

# A Sparsity Driven Kernel Machine Based on Minimizing a Generalization Error Bound

Dori Peleg <sup>a,\*</sup>, Ron Meir <sup>a</sup>

<sup>a</sup>*Technion- Israel Institute of Technology, Department of Electrical Engineering,  
Technion, Haifa 32000, Israel*

---

## Abstract

A new sparsity driven kernel classifier is presented based on the minimization of a recently derived data-dependent generalization error bound. The objective function consists of the usual hinge loss function penalizing training errors and a concave penalty function of the expansion coefficients. The problem of minimizing the non-convex bound is addressed by a successive linearization approach, whereby the problem is transformed into a sequence of linear programs. The algorithm produced comparable error rates to the standard Support Vector Machine but significantly reduced the number of support vectors and the concomitant classification time.

*Key words:* Sparsity, Dimensionality Reduction, Classification, Generalization Error Bounds, Statistical Learning Theory.

---

## 1 Introduction

This paper presents a novel Sparsity driven Kernel Machine (SKM) algorithm based on the minimization of a state-of-the-art generalization error bound. The primary goal of the classifier is to learn a decision rule from a training set of pairs  $S_n = \{x^{(i)}, y^{(i)}\}_{i=1}^n$ , where  $x^{(i)} \in \mathbb{R}^d$  are input patterns and  $y^{(i)} \in \{-1, 1\}$  are the corresponding labels. The objective of a classification algorithm is to find a separating function  $f(\cdot)$ , based on the training set, which will generalize well, i.e. classify new patterns with as few errors as possible. We focus on kernel

---

\* Corresponding author.

*Email addresses:* `peleg.dori@gmail.com` (Dori Peleg ),  
`rmeir@ee.technion.ac.il` (Ron Meir).

classifiers from the class

$$\mathcal{F} = \left\{ f : f(x) = \sum_{i=1}^n \alpha_i y^{(i)} K(x^{(i)}, x) + b, \sum_{i=1}^n \alpha_i = 1, \alpha_1, \dots, \alpha_n \geq 0, b \in \mathbb{R} \right\}, \quad (1)$$

where  $K(\cdot, \cdot) : \mathcal{X} \times \mathcal{X} \mapsto [0, 1]$  is a kernel [1]. Points for which  $\alpha_i > 0$  will be referred to as support vectors (SV). As a secondary objective the classifier should be as sparse as possible, i.e. utilize as few SVs as possible. The motivation behind the secondary objective is twofold:

- **Theoretical:** Several generalization error bounds, such as [2–4] indicate that the number of SVs is related to the complexity term in the bound. Hence sparse classifiers are expected to possess better generalization qualities. We will base the SKM algorithm on the recent bound presented in [5].
- **Practical:** The computational load of using a classifier to classify a point is proportional to the number of SVs. Therefore SV sparsity is extremely important for large datasets.

Previous work related to sparse kernel algorithms include the use of the  $l_1$  norm of  $\alpha$ , [6,7] and an approximation of the ‘pound’ function in [8] which is a linear combination of the  $l_1$  norm and the step function that measures both the magnitude and the presence of an error. Additional classifiers emphasizing sparse solutions are [9–11] and the Relevance vector Machine [12]. However, the motivation of [6,7,9–12] is not based on minimizing a statistical estimate of the generalization error, and is thus hard to justify theoretically. In [8] an upper bound is introduced, however the objective function is not an upper bound.

In order to reduce the number of SVs one must minimize the number of non-zero coefficients in the vector  $\alpha$  (the so called ‘ $l_0$  norm’ of  $\alpha$ ). Minimizing the ‘ $l_0$  norm’ in a straightforward fashion inevitably leads to the introduction of binary variables and a mixed binary optimization problem. Such problems are NP-hard and are expected to scale poorly with the number of training patterns  $n$ . Typically the zero norm is heuristically replaced with some sort of concave function approximation [13].

The significance of the upper bound in [5], upon which we base our algorithm, is twofold. First, the bound is not expressed as a function of the cardinality of  $\alpha$ , but rather consists of concave expressions which ‘encourage’ sparsity. Secondly, the expressions are *data dependent*. In order to find a local minimum of the objective function we will use the standard technique of sequential linearization [13], which requires the solution of a series of Linear Programs (LP). Current LP solvers are able to solve large scale problems with many thousands of variables and constraints very efficiently (e.g. [14]).

The SKM algorithm produced error rates comparable to the standard Support Vector Machine (SVM) algorithm but with the major advantage of producing significantly sparser classifiers, thus directly reducing the computational load of the final classifier.

The outline of this paper is as follows. In Section 2 the generalization error bound is formulated and the SKM algorithm is outlined. Section 2.2 lists related versions of the SVM algorithm and makes a theoretical comparison to the SKM algorithm. Section 3 provides a numerical comparison on several synthetic and real-world datasets. Finally Section 4 is a summary discussion.

## 2 The SKM algorithm

The generalization bound underlying the SKM algorithm is based on Theorem 3 from [5]. A significant aspect of the bound introduced in [5] is that the family of classifiers  $\mathcal{F}$  is *data dependent*. This distinction is indispensable for a kernel classifier (1). For a real-valued classifier  $f$ ,  $\mathbf{P}(yf(x) \leq 0)$  denotes the probability of error, while the empirical margin error is  $\mathbf{P}_n(yf(x) \leq \delta) = (1/n) \sum_{i=1}^n I[y_i f(x_i) \leq \delta]$ , where  $I[\cdot]$  is the indicator function.

**Theorem 2.1 (Adapted from [5])** *For any  $t > 0$  with probability at least  $1 - e^{-t}$ , for any  $n$  pairs  $(x^{(1)}, y^{(1)}), \dots, (x^{(n)}, y^{(n)})$  drawn independently at random with respect to a distribution  $\mathbb{P}$ , for all  $f \in \mathcal{F}$ ,*

$$\mathbf{P}(yf(x) \leq 0) \leq \inf_{\delta \in (0,1]} \left\{ (1 + \epsilon) \mathbf{P}_n(yf(x) \leq 2\delta) + \left(2 + \frac{1}{\epsilon} + \epsilon\right) U_\delta \right\} + \frac{1}{n}(\epsilon + 2) \quad (2)$$

where  $U_\delta = \frac{1}{n} \left( t + \ln \frac{4}{\delta} + \left( \ln 4n + \ln \left( \frac{8}{\delta^2} \ln n \right) \right) e_n(f, \delta) + \ln(8n + 4) \right)$ ,  $e_n(f, \delta) = \sum_{i=1}^n h(\alpha_i, \delta)$  and  $h(\alpha_i, \delta) = 1 - (1 - \alpha_i)^{\frac{8 \ln n}{\delta^2}}$ .

The bound (2) follows directly from Theorem 3 in [5] by using the inequality  $2\sqrt{uv} \leq u/\epsilon + \epsilon v$  which holds for any non-negative  $u, v$  and positive  $\epsilon$ .

The bound (2) consists of a penalty on the training error (first term) and a complexity penalty (second term). The variable  $\delta$  is the margin. Note that the training error is a monotonically nondecreasing function of  $\delta$  while the complexity term is a monotonically nonincreasing function of  $\delta$ . The intuition is that as the margin increases, the penalty on the training error should increase since a larger separation between the classes is required. On the other hand, as the margin requirement increases, the solution is more robust and therefore the complexity term penalizes smaller values of  $\delta$ .

The complexity term depends on  $\alpha$  through the functions  $h(\alpha_i, \delta)$ . These functions are concave penalty function of the components of  $\alpha$  (see Figure 1) and consequently induce sparsity of  $\alpha$ . Consider the two extreme cases. If one component of  $\alpha$  is one and the rest are zeros, then  $e_n(f, \delta) = 1$  and this is the global minimum under the constraints  $\sum_{i=1}^n \alpha_i = 1, \alpha_i \geq 0$ . The other extreme is the uniform case where all the components of  $\alpha$  are equal to  $\frac{1}{n}$ . In this case  $e_n(f, \delta) = n - \sum_{i=1}^n \left(1 - \frac{1}{n}\right)^a$ , where  $a = \frac{8 \ln n}{\delta^2}$  and  $a > 1$  given that  $n \geq 2$ . For example, if  $n = 1000$  and  $\delta = 0.5$ , then in this case  $e_n(f, \delta) = 198.41$ .

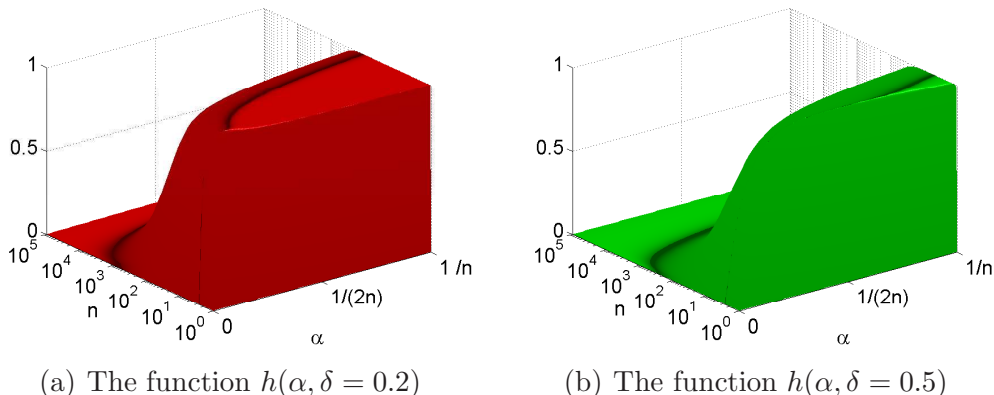


Fig. 1. An illustration of the dependance of the function  $h(\alpha, \delta)$  on  $n$  for two values of  $\delta$ . The  $\alpha$  axis is normalized to the region  $[0, \frac{1}{n}]$  for every  $n$ .

The dependence of the function  $h(\alpha_i, \delta)$  on the number of training points is more complex since the constraint  $\sum_{i=1}^n \alpha_i = 1$  must be satisfied. Figure 1 depicts the dependance of  $h(\alpha_i, \delta)$  on the number of training patterns. Note that as  $\delta$  increases the function  $h(\alpha_i, \delta)$  becomes less concave.

Next, the classifier which minimizes the bound (2) is derived from solving the following optimization problem.

$$\begin{aligned}
 & \text{minimize} \quad \mathbf{1}^T \xi + C \left( \ln 4n + \ln \left( \frac{8}{\delta^2} \ln n \right) \right) \sum_{i=1}^n h(\alpha_i, \delta) + C \ln \frac{4}{\delta} \\
 & \text{subject to} \quad Y (KY\alpha + b\mathbf{1}) \geq (2\delta)\mathbf{1} - \xi \\
 & \quad \quad \quad \mathbf{1}^T \alpha = 1 \\
 & \quad \quad \quad 0 < \delta \leq 1 \\
 & \quad \quad \quad \xi, \alpha \geq 0,
 \end{aligned} \tag{3}$$

with the variables  $\alpha, \xi \in \mathbb{R}^n, b, \delta \in \mathbb{R}$  and where  $\mathbf{1}$  is a vector of ones,  $K$  is the kernel matrix satisfying  $K_{ij} = K(x^{(i)}, x^{(j)})$ , and  $Y = \text{diag}(y^{(1)}, \dots, y^{(n)})$ .

In order to obtain a tractable optimization problem, we first replaced the 0 - 1 margin loss for the training error with the hinge loss  $\phi : \mathbb{R} \rightarrow \mathbb{R}_+$

which is defined as  $\phi(z) = 1 - z$  if  $z \leq 1$ , and zero otherwise (note that  $I(yf(x) \leq 0) \leq \phi(yf(x))$ ). Secondly, the third term in (2) was discarded because it is a constant with regard to the optimization problem. Thirdly, the terms  $\frac{t}{n}$  and  $\frac{\ln(8n+4)}{n}$  in the expression  $U_\delta$  were also discarded because they are constants. Finally, we divided the objective function by the positive constant  $1 + \epsilon$  and defined the constant  $C = (1 + \epsilon)^{-1} (2 + 1/\epsilon + \epsilon)$ .

Problem (3) is a *non-convex* optimization problem due to the second expression in the objective function which is not a convex function of  $\delta$  or  $\alpha$ . It is well known that in general it is very difficult to find the global solution of non-convex optimization problems [15]. In the following subsection we propose an approach which finds a local minimum using only a few LPs.

### 2.1 Alternating approach

We propose the following alternating minimization approach to finding a local minimum of Problem (3). First, we assume that  $\delta$  is known, and optimize over  $\{\alpha, b, \xi\}$  denoting the resulting objective function by  $f_0(\delta)$  as described in Table 1. Since  $\delta$  is a scalar, we used the ‘Golden section’ line-search algorithm [16, p.726–728] to find the optimal  $\delta$  with respect to the function  $f_0(\delta)$  on the interval  $(0, 1]$ .

However, even when  $\delta$  is given, the remaining optimization problem is a concave minimization problem. The standard approach to such non-convex optimization problems is successive linearization [13]. At each iteration the concave expressions are replaced by a linear approximation at the current point and the LP is solved. A major advantage of successive linearization is that a local minimum can be achieved with only a small number of LPs. Note that the initialization step in Table 1 is a ‘warm start’ approach which can significantly speed up the algorithm.

An important question related to sparsity concerns the nature of training patterns required to be support vectors. It is a well known fact that the set of SVs for the SVM classifier contains all the training patterns which are either inside the margin or are misclassified [17]. This claim is proved by analysis of the KKT conditions and complementary slackness. Furthermore, the SVs of the SVM algorithm cannot all belong to the same class due to the constraints  $\alpha^T y = 0$  and  $\alpha \geq 0$  (see (5)). A similar analysis for the SKM (and  $l_1, l_0$ -SVM) algorithm shows that the type, defined by the relation to the margin, or label of a training point do not determine whether a point is a SV as opposed to the SVM case.

Table 1

The calculation of  $f_0(\delta)$ .

**Input:**  $S_n, C, \tau, \delta$ .

**Output:**  $\alpha^*, b^*$ .

**Algorithm:**

**Require**  $C > 0, 0 < \tau \ll \frac{1}{n}, \delta \in (0, 1]$ .

**Initialize**  $\alpha^* \leftarrow \frac{1}{n}\mathbf{1}, \alpha \leftarrow \mathbf{0}$

**while**  $\|\alpha - \alpha^*\|_1 > \tau$  for 3 iterations

$\alpha \leftarrow \alpha^*$

Solve the problem

$$\begin{aligned}
 & \text{minimize} \quad \mathbf{1}^T \xi + C (\ln 4n + \ln (\frac{8}{\delta^2} \ln n)) v^T \alpha \\
 & \text{subject to} \quad Y (KY\alpha + b\mathbf{1}) \geq (2\delta)\mathbf{1} - \xi \\
 & \quad \quad \quad \mathbf{1}^T \alpha = 1 \\
 & \quad \quad \quad \xi, \alpha \geq 0,
 \end{aligned} \tag{4}$$

where  $v = \frac{8 \ln n}{\delta^2} (1 - \alpha^*)^{\frac{8 \ln n}{\delta^2} - 1}$  and  $\alpha^*$  is the solution of (4).

**end while**

$f_0(\delta) = \mathbf{1}^T \xi^* + C (\ln \frac{4}{\delta} + (\ln 4n + \ln (\frac{8}{\delta^2} \ln n)) \sum_{i=1}^n h(\alpha_i^*, \delta))$ .

## 2.2 Comparison of the SKM algorithm to three SVM algorithms

Since Vapnik's introduction of the SVM algorithm [17] many variants of his algorithm have been introduced. In this section we will review several and compare their theoretical qualities with the SKM algorithm. An important question in regard to sparsity is whether there are types of training patterns that must be SVs?

### Vapnik's SVM

The dual formulation of the SVM algorithm is

$$\begin{aligned}
 & \text{minimize} \quad \frac{1}{2} \alpha^T Y K Y \alpha - \mathbf{1}^T \alpha \\
 & \text{subject to} \quad \alpha^T y = 0 \\
 & \quad \quad \quad 0 \leq \alpha \leq C,
 \end{aligned} \tag{5}$$

with variable  $\alpha \in \mathbb{R}^n$ .

The SVM algorithm is known to often produce classifiers with a relatively large percentage of SVs. The reason for this is clear from its dual formulation (5). The target function is a quadratic function of the variable  $\alpha$ . It is well known [18] that a  $l_2$  norm objective function does not induce sparse solutions in general. Even if the value of the quadratic expression were relatively negli-

gible compared to the linear expression, the latter expression aims to produce solutions with larger values of  $\alpha$ , i.e. the opposite of sparsity.

Another issue related to the lack of sparsity of the SVM algorithm is the fact that all the training patterns which are either inside the margin or which are misclassified must be SVs in the SVM algorithm [17]. Thus there are types of training points that must be SVs in the SVM algorithm. In typical non-separable datasets these training patterns constitute a large percentage of the training set. Furthermore, the SVs cannot all belong to the same class because of the constraints  $\alpha^T y = 0$  and  $\alpha \geq 0$ .

### *The $l_1$ norm SVM*

The  $l_1$  norm SVM [6, pp. 141-142] consists of solving the following optimization problem

$$\begin{aligned}
& \text{minimize} && \mathbf{1}^T \alpha + C \mathbf{1}^T \xi \\
& \text{subject to} && Y (KY\alpha + b\mathbf{1}) \geq \mathbf{1} - \xi \\
& && 0 \leq \alpha \leq C \\
& && \xi \geq 0.
\end{aligned} \tag{6}$$

**Lemma 2.1** *The dual problem related to problem (6) is*

$$\begin{aligned}
& \text{maximize} && \mathbf{1}^T \mu - C \mathbf{1}^T \eta \\
& \text{subject to} && \eta \geq YKY\mu - \mathbf{1} \\
& && \mathbf{0} \leq \mu \leq C\mathbf{1} \\
& && \mu^T y = 0 \\
& && \eta \geq 0,
\end{aligned} \tag{7}$$

*with variables  $\mu, \eta \in \mathbb{R}^n$ .*

**Proof Sketch.** The Lagrangian of Problem (6) is

$$\begin{aligned}
& L(\alpha, b, \xi; \mu, \lambda, \eta, \nu) \\
& = \mathbf{1}^T \alpha + C \mathbf{1}^T \xi + \mu^T (\mathbf{1} - \xi - YKY\alpha - by) - \lambda^T \alpha + \eta^T (\alpha - C\mathbf{1}) - \nu^T \xi \\
& = (\mathbf{1} - YKY\mu - \lambda + \eta)^T \alpha - b (\mu^T y) + (C\mathbf{1} - \mu - \nu)^T \xi + \mathbf{1}^T \mu - C \mathbf{1}^T \eta
\end{aligned} \tag{8}$$

Going through a sequence of standard steps (e.g. [18]), the dual problem is

$$\begin{aligned}
& \text{maximize } \mathbf{1}^T \mu - C \mathbf{1}^T \eta \\
& \text{subject to } \mathbf{1} - YKY\mu - \lambda + \eta = 0 \\
& \mu + \nu = C \mathbf{1} \\
& \mu^T y = 0 \\
& \mu, \lambda, \eta, \nu \succeq 0
\end{aligned} \tag{9}$$

It is straightforward to see that Problems (7) and (9) are equivalent.  $\square$

We claim that the type of the training point in the  $l_1$  norm SVM algorithm does not determine whether the point is a support vector. From the constraints of the dual Problem (9) for every  $i$ ,  $\mu_i + \nu_i = C$  and from complementary slackness for every  $i$ ,  $\nu_i \xi_i = 0$ . Thus if  $\xi_i > 0$  then  $\nu_i = 0$  and then  $\mu_i = C > 0$ . This only implies that  $y^{(i)} (K_i Y \alpha + b) = 1 - \xi_i$ , where  $K_i$  is the  $i$ -th row of the matrix  $K$ . Nothing can be construed on the value of  $\alpha_i$  from this equality relationship.

Note that there is no requirement that the SV set contains training patterns from both classes. Thus we may conclude that *no* training pattern are a-priori constrained to be SVs due to their type or label, as opposed to the SVM algorithm.

### *The $l_0$ norm SVM*

Sparsity is measured with the cardinality function, namely the number of nonzero elements in the vector. However, the cardinality function, also termed as the ' $l_0$  norm', is extremely difficult to minimize because it is a discontinuous function. A popular approach [13] to this difficulty is substituting the continuous logarithm function with  $\delta > 0$ . Thus, instead of minimizing  $\sum_{i=1}^n I_{[x_i \neq 0]}$ , where  $I$  is the indicator function, we minimize  $\sum_{i=1}^n \log(|x_i| + \delta)$ .

Therefore the  $l_0$  norm SVM consists of solving the following concave minimization problem

$$\begin{aligned}
& \text{minimize } \sum_{i=1}^n \log(\alpha_i + \delta) + C \mathbf{1}^T \xi \\
& \text{subject to } Y(KY\alpha + b\mathbf{1}) \geq \mathbf{1} - \xi \\
& \alpha, \xi \geq 0.
\end{aligned} \tag{10}$$

As with the SKM algorithm, we have taken the successive linearization ap-

proach to solve concave minimization problems, by linearizing the concave terms of the objective function. The linearized optimization problem is the following LP

$$\begin{aligned}
& \text{minimize} && \sum_{i=1}^n \frac{\alpha_i}{(\alpha_i^* + \delta)} + C\mathbf{1}^T \xi \\
& \text{subject to} && Y(KY\alpha + b\mathbf{1}) \geq \mathbf{1} - \xi \\
& && \alpha, \xi \geq 0,
\end{aligned} \tag{11}$$

where  $\alpha^*$  is the solution of the previous iteration and in the first iteration it is initialized to a vector of ones. The linearized optimization problem is in fact a reweighted  $l_1$  SVM algorithm; cf. (6). Therefore the analysis of the  $l_1$  SVM algorithm is applicable.

#### *Analysis of the SKM optimization problem*

The analysis of the SKM algorithm is very similar to that of the  $l_1$  SVM algorithm but for completeness we have detailed the full derivation. First the dual problem is derived.

**Lemma 2.2** *The dual problem related to Problem (4) is*

$$\begin{aligned}
& \text{maximize} && (2\delta)\mathbf{1}^T \mu - \lambda \\
& \text{subject to} && YKY\mu \leq \lambda\mathbf{1} + \tilde{C}v \\
& && \mathbf{0} \leq \mu \leq \mathbf{1} \\
& && \mu^T y = 0,
\end{aligned} \tag{12}$$

with variables  $\mu \in \mathbb{R}^n, \lambda \in \mathbb{R}$  and where  $\tilde{C} = C \left( \ln 4n + \ln \left( \frac{8}{\delta^2} \ln n \right) \right)$ .

**Proof Sketch.** The Lagrangian of Problem (4) is

$$\begin{aligned}
& L(\alpha, b, \xi; \mu, \lambda, \nu, \zeta) && \tag{13} \\
& = \mathbf{1}^T \xi + \tilde{C}v^T \alpha + \mu^T ((2\delta)\mathbf{1} - \xi - YKY\alpha - by) + \lambda (\mathbf{1}^T \alpha - 1) - \nu^T \xi - \zeta^T \alpha \\
& = \left( \tilde{C}v - YKY\mu + \lambda\mathbf{1} - \zeta \right)^T \alpha + (\mathbf{1} - \mu - \nu)^T \xi - (\mu^T y) b + (2\delta)\mathbf{1}^T \mu - \lambda
\end{aligned}$$

Going through a sequence of standard steps (e.g. [18]), the dual problem is

$$\begin{aligned}
& \text{maximize } (2\delta)\mathbf{1}^T\mu - \lambda \\
& \text{subject to } \tilde{C}v - YKY\mu + \lambda\mathbf{1} - \zeta = 0 \\
& \mu + \nu = \mathbf{1} \\
& \mu^T y = 0 \\
& \mu, \nu, \zeta \succeq 0
\end{aligned} \tag{14}$$

It is straightforward to see that Problems (12) and (14) are equivalent.  $\square$

We claim that the type of the training point in the SKM algorithm does not determine whether the point is a support vector. From the constraints of the dual Problem (14) for every  $i$ ,  $\mu_i + \nu_i = 1$  and from complementary slackness for every  $i$ ,  $\nu_i \xi_i = 0$ . Thus if  $\xi_i > 0$  then  $\nu_i = 0$  and then  $\mu_i = 1 > 0$ . This only implies that  $y^{(i)}(K_i Y \alpha + b) = 2\delta - \xi_i$ , where  $K_i$  is the  $i$ -th row of the matrix  $K$ . Nothing can be construed on the value of  $\alpha_i$  from this equality relationship.

Note that there is no requirement that the SV set contains training patterns from both classes. Thus we may conclude that *no* training pattern are a-priori constrained to be SVs due to their type or label, as opposed to the SVM algorithm.

### 3 Experiments

In this section we compare the performance of the SVM,  $l_1$ -norm SVM,  $l_0$ -norm SVM<sup>1</sup> and SKM on a number of artificial and real-world two class problem datasets.

#### 3.1 Experimental Methodology

The algorithms are compared by two criteria: the number of SVs and the error rates. The weight  $\alpha_i$  assigned by a classifier to a training pattern  $x^{(i)}$ , determines whether it is regarded as a SV. This weight must fulfil the relative measure -  $\frac{\alpha_i}{\max_i\{\alpha_i\}} \geq \epsilon$ . In this paper  $\epsilon = 0.01$  was used. The definition of the error rate is intrinsically entwined with the protocol for determining the hyperparameter. For the linear kernel the only hyperparameter is  $C$ , while for

---

<sup>1</sup> In all the experiments with this algorithms  $\delta$  was set to 0.1.

the RBF kernel, the kernel width is an additional hyperparameter. Given an *a-priori* partitioning of the dataset into training and test sets, the following protocol for determining the hyperparameter value and defining the error rate is suggested:

- (1) Define a set  $\mathcal{R}$  of values of the hyperparameters  $R$  for all datasets. The set  $\mathcal{R}$  consists of a *predetermined* number of values.
- (2) Calculate the N-fold Cross Validation (CV) (see e.g. [19]) error for each value of  $R$  from set  $\mathcal{R}$  on the *training* set. Five fold CV was used throughout all the datasets.
- (3) Use the classifier with the value of  $R$  which produced the lowest CV error to classify the test set. This is the reported error rate.

If the dataset is not partitioned *a-priori* into a training and test set, it can be randomly divided into  $n_p$  contiguous training and ‘test’ sets. Each training set contains  $n \frac{n_p-1}{n_p}$  patterns and the corresponding test set consists of  $\frac{n}{n_p}$  patterns. Once the dataset is thus partitioned, the above steps can be implemented. The error rate and the number of selected features are then defined as the average on the  $n_p$  problems. The value  $n_p = 10$  was used for all datasets, where an *a-priori* partitioning was not available.

An exhaustive search for the optimal value of the hyperparameters would have produced better results for all algorithms. However, in such an approach a fair comparison would have been much more complicated to quantify.

### 3.2 Data sets

The synthetic datasets are drawn randomly from gaussian distributions with different means for each class. The datasets were constructed in the following fashion. The number of features was  $d = 2$ . The probability of  $y = 1$  or  $-1$  was equal. Features  $x_j, j = 1, 2$  were drawn as  $x|y \sim \mathcal{N}(y\mathbf{1}, \Lambda)$ , with the covariance matrix  $\Lambda = VDVT^T$ . The matrix  $D$  was diagonal with the values 1, 2 and  $V$  a randomly generated unitary matrix. Finally, the label of each pattern was reversed, i.e. multiplied by -1, with independent probability  $p_{out}$ .

The number of features, the number of patterns and the partitioning into train and test sets of the real-world datasets are detailed in Table 2. The datasets were taken from the UCI repository.

Table 2

The real-world datasets. Dataset 5 was divided into a training set of 2000 patterns and the remaining were the test set.

No.	Name	Features	Patterns
1	Ionosphere	34	351
2	Pima	8	768
3	Bupa	6	345
4	Cleveland heart	13	297
5	Mushroom	22	2000/6124

### 3.3 Experimental results

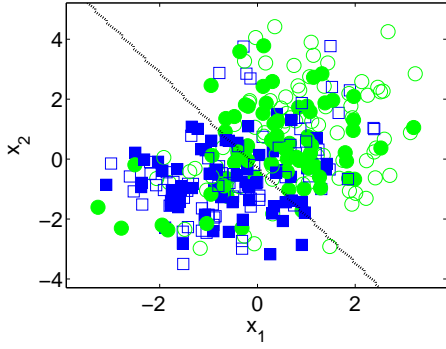
The training features of all the datasets and for all the algorithms were preprocessed to have zero mean and unit variance. The test examples were processed accordingly. Two kernels were used:

- (1) Linear -  $K_{i,j} = \langle x^{(i)}, x^{(j)} \rangle$ .
- (2) Gaussian -  $K_{i,j} = \exp(-(\langle x^{(i)}, x^{(j)} \rangle)^2 / (2\sigma^2))$ .

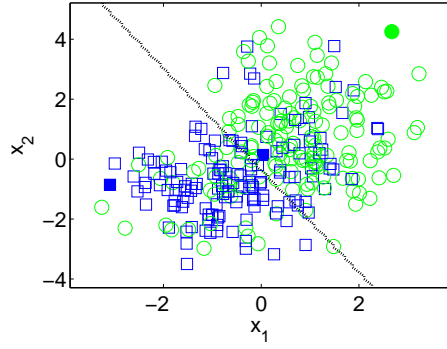
The grids used for the hyperparameters were selected numerically to produce performance of *all* algorithms similar to the best results in the literature. The hyperparameter set  $\mathcal{R}$  for the linear kernel consisted of 10 logarithmically spaced values of  $C$  between  $10^{-3}$  and  $10^1$ . The hyperparameter set  $\mathcal{R} = \mathcal{C} \times \sigma$  for the Gaussian kernel consisted of 25 values for the two hyperparameters. The hyperparameter set  $\mathcal{C}$  consisted of 5 logarithmically spaced values of  $C$  between  $10^{-3}$  and  $10^1$ . The hyperparameter set  $\sigma$  consisted of 5 linearly spaced values of  $\sigma$  between 1 and 20 for datasets 2 and 5, and between 1 and 5 for datasets 1,3 and 4.

Note that only in the SKM algorithm the hyperparameter  $C$  multiplies the complexity term rather than the penalty on the training error. Therefore, in order to maintain consistency, the set  $\mathcal{C}^{-1}$ , namely logarithmically spaced values of  $C$  between  $10^{-1}$  and  $10^3$ , was used for the SKM algorithm.

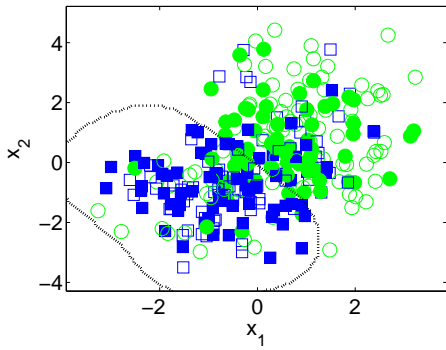
For illustration purposes a synthetic dataset with  $p_{out} = 0.2$  was generated. The training set consisted of 300 patterns and the test set of 1000 patterns. The results are presented in Figure 2. Additionally ten datasets were generated with 1000 training patterns, 2000 test patterns and  $p_{out} = 0.25$ . The results are summarized in Table 3. It is clear especially from the results of Table 3 that all the algorithms produce similar classifiers with similar error rates. However, the SKM algorithm achieves these error rates with only a fraction of the SVs needed for the SVM algorithm, as seen in Figure 2.



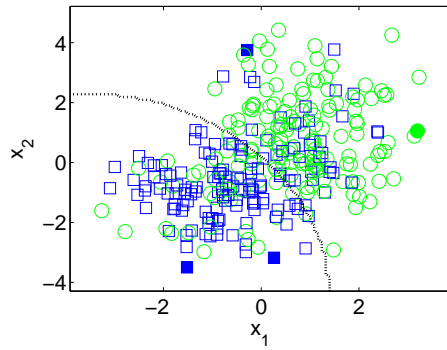
(a) SVM linear kernel: 26.3% errors and 71.3% SVs.



(b) SKM linear kernel: 26.5% errors and 1.0% SVs.



(c) SVM Gaussian kernel: 26.9% errors and 72.7% SVs.



(d) SKM Gaussian kernel: 26.2% errors and 1.3% SVs.

Fig. 2. The error rate and the percentage of SVs for the synthetic dataset 1. The dotted line signifies the decision line,  $f(x) = 0$ . The squares and circles mark the locations and labels of the  $\{-1, +1\}$  training patterns. The filled squares and circles are the SVs.

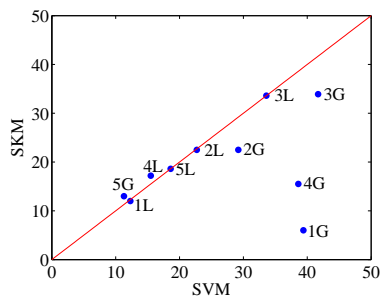
Table 3

The error and SV percentage (in parentheses) for the synthetic datasets with linear and gaussian (RBF) kernels. The number following the  $\pm$  represents the standard deviation of either the error or SV percentage. The lowest error and sparsest solution for every dataset are marked in boldface.

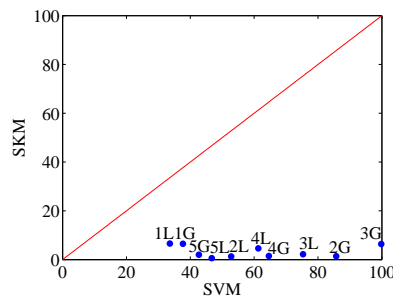
Kernel	Linear	RBF
SVM	30.6 $\pm$ 0.4 (75.1 $\pm$ 0.9)	<b>30.4 <math>\pm</math> 0.3</b> (71.4 $\pm$ 0.9)
$l_1$ SVM	30.6 $\pm$ 0.4 (0.9 $\pm$ 0.4)	30.5 $\pm$ 0.3 (3.9 $\pm$ 1.2)
$l_0$ SVM	<b>30.4 <math>\pm</math> 0.4</b> ( <b>0.2<math>\pm</math>0.1</b> )	30.5 $\pm$ 0.3 ( <b>0.7 <math>\pm</math> 0.2</b> )
SKM	<b>30.4 <math>\pm</math> 0.4</b> (0.3 $\pm$ 0.1)	30.5 $\pm$ 0.3 (1.1 $\pm$ 0.3)

The performance of the algorithms on the real-world datasets is detailed in Tables 4-7. Figure 3 is provided for a more visual comparison. It is clear that for both kernels that the SKM algorithm produced significantly sparser

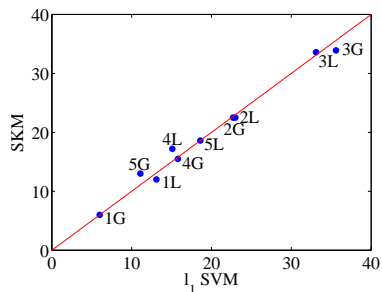
solutions than the SVM and  $l_1$  SVM while yielding error rates comparable to the aforementioned algorithms. Additionally, the performance of the SKM in regard to sparsity and error rate is competitive with that of the  $l_0$  SVM algorithm.



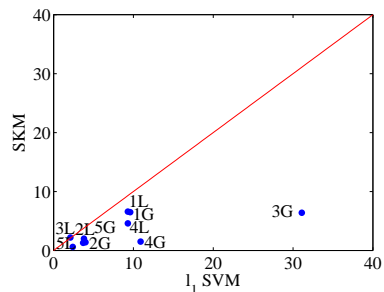
(a) SKM vs. SVM: error percentage.



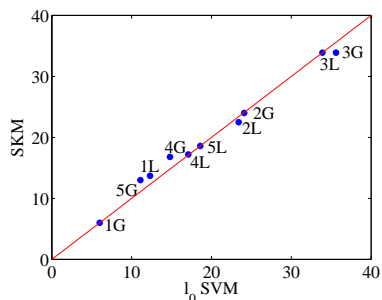
(b) SKM vs. SVM: SV percentage.



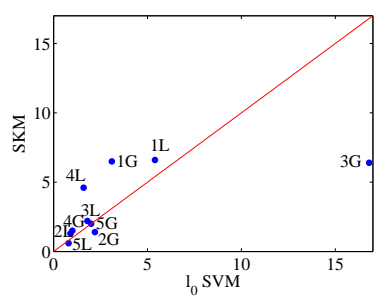
(c) SKM vs.  $l_1$  SVM: error percentage.



(d) SKM vs.  $l_1$  SVM: SV percentage.



(e) SKM vs.  $l_0$  SVM: error percentage.



(f) SKM vs.  $l_0$  SVM: SV percentage.

Fig. 3. A comparison of the error rate and SV percentage for the real-world datasets between the SKM and SVM algorithms. The number indicates the dataset index and the letters L,G indicate the type of kernel used (linear or gaussian respectively).

The runtime of the SKM is comparable to all the other competitors as it requires on average solving only approximately three LPs for a given  $\delta$ .

Table 4

The error percentage for the real-world datasets with a linear kernel.

SVM	$l_1$ SVM	$l_0$ SVM	SKM
$12.3 \pm 7.1$	$13.1 \pm 7.7$	$12.3 \pm 8.0$	<b><math>12.0 \pm 8.2</math></b>
$22.7 \pm 5.1$	$22.7 \pm 5.1$	$23.4 \pm 6.0$	<b><math>22.5 \pm 5.0</math></b>
$33.6 \pm 12.6$	<b><math>33.1 \pm 11.1</math></b>	$33.9 \pm 11.8$	$33.6 \pm 11.2$
$15.5 \pm 5.9$	<b><math>15.1 \pm 6.5</math></b>	$17.1 \pm 7.5$	$17.2 \pm 5.6$
<b>18.6</b>	<b>18.6</b>	<b>18.6</b>	<b>18.6</b>

Table 5

The SV percentage for the real-world datasets with a linear kernel.

SVM	$l_1$ SVM	$l_0$ SVM	SKM
$33.6 \pm 13.0$	$9.3 \pm 1.2$	<b><math>5.4 \pm 1.5</math></b>	$6.6 \pm 1.3$
$52.8 \pm 1.4$	$3.7 \pm 2.5$	<b><math>0.9 \pm 0.2</math></b>	$1.3 \pm 0.1$
$75.3 \pm 2.7$	$2.1 \pm 0.5$	<b><math>1.8 \pm 0.2</math></b>	$2.2 \pm 0.2$
$61.3 \pm 10.5$	$9.3 \pm 3.4$	<b><math>1.6 \pm 0.9</math></b>	$4.6 \pm 0.7$
46.7	2.4	0.8	<b>0.6</b>

Table 6

The error percentage for the real-world datasets with a gaussian kernel.

SVM	$l_1$ SVM	$l_0$ SVM	SKM
<b><math>4.8 \pm 1.0</math></b>	$6.6 \pm 1.7$	$5.4 \pm 1.4$	$5.7 \pm 1.4$
$29.2 \pm 8.1$	$23.0 \pm 6.2$	$24.1 \pm 5.5$	<b><math>22.5 \pm 6.2</math></b>
$34.5 \pm 3.5$	$33.3 \pm 3.8$	<b><math>29.8 \pm 1.8</math></b>	$30.2 \pm 3.0$
$15.8 \pm 1.9$	$16.8 \pm 2.0$	$5.8 \pm 2.0$	<b><math>15.4 \pm 2.5</math></b>
11.3	<b>11.1</b>	<b>11.1</b>	13.0

Table 7

The SV percentage for the real-world datasets with a gaussian kernel.

SVM	$l_1$ SVM	$l_0$ SVM	SKM
$46.6 \pm 0.4$	$10.6 \pm 0.3$	<b><math>2.4 \pm 0.1</math></b>	$8.3 \pm 0.5$
$85.7 \pm 0.6$	$4.0 \pm 1.2$	$2.2 \pm 1.9$	<b><math>1.4 \pm 0.3</math></b>
$81.3 \pm 0.6$	$15.7 \pm 1.2$	<b><math>2.6 \pm 0.1</math></b>	$6.7 \pm 0.9$
$59.4 \pm 3.0$	$10.1 \pm 0.9$	<b><math>1.9 \pm 0.2</math></b>	$2.0 \pm 0.3$
42.7	3.8	<b>2.0</b>	<b>2.0</b>

## 4 Discussion

This paper presents a new binary classification algorithm based on a direct attempt to minimize a generalization error bound. Finding the global minimum requires solving a concave optimization problem. The approach taken here was based on a successive linearization procedure. Thus the algorithm consists of solving a number of LPs for which extremely efficient solvers are available.

It is an arduous task to significantly improve on the SVM algorithm's performance on two-class datasets possessing no unique traits, e.g. many noisy features. However, there is room for considerable improvement in the computational load and representational simplicity of the final classifier. Indeed the SKM algorithm produced very similar error rates to the SVM algorithm. However, it did so by using only a small fraction of SVs. Thus the final classifier is proportionally faster than the SVM algorithm.

Additionally, the SKM algorithm produced similar error rates to the  $l_1$  and  $l_0$  SVMs. However, it used significantly less SVs than the  $l_1$  SVM and was highly competitive with the  $l_0$  SVM. As opposed to the  $l_0$  algorithm, it was based on a rigorous generalization error bound, which motivates further algorithmic improvements as bounds get tighter.

## References

- [1] B. Schölkopf, A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press, Cambridge, MA, 2002.
- [2] R. Herbrich, *Learning Kernel Classifiers, theory and algorithms*, 1st Edition, MIT Press, 2002.
- [3] I. Steinwart, Sparseness of support vector machines, *The Journal of Machine Learning Research* 4 (2003) 1071–1105.
- [4] Y. Gat, Adaptive sparseness for supervised learning, *Machine Learning* 42 (2001) 233–239.
- [5] S. A. Jaeger, Generalization bounds and complexities based on sparsity and clustering for convex combinations of functions from random classes, *The Journal of Machine Learning Research* 6 (2005) 307–340.
- [6] O. L. Mangasarian, D. R. Musicant, *Complementarity: Applications, Algorithms and Extensions*, Kluwer Academic Publishers, 2001.
- [7] P. M. Drezet, R. F. Harrison, A new method for sparsity control in support vector classification and regression, *Pattern Recognition* 34 (2001) 111–125.

- [8] G. M. Fung, O. L. Mangasarian, A. J. Smola, Minimal kernel classifiers, *The Journal of Machine Learning Research* 3 (2002) 303–321.
- [9] M. Figueiredo, Adaptive sparseness for supervised learning, *IEEE Trans. Pattern Analysis and Mach. Intell.* 25 (2003) 1150–1159.
- [10] P. B. Nair, A. Choudhury, A. J. Keane, Some greedy learning algorithms for sparse regression and classification with mercer kernels, *The Journal of Machine Learning Research* 3 (2002) 781–801.
- [11] M. Wu, B. Schölkopf, G. Bakir, A direct method for building sparse kernel learning algorithms, *Journal of Machine Learning Research* 7 (2006) 603–624.
- [12] M. E. Tipping, Sparse bayesian learning and the relevance vector machine, *The Journal of Machine Learning Research* 1 (2001) 211–244.
- [13] M. Fazel, H. Hindi, S. P. Boyd, Log-det heuristic for matrix rank minimization with applications to hankel and euclidean distance matrices, *Proceedings of the American Control Conference* 3 (2003) 2156–2162.
- [14] D. Bertsimas, J. N. Tsitsiklis, *Introduction to Linear Optimization*, Athena Scientific, 1997.
- [15] R. Horst, P. M. Pardalos, *Handbook of Global Optimization*, 1st Edition, Kluwer academic publishers, 1995.
- [16] D. P. Bertsekas, *Nonlinear Programming*, 2nd Edition, Athena Scientific, 1999.
- [17] V. Vapnik, *Statistical Learning Theory*, Wiley Interscience, New York, 1998.
- [18] S. Boyd, L. Vandenberghe, *Convex Optimization*, 1st Edition, Cambridge University Press, 2004.
- [19] R. O. Duda, P. E. Hart, D. G. Stork, *Pattern Classification*, 2nd Edition, Cambridge University Press, 2000.

About the Author DORI PELEG received his B.Sc., M.Sc. and Ph.D. degrees in Electrical Engineering from the Technion- Israel Institute of Technology in 2002, 2004 and 2008. His research areas include optimization and pattern recognition.

About the Author RON MEIR received the B.Sc. degree in Physics and Mathematics from the Hebrew university in Jerusalem in 1982, and the M.Sc. and Ph.D. degrees in theoretical Physics from the Weizmann Institute of Science in 1984 and 1988, respectively. After two years as a Weizmann research fellow at Caltech he spent two years working at Bell Communications Research on various aspects of neural network design and analysis. He joined the Department of Electrical Engineering at the Technion in 1992. His main current research areas are computational neuroscience and the theory of learning.