + e^T\sum_{i=1}^h z^i \geq \mu \\
\mu \in [0, \infty)
\end{array}
\end{array}
\tag{37}
$$

Here $\mu$ is a parameter that represents the number of points correctly classified, and will equal $m+k$ if complete separation is achieved by the neural network. The largest value of $\mu$ for which the objective function has a minimum of zero is the maximum value of the MPEC (36), which corresponds to training a neural network on the sets $\mathcal{A}$ and $\mathcal{B}$ . Note that $g(\mu)$ is a nondecreasing function of $\mu$, and the largest value $\bar{\mu}$ for which $g(\bar{\mu}) = 0$, constitutes a maximum to problem (35). The parametric approach consists of starting at some large $\mu$, say $\mu = m+k$, solving (37), for decreasing values of $\mu$ for which $g(\mu) > 0$, until $\bar{\mu}$ such that $g(\bar{\mu}) = 0$ is reached. Efficient estimation of successive values of $\mu$ can be achieved by a secant method applied to the nondecreasing function $g(\mu)$. Note that the nonconvex problem (37) has a bilinear objective and two sets of bilinear constraints. Although no computation has been done with this model of a neural network, it is felt that the Frank-Wolfe approach utilized to solve efficiently numerous NP-complete problems in [5] could also be effective here as well. Briefly the approach would consist of fixing $t_i,\ i = 1,\ldots,h$ and solving (37) by the bilinear approach of [5] which involves successive linear programs and line searches. Then $(t_i, y^i, z^i)\ \ i = 1,\ldots,h$ and $\tau$ are updated by solving a single linear program. The bilinear approach corresponds to adjusting the thresholds and incoming arc weights for the hidden units of the neural network as well as adjusting the threshold of the output unit, while holding the weights of the incoming arcs to the output unit fixed. The linear program then attempts to get a best linear separation between vertices of the unit cube in $R^h$ that represent $\mathcal{A}$ and $\mathcal{B}$ , by readjusting the threshold of the output unit of the neural network as well as the weights of its incoming arcs.

# 5 Conclusion

Significant problems associated with machine learning have been cast as a variety of mathematical programs, ranging in complexity from polynomial-time-solvable linear programs to NP-complete problems. Effective methods for solving some of these problems have been outlined. Modeling and efficiently solving many of these problems constitute an important and challenging field of research for mathematical programming.

# References


[1] 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, Utica, Illinois, 1992.

[2] K. P. Bennett and E. J. Bredensteiner. A parametric optimization method for machine learning. Department of Mathematical Sciences Report No. 217, Rensselaer Polytechnic Institute, Troy, NY 12180, 1994.

[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] K. P. Bennett and O. L. Mangasarian. Bilinear separation of two sets in n-space. *Computational Optimization & Applications*, 2:207–227, 1993.

[6] 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, D.C., 1964. Spartan Books.

[7] Chunhui Chen and O. L. Mangasarian. Hybrid misclassification minimization. Technical Report 95-05, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, February 1995. Advances in Computational Mathematics, submitted. Available from ftp://ftp.cs.wisc.edu/math-prog/tech-reports/95-05.ps.Z.

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

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

[10] David Heath. *A geometric Framework for Machine Learning.* PhD thesis, Department of Computer Science, Johns Hopkins University–Baltimore, Maryland, 1992.

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

[12] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[13] Y. le Cun, J. S. Denker, and S. A. Solla. Optimal brain damage. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems II (Denver 1989)*, pages 598–605, San Mateo, California, 1990. Morgan Kaufmann.

[14] Z.-Q. Luo, J.-S. Pang, D. Ralph, and S.-Q. Wu. Exact penalization and stationarity conditions of mathematical programs with equilibrium constraints. Technical Report 275, Communications Research Laboratory, McMaster University, Hamilton, Ontario, Hamilton, Ontario L8S 4K1, Canada, 1993. Mathematical Programming, to appear.

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

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

[17] O. L. Mangasarian. Mathematical programming in neural networks. *ORSA Journal on Computing*, 5(4):349–360, 1993.

[18] O. L. Mangasarian. Misclassification minimization. *Journal of Global Optimization*, 5:309–323, 1994.

[19] 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, *Large-Scale Numerical Optimization*, pages 22–31, Philadelphia, Pennsylvania, 1990. SIAM. Proceedings of the Workshop on Large-Scale Numerical Optimization, Cornell University, Ithaca, New York, October 19-20, 1989.

[20] O. L. Mangasarian and M. V. Solodov. Serial and parallel backpropagation convergence via nonmonotone perturbed minimization. *Optimization Methods and Software*, 4(2):103–116, 1994.

[21] O. L. Mangasarian, W. Nick Street, and W. H. Wolberg. Breast cancer diagnosis and prognosis via linear programming. Technical Report 94-10, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin 53706, 1994. Operations Research 43(4) 1995, to appear. Available from ftp://ftp.cs.wisc.edu/math-prog/tech-reports/94-10.ps.Z.

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

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

[24] C. Schaffer. Overfitting avoidance as bias. *Machine Learning*, 10:153–178, 1993.

[25] M. V. Solodov and S. K. Zavriev. Stability properties of the gradient projection method with applications to the backpropagation algorithm. Computer Sciences Department, Mathematical Programming Technical Report 94-05, University of Wisconsin, Madison, Wisconsin, June 1994. SIAM Journal on Optimization, submitted.

[26] M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society*, 36:111–147, 1974.

[27] W. Nick Street and O. L. Mangasarian. Improved generalization via tolerant training. Technical report, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, 1995. To appear.

[28] D. H. Wolpert, editor. *The Mathematics of Generalization*, Reading, MA, 1995. Addison-Wesley.