Int. J. Appl. Math. Comput. Sci., 2005, Vol. 15, No. 2, 295–304

STRICT MAXIMUM SEPARABILITY OF TWO FINITE SETS: AN ALGORITHMIC APPROACH

DOROTA CENDROWSKA

Polish-Japanese Institute of Information Technology
Koszykowa 86, 02–008 Warsaw, Poland
e-mail: ddan@pjwstk.edu.pl

The paper presents a recursive algorithm for the investigation of a strict, linear separation in the Euclidean space. In the case when sets are linearly separable, it allows us to determine the coefficients of the hyperplanes. An example of using this algorithm as well as its drawbacks are shown. Then the algorithm of determining an optimal separation (in the sense of maximizing the distance between the two sets) is presented.

Keywords: binary classifiers, recursive methods, optimal separability

"*The states of things are not just simply good or bad, and actions are neither erroneous nor plausible. All kinds of worth scales have completely relative character.*"

I.D. Bross

1. Introduction

The problem of pattern classification is one of the most frequently occurring problems in the field of artificial intelligence. Therefore, there is a wide range of methods for correct classification of known patterns as well as new ones (with unknown attachment). One of these methods is the k-Nearest-Neighbour rule. This rule classifies x by assigning to it the label which is most frequently represented among the k nearest labelled samples. What should be done is to specify the value of k (if the metric is known).

It should be mentioned that, as k increases, the k-Nearest-Neighbour rule becomes optimal, close to the best theoretically known classifier—the Bayes classifier. On the other hand, a disadvantage of the k-Nearest-Neighbour rule is the necessity of keeping a set of labelled samples during the classification process. The set should be rather large for correct classification. In order to eliminate the need of having this set, one should approximate the classifier based on the k-Nearest-Neighbour rule by a piecewise classifier. Therefore, to define this classifier, an algorithm of investigating the linear separability is needed. Such algorithms may be grouped according to their nature. The first group comprises gradient methods (Duda *et al.*, 1995), which minimize an error function. In general, except the Ho-Kashyap algorithm (Duda *et al.*, 1995), these methods do not enable us to find out if sets are non-separable. But there is no known estimate of the maximum number of steps to ensure that it is impossible to separate linearly sets in the case of Ho-Kashyap's algorithm.

Another approach is to use algorithms which determine the points from $X = X_1 \cup X_2$ or their convex combinations. The separating hyperplane is spread on them. Kozinec's algorithm (Kozinec, 1973) and its modifications (Franc and Hlaváč, 2003) belong to this group. A drawback of these algorithms is the assumption that the sets are linearly separable. Sometimes it is impossible to fulfil this condition *a priori*. Jóźwik's algorithm (Jóźwik, 1983), though it belongs to that group, does not have this disadvantage. Additionally, in this algorithm an upper estimate of the number of steps needed to decide whether sets are not separable (Jóźwik, 1998) is known.

Moreover, support vector machines methods should be mentioned. They allow us to separate sets which are originally non-separable. But this is done by investigating the solution in a space whose dimension is much higher than the original one (Cristianini and Shawe-Taylor, 2000; Jankowski, 2003; Mangasarian, 2000). Unlike the above-mentioned group, the algorithm which is

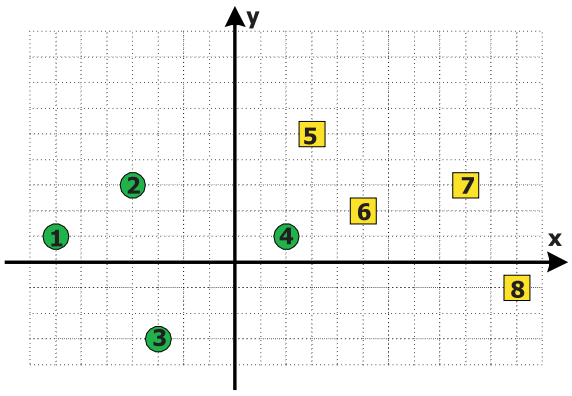
presented here finds a solution in the space E^{n+2}, where n is the dimension of the original space.

In this paper an example of using the original algorithm proposed by Jóźwik (1975) is presented to show its disadvantage. Then modifications are presented which were made to separate correctly (if possible) all samples from two finite sets. In general, both sets consist of n-dimensional vectors in E^n, though in the presented examples which ilustrate the mentioned algorithms the sets are two-dimensional to make the analysis of the algorithms easier.

2. Strict Versus Linear Separability of Two Sets

The presentation of the algorithm regarding strict separability of two finite sets commences with providing an example of the use of the algorithm presented by Jóźwik (1998). The example shows a property of the found separating hyperplane that does not assure strict separability, despite the fact that the division of the sets in such a way is possible. This gives a basis to define the algorithm which assures that every vector in the separated sets will be separated correctly in the case of linear separability.

In the examples presented below, the sets X_1, X_2 shown in Fig. 1 are considered.



$$X_1 = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4\}$$
$$X_2 = \{\boldsymbol{x}_5, \boldsymbol{x}_6, \boldsymbol{x}_7, \boldsymbol{x}_8\}$$

$$x_1 = [-7, 1] \quad x_2 = [-4, 3]$$
$$x_3 = [-3, -3] \quad x_4 = [2, 1]$$
$$x_5 = [3, 5] \quad x_6 = [5, 2]$$
$$x_7 = [9, 3] \quad x_8 = [11, -1]$$

Fig. 1. The sets X_1 i X_2.

2.1. Linear Separability

The sets X_1 and X_2 are linearly separated if there exists a function $g(\boldsymbol{x})$ such that

$$\begin{cases} g(\boldsymbol{x}) \geq 0 & \text{when } \boldsymbol{x} \in X_1, \\ g(\boldsymbol{x}) \leq 0 & \text{when } \boldsymbol{x} \in X_2, \\ g(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i x_i + \alpha, \\ \sum_{i=1}^{n} \alpha_i^2 \neq 0. \end{cases} \tag{1}$$

The condition (1) related to separability can be presented as one condition if the transformation below is used:

$$\boldsymbol{t} = f(\boldsymbol{x}), \quad \boldsymbol{x} \in X = X_1 \cup X_2,$$
$$f(\boldsymbol{x}) = \begin{cases} [x_1, x_2, \ldots, x_n, 1] & \text{when } \boldsymbol{x} \in X_1, \\ [-x_1, -x_2, \ldots, -x_n, -1] & \text{when } \boldsymbol{x} \in X_2. \end{cases} \tag{2}$$

Then the first part of the condition (1) amounts to

$$\langle \boldsymbol{a}, \boldsymbol{t} \rangle \geq 0, \quad \boldsymbol{t} \in T = \{\boldsymbol{x} \in X_1 \cup X_2 : f(\boldsymbol{x})\}, \tag{3}$$

where $\langle \cdot, \cdot \rangle$ means a scalar product, and $\boldsymbol{a} = [\alpha_1, \alpha_2, \ldots, \alpha_n, \alpha] \in \mathrm{E}^{n+1}$.

It is important to notice that after the transformation (2) it is still possible to establish whether each $\boldsymbol{t}$ belongs to X_1 or X_2. This is done by the $(n+1)$-th component of $\boldsymbol{t}$.

The condition (3) is interpreted in the following way: The first n components of the vector $\boldsymbol{a}$ in the original space E^n form a normal vector to the hyperplane $\boldsymbol{H}$. On the other hand, the vector $\boldsymbol{a}$ in the space E^{n+1} is a normal vector to the separating hyperplane going through the point $(0, \ldots, 0) \in E^{n+1}$ in such a way that all vectors complying with elements in X after the transformation (1) are placed in the same part of the space E^{n+1} produced as a result of the separation of E^{n+1} by this hyperplane.

Unlike gradient methods, the algorithm examining linear separability of two sets (shown below) allows finding the right vector $\boldsymbol{a}$ in a finite number of steps (an upper estimate of the number of steps can be found in (Jóźwik, 1975)).

A basic property of the hyperplane found as $\boldsymbol{a}$, as well as the consequence of the use of the algorithm (unambiguously defined by the vector $\boldsymbol{a}$), will be presented by executing the algorithm step by step to separate the sets X_1 and X_2.

Therefore the algorithm LS2S presented in (Jóźwik, 1998) will be cited. The algorithm requires two parameters:

- T_p – a list or a sublist of points to be separated,
- K_p – a set of points through which the separating hyperplane is to be passing.

At the beginning we set $p = 0$. All given points are to be separated, so the first parameter is equal to $T_0 = \{f(\boldsymbol{x}) \in E^{n+1} : \boldsymbol{x} \in X_1 \cup X_2 \in E^n\}$. There is no information about points to be passed through the separating hyperplane, so $K_0 = \emptyset$.

In the original algorithm (Jóźwik, 1998) the second parameter was the space where the solution was sought, i.e., at the beginning the space $S_0 = E^{n+1}$. This alteration is made to simplify the understanding of the algorithm. This space is specified in the first step of the algorithm.

The label p is used to make the algorithm more legible, and to easily know the current level of recursion at every moment. In the numbering of algorithm steps, p expresses how many recursion calls (in depth) have already been done, so $n - p$ may still be done.

In the gradient methods the process of finding a satisfactory solution $\boldsymbol{a}_g$ is performed step by step by correcting the components of the current solution. In the presented algorithm every recursion call adds one point to the set of points to be "a good candidate" to go through the separating hyperplane. This is so because in order to separate correctly an incorrectly separated point this point must at least go through the separating hyperplane. So the maximum depth of recursion calls is specified by the original size (n) of vectors in the sets X_1 and X_2. In Fig. 2 examples of the current solutions are presented (in the original space) according to the level of recursion. Unlike the gradient methods (in the figure the examples $\boldsymbol{a}_g$), the solution is always built on the points from the list which are to be separated (except for the first calculation of the vector $\boldsymbol{a}$ in Step 0.2).

In Fig. 2 the points $\boldsymbol{t}_1$ and $\boldsymbol{t}_2$ are incorrectly separated by the line $\boldsymbol{a}_{0.2}$ (the solution $\boldsymbol{a}$ obtained in Step 0.2 of the algorithm). Thus the recursion call is done for a new list without these points: $T_1 = T_0 \setminus \{\boldsymbol{t}_1, \boldsymbol{t}_2\}$. The point $\boldsymbol{t}_1$ is chosen to go through the separating plane $\boldsymbol{a}_{1.2}$, so $K_1 = \{\boldsymbol{t}_1\}$. It is enough to separate the list T_1 but not enough to separate all points from the list T_0, so another recursive call will be done.

As a result, two parameters are obtained: the number *flag* and the vector $\boldsymbol{a}$. The former is information on the separability of the sets: when *flag* $= -1$, the sets are inseparable linearly but when *flag* $= 1$, the sets are separable. The latter is important only in the case of sets

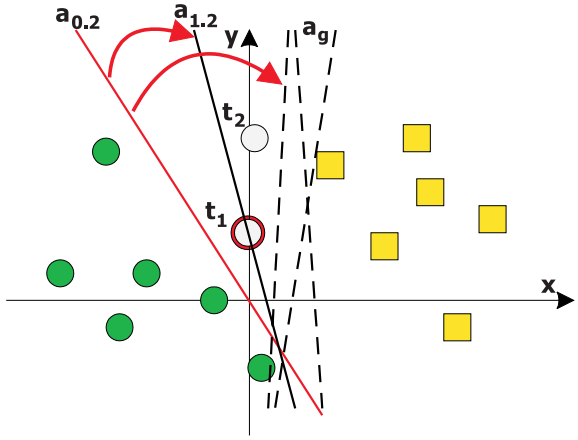


Fig. 2. Example of Step 2 of the LS2S algorithm on two levels of recursion.

being linearly separable—it defines a vector whose first n components form a normal vector to the separating hyperplane.

In the cited algorithm its second point is modified so as to express uniquely the choice of the vector from the list T_p.

The header of Algorithm LS2S: $(\boldsymbol{a}, \mathit{flag}) \leftarrow \mathrm{LS2S}(T_p,\ K_p)$

The body of the algorithm:

p.1 $S_p = E^{n+1} \cap K_p^{\perp}$ – the space where the solution is to be found;

p.2 $\boldsymbol{a}$ – any vector from the list T_p whose projection on the subspace S_p is different from zero. *In the case when there are several non-zero projections of the vectors from the list T_p with such properties, the projection of the vector that is the first in the list T_p is chosen*;

p.3 if in T_p there is no vector whose orthogonal projection on S_p is not equal to zero, then set *flag* $= -1$ and proceed to p.14;

p.4 $NC_p = \{\boldsymbol{t} \in T_p : \langle \boldsymbol{a}, \boldsymbol{t} \rangle < 0\}$ – the list of incorrectly separated points;

p.5 if $NC_p = \emptyset$, then set *flag* $= 1$ and proceed to p.14, i.e., the points from the list T_p are separated correctly;

p.6 if $p = n$ and $NC_p \neq \emptyset$, *flag* $= -1$ then proceed to p.14, i.e., it is impossible to define a hyperplane in E^{n+1} which contains more than $n+1$ non-colinear points (n points from K_p and the origin: $\mathcal{O} = (0,\ 0,\ \ldots,\ 0)$;

p.7 $T_{p+1} = T_p \setminus NC_p$;

p.8 $\boldsymbol{b}$ – the first vector from the list NC_p;

p.9 $K_{p+1} = K_p \cup \{\boldsymbol{b}\}$;

p.10 execute the algorithm
$(\boldsymbol{a}, \textit{flag}) \leftarrow$LS2S$(T_{p+1}, K_{p+1})$;

p.11 if $\textit{flag} = -1$, then proceed to $p.14$;

p.12 $NC_p = \{\boldsymbol{t} \in NC_p : \langle \boldsymbol{a}, \boldsymbol{t}\rangle < 0\}$;

p.13 if $NC_p \neq \emptyset$, then proceed to $p.7$;

p.14 if $\textit{flag} = -1$, then the sets are not linearly separable; if, on the other hand, $\textit{flag} = 1$ and $p = n$, then the sets are linearly separable and the first n components of the vector $\boldsymbol{a}$ form the sought normal vector to the separating hyperplane.

For better undestanding of the proposed algorithm in the space E^3, the solution of the problem for the list

$$\begin{aligned} T_0 &= \{\boldsymbol{t}_1, \boldsymbol{t}_2, \boldsymbol{t}_3, \boldsymbol{t}_4, \boldsymbol{t}_5, \boldsymbol{t}_6, \boldsymbol{t}_7, \boldsymbol{t}_8\} \\ &= \{f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), f(\boldsymbol{x}_3), f(\boldsymbol{x}_4), \\ &\quad f(\boldsymbol{x}_5), f(\boldsymbol{x}_6), f(\boldsymbol{x}_7), f(\boldsymbol{x}_8)\} \end{aligned}$$

will be sought.

The algorithm is executed as follows: LS2S(T_0, $\emptyset$). The Gram-Schmidt orthogonalization is used in Step 2.

0.1 $S_0 = E^3$

0.2 $\boldsymbol{a} = \boldsymbol{t}_1$

0.3 the condition is not satisfied

0.4 $NC_0 = \{\boldsymbol{t}_4\}$

0.5 the condition is not satisfied

0.6 the condition is not satisfied

0.7 $T_1 = \{\boldsymbol{t}_1, \boldsymbol{t}_2, \boldsymbol{t}_3, \boldsymbol{t}_5, \boldsymbol{t}_6, \boldsymbol{t}_7, \boldsymbol{t}_8\}$

0.8 $\boldsymbol{b} = \boldsymbol{t}_4$

0.9 $K_1 = \emptyset \cup \{\boldsymbol{t}_4\} = \{\boldsymbol{t}_4\}$

0.10 execute the algorithm
$(\boldsymbol{a}, \textit{flag}) \leftarrow$LS2S$(T_1, K_1)$

1.1 $S_1 = E^3 \cap \{\boldsymbol{t}_4\}^{\perp}$

1.2 $\boldsymbol{a} = \boldsymbol{t}_1 - \dfrac{\langle \boldsymbol{t}_4, \boldsymbol{t}_1\rangle}{\langle \boldsymbol{t}_4, \boldsymbol{t}_4\rangle}\boldsymbol{t}_4 = 3\begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix}$

1.3 the condition is not satisfied

1.4 $NC_1 = \{\boldsymbol{t}_5\}$

1.5 the condition is not satisfied

1.6 the condition is not satisfied

1.7 $T_2 = \{\boldsymbol{t}_1, \boldsymbol{t}_2, \boldsymbol{t}_3, \boldsymbol{t}_6, \boldsymbol{t}_7, \boldsymbol{t}_8\}$

1.8 $\boldsymbol{b} = \boldsymbol{t}_5$

1.9 $K_2 = K_1 \cup \{\boldsymbol{t}_5\} = \{\boldsymbol{t}_4, \boldsymbol{t}_5\}$

1.10 execute the algorithm
$(\boldsymbol{a}, \textit{flag}) \leftarrow$LS2S$(Z_2, K_2)$

2.1 $S_2 = E^3 \cap \{\boldsymbol{t}_4, \boldsymbol{t}_5\}^{\perp}$

2.2 $\boldsymbol{a} = \boldsymbol{t}_1 - \dfrac{\langle \boldsymbol{t}_4, \boldsymbol{t}_1\rangle}{\langle \boldsymbol{t}_4, \boldsymbol{t}_4\rangle}\boldsymbol{t}_4 - \dfrac{\langle \boldsymbol{p}, \boldsymbol{t}_1\rangle}{\langle \boldsymbol{p}, \boldsymbol{p}\rangle}\boldsymbol{p}$

$= \dfrac{6}{11}\begin{bmatrix} -4 \\ 1 \\ 7 \end{bmatrix}$

where $\boldsymbol{p} = \boldsymbol{t}_5 - \dfrac{\langle \boldsymbol{t}_4, \boldsymbol{t}_5\rangle}{\langle \boldsymbol{t}_4, \boldsymbol{t}_4\rangle}\boldsymbol{t}_4 = \begin{bmatrix} 1 \\ -3 \\ 1 \end{bmatrix}$

2.3 the condition is not satisfied

2.4 $NC_2 = \emptyset$

2.5 the condition is satisfied

2.14 the condition is not satisfied

1.11 the condition is not satisfied

1.12 $NC_1 = \emptyset$

1.13 the condition is not satisfied

1.14 the condition is not satisfied

0.11 the condition is not satisfied

0.12 $NC_0 = \emptyset$

0.13 the condition is not satisfied

0.14 the condition is satisfied and $\boldsymbol{a}$ is the searched vector.

The obtained solution $\boldsymbol{a}$ is a normal vector to the plane separating the points from the list T_0 placed after the transformation f in the space E^3—the plane goes through the origin $\mathcal{O} = (0, 0, 0)$. Simultaneously, the vector $\boldsymbol{a}$ determines the straight line separating on the plane the points from the original set $X = X_1 \cup X_2$. The straight line goes through the points x_4 and x_5, making the separation of the remaining points from the set X possible. But the obtained solution (I) does not allow us to separate the points x_4 and x_5, although in this case it is easy to show solutions (II, III) which would not have that drawback. So, the most important property is the possibility to separate, if possible, all points from the set X.

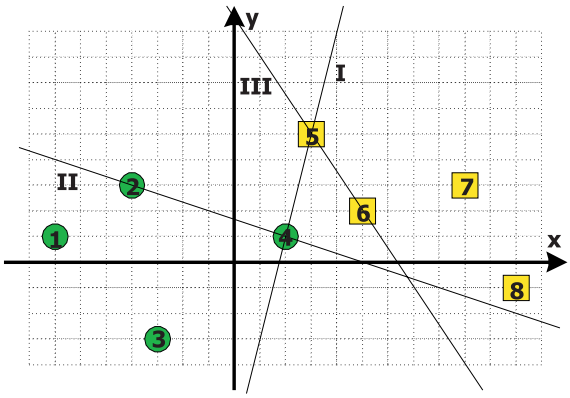


Fig. 3. The set of possible solutions to obtain after using the algorithm LS2S.

The discussed example raises two questions: first, is it possible, in the case where there are multiple solutions, to obtain other solutions by means of the described algorithm? And second, what does obtaining a given solution depend on? The answer is: it depends on the chosen definition of the word *whichever* in Step p.2 of the algorithm, and on the way in which the incorrectly separated point is selected in Step p.8. That is why T_p and NC_p are not sets but lists—some order relation must be defined. The sequence of elements in these lists does have some influence on the form of the solution.

The solutions already mentioned (II and III) will be obtained after executing the algorithm for the following lists, respectively:

$$T_0 = \{f(\boldsymbol{x}_5), f(\boldsymbol{x}_2), f(\boldsymbol{x}_8), f(\boldsymbol{x}_1), f(\boldsymbol{x}_3), f(\boldsymbol{x}_4), f(\boldsymbol{x}_6), f(\boldsymbol{x}_7)\},$$

$$T_0 = \{f(\boldsymbol{x}_4), f(\boldsymbol{x}_5), f(\boldsymbol{x}_6), f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), f(\boldsymbol{x}_3), f(\boldsymbol{x}_7), f(\boldsymbol{x}_8)\}.$$

2.2. Strict Separability

Two sets are strictly separable when there exists at least one separating hyperplane $\boldsymbol{H}$ which assures a correct separation of all elements in the separated sets (the solution I from the previous point does not fulfil that condition). In practice, if there exist one such hyperplane, at the same time there exist an infinite number of them, which still applies to the above-mentioned condition. A few examples of separable straight lines that enable a correct separation of each point are presented below. The abundance of solutions brings up the question which of them is the best one.

The question is impossible to be answered satisfactorily. But we may replace the question about the "*taste*" with a far more pragmatic one: which of the hyperplanes can be calculated most quickly (i.e., in the smallest amount of steps)?

The algorithm that finds a hyperplane enabling us to separate the sets (if such a hyperplane exists) is presented below.

The algorithm uses the following observation: among all the possible solutions assuring strict separability of sets, **there is always one** (and only one) quite exceptional. It is characterized by the fact that the minimum distance between the hyperplane and the points representing the elements of the sets is maximum. For elements belonging to the plane an example of such a hyperplane is the straight-line h in Fig. 4.

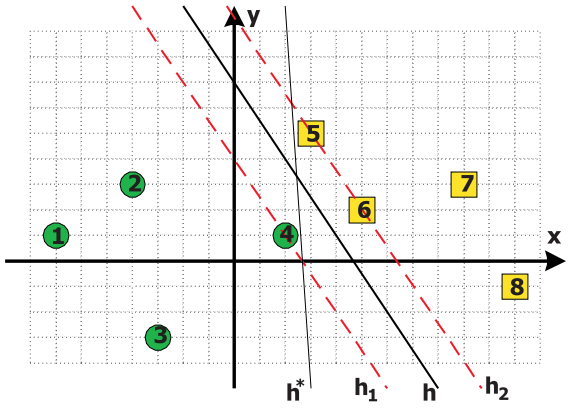


Fig. 4. Some of the instances of the strict separability of the exemplary sets.

In the next steps the algorithm does not determine the characteristic hyperplane h, but two lines parallel to it, h_1 and h_2, cf. Fig. 4.

The algorithm thus looks for a solution such that

$$\begin{cases} h_1(\boldsymbol{x}_1) \geq 0 \ \wedge \ h_2(\boldsymbol{x}_1) > 0, \\ h_1(\boldsymbol{x}_2) < 0 \ \wedge \ h_2(\boldsymbol{x}_2) \leq 0, \\ \boldsymbol{x}_1 \in X_1, \quad \boldsymbol{x}_2 \in X_2, \\ h_1(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i x_i + \alpha, \\ h_2(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i x_i + \alpha + \varepsilon, \\ \sum_{i=1}^{n} \alpha_i = 1. \end{cases} \tag{4}$$

If in the solution $\varepsilon > 0$, then the sets are strictly separable. The conditions related to the hyperplanes h_1 and h_2, which are sought, can be presented as one condition, and therefore to make the algorithm less complicated we use

the transformation below:

$$\boldsymbol{y} = f^*(\boldsymbol{x}), \quad \boldsymbol{x} \in X = X_1 \cup X_2,$$

$$f^*(\boldsymbol{x}) = \begin{cases} [x_1, x_2, \ldots, x_n, 0, 1] & \text{when } \boldsymbol{x} \in X_1, \\ [-x_1, -x_2, \ldots, -x_n, -1, -1] & \text{when } \boldsymbol{x} \in X_2. \end{cases} \quad (5)$$

Then conditions equivalent to (4) are

$$\langle \boldsymbol{a}^*, \boldsymbol{y} \rangle \geq 0, \quad \boldsymbol{y} \in Y = \{\boldsymbol{x} \in X_1 \cup X_2 : f^*(\boldsymbol{x})\}, \quad (6)$$

where $\boldsymbol{a}^* = [\alpha_1, \alpha_2, \ldots \alpha_n, \varepsilon, \alpha] \in \mathrm{E}^{n+2}$.

Thus the task of the algorithm is to define $\boldsymbol{a}^*$ describing an accepted solution allowing us to determine the hyperplanes h_1, h_2 or the hyperplane h. Algorithm SLS2S shown further finds a solution which is **not only strict but with a maximum value of** ε.

As was presented in Algorithm LS2S, the calculated solution is unambiguously defined only in Step n.2. That is the reason why getting different solutions is possible.

The idea of Algorithm SLS2S is as follows: To define in the space E^n $(n \geq 2)$ two parallel $(n-1)$-dimensional hyperplanes, which additionally are as distant from each other as possible, at least two points must be known, one from the set X_1 and one from X_2. In this case, which is the simplest one, the distance between these points is at the same time the maximum value of the component ε. After the transformation (5) it is still possible to establish whether each $\boldsymbol{y}$ belongs to X_1 or X_2. It is done by the $(n+2)$-th component of $\boldsymbol{y}$. Steps p.2 and p.3 of Algorithm SLS2S are to make sure that those two points are given.

Therefore, to make the algorithm more legible, the parameter K_p, which is a subset of points through which the separating hyperplanes h_1, h_2 are to be passing, is altered by two subsets $K_p^{(1)}$ and $K_p^{(2)}$ to get to know easily which points originally belong to X_1 or X_2.

The algorithm presented further requires three parameters:

- Y_p – a list or a sublist of points to be separated,
- $K_p^{(1)}$ – a set of points through which the separating hyperplane h_1 is to be passing,
- $K_p^{(2)}$ – a set of points through which the separating hyperplane h_2 is to be passing.

At the beginning, we set $p = 0$. All given points are to be separated, so the first parameter is equal to $Y_0 = \{f^*(\boldsymbol{x}) \in E^{n+2} : \boldsymbol{x} \in X_1 \cup X_2 \in E^n\}$. There is no information about points to be passed through the separating hyperplanes, so $K_0^{(1)} = \emptyset$ and $K_0^{(2)} = \emptyset$.

Additionally, to make Algorithm SLS2S more legible, the following function is defined:

$$\mathrm{class}(\boldsymbol{y}) = \begin{cases} 1 & \text{when the } (n+2)\text{-th component} \\ & \text{of } \boldsymbol{y} \text{ is equal to } 1, \\ 2 & \text{when the } (n+2)\text{-th component} \\ & \text{of } \boldsymbol{y} \text{ is equal to } -1. \end{cases}$$

The header of Algorithm SLS2S: $(\boldsymbol{a}^*, \mathit{flag}) \leftarrow \mathrm{SLS2S}(Y_p, K_p^{(1)}, K_p^{(2)})$

The body of the algorithm:

p.1 let $K = \emptyset$ be a temporary set of points through which the separating hyperplanes h_1, h_2 are to be passing;

p.2 if card $\left(K_p^{(1)}\right) = 0$, then $K = \{\text{any } \boldsymbol{y} \in Y_p \text{ with } \mathrm{class}(\boldsymbol{y}) = 1\}$;

p.3 if card $\left(K_p^{(2)}\right) = 0$, then $K = K \cup \{\text{any } \boldsymbol{y} \in Y_p \text{ with } \mathrm{class}(\boldsymbol{y}) = 2\}$;

p.4 $\boldsymbol{a}^*$ – the vector in E^{n+2} perpendicular to every vector belonging to the set $K \cup K_p^{(1)} \cup K_p^{(2)}$, with the maximum component value of $\alpha_{n+1} = \varepsilon$ (there is only one such vector);

p.5 $NC_p = \{\boldsymbol{y} \in Y_p : \langle \boldsymbol{a}^*, \boldsymbol{y} \rangle < 0\}$, i.e., the list of incorrectly separated points;

p.6 if $NC_p = \emptyset$, then set $\mathit{flag} = 1$ and proceed to p.17, i.e., the points from the list T_p are separated correctly;

p.7 if $p = n + 1$ and $NC_p \neq \emptyset$, then set $\mathit{flag} = -1$ and proceed to p.17;

p.8 $Y_{p+1} = Y_p \setminus NC_p$;

p.9 $\boldsymbol{b}$ – the first vector from the list NC_p;

p.10 $c = \mathrm{class}\,(\boldsymbol{b})$;

p.11 if card $\left(K_p^{(c)}\right) < n$,
then $K_{p+1}^{(c)} = K_p^{(c)} \cup \{\boldsymbol{b}\}$, $K_{p+1}^{(3-c)} = K_p^{(3-c)}$

p.12 if card $\left(K_p^{(c)}\right) = n$, then set $\mathit{flag} = -1$ and proceed to p.17;

p.13 execute the algorithm
$(\boldsymbol{a}^*, \mathit{flag}) \leftarrow \mathrm{SLS2S}(Y_{p+1}, K_{p+1}^{(1)}, K_{p+1}^{(2)})$;

p.14 if $\mathit{flag} = -1$, then proceed to p.17;

p.15 $NC_p = \{\boldsymbol{y} \in NC_p : \langle \boldsymbol{a}^*, \boldsymbol{y} \rangle < 0\}$;

p.16 if $NC_p \neq \emptyset$, then proceed to p.8;

p.17 if *flag* $= -1$, then the sets are not linearly separable; if, on the other hand, *flag* $= 1$ and $p = 0$, then the sets are linearly separable and the vector $\boldsymbol{a}^*$ is the sought vector, describing the sought hyperplanes h_1 and h_2, and, additionally, the solution assures the maximum separability.

A closer look at the executing of the algorithm for the list Y_0 ordered in the way shown below seems to be necessary:

$$\begin{aligned} Y_0 &= \{\boldsymbol{y}_4, \boldsymbol{y}_5, \boldsymbol{y}_2, \boldsymbol{y}_3, \boldsymbol{y}_1, \boldsymbol{y}_6, \boldsymbol{y}_7, \boldsymbol{y}_8\} \\ &= \big\{f^*(\boldsymbol{x}_4), f^*(\boldsymbol{x}_5), f^*(\boldsymbol{x}_2), f^*(\boldsymbol{x}_3), f^*(\boldsymbol{x}_1), \\ &\qquad f^*(\boldsymbol{x}_6), f^*(\boldsymbol{x}_7), f^*(\boldsymbol{x}_8)\big\}. \end{aligned}$$

The algorithm is executed as follows: SLS2S($Z_4, \emptyset, \emptyset$).

0.1 $K = \emptyset$

0.2 the condition is satisfied and $K = \{\boldsymbol{y}_4\}$

0.3 the condition is satisfied and $K = K \cup \{\boldsymbol{y}_5\} = \{\boldsymbol{y}_4, \boldsymbol{y}_5\}$

0.4 solve the after-mentioned equations (only one solution exists):

$$\begin{cases} \langle \boldsymbol{a}^*, \boldsymbol{y}_4\rangle = 0 \\ \langle \boldsymbol{a}^*, \boldsymbol{y}_5\rangle = 0 \\ \alpha_1^2 + \alpha_2^2 = 1 \\ \max(\varepsilon) \end{cases} \Longrightarrow \begin{cases} 2\alpha_1 + \alpha_2 + \alpha = 0, \\ -3\alpha_1 - 5\alpha_2 - \varepsilon - \alpha = 0, \\ \alpha_1^2 + \alpha_2^2 = 1, \\ \max(\varepsilon). \end{cases}$$

Determining α from the first equation and α_1 from the second after substitution in the third one gives

$$17\alpha_2^2 + 8\varepsilon\alpha_2 + \varepsilon^2 - 1 = 0.$$

This equation is treated as a quadratic equation where ε is a parameter. Then

$$\Delta_{\alpha_2} = -4\varepsilon^2 + 68.$$

To satisfy the fourth condition, the maximum value of $\varepsilon = \sqrt{17}$ is set. As a result, only one solution passing through the points $\boldsymbol{x}_4$ and $\boldsymbol{x}_5$ exists:

$$\boldsymbol{a}^* = \frac{\sqrt{17}}{17}\begin{bmatrix} -1 \\ -4 \\ 17 \\ 6 \end{bmatrix}.$$

0.5 $NC_0 = \{\boldsymbol{y}_2, \boldsymbol{y}_6, \boldsymbol{y}_7, \boldsymbol{y}_8\}$

0.6 the condition is not satisfied

0.7 the condition is not satisfied

0.8 $Y_1 = \{\boldsymbol{y}_4, \boldsymbol{y}_5, \boldsymbol{y}_3, \boldsymbol{y}_1\}$

0.9 $\boldsymbol{b} = \boldsymbol{y}_2$

0.10 $c = 1$;

0.11 the condition is satisfied and $K_1^{(1)} = \{\boldsymbol{y}_2\}$, $K_1^{(2)} = K_0^{(2)} = \emptyset$

0.12 the condition is not satisfied

0.13 execute the algorithm
($\boldsymbol{a}^*$, *flag*) ←SLS2S($Y_1, K_1^{(1)}, K_1^{(2)}$)

1.1 $K = \emptyset$

1.2 the condition is not satisfied

1.3 the condition is satisfied and $K = \{\boldsymbol{y}_5\}$

1.4

$$\begin{cases} \langle \boldsymbol{a}^*, \boldsymbol{y}_2\rangle = 0 \\ \langle \boldsymbol{a}^*, \boldsymbol{y}_5\rangle = 0 \\ \alpha_1^2 + \alpha_2^2 = 1 \\ \max(\varepsilon) \end{cases} \Longrightarrow \boldsymbol{a}^* = \frac{\sqrt{53}}{53}\begin{bmatrix} -7 \\ -2 \\ 53 \\ -22 \end{bmatrix},$$

1.5 $NC_1 = \{\boldsymbol{y}_4\}$

1.6 the condition is not satisfied

1.7 the condition is not satisfied

1.8 $Y_2 = \{\boldsymbol{y}_5, \boldsymbol{y}_3, \boldsymbol{y}_1\}$

1.9 $\boldsymbol{b} = \boldsymbol{y}_4$

1.10 $c = 1$;

1.11 the condition is satisfied and $K_2^{(1)} = K_1^{(1)} \cup \{\boldsymbol{y}_4\} = \{\boldsymbol{y}_2, \boldsymbol{y}_4\}$, $K_2^{(2)} = K_1^{(2)} = \emptyset$

1.12 the condition is not satisfied

1.13 execute the algorithm
($\boldsymbol{a}^*$, *flag*) ←SLS2S($Y_2, K_2^{(1)}, K_2^{(2)}$)

2.1 $K = \emptyset$

2.2 the condition is not satisfied

2.3 the condition is satisfied and $K = \{\boldsymbol{y}_5\}$

2.4

$$\begin{cases} \langle \boldsymbol{a}^*, \boldsymbol{y}_2\rangle = 0 \\ \langle \boldsymbol{a}^*, \boldsymbol{y}_4\rangle = 0 \\ \langle \boldsymbol{a}^*, \boldsymbol{y}_5\rangle = 0 \\ \alpha_1^2 + \alpha_2^2 = 1 \\ \max(\varepsilon) \end{cases} \Longrightarrow \boldsymbol{a}^* = \frac{\sqrt{10}}{10}\begin{bmatrix} -1 \\ -3 \\ 13 \\ 5 \end{bmatrix},$$

2.5 $NC_2 = \emptyset$

2.6 the condition is satisfied

2.17 the condition is not satisfied

1.14 the condition is not satisfied

1.15 $NC_1 = \emptyset$

1.16 the condition is not satisfied

1.17 the condition is not satisfied

0.14 the condition is not satisfied

0.15 $NC_0 = \{\boldsymbol{y}_6, \boldsymbol{y}_8\}$

0.16 the condition is satisfied

0.8 $Y_1 = \{\boldsymbol{y}_4, \boldsymbol{y}_5, \boldsymbol{y}_2, \boldsymbol{y}_3, \boldsymbol{y}_1, \boldsymbol{y}_7\}$

0.9 $\boldsymbol{b} = \boldsymbol{y}_6$

0.10 $c = 2$;

0.11 the condition is satisfied and $K_1^{(1)} = K_0^{(1)} = \emptyset$, $K_1^{(2)} = \{\boldsymbol{y}_6\}$

0.12 the condition is not satisfied

0.13 execute the algorithm
$(\boldsymbol{a}^*, \mathit{flag}) \leftarrow$SLS2S$(Y_1, K_1^{(1)}, K_1^{(2)})$

1.1 $K = \emptyset$

1.2 the condition is satisfied and $K = \{\boldsymbol{y}_4\}$

1.3 the condition is not satisfied

1.4

$$\begin{cases} \langle \boldsymbol{a}^*, \boldsymbol{y}_4 \rangle = 0 \\ \langle \boldsymbol{a}^*, \boldsymbol{y}_6 \rangle = 0 \\ \alpha_1^2 + \alpha_2^2 = 1 \\ \max(\varepsilon) \end{cases} \Longrightarrow \boldsymbol{a}^* = \frac{\sqrt{10}}{10} \begin{bmatrix} -3 \\ -1 \\ 10 \\ 7 \end{bmatrix},$$

1.5 $NC_1 = \{\boldsymbol{y}_5\}$

1.6 the condition is not satisfied

1.7 the condition is not satisfied

1.8 $Y_2 = \{\boldsymbol{y}_4, \boldsymbol{y}_2, \boldsymbol{y}_3, \boldsymbol{y}_1, \boldsymbol{y}_7\}$

1.9 $\boldsymbol{b} = \boldsymbol{y}_5$

1.10 $c = 2$;

1.11 the condition is satisfied and $K_2^{(1)} = K_1^{(1)} = \emptyset$, $K_2^{(2)} = K_1^{(2)} \cup \{\boldsymbol{y}_5\} = \{\boldsymbol{y}_5, \boldsymbol{y}_6\}$

1.12 the condition is not satisfied

1.13 execute the algorithm
$(\boldsymbol{a}^*, \mathit{flag}) \leftarrow$SLS2S$(Z_2, K_2^{(1)}, K_2^{(2)})$

2.1 $K = \emptyset$

2.2 the condition is satisfied and $K = \{\boldsymbol{y}_4\}$

2.3 the condition is not satisfied

2.4

$$\begin{cases} \langle \boldsymbol{a}^*, \boldsymbol{y}_4 \rangle = 0 \\ \langle \boldsymbol{a}^*, \boldsymbol{y}_5 \rangle = 0 \\ \langle \boldsymbol{a}^*, \boldsymbol{y}_6 \rangle = 0 \\ \alpha_1^2 + \alpha_2^2 = 1 \\ \max(\varepsilon) \end{cases} \Longrightarrow \boldsymbol{a}^* = \frac{\sqrt{13}}{13} \begin{bmatrix} -3 \\ -2 \\ 11 \\ 8 \end{bmatrix},$$

2.5 $NC_2 = \emptyset$

2.6 the condition is satisfied

2.17 the condition is not satisfied

1.14 the condition is not satisfied

1.15 $NC_1 = \emptyset$

1.16 the condition is not satisfied

1.17 the condition is not satisfied

0.14 the condition is not satisfied

0.15 $NC_0 = \emptyset$

0.16 the unsatisfied condition

0.17 the condition is satisfied and $\boldsymbol{a}^*$ is the searched solution.

It has just been shown that the list order of T_p is crucial in Algorithm LS2S, cf. the examples of the obtained solutions I, II and III. The list order of Y_p in Algorithm SLS2S does not lead to multiple solutions—there is only one. The list order may effect the number of steps which are necessary to find a solution (if it exists). The fastest results are obtained when two points with the following properties are in the list Y_0 at the beginning:

- one point is originally from X_1 and the other from X_2,
- these points are the closest ones in the Euclidean metrics in the original space.

It is impossible to find the hyperplanes h_1 and h_2 with a larger value of ε than the distance between those two points.

In the case of the discussed sets, for the list (the points with the described properties are $\boldsymbol{x}_4$ and $\boldsymbol{x}_6$):

$$Y_0 = \{f^*(\boldsymbol{x}_4), f^*(\boldsymbol{x}_6), f^*(\boldsymbol{x}_5), f^*(\boldsymbol{x}_2), f^*(\boldsymbol{x}_3), f^*(\boldsymbol{x}_1), f^*(\boldsymbol{x}_7), f^*(\boldsymbol{x}_8)\}$$

executing Algorithm SLS2S is "much faster" (in the sense of the number of steps to be done) and is shown as follows:

0.1 $K = \emptyset$

0.2 the condition is satisfied and $K = \{\boldsymbol{y}_4\}$

0.3 the condition is satisfied and $K = K \cup \{\boldsymbol{y}_6\} = \{\boldsymbol{y}_4, \boldsymbol{y}_6\}$

0.4

$$\begin{cases} \langle \boldsymbol{a}^*, \boldsymbol{y}_4 \rangle = 0 \\ \langle \boldsymbol{a}^*, \boldsymbol{y}_6 \rangle = 0 \\ \alpha_1^2 + \alpha_2^2 = 1 \\ \max(\varepsilon) \end{cases} \Longrightarrow \boldsymbol{a}^* = \frac{\sqrt{10}}{10} \begin{bmatrix} -3 \\ -1 \\ 10 \\ 7 \end{bmatrix},$$

0.5 $NC_0 = \{\boldsymbol{y}_5\}$

0.6 the condition is not satisfied

0.7 the condition is not satisfied

0.8 $Y_1 = \{\boldsymbol{y}_4, \boldsymbol{y}_6, \boldsymbol{y}_2, \boldsymbol{y}_3, \boldsymbol{y}_1, \boldsymbol{y}_7 \boldsymbol{y}_8\}$

0.9 $\boldsymbol{b} = \boldsymbol{y}_5$

0.10 $c = 2$;

0.11 the condition is satisfied and $K_1^{(1)} = K_0^{(1)} = \emptyset$, $K_1^{(2)} = K_0^{(2)} \cup \{\boldsymbol{y}_5\} = \{\boldsymbol{y}_5\}$

0.12 the condition is not satisfied

0.13 execute the algorithm $(\boldsymbol{a}^*, \textit{flag}) \leftarrow \text{SLS2S}(Y_1, K_1^{(1)}, K_1^{(2)})$

1.1 $K = \emptyset$

1.2 the condition is satisfied and $K = \{\boldsymbol{y}_4\}$

1.3 the condition is not satisfied

1.4

$$\begin{cases} \langle \boldsymbol{a}^*, \boldsymbol{y}_4 \rangle = 0 \\ \langle \boldsymbol{a}^*, \boldsymbol{y}_5 \rangle = 0 \\ \alpha_1^2 + \alpha_2^2 = 1 \\ \max(\varepsilon) \end{cases} \Longrightarrow \boldsymbol{a}^* = \frac{\sqrt{17}}{17} \begin{bmatrix} -1 \\ -4 \\ 17 \\ 6 \end{bmatrix},$$

1.5 $NC_1 = \{\boldsymbol{y}_6, \boldsymbol{y}_2, \boldsymbol{y}_7, \boldsymbol{y}_8\}$

1.6 the condition is not satisfied

1.7 the condition is not satisfied

1.8 $Y_2 = \{\boldsymbol{y}_4, \boldsymbol{y}_3, \boldsymbol{y}_1\}$

1.9 $\boldsymbol{b} = \boldsymbol{y}_6$

1.10 $c = 2$;

1.11 the condition is satisfied and $K_2^{(1)} = K_1^{(1)} = \emptyset$, $K_2^{(2)} = K_1^{(2)} \cup \{\boldsymbol{y}_6\} = \{\boldsymbol{y}_5, \boldsymbol{y}_6\}$

1.12 the condition is not satisfied

1.13 execute the algorithm $(\boldsymbol{a}^*, \textit{flag}) \leftarrow \text{SLS2S}(Y_2, K_2^{(1)}, K_2^{(2)})$

2.1 $K = \emptyset$

2.2 the condition is satisfied and $K = \{\boldsymbol{y}_4\}$

2.3 the condition is not satisfied

2.4

$$\begin{cases} \langle \boldsymbol{a}^*, \boldsymbol{y}_4 \rangle = 0 \\ \langle \boldsymbol{a}^*, \boldsymbol{y}_5 \rangle = 0 \\ \langle \boldsymbol{a}^*, \boldsymbol{y}_6 \rangle = 0 \\ \alpha_1^2 + \alpha_2^2 = 1 \\ \max(\varepsilon) \end{cases} \Longrightarrow \boldsymbol{a}^* = \frac{\sqrt{13}}{13} \begin{bmatrix} -3 \\ -2 \\ 11 \\ 8 \end{bmatrix},$$

2.5 $NC_2 = \emptyset$

2.6 the condition is satisfied

2.17 the condition is not satisfied

1.14 the condition is not satisfied

1.15 $NC_1 = \emptyset$

1.16 the condition is not satisfied

1.17 the condition is not satisfied

0.14 the condition is not satisfied

0.15 $NC_0 = \emptyset$

0.16 the condition is not satisfied

0.17 the condition is satisfied and $\boldsymbol{a}^*$ is the searched vector.

3. Concluding Remarks

The presented algorithm SLS2S still has the theoretical properties of the LS2S algorithm (Jóźwik, 1975). Though the computation complexity is $\mathrm{O}(m^{n+1})$ ($m = \text{card}(X_1 \cup X_2)$) the maximum number of steps to be done is less than $\binom{m+n}{n+1}$.

Unlike LS2S, the presented algorithm uses information concerning the label of every point without a special effort. And the most important fact is that the result of executing the SLS2S algorithm does not depend on the sequence of the classified patterns. If sets are separable, the obtained solution is always the same, optimal and maximises the distance between the separated sets.

In forthcoming papers a mathematical proof of the correctness of the algorithm will be presented. The proof makes it possible to assure numerical stability of the implementations of the algorithm and its full optimization.

References

Cristianini N. and Shawe-Taylor J. (2000): *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. — Cambridge: Cambridge University Press.

Duda R.O., Hart P.E. and Stork D.G. (1995): *Pattern Classification*. — New York: Wiley.

Franc V. and Hlaváč V. (2003): *An iterative algorithm learning the maximal margin classifier*. — Pattern Recogn., Vol. 36, No. 9, pp. 1985–1996.

Jankowski N. (2003): *Ontogenic Neural Networks*. — Warsaw: EXIT, (in Polish).

Jóźwik A. (1975): *Method of investigaton of separability of two finite sets in n-dimensional space*. — Scientific Works of the Institute of Organization and Manegement, Series: Applied Cybernetics and Computer Science, Vol. 18, (in Polish).

Jóźwik A. (1983): *A recursive method for the investigation of the linear separability of two sets*. — Pattern Recogn., Vol. 16, No. 4, pp. 429–431.

Jóźwik A. (1998): *Algorithm of investigaton of separability of two sets, prospects of reusing this algorithm to construct the binary classifier*. — Proc. 6-th Conf. *Networks and Information Systems—Theory, Projects and Applications*, Łódź, Poland, pp. 311–316, (in Polish).

Kozinec B.N. (1973): *Recurrent algorithm separating convex hulls of two sets*, In: Learning Algorithms in Pattern Recognition (V.N. Vapnik, Ed.). — Moscow: Soviet Radio, pp. 43–50, (in Russian).

Mangasarian O.L. (2000): *Generalized Support Vector Machines, Advances in Large Margin classifiers*, pp. 135–146, MIT Press, available at `ftp://ftp.cs.wisc.edu/math-prog/tech-reports/98-14.ps`

Vapnik V.N. (2000): *The Nature of Statistical Learning Theory*. — New York: Springer.

Received: 15 April 2004
Revised: 9 July 2004
Re-revised: 11 January 2005