BINARY CLASSIFICATION BY MINIMIZING THE MEAN SQUARED SLACK

Konstantinos I. Diamantaras and Margarita Kotti

TEI of Thessaloniki, Department of Informatics, Sindos 57400, Greece
email: kdiamant@it.teithe.gr, mkotti@it.teithe.gr

ABSTRACT

The paper presents a new binary classification method based on the minimization of the slack variables energy called the Mean Squared Slack (MSS). We deliver preliminary mathematical results which support the motivation behind our approach. We show that (a) in the linearly separable case the minimum MSS is attained at a separating vector, while (b) the minimizer in the linearly non-separable case is bounded but not zero. The method is conceptually simple: it solves a linear system at each iteration and it converges, typically, within a few iterations. Its complexity is obviously related to the size of the system which, in the linear case, is equal to the input pattern dimension. The method is extended to the non-linear case using kernels. Simulations demonstrate that the method is competitive with respect to computation time, accuracy, and generalization performance compared to state of the art SVM methods.

Index Terms— Binary classification, Kernel methods, slack minimization

1. INTRODUCTION

Machine Learning (ML) has been a very active research area over the past decades and pattern classification is a core problem in ML. The increased interest in recent years can be partly justified by the ever increasing size of the databases that need to be effectively processed. Support Vector Machines (SVMs) are considered to be the state-of-the-art methods for the classification task and thus have found application in practically every machine learning problem. SVMs achieved the state-of-art performance on the MNIST benchmark set, whereas they are widely applied in computer vision, including tasks such as object recognition and detection. They are most exploited in bio-informatics and natural language processing. This may partially attributed to the fact that both fields deal with high-dimensional problems, such as microarray processing tasks, and text categorization. Additionally, SVMs have been tested for speech and speaker recognition, emotion classification, object detection, e-learning, database marketing, intrusion detection, geo- and environmental sciences, finance time series forecasting and high energy physics. The aforementioned list of applications is just indicative but not exhaustive. Currently, there is a number of readily available software packages for SVM optimization. Here, we resort to SVMlight [1] and libsvm[2], for sake of comparison with the proposed approach.

Boser et al. [3] were the first to use kernels to construct a nonlinear estimation algorithm, which is the hard margin predecessor of the SVM [4] [5]. The substitution of kernels for dot products aims to transform a linear geometric algorithm into a nonlinear one. This way, hyperplane classifiers evolved to SVMs [6].

Let us present a concise introduction to SVMs [7]. Considering a set of training data

$$\mathcal{X} = \{(\mathbf{x}(i), t(i)) \mid \mathbf{x}(i) \in \mathbb{R}^n\}_{i=i}^N \tag{1}$$

where $\mathbf{x}(i)$ is a feature vector, $t(i) \in \{-1, 1\}$ is the class label of $\mathbf{x}(i)$, and N is the size of $\mathcal{X}$. SVMs aim to search a hyperplane in the Reproducing Kernel Hilbert Space (RKHS) that maximizes the margin between the two classes of data in $\mathcal{X}$ with the smallest training error [8]. This problem can be formulated as a quadratic optimization problem: $\min P(\mathbf{w}, b, \xi) = \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^N \xi(i)$ subject to $1 - t(i)(\mathbf{w}^T\mathbf{\Phi}(\mathbf{x}(i)) + b) \le \xi(i), \quad \xi(i) > 0$, where $\mathbf{w}$ is a weight vector, b is a threshold, C a regularization hyperparameter, and $\mathbf{\Phi}$ a function which maps $\mathbf{x}(i)$ to an RKHS space. The decision function of SVMs is $f(\mathbf{x}) = \mathbf{w}^T\mathbf{\Phi}(\mathbf{x}) + b$. Normally, $\mathbf{w}$ and b would be attained by minimizing P; however, $\mathbf{\Phi}$ can be a high (or even infinite) dimensional function. Therefore, we make the assumption that $\mathbf{w} = \sum_{i=1}^N a_i\mathbf{\Phi}(\mathbf{x}(i))$ and we resort to the solution of the dual problem, i.e. $\min D(\mathbf{a}) = \frac{1}{2}\mathbf{a}^T\mathbf{Q}\mathbf{a} - \mathbf{a}^T\mathbf{1}$, subject to $0 \le \mathbf{a} \le \mathbf{C}, \quad \mathbf{t}^T\mathbf{a} = 0$ where $t(i)t(j)\mathbf{\Phi}^T(\mathbf{x}(i))\mathbf{\Phi}(\mathbf{x}(j)) = [Q]_{ij}$ is the Lagrangian multiplier variable (or dual variable). SVMs utilize the kernel trick by specifying a kernel function $K(\mathbf{x}(i), \mathbf{x}(j))$ defined as the inner-product $\mathbf{\Phi}^T(\mathbf{x}(i))\mathbf{\Phi}(\mathbf{x}(j))$. Thus, $[Q]_{ij} = t(i)t(j)K(\mathbf{x}(i), \mathbf{x}(j))$. When the given kernel function K is psd (positive semidefinite), the dual problem is a convex quadratic programming (QP) problem with linear constraints and can be solved in polynomial time.

Recent methods exploit the idea of constructing kernel algorithms, where the starting point is not a linear algorithm [6], but a linear criterion. The latter can be turned into a condition involving an efficient optimization over a large function class using kernels, thus yielding tests for independence of random variables, or tests for solving the two-sample problem. A linear criterion may be for example that two random variables have zero covariance, or that the means of two samples are identical. Other alternatives try to improve scalability, exploiting parallel SVM (PSVM) [7], which reduces memory use through performing a row-based, approximate matrix factorization, and which loads only essential data to each machine to perform parallel computation.

2. MEAN SQUARED SLACK MINIMIZATION

Consider the classification task for the training set $\mathcal{X}$ in (1). The task amounts to finding a proper weight vector $\mathbf{w} \in \mathbb{R}^n$ and bias b that solve the following set of inequalities:

$$t(i)(\mathbf{w}^T\mathbf{x}(i) + b) \ge \gamma, \quad i = 1, \cdots, N \tag{2}$$

for some positive margin $\gamma > 0$.

In general, there may not exist any feasible solution for the set of inequalities (2). It is then useful to define a slack variable $\xi(i)$, associated with pattern i,

$$\xi(i) = \max\{\gamma - t(i)[\mathbf{w}^T\mathbf{x}(i) - b],\ 0\} \tag{3}$$

so that

$$t(i)[\mathbf{w}^T\mathbf{x}(i)+b] \geq \gamma - \xi(i), \quad \xi(i) \geq 0. \tag{4}$$

Typically a maximum margin classifier, such as an SVM, seeks to minimize the norm of the weight vector $\mathbf{w}$ under the constraints described in (4) and putting a penalty on the slack variables. However, the computational complexity of the resulting quadratic programming problem can be quite high, especially for large datasets (e.g. $N >> 10000$).

An alternative approach would be to minimize the **Mean Squared Slack** (**MSS**) defined as

$$J_{MSS} = \frac{1}{2}\bar{E}\{\xi^2 \mid \xi > 0\} \tag{5}$$

$$= \frac{1}{2}\bar{E}\Big\{(\gamma - t[\mathbf{w}^T\mathbf{x} - b])^2 \,\Big|\, \gamma > t[\mathbf{w}^T\mathbf{x}+b]\Big\} \tag{6}$$

where $\bar{E}\{X \mid Y\}$ is the empirical average of the sequence $X(i)$ under condition Y:

$$\bar{E}\{X \mid Y\} = \frac{1}{N_Y}\sum_{\substack{\text{all } i \text{ where} \\ Y \text{is true}}} X(i) \tag{7}$$

and N_Y is the number of instances where Y is true. Note that the average operator in (5) is conditioned on the inequality $\xi > 0$. In other words, we care only for those patterns which give $t(i)(\mathbf{w}^T\mathbf{x} + b) < \gamma$. This is reasonable, since, in the classification context, *only* the "bad" patterns that fail to satisfy inequality (2) should contribute to the cost, while all the others should not.

Motivated by our quest for a faster algorithm we explore, here, methods minimizing the MSS. First, however, we need to establish a theoretical basis justifying the use of this approach. This is provided by the following Lemma:

Lemma 1 *The following statements are true:*

(a) if problem (2) *is linearly separable, then the minimum* $J_{MSS} = 0$ *is attained by* $[\mathbf{w}^T,\, b]$ *iff* $[\mathbf{w}^T,\, b]$ *is a separating vector;*

(b) if the problem is not linearly separable, then the cost function J_{MSS} *attains its minimum for some* $[\mathbf{w}^T,\, b]$ *with* $0 < \|[\mathbf{w}^T,\, b]\| < \infty$.

PROOF.

(a) By definition, $J_{MSS} \geq 0$. According to (5) the minimum value $J_{MSS} = 0$ is attained iff $\xi(i) = 0$ for all i. However, according to (3), $\xi(i) = 0$ for all i, iff $[\mathbf{w}^T,\, b]$ is a separating vector.

(b) Since J_{MSS} is bounded from below by zero it has an infimum. We need to show this infimum is not attained neither at zero nor at infinity.

First we shall show that the vector $[\mathbf{w}^T,\, b] = 0$ does not attain the infimum. Consider any vector $[\mathbf{w}_0^T,\, b_0]$ such that

$$\sum_{i=1}^{N} t(i)[\mathbf{w}_0^T\mathbf{x}(i) + b_0] > 0. \tag{8}$$

We can always select such a vector: simply pick any $[\mathbf{w}_1^T,\, b_1]$ for which $\sum_{i=1}^N t(i)[\mathbf{w}_1^T\mathbf{x}(i) + b_1] \neq 0$, and set $[\mathbf{w}_0^T,\, b_0] = -[\mathbf{w}_1^T,\, b_1]$, if $\sum_{i=1}^N t(i)[\mathbf{w}_1^T\mathbf{x}(i)+b_1] < 0$, otherwise let $[\mathbf{w}_0^T,\, b_0] = [\mathbf{w}_1^T,\, b_1]$.

Now consider the vector $[\mathbf{w}^T,\, b] = \varepsilon[\mathbf{w}_0^T,\, b_0]$ for some small $\varepsilon > 0$, such that $\varepsilon\, t(i)[\mathbf{w}_0^T\mathbf{x}(i) + b_0] < \gamma$, $\forall i$. We have

$$J_{MSS}(\varepsilon\mathbf{w}_0, \varepsilon b_0) = \frac{1}{2}\gamma^2 - \varepsilon\delta_1 + \varepsilon^2\delta_2$$

where $\delta_1 = \gamma\sum_{i=1}^N t(i)[\mathbf{w}_0^T\mathbf{x}(i) + b_0]$, $\delta_2 = \frac{1}{2}\sum_{i=1}^N [\mathbf{w}_0^T\mathbf{x}(i) + b_0]^2$.

Since both $\delta_1 > 0$ and $\delta_2 > 0$, it follows that, if we impose the additional constraint $\varepsilon < \frac{\delta_1}{\delta_2}$, then

$$J_{MSS}(\varepsilon\mathbf{w}_0, \varepsilon b_0) < \frac{1}{2}\gamma^2 = J_{MSS}(0).$$

Therefore, $[\mathbf{w}^T,\, b] = 0$ is not a minimizer.

It remains to show that the infimum is not attained at infinity. To that end we shall fix any arbitrary vector $[\mathbf{w}^T,\, b] \neq 0$ and we shall consider the behavior of the cost along the line $\lambda[\mathbf{w}^T,\, b]$ as $\lambda \to \infty$. We'll show that the function

$$J_{MSS}(\lambda) = \frac{1}{2}\bar{E}\Big\{(\gamma - \lambda t[\mathbf{w}^T\mathbf{x}+b])^2 \,\Big|\, \gamma > \lambda t[\mathbf{w}^T\mathbf{x}+b]\Big\} \tag{9}$$

increases monotonically for sufficiently large λ. We rewrite (5) as follows

$$J_{MSS}(\lambda) \;=\; J_1(\lambda) + J_2(\lambda) \tag{10}$$

$$J_1(\lambda) = \frac{1}{2}\bar{E}\Big\{(\gamma - \lambda t[\mathbf{w}^T\mathbf{x}+b])^2 \,\Big|\, 0 \geq \lambda t[\mathbf{w}^T\mathbf{x}+b]\Big\} \tag{11}$$

$$J_2(\lambda) = \frac{1}{2}\bar{E}\Big\{(\gamma - \lambda t[\mathbf{w}^T\mathbf{x}+b])^2 \,\Big|\, \gamma \geq \lambda t[\mathbf{w}^T\mathbf{x}+b] > 0\Big\} \tag{12}$$

We see that

$$\begin{aligned} J_1(\lambda) &= \frac{1}{2}\bar{E}\Big\{(\gamma - \lambda t[\mathbf{w}^T\mathbf{x}+b])^2 \,\Big|\, \\ &\qquad\qquad 0 \geq t[\mathbf{w}^T\mathbf{x}+b]\Big\} \\ &= \frac{1}{2}\gamma^2 + \lambda c_1 + \lambda^2 c_2 \end{aligned} \tag{13}$$

where

$$\begin{aligned} c_1 &= \gamma\bar{E}\Big\{-t[\mathbf{w}^T\mathbf{x}+b] \,\Big|\, 0 \geq t[\mathbf{w}^T\mathbf{x}+b]\Big\} \\ c_2 &= \bar{E}\Big\{\frac{1}{2}|\mathbf{w}^T\mathbf{x}+b|^2 \,\Big|\, 0 \geq t[\mathbf{w}^T\mathbf{x}+b]\Big\} \end{aligned}$$

Since the problem is non-separable, the set $\{i \,:\, 0 \geq t(i)[\mathbf{w}^T\mathbf{x}(i) + b]\}$ is non-empty. It follows that $c_1, c_2 > 0$, thus by Eq. (13),

$$J_1(\lambda) \to \infty \text{ as } \lambda \to \infty. \tag{14}$$

On the other hand, J_2 is bounded below by zero:

$$J_2(\lambda) > 0. \tag{15}$$

Combining (14) and (15) with (10) we conclude that

$$J_{MSS}(\lambda) \to \infty \text{ as } \lambda \to \infty. \tag{16}$$

■

With the above result in mind, we can study the optimization of J_{MSS} through the Karush-Kuhn-Tucker (KKT) conditions. We then compute the gradient of J_{MSS} w.r.t. $\mathbf{w}$, and b as

$$\mathbf{g}_w = \bar{E}\Big\{\mathbf{x}\mathbf{x}^T\mathbf{w} + b\mathbf{x} - \gamma t\mathbf{x} \,\Big|\, \gamma > t[\mathbf{w}^T\mathbf{x}+b]\Big\} \tag{17}$$

$$g_b = \bar{E}\Big\{\mathbf{x}^T\mathbf{w} + b - \gamma t \,\Big|\, \gamma > t[\mathbf{w}^T\mathbf{x}+b]\Big\} \tag{18}$$

or

$$\mathbf{g}_w = \mathbf{R}_x\mathbf{w} + b\mathbf{m}_x - \gamma\mathbf{m}_{tx} \quad (19)$$

$$g_b = \mathbf{m}_x^T\mathbf{w} + b - \gamma\mathbf{m}_t \quad (20)$$

where $\mathbf{R}_x = \bar{E}\{\mathbf{x}\mathbf{x}^T \mid \gamma > t[\mathbf{w}^T\mathbf{x} + b]\}$, $\mathbf{m}_{tx} = \bar{E}\{t\mathbf{x} \mid \gamma > t[\mathbf{w}^T\mathbf{x} + b]\}$, $\mathbf{m}_x = \bar{E}\{\mathbf{x} \mid \gamma > t[\mathbf{w}^T\mathbf{x} + b]\}$, $\mathbf{m}_t = \bar{E}\{t \mid \gamma > t[\mathbf{w}^T\mathbf{x} + b]\}$. Setting the gradient equal to zero we obtain

$$\begin{bmatrix} \mathbf{w}^* \\ b^* \end{bmatrix} = \gamma \begin{bmatrix} \mathbf{R}_x & \mathbf{m}_x \\ \mathbf{m}_x^T & 1 \end{bmatrix}^+ \begin{bmatrix} \mathbf{m}_{tx} \\ \mathbf{m}_t \end{bmatrix} \quad (21)$$

Unfortunately, (21) is not a computable solution because the unknown vector $[\mathbf{w}^T, b]$ appears on both sides of the equality (note that $\mathbf{R}_x$, $\mathbf{m}_{tx}$, $\mathbf{m}_x$ and $\mathbf{m}_t$ are all functions of $\mathbf{w}$, b).

This analysis however, suggests the following iterative procedure for minimizing the average squared slack:

Algorithm 1 (Linear Minimum MSS)

Step 0: Select an initial random vector $\mathbf{w}_0$ *and bias* b_0*;*
Step 1: Let $k = 0$*;*
Step 2: Find the set $S = \{i : \gamma > t(i)[\mathbf{w}_k^T\mathbf{x}(i) + b_k]\}$*;*
Let $\mathbf{R}_x = \frac{1}{|S|}\sum_{i\in S}\mathbf{x}(i)\mathbf{x}(i)^T$, $\mathbf{m}_{tx} = \frac{1}{|S|}\sum_{i\in S}t(i)\mathbf{x}(i)$, $\mathbf{m}_x = \frac{1}{|S|}\sum_{i\in S}\mathbf{x}(i)$, *and* $\mathbf{m}_t = \frac{1}{|S|}\sum_{i\in S}t(i)$.
Step 3: Update

$$\begin{bmatrix} \mathbf{w}_{k+1} \\ b_{k+1} \end{bmatrix} = \gamma \begin{bmatrix} \mathbf{R}_x & \mathbf{m}_x \\ \mathbf{m}_x^T & 1 \end{bmatrix}^+ \begin{bmatrix} \mathbf{m}_{tx} \\ \mathbf{m}_t \end{bmatrix} \quad (22)$$

Step 5: Set $k \leftarrow k + 1$*;*
Step 6: If terminating condition is met then STOP
else goto Step 2;

■

Termination can be decided using a combination of the following conditions:

(a) the maximum number of iterations is reached;

(b) the number of misclassified patterns is less or equal to a certain threshold;

(c) the size of the update step is below a certain threshold.

3. THE KERNEL TRICK

In order to facilitate the computation of nonlinear separating surfaces we can use a nonlinear mapping $\mathbf{\Phi} : \mathbb{R}^n \to \mathbb{R}^m$ of the input vectors $\mathbf{x}$ into an m-dimensional space, where $m > n$ (sometimes $m = \infty$). The separating vector $\mathbf{w}$ in $\mathbb{R}^m$ is a linear combination of the mapped inputs

$$\mathbf{w} = \sum_{j\in G} a(j)\mathbf{\Phi}(\mathbf{x}(j))$$

where G is some subset of $\{1, 2, \cdots, N\}$. We shall avoid the explicit computation of $\mathbf{\Phi}()$ using the scalar kernel function $K(\mathbf{x}, \mathbf{y}) = \mathbf{\Phi}(\mathbf{x})^T\mathbf{\Phi}(\mathbf{y})$. Now, S is the set of indexes i for which

$$t(i)y(i) < \gamma, \quad (23)$$

where

$$y(i) = \sum_{j\in G} K(\mathbf{x}(i), \mathbf{x}(j))a(j) + b. \quad (24)$$

Eq. (21) becomes

$$\begin{bmatrix} \sum_{i\in S}\mathbf{\Phi}(\mathbf{x}(i))\mathbf{\Phi}(\mathbf{x}(i))^T & \sum_{i\in S}\mathbf{\Phi}(\mathbf{x}(i)) \\ \sum_{i\in S}\mathbf{\Phi}(\mathbf{x}(i))^T & |S| \end{bmatrix} \cdot \begin{bmatrix} \sum_{j\in G}\mathbf{\Phi}(\mathbf{x}(j))a(j) \\ b \end{bmatrix} = \gamma \begin{bmatrix} \sum_{i\in S}t(i)\mathbf{\Phi}(\mathbf{x}(i)) \\ \sum_{i\in S}t(i) \end{bmatrix}. \quad (25)$$

Call $K_{ij} = K(\mathbf{x}(i), \mathbf{x}(j))$ and define the row vectors $\mathbf{k}_i^T = [K_{ij}]_{j\in G}$ for $i = 1, \cdots, N$. Then we have

$$\begin{bmatrix} \sum_{i\in S}\mathbf{\Phi}(\mathbf{x}(i))\mathbf{k}_i^T & \sum_{i\in S}\mathbf{\Phi}(\mathbf{x}(i)) \\ \sum_{i\in S}\mathbf{k}_i^T & |S| \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ b \end{bmatrix} = \gamma \begin{bmatrix} \sum_{i\in S}t(i)\mathbf{\Phi}(\mathbf{x}(i)) \\ \sum_{i\in S}t(i) \end{bmatrix} \quad (26)$$

which can be simplified into

$$\begin{bmatrix} \mathbf{\Phi}_S \\ \mathbf{1}^T \end{bmatrix} [\mathbf{K}\ \mathbf{1}] \begin{bmatrix} \mathbf{a} \\ b \end{bmatrix} = \gamma \begin{bmatrix} \mathbf{\Phi}_S \\ \mathbf{1}^T \end{bmatrix} \mathbf{t}_S \quad (27)$$

where $\mathbf{a} = [a(i)]_{i\in G} \in \mathbb{R}^{|G|\times 1}$, $\mathbf{\Phi}_S = [\mathbf{\Phi}(\mathbf{x}(i))]_{i\in S} \in \mathbb{R}^{m\times|S|}$, $\mathbf{K} = [K_{ij}]_{i\in S, j\in G} = [\mathbf{k}_i]_{j\in G}^T \in \mathbb{R}^{|S|\times|G|}$, $\mathbf{1} = [1, \cdots, 1]^T \in \mathbb{R}^{|S|\times 1}$, and $\mathbf{t}_S = [t(i)]_{i\in S} \in \mathbb{R}^{|S|\times 1}$. It is sufficient[1] that

$$[\mathbf{K}\ \mathbf{1}] \begin{bmatrix} \mathbf{a} \\ b \end{bmatrix} = \gamma\mathbf{t}_S \quad (28)$$

hence

$$\begin{bmatrix} \mathbf{a} \\ b \end{bmatrix} = \gamma\,[\mathbf{K}\ \mathbf{1}]^+\,\mathbf{t}_S. \quad (29)$$

Since G is the set containing the support vectors, it is sufficient that $G \subseteq S$. We may use two options:

(a) $G = S$ or

(b) fix an upper limit to the cardinality $|G|$ and form $G \subset S$ by picking at most $|G|$ elements out of S.

This leads to the kernel learning rule summarized below.

Algorithm 2 (Kernel Minimum MSS)

Step 0: Select initial random values $\mathbf{a}_0$, b_0*;*
Step 1: Let $k = 0$*;*
Step 2: Find the set S *using* (24)*;*
Select a subset G *of* S *using either option (a) or (b)*
Step 3: Update $\mathbf{a}$ *and* b *as*

$$\begin{bmatrix} \mathbf{a}_{k+1} \\ b_{k+1} \end{bmatrix} = \gamma\,[\mathbf{K}\ \mathbf{1}]^+\,\mathbf{t}_S \quad (30)$$

Step 4: Set $k \leftarrow k + 1$*;*
Step 5: If terminating condition is met then STOP
else goto Step 2;

■

[1] although not necessary unless $\begin{bmatrix} \mathbf{\Phi}_S \\ \mathbf{1}^T \end{bmatrix}$ has full column rank

Table 1. Adult Results slack min

Kernel	$\|G\|$	Iterations	Time elapsed training (sec)	Accur. (Train set), (Test set)	Precision (Train set), (Test set)	Recall (Train set), (Test set)
linear	-	6	1.058	84.2%, 83.7%	74.4%, 75.4%	51.9%, 51.0%
poly	250	5	98.199	85.1%, 84.1%	74.8%, 74.9%	56.7%, 54.3%

Table 2. Adult Results SVMlight

Kernel	Time elapsed training (sec)	Accuracy (Train set), (Test set)	Precision (Train set), (Test set)	Recall (Train set), (Test set)
linear	8.3333	83.69%, 83.65%	75.70%, 74.48%	49.34%, 48.27%
poly	69.0340	84.63%, 84.18%	75.53%, 74.54%	53.65%, 51.60%

4. SIMULATIONS AND COMPARISON

In our experiments we used the "Adult" data set [9] available at the UCI machine learning repository. The data set is a selection of fields from the 1994 United States census data. It is among the larger datasets, having more than 32,561 training examples. The task is to predict if the income of a person is greater than 50K based on several census parameters, such as age, education, marital status, and so forth. Experimental results using the slack minimization algorithm with a linear kernel as well as with a polynomial kernel having an offset equal to 1 and a power equal to 2 is depicted in Table 1. For comparison purposes we resort to readily available software packages for SVMs, namely SVMlight [1] and libsvm [2]. The results are shown in Table 2 and Table 3, respectively. All experiments were performed on a 2.67 MHz processor with 4GB of RAM, with a Windows-7 32 bit operating system. Finally, the convergence of the algorithm is demonstrated in Figure 1 with the plots of the misclassification error as a function of the iterations (on the left for the linear kernel, on the right for the polynomial kernel). We see that the algorithm typically converges in less than 5 iterations.

Table 3. Adult Results LIBSVM

Kernel	Time elapsed training (sec)	Accuracy (Train set), (Test set)	Precision (Train set), (Test set)	Recall (Train set), (Test set)
linear	14.7736	84.33%, 83.60%	76.09%, 73.79%	51.03%, 48.91%
poly	69.0340	83.65%, 83.308%	75.63%, 74.207%	47.48%, 46.474%

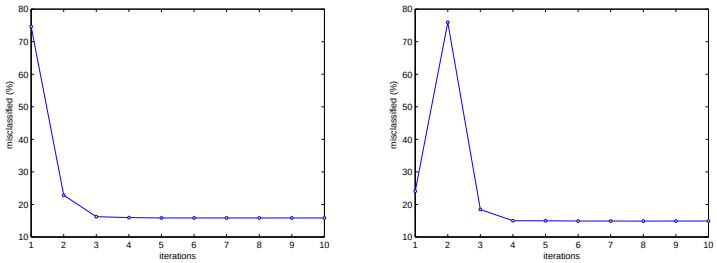

Fig. 1. The misclassification error as a function of the iterations for the slack minimization algorithm. Left: linear kernel; Right: polynomial kernel.

5. CONCLUSIONS

We have presented a method for solving the binary classification problem through the minimization of the energy of the slack variables. The method involves the solution of a linear system per iteration and it converges in a few iterations. We have presented preliminary mathematical results studying the minimizers of MSS and showing that if the problem can be linearly solved the algorithm will find a solution. The experiments demonstrate computational advantage over classical methods, in the linear case, while the performance is comparable. Due to lack of space only one set of experiments has been shown. In the full paper, we shall present a more thorough treatment of the algorithm convergence including more simulation results and comparisons with classical models in various data sets.

6. REFERENCES

[1] T. Joachims, *Making large-Scale SVM Learning Practical*, MIT Press, 1999.

[2] Chih-Chung Chang and Chih-Jen Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 1–27, 2011, Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[3] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proc. 5th annual ACM Workshop on Computational Learning Theory*, 1992, pp. 144–152.

[4] V. Vapnik, *Estimation of dependences based on empirical data*, Springer-Verlag, 1982.

[5] V. Vapnik, "Pattern recognition using generalized portrait method," *Automation and Remote Control*, vol. 24, pp. 774–780, 1963.

[6] T. Hofmann, B. Schölkopf, and A. J. Smola, "Kernel methods in machine learning," *Annals of Statistics*, vol. 36, no. 3, pp. 1171–1220, 2008.

[7] E. Y. Chang, K. Zhu, H. Wang, H. Bai, J. Li, Z. Qiu, and H. Cui, "Psvm: Parallelizing support vector machines on distributed computers," in *Proc. 21st Annual Conference on Neural Information Processing Systems*, 2007, pp. 144–152.

[8] V. Vapnik, *The nature of statistical learning theory*, Springer, NY, USA, 1995.

[9] A. Frank and A. Asuncion, "UCI machine learning repository," 2010.