A Fast Linear Separability Test by Projection of Positive Points on Subspaces

Yogananda A P YOGANANDA_AP@SATYAM.COM
Applied Research Group, Satyam Computer Services Limited, SID Block, IISc, Bangalore 560012, India

M Narasimha Murthy MNM@CSA.IISC.ERNET.IN
Department of Computer Science and Automation, Indian Institute of Science, Bangalore 560012, India

Lakshmi Gopal LAKSHMI_GOPAL@SATYAM.COM
Applied Research Group, Satyam Computer Services Limited, SID Block, IISc, Bangalore 560012, India

Abstract

A geometric and non parametric procedure for testing if two finite set of points are linearly separable is proposed. The Linear Separability Test is equivalent to a test that determines if a strictly positive point $h > \mathbf{0}$ exists in the range of a matrix A (related to the points in the two finite sets). The algorithm proposed in the paper iteratively checks if a strictly positive point exists in a subspace by projecting a strictly positive vector with equal co-ordinates (p), on the subspace. At the end of each iteration, the subspace is reduced to a lower dimensional subspace. The test is completed within $r \leq \min(n, d+1)$ steps, for both linearly separable and non separable problems (r is the rank of A, n is the number of points and d is the dimension of the space containing the points). The worst case time complexity of the algorithm is $O(nr^3)$ and space complexity of the algorithm is $O(nd)$. A small review of some of the prominent algorithms and their time complexities is included. The worst case computational complexity of our algorithm is lower than the worst case computational complexity of Simplex, Perceptron, Support Vector Machine and Convex Hull Algorithms, if $d < n^{\frac{2}{3}}$.

Appearing in *Proceedings of the 24th International Conference on Machine Learning*, Corvallis, OR, 2007.

1. Introduction

The Linear separability test determines if a hyperplane separates two sets P and $Q \subset R^d$. A hyperplane $H = \{v | w^t v + b = 0, v \in R^d, w \in R^d, b \in R, w, b \text{ are fixed}\}$ separates the two sets if

$$\begin{aligned} w^t y + b &> 0 \ \forall y \in P \\ w^t y + b &< 0 \ \forall y \in Q \\ \text{or } w^t(-y) + b(-1) &> 0 \ \forall y \in Q \end{aligned}$$

Let $P = \{v'_i | v'_i \in R^{1\times d}, i = 1, \ldots, N_1\}$
$Q = \{u'_i | u'_i \in R^{1\times d}, i = 1, \ldots, N_2\}$
$n = N_1 + N_2$
$x = [w_1, w_2, \ldots, w_d, \ b]^t$

$$\text{and } A = \begin{bmatrix} v'_{1,1} & v'_{1,2} & \ldots & v'_{1,d} & 1 \\ v'_{2,1} & v'_{2,2} & \ldots & v'_{2,d} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ v'_{N_1,1} & v'_{N_1,2} & \ldots & v'_{N_1,d} & 1 \\ -u'_{1,1} & -u'_{1,2} & \ldots & -u'_{1,d} & -1 \\ -u'_{2,1} & -u'_{2,2} & \ldots & -u'_{2,d} & -1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -u'_{N_2,1} & -u'_{N_2,2} & \ldots & -u'_{N_2,d} & -1 \end{bmatrix}.$$

The linear separability test is equivalent to verifying if $B = \{x | Ax > \mathbf{0}\}$ is non-empty. Many algorithms have been proposed for the test and a comprehensive discussion of the algorithms can be found in (Elizondo, 2006) and Ch. 5 in (Duda et al., 2000).

1.1. Algorithms

A general discussion of Linear Programming, Quadratic Programming, Convex Hull and the Perceptron Algorithms to solve the linear separability problem is included in this subsection. The time complexities of the algorithms are discussed subsequently.

1.1.1. Linear Programming

Let $p = [1, 1, \ldots, 1]^t \in R^n, \tau \in R$. The linear programming formulation for solving the problem is

$$\begin{aligned} \min_{x,\tau} \quad & \tau \\ \text{subject to} \quad & Ax + \tau p \geq \mathbf{1} \\ & \tau \geq 0 \\ B \neq \emptyset \quad \Leftrightarrow \quad & \text{optimal } \tau = \tau^* = 0 \end{aligned}$$

The linear programming problem is solved either by Simplex or Interior Point methods. A feasible solution to the above problem is easy to determine. Simplex solves the optimization problem iteratively by moving on the boundary of the convex polyhedral region defined by the constraints. The Interior Point algorithms can move through the interior of the convex region.

1.1.2. Quadratic Programming

Let $\xi \in R^{n \times 1}$. Let $x = [w_1, w_2, \ldots, w_d,\ b]^t$ as defined previously. Let $C > 0$ be an arbitrarily large constant.

$$\begin{aligned} \min_{w,b} \quad & \tfrac{1}{2}\|w\|^2 + C\|\xi\|^2 \\ \text{sub:} \quad & Ax \geq \mathbf{1} - \xi \\ & \xi \geq \mathbf{0} \end{aligned}$$

$B \neq \emptyset \quad \Leftrightarrow \|\xi\|^2 = \|\xi^*\|^2 = 0$ in the optimal solution

The feasible region determined by the constraints is convex and polyhedral. An incremental algorithm finds a separating hyperplane for pairs of points in P and Q, and refines a common separating hyperplane between all pairs of points. If the problem is linearly separable, the common separator can be written as a linear combination of very few pairs of boundary points in P and Q, called support vectors (Cristianini & Taylor, 2000). SVM finds a hyperplane which has large margin (perpendicular distance of the points in P and Q closest to the hyperplane). Given that most real life data sets are linearly non separable (Wasserman, 1989, Linear Separability : Ch. 2), SVM identifies an unique hyperplane which has less misclassifications irrespective of the distributions from which sets P and Q are drawn, if feature vectors in P and Q are drawn independently and identically from fixed distributions and n is large (Cristianini & Taylor, 2000, Ch. 4). The number of iterations that the SVM requires to find a separator for the linearly separable problems is independent of the proximity of P and Q (Tsang et al., 2005).

1.1.3. Convex Hull Separation

The convex hull of a set of points is the smallest convex region that encloses the points of the set. Let $C_v(P)$ be the convex hull of P. Let $C_v(Q)$ be the convex hull of Q. B is nonempty if and only if $C_v(P) \cap C_v(Q) = \emptyset$. There are many algorithms to compute the Convex hull of a finite set of points. Quick Hull is a fast incremental algorithm for finding the convex hull (Barber et al., 1996). Given an initial convex hull (a simplex of $d+1$ points in R^d), each unprocessed point is either inside or outside the current convex hull. If the point is within the convex hull, it is discarded, otherwise a convex hull is constructed with the new point as one of the vertices.

1.1.4. Perceptron

Perceptron is an iterative procedure to find the separating hyperplane between sets P and Q. The algorithm terminates only if the two sets are separable. It has been proved that the algorithm converges in finite number of iterations when there is a separating hyperplane (Duda et al., 2000). In every iteration, x is changed to increase the sum of the non-positive coordinates of the vector Ax (misclassified examples in P and Q). Let $A_i \in R^{1 \times (d+1)}, i = 1, \ldots, n$ be the rows of matrix A. Let $\eta > 0$ be a fixed small value. Algorithm 1 is the Batch Perceptron algorithm for verifying $B \neq \emptyset$.

Algorithm 1 $B = \{x | Ax > \mathbf{0}, A \in R^{n \times (d+1)}, x \in R^{d+1}\}$, Test if $B \neq \emptyset$

Input: $n \times d + 1$ matrix A
$L \leftarrow 0$ $//L = 1$ if B is non-empty
$x \leftarrow$ random vector
$a(x) \leftarrow Ax$
if $a(x) > \mathbf{0}$ **then**
  $L \leftarrow 1$
end if
while $L == 0$ **do**
  $x \leftarrow x + \eta \sum_{a_i(x) \leq 0} A_i^t$
  $a(x) \leftarrow Ax$
  if $a(x) > \mathbf{0}$ **then**
    $L \leftarrow 1$
  end if
end while
return L

1.2. Complexity

A few algorithms converge in finite number of iterations, if and only if $B \neq \emptyset$. Perceptron algorithm and Ho-Kashyap Procedure belong to this category. Upper Bounds on the number of iterations for convergence of the algorithms on linearly separable problems can be found in (Duda et al., 2000), (Elizondo, 2006).

The bounds are dependent on the proximity of the two sets P, Q and the number of points in the two sets. The Perceptron algorithm has time complexity of $O(n^2)$ in the number of points. Examples of algorithms which converge in finite number of iterations for both linearly separable and non separable problems are Linear Programming Algorithms, Convex Hull procedure and Quadratic Programming Algorithms. Convex Hull procedure provides a direct solution to the minimum number of misclassifications of any hyperplane, subsequent to identifying the intersection region of the convex hulls of the two sets P and Q. The worst case time complexity of Linear Programming Algorithms (LP solution by Simplex) and Convex Hull procedure is exponential in dimension (Elizondo, 2006) though on most problems Simplex has low computational complexity of $O(n)$. Interior point algorithms for solving LP have worst case complexity of $O(n^3L)$, where L is a measure of the machine precision (Gonzaga, 1988). The worst case computational complexity of Quadratic Programming Algorithm is $O(n^3)$. The algorithm presented (Subspace Projection - SP) in this paper is a method to verify if a strictly positive vector exists in a subspace. It avoids computation of a hyperplane which misclassifies the least number of points.

2. Iterative Reduction of Subspaces

In this section, a verification scheme for existence of $x \in R^{d+1}$ such that $Ax > \mathbf{0}$ (A is a $n \times (d+1)$ matrix) is described. Consider an orthonormal basis of range of A. Let $U = \{u_1, u_2, \ldots, u_r \in R^n\}$ be the basis. Let the subspace spanned by U be W_r. Let a positive point p in R^n be projected on the subspace W_r. Let the projected point be $z(p) = (z_1(p), z_2(p), \ldots, z_n(p))^t \in W_r$. In brief, the following results are proved in this section. If $z(p)$ is a strictly positive point, then x exists. If $z(p) \geq \mathbf{0}$ and $I_z = \{i | z_i(p) = 0\}$ is non-empty, a strictly positive point exists in W_r if and only if a vector v of the form $v_i > 0 \ \forall i \in I_{z(p)}$ exists in W_r. If $p > \mathbf{0}$ has equal co-ordinates and $z(p) = \mathbf{0}$, then a strictly positive vector does not exist in W_r. If $z_i(p) < 0$ for some i, a subspace of $r-1$ (W_{r-1}) vectors of W_r not including z, intersects with an unique $n-1$ dimensional subspace in R^n. W_{r-1} is the intersection of W_r and $H_{n-1} = \{v \in R^n | v^t e_j = 0, j = \arg\min_{i, z_i(p)<0} \frac{p_i}{p_i - z_i(p)}\}$. A strictly positive vector exists in W_r if and only if a positive vector exists in W_{r-1}. A simple method to compute an upper bound on the number of misclassifications is proved in the last theorem.

The algorithm performs the test by projecting points with positive co-ordinates on W_r. Either the projected point $z(p)$ has strictly positive co-ordinates or some negative co-ordinates, or some zero co-ordinates. If $z(p)$ has strictly positive co-ordinates, then the sets P and Q are linearly separable. If $z(p) \geq \mathbf{0}$ the test is performed on linear combinations of rows with indices I_z of matrix A. If $z(p)$ has some negative co-ordinates, the test is performed on a lower dimensional subspace $W_{r-1} \subset W_r$ which excludes $z(p)$. At each step, the dimensionality of the subspace reduces and a test in one dimensional subspace is needed in the final step. It is easy to verify whether an one dimensional subspace is a scaling of a strictly positive point. Hence, the maximum number of steps needed for the completion of the test is equal to the dimension of the range of A.

If a vector with strictly positive co-ordinates $h > \mathbf{0}$ exists in the range of A matrix, then a separating hyperplane exists. The following theorem proves the same result when $h = z(p) > \mathbf{0}$.

Theorem 2.1 *If $z(p) > \mathbf{0}$, then $\exists\ x$ such that $Ax > \mathbf{0}$.*

Proof $z(p) \in W_r$. W_r is the range of A. Hence $\exists\ x$ such that $Ax = z(p) > \mathbf{0}$. ∎

If $z(p)$ is not a strictly positive vector, either $z(p) \geq \mathbf{0}$ or $z_i(p) < 0$ for some i. Let $z(p) \geq \mathbf{0}$. Let I_z be the co-ordinate indices corresponding to $z_i(p) = 0$. The following two theorems prove that $h > \mathbf{0}$ exists in the range of A if and only if a vector with positive values in co-ordinate indices I_z exists in the range of A. Consequently, to verify if $h > \mathbf{0}$ exists in W_r, it is sufficient to test if $h' > \mathbf{0}$ exists in a subspace of $R^{|I_z|}$. The subspace $W(I_z)$ is the set of linear combinations of $v_j = (A_{I_z(1),j}, A_{I_z(2),j}, \ldots, A_{I_z(|I_z|),j}), j = 1, \ldots, d+1$, where $A_{i,j}$ is the element in i^{th} row and j^{th} column of A.

Theorem 2.2 *If a strictly positive vector exists in W_r and $z(p) \geq \mathbf{0}$, a vector with positive values in co-ordinates $I_{z(p)}$ exists in W_r.*

Proof Let $h > \mathbf{0} \in W_r$. h has positive values in the co-ordinates $I_{z(p)}$. ∎

Theorem 2.3 *If a vector with positive values in co-ordinates $I_{z(p)}$ exists in W_r and if $z(p) \geq \mathbf{0}$, then a strictly positive vector exists in W_r.*

Proof Let $v \in W_r$ with $v_i > 0 \ \forall i \in I_{z(p)}$. Then $\alpha z(p) + v > \mathbf{0}$ for $\alpha > \max_{i \notin I_{z(p)}} \frac{-v_i}{z_i(p)}$. ∎

If $|I_z| = n$, the subspace $W(I_z)$ is the same as W_r. The following theorem proves that $h > \mathbf{0}$ does not exist in W_r if $|I_z| = n$ (i.e. $z(p) = \mathbf{0}$), provided the positive vector p has equal co-ordinates.

Theorem 2.4 *If p is the unit vector with all components equal, and $z(p) = \mathbf{0}$ then W_r does not contain a strictly positive vector.*

Proof Every vector $x \in R^n$ of the form $0 \leq x_i \leq \frac{1}{\sqrt{n}}, \forall i$ is at a distance less than or equal to 1 from p. Every point of $W_r - \{\mathbf{0}\}$ is at a distance greater than 1 from p, because $\mathbf{0}$ is the unique closest point in W_r to p. $\mathbf{0}$ is at distance 1 from p. Hence, other than $\mathbf{0}$, no vector of W_r is in the set $\{x = (x_1, x_2, \ldots, x_n) \in R^n | \ 0 \leq x_i \leq \frac{1}{\sqrt{n}}, i \in 1, \ldots, n\}$. Hence $h = \alpha x, \alpha > 0$ does not exists in W_r. Hence $h > \mathbf{0}$ does not exist in W_r. ∎

It is easy to see that, if $z(p) = \mathbf{0}$, then $\alpha z(p) = z(\alpha p) = \mathbf{0}, \alpha \in R$.

Let $p = [1, 1, \ldots, 1]^t$ be projected on W_r. Let the projection be $z(p)$. Let $z_i(p) < 0$ for some i. The following two theorems prove that a vector $h > \mathbf{0}$ exists in W_r if and only if a vector $h' > \mathbf{0}$ exists in a subspace with lesser dimension W_{r-1}. W_{r-1} does not include $z(p)$ and one of the co-ordinates of vectors v in W_{r-1} is zero ($z_i(p) < 0$ and $v_i = 0$). If more than one co-ordinate of $z(p)$ is less than zero, the co-ordinate with index $i = \arg\min_i z_i(p)$ is set to zero. Hence the linear separability test can be performed in a lower dimensional subspace.

Theorem 2.5 *If a strictly positive vector exists in W_r and $z_i(p) < 0$ for some i, there exists a positive vector in W_{r-1}.*

Proof Let $h > \mathbf{0} \in W_r$. Let a non negative point l closest to $z(p)$ on the line joining $z(p)$ and p be considered (Figure 1). $l_j = 0$ for some j such that $z_j(p) < 0$. $z(l) = z(p)$. Let $z(l) - l = \sum_i \beta_i w_i^{\perp}$ ($w_i^{\perp}$ are orthonormal to W_r and belong to the complementary subspace). Let a point q be projected on W_r. Let the projected point be $z(q)$. Let $z(q) - q = \sum_i \alpha_i w_i^{\perp}$. If $(q - l)^t(z(l) - l) \leq 0$, $\sum_i (\beta_i - \alpha_i)\beta_i \leq 0$, $\sum_i \beta_i^2 \leq \sum_i \alpha_i \beta_i \leq \sqrt{\sum_i \alpha_i^2 \sum_i \beta_i^2}$. Hence $\sum_i \beta_i^2 \leq \sum_i \alpha_i^2$. The least distance of q from W_r is greater than or equal to the least distance of l from W_r.

Let $S_l(z) = \{i | z_i(l) < 0, l_i = 0\}$, $q = l + \sum_{i \in S_l(z)} \gamma_i e_i, \gamma_i > 0$. $(q-l)^t(z(l)-l) = \sum_i \gamma_i z_i(l) < 0$. Hence q is at a distance greater than l from W_r. If every positive point in the set $T = \{v | v \in R^n, v \neq \mathbf{0}, v_j = 0 \ \forall \ j \in S_l(z)\}$ has a projection $z(v) \in W_r$ such that $z_j(v) < 0, j \in S_l(z)$, then every point $q > \mathbf{0}$ is at a distance greater than 0 from the subspace W_r. Contradiction ($h > \mathbf{0} \in W_r$ exists; geometrically, all points of T cannot be above W_r).

Hence a point $t \in T$ exists either on or below ($z_j(t) > 0$ for some $j \in S_l(z)$) the subspace W_r. The signed distance of points of T from W_r is a continuous function. Hence the distance between $z(v)$ and $v \in T$ takes all values between $d(t, z(t))$ and $d(l, z(l))$. Therefore a point $t' \in T$ exists at a distance 0 from W_r. $T = H_{n-1} = \{v \in R^n | \ v^t e_j = 0, j \in S_l(z), j = \arg\min_{i, z_i(p)<0} \frac{p_i}{p_i - z_i(p)}\}$. Let W_{r-1} be the intersection of H_{n-1} and W_r. It is easy to prove that the intersection of H_{n-1} and W_r is at most $r-1$ dimensional (vectors of $W_{r-1} = W_r \cap H$ are linearly independent of $z(p)$, as $z_j(p) < 0$ and $v_j = 0 \ \forall \ v \in W_{r-1}$). Hence, if $h > \mathbf{0}$ exists in W_r, a point $t' \geq \mathbf{0}$ exists in W_{r-1}. ∎

The existence of a positive point $t' \geq \mathbf{0}$ can be verified by projecting a positive point $p' \geq \mathbf{0} \in H_{n-1}$ on W_{r-1}. Hence the test is recursive. The recursion has a maximum depth of r, because testing whether an one dimensional subspace has a strictly positive point, is trivial. Let $R^n = H_n$. $H_n, H_{n-1}, \ldots, H_{n-r+k}, \ldots H_{n-r+1}$ and $W_r, W_{r-1}, \ldots, W_k, \ldots, W_1$ form a decreasing sequence of subspaces.

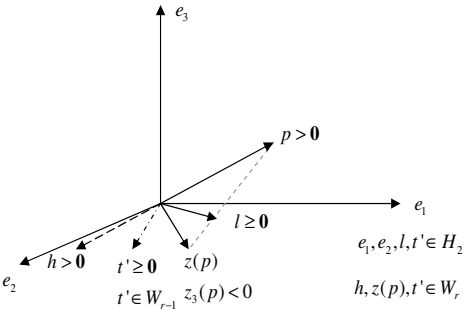

Figure 1. Projection of $p > \mathbf{0}$ on subspace W_r

Theorem 2.6 *If $z_i(p) < 0$ for some i and $v \geq \mathbf{0}$ exists in W_r such that $v_i > 0 \ \forall \ i \in G_{z(p)} = \{i | z_i(p) \geq 0\}$, then W_r contains a strictly positive vector.*

Proof $h = -z(p) + \alpha v > \mathbf{0}$ exists in W_r, for $\alpha > \max_{i \in G_{z(p)}} \frac{z_i(p)}{v_i}$. ∎

Let the standard ordered basis of R^n be $B_n = \{e_1, e_2, \ldots, e_n\}$. Let the basis vectors of H_{n-r+1} be

$B_k = \{e_{i_1}, e_{i_2}, \ldots e_{i_k}\}$. Let $p_1 = e_{i_1} + e_{i_2} + \ldots + e_{i_k} \in H_{n-r+1}$. Let $z(p_1)$ be the projection of p_1 on W_1. Let $z(p_k)$ be the projection of $p_k \in H_{n-r+k}$ on W_k. The following theorem proves that the number of non positive co-ordinates of some vector $v \in W_r$ is equal to the number of non positive co-ordinates of $z(p_1)$ (excluding the co-ordinates of $B_n - B_k$), assuming $z_{i_1}(p_r) < 0, z_{i_2}(p_{r-1}) < 0, \ldots, z_{i_r}(p_1) < 0$ for some indices $i_1, i_2, \ldots, i_r$. Equivalently, there exists a vector v in W_r, such that the number of strictly positive co-ordinates of v is equal to the number of strictly positive co-ordinates of $z(p_1)$ plus $n-k$. The minimum number of misclassifications of the points in P and Q is bounded by the number of non positive co-ordinates of a vector in $v \in W_r$.

Theorem 2.7 *The minimum number of misclassifications of any hyperplane M is bounded by the cardinality of $N_{z(p_1)}(W_1) = \{i | z_i(p_1) \leq 0, z(p_1) \in W_1, p_{1,i} > 0, p_1 \in H_{n-r+1}, p_1 = [1, 1, \ldots, 0, 1, \ldots, 1]^t\}$, for a linearly non separable problem.*

Proof Let p_k be a vector with equal co-ordinates in standard order basis of H_{n-r+k}, and the rest of the co-ordinates set to 0. Let $z(p_k)$ be the projection of p_k on $W_k, k \leq r$. Let $r = 1$. $\exists\ x$ such that $Ax = z(p_1)$. Hence, $M \leq |N_{z(p_1)}(W_1)|$. Let $r > 1$, $v = \alpha z(p_1) - z(p_2), \alpha > 1 + \max_{z_i(p_1)>0} \frac{z_i(p_2)}{z_i(p_1)}$. Hence, $v_i > 0\ \forall i \in \{j | z_j(p_1) > 0 \vee (z_i(p_1) = 0, z_i(p_2) < 0)\}$. $v \in W_2$ contains $N_{z(p_1)}(W_1) + |\{i | p_{2,i} = 0\}|$ number of co-ordinates less than or equal to zero. Similarly, let $v_3 = \alpha' v - z(p_3), \alpha' > \max_{v_i>0} \frac{z_i(p_3)}{v_i}$. The number of co-ordinates less than or equal to 0 is $N_{z(p_1)}(W_1) + |\{i | p_{3,i} = 0\}|$. By induction $v_k = \alpha v_{k-1} - z_{p_k}$ has $N_{z(p_1)}(W_1) + |\{i | p_{k,i} = 0\}|$ co-ordinates less than or equal to zero. $|\{i | p_{r,i} = 0\}| = 0$, ($p_r > \mathbf{0}$ is a vector with all co-ordinates equal in R^n). Hence v_r has $N_{z(p_1)}(W_1)$ co-ordinates less than or equal to zero. ∎

Further, it follows that the minimum number of misclassifications U_B is bounded by the minimum of $|N_{z(p_1)}(W_1)|$ and $|N_{-z(p_1)}(W_1)|$.

2.1. Examples

2.1.1. XOR Problem

Let $P = \{(0,1), (1,0)\}$, $Q = \{(0,0), (1,1)\}$. $A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & -1 \\ -1 & -1 & -1 \end{bmatrix}$. An orthonormal basis of the range of A is $U = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & -\frac{1}{2} \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{2} \\ 0 & -\frac{1}{\sqrt{2}} & \frac{1}{2} \\ -\frac{1}{\sqrt{2}} & 0 & -\frac{1}{2} \end{bmatrix}$. The projection of $p = \frac{1}{2}(1,1,1,1)^t$ on the range of A is $U(U^t p) = (0,0,0,0)^t$. Hence no strictly positive point exists in W_k. Hence P and Q are linearly non separable.

2.1.2. AND Problem

Let $P = \{(1,1)\}$, $Q = \{(0,0), (0,1), (1,0)\}$. $A = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & -1 \\ 0 & -1 & -1 \\ -1 & 0 & -1 \end{bmatrix}$. The projection of $\frac{1}{2}(1,1,1,1)^t$ on the range of A is $(0.25, 0.75, 0.25, 0.25)^t$. A strictly positive point exists in W_k. Hence P and Q are linearly separable.

2.2. Algorithm

In Algorithm 2 an iterative procedure to verify if a strictly positive point exists in a subspace W_r (range of A) is presented. It repeatedly projects a point $p \geq \mathbf{0}$ on progressively smaller subspaces. In the algorithm, projection of p on W_r is performed by first orthonormalizing the vectors which span W_r. This can be replaced by a function to solve the least squares problem $\arg\min_x \|Ax - p\|^2$. For simplicity of presentation, the recursion is described as an iterative procedure.

3. Experiments

Subspace Projection algorithm, Linear Programming Procedure, Perceptron algorithm and Support Vector Machines (SVM^{light} (Joachims, 1999)) give identical results on a representative set of benchmark datasets in UCI Machine Learning Repository (Newman et al., 1998). The SVM algorithm is trained with large value of C. C is set to 10^{12}. Experiments were performed on 3.2GHz Pentium4 machine with 512MB RAM. In Table 1, the execution time of SVM^{light} (implemented in C++) and Subspace Projection (implemented in C++) are compared (processing time includes file read). The execution time of linear programming procedure (LP - implemented as an optimized MATLAB® function `linprog`), Perceptron Algorithm (PR implemented in MATLAB®) and Subspace Projection (SP - implemented in MATLAB®) are compared in Table 2 (processing time excludes file read). The last column describes whether the data sets are Linearly Separable (LS) or not. The Perceptron algorithm does not converge on linearly non separable problems. Hence the execution time of Perceptron algorithm on linearly non

Algorithm 2 $B = \{x | Ax > \mathbf{0}, A \in R^{n\times(d+1)}, x \in R^{d+1}\}$, Test if $B \neq \emptyset$

Input: $n \times d + 1$ matrix A
$L \leftarrow 0$ //$L = 1$ if B is non-empty
$A \leftarrow$ orthonormal basis of the range of A
$r \leftarrow \text{rank}(A)$
$N \leftarrow r$
$z \leftarrow \mathbf{0} \in R^n$
while $((N \neq 0) \wedge (L == 0) \wedge (A \neq \emptyset))$ **do**
  $p \leftarrow [1, 1, \ldots, 1]^t \in R^n$
  $z \leftarrow A(A^t p)$
  if $z > \mathbf{0}$ **then**
    $L \leftarrow 1$ //by Theorem 2.1
  else if $z == \mathbf{0}$ **then**
    $N \leftarrow 1$ //by Theorem 2.4
  else if $z \geq \mathbf{0}$ **then**
    $I_z \leftarrow \{i | z_i = 0\}$ //by Theorem 2.2 and Theorem 2.3
    $i' \leftarrow 0$
    for $i = 1$ to n **do**
      if $i \in I_z$ **then**
        $i' \leftarrow i' + 1$
        $A_{i',k} \leftarrow A_{i,k}\ \forall\ k = 1, \ldots, r$
      end if
    end for
    $n \leftarrow |I_z|$
  else
    $j \leftarrow \arg\min_{i, z_i(p)<0} \frac{p_i}{p_i - z_i(p)}$ //by Theorem 2.5 and Theorem 2.6
    for $k = 1$ to r **do**
      $\alpha \leftarrow A_{j,k}/z_j(p)$
      $A_{i,k} \leftarrow A_{i,k} - \alpha z_i(p),\ \forall\ i \in 1, \ldots, n$
    end for
    $E_z \leftarrow \{i | z_i(p) < 0, A_{i,k} = 0\ \forall\ k = 1, \ldots, r\}$
    $i' \leftarrow 0$
    for $i = 1$ to n **do**
      if $i \notin E_z$ **then**
        $i' \leftarrow i' + 1$
        $A_{i',k} \leftarrow A_{i,k}\ \forall\ k = 1, \ldots, r$
      end if
    end for
    $n \leftarrow n - |E_z|$
  end if
  $A \leftarrow$ orthonormal basis of the range of A
  $r \leftarrow \text{rank}(A)$
  $N \leftarrow N - 1$
end while
$U_B \leftarrow \min(|\{i | z_i \leq 0, i = 1, \ldots, n\}|, |\{i | z_i \geq 0, i = 1, \ldots, n\}|)$ //U_B is the upper bound on the minimum number of misclassifications by any hyperplane, by Theorem 2.7
return L, U_B

separable problems is not included in Table 2.

3.1. Description of Datasets

In Iris1 dataset, Iris-setosa examples are considered to be set P, Iris-versicolor and Iris-virginica examples are considered to be set Q. In Iris2 dataset, Iris-virginica examples are considered to be set P, Iris-setosa examples and Iris-versicolor examples are considered to be set Q. The Breast Cancer dataset has attributes with values in a specified range (e.g. 0-4). Such attributes are split into two derived attributes (e.g. 0 and 4). Nominal attributes are converted into discrete values. Examples with missing values are removed from the dataset. In the Glass1 dataset, class 1 (building_windows_float_processed) feature vectors are considered to be set P and the rest of the 6 classes are considered to be set Q. Similarly, in Glass2 dataset class 2 (building_windows_non_float_processed) feature vectors are considered to be set P and the rest of the 6 classes are considered to be set Q. A small sample of the Spam dataset (Spam1) is used for testing the algorithm, because small samples of spam dataset are linearly separable. The Ionosphere dataset(Ion) is a binary classification problem of radar signals. The column containing indicies of training sample is ignored in all the datasets. E_S is the number of misclassified points of the hyperplane found by SVM, U_B is the upper bound on the misclassifications, as computed by our algorithm. Table 3 is a comparison of E_s and U_B. The test results suggest that subspace projection (SP) has much lower time complexity than LP, PR and SVM. The time complexity of SP is less than $O(nr^3)$ due to compiler optimizations. On the spam dataset, Perceptron converges fast only if the initial value of the vector x is chosen close to the optimal vector x^* satisfying $Ax^* > \mathbf{0}$.

Table 1. Computation Time SVM, and SP in seconds

DATA	n	d	r	SVM	SP	LS
IRIS1	150	4	5	0.03	0.03	YES
IRIS2	150	4	5	17.1	0.05	NO
ION	351	33	34	22.8	0.14	NO
BREAST	277	12	10	16.7	0.05	NO
GLASS1	214	9	10	46.8	0.03	NO
GLASS2	214	9	10	46.8	0.03	NO
SPAM1	359	57	58	307	0.33	YES

3.2. Scalability

The algorithm checks linear separability in linear time, if the dimension is fixed. A randomly labeled data set

Table 2. Computation Time LP, PR and SP in seconds

DATA	n	d	r	LP	PR	SP	LS
IRIS1	150	4	5	0.26	.002	.003	YES
IRIS2	150	4	5	0.46	×	.008	NO
ION	351	33	34	3.95	×	0.45	NO
BREAST	277	12	10	10.89	×	0.02	NO
GLASS1	214	9	10	4.33	×	0.02	NO
GLASS2	214	9	10	5.2	×	0.02	NO
SPAM1	359	57	58	3.97	14.1	1.20	YES

Table 3. Number of misclassifications by SVM (E_s) and Subspace Projection (U_B)

DATA	n	d	E_S	U_B
IRIS1	150	4	0	0
IRIS2	150	4	2	4
IONOSPHERE	351	33	102	126
BREAST	277	12	174	133
GLASS1	214	9	98	92
GLASS2	214	9	98	105
SPAM1	359	57	0	0

with large number of points is used to verify this result. It is a collection of linearly non separable data sets with points in 153 dimensions. Number of points n is varied between 500 and 1500 in steps of 100. The plot of CPU time in seconds of our algorithm for various values of n is presented in Figure 2. From the plot it appears that the number of computations increases almost linearly with number of points.

3.3. Conclusions

SP is a simple non-parametric algorithm. Experimental results and complexity estimates suggest SP is faster than Linear Programming, Perceptron and SVM. It has lower worst case time complexity than Convex Hull Algorithm. On linearly non-separable problems, SVM takes considerably more time than our algorithm to find a hyperplane with less error. Hence SP can be used to quickly identify an initial hyperplane with few misclassifications. Subsequently the hyperplane with least misclassifications can be obtained by training a SVM, hence reducing training time of SVM. On linearly separable problems, SP finds a separating hyperplane very fast compared to SVM if the points in the two classes are close. Identifying the maximum margin hyperplane given a separating hyperplane is geometrically not a difficult one. Hence SP can be used as a method for seeding a primal space SVM algorithm. However, our algorithm is not incremental. SP can be made incremental if incremental projection can be achieved. Another interesting problem which can be explored is a procedure to find a point in the range of A with few negative co-ordinates, by taking projections of the convex region $v > \mathbf{0}, v \in R^n$ on the range of A.

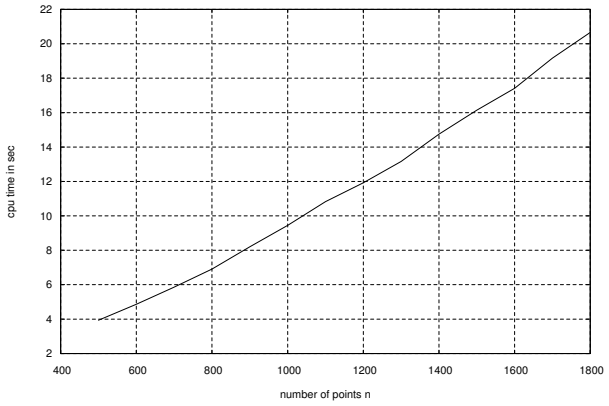

Figure 2. Computational Complexity of SP on a randomly labeled dataset

3.3.1. CONVEX HULLS

SP can be used to test if a separating hyperplane exists between a finite set of points P and a single point set $Q = \{v \in R^d\}$. Hence SP can be used to test if a point v is inside or outside the convex hull of the set P in linear time. Consequently, an incremental procedure to identify the vertices of the convex hull, without constructing the faces of the convex hull can be developed from our algorithm.

Acknowledgments

We want to thank the reviewers for their valuable comments that significantly improved the presentation.

References

Barber, C. B., Dobkin, D. P., & Huhdanpaa, H. (1996). The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software, 22*, 469–483.

Cristianini, N., & Taylor, J. S. (2000). *An introduction to support vector machines.* Cambridge: Cambridge University Press.

Duda, R. O., Hart, P. E., & Stork, D. G. (2000). *Pat-*

tern classification. New York: Wiley-Interscience.

Elizondo, D. (2006). The linear separability problem: Some testing methods. *IEEE Transactions on Neural Networks*, *17*, 330–344.

Gonzaga, C. C. (1988). An algorithm for solving linear programming problems in $O(n^3L)$ operations. In N. Megiddo (Ed.), *Progress in mathematical programming*, 1–28. Springer-Verlag.

Joachims, T. (1999). Making large-Scale SVM learning practical. In B. Schlkopf, C. Burges and A. Smola (Eds.), *Advances in Kernel Methods - Support Vector Learning*. MIT-Press.

Newman, D., Hettich, S., Blake, C., & Merz, C. (1998). UCI repository of machine learning databases.

Tsang, I. W., Kwok, J. T., & Cheung, P.-M. (2005). Core vector machines: Fast SVM Training on Very Large Data Sets. *Journal of Machine Learning Research*, *6*, 363–392.

Wasserman, P. (1989). *Neural computing theory and practice*. Van Nostrand Reinhold.