Workshop on the Kernel and Subspace Methods for Computer Vision adjoint to the International Conference on Artifical Neural Networks, Proceedings A. Leonardis and H. Bischof (Ed.), August 25, 2001, Technische Universitaet, Wien, Austria, pages 1–11.

A Simple Learning Algorithm for Maximal Margin Classifier

Vojtěch Franc and Václav Hlaváč

Czech Technical University, Faculty of Electrical Engineering
Department of Cybernetics, Center for Machine Perception
121 35 Praha 2, Karlovo náměstí 13, Czech Republic

Abstract. This paper describes a simple learning algorithm for maximal margin classifiers. The algorithm is based on the Schlesinger-Kozinec's algorithm (S-K-Algorithm) which finds maximal margin hyperplane with a given precision for separable data. We propose (i) a generalization of the S-K-Algorithm to the non-linear algorithm using kernel functions and (ii) a method which allows the use of the S-K-Algorithm for non-separable data. The proposed algorithm is simple to implement and, as the experiments showed, competitive to the state-of-the-art algortihms.

1 Introduction, related work and notation

Maximal margin classifiers find a decision strategy which separates the training data with the maximal margin. This leads to minimization of the structural risk and to good generalization capabilities of the classifier. This is justified by both experimental evidence and theoretical studies [11, 9, 10]. The Support Vector Machines (SVMs) [10] transform the learning process of the maximal margin classifier to the Quadratic Programming (QP). Although the QP task is well explored it requires complicated optimization algorithms and, moreover, processing of large amounts of data is computationally demanding.

We propose a simple, on-line learning algorithm which finds non-linear maximal margin classifier with the given precision. The approach is based on an *veps*-solution algorithm which yields linear decision rule which have margin different from the optimal one at most by a constant ε. The *veps*-solution algorithm was proposed by M.I. Schlesinger et al. [7] and its more recent analysis is available in the monograph [6]. This iterative algorithm uses an update rule that was proposed by B.N. Kozinec [4]. We will call the algorithm the Schlesinger-Kozinec algorithm and abbreviate it as S-K-Algorithm.

The advantages of the S-K-Algorithm are its simplicity for programming, fast convergence to the solution and its on-line processing of data. The disadvantages of the S-K-Algorithm are that it finds only linear classifiers and requires linearly separable data.

Our contribution lies in the generalization of the S-K-Algorithm to the non-linear one using kernel functions [1]. We also propose the generalization to the non-separable case where the generalized optimal hyperplane [10] with quadratic cost function is searched for. Experiments conducted have shown that the proposed algorithm is competitive to the SVMs while its programming is considerably simpler.

The rest of the paper is structured as follows. Section 2 defines the optimal and the ε-optimal hyperplane. Section 3 describes the original S-K-Algorithm for linear and separable cases. In Section 4, the kernel version of the S-K-Algorithm is proposed and Section 5 provides generalization to the non-separable case. Experiments are given in Section 6. Section 7 concludes the paper.

2 Optimal and ε-optimal separating hyperplane

The sets $X_1 = \{x_i : i \in I_1\}$ and $X_2 = \{x_i : i \in I_2\}$ denote training patterns for the first and the second class, where $I = I_1 \cup I_2$ are the sets of indices. The numbers $y_i = +1$, $i \in I_1$ and $y_i = -1$, $i \in I_2$ denote pattern labels.

The classes, i.e vector sets X_1 and X_2, are linearly separable if there exist a vector w and a threshold θ such that the set of inequalities

$$\langle w, x_i \rangle \geq \theta \, , \quad i \in I_1 \, , \qquad \langle w, x_i \rangle < \theta \, , \quad i \in I_2 \tag{1}$$

holds. The vector w and threshold θ determine a separating hyperplane $\langle w, x \rangle = \theta$. Let us introduce function

$$m(w, \theta) = \min_{i \in I} \left(\frac{y_i \cdot (\langle w, x_i \rangle - \theta)}{||w||} \right) \, .$$

Its value is the shortest distance from the hyperplane $\langle w, x \rangle = \theta$ to vector sets X_1 and X_2. If the inequalities (1) have a solution then there is an *optimal separating hyperplane* [10], which is the solution of

$$\{w^*, \theta^*\} = \operatorname*{argmax}_{w,\theta} m(w, \theta) \, . \tag{2}$$

The real positive number $m^* = m(w^*, \theta^*)$ is the maximal margin between the sets X_1 and X_2. Let us define so called *ε-optimal separating hyperplane* [6] now, which is determined by the vector w and θ satisfying the following condition

$$m^* - m(w, \theta) \leq \varepsilon \, , \tag{3}$$

where ε is an arbitrary positive constant. In other words, the margin defined by the (3) differs from the optimal margin by ε at most.

3 S-K-Algorithm

In this section, we will describe the S-K-Algorithm, which finds the ε-optimal separating hyperplane (3). We will introduce the S-K-Algorithm without detailed analysis and proof of convergence, which can be found in [6].

The algorithm is based on the idea that two point sets X_1, X_2 can be optimally separated by the hyperplane which is perpendicular to the vector $w_1^* - w_2^*$ and passes through the center of the abscissa joining the points w_1^* and w_2^*. The points w_1^* and w_2^* belong into the convex hulls $\overline{X}1$ and $\overline{X}2$, and they are the points which determine the shortest distance between these convex hulls, i.e.

$$\{w_1^*, w_2^*\} = \operatorname*{argmin}_{w_1 \in \overline{X}1, w_2 \in \overline{X}2} ||w_1 - w_2|| \,. \tag{4}$$

The vector $w^* = w_1^* - w_2^*$ and the threshold $\theta = \frac{1}{2}(||w_1^*||^2 - ||w_2^*||^2)$ determine the hyperplane which coincide to the optimal hyperplane defined by (2).

The idea mentioned above is implemented into the algorithm as follows. At the beginning, w_1, w_2 is set to an arbitrary point from the set X_1, X_2, respectively. Then the point w_2 is fixed and w_1 is moved toward w_2 as much as possible. Similarly, the point w_1 found is fixed and the point w_2 is moved toward w_1. This process is repeated until the ε-optimality condition is satisfied. The adaptation rule used in the algorithm ensures that the vector w_1, w_2 does not leave the convex hull $\overline{X}1$,$\overline{X}2$ respectively. A modification of (3) is used as the stopping condition. The the first term m^* is unknown since it is a width of the optimal margin. The value m^* can be replaced by its upper bound. It is obvious that the norm of the vector $w = w_1 - w_2$, $w_1 \in \overline{X}1$, $w_2 \in \overline{X}2$ is the closest upper bound we have. The algorithm outlined is described below.

1. The vector w_1 is set to an arbitrary vector from the set X_1 and w_2 to an arbitrary vector from the set X_2.
2. A vector $x_t \in X_1$ satisfying condition

$$\left\langle x_t - w_2, \frac{w_1 - w_2}{||w_1 - w_2||} \right\rangle \leq ||w_1 - w_2|| - \frac{\varepsilon}{2} \tag{5}$$

is searched for. If such vector x_t is found then go to Step 3. Otherwise a vector $x_t \in X_2$ which satisfies condition

$$\left\langle x_2 - w_1, \frac{w_2 - w_1}{||w_1 - w_2||} \right\rangle \leq ||w_1 - w_2|| - \frac{\varepsilon}{2} \,. \tag{6}$$

is searched for. If such vector x_t is found then go to Step 4. If neither (5) nor (6) is satisfied for any vector then the vectors w_1 and w_2 define the ε-optimal hyperplane $\langle w, x \rangle = \theta$, where $w = w_1 - w_2$ and $\theta = \frac{1}{2}(||w_1||^2 - ||w_2||^2)$. The margin of found hyperplane is equal to $||w||$.
3. If the point x_t which satisfies the condition (5) exists then $w_2^{new} = w_2$ and the new vector w_1^{new} is computed as

$$w_1^{new} = (1 - k) \cdot w_1 + k \cdot x_t \,,$$

where

$$k = \min\left(1, \frac{\langle w_1 - w_2, w_1 - x_t\rangle}{||w_1 - x_t||^2}\right) .$$

The algorithm continues with Step 2.

4. If the point x_t which satisfies the condition (6) exists then $w_1^{new} = w_1$ and the new vector w_2^{new} is computed as

$$w_2^{new} = (1-k)\cdot w_2 + k \cdot x_t \ ,$$

where

$$k = \min\left(1, \frac{\langle w_2 - w_1, w_2 - x_t\rangle}{||w_2 - x_t||^2}\right) .$$

The algorithm continues to Step 2.

This algorithm builds a sequence of vectors $w = w_1 - w_2$, which converges to the optimal solution w^* when the input data are linearly separable. If the data are not linearly separable then the algorithm converges to zero vector. In each algorithm step, the norm $||w||$ decreases to $||w^{new}|| = K \cdot ||w||$, $0 < K < 1$, which means that for $\varepsilon > 0$ the algorithm stops after finite number steps. The proof as well as an estimate of K is given in [6].

Let us note, that a speed of the algorithm convergence can be estimated from the upper and lower bound of the optimal solution m^*. The upper bound is the norm of vector $||w||$ and the lower bound is the term in the left side of condition (5) and (6), respectively. We will show a plot of these bounds in experiments Section 6.

4 Kernel S-K-algorithm

This section describes the way how to generalize S-K-Algorithm to learn non-linear discriminant functions. The underlying idea is to express the S-K-Algorithm in terms of dot products and use the kernel trick [1–3].

Let us mention the idea of the kernel trick at the beginning. The input data from the original space $\mathcal{X}$ are non-linearly mapped into a new high (possible infinite) dimensional space $\mathcal{Y}$. The mapping is defined by a function $\phi: \mathcal{X} \to \mathcal{Y}$. Let $k: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a function, a value of which corresponds to the dot product of the non-linearly mapped vectors $\phi(x_i)$ and $\phi(x_j)$, i.e.

$$k(x_i, x_j) = \langle \phi(x_i), \phi(x_j)\rangle, \qquad \forall x_i, x_j \in \mathcal{X} .$$

It implies that any linear algorithm in which data appear in the term of the dot product can be non-linearized by substituting kernel functions for these dot products. The rest of this section shows how to rewrite the S-K-Algorithm to the form of dot products.

The idea is to express the iterated vectors w_1 and w_2 as the convex combination of point sets X_1 and X_2. We introduce multipliers $\alpha_i, i \in I$ which corresponds to the training data $x_i, i \in I$. The relation of the multipliers α_i and the vectors w_1 and w_2 is expressed as

$$w_1 = \sum_{i \in I_1} \alpha_i \cdot x_i \ , \quad w_2 = \sum_{i \in I_2} \alpha_i \cdot x_i \ , \quad \sum_{i \in I_1} \alpha_i = 1 \ , \quad \sum_{i \in I_2} \alpha_i = 1 \ . \tag{7}$$

We will show that the adaptation step of the S-K-Algorithm can be carried out as the change of the coefficient vectors α_1, α_2. Let us suppose that the algorithm arrived to Step 3 and the coefficient k and vector $x_t \in X_1$ have been already computed. The adaptation of the vector w_1 is computed as

$$w_1^{new} = (1-k) \cdot w_1 + k \cdot x_t \ . \tag{8}$$

Substituting (7) to (8) we get

$$\sum_{i \in I_1} \alpha_i^{new} \cdot x_i = (1-k) \cdot \sum_{i \in I_1} \alpha_i \cdot x_i + k \cdot x_t = \sum_{i \in I_1} [(1-k) + k \cdot \sigma_{it}] \cdot \alpha_i \cdot x_i \ ,$$

where we used the Kronecker delta $\sigma_{it} = 0$ for $i \neq t$ and $\sigma_{it} = 1$ for $i = t$. It means that the adaptation step for $\alpha_i, i \in I_1$ can be carried out as

$$\alpha_i^{new} = \begin{cases} (1-k) \cdot \alpha_i, & \text{for} \quad i = t, \\ (1-k) \cdot \alpha_i + k, & \text{otherwise} \end{cases}$$

and similarly for the case when $\alpha_i, i \in I_2$ are adapted. Also the rest of the algorithm, i.e. equations in Step 2 and computation of the number k in Step 3, 4 can be expressed using the multipliers α_i. It turns out that after this expression the data appear in terms of the dot product only and the kernel functions can be used.

We will derive this expression only for the condition (5) which is evaluated in Step 2. The other relations we will introduce without derivation as they are similar. The evaluation of the condition (5) requires values of the terms $\langle x_t - w_2, w_1 - w_2 \rangle$ and $||w_1 - w_2||^2$. Substituting (7) to the first term yields

$$\begin{aligned} &\langle x_t - w_2, w_1 - w_2 \rangle = \\ &= \langle x_t, w_1 \rangle - \langle w_1, w_2 \rangle - \langle x_t, w_2 \rangle + \langle w_2, w_2 \rangle \\ &= \sum_{i \in I_1} \alpha_i \langle x_t, x_i \rangle - \sum_{i \in I_1} \sum_{j \in I_2} \alpha_i \alpha_2 \langle x_i, x_j \rangle - \sum_{i \in I_2} \alpha_i \langle x_t, x_i \rangle + \sum_{i \in I_2} \sum_{j \in I_2} \alpha_i \alpha_2 \langle x_i, x_j \rangle \ . \end{aligned}$$

Similarly for the second term we get

$$\begin{aligned} &||w_1 - w_2||^2 = \\ &= \sum_{i \in I_1} \sum_{j \in I_1} \alpha_i \alpha_j \langle x_i, x_j \rangle - 2 \sum_{i \in I_1} \sum_{j \in I_2} \alpha_i \alpha_j \langle x_i, x_j \rangle + \sum_{i \in I_2} \sum_{j \in I_2} \alpha_i \alpha_j \langle x_i, x_j \rangle \\ &= \sum_{i \in I} \sum_{j \in I} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \ . \end{aligned}$$

It is obvious that the data appear in terms of the dot product only. It allows us to substitute the kernel functions $k(x_i, x_j)$ for the dot products $\langle x_i, x_j \rangle$. The result of the S-K-Algorithm, after the transformation mentioned above, are the multipliers α_i and the threshold θ. The recognition of unknown pattern x is carried out according to the sign of the function

$$f(x) = \langle \phi(w), \phi(x) \rangle - \theta = \sum_{i \in I_1} \alpha_i k(x_i, x) - \sum_{i \in I_2} \alpha_i k(x_i, x) - \theta = \sum_{i \in I} \alpha_i y_i k(x_i, x) - \theta .$$

The Kernel version of the S-K-Algorithm is introduced bellow.

1. *Initialization.* The coefficients $\alpha_i, i \in I$ are set to

$$\alpha_i = \begin{cases} 1, & \text{if} \quad i = i_1 \quad \text{or} \quad i = i_2, \\ 0, & \text{otherwise}, \end{cases}$$

   where i_1 is an arbitrary index from I_1 and i_2 is an arbitrary index from I_2.
2. *Auxiliary variables computation.* The following numbers are computed

$$a = \sum_{i \in I_1} \sum_{j \in I_1} \alpha_i \alpha_j k(x_i, x_j), \qquad b = \sum_{i \in I_2} \sum_{j \in I_2} \alpha_i \alpha_j k(x_i, x_j),$$

$$c = \sum_{i \in I_1} \sum_{j \in I_2} \alpha_i \alpha_j k(x_i, x_j),$$

$$d_i = \sum_{j \in I_1} \alpha_j k(x_i, x_j), \qquad e_i = d_i - \sum_{j \in I_2} \alpha_j k(x_i, x_j), \quad \text{for} \quad i \in I_1 \ ,$$
$$f_i = \sum_{j \in I_2} \alpha_j k(x_i, x_j), \qquad g_i = f_i - \sum_{j \in I_1} \alpha_j k(x_i, x_j), \quad \text{for} \quad i \in I_2 \ .$$

3. *Multipliers $\alpha_i, i \in I_1$ update.* The index $t = \operatorname*{argmin}_{i \in I_1} e_i$ is found. If the condition

$$\frac{e_t + b - c}{\sqrt{a - 2c + b}} \leq \sqrt{a - 2c + b} - \frac{\varepsilon}{2} \tag{9}$$

   holds then $\alpha_i^{new} = \alpha_i, i \in I_2$ and update $\alpha_i^{new}, i \in I_1$ as

$$\alpha_i^{new} = \begin{cases} \alpha_i(1-k) + k, & \text{for} \quad i = t, \\ \alpha_i(1-k), & \text{otherwise}, \end{cases}$$

   where

$$k = \frac{a - e_t - c}{\sqrt{a - 2d_t + k(x_t, x_t)}}$$

   and go to Step 2. If the condition (9) does not hold then go to Step 4.
4. *Multipliers $\alpha_i, i \in I_2$ update.* The index $t = \operatorname*{argmin}_{i \in I_2} g_i$ is found. If the condition

$$\frac{g_t + a - c}{\sqrt{a - 2c + b}} \leq \sqrt{a - 2c + b} - \frac{\varepsilon}{2} \tag{10}$$

holds then $\alpha_i^{new} = \alpha_i, i \in I_1$ and update $\alpha_i^{new}, i \in I_2$ as

$$\alpha_i^{new} = \begin{cases} \alpha_i(1-k)+k, & \text{for} \quad i = t, \\ \alpha_i(1-k), & \text{otherwise,} \end{cases}$$

where

$$k = \frac{b - g_t - c}{\sqrt{b - 2f_t + k(x_t, x_t)}}$$

and go to Step 2. If the condition (10) does not hold then go to Step 5.

5. *Threshold computation.* The threshold is computed as $\theta = \frac{1}{2}(a-b)$ and the margin is $m = \frac{1}{2}\sqrt{a - 2c + b}$.

If the input data are linearly separable in the non-linear feature space then the algorithm exits after finite number of iterations. For non-separable data the algorithm converges to the zero vector.

5 Non-separable case

In this section, we will describe the way how to cope with the non-separable case. To this end, we will first introduce another formulation of the problem of finding optimal hyperplane which is

$$\{w, \theta\} = \operatorname*{argmin}_{w,b} \frac{1}{2}||w||^2 , \tag{11}$$

subject to

$$y_i(\langle w, x_i \rangle - \theta) \geq 1 , \quad i \in I .$$

This problem formulation is used in the Support Vector Machines and the found hyperplane $\langle w, x \rangle = \theta$ is called to be in the canonical form since the $\min\limits_{i \in I} y_i(\langle w, x_i \rangle - \theta) = 1$. The hyperplane $\langle w', x \rangle = \theta'$ defined by (4) which the S-K-Algorithm returns coincide to the hyperplane $\langle w, x \rangle = \theta$ but it has not the canonical form since $\min\limits_{i \in I} y_i(\langle w', x_i \rangle - \theta') = ||w||^2/2$. It is obvious that

$$w = s \cdot w' , \qquad \theta = s \cdot \theta' ,$$

where $s \in \mathbb{R}$ and can be computed as

$$s = \frac{2}{||w||^2} .$$

The normalization of the solution returned by the Kernel S-K-Algorithm to the canonical form lies in a change of the Step 5 to

$$\theta' = \frac{(a-b)}{(a-2c+b)} , \qquad \alpha'_i = \frac{2\alpha_i}{(a-2c+b)} , \quad i \in I . \tag{12}$$

Then the function $f(x) = \sum_{i \in I} \alpha'_i k(x_i, x) + \theta'$ defines canonical hyperplane in the non-linear feature space induced by the given kernel. Let us note, that the multipliers α'_i computed by the Kernel S-K-Algorithm coincide to the Lagrangian multipliers of the dual form of (11) (see [10]), which SVMs compute.

When the data are not-linearly separable then there is no hyperplane without classification error on the training data. The *generalized optimal hyperplane* [10] which is defined as a solution of

$$\{w, \theta, \xi_i, i \in I\} = \operatorname*{argmin}_{w,b} \frac{1}{2} ||w||^2 + C \sum_{i \in I} (\xi_i)^p , \tag{13}$$

subject to

$$y_i(\langle w, x_i \rangle - \theta) \geq 1 - \xi_i , \quad i \in I, \qquad \xi_i \geq 0 ,$$

can be found instead. The prescribed constant $C \in \mathbb{R}$ determines trade-off between a width of the margin and the training error. The training error is approximated by the cost function $\sum_{i \in I} (\xi_i)^p$. The variables ξ_i relax the set of inequalities and they are called slack variables. The SVMs use a linear cost function $p = 1$ for which the problem leads to the quadratic programming. Further on, we will be interested in the quadratic cost function $p = 2$.

In [8], a transformation is proposed, which modifies the task (13) with quadratic cost function to the task (11) where no cost function is presented. To this end, the input n-dimensional space $\mathcal{X}$ is transformed to a new $n+l$-dimensional space $\mathcal{X}'$, where the non-separable data become separable. The input vector $x_i \in \mathcal{X}$ is mapped to a new $x'_i \in \mathcal{X}'$ as

$$x_i^{'j} = \begin{cases} x_i^j , & \text{for} \quad j = 1, ..., n , \\ \frac{y_i}{\sqrt{2C}} , & \text{for} \quad j = n + i , \\ 0 , & \text{otherwise} , \end{cases} \tag{14}$$

where the upper index denotes coordinate number. Next, we introduce a new normal vector

$$w^{'j} = \begin{cases} w^j , & \text{for} \quad j = 1, ..., n , \\ \sqrt{2C} \xi_{j-n} , & \text{otherwise} . \end{cases} \tag{15}$$

If we substitute w' and x'_i to (11) then it turns to the problem (13) with quadratic cost function $p = 2$.

To apply the mentioned transformation to the Kernel S-K-Algorithm, we must only change the kernel function $k(x_i, x_j)$ to a new $k'(x_i, x_j)$, which is defined as

$$k'(x_i, x_j) = \begin{cases} k(x_i, x_j) , & \text{for} \quad j \neq i , \\ k(x_i, x_j) + \frac{1}{2C} , & \text{for} \quad i = j . \end{cases} \tag{16}$$

At the end, we will summarize mentioned extension of the Kernel S-K-Algorithm. The Step 5 is replaced by the relations (12) and the kernel function $k(x_i, x_j)$ is replaced by the $k'(x_i, x_j)$ defined by (16). After this modification the solution returned by the S-K-Algorithm coincides to the solution of the task (13) with cost quadratic cost function $p = 2$.

6 Experiments

We tested the proposed Kernel S-K-Algorithm on the synthetic Ripley's [5] data set. As a reference algorithm we used SVMs with quadratic cost function (see Section 5). For the quadratic programming task of SVMs, we used Matlab Optimization Toolbox.

The Ripley's data set constitutes a well-known synthetic problem of small size and in a two-dimensional space. The set consists of 250 training and 1000 testing patterns belonging into two classes which are heavily overlapped. The parameters of the Kernel S-K-Algorithm are: (i) kernel function, (ii) trade-off constant C and (iii) precision constant set to $\varepsilon = 0.01$. The SVMs do not have the precision parameter which is implicitly hidden in the used quadratic programming algorithm. We learned classifiers with (i) linear kernel, (ii) RBF kernel with $\sigma = 1$, (iii) polynomial kernel of degree $d = 2$. We used trade-off constant ranging from $C = \{1, 10, 100, 1000, 10000\}$. The best obtained classifier, having the smallest classification error on test data, was enlisted to the comparative Table 1. We also measured (i) training classification error, (ii) number of floating point operations used for training, (iii) and the number of patterns determining the classifier, i.e. Support Vectors.

Table 1. Comparison between Kernel S-K-Algorithm (KSK) and Support Vector Machines (SVM) on Ripley data set. Both algorithms use quadratic cost function.

	Class. error (test) [%]	Class. error (train) [%]	Floating point operations	Number of Support Vectors
SVM (linear, C=1)	11.30	14.80	$4590 \cdot 10^6$	165
KSK (linear, C=10)	11.40	14.40	$385 \cdot 10^6$	149
SVM (RBF $\sigma = 1$, C=10000)	9.40	14.80	$5995 \cdot 10^6$	120
KSK (RBF $\sigma = 1$, C=10)	9.00	13.20	$242 \cdot 10^6$	145
SVM (poly d=2, C=10)	9.90	14.80	$5180 \cdot 10^6$	152
KSK (poly d=2, C=10)	10.50	14.00	$1081 \cdot 10^6$	156

To demonstrate an effect of the constant ε in the Kernel S-K-Algorithm we visualized the upper and lower bounds on the optimal solution. By the optimal solution, we mean the maximal margin in the high dimensional feature space, where the data are non-linearly mapped and are separable. The bounds are evaluated in each algorithm step (see Section 3) and determine its stop condition. The algorithm exits when the difference between the upper and the lower bound, denoted as ε-precision, is less than given ε. Figure 1 shows the plot of these bounds with respect to iteration number. The plots correspond to the classifiers listed in Table 1.

An implementation of the Kernel S-K-Algorithm can be found in the Statistical Pattern Recognition toolbox for Matlab. The toolbox is freely downloadable from `http://cmp.felk.cvut.cz/ xfrancv/stprtool`.

7 Conclusion

We have proposed the method based on the S-K-Algorithm which learns linear maximal margin classifier with prescribed precision. The advantage of the S-K-Algorithm is its simple programming, on-line processing of data and fast convergence to the solution. A restricting property of the S-K-Algorithm is the requirement on separable training sets. Our contribution lies in generalization of the S-K-Algorithm to the non-linear case by the use of kernel functions. An extension to the non-separable case is proposed as well. The proposed algorithm finds the generalized optimal margin with quadratic cost function. The algorithm is simple to implement and does not require any additional optimization packages. The experiments conducted show that is competitive to the SVMs.

Acknowledgement

This research was supported by the Grant Agency of the Czech Republic under the grant GACR 102/00/1679 Geometry and appearance of a 3D scene from a large collection of images.

References

1. A. Aizerman, M., M. Braverman, E., and L. I. Rozoner. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.
2. E. Boser, B., M. Guyon, I., and N. Vapnik, V. A training algorithm for optimal margin classifiers. In *Proc. 5th Annual Workshop on Comput. Learning Theory*, pages 144–152. ACM Press, New York, NY, 1992.
3. Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20:273, 1995.
4. B.N. Kozinec. Rekurentnyj algoritm razdelenia vypuklych obolochek dvuch mnozhestv, in Russian (Recurrent algorithm separating convex hulls of two sets). In V.N. Vapnik, editor, *Algoritmy obuchenia raspoznavania (Learning algorithms in pattern recognition)*, pages 43–50. Sovetskoje radio, Moskva, 1973.
5. B.D. Riplay. Neural networks and related methods for classification (with discusion). *J. Royal Statistical Soc. Series B*, 56:409–456, 1994.
6. M. I. Schlesinger and V. Hlaváč. *Deset přednášek z teorie statistického a strukturního rozpoznávání, in Czech (Ten lectures on statistical and structural pattern recognition)*. Czech Technical University Publishing House, Praha, Czech Republic, 1999. English version is supposed to be published by Kluwer Academic Publishers in 2001.

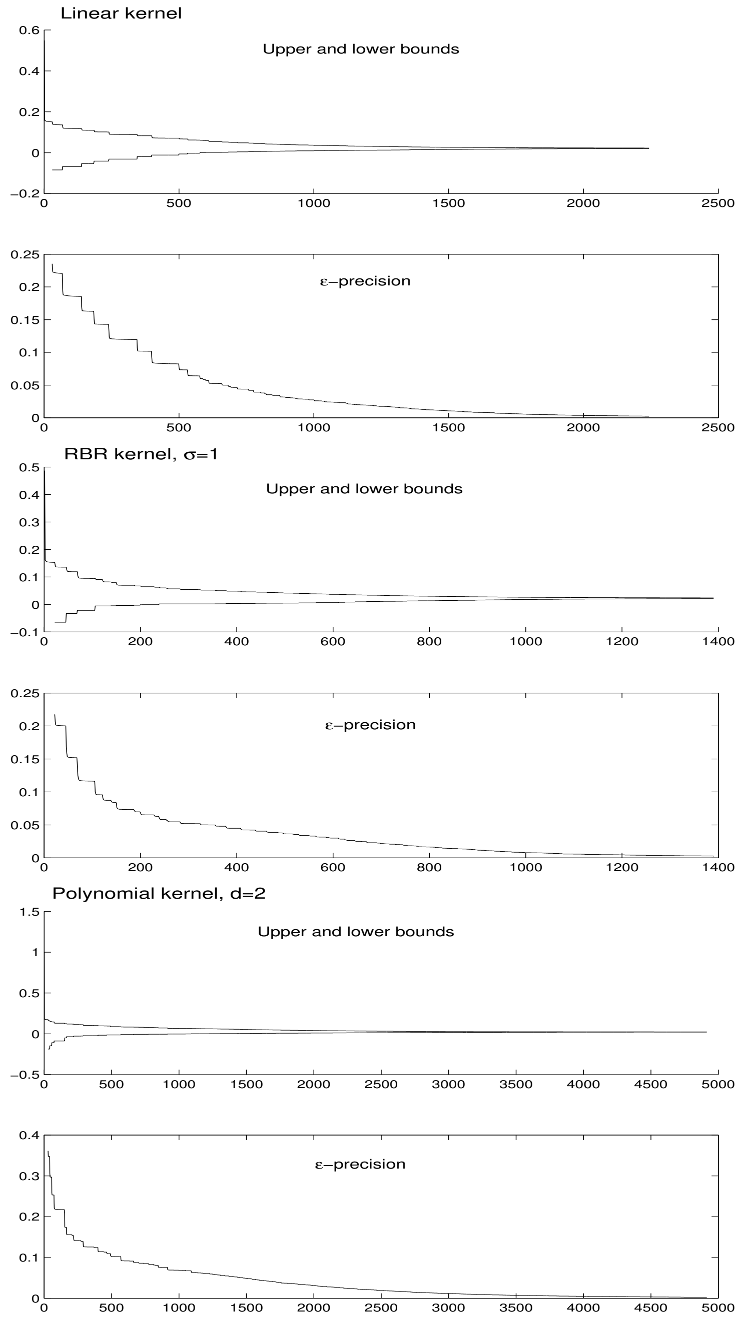

Fig. 1. Convergence of the Kernel S-K-Algorithm to the solution for linear, RBF and polynomial kernels (see Table 1).

7. M.I. Schlesinger, V.G. Kalmykov, and A.A. Suchorukov. Sravnitelnyj analiz algoritmov sinteza linejnogo reshajushchego pravila dlja proverki slozhnych gipotez, in Russian (Comparative analysis of algorithms synthesising linear decision rule for analysis of complex hypotheses). *Automatika*, (1):3–9, 1981.
8. J. Shawe-Taylor and N. Cristiani. Robust bounds on generalization from the margin distribution. Technical report, NeuroCOLT2, October 1998.
9. V. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer Verlag, 1982.
10. V. Vapnik. *The nature of statistical learning theory*. Springer Verlag, 1995.
11. V. Vapnik and Ya. Chervonenkis, A. The theory of pattern recognition. *Nauka, Moscow*, 1974.