7-1-1992

Classification of Linearly Non-Separable Patterns by Linear Threshold Elements

VWANI P. ROYCHOWDHURY
Purdue University, School of Electrical Engineering

KAI-YEUNG SIU
Purdue University, School of Electrical Engineering

THOMAS KAILATH
Purdue University, School of Electrical Engineering

Classification of Linearly Non-Separable Patterns by Linear Threshold Elements

Vwani P. Roychowdhury
Kai-Yeung Siu
Thomas Kailath

TR-EE 92-28
July 1992

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907-1285

Classification of Linearly Non-Separable Patterns by Linear Threshold Elements

Vwani P. Roychowdhury * Kai-Yeung Siu † Thomas Kailath ‡

Email: vwani@ecn.purdue.edu

Abstract

Learning and convergence properties of linear threshold elements or perceptrons are well understood for the case where the input vectors (or the training sets) to the perceptron are linearly separable. However, little is known about the behavior of a linear threshold element when the training sets are ***linearly non-separable.*** In this paper we present the first known results on the structure of linearly non-separable training sets and on the behavior of perceptrons when the set of input vectors is linearly non-separable. More precisely, we show that using the well known perceptron learning algorithm a linear threshold element can learn the input vectors that are provably learnable, and identify those vectors that cannot be learned without committing errors. We also show how a linear threshold element can be used to learn large linearly separable subsets of any given non-separable training set. In order to develop our results, we first establish formal characterizations of linearly non-separable training sets and define learnable structures for such patterns. We also prove computational complexity results for the related learning problems. Next, based on such characterizations, we show that ***a perceptron does the best*** one can expect for linearly non-separable sets of input vectors and learns as much as is theoretically possible.

*Currently with the School of Electrical Eng., Purdue University, West Lafayette, IN 47907.

†Currently with the Department of Electrical and Computer Eng., University of California at Irvine, Irvine, CA 92717.

‡With the Information Systems Laboratory, Stanford University, Stanford CA 94305.

1 Introduction

A Linear 'Threshold Element (LTE) or a perceptron has been a widely studied model for neural computation since its introduction by McCulloch and Pitts in 1943 [8]. Moreover, one of the early results in machine learning involves the perceptron learning algorithm developed by Rosenblatt in 1960 [13] and other related algorithms (see [3, 15, 16]). The seminal work of Rosenblatt showed that a simple algorithm can be used to train an LTE to learn an input pattern, *if* such a pattern is learnable. However, the behavior of linear threshold elements when fed with patterns that it cannot learn, is not well understood. This paper answers some of the open questions in this regard for the first time.

The problem of learning in a perceptron can be simply stated as follows: given a set of m input vectors $\{X_1, \cdots, X_m\}$ in $\mathcal{R}^d$ (this set of input vectors is often referred to as a training set), determine a hyperplane such that each vector X_i lies on a pre-assigned side of the hyperplane (*i.e.*, above or below the hyperplane). If such a hyperplane exists for the given set of vectors (i.e., the training set) then the set of input vectors is referred to as a linearly *separable* training set, and the hyperplane is referred to as a *separating* hyperplane. For linearly separable training sets, the perceptron learning algorithm uses simple relaxation type operations to determine such a separating hyperplane in a finite number of steps [10].

A set of input vectors (or a training set) will be said to be linearly non-separable if no hyperplane exists such that each vector lies on the pre-assigned side of the hyperplane. As mentioned above, the learning and convergence properties of perceptrons are well understood when the training sets are linearly separable. However, very little is known about the behavior of perceptrons when the input patterns are linearly non-separable. In this paper we present new results regarding: (1) the learnable structures for linearly non-separable patterns, (2) the computational complexities of the related learning problems, (3) the behavior of perceptrons when the input patterns are linearly non-separable; in particular, we show that the perceptron learning algorithm, which successfully converges when input patterns are linearly separable, also derives all the 'learnable' information from linearly non-separable input patterns, and (4) the application of perceptron and other learning algorithms in learning a large linearly separable sub-set of the given linearly non-separable training set.

In order to define learnable structures for linearly non-separable patterns, we first develop a

necessary itnd sufficient condition for linear non-separability. Based on this analysis we are able to separate the input vectors into two classes: (1) Non-separable input vectors and (2) separable input *vectors.* Mathematical definitions for such classifications are given in Section 4; however, brief descriptions of each of these classes can be presented here as follows.

1. *Non-separable* input vectors are the ones that are responsible for linear non-separability and cannot be learned without forcing errors on the rest of the inputs. That is, if a non-separable vector is correctly 'learned' (*i.e.* it is assigned to the designated side of a hyperplane) then there must exist at least another input vector which is on the wrong side (*i.e.*, opposite to its designated side) of the hyperplane. Thus learning a non-separable vector forces an error to be committed in the learning process.

2. *Separable* input vectors, on the other hand, are those that are provably not responsible for linear non-separability and can be learned (*i.e.*, put on the pre-assigned or designated side of a hyperplane) without forcing errors in the learning process of the rest of the input vectors.

The above mentioned analysis of linearly non-separable sets of vectors is independent of any particular learning algorithm and can be useful for analyzing the performance of all learning algorithms.

Based on the above classifications, we can define one of the learning objectives for linearly non-separable input patterns as determining the sets of separable and non-separable vectors. We shall show that this is the most that an uncommitted learning system (see Remark 4) can learn from a linearly non-separable training set. In Sections 3 and 5 we also present additional arguments (based on the computational complexities of the related learning problems) justifying why such a learning objective is the most reasonable that one can expect from a simple system such as a single linear threshold element. Such learning would allow the system to identify the structure of the training set and determine those vectors that are responsible for non-separability. Furthermore, such information would indicate as to which vectors to be dropped from the set so as to make the rest of the vectors linearly separable. This property can be also used to learn a large linearly sepamble subset of the given non-separable training set.

Our results show that if the well known perceptron algorithm is applied to linearly non-separable input patterns, then it can learn the set of separable vectors and identify the set of non-separable vectors in the training set. Hence, the perceptron does the best one can expect for linearly non-separable training sets, and derives all the information that is theoretically feasible. Moreover, we

also show how to use the perceptron learning algorithm to learn a large linearly separable subset of the given non-separable training set.

The rest of the paper is organized as follows. In Section 2 we provide some basic definitions and relevant previous results. In Section 3 we provide a technical summary of the results of this paper.

Section 4 introduces the linear programming tools that are necessary for developing our results. In addition to leading to new results, our linear programming formalism would often yield simple proofs and increased understanding for some known results in the literature. We present a necessary and sufficient condition for linear non-separability and develop a theory for identifying structures within linearly non-separable training sets.

In Section 5 we shall define two learning problems for linearly non-separable sets, and ***establish their computational complezities.*** In particular, we shall show that one of the learning problems can be solved by efficient polynomial time algorithms. We shall also show that a second related learning problem is harder and is NP-complete. These results complement results reported in other articles [14, 2, 9, 1, 7] on the computational complexity of learning problems arising in networks of LTEs.

Section. 6 establishes the behavior of perceptrons when the training set is linearly non-separable. A dual learning problem is developed and it is shown that a simultaneous execution of a dual algorithm might increase the efficiency of the perceptron learning algorithm. Finally, Section 7 has some concluding remarks.

2 Definitions and Background

Let $\{X_1, X_2, \cdots, X_m\}$ be the set of input vectors to a linear threshold element (LTE), where each input X_i is a d-dimensional (column) vector; thus the LTE has d-inputs. If the weight associated with the k^{th} input of the LTE is denoted by w_k, then the output y_i of the LTE corresponding to an input vector X_i is given by

$$y_i = sgn(\sum_{k=1}^{d} w_k X_{ik} - t) = \begin{cases} 1 & \text{if } (\sum_{k=1}^{d} w_k X_{ik} - t) > 0 \\ -1 & \text{if } (\sum_{k=1}^{d} w_k X_{ik} - t) < 0 \end{cases}$$

where t is the threshold value. Note that without loss of generality (see [10]) we have assumed that $(\sum_{k=1}^{d} w_k X_{ik} - t) \neq 0$

Remark 1 Without loss of generality the threshold value t can be assumed to equal zero. This can be simply achieved by increasing the dimension of every input vector by augmenting it with -1, and by increasing the dimension of the weight vector by augmenting it with t. Then the output of the perceptron can be written as

$$y_i = sgn(\sum_{k=1}^{d+1} w_k X_{ik}) = sgn(\mathbf{w}^T X_i)$$

where, $\mathbf{w}^T = [w_1 \cdots w_d\ t]$ and $X_i^T = [X_{i1} \cdots X_{id}\ -1]$.

The problem of learning in perceptron can now be defined as follows:

Problem 1 Given a set of vectors $\{X_1, X_2, \cdots, X_m\}$, $X_i \in \mathcal{R}^d$, and a set of desired output values $\{y_1, y_2, \cdots y_m\}$, $y_i \in \{+1, -1\}$, determine a weight vector $w \in \mathcal{R}^d$, if there exists one, such that $sgn(\mathbf{w}^T X_i) = y_i$ for all $1 \leq i \leq m$.

Remark 2 If every input vector, X_i, that is assigned to $y_i = -1$ is replaced by $-X_i$ then Problem 1 can be equivalently stated as follows.

Problem 2 Given a set of vectors $\{X_1, X_2, \cdots, X_m\}$, $X_i \in R^d$, determine a weight vector $\mathbf{w} \in \mathcal{R}^d$ such that $\mathbf{w}^T X_i > 0$ for all $1 \leq i \leq m$.

Thus the learning problem reduces to determining a hyperplane in R^d (whose normal is given by the vector w) such that all the input vectors lie on one side of it.

Definition 1 A set of vectors $\{X_1, X_2, \cdots, X_m\}$, $X_i \in \mathcal{R}^d$ is said to be linearly separable if and only if there exists a vector $w \in \mathcal{R}^d$ such that $X_i^T \mathbf{w} > 0$ for all $1 \leq i \leq m$.
A set of vectors, $\{X_1, X_2, \cdots, X_m\}$, $X_i \in \mathcal{R}^d$, is said to be linearly *non-separable* if it is not linearly separable.

Remark 3 In the above definition, the desired separating hyperplane is constrained to go through the origin and all the vectors in the given set are required to lie on one side of it. However, as explained in Remarks 1 and 2, this is a general situation if the vectors are appropriately preprocessed. In this paper we shall assume that the given input vectors X_i always satisfy the properties in Remarks 1 and 2; see Example 1.

In [13] (see also [10, 3]) Rosenblatt proposed a simple algorithm, that learns a weight vector $\mathbf{w}$ for a given set of *linearly separable input vectors.* The *perceptron learning algorithm* can be simply stated as follows:

START: Choose an arbitrary weight vector $\mathbf{w}_0$.

TEST: Choose an arbitrary X_i, and

If $X_i\mathbf{w}_l \leq 0$, then go to ADD; Else go to TEST.

ADD: $\mathbf{w}_{l+1} = \mathbf{w}_l + X_i$

It has been shown that if the set of input vectors is linearly separable then the above learning algorithm converges in a *finite number of steps.* In other words, the above algorithm will go to ADD only a finite number of steps.

3 Sunnmary of Results

This paper addresses the issue of learning for linearly non-separable training sets. In Sections 4 and 5 we shall establish some basic properties about the structure and information content of linearly non-separable sets, *These results are independent of any learning algorithm.* In Section 6 we shall discuss how the perceptron learning algorithm can be used to learn the structure of linearly non-separable sets.

In Section 4 we first establish a necessary and sufficient condition for linear non-separability based on a linear programming formulation.

Theorem. 1 A set of vectors is linearly non-separable if and only if there exists a positive linear combination of the vectors that equals 0, i.e., $\exists\, q \neq 0$ such that

$$\sum_{i=1}^{m} q_i X_i = 0 \quad \text{and} \quad q_i \geq 0.$$

As explained in Remark 3, the above theorem is applicable for any general set of vectors, as long as the vectors are appropriately modified (see Remarks 1 and 2).

An indirect form of Theorem 1 appears in the early literature on threshold logic (see for example, [11]) for the special case of Boolean vectors, i.e., when entries of X_i are restricted to binary values 0 and 1. Using the duality theory of linear programming (Section 4) and the formalism used in

this paper, we shall give a much simpler proof; moreover our result is valid when the X_i's have arbitrary real-valued entries.

We next explore possible structures within a linearly non-separable set of vectors. In fact, closer observation reveals that if a set of input vectors is linearly non-separable then ***not all the vectors in the set are responsible for non-separability.*** Based on our analysis, in Section 4.1 we provide the following classification of the vectors in a linearly non-separable set:

Given a linearly non-separable set, $S = \{X_1, X_2, \cdots, X_n\}$, a vector $X_i \in S$ will be said to be ***separable*** if and only if it never participates in a positive linear combination of the vectors in S that equals 0. That is,

$$\sum_{j=1}^{m} q_j X_j = 0; \quad q_j \geq 0 \quad \Longrightarrow \quad q_i = 0 .$$

Similarly, a vector X_i in the set S will be said to be ***non-separable*** if and only if it participates in a positive linear combination that equals 0. That is, there exists a, $q_i > 0$ such that $\sum_{j=1}^{m} q_j X_j = 0; \quad q_j \geq 0.$

In Section 4.1 (see Theorem 2) we show that the set of separable and non-separable vectors have the following properties:

1. Suppose, if one were to determine a separating hyperplane for the given linearly non-separable set then it must be the case that some of the vectors do not lie on the designated side of it. If a vector lies on the pre-specified/designated side of the hyperplane then we will consider the given vector to have been learned. Now Theorem 2 shows the following:
   If a ***non-separable vector*** lies on the designated side of a hyperplane then, there must exist another vector which must lie on the wrong side. Thus, ***if one wants to learn a non-separable vector, then one must commit at least one error.***

2. On the other hand Theorem 2 shows that one can separate all the separable vectors without committing errors.

In fact Theorem 2 proves something very surprising: there exists a hyperplane such that all the ***separable vectors lie on one side of it*** and ***all*** the ***non-separable vectors lie on it.*** This result can be interpreted as follows: there is a vector w, such that it separates all the separable vectors and remains ambiguous with respect to the non-separable ones. This is illustrated in Fig. 1.

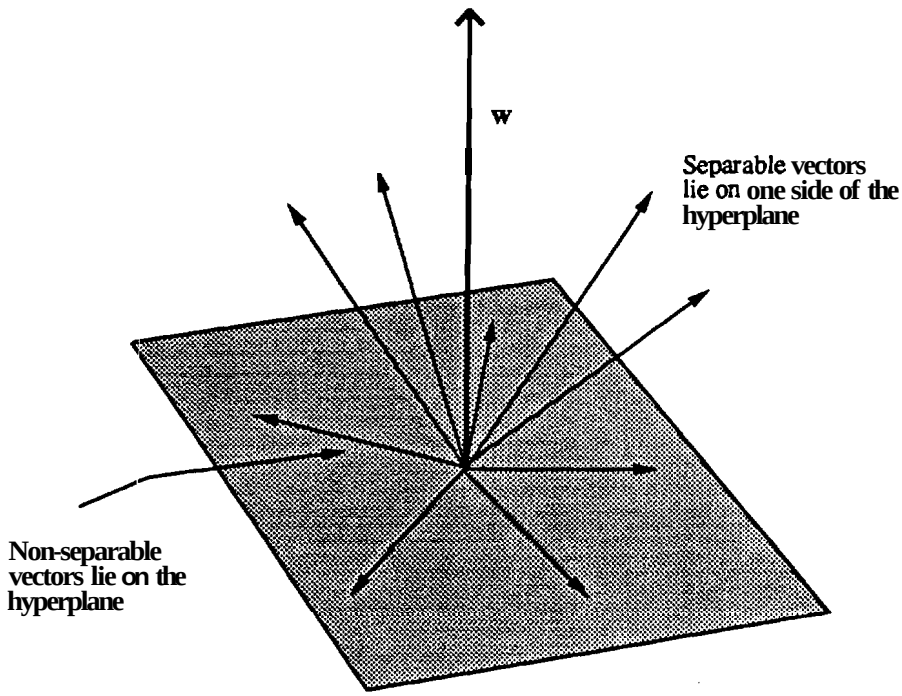

Figure 1: A hyperplane that can be determined for linearly non-separable training sets; see Theorem 2. The separable vectors can be separated by putting them on one side of the hyperplane and all the non-separable vectors can be made to lie on the hyperplane. If any of the non-separable vectors is put on the designated side of a trial hyperplane, then there must exist other vectors that must lie on the wrong side of it thereby leading to errors.

We have assumed, without loss of generality, that the hyperplane passes through the origin; see Remarks 1, 2 and 3 for justification.

Remark 4 Given a learning problem where all the inputs cannot be simultaneously learned, a learning system is said to be ***uncommitted*** if it does not commit errors. If an input cannot be learned without forcing errors on other inputs then the uncommitted learning system should remain ***ambiguous*** with respect to it, and thereby not commit errors on other inputs.

Thus the above results imply that if a linearly non-separable set is to be learned by an uncommitted learning system, then the best the system can do is to learn the separable vectors and remain ambiguous with regard to all the non-separable ones. This is because, if the system decides to learn any of the non-separable vectors then it must commit errors on some other vectors, thereby ***committing*** itself to make a decision about which of the non-separable vectors to learn and which ones not to learn.

Given the above analysis, one can define the following two learning objectives for a linearly non-separable set of input vectors:

Problem 3 Given a set of m vectors in $\mathcal{R}^d$ $\{X_1, X_2 \cdots X_m\}$, determine the set of separable vectors, and the set of non-separable vectors.

Solving this problem will give information about the input vectors that are responsible for linear non-separability and the input vectors that are not. Moreover, it follows from Remark 4 that Problem 3 is the best that an uncommitted learning system can solve.

Problem 4 Given a set of m vectors in $\mathcal{R}^d$ $\{X_1, X_2 \cdots X_m\}$, determine a linearly separable subset of ***maximum*** cardinality.

This problem is equivalent to determining the minimum number of input vectors that need to be deleted before rest of the vectors form a linearly separable set. Solving this problem would allow the learner to choose the maximum number of vectors from the linearly non-separable set such that the chosen vectors are linearly separable.

In Section 5 we establish the computational complexities of the above two problems and prove the following two theorems.

Theorem 3 Problem 3 can be solved by solving at most m linear programming problems. Hence, there is a polynomial time algorithm for solving Problem 3.

Theorem 4 Problem 4 is NP-complete.

The NP-Completeness reduction for the above theorem can be derived from the ***Feedback Arc Set Problem*** [12].

Proving a problem NP-complete shows that solving it is as hard as solving many infamous hard problems such as the Traveling Salesman Problem. Although no proof is known that shows that no efficient polynomial time algorithm exists for NP-complete problems, the general conjecture is that it is highly unlikely that such algorithms would exist [4]. The best known algorithms for exactly solving thousands of NP-complete problems have exponential time complexity.

Thus Problem 4 is inherently harder than Problem 3, and consequently one should expect that the corresponding learning problem will be much harder too. However, fast ***heuristic algorithms*** can be developed for solving this problem if an efficient algorithm is developed for solving Problem 3. Based on our analysis of linearly non-separable training sets, we present one such algorithm in Section 5.

The main question that we ask is whether a single Linear Threshold Element (LTE) can learn to solve Problem 3. Problem 4 on the other hand is NP-complete and we cannot expect a single LTE to be able to learn solutions to such hard problems. In fact it is well known that even interconnected networks of LTEs (such as Hopfield networks) cannot exactly solve NP-complete problems. Based on the algorithm presented in Section 5, we shall however, show how the perceptron learning algorithm can be applied to obtain approximate solution to Problem 4.

In Section 6 (see Theorem 6) we show that the perceptron algorithm (which is much simpler than any algorithm to solve a linear programming problem) can indeed be used to learn the set of separable vectors and identify non-separable vectors in a ***finite number of steps.*** This shows that the perceptron learning algorithm is as effective in learning linearly non-separable patterns as it is in learning; separable ones. In Section 6.1 we strengthen our result on the learning capabilities of an LTE by developing a dual problem based on the null-space of the input training set. In particular we show that the power of the perceptron learning algorithm can be enhanced if one simultaneously runs an independent learning algorithm on the dual problem.

In Section 6.2 we shall also show how to use the perceptron learning algorithm to determine large linearly separable subsets of the given non-separable set. Finally, Section 7 contains some concluding remarks.

4 Analysis of Linearly Non-Separable Sets of Input Vectors

In this section we study the case where the set of input vectors, $S = \{X_1, \cdots, X_m\}$, $X_i \in \mathcal{R}^d$, is not linearly separable. We shall first determine necessary and sufficient conditions for a set to be linearly non-separable, and then identify structures within such sets. These results are independent of any learning algorithms and relate to inherent properties of linearly non-separable sets.

Let us first develop a linear programming formulation for the learning problems to be discussed in this paper. Recall that in Problem 2 the goal is to determine a weight vector w, such that $\mathbf{w}^T X_i > 0$, for all $1 \leq i \leq m$. Hence if the weight vector w is properly scaled then the problem of learning is to determine a vector w (if such a vector exists) such that the following matrix inequality is satisfied (we shall refer to it as a Linear Programming (LP) formulation):

$$\mathbf{w}^T [X_1 \; X_2 \; \cdots X_m] \geq [1 \; 1 \; \cdots \; 1] \tag{1}$$

This is exactly the problem of determining a feasible solution to a set of linear inequalities and can be solved by using linear programming algorithms in polynomial time (in m and d) [12]. Hence, from a computational perspective the problem of learning in perceptrons (Problem 2) can be solved efficiently by using any of the polynomial time algorithms for linear programming [12].

We should note here that the perceptron learning algorithm is much simpler than any linear programming algorithm and has the essence of learning. That is, it uses simple operations, is iterative and makes only 'local' decisions. There are, however, significant advantages to using a linear programming formulation. As we shall see repeatedly in this paper, the analysis of the linearly non-separable inputs and also the study of the behavior of the perceptron learning algorithm is greatly facilitated by using it.

Let us first present a sufficient condition for linear non-separability.

Lemma 1 If there exists a positive linear combination (*plc*) of the given set of input vectors $\{X_1, \cdots, X_m\}$ (where, $X_i \in \mathcal{R}^d$) that equals **0** then the given set of vectors is linearly non-separable.

Proof: If a positive linear combination *plc* of the vectors equals **0**, then it implies that there exist numbers $q_i \geq 0$, such that they are not all equal to 0 (*i.e.*, there exists $q_j > 0$ for some j) and that

$$\sum_{i=1}^{m} q_i X_i = 0$$

We will prove the lemma by contradiction. Suppose that the given set of vectors is Linearly separable. Then there exists a weight vector $\boldsymbol{w}$, such that $\mathbf{w}^T X_i > 0$ for all $1 \leq i \leq m$. Since $q_i \geq 0$ and there exists at least one $q_j > 0$ (for some $1 \leq j \leq m$), we have

$$\sum_{i=1}^{m} q_i(\mathbf{w}^T X_i) = \mathbf{w}^T \sum_{i=1}^{m} q_i X_i > 0$$

However, this is a contradiction since $\sum_{i=1}^{m} q_i X_i = 0$. Hence, there exists no weight vector that separates all the input vectors. □

Example 1 Let us show using the above lemma that the 2-input Exclusive-OR (XOR) function cannot be computed by an LTE. One can easily verify that in order to implement the XOR function the input vectors and the desired output for an LTE should be as follows:

$$X_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, y_1 = -1; \quad X_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, y_2 = -1$$

$$X_3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, y_3 = 1; \quad X_4 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, y_4 = 1 .$$

Next let us preprocess the vectors so that they are in the form given in Remarks 1 and 2. That is, if we eliminate the threshold (by augmenting the input vectors with -1) and negate the input vectors for which the desired output is -1, then the problem of learning XOR function is equivalent to determining a separating hyperplane for the following vectors:

$$X_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}; \quad X_2 = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix}; \quad X_3 = \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}; \quad X_4 = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} .$$

However $\sum_{i=1}^{4} X_i = 0$, and Lemma 1 implies that there exists no separating hyperplane. Hence, XOR cannot be implemented by any LTE.

We next show that the sufficient condition for linear non-separability shown in Lemma 1 is also necessary.

Theorem 1 A set of vectors is linearly non-separable if and only if there exists a positive linear combination of the vectors that equals 0, i.e., $\exists\, q \neq 0$ such that

$$\sum_{i=1}^{m} q_i X_i = 0 \quad \text{and} \quad q_i \geq 0.$$

Proof: Let D be an (d x m) matrix whose columns are the input vectors $X_1, X_2, \cdots, X_m$, *i.e.*, D = $[X_1\ X_2\ \cdots X_m]$. From eqn. (1) we know that the given set of vectors is Linearly separable if and only if the following LP has a feasible solution.

$$\begin{aligned} &\text{Minimize} \quad 0^T \mathbf{w} \\ &\mathbf{w}^T D \geq [1\,1\,1 \cdots 1] \end{aligned} \tag{2}$$

The feasibility of an LP can be often determined by studying the dual LP problem of the original LP formulation. The dual LP for eqn. (2) can be stated as follows (see [12]):

$$\begin{aligned} &\text{Maximize} \quad [1\,1 \ldots 1]\mathbf{q} \\ &D\mathbf{q} = 0; \quad q_i \geq 0, \ \forall\, 1 \leq i \leq m \end{aligned} \tag{3}$$

Note that: (1) The quantity $[1\,1 \cdots 1]\mathbf{q}$ is referred to as the cost or objective function of the linear program.

(2) Dq, $q_i \geq 0$ represents non-negative linear combination of the columns of D. Since the columns of D are the input vectors X_i, $D\mathbf{q}$ ($q_i \geq 0$) represents non-negative linear combination of the vectors X_i.

It follows from duality theorem of linear programming (especially Farkas' Lemma; see *e.g.*, [12]) that an original or primal LP (*e.g.*, eqn. (2)) has a feasible solution if and only if its dual LP (*e.g.*, eqn. (3)) has a bounded objective function.

Notice that in eqn. (3) the objective function is finite (in fact = 0) if and only if the only solution to the equation D q = 0, $q_i \geq 0$ is q = 0. If there is a solution q # 0, then $[1\,1 \cdots 1]\mathbf{q} > 0$. Moreover, for any a > 0, q' = aq is also a feasible solution for LP in eqn. (3). Hence the objective function $[1\,1 \cdots 1]\mathbf{q}'$ (> a), can be made unbounded by choosing a arbitrarily large.

Now, proof for Lemma 1 (showing the sufficiency part) follows immediately. If there is a positive linear combination (*plc*) of the vectors X_i that equals 0, then there is a non-zero q, satisfying the constraints D q = 0, q ≥ 0. Hence, eqn. (3) has an unbounded objective function. Now, applying

the duality. theorem we obtain that eqn. (2) is infeasible, which implies that the set of vectors is linearly non-separable.

Next, consider the case when the set of vectors is linearly non-separable: it implies that eqn. (2) has no feasible solution. Hence, by Farkas' Lemma we know that eqn. (3) has an. unbounded objective function. However, from the preceding discussions we know that eqn. (3) has an unbounded objective function if and only if there is a solution $q \neq 0$ that satisfies $Dq = 0$, $q \geq 0$. Thus there exists a *plc* of the vectors X_i that equals zero. □

4.1 Structures Within Linearly Non-Separable Training Sets

Here we shall study possible structures within a set of linearly non-separable vectors. Closer observation would reveal that if a set of input vectors is linearly non-separable then not all the vectors in the set are responsible for non-separability.

Example 2 Consider the following set of three vectors.

$$X_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}; \quad X_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}; \quad X_3 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

The above set is linearly non-separable, because $X_2 + X_3 = 0$. However, if we want to solve the following equation

$$aX_1 + bX_2 + cX_3 = 0; \quad a, b, c \geq 0$$

then one can show that $a = 0$. That is, there is no positive linear combination (*plc*) of X_1, X_2 and X_3 that equals 0 and in which X_1 participates. In other words, only X_2 and X_3 are responsible for linear non-separability of the above set of vectors.

This observation and the results proved in Theorem 2 prompt the following classification of the input vectors.

Definition 2 Given a linearly non-separable set $S = \{X_1, \cdots, X_n\}$, $X_j \in \mathcal{R}^d$ a vector X_i is defined to be **separable** if and only if it never participates in a positive linear combination that equals 0. That is,

$$\sum_{j=1}^{m} X_j q_j = 0; \quad q_j \geq 0 \quad \Longrightarrow \quad q_i = 0 .$$

Similarly, a vector Xi in the set S is defined to be non-separable if and only if it participates in a positive linear combination that equals 0. That is, there exists a $q_i > 0$ such that $\sum_{j=1}^{m} X_j q_j = 0; \quad q_j \geq 0.$

Example 3 In Example 2, the set of separable vectors comprises only X_1, and the set of non-separable vectors consists of X_2 and X_3.

In Example 1, however, the set of separable vectors is empty and every vector is non-separable.

Let us denote the separable vectors as $X_1, \ldots, X_k$, and the non-separable vectors as $X_{k+1}, \cdots, X_m$. If k = 0 then the set of separable vectors is empty, and if k = m then the set of non-separable vectors is empty (in other words, the given set of vectors is linearly separable).

Remark 5 Following are some important observations about separable and non-separable vectors in any training set:

1. It follows easily from the definition of non-separable vectors, that there exist $q_i > 0$, $k+1 \leq i \leq m$, such that
$$\sum_{i=k+1}^{m} q_i X_i = 0$$
Thus there is a single *plc* of the non-separable vectors which equals 0, and in which *all* the non-separable vectors participate.

2. Whether a given vector X_i in S is separable or non-separable is determined by the other vectors in the set. For example, if some vectors are deleted from S, then in the reduced set, say S', vectors which are non-separable in S might become separable.

Example 4 Consider the set of vectors $\{X_1, X_2, X_3, X_4\}$ in Example 1. Here the set of separable vectors is empty, and every vector is non-separable. However, if for example, X_1 is dropped from the set then one can verify that in the reduced set, $S' = \{X_2, X_3, X_4\}$, all the vectors are separable. This will be true if any of the vectors in the set $\{X_1, X_2, X_3, X4)$ is dropped from the set.

Similarly in Example 2, if X_2 is dropped from the set then in the reduced set, $S' = \{X_1, X_3\}$, both the vectors are separable.

3. The set of separable vectors can indeed be arbitrarily large in a linearly non-separable training set.

Example 5 Consider the following set of vectors:

$$X_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}; X_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix}; X_3 = \begin{bmatrix} 1 \\ 1 \\ -1 \\ 1 \end{bmatrix}; X_4 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \end{bmatrix}; X_5 = \begin{bmatrix} 1 \\ -1 \\ -1 \\ -1 \end{bmatrix}; X_6 = \begin{bmatrix} -1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

One can show that the set of separable vectors is $\{X_1, X_2, X_3, X_4\}$ and the set of non-separable vectors is $\{X_5, X_6\}$.

4. If the set of non-separable vectors is nonempty, then it must consist of at least two vectors.

5. ***Even if all the vectors in a given training set are non-separable, the analysis in our paper will give useful information.*** For example, in Section 5 we present an algorithm to determine a large linearly separable subset. The algorithm reduces the size of the non-separable set by successively deleting non-separable vectors. As the size becomes smaller, the subsets are going to have ***non-empty*** sets of separable vectors which need to be identified. Thus at every successive step, our analysis will indicate the set of vectors from which a vector has to be dropped in order to make the rest of the vectors linearly separable.

We next study some of the properties of the separable and non-separable sets of vectors. These properties further clarify the motivation behind the definitions of separable and non-separable sets of vectors; for a detailed discussion see Section 3 on page 7. ***Note that*** if the set of separable vectors is empty then part 2 of the following theorem is trivially satisfied by choosing $\mathbf{w}_0 = 0$.

Theorem! 2

1. If $\mathbf{w}^T X_i > 0$ for some k + 1 ≤ i ≤ *m* (*i.e.* X_i is a non-separable vector), then there must exist another non-separable vector, X_l, k + 1 ≤ *l* ≤ m, such that $\mathbf{w}^T X_l < 0$.

2. There always exists a weight vector $\mathbf{w}_0$ such that:
   1. $\mathbf{w}_0^T X_i \geq 1$, $\forall\, 1 \leq i \leq$ k.
   2. $\mathbf{w}_0^T X_j = 0$, V k + 1 ≤ j ≤ *m*.

Proof: In order to show the first part, assume that there exists a non-separable vector X_i ($k+1 \leq i \leq m$) such that $\mathbf{w}^T X_i = \gamma > 0$. Now, by the definition of non-separable vectors we know that there exists $q_i > 0$, such that $\sum_{j=k+1}^{m} q_j X_j = 0$, $q_j \geq 0$. Hence,

$$\mathbf{w}^T \sum_{j=k+1}^{m} q_j X_j = q_i \mathbf{w}^T X_i + \sum_{j=k+1; j \neq i}^{m} \mathbf{w}^T q_j X_j = 0$$

In other words,

$$\sum_{j=k+1; j \neq i}^{m} q_j \mathbf{w}^T X_j = -\gamma < 0$$

Since $q_j \geq 0$, it implies that there must exist a $l \neq i$ ($k+1 \leq l \leq m$) such that $\mathbf{w}^T X_l < 0$.

In order to prove the second part, we first show that the following LP always has a feasible solution.

$$\begin{gathered} \text{Minimize } \mathbf{0}^T \mathbf{w} \text{ Such that} \\ \mathbf{w}^T [X_1\ X_2 \cdots X_k\ X_{k+1} \cdots X_m] \geq [1\ 1 \cdots 1\ 0 \cdots 0] \end{gathered} \tag{4}$$

where, the vector $[1\ 1 \cdots 1\ 0 \cdots 0]$ has the first k entries (*i.e.*, the entries corresponding to separable vectors) as 1 and last $m-k$ entries (*i.e.*, the entries corresponding to non-separable vectors) as 0. As discussed in the proof of Theorem 1, one can show that the above LP admits a feasible solution by showing that its dual has a bounded objective function. Now the dual of eqn. (4) is given by

$$\begin{gathered} \text{Maximize } [1\ 1 \cdots 1\ 0 \cdots 0]\mathbf{q} \text{ Such that} \\ [X_1\ X_2 \cdots X_k\ X_{k+1} \cdots X_m] \begin{bmatrix} q_1 \\ \vdots \\ q_k \\ q_{k+1} \\ \vdots \\ q_m \end{bmatrix} = \mathbf{0}; \quad q_i \geq 0 \end{gathered} \tag{5}$$

Since, $X_1, \cdots, X_k$ are separable vectors, it follows from Definition 2 that any feasible solution vector, q for eqn. (5) has the first k entries equal to 0, *i.e.*, $q_i = 0$, $\forall\ 1 \leq i \leq k$.

Now the objective function of eqn. (5) is $\sum_{i=1}^{k} q_i$. Since $q_i = 0$, $\forall\ 1 \leq i \leq k$, it follows that the objective function of eqn. (5) equals 0 and hence is bounded. Thus, applying Farkas' Lemma we obtain that eqn. (4) always admits a feasible solution. In other words, there always exists a $\mathbf{w}_0$ such that (1) $\mathbf{w}_0^T X_i \geq 1 > 0$, $\forall\ 1 \leq i \leq k$., and (2) $\mathbf{w}_0^T X_j \geq 0$, $\forall\ k+1 \leq j \leq m$.

We next show that any feasible solution, $\mathbf{w}_0$, of eqn. (4) must satisfy $\mathbf{w}_0^T X_j = 0$, $\forall$ k + 1 $\leq$ $j \leq$ m. Let

$$\mathbf{w}^T[X_1\ X_2 \cdots X_k\ X_{k+1} \cdots X_m] = [\delta_1 \cdots \delta_k\ \delta_{k+1} \cdots \delta_m]$$

We know that $\delta_i \geq 1$, $\forall$ 1 $\leq$ i $\leq$ k and $\delta_i \geq 0$, $\forall$ k + 1 $\leq$ i $\leq$ m. We have to show that $\delta_i = 0$, $\forall\ k + 1 \leq i \leq$ m. Since X_i, (k + 1) $\leq$ i $\leq$ m are non-separable vectors, there exists a non-negative vector q such that $[X_1\ X_2 \cdots X_k\ X_{k+1} \cdots X_m]\mathbf{q} = 0$, $q_i > 0$ for k + 1 $\leq i \leq$ m and $q_i = 0$ for $1 \leq i \leq$ k. Hence,

$$\mathbf{w}^T[X_1\ X_2 \cdots X_k\ X_{k+1} \cdots X_m]\mathbf{q} = \sum_{i=k+1}^{m} \delta_i q_i = 0$$

Since $q_i > 0$ for k + 1 $\leq$ i $\leq$ m, it must be the case that $\delta_i = 0$, $\forall$ k + 1 $\leq$ i $\leq m$. Hence, $\mathbf{w}_0$ must satisfy $\mathbf{w}_0^T X_j = 0$, $\forall$ k + 1 $\leq j \leq m$. □

5 Learning Problems for Linearly Non-Separable Training Sets and Their Computational Complexity

Given the above analysis, one can have the following two learning objectives for a linearly non-separable set of input vectors:

Problem 3 Given a set of m vectors in $\mathcal{R}^d$ $\{X_1, X_2 \cdots X_m\}$, determine the set of separable vectors, and the set of non-separable vectors.

Solving this problem will give information about the input vectors that are responsible for linear non-separability and the input vectors that are not. We are able to show that this problem can be solved by executing at most m linear programming problems and hence can be solved in polynomial-time.

Problem 4 Given a set of m vectors in $\mathcal{R}^d$ $\{X_1, X_2 \cdots X_m\}$, determine a linearly separable subset of *maximum* cardinality.

Example 6 Consider the set of vectors $\{X_1, X_2, X_3, X_4\}$ in Example 1. As discussed in Example 4, if any of the vectors is dropped from this set, then the set becomes a linearly separable set. Hence, an example of a linearly separable subset of maximum cardinality will be the set $S' = \{X_2, X_3, X_4\}$.

Solving Problem 4 would allow the learner to choose the 'best' set of input vectors as far as linear separation is considered. It turns out that Problem 4 is a computationally. hard problem and we show it to be NP-complete. However, a fast heuristic algorithm can be developed for solving this problem if an efficient algorithm is developed for solving Problem **3.**

Theorem 3 Problem **3** can be solved by solving at most m linear programming problems. Hence, there is a polynomial time algorithm for solving Problem **3.**

Proof: To determine whether a given vector is separable or non-separable, we have to determine whether it participates in a *plc* of the given set vectors that equals **0.** Given a vector Xi, there are several ways of formulating the above query in terms of a LP problem and following is one particular formulation. Define a unit vector $e_i = [0 \cdots 0\ 1\ \cdots 0]$, *i.e.* , its i^{th} entry is 1 and rest of the entries are 0. Now consider the following LP:

$$\begin{aligned} &\text{Maximize } e_i^T \mathbf{q} \text{ such that} \\ &[X_1 \cdots X_m]\,\mathbf{q} = 0;\ \ q_i \geq 0 \end{aligned} \tag{6}$$

It is easy to prove that X_i is a separable vector if and only if the objective function of the above LP is bounded. (= 0); the reasonings can be summarized as follows: (1) the objective function is q_i, and (2) it will be unbounded if and only if there is a non-negative vector $\mathbf{q}_0$ satisfying $[X_1 \cdots X_m]\mathbf{q}_0 = 0$ and $q_i > 0$ (which implies that X; is non-separable).

One can thus determine the sets of separable and non-separable vectors by solving at most m LP problems. □

We next show that the Problem 4 is NP-complete. In order to show the NP-completeness result, we reduce the following NP-complete problem to Problem 4.

Problem 5 ***Feedback Arc Set Problem*** [12]
Given a directed graph G = (V, E) (where V is the set of nodes and E is the set of directed edges), determine the minimum number of edges that needs to be removed from E so that the resulting subgraph is *acyclic.*

Theorem 4 Problem 4 is NP-complete.

Proof: It follows easily that Problem 4 is in the class NP. This is because if a trial solution is given for the corresponding decision problem then using a polynomial time algorithm for Linear

Programming [12] one can verify whether the given subset is linearly separable or not. The next step in showing NP-Completeness is to show that a known NP-complete problem (the Feedback Arc Set Problem for our case) can be reduced in polynomial time to Problem 4.

Before we present the reduction, let us review some basic material on scheduling of directed graphs. Given a directed graph G = (V, E), a scheduling is an indexing of the vertices of V such that if there is a directed edge $v_i \rightarrow v_j$ in E then the schedule assigned to v_i, say S_{v_i}, should be greater than the schedule assigned to v_j, *i.e.*, $S_{v_i} - S_{v_j} \geq 1$. Let $S^T = [S_{v_1}\ S_{v_2} \cdots S_{v_{|V|}}]$ be the scheduling vector (where S_{v_i} represents the schedule for node v_i). If we write down the constraints that the schedules must satisfy for each edge in E, then one gets the following matrix inequality:

$$S^T \mathbf{C} \geq [1\ 1 \cdots 1] \tag{7}$$

where $\boldsymbol{C}$ is referred to as the connection *matrix* and has the following properties: (1) it is of dimension $|V|$ x $|E|$, *i.e.*, it has one row for every node and one column for every edge in G. **(2)** the entries of the i^{th} column, defined by the i^{th} edge $e_i = (v_k, v_l) \in E$, are defined as follows: it has a -1 entry at the k^{th} row (that is the row corresponding to node v_k, where e_i originates), it has a $+1$ entry at the l^{th} row (that is the row corresponding to node v_l, where e_i terminates), and the rest of the entries are set to 0.

It is easy to show that there is a valid schedule for a given *graph* G, if and only if G is acyclic. In other words, there is a solution to eqn. (7) if and only if the underlying graph is acyclic. Hence, the problem of determining the minimum number of edges to be deleted so that the resulting graph is acyclic is equivalent to the problem of determining the minimum number of edges to be deleted so that the resulting subgraph admits a valid schedule.

Given the above introduction, a polynomial time reduction from the Feedback Arc Set problem to Problem 4 follows rather directly. Given a directed graph $G(V, E)$, first determine its connection matrix $\boldsymbol{C}$; this can be done in linear time in $|V|$ and $|E|$. Assign the columns of $\boldsymbol{C}$ as the input vectors to Problem 4, *i.e.*, set the vector X_i as the i^{th} column of $\boldsymbol{C}$.

Now, dletermining whether there is a weight vector w that separates $X_1, \ldots, X_m$ is equivalent to determining whether G is acyclic. This is because if such a w exists then it must satisfy (see equation (1)):

$$\mathbf{w}^T [X_1\ X_2 \cdots X_m] \geq [1\ 1 \cdots 1]$$

Since Xi's are the columns of the connection matrix C, S = w would satisfy equation (7) and hence can be considered as a schedule for G. One the other hand a schedule exists for a graph G if and only if it is acyclic.

Therefore, if the graph G is cyclic then the set of vectors $X_1, \cdots, X_m$ is linearly non-separable. However, if a subset of $\{X_1, \cdots, X_m\}$ is linearly separable then the corresponding sub-graph (defined by deleting those edges in G that do not appear in the subset) will be acyclic. Hence, the problem of determining a linearly separable subset of $\{X_1, \cdots, X_m\}$ of maximum cardinality, corresponcls directly to the problem of deleting the minimum number of edges so that the resulting subgraph G is acyclic. □

Thus Problem 4 is inherently harder than Problem 3, and consequently one should expect that the corresponding learning problem will be much harder too. However, based on the properties of separable and non-separable vectors, one can give a heuristic algorithm for solving Problem 4.

Input: A set of vectors: $S = \{X_1, X_2, \cdots, X_m\}$.

Output: A linearly separable set of vectors: V = $\{X_{i_1}, X_{i_2}, \cdots, X_{i_k}\} \subseteq S$.

In the following algorithm let ϕ denote the empty set.

Algorithm

Let $S = \{X_1, X_2, \cdots, X_m\}$, and $V = \phi$

While $S \neq \phi$ **Do**

Begin

Decompose S into the set of separable vectors, say S_1, and the set of non-separable vector, say S_2.

Set V = $V \bigcup S_1$

If $S_2 \neq \phi$ then randomly pick a vector $X_k \in S_2$

Set S = $S_2 - \{X_k\}$

End

At each step of the above algorithm, one can use a polynomial time algorithm (or any other learning algorithm that can solve Problem 3) to determine the sets S_1 (the set of separable vectors) and S_2 (the set of non-separable vectors). Since the vectors in S_1 are not responsible for linear non-separability, one can append it directly to the desired output set V. However, since the vectors in S_2 are responsible for linear non-separability one needs to drop at least one vector to make the rest linearly separable. Our analysis thus enables us to make an 'intelligent choice' of vectors that

need to be deleted: in particular, only vectors in S_2 need to be considered for deletion, and all the vectors in S_1 can be directly appended to the output set. In the above algorithm, a randomly chosen vector (X_k) is deleted from S_2 and the reduced set $(S_2 - \{X_k\})$ is again checked for linear non-separability. The following lemma shows that there always exists a choice of the vectors being deleted such that V will be the linearly separable subset of maximum cardinality.

Lemma 2 In the above algorithm there always exists a choice of X_k's such that the resulting output set V is a linearly separable subset of $\{X_1, X_2, \cdots, X_m\}$ of maximum cardinality.

The proof is straight-forward and will be skipped. Important implication of the above lemma is that by varying the choice of vectors being deleted at every step in the above algorithm one can get very good approximate solutions to Problem 4.

Remark 6

1. The above algorithm can be further improved in several ways. For example, once a linearly separable set V is obtained one can try to increase its size by checking whether any of the deleted vectors can be added to V without making it linearly non-separable. Since the size of V is determined by the choice of vectors being deleted at each step (as implied by the above Lemma), one may be able to increase its size by adding some of the vectors which were deleted earlier.

2. The above algorithm shows how our analysis can be useful even if all the vectors in a given set $\{X_1, X_2, \cdots, X_m\}$ are linearly non-separable. In such a case, during the first step S_1 will be empty; however, as vectors are dropped, the set of separable vectors (S_1) in subsequent steps will become non-empty.

6 Behavior of The Perceptron Learning Algorithm

The main question that we ask in this section is whether the simple perceptron learning algorithm (which is much simpler than any algorithm to solve a linear programming problem) can solve Problem 3. We show that perceptron algorithm can indeed be used to learn the set of separable and identify the set of non-separable vectors in ***a finite number of steps.*** This shows that ***the perceptron learning algorithm is as efficient in learning linearly non-separable patterns as it is in***

learning separable ones. We also show (see Section **6.2)** how the perceptron learning algorithm can be used to determine large linearly separable subsets of any given non-separable training set.

Let us now state an important result about the length of the weight vector $\mathbf{w}_l$ in the perceptron learning algorithm when the input vectors are linearly non-separable.

Theorem 5 If the perceptron learning algorithm is applied to a linearly non-separable set of vectors $S = \{X_1, \cdots X_m\}$, then the length of the weight vector $\mathbf{w}_l$ remains bounded, i.e., there exists a constant N_S such that $||\mathbf{w}_l|| \leq N_S$ for all $l \geq 0$.

A proof for this theorem can be found in [10].

The perceptron learning algorithm (as stated in Section 2) does not converge if the input vectors are linearly non-separable. Hence, the above theorem states that even if the algorithm iterates indefinitely the length of the weight vector will remain bounded.

In the perceptron algorithm, the step ([ADD]) where a vector X_i is added to the current value of $\mathbf{w}_l$ to generate the next value, $\mathbf{w}_{l+1}$, will be referred to as an ***update step.*** The algorithm will be said to have ***converged*** with respect to a vector X_i after the k^{th} step only if $X_i^T \mathbf{w}_l > 0$ for ***all*** $l > k$. That is, after a finite number of updates (= k), X_i will ***never*** be used to update the weight vector $\mathbf{w}_l$. In suclh a case, we will also say that ***the algorithm has learned*** the vector X_i.

Let $\mathbf{w}_l$ be the value of the weight vector after the l^{th} update. The total number of updates, 1, can be written as $I = l_1 + l_2$, where l_1 is the number of updates using only the separable vectors and l_2 is the number of updates using only the non-separable vectors. We are going to show that l_1 is finite. This would show that after a finite number of steps, the separable vectors are never used for updating the weight vector $\mathbf{w}_l$. Equivalently, we can say that the perceptron learning algorithm learns all the separable vectors after a finite number of updates.

Theorem 6 Let l_1 be the total number of updates of the weight vector $\mathbf{w}_l$ in the perceptron learning algorithm using only the separable vectors. Then l_1 is finite, i.e., after a finite number of updates, the perceptron learning algorithm learns all the separable vectors.

Proof: Without loss of generality, assume that the first k vectors, $X_1, \cdots, X_k$ are the separable vectors, and the rest, $X_{k+1}, \cdots, X_m$ are the non-separable ones. We can always write the weight vector $\mathbf{w}_l$ (after 1 updates) as

$$\mathbf{w}_l = \sum_{i=1}^{k} \alpha_i X_i + \sum_{i=k+1}^{m} \alpha_i X_i$$

where integers $a_i \geq 0$ represent the number of times the vector X_i has been used in updating the weight vector. Let $\sum_{i=1}^{k} \alpha_i = l_1$ and $\sum_{i=k+1}^{m} \alpha_i = l_2$. Thus, l_1 is the total number of updates using only separable vectors and l_2 is the total number of updates using only non-separable vectors.

We know by Theorem 2 that there exists a weight vector $\mathbf{w}_0$ such that $\mathbf{w}_0^T X_i \geq 1$, $\forall\, 1 \leq i \leq k$, and $\mathbf{w}_0^T X_j = 0$, $\forall\, k+1 \leq j \leq m$. Hence,

$$\mathbf{w}_0^T \mathbf{w}_l = \sum_{i=1}^{k} \alpha_i \mathbf{w}_0^T X_i + \sum_{j=k+1}^{m} \alpha_j \mathbf{w}_0^T X_j = \sum_{i=1}^{k} \alpha_i \mathbf{w}_0^T X_i \geq l_1$$

Now if $||\mathbf{w}_0|| \leq L$ then applying Cauchy-Schwarz inequality we get that

$$l_1 \leq ||\mathbf{w}_0||||\mathbf{w}_l|| \leq L||\mathbf{w}_l||$$

However, by Theorem 5 we have $||\mathbf{w}_l|| \leq N_S$. Hence, we obtain

$$l_1 \leq LN_S .$$

Thus, the the perceptron learning algorithm learns the separable vectors in a finite number of updates. □

The perceptron learning algorithm never converges with respect to the non-separable inputs. Hence, the non-separable inputs can be distinguished by the fact that for every non-separable input X_i, the inner-product $\mathbf{w}_l^T X_i$ will always become negative during some updating after finite number of steps. Thus, the perceptron learning algorithm can be used to learn the structure of linearly non-separable training sets in the following manner:

> Apply the perceptron learning algorithm to the given set of vectors and record the vectors that are being used for updating the weight vector. As the algorithm keeps running, separate the vectors in to two sets: 1) a set of vectors that are no longer being used for updating (call it the set B), and 2) a set of vectors that are being recurringly used to update the weight vector (call it the set C).

Theorem 6 shows that the set B will contain all the separable vectors after a finite number of updates. Moreover, the property of a nonseparable vector implies that as the algorithm keeps running it must end up in the set C. So our analysis shows that there exists a finite number of updates such that if the perceptron learning algorithm is stopped then sets B and C will respectively correspond to the sets of separable and non-separable vectors.

Example 7 Consider the set of input vectors studied in Example 2, where the input vectors are given as:

$$X_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}; \; X_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}; \; X_3 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Without loss of generality, let us assume that the perceptron learning algorithm chooses the vectors in the following order: X_1, X_2, X_3. One can then verify that the weight vector converges with respect to X_1 in one step. That is, $\mathbf{w}_l^T X_1 > 0$ for all $l \geq 1$. As the algorithm keeps iterating indefinitely, only vectors X_2 and X_3 are used alternately for updating the weight vector. In fact, the weight vector $\mathbf{w}_l$ would start repeating its value after 3 iterations. One can conclude then from the above results, that X_1 is a separable vector, and X_2, X_3 are non-separable vectors. Thus the perceptron algorithm correctly learns the structure of the training set in only a few number of iterations.
One can verify that a similar behavior can be observed in Example 5. In fact, after the first pass (i.e., after all the vectors have been checked once), the perceptron algorithm will learn the set of separable vectors (X_1, X_2, X_3, X_4), and from then on only X_5 and X_6 will be used alternately for updating the weight vector indicating that these are the non-separable vectors.

Example 8 If the perceptron learning algorithm is applied to the four vectors in Example 1, then one can verify that the algorithm will not converge with respect to any of the vectors. In fact, if the vectors are considered in order (i.e., X_1, X_2, X_3, X_4) during the execution of the algorithm, then the weight vector $\mathbf{w}_l$ will become 0 every four iterations, thereby indicating that every vector will be used for updates infinitely often. Hence, within a few iterations the perceptron learning algorithm would reach the correct conclusion that all the input vectors are linearly non-separable.

The following table presents some experimental results that indicate the performance of the perceptron learning algorithm for randomly generated training sets.

# of vectors (m)	dimension (d)	# of separable vectors (k)	maximum # of iterations for the separable vectors
10	4	3	49
20	10	20	425
30	15	10	538
40	20	15	717
50	23	16	995

Each coordinate of every vector in our experiment was generated randomly and independently according to the uniform distribution over $[-1, 1]$. After generating a complete training set we used linear programming techniques (as explained in the proof of Theorem 3) to determine the sets of separable vectors and non-separable vectors. Then we applied the perceptron learning algorithm to the training set and observed the number of iterations before the algorithm stopped using any of the separable vectors for updating the weight vectors. In case the perceptron algorithm did not converge, we terminated the algorithm after 5000 iterations. As shown in the above table, the perceptron algorithm converged with respect to the separable vectors in a few :hundred iterations. As we let the perceptron algorithm run up to 5000 iterations, we also observed that every non-separable vector would be used recurringly for updating the weight vector. This shows that the perceptron learning algorithm learned the correct structure of the training sets in our experiments within a few thousand iterations.

6.1 A Dual Learning Problem

In this section we present a dual problem that can substantially add to the power of the perceptron learning algorithm in classifying linearly non-separable training sets. The formulation and properties of the dual problem is independent of any particular learning algorithm and hence the related concepts can be also used for enhancing the performance of learning algorithms other than the perceptron learning algorithm.

Let $D := [X_1\ X_2 \cdots X_k\ X_{k+1} \cdots X_m]$ be the matrix formed by the set of input vectors. Without loss of generality assume that D has full row-rank (= d). Then we can denote the right null-space

of D as follows:

$$\mathbf{N}_D = \begin{bmatrix} Y_1^T \\ \vdots \\ Y_k^T \\ Y_{k+1}^T \\ \vdots \\ Y_m^T \end{bmatrix} \tag{8}$$

where, $Y_i \in \mathcal{R}^{m-d}$. Note that the null-space of a matrix D can be computed relatively easily by using well known techniques such as the Gaussian elimination algorithm and other iterative algorithms, which have much less computational complexity than the known algorithms for linear programming.

Recall that since $\{X_1, \cdots, X_k\}$ is the set of separable vectors and $\{X_{k+1}, \cdots X_m\}$ is the set of non-separable vectors, there exists a vector $\mathbf{q_0} \geq 0$ such that $\mathbf{D}\ \mathbf{q} = 0$, $q_{0i} = 0$, $1 \leq i \leq k$ and $q_{0i} \geq 1$, $k + 1 \leq i \leq m$. Also, since $\mathbf{q_0}$ is in the null-space of D, there exists a vector $\mathbf{a} \in \mathcal{R}^{m-d}$ such that

$$\alpha^T \mathbf{N}_D^T = \alpha^T [Y_1 \cdots Y_k\ Y_{k+1} \cdots Y_m] = \mathbf{q}_0^T \tag{9}$$

Given this introduction we can prove the following property about the null-space vectors Y_i.

Theorem 7 For the set $\mathbf{A} = \{Y_1, Y_2, \cdots, Y_k, Y_{k+1}, \cdots Y_m\}$, $Y_i \in \mathcal{R}^{m-d}$, as defined in eqn. (8), $\{Y_1, Y_2, \cdots, Y_k\}$ is the set of non-separable vectors and $\{Y_{k+1}, Y_{k+2} \cdots Y_m\}$ is the set of separable vectors.

Proof: First we shall show that the set $\mathbf{A'} = \{Y_{k+1}, Y_{k+2} \cdots Y_m\}$ is a sub-set of the set of separable vectors of A. Then we shall argue that the set of separable vectors cannot be any larger than $\mathbf{A'}$. In order to show that the vectors $Y_{k+1}, Y_{k+2} \cdots Y_m$ are separable, it suffices to show that the following LP has a bounded objective function (in fact, = 0) (see the definition of separable vectors, and the

LP formalism introduced in Section 4):

$$\text{Maximize } [0 \cdots 0\ 1\ 1 \cdots 1]\mathbf{p} \text{ Such that}$$

$$[Y_1\ Y_2 \cdots Y_k\ Y_{k+1} \cdots Y_m] \begin{bmatrix} p_1 \\ \vdots \\ p_k \\ p_{k+1} \\ \vdots \\ p_m \end{bmatrix} = \mathbf{0}; \quad p_i \geq 0 \tag{10}$$

Now from duality theory we know that the above LP will have a bounded objective function, if and only if its dual (as given below) has a feasible solution.

$$\begin{gathered} \text{Minimize } \mathbf{0}^T\mathbf{b} \text{ Such that} \\ \mathbf{b}^T[Y_1\ Y_2 \cdots Y_k\ Y_{k+1} \cdots Y_m] \geq [0 \cdots 0\ 1\ 1 \cdots 1] \end{gathered} \tag{11}$$

However, a feasible solution to the above LP is already given by the equation (9): set $\mathbf{b} = \mathbf{a}$. Then,

$$\mathbf{b}^T[Y_1\ Y_2 \cdots Y_k\ Y_{k+1} \cdots Y_m] = \mathbf{q}_0 \geq [0 \cdots 0\ 1\ 1 \cdots 1]$$

The proof of the theorem can be completed by showing that no other vectors in A are separable. One of way proving this would be by contradiction. For example, if we assume that Y_i is also separable for some $1 \leq i \leq k$, then following arguments very similar to the ones used above, one can show that X_i is a non-separable vector, which contradicts our assumption that X_i is a separable vector. □

The above theorem shows that if a vector X_i is separable in the set $\{X_1, \cdots, X_m\}$, then the corresponding vector, Y_i, is non-separable in the set $\{Y_1, \ldots, Y_m\}$ and vice-versa.

Given the set $\{X_1, \cdots, X_m\}$, let us define the ***dual learning*** problem as the learning problem for the null-space vectors $\{Y_1, \cdots, Y_m\}$ (as defined in equation (9)). We can make the following remarks about the dual learning problem:

1. It follows from Theorem 7 that the structure of the set $\{X_1, \cdots, X_m\}$ can be directly obtained by learning the structure of the set $\{Y_1, \cdots, Y_m\}$: if $\{Y_{i_1}, \cdots Y_{i_k}\}$ is the set of ***non-separable*** vectors in $\{Y_1, \cdots, Y_m\}$, then the set of ***separable*** vectors in $\{X_1, \cdots, X_m\}$ is the corresponding set: $\{X_{i_1}, \cdots X_{i_k}\}$; moreover, the set of non-separable vectors is comprised of the rest of the

vectors.

A special case where the dual learning problem will be obviously useful is when all the vectors in $\{X_1, \cdots, X_m\}$ are non-separable. In such a case the perceptron learning: algorithm applied to $\{X_1, \cdots, X_m\}$ will not converge with respect to any of the vectors. However, the set $\{Y_1, \cdots, Y_m\}$ for the dual problem is linearly separable (as a consequence of Theorem 7), and the dual learning algorithm will converge for all the vectors. Hence, the structure of the learning problem can be learned without any errors by the outcome of the! dual problem; see Exarnple 10.

2. ***In general, enhanced performance can be achieved by simulatneously running the perceptron learning algorithm on the original vectors*** *($\{X_1, \cdots, X_m\}$)*, ***and on the null-space vectors*** *($\{Y_1, \cdots, Y_m\}$)*. One can keep running the two algorithms, until their predictions match: the set of separable vectors predicted by the algorithm operating on the original vectors should match (i.e., if X_j is predicted as a separable vector in the original learning problem, then Y_j should be predicted as a non-separable vector in the dual learning problem) the set of non-separable vectors predicted by the dual learning problem. Similarly, the set of separable vectors predicted by the dual learning problem must match the set of non-separable vectors predicted by learning algorithm applied to the original vectors.

Example 9 Consider the set of vectors, $S = \{X_1, X_2, X_3\}$, in Example 2. Here, the null space is of dimension 1 (= m − d) and is given by:

$$N_D = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

Hence, $Y_1 = [0], Y_2 = [1], Y_3 = [1]$. Since $Y_1 = [0]$, $\mathbf{w}^T Y_1 = 0$ for every choice of a weight vector. Hence, one can conclude from inspection that Y_1 is a non-separable vector. In fact if the perceptron learning algorithm is run on the Y_i's then one can easily observe that the weight vector $\mathbf{w}_l = [1]$ for all $l \geq 2$ (assuming that $\mathbf{w}_0 = 0$). Hence, within at most 2 iterations one can determine that the learning algorithm has converged with respect to Y_2, Y_3 and will never converge with respect to Yo. Theorem 6 then would imply that $\{Y_2, Y_3\}$ is the set of separable vectors and Y_1 is a non-separable vector. Hence, it follows from Theorem 7 that X_2, X_3 are the non-separable vecrors, and X_1 is the

separable vector in S, which result was proved in Example 2.

This example thus shows that applying the perceptron learning algorithm on the dual problem can give very clirect answers regarding the sets of separable and non-separable vectors.

Example 10 Consider the set of vectors $S = \{X_1, X_2, X_3, X4)$ in Example 1. The dimension of the null space is again 1, and the space is given by:

$$N_D = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Hence, $Y_1 = Y_2 = Y_3 = Y_4 = [1]$, and the perceptron learning algorithm would converge in a single iteration, e.g., w = [1] is a solution. Hence, $\{Y_1, Y_2, Y_3, Y_4\}$ forms a linearly separable set. Theorem 7 then would imply that X_1, X_2, X_3, X_4 are all non-separable vectors.

This example again illustrates that applying the perceptron algorithm on the null-space vectors can lead to very efficient means of identifying the structure of a linearly non-separable set.

6.2 Determining Linearly Separable Subsets

Since the perceptron learning algorithm can learn the sets of separable and non-separable vectors, it can be applied to obtain approximate solution to Problem 4. In other words, one can use the perceptron algorithm to learn large linearly separable subsets of any given non-separable training set by following the algorithm outlined in Section 5.

Example 11 Consider the set of vectors in Example 1. If the algorithm in Section 5 is applied to the set then in the first step, $S_1 = \phi$ and $S_2\{X_1, X_2, X_3, X_4\}$. Let X_1 be chosen to be deleted from S_2. In the next step of the algorithm one can verify that one will have $S_1 = \{X_2, X_3, X_4\}$ and $S_2 = \phi$. Thus the output of the algorithm will be V = $\{X_2, X_3, X_4\}$, which is indeed a linearly separable subset of maximum cardinality.

One can also apply the algorithm to the linearly non-separable set in Example 5. In the first pass one will have $S_1 = \{X_1, X_2, X_3, X_4\}$ = V (set of separable vectors), and $S_2 = \{X_5, X_6\}$ (set of non-separable vectors). Let X_5 be the vector dropped from S_2. Then in the second pass one will have $S_1 = \{X_6\}$ and $S_2 = \phi$. Thus the output linearly separable set becomes V = $\{X_1, X_2, X_3, X_4, X_6\}$, which is again a linearly separable subset of maximum cardinality.

It will be an interesting research topic to integrate the approach used in this; paper with other algorithmri (e.g., the Ho-Kashyap algorithm) [5, 6] for enhancing the performance in determining large linearly separable subsets from a given linearly nonseparable training set.

7 Concluding Remarks

In this paper we have presented novel learning issues for linearly non-separable training sets. In the first part, we have developed results on the possible structures within linearly non-separable training sets, and defined learning problems for such sets. Based on our analysis, we have evaluated the perforimance of the well known perceptron learning algorithm. Our results show that one can use the perceptron learning algorithm to learn some of the structures inherent in a linearly non-separable training set. We have also presented efficient algorithms to learn large linearly searable subset of a given linearly non-separable training set.

Since the analysis of linearly non-separable training sets (and the associated learning problems) presented in this paper is independent of any particular learning algorithm, an. interesting future research piroblem will be to evaluate the performance of learning algorithms other than the perceptron learning algorithm in classifying linearly non-separable training sets. That; is, one would like to investigate how other learning algorithms would perform in solving the learning problems posed in this paper for linearly non-separable training sets.

References

[1] Y. S. Abu-Mostafa. The Complexity of Information Extraction. IEEE *Trans.* on *Information Theory*, IT-32:513-525, 1988.

[2] A. L. Blum and R. L. Rivest. Training a 3-Node Neural Network is NP-complete. Neural Networks, 5(1):117–127, 1992.

[3] T. M. Cover. Geometrical and Statistical Properties of Systems of Lineal: Inequalities with Applications in Pattern Recognition. IEEE Trans. on Electronic *Computers*, EC-14:326–34, 1965.

[4] M. Garey and D. Johnson. ***Computers and Intractability: A Guide to the Theory of NP-Completeness.*** W. H. Freeman: San Francisco, CA, 1979.

[5] M. H. Hassoun and J. Song. Adaptive Ho-Kashyap Rules for perceptron Training. ***IEEE Tmns. on Neural Networks,*** Vol. 3, No. 1, pp. 51–61, January 1992.

[6] M. H. Hassoun. Optical threshold gates and logical signal processing. ***Ph.D. Thesis,*** Wayne State University, MI, 1986.

[7] J. S. Judd. ***Neuml Network Design and The Complezity of Learning.*** The MIT Press, 1990.

[8] W. S. McCulloch and W. Pitts. A Logical Calculus of Ideas Immanent in Nervous Activity. ***Bulletin of Mathematical Biophysics,*** 5:115–133, 1943.

[9] N. Megiddo. On the Complexity of Polyhedral Separability. Technical Report RJ 5252, IBM Almaden Research Center, San Jose, CA, 1986.

[10] M. Minsky and S. Papert. ***Perceptrons.*** The MIT Press, Expanded edition, 1988.

[11] S. Muroga. ***Threshold Logic and its Applications.*** John Wiley & Sons Inc., 1971.

[12] C. H. Papadimitrou and K. Steiglitz. ***Combinatorial Optimization: Algorithms and Complezity.*** Prentice Hall, Inc., New Jersey, 1982.

[13] F. Rosenblatt. ***Principles of Neurodynamics.*** New York: Spartan, 1962.

[14] A. Vergis, K. Steiglitz, and B. Dickinson. The Complexity of Analog computation. ***Mathematics and Computers in Simulation,*** 28:91–113, 1986.

[15] B. Widrow and M. Lehr. 30 Years of Adaptive Neural Networks: Perceptron, Madaline, and BackPropagation. ***Proceedings of the IEEE,*** 78, No. 9:1415–1442, Sept. 1990.

[16] B. Widrow and Jr. M.E. Hoff. Adaptive Switching Circuits. 1960 ***IRE Western Electric Show and Convention Record, Part 4,*** pages 96–104, August 1960.