M. A. Sartori and P. J. Antsaklis, "A New Test for Linear Separability and Solution of the Classification Problem," Technical Report # 90-10-02, Dept. of Electrical Engineering, University of Notre Dame, October 1990.

A NEW TEST FOR LINEAR SEPARABILITY AND SOLUTION OF THE CLASSIFICATION PROBLEM

Michael A. Sartori and Panos J. Antsaklis
Department of Electrical Engineering
University of Notre Dame
Notre Dame, Indiana 46556

ABSTRACT

A new solution to the linear separability problem is given. As is known, only training sets that are linearly separable can be implemented with a single neuron, or a single layer neural network. A new test to determine whether a given training set is linearly separable is developed; the test is necessary and sufficient. Linear programming techniques are employed to implement the test and analyze the problem. If the training set is linearly separable, the weights for a neuron, or a single layer of neurons, that correctly implement the separation of the training set are a by-product of the test.

1 INTRODUCTION

It is important to know if a classification training set is linearly separable, and if it is linearly separable, to have an efficient technique to compute the weights of a neuron or a single layer neural network to implement the set. In this way, if the classification training set is linearly separable, a single neuron or a single layer neural network can be used instead of a multi-layer neural network, and the significant computational effort typically associated with the training of the multi-layer neural network with the back-propagation algorithm can be avoided. The purpose of this paper is two-fold: First, to stress that there exist other methods besides the commonly used gradient descent procedures to test for linear separability, namely linear programming techniques. Secondly, to present one such method which has advantages over previous methods.

As is well known for classification training sets, a single neuron can only implement those which are linearly separable [1-4]; the classification training set consists of input patterns and desired output values that describe which of two possible sets the corresponding input patterns belong. Many previous methods to determine linear separability assumed that a threshold logic gate was used in conjunction with a switching function (that is, a function with binary inputs and binary outputs), and some of these may be extended to real-valued functions. For instance, in [1], linear programming techniques as well as a method based on the development of a "function tree" were presented to determine if a switching function is linearly separable. The linear separability of a switching function is tested via the successive-elimination-of-variables method in [3]. In [5], to determine if a switching function was linearly separable, the switching function is iteratively examined for invariant variable combinations and reduced accordingly. Gradient descent methods have also been employed to determine the linear separability of a training set. The Perceptron Convergence Theorem of [6] with its proof of convergence if the training set is linearly separable, the least mean squares procedure of [7], and the Ho-Kashyap algorithm of [8,9] are three such examples. A good collection of procedures to solve the linear separability problem is contained in [10].

In this paper, the linear separability problem is formulated as a linear programming problem. With this, the testing of linear separability of the training set is equivalent to the determination of a feasible solution of the linear programming problem; if a feasible solution exists, the training set is linearly separable, and if not, the training set is not linearly separable. Compared to other methods, such as Rosenblatt's Perceptron Convergence Theorem which only terminates if the training set is linearly separable, the linear programming problem has well understood terminating criterion. The linear programming formulation also has a clear computational advantage as there are powerful methods, such as the simplex method, to solve linear programming problems of large dimensions. In addition, by reducing the problem to such a well studied mathematical problem, the results already developed for linear programming can be applied to the linear separability problem examined here. For example, if only part of the given training set changes, the linear programming solution does not need to be recalculated from the beginning. Also, sensitivity, or postoptimality, analysis can be used to study variations in the training set and their effect on the linear programming solution.

The linear separability problem formulated and solved as a linear programming problem has p constraints and p variables, where p is the number of patterns. This is different from the earlier formulation of [10] where the linear programming problem has p constraints and 2m variables, where m is the number of weights. Furthermore, the approach here has the advantage of dealing directly with the test patterns rather than with the weights, which is convenient in deciding the location of the separation surface. By using the linear optimization function of the linear programming formulation, the choice of different feasible solutions is possible, which results in a choice of different separating hyperplanes.

The test for the single neuron is extended to the single layer neural network; the linear separability test for the single neuron is applied individually to each neuron in the single layer. If the test is successful for each neuron, then the training set is linearly separable, and the weights which implement the single layer neural network can be easily computed.

In Section 2, a single neuron and the problem of finding its weights are discussed. The problem of linear separability is introduced in Section 3 where a new test for linear separability is developed and formally stated in Theorem 3.2; it is also shown that the appropriate weights of the neuron are easily obtained as a by-product of the test. In Section 4, the implementation of the linear separability test via linear programming is described. Two examples illustrating the linear separability test for a single neuron are presented in Section 5. A single layer neural network is then discussed in Section 6, and the linear separability test developed for the single neuron is extended to this case in Theorem 6.1.

2 THE NEURON

The neuron considered here is described by

$$y = f\Big(\sum_{i=1}^{m} u_i\, w_i\Big) = f(\mathbf{u}'\mathbf{w}), \tag{1}$$

where $f:\mathbb{R} \rightarrow \mathbb{R}$ is the nonlinearity of the neuron, $\mathbf{u} := [u_1, ..., u_m]' \in \mathbb{R}^{m\mathrm{x}1}$ is the input vector, $\mathbf{w} := [w_1, ..., w_m]' \in \mathbb{R}^{m\mathrm{x}1}$ is the weight vector, and $u_m = 1$ is the bias input for the neuron. The type of nonlinear function considered here for the neuron must satisfy three conditions stated in the next section. This is not however a significant restriction since those nonlinearities commonly used, such as the hyperbolic tangent and signum functions, all satisfy these conditions.

Assume that a training set $\{\mathbf{u}(j), d(j)\}$ for $1 \leq j \leq p$ consists of p pairs of input vectors and desired output scalars, where $\mathbf{u}(j) \in \mathbb{R}^{m\mathrm{x}1}$, $u_m(j) = 1$, and $d(j) \in \mathbb{R}$ for $1 \leq j \leq p$. The Neuron Training Problem (N) is defined as follows:

$$\left.\begin{array}{c} \min\limits_{\mathbf{w}} \hat{F}(\mathbf{w}) \\ \\ \hat{F}(\mathbf{w}) = (\mathbf{d} - \phi(\mathbf{U}'\mathbf{w}))'(\mathbf{d} - \phi(\mathbf{U}'\mathbf{w})) \end{array}\right\} \tag{N}$$

where $\mathbf{d} := [d(1), ..., d(p)]' \in \mathbb{R}^{p\mathrm{x}1}$ is the desired output vector, $\mathbf{U} := [\mathbf{u}(1), ..., \mathbf{u}(p)] \in \mathbb{R}^{m\mathrm{x}p}$ is the matrix of input vectors, and $\phi(\mathbf{z}) := [f(z_1), ..., f(z_p)]' \in \mathbb{R}^{p\mathrm{x}1}$ with $\mathbf{z} = [z_1, ..., z_p]' \in \mathbb{R}^{p\mathrm{x}1}$. The notation $\phi(\mathbf{z})$ represents a map which takes a p-dimensional vector z and returns another p-dimensional vector with elements $f(z_i)$, where f is the neuron's nonlinearity. As can be easily shown, $\hat{F}(W)$ in (N) is actually the sum of the squares of the error between the desired output and the output of the neuron:

$$\hat{F}(W) = \sum_{j=1}^{p} (d(j) - f(\mathbf{u}(j)'\mathbf{w}))^2. \tag{2}$$

For the neuron to accurately classify the p input patterns $\mathbf{u}(j)$, it must produce outputs $y(j)$ not necessarily equal to the given $d(j)$ but "close enough" so that the inputs are classified in the correct set. Next, the case when the training set can be exactly implemented via a single neuron is examined.

Since

$$\hat{F}(\mathbf{w}) \geq 0 \tag{3}$$

for any $\mathbf{w}$, if a $\mathbf{w}$ exists such that

$$\hat{F}(\mathbf{w}) = 0, \tag{4}$$

then this $\mathbf{w}$ minimizes $\hat{F}(\mathbf{w})$ and solves (N). In this case, the output of the neuron exactly matches the desired one.

Theorem 2.1:

If there exists a **w** such that

$$\mathbf{U}'\mathbf{w} = \mathbf{v} \tag{5}$$

where $\phi(\mathbf{v}) = \mathbf{d}$, then

$$\hat{F}(\mathbf{w}) = 0. \tag{6}$$

Proof:

Applying the neuron's nonlinearity to both sides of (5),

$$\phi(\mathbf{U}'\mathbf{w}) = \phi(\mathbf{v}) = \mathbf{d} \tag{7}$$

or

$$\mathbf{d} - \phi(\mathbf{U}'\mathbf{w}) = \mathbf{0}. \tag{8}$$

Substituting (8) into (N),

$$\hat{F}(\mathbf{w}) = 0. \ \blacklozenge \tag{9}$$

It is known from the theory of linear algebraic equations that (5) has a solution if and only if rank[**U**':**v**] = rank[**U**']. Next, two cases are examined: (i) when there are at least as many weights as there are patterns and (ii) when there are more patterns than weights.

Case (i): If there are at least as many weights as there are patterns, that is $m \geq p$, and rank[**U**'] = p, then a solution **w** to (5) always exists for any **v**. In this case, there is typically an infinite number of solutions **w**. The following corollary states this result; the proof is obvious.

Corollary 2.2:

If rank[**U**'] = $p \leq m$, then there always exist **w** such that $\hat{F}(\mathbf{w}) = 0$; these **w** are solutions to (5).

That is, given any training set, with rank[**U**'] = $p \leq m$, it can always be implemented via a single neuron. There are, in general, an infinite number of weights **w*** which can accomplish this, namely, solving Problem (N) such that $\hat{F}(\mathbf{w}^*) = 0$ where **w*** is any solution of (5). Clearly, for a classification training set with rank[**U**'] = $p \leq m$, the neuron can correctly classify the given input patterns.

Case (ii): If there are more patterns than weights, that is $p > m$, then there is no guarantee that rank[**U**':**v**] = rank[**U**'] or that (5) will have a solution for a given **v**. In this case, a solution to (N) with zero error does not necessarily exist. When the training set is a classification training set, this also implies that the set may or may not be linearly separable.

M. A. Sartori and P. J. Antsaklis, "A New Test for Linear Separability and Solution of the Classification Problem," Technical Report # 90-10-02, Dept. of Electrical Engineering, University of Notre Dame, October 1990.

In this paper, given that the training set is a classification training set, a test for the linear separability of the set is developed, and the necessary weights to correctly classify the set via a single neuron are a by-product of the test; this test is the subject of the next section.

3 THE TEST FOR LINEAR SEPARABILITY

To perform the linear separability test, let there be more patterns than weights, that is $p > m$. After the linear separability test has been successfully completed, the solution $\mathbf{w}$ to (N) can be easily computed.

Definition 3.1:

A training set $\{\mathbf{u}(j),d(j)\}$ for $1 \le j \le p$ consisting of p pairs of input vectors and desired output scalars is called a classification training set if and only if the input vectors are considered to be in one of two distinct sets, say T or F.

Depending on the particular nonlinearity of the neuron, the distinction between the sets T and F can be specified in different manners. If the hyperbolic tangent function or signum function is used, then the distinction between the sets T and F can be defined as:

$$\begin{cases} \text{if } \mathbf{u}(j) \in T \text{ then } d(j) > 0 \\ \text{if } \mathbf{u}(j) \in F \text{ then } d(j) < 0 \end{cases} . \tag{10}$$

If the sigmoid function, $f(x) = 1/(1 + e^{-x})$, is used, then the distinction between T and F can be defined as:

$$\begin{cases} \text{if } \mathbf{u}(j) \in T \text{ then } d(j) > 0.5 \\ \text{if } \mathbf{u}(j) \in F \text{ then } d(j) < 0.5 \end{cases} . \tag{11}$$

However, by appropriately scaling and shifting the sigmoid to $2f(x) - 1$, the distinction between T and F described by (11) can be given by (10). The linear separability test developed here does not depend on the particular nonlinearity used for the neuron; the goal of training the neuron is to classify the input of the neuron as being in one class or the other. In the following, the distinction described by (10) is used.

Let the classification training set $\{\mathbf{u}(j),d(j)\}$ for $1 \le j \le p$, denoted herein as $(\mathbf{U}, \mathbf{d})$, be given. As above, $\mathbf{U} = [\mathbf{u}(1), ..., \mathbf{u}(p)] \in \mathbb{R}^{m\text{x}p}$ and $\mathbf{d} = [d(1), ..., d(p)]' \in \mathbb{R}^{p\text{x}1}$.

Definition 3.2:

A classification training set $(\mathbf{U}, \mathbf{d})$ is linearly separable if and only if there exists a hyperplane

$$\sum_{i=1}^{m} u_i w_i = 0. \tag{12}$$

that separates the input vectors belonging to the set T from those belonging to the set F.

It is known that there exists a hyperplane that separates the input vectors belonging to the set T from those belonging to the set F if and only if

$$\begin{array}{lcl} u_1(1)w_1 + u_2(1)w_2 + ... + u_m(1)w_m \ (< \text{or} >)\ 0 & & d(1)\ (< \text{or} >)\ 0 \\ u_1(2)w_1 + u_2(2)w_2 + ... + u_m(2)w_m \ (< \text{or} >)\ 0 & \text{and} & d(2)\ (< \text{or} >)\ 0 \\ \vdots & & \vdots \\ u_1(p)w_1 + u_2(p)w_2 + ... + u_m(p)w_m \ (< \text{or} >)\ 0 & & d(p)\ (< \text{or} >)\ 0 \end{array} \qquad (13)$$

is satisfied where (< or >) is appropriately assigned according to (**U**, **d**). It is clear that a classification training set is linearly separable if and only if (13) is satisfied. If a w can be found which satisfies (13), then this w is the weight vector which can be used to implement the neuron. Thus, (13) describes the necessary and sufficient conditions for the linearly separability of the classification training set. However, this does not necessarily imply that f(u(j)'w) equals the desired vector d(j) from the training set.

Next, the conditions and nomenclature for the linear separability test are developed. Consider a single neuron described by (1) with the following restrictions on its nonlinearity:

(i) $f(-z) = -f(z)$,

(ii) $f(z) > 0$ if and only if $z > 0$, and

(iii) $f(z) < 0$ if and only if $z < 0$.

These conditions imply that the graph of the neuron's nonlinearity lies only in the first and third quadrants and is symmetric about the origin. The conditions also place no restrictions on the input set (e.g., the input set does not need to be binary). Given the desired output vector **d**, construct a diagonal matrix $\mathbf{E} \in \mathbb{R}^{p\mathrm{x}p}$ with diagonal elements

$$e_{jj} = \text{signum}(d(j)), \qquad (14)$$

that is $e_{jj} = 1$ if $d(j) > 0$ and $e_{jj} = -1$ if $d(j) < 0$ for $1 \le j \le p$ and with the rest of the elements in **E** being zero. The matrix is denoted as **E** = **E**(**d**) and has the properties that **E** = **E**' and **EE** = **I**. Writing the left hand side of (13) as the product U'w and pre-multiplying by E, the following is obtained:

$$\tilde{U}'w > 0 \qquad (15)$$

where $\tilde{\mathbf{U}}' := \mathbf{EU}' \in \mathbb{R}^{p\mathrm{x}m}$ and $\mathbf{0} := [0, ..., 0]' \in \mathbb{R}^{p\mathrm{x}1}$; note that $\tilde{\mathbf{d}} = \mathbf{Ed} > \mathbf{0}$ with $\tilde{\mathbf{d}} \in \mathbb{R}^{p\mathrm{x}1}$ by the definition of **E**.

Equation (15) capsulizes the linear separability conditions for the neuron given a particular classification training set since the information in **d** needed for classification is incorporated in **E**.

Lemma 3.1:

Given (**U**, **d**), define **E** = **E**(**d**) and let $\tilde{\mathbf{U}}' := \mathbf{EU}'$. The classification training set (**U**, **d**) is linearly separable if and only if there exists a w which satisfies (15).

Given the classification training set (**U**, **d**), define **E** = **E**(**d**) as above. Let rank[**U**'] = m so that **UU**' is nonsingular. Define

$$\tilde{M} := (EU)'(U^+E) = \tilde{U}'\tilde{U}^+ \tag{16}$$

where $\tilde{M} \in \mathbb{R}^{pxp}$ and $U^+ := (UU')^{-1}U \in \mathbb{R}^{mxp}$ is the pseudo-inverse of U'. The test for linear separability is presented in the next theorem.

Theorem 3.2:

The classification training set (U, d) is linearly separable if and only if there exists a $\tilde{v} \in \mathbb{R}^{px1}$ which satisfies

$$\tilde{M}\tilde{v} > 0 \text{ and } \tilde{v} > 0. \tag{17}$$

Furthermore, if the set is linearly separable, the weight vector of the neuron can be computed by

$$w = \tilde{U}^+\tilde{v}. \tag{18}$$

The proof of Theorem 3.2 is based on the following lemma. Assume that the classification training set (U, d) is given, and let E = E(d) and $\tilde{U}' = EU'$.

Lemma 3.3:

$$\tilde{U}'w > 0 \text{ and } \tilde{y} > 0 \tag{19}$$

where $w = \hat{U}v$ for some $\hat{U} \in \mathbb{R}^{mxp}$ and $v \in \mathbb{R}^{px1}$, $y = \phi(v)$ and $\tilde{y} = Ey$, if and only if

$$\tilde{M}\tilde{v} > 0 \text{ and } \tilde{v} > 0 \tag{20}$$

where $\tilde{M} := \tilde{U}'\hat{U}E \in \mathbb{R}^{mxp}$ and $\tilde{v} = Ev$.

Proof:

(⇒) Let (19) be true. Then

$$\tilde{U}'w = \tilde{U}'\hat{U}v = (\tilde{U}'\hat{U}E)(Ev) = \tilde{M}\tilde{v} \tag{21}$$

since EE = I; so $\tilde{U}'w > 0$ implies $\tilde{M}\tilde{v} > 0$. Note that $E\phi(v) = \phi(Ev) = \phi(\tilde{v})$ in view of condition (i) for the neuron's nonlinearity; $\phi(v) = y$ now implies that $E\phi(v) = Ey = \tilde{y}$, that is $\phi(\tilde{v}) = \tilde{y}$. With $\tilde{y} > 0$, conditions (ii) and (iii) imply that $\tilde{v} > 0$.

(⇐) Let (20) be true. Clearly, $\tilde{M}\tilde{v} > 0$ implies that $\tilde{U}'w > 0$ in view of the definitions of $\tilde{M}$ and $\tilde{v}$. Similarly, as above, in view of conditions (ii) and (iii), $\tilde{v} > 0$ implies that $\phi(\tilde{v}) > 0$. In view of assumption (i), $\phi(\tilde{v}) = E\phi(v)$. With $\phi(v) = y$, $\phi(\tilde{v}) = Ey = \tilde{y}$; therefore $\tilde{y} > 0$. ♦

In the proof of Lemma 3.3, note that $\tilde{y} = Ey > 0$ where E = E(d) implies that y and d have exactly the same sign for the corresponding elements. Next, using Lemma 3.3, the proof for Theorem 3.2 is given.

Proof:

(⇒) Assume that (U, d) is linearly separable, that is (15) is satisfied for some w (Lemma 3.1); note that Ed > 0. Define

$$v := U'w \text{ and } y := \phi(v). \tag{22}$$

In general, $y \neq d$. However, y and d have the same sign in each entry ($E(d) = E(y)$) and $Ey > 0$ as is now shown:

$$\tilde{y} = Ey = E\phi(v) = \phi(Ev) \tag{23}$$

because of condition (i). Also,

$$\phi(Ev) = \phi(EU'w) = \phi(\tilde{U}'w), \tag{24}$$

that is, $\tilde{y} = \phi(\tilde{U}'w)$. Since $\tilde{U}'w > 0$, in view of conditions (ii) and (iii), $\tilde{y} = \phi(\tilde{U}'w) > 0$. So,

$$\tilde{U}'w > 0 \text{ and } \tilde{y} > 0$$

where w satisfies $\tilde{U}'w = v$ and $y = \phi(v)$. In view of Lemma 3.3, this implies that

$$\tilde{M}\tilde{v} > 0 \text{ and } \tilde{v} > 0$$

where $\tilde{M} = EU'(U^+E) = \tilde{U}'\tilde{U}^+$ and $\tilde{v} = Ev$. Note that $w = \tilde{U}^+\tilde{v} = U^+EEv = U^+v$ satisfies $U'w = v$ exactly because of the way v was defined.

($\Leftarrow$) Assume there exists a $\tilde{v}$ which satisfies

$$\tilde{M}\tilde{v} > 0 \text{ and } \tilde{v} > 0.$$

Let $v = E\tilde{v}$ and define $y := \phi(v)$, $\tilde{y} := Ey$, and $w := U^+v$. In view of Lemma 3.3,

$$\tilde{U}'w > 0 \text{ and } \tilde{y} > 0$$

is true, which implies that U and d are linearly separable in view of Lemma 3.1.

If (U, d) is linearly separable and, hence, (17) is satisfied, $w = \tilde{U}^+\tilde{v} = U^+EEv = U^+v$ is an appropriate weight vector for the neuron since it satisfies (15). ♦

For a particular solution $\tilde{v}$ of (17) and $w = \tilde{U}^+\tilde{v}$, the actual output of the neuron is given by:

$$y = \phi(U'\tilde{U}^+\tilde{v}) = \phi(E\tilde{M}\tilde{v}). \tag{25}$$

In general, $y \neq d$. However, the corresponding elements of y and d have the same sign, and, as shown in the proof of Theorem 3.2, the neuron with the appropriate set of weights will classify the training set correctly. Clearly, if the nonlinearity is the signum function and the training set is linearly separable, $y = d$; so, if $w = \tilde{U}^+\tilde{v}$ is used with the signum function, $y = d$. If the same w is used with a different nonlinearity, $y \neq d$ may be the case, but clearly the training set is classified correctly. This is illustrated in Example 5.1.

In the next section, performing the linear separability test via linear programming is discussed.

4 LINEAR SEPARABILITY VIA LINEAR PROGRAMMING

In this section, linear programming methods are employed to determine the existence of a $\tilde{v}$ such that (17) is satisfied. The linear programming formulation of interest here is:

$$\begin{array}{ll} \text{minimize} & c'x \\ \text{subject to} & Ax \geq b \\ & x \geq 0 \end{array} \tag{LP}$$

where $A \in \mathbb{R}^{pxp}$, $x \in \mathbb{R}^{px1}$, $c \in \mathbb{R}^{px1}$, and $b \in \mathbb{R}^{px1}$. In order to implement the linear separability test via linear programming, (17) must be in the form of (LP). The following lemma aids in this.

Lemma 4.1:

There exists a $\tilde{v}$ such that

$$\tilde{M}\,\tilde{v} > 0 \text{ and } \tilde{v} > 0, \tag{17}$$

if and only if there exists a $\tilde{v}$ such that

$$\tilde{M}\,\tilde{v} \geq \underline{\sigma} \text{ and } \tilde{v} \geq \underline{\tau} \tag{26}$$

where $\sigma > 0$, $\tau > 0$, $\underline{\sigma} := [\sigma, ..., \sigma]' \in \mathbb{R}^{px1}$, and $\underline{\tau} := [\tau, ..., \tau]' \in \mathbb{R}^{px1}$.

Proof:

($\Rightarrow$) Clearly, if (17) is true for some $\tilde{v}$, then there exist positive numbers σ and τ such that (26) is true for the same $\tilde{v}$; note that in some cases σ and τ may be small numbers.

($\Leftarrow$) If (26) is true for some $\tilde{v}$, then (17) is true since σ and τ are both positive. ♦

With Lemma 4.1, the linear separability test of Theorem 3.2 can be implemented with linear programming. Choose a vector $\underline{\sigma} := [\sigma, ..., \sigma]' \in \mathbb{R}^{px1}$ such that $\sigma > 0$ and choose another vector $\underline{\tau} := [\tau, ..., \tau]' \in \mathbb{R}^{px1}$ such that $\tau > 0$. Then, solving

$$\begin{array}{ll} \text{minimize} & c'\tilde{v} \\ \text{subject to} & \tilde{M}\tilde{v} \geq \underline{\sigma} \\ & \tilde{v} \geq \underline{\tau} \end{array} \tag{27}$$

ensures that $\tilde{M}\tilde{v} > 0$ and $\tilde{v} > 0$. Actually, via Lemma 4.2 in the following Section 4.1, σ and τ are not restricted to be small in magnitude as is suggested by Lemma 4.1. Using the standard linear programming manipulation of converting lower-bounded variables to nonnegative variables, a new variable is defined: $\hat{\tilde{v}} := \tilde{v} - \underline{\tau}$ and $\hat{\tilde{v}} \geq 0$ is equivalent to $x \geq 0$ of (LP). Rewriting (27), the linear programming problem becomes:

$$\begin{array}{ll} \text{minimize} & c'(\tilde{v} - \underline{\tau}) \\ \text{subject to} & \tilde{M}(\tilde{v} - \underline{\tau}) \geq (\underline{\sigma} - \tilde{M}\ \underline{\tau}) \\ & (\tilde{v} - \underline{\tau}) \geq 0 \end{array} \tag{28}$$

Comparing (28) to (LP), $A = \tilde{M}$, $x = (\tilde{v} - \underline{\tau})$, and $b = (\underline{\sigma} - \tilde{M}\ \underline{\tau})$. The choice of the vector c is left to the particular implementation.

Applying linear programming to solve (28), the determination of the existence of a $\tilde{v}$ which satisfies (17) is accomplished. If a $\tilde{v}$ can not be found that satisfies (28), that is a feasible solution to (28) does not exist, then the training set is not linearly separable. If a $\tilde{v}$ can be found that satisfies (28), then the training set is linearly separable, and the weights that implement the neuron are given by (18).

4.1 Discussion

In this section, the solutions determined via linear programming are examined in three respects. First, it is shown that σ and τ can be chosen arbitrarily ($\sigma > 0$, $\tau > 0$). Secondly, with the choice of σ arbitrary, the magnitudes of the individual elements of the output vector $\mathbf{y}$ can be arbitrarily bounded from below, which implies an increase in the robustness of the output. Thirdly, with the choice of $\mathbf{c}$ in the optimization function unrestricted, particular feasible solutions can be chosen over others.

First, via Lemma 4.2, it is shown that σ and τ can be chosen arbitrarily.

Lemma 4.2:

There exists a $\tilde{\mathbf{v}}$ such that

$$\tilde{\mathbf{M}}\tilde{\mathbf{v}} \geq \underline{\sigma} \text{ and } \tilde{\mathbf{v}} \geq \underline{\tau}, \tag{26}$$

where $\sigma > 0$ and $\tau > 0$ if and only if there exists a $\approx{\mathbf{v}}$ such that

$$\tilde{\mathbf{M}}\overset{\approx}{\mathbf{v}} \geq \underline{\eta} \text{ and } \overset{\approx}{\mathbf{v}} \geq \underline{\mu} \tag{29}$$

where $\eta = \sigma\gamma > 0$, $\mu = \tau\gamma > 0$, $\gamma > 0$, $\underline{\eta} := [\eta, \ldots, \eta]' \in \mathbb{R}^{p\times 1}$, and $\underline{\mu} := [\mu, \ldots, \mu]' \in \mathbb{R}^{p\times 1}$. Furthermore, the weights $\mathbf{w} = \tilde{\mathbf{U}}^{+}\tilde{\mathbf{v}}$ and $\overset{\approx}{\mathbf{w}} = \tilde{\mathbf{U}}^{+}\overset{\approx}{\mathbf{v}}$ are related by

$$\overset{\approx}{\mathbf{w}} = \mathbf{w}\gamma. \tag{30}$$

Proof:

($\Rightarrow$) Assume that (26) is true. Multiplying all of (26) by $\gamma > 0$,

$$\tilde{\mathbf{M}}\tilde{\mathbf{v}}\gamma \geq \underline{\sigma}\gamma \text{ and } \tilde{\mathbf{v}}\gamma \geq \underline{\tau}\gamma. \tag{31}$$

Letting $\overset{\approx}{\mathbf{v}} = \tilde{\mathbf{v}}\gamma$,

$$\tilde{\mathbf{M}}\overset{\approx}{\mathbf{v}} \geq \underline{\sigma}\gamma = \underline{\eta} \text{ and } \overset{\approx}{\mathbf{v}} \geq \underline{\tau}\gamma = \underline{\mu}. \tag{32}$$

($\Leftarrow$) Clearly, the reverse is true by multiplying (29) by $1/\gamma$ for $\gamma > 0$.

Furthermore, $\overset{\approx}{\mathbf{w}} = \tilde{\mathbf{U}}^{+}\overset{\approx}{\mathbf{v}} = \tilde{\mathbf{U}}^{+}\tilde{\mathbf{v}}\gamma = \mathbf{w}\gamma$. ♦

Thus, by scaling σ and τ by an arbitrary $\gamma > 0$, both $\tilde{\mathbf{v}}$ and $\mathbf{w}$ are also scaled by γ. If there exists a $\tilde{\mathbf{v}}$ such that (26) is true for a particular σ and τ, then there exists a $\tilde{\mathbf{v}}$ such that (26) is true for any σ and τ. With respect to solving (27) using linear programming, the choice of σ and τ is not important; if the training set is linearly separable, then there exists a $\tilde{\mathbf{v}}$ such that (26) is true for arbitrary σ and τ. In terms of the separating hyperplane described by $\mathbf{w}$, *the choice of γ does not move the hyperplane but merely scales the weights by γ*, which increases the robustness of the output of the neuron as is shown next.

Secondly, the significance of the choice of σ and its relation to the neuron's output are examined. As stated in (25), the output of the neuron is given by

$$\mathbf{y} = \phi(\mathbf{E}\tilde{\mathbf{M}}\tilde{\mathbf{v}}).$$

In view of (27), the magnitude of the individual elements can be bounded from below by:

$$\mathbf{E}\mathbf{y} = \phi(\tilde{\mathbf{M}}\tilde{\mathbf{v}}) \geq \phi(\underline{\sigma}). \tag{33}$$

Thus, the choice of σ bounds the output y. With the choice of σ arbitrarily large, the magnitude of the elements of y can be forced to be arbitrarily large by increasing the robustness of the neuron's output.

Thirdly, with the formulation of the linear separability test of (17) as the linear programming problem of (27), the choice of different feasible solutions is possible via the vector c in the linear optimization function. Since the jth element of $\tilde{v} \in \mathbb{R}^{p\times 1}$ corresponds directly to the jth pattern of the training set and since $\tilde{v}$ is closely related to the output of the neuron, the manipulation of c causes certain feasible solutions to be chosen over others. For instance, if all the elements of the output need to be bounded from below by the same value, as with $\underline{\sigma}$ of (33), but the magnitudes of certain elements need to be smaller than others, this can be accomplished through the choice of c; if the magnitude of the desired output y(j) needs to be as small as possible compared to the magnitudes of the other patterns, choose the element c_j of c larger than the other components of c. As another example, if the magnitude of the desired output y(j) needs to be as large as possible compared to the magnitudes of the other patterns, choose the element c_j of c smaller than the other components of c. This is illustrated in Example 5.2. These types of choices of feasible solutions are not possible with the linear programming formulation of [10], as is discussed next.

4.2 Relation to Previous Linear Programming Formulation

Clearly, with Lemma 3.1, the linear separability problem can be solved by implementing (15) with linear programming (LP). Defining two new variables w^+ and w^- in the standard linear programming manipulation such that

$$w = w^+ - w^- \tag{34}$$

and using Lemma 4.1, the linear programming formulation of (15) is

$$\begin{aligned} &\text{minimize} && c'\begin{bmatrix} w^+ \\ w^- \end{bmatrix} \\ &\text{subject to} && [\tilde{U}' \; -\tilde{U}']\begin{bmatrix} w^+ \\ w^- \end{bmatrix} \geq \underline{\alpha} \\ & && \begin{bmatrix} w^+ \\ w^- \end{bmatrix} \geq 0 \end{aligned} \tag{35}$$

Comparing (35) to (LP), $A = [\tilde{U}' \; -\tilde{U}']$, $x = [w^{+\prime} \; w^{-\prime}]'$, and $b = \underline{\alpha}$. The variables in this formulation are w^+ and w^- , and it is not clear how to appropriately select c in the optimization function to stress certain patterns over other ones, which was accomplished with the previous linear programming formulation; it is not easy to manipulate the physical variables of the problem by selecting parameters in (35). In previous formulations [10], it is suggested that c be set to zero; the existence of a feasible solution is of main interest with this formulation.

Comparing (27) and (35), both require p constraints, but (27) has p variables while (35) has 2m variables. Furthermore, the optimization function is not utilized in (35) but is utilized in (27). As discussed previously, with the use of the optimization function in (27), the choice of feasible solutions is possible, and hence the choice of the placement of the separating hyperplane is possible.

5 EXAMPLES

Example 5.1:

In [11], the following input and output training patterns are shown to be linearly separable but to cause local minima entrapment for a gradient descent algorithm; in fact, this example was used to illustrate the drawbacks of the back-propagation algorithm:

$$\mathbf{U}' = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{5\times 4}$$

and

$$\mathbf{d} = [1\ 1\ 1\ -1\ -1]' \in \mathbb{R}^{5\times 1}.$$

If the input vectors are plotted on the vertices of a cube, it is clear that there exists a hyperplane that separates the two sets of input vectors. Solving Theorem 3.2 using (27) with $\mathbf{c} \in \mathbb{R}^{5\times 1}$ chosen arbitrarily as a vector of units, $\sigma = 1$, and $\tau = 1$, a $\tilde{\mathbf{v}} > 0$ is found, and the training set is linear separable. Using $\mathbf{v} = \mathbf{E}\tilde{\mathbf{v}}$,

$$\mathbf{v} = [1\ \ 5\ \ 1\ \ -1\ \ -1]'. \tag{36}$$

Using (18), the weight vector is

$$\mathbf{w} = [2\ \ 2\ \ 4\ \ -3]', \tag{37}$$

and $\mathbf{U}'\mathbf{w} = \mathbf{v}$. The training set and the separating plane described by $\mathbf{w}$ are shown in Figure 1. With the signum as the nonlinearity, the output of the neuron is

$$\mathbf{y} = [1\ \ 1\ \ 1\ \ -1\ \ -1]' = \mathbf{d}.$$

If the hyperbolic tangent is used instead of the signum function as the neuron's nonlinearity, then the output of the neuron is

$$\mathbf{y} = [0.7616\ \ 0.9999\ \ 0.7616\ \ -0.7616\ \ -0.7616]' \neq \mathbf{d}, \tag{38}$$

but signum(y(j)) = signum(d(j)) for $1 \le j \le p$; that is, the neuron classified correctly, as expected.

If (27) is solved with $\mathbf{c} \in \mathbb{R}^{5\times 1}$ as the unit vector, $\sigma = 10$, and $\tau = 10$,

$$\mathbf{v} = [10\ \ 50\ \ 10\ \ -10\ \ -10]',$$

$$\mathbf{w} = [20\ \ 20\ \ 40\ \ -30]',$$

and $\mathbf{U}'\mathbf{w} = \mathbf{v}$; this is equivalent to scaling $\mathbf{v}$ and $\mathbf{w}$ of (36) and (37) by $\gamma = 10$ as predicted in Lemma 4.2. With this weight vector, the same separating plane as in Figure 1 results. With the hyperbolic tangent function as the neuron's nonlinearity, the output of the neuron is

$$\mathbf{y} = [0.9999\ \ 1.0000\ \ 0.9999\ \ -0.9999\ \ -0.9999]' \neq \mathbf{d},$$

and the robustness of the result is increased compared to (38).

Example 5.2:

A 4x4 retina, as described in [12], is used in this example. Twenty-four different patterns are used in the classification training set as shown in Figure 2. Six different letters are used, and each is translated four times over the retina. The input patterns generated for these patterns consist of either a 1 (black) or a -1 (white) and are formed by copying the elements of the retina matrix row-wise into a vector such that $U \in \mathbb{R}^{17x24}$ is the input matrix. It is desired to map the letters X, T, and C to one class and the letters L, J, and H to another. With the signum used as the neuron's nonlinearity, the desired scalars associated with the input patterns for X, T, and C are assigned a 1, and those associated with the input patterns for L, J, and H are assigned a -1 such that $d \in \mathbb{R}^{24x1}$ is the vector of desired outputs. The input space is of high dimension, and it is unclear if the training set is linearly separable. Applying Theorem 3.2 and (27) with $c \in \mathbb{R}^{24x1}$ as the unit vector, $\sigma = 1$, and $\tau = 1$, a $\tilde{v} > 0$ is found, which implies that the training set is linearly separable. Using $v = E\tilde{v}$,

$$v = \begin{matrix} 1.3333 \\ 1.0000 \\ 1.0000 \\ 1.0000 \\ 8.8333 \\ 8.0000 \\ 1.5000 \\ 1.0000 \\ 1.0000 \\ 1.0000 \\ 1.0000 \\ 1.0000 \\ -1.3333 \\ -1.0000 \\ -1.0000 \\ -1.0000 \\ -14.8333 \\ -1.0000 \\ -3.5000 \\ -1.3333 \\ -1.0000 \\ -1.0000 \\ -1.0000 \\ -2.5000 \end{matrix}$$

Using (18), the weight vector is

$$w = \begin{matrix} 5.6667 \\ 7.6667 \\ 1.7500 \\ -0.7500 \\ 2.2500 \\ -2.5000 \\ -0.5000 \\ 1.5000 \\ -0.5000 \\ -5.0000 \\ 0.2500 \\ 3.0833 \\ -1.2500 \\ 3.9167 \\ 5.7500 \\ 1.3333 \\ -3.3333 \end{matrix}$$

Furthermore,

$$U'w = \begin{bmatrix} 1.0000 \\ 1.0000 \\ 1.0000 \\ 1.0000 \\ 8.8333 \\ 7.6667 \\ 1.8333 \\ 1.0000 \\ 1.0000 \\ 1.6667 \\ 1.0000 \\ 1.0000 \\ -1.0000 \\ -1.3333 \\ -1.0000 \\ -1.0000 \\ -14.8333 \\ -1.3333 \\ -3.5000 \\ -1.0000 \\ -1.0000 \\ -1.0000 \\ -1.0000 \\ -2.8333 \end{bmatrix} \tag{39}$$

and the output of the neuron is

$$\mathbf{y} = [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ -1\ -1\ -1\ -1\ -1\ -1\ -1\ -1\ -1\ -1\ -1\ -1]' = \mathbf{d}.$$

As explained in Section 4.1, by changing c, the choice of the feasible solution can be changed. As an example, suppose it is desired that the magnitudes of the neuron's outputs for the T input patterns are as small as possible compared to the magnitudes of the outputs for the remaining input patterns. Further, assume that the magnitudes of the outputs should be at least one. To satisfy these requirements, choose $\mathbf{c} = [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1000\ 1000\ 1000\ 1000\ 1\ 1\ 1\ 1]'$ and $\sigma = \tau = 1$. Solving Theorem 3.2 using (27), a $\tilde{v} > \mathbf{0}$ is found. Using $v = E\tilde{v}$,

$$v = \begin{bmatrix} 1.0000 \\ 1.0000 \\ 1.0000 \\ 1.0000 \\ 8.7500 \\ 12.5000 \\ 10.5000 \\ 13.0000 \\ 1.0000 \\ 1.0000 \\ 6.5000 \\ 1.0000 \\ -2.7500 \\ -1.0000 \\ -1.0000 \\ -1.0000 \\ -1.0000 \\ -1.0000 \\ -1.0000 \\ -1.0000 \\ -1.0000 \\ -1.0000 \\ -1.0000 \\ -2.7500 \end{bmatrix} \tag{40}$$

Using (18), the weight vector is

$$\mathbf{w} = \begin{matrix} -1.8750 \\ 3.6250 \\ 0.2500 \\ 1.6250 \\ -0.6250 \\ -3.8750 \\ -1.8750 \\ 2.8750 \\ 0.8750 \\ -5.0000 \\ 1.7500 \\ 3.1250 \\ -2.7500 \\ -0.5000 \\ 8.6250 \\ 1.2500 \\ -9.2500 \end{matrix}$$

To examine the magnitude of the neuron's output, $\mathbf{U}'\mathbf{w} = \mathbf{v}$ here, and the constraints are clearly satisfied. Comparing elements 17 thru 20 of (39) and (40), a feasible solution with the desired properties has been selected by the choice of $\mathbf{c}$.

6 SINGLE LAYER NEURAL NETWORK

In this section, the results for a single neuron are extended to a single layer neural network. First, the single layer and its corresponding training problem are described. The linear separability for the single layer neural network is presented in Theorem 6.1; the single neuron's test is performed on each neuron in the layer.

The <u>single layer neural network</u> is comprised of n parallel neurons each described by

$$y_i = f(\mathbf{u}'\mathbf{w}_i) \tag{41}$$

for $1 \le i \le n$. For the i^{th} neuron, the function $f:\mathbb{R} \to \mathbb{R}$ is the nonlinearity of the neuron, $\mathbf{u} := [u_1, ..., u_m]' \in \mathbb{R}^{m\times 1}$ is the input vector, $\mathbf{w}_i := [w_{1,i}, ..., w_{m,i}]' \in \mathbb{R}^{m\times 1}$ is the weight vector, and $u_m = 1$ is the bias input for the neuron. Once again, the type of nonlinear function considered here for the neuron must satisfy the three conditions stated in Section 3.

Assume that a training set consisting of p pairs of input vectors and desired output vectors $\{\mathbf{u}(j), \mathbf{d}(j)\}$ for $1 \le j \le p$ is given, where $\mathbf{u}(j) \in \mathbb{R}^{m\times 1}$, $u_m(j) = 1$, and $\mathbf{d}(j) = [d_1(j), ..., d_n(j)]' \in \mathbb{R}^{n\times 1}$ for $1 \le j \le p$. The output of the single layer neural network is described by

$$\mathbf{Y} = \Phi(\mathbf{U}'\mathbf{W}) \tag{42}$$

where $\mathbf{Y} := [\mathbf{y}_1, ..., \mathbf{y}_n]' \in \mathbb{R}^{p\times n}$ is the matrix of the single layer's outputs, $\mathbf{y}_i := [y_i(1), ..., y_i(p)]' \in \mathbb{R}^{p\times 1}$ for $1 \le i \le n$ is the vector of a particular neuron's output, $\mathbf{U} := [\mathbf{u}(1), ..., \mathbf{u}(p)]' \in \mathbb{R}^{m\times p}$ is the matrix of input vectors, $\mathbf{W} := [\mathbf{w}_1, ..., \mathbf{w}_n] \in \mathbb{R}^{m\times n}$ is the matrix of weight vectors, and $\Phi(\mathbf{Z}) := [\phi(\mathbf{z}_1), ..., \phi(\mathbf{z}_n)] \in \mathbb{R}^{p\times n}$ with $\mathbf{Z} := [\mathbf{z}_1, ..., \mathbf{z}_n] \in \mathbb{R}^{p\times n}$. The notation $\Phi(\mathbf{Z})$ represents a map which takes a matrix $\mathbf{Z}$ with elements z_{ji} and returns another matrix of the same size with elements $f(z_{ji})$, where f is the neuron's nonlinearity. The Single Layer Neural Network Training Problem (L) is defined as follows:

M. A. Sartori and P. J. Antsaklis, "A New Test for Linear Separability and Solution of the Classification Problem," Technical Report # 90-10-02, Dept. of Electrical Engineering, University of Notre Dame, October 1990.

$$\left.\begin{aligned} &\min_{W} \hat{F}(W) \\ &\hat{F}(W) = \mathrm{tr}((D - \Phi(U'W))'(D - \Phi(U'W))) \end{aligned}\right\} \tag{L}$$

where "tr" is the trace of a square matrix, $D := [d_1, ..., d_n] \in \mathbb{R}^{p\times n}$ is the matrix of desired outputs, $d_i := [d_i(1), ..., d_i(p)]' \in \mathbb{R}^{p\times 1}$ for $1 \le i \le n$ are the desired output vectors. Note that with $n = 1$, (L) reduces to (N). In equation (L), $\hat{F}(W)$ is actually a sum of the squares of the error between the individual desired output elements and the outputs of the neurons:

$$\hat{F}(W) = \sum_{k=1}^{n} \sum_{j=1}^{p} (d_k(j) - f(u(j)'w_k))^2. \tag{43}$$

Given the classification training set (U, D), define $E_i = E_i(d_i)$ for $1 \le i \le n$ and let $\tilde{U}_i' = E_iU'$ for $1 \le i \le n$. Let rank[U'] = m. Define

$$\tilde{M}_i := (E_iU)'(U^+E_i) = \tilde{U}_i'\tilde{U}_i^+ \tag{44}$$

for $1 \le i \le n$ where $U^+ := (UU')^{-1}U$ is the pseudo-inverse of U'. The test for linear separability is presented in the next theorem.

Theorem 6.1:

The classification training set (U, D) is linearly separable if and only if there exists a $\tilde{v}_i \in \mathbb{R}^{p\times 1}$ for $1 \le i \le n$ which satisfies

$$\tilde{M}_i\tilde{v}_i > 0 \text{ and } \tilde{v}_i > 0. \tag{45}$$

Furthermore, if the set is linearly separable, the vectors of the weight matrix W of the single layer can be computed by

$$w_i = \tilde{U}_i^+\tilde{v}_i. \tag{46}$$

Proof:

The proof is based on applying Lemma 3.3 to each neuron in the layer and following the proof of Theorem 3.2 for each neuron in the layer. ♦

For a particular solution $\tilde{v}_i$ and $w_i = \tilde{U}_i^+\tilde{v}_i$ for $1 \le i \le n$, the output of the ith neuron is given by:

$$y_i = \phi(U'\tilde{U}_i^+\tilde{v}_i) = \phi(E_i\tilde{M}_i\tilde{v}_i) \tag{47}$$

for $1 \le i \le n$.

The test for linear separability for the single layer neural network can be implemented with linear programming by using the method discussed in Section 4 for the single neuron. Using the linear programming formulation

$$\begin{aligned} &\text{minimize} && c'\tilde{v}_i \\ &\text{subject to} && \tilde{M}_i\tilde{v}_i \ge \underline{\sigma} \\ & && \tilde{v}_i \ge \underline{\tau} \end{aligned} \tag{48}$$

for each neuron in the single layer (which is the same formulation for the single neuron given by (27) in Section 4), the existence of a $\tilde{v}_i$ such that (45) is satisfied for $1 \le i \le n$ can be determined.

7 CONCLUDING REMARKS

A new solution to the linear separability problem is presented here for both a single neuron and a single layer neural network. When there are at least as many m weights as there are p patterns, that is $m \ge p$, and rank[U'] = p, a training set is always implementable by either a single neuron or a single layer neural network and is always linearly separable. For the case where there are more patterns than weights, that is $p > m$, a new test is provided to determine if a given classification training set is linearly separable; the test is necessary and sufficient. If the training set is linearly separable, the weights for a neuron or a single layer that correctly classifies the training set are a by-product of the test. The test can be implemented using linear programming. This, in addition to numerical advantages, reduces the problem to a very well studied mathematical problem, and the variety of results developed for linear programming, such as postoptimality analysis, can be used to study the linear separability problem. Furthermore, through the use of the linear optimization function, different feasible solutions are possible, and hence different separating hyperplanes are possible. Note that some of the results in this paper have first appeared in [13] and [14].

It is suggested here that if a classification training set is given, the test in Theorem 3.2, or in Theorem 6.1, should first be conducted. The linear separability of the training set can thus be immediately determined. This simple procedure could save a large amount of time. For a training set that is linearly separable, instead of training a multi-layer neural network with the back-propagation algorithm, the test described in this paper can first be performed, and the weights which implement this training set via a single neuron or a single layer neural network can be derived.

Acknowledgement

This work was supported in part by the Jet Propulsion Laboratory, Pasadena, CA under contract No. 957856.

8 REFERENCES

[1] Lewis P.M., Coates C.L., *Threshold Logic*, Wiley, New York, 1967.
[2] Kohavi Z., *Switching and Finite Automata Theory*, McGraw-Hill, New York, 1978.
[3] Trong H.C., *Theory and Logic Design*, Addison-Wesley, Reading, MA, 1972.
[4] Hill F.J., Peterson G.R., *Introduction to Switching Theory and Logic Design*, Wiley, New York, 1968.
[5] Minsky M.L., Papert P., *Perceptrons: An Essay in Computational Geometry*, MIT Press, Cambridge, MA, 1969.
[6] Rosenblatt F., *Principles of Neurodynamics*, Spartan Books, Washington, D.C., 1962.
[7] Widrow B., Hoff M.E., "Adaptive Switching Circuits," *1960 IRE WESCON Convention Record*, Part 4, pp 96-104, August 1990.
[8] Ho Y.-C., Kashyap R.L., "An Algorithm for Linear Inequalities and Applications," *IEEE Transactions of Elec. Comp.*, vol EC-14, pp 683-688, October 1965.

[9] Ho Y.-C., Kashyap R.L., "A Class of Iterative Procedures for Linear Inequalities," *J. SIAM Control*, vol. 4, pp 112-115, 1966.
[10] Duda R.O., Hart P.E., *Pattern Classification and Scene Analysis*, Wiley, New York, 1973.
[11] Brady M.L., Raghavan R., Slawny J., "Back Propagation Fails to Separate Where Perceptrons Succeed," *IEEE Transactions on Circuits and Systems*, vol. 36, May 1989, pp. 665-674.
[12] Widrow B., "ADALINE and MADALINE -- 1963", *Proceedings of the 1987 IEEE International Conference on Neural Networks*, vol 1, 1987, pp 145-157.
[13] Sartori M.A., Antsaklis P.J., "Perceptron Learning via Quadratic Optimization," Technical Report # 90-05-01, Department of Electrical Engineering, University of Notre Dame, May 1990.
[14] Sartori M.A., Antsaklis P.J., "A New Test for Linear Separability and Solution of the Classification fo the Classification Problem," Technical Report # 90-10-02, Department of Electrical Engineering, University of Notre Dame, October 1990.

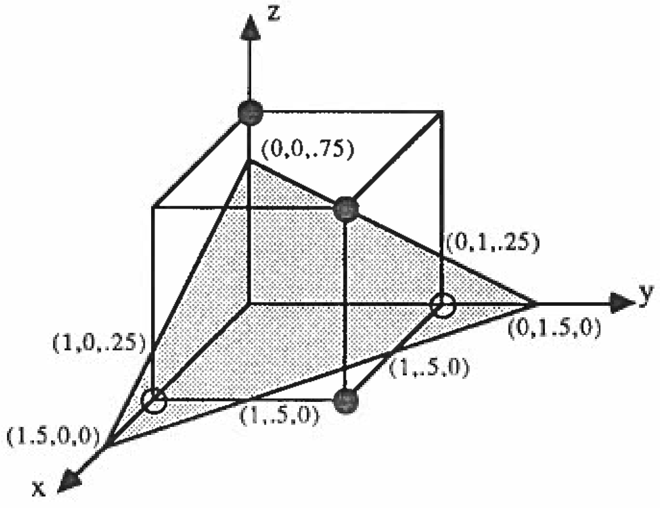

Figure 1 Training set and separating hyperplane found with $\sigma = 1$ and $\tau = 1$.

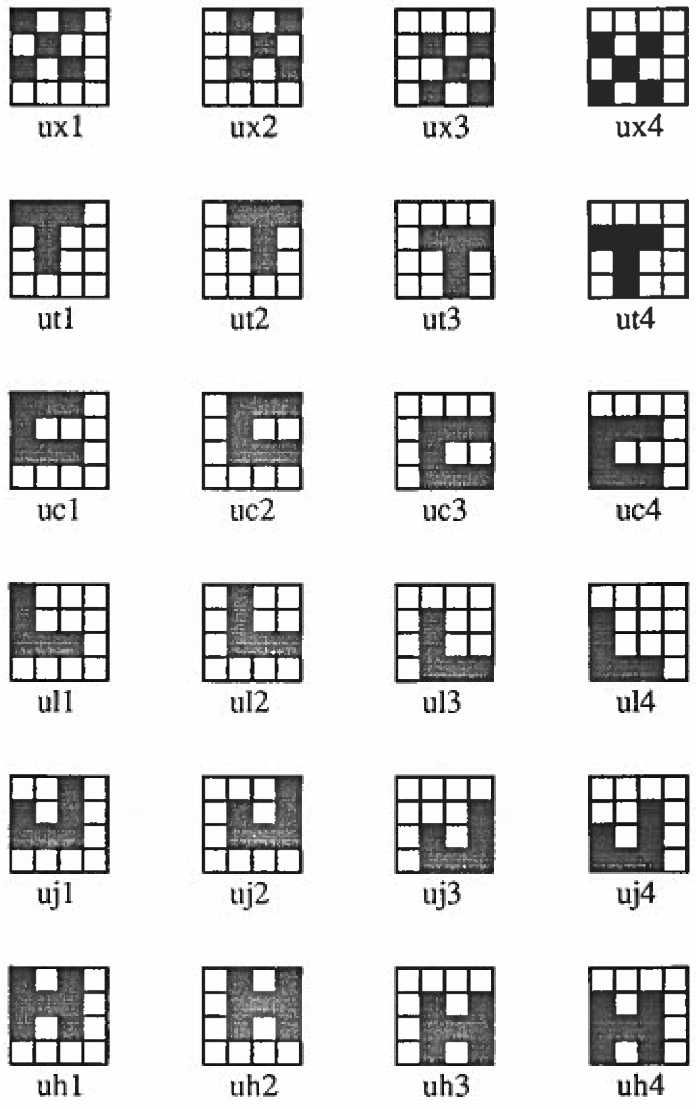

Figure 2 Retina patterns for the letters X, T, C, L, J, and H.