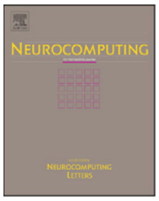
Neurocomputing 77 (2012) 253–260

Contents lists available at SciVerse ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom



Letters

Using Sequential Unconstrained Minimization Techniques to simplify SVM solvers

Sachindra Joshi [a,*], Jayadeva [b], Ganesh Ramakrishnan [c], Suresh Chandra [d]

[a] IBM Research - India, 4C, Vasant Kunj, New Delhi, India
[b] Department of Electrical Engineering, Indian Institute of Technology, Delhi, Hauz Khas, New Delhi, India
[c] Department of Computer Science and Engineering, Indian Institute of Technology, Bombay, Powai, Mumbai, India
[d] Department of Mathematics, Indian Institute of Technology, Delhi, Hauz Khas, New Delhi, India

ARTICLE INFO

Article history:
Received 16 May 2011
Received in revised form
28 June 2011
Accepted 3 July 2011
Communicated by R.W. Newcomb
Available online 10 September 2011

Keywords:
Support vector machines
SVM
SMO

ABSTRACT

In this paper, we apply Sequential Unconstrained Minimization Techniques (SUMTs) to the classical formulations of both the classical L1 norm SVM and the least squares SVM. We show that each can be solved as a sequence of unconstrained optimization problems with only box constraints. We propose relaxed SVM and relaxed LSSVM formulations that correspond to a single problem in the corresponding SUMT sequence. We also propose a SMO like algorithm to solve the relaxed formulations that works by updating individual Lagrange multipliers. The methods yield comparable or better results on large benchmark datasets than classical SVM and LSSVM formulations, at substantially higher speeds.

1. Introduction

In recent years, support vector machines (SVMs) have emerged as a powerful paradigm for pattern classification and regression [18,1–3]. The classical maximum margin SVM classifier aims to minimize an upper bound on the generalization error through maximizing the L1 norm margin between two disjoint half planes [18,2]. A popular alternative formulation is due to Suykens and Vandewalle who proposed the "least squares SVM" (LSSVM) [16]. The LSSVM employs equality constraints.

Both the SVM and the LSSVM formulations require the solution of a constrained quadratic optimization problem. When the sample set size is large, the computational and memory costs of solving the constrained Quadratic Programming Problem (QPP) can be prohibitive. Over the past decade, a lot of efforts have focused on solving these QPP efficiently. In the case of SVM, decomposition based methods that solve a series of subproblems of smaller size have been proposed [13,12]. Further extensions to these methods that focus on how to choose a good subproblem in each iteration so that maximum progress towards the optimal solution is made have also been proposed [14,10,15,8,7]. For the LSSVM formulation, Suykens and Vandewalle showed a more efficient approach for the LSSVM that involves solving two sets of linear equations [16]. SMO type update algorithms for LSSVM have also been proposed [19,9].

In this paper, we examine the use of Sequential Unconstrained Minimization Techniques (SUMTs) [4] for solving SVM as well as LSSVM optimization problems. Solving a constrained optimization problem C with SUMT involves solving a sequence of unconstrained optimization problems $\{U_1, U_2, \ldots, U_n\}$. At each step, an unconstrained optimization problem is obtained by adding a penalty term to the objective function of the constrained problem. The addition of a penalty term *favours* points from the feasible region. At any step, if the solution of U_i lies within the feasible region of the original constrained problem C, then we stop. Otherwise, a new but related unconstrained optimization problem is constructed, and the solution to earlier unconstrained problem is used as the initialization point.

In the case of SVM and LSSVM, we derive a sequence of box constrained problems. We refer to any of the box constrained optimization problems that need to be solved for the SVM as a relaxed SVM. Similarly, each of the optimization problems in the SUMT sequence for the LSSVM is referred to as a relaxed LSSVM. We show that solving the sequence of relaxed SVM and relaxed LSSVM problems leads to the solution of the original SVM and LSSVM, respectively.

Interestingly, solving only one of the relaxed SVM or relaxed LSSVM problems, instead of the entire sequence, yields comparable or better solutions as compared to the classical SVM and LSSVM, respectively. We derive an update rule for the relaxed

* Corresponding author.
E-mail addresses: jsachind@in.ibm.com (S. Joshi), jayadeva@ee.iitd.ac.in (Jayadeva), ganramkr@cse.iitb.ac.in (G. Ramakrishnan), chandras@maths.iitd.ac.in (S. Chandra).

0925-2312/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2011.07.010

versions, termed as 1SMO in the sequel, that allows a single variable to be updated at a time. The standard SMO uses a nested loop to update pairs of variables [13]. Empirically, we find that the relaxed SVM and relaxed LSSVM, solved by using the 1SMO algorithm, provide identical or superior generalization performance as compared to the classical SVM and LSSVM, respectively, but are 2–4 times faster.

The relaxed SVM problem that we propose is similar to the formulation proposed in [17]. However, we additionally show that solving a sequence of different but related relaxed SVM problems, lead to solving the classical SVM problem. The relaxed LSSVM formulation that we propose is entirely new. In a nutshell, the specific contributions made in this paper are three fold. First we provide a SUMT based framework for solving both, classical SVM as well as LSSVM problems. We show that the classical formulations of both, which are linearly constrained optimization problems with box constraints, can be solved as a sequence of box constrained optimization problems. Second, we propose relaxed SVM and relaxed LSSVM formulations that correspond to a single problem in the corresponding SUMT sequence. Third, we propose the 1SMO algorithm to solve relaxed SVM and relaxed LSSVM formulations that works by updating individual Lagrange multipliers. Relaxed SVM and relaxed LSSVM formulations, solved with 1SMO, yield comparable or better results than their classical counterparts, at substantially higher speeds.

The remainder of the paper is organized as follows. In Section 2, we describe the SUMT framework and discuss the classical SVM and LSSVM formulations. Section 3 discusses the solution of SVM and LSSVM formulations by posing them as sequences of unconstrained sub-problems under the SUMT framework. We also propose relaxed SVM and relaxed LSSVM formulations in this section. In Section 4 we provide an efficient algorithm to solve the relaxed SVM and relaxed LSSVM formulations. In Section 5, we discuss SUMT based reformulations and provide their primal versions. In Section 6 we describe the experimental set up and results. Section 7 is devoted to concluding remarks and scope for future work.

2. Background material

2.1. Sequential Unconstrained Minimization Techniques (SUMT)

Given an optimization problem of the form

$$\min f(x) \tag{1}$$

subject to constraints

$$h_j(x)=0, \quad j=1,2,\ldots,L \tag{2}$$

the solution to (1)–(2) may be found by solving a sequence of unconstrained optimization problems [4] of the form

$$\min E_p(x)=f(x)+\alpha_p \sum_{j=1}^{L} h_j^2(x) \tag{3}$$

A procedure to achieve this may be summarized as follows:

1. Set $p=0$. Choose the value of the coefficient α_0, and an initial state x_0.
2. Find the minimum of $E_p(x)$. Denote the solution as x^{p^*}.
3. If all the constraints in (2) are satisfied, stop.
4. If not, choose x^{p^*} as the new initial state, and choose α_{p+1} such that $\alpha_{p+1}>\alpha_p$. Set $p=p+1$. Go to step 2.
5. In the limit, as $p\to\infty$, the sequence of minima $x^{1^*}, x^{2^*}, \ldots x^{p^*}$, …, will converge to the solution of the original problem (1)–(2).

For the sake of brevity, we skip the mathematical details and conditions under which convergence is guaranteed. The interested reader is referred to [4]. Suffice it to say that in the case of the SVM and the LSSVM, we are dealing with convex programming problems with equality constraints and the SUMT approach can be applied.

2.2. The classical SVM

Given a set of M data points $\{x^1,x^2,\ldots,x^M\}$, where $x^i\in\mathcal{R}^N$, and their associated class labels $y_i\in\{-1,1\}$, the classical maximum soft margin SVM classifier aims to find a hyperplane of the form $h_+ = w^T\phi(x)+b=0$ such that data points with different class labels lie on opposite sides of the hyperplane. The parameters w and b of the hyperplane are found by solving the following quadratic optimization problem:

$$\min_{q,w} \tfrac{1}{2}w^Tw+Ce^Tq \tag{4}$$

subject to the constraints

$$y_k[w^T\phi(x^k)+b]\geq 1-q_k$$

$$q_k\geq 0, \quad k=1,2,\ldots,M \tag{5}$$

where e is a vector of ones of dimension M, and q is a vector of error variables. In practice, the dual formulation is solved, which is given by

$$\min_{\lambda} \tfrac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M} y_iy_j\lambda_i\lambda_jK_{ij}-\sum_{i=1}^{M}\lambda_i \tag{6}$$

subject to the linear equality constraint

$$\sum_{i=1}^{M}\lambda_iy_i=0 \tag{7}$$

and bound/box constraints on the dual variables

$$0\leq\lambda_i\leq C, \quad i=1,2,\ldots,M \tag{8}$$

Here, λ_i, $i=1,2,\ldots,M$ denote the Lagrange multipliers, and the matrix K with entries $K_{ij}=[\phi(x^i)^T\phi(x^j)]$ is termed as the Kernel matrix.

Platt's Sequential Minimal Optimization (SMO) algorithm [13] is an efficient way to solve (8). It begins by assigning all multiplier values to zero, and iteratively updates two Lagrange multipliers at a time, in a manner that constraints (7) and (8) are always satisfied.

2.3. The classical LSSVM

Suykens and Vandewalle proposed the least squares SVM which solves the following QPP:

$$\min_{q,w} \tfrac{1}{2}w^Tw+Cq^Tq \tag{9}$$

subject to the constraints

$$y_i[w^T\phi(x^i)+b]+q_i=1, \quad i=1,2,\ldots M \tag{10}$$

where $C>0$ is a parameter. The first term on the R.H.S. of (9) is a regularization one, whereas the second term is the empirical error. The constant C determines the relative importance of the two. Writing the Karush–Kuhn–Tucker (KKT) necessary and sufficient optimality conditions and simplifying, Suykens and Vandewalle showed that the LSSVM classifier parameters w and b may be determined by solving the following linear system of equations:

$$\begin{bmatrix} 0 & -y^T \\ y & K+\frac{I}{C}\end{bmatrix}\begin{bmatrix} b \\ \lambda\end{bmatrix}=\begin{bmatrix} 0 \\ e\end{bmatrix} \tag{11}$$

where λ is the vector of Lagrange multipliers, e is a vector of M ones, I is an identity matrix of size $M\times M$, and K is the kernel matrix, whose entries are given by

$$K_{ij} = [\phi(x^i)]^T \phi(x^j), \quad i,j = 1,2,\ldots,M \tag{12}$$

As pointed out by Suykens and Vandewalle [16], the system of equations in (11) can be solved by iterative methods. However, the matrix on the L.H.S. of (11) is not positive definite. By using appropriate transformations such as preconditioning, the matrix may be transformed into a positive definite one so that iterative methods such as conjugate gradient or successive over-relaxation may be applied.

The Lagrangian of (9) subject to the constraints (10) is given by

$$L = \frac{1}{2} w^T w + \frac{C}{2} q^T q - \sum_{k=1}^{M} \lambda_k [1 - q_k - y_k [w^T(\phi(x^k) + b]] \tag{13}$$

The KKT optimality conditions can be given as

$$\nabla_w L = 0 \Rightarrow w = \sum_{k=1}^{M} \lambda_k y_k \phi(x^k) \tag{14}$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{k=1}^{M} \lambda_k y_k = 0 \tag{15}$$

$$\frac{\partial L}{\partial q_k} = 0 \Rightarrow q_k = \frac{\lambda_k}{C} \tag{16}$$

$$y_k [w^T \phi(x^k) + b] = 1 - q_k = 1 - \frac{\lambda_k}{C} \tag{17}$$

The dual formulation is obtained by maximizing L, which on simplification yields

$$\min_{\lambda} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{M} y_i y_j \lambda_i \lambda_j P_{ij} - \sum_{i=1}^{M} \lambda_i \tag{18}$$

subject to the constraints

$$\sum_{i=1}^{M} \lambda_i y_i = 0 \tag{19}$$

where

$$P_{ij} = \left\{ \begin{array}{ll} K_{ij}, & i \neq j \\ K_{ii} + \frac{1}{C}, & i = j \end{array} \right\} \tag{20}$$

3. Solving SVM and LSSVM formulations using SUMT

3.1. SVM using SUMT and relaxed-SVM

To apply SUMT to the classical SVM, we consider the optimization problem (6) subject to constraints (7)–(8). We note that our updates always ensure that constraints (8) are satisfied, hence these are not considered separately; the feasible region being convex, we are ensured of convergence to the global optimum. The SUMT based procedure outlined in Section 2.1 indicates that we need to solve a sequence of minimization problems $(p = 1,2,\ldots)$ of the form

$$\begin{aligned} &\min_{\lambda} \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} y_i y_j \lambda_i \lambda_j K_{ij} - \sum_{i=1}^{M} \lambda_i + \alpha_p \left| \sum_{i=1}^{M} y_i \lambda_i \right|^2 \\ &= \min_{\lambda} \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} y_i y_j \lambda_i \lambda_j (K_{ij} + \alpha_p) - \sum_{i=1}^{M} \lambda_i \end{aligned} \tag{21}$$

Note that the sequence of minima of (21), $(p = 1,2,\ldots)$, subject to (8), converge to the solution for the classical SVM formulation, in the limit, as $p \to \infty$.

The relaxed SVM solves only a single problem from the sequence, i.e. it solves the optimization problem

$$\min_{\lambda} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{M} y_i y_j \lambda_i \lambda_j (K_{ij} + \alpha_p) - \sum_{i=1}^{M} \lambda_i \tag{22}$$

subject to the constraints

$$0 \leq \lambda_i \leq C, \quad i = 1,2,\ldots,M \tag{23}$$

The problem (22) subject to constraints (23) is a QPP with only box constraints, and without the usual linear constraint. In the sequel, we see that it is easier to solve than the conventional SVM formulation. We subsequently develop an efficient algorithm, termed as 1SMO, for determining the Lagrange multipliers.

3.2. LSSVM using SUMT and relaxed-LSSVM

The SUMT based procedure outlined in Section 2.1 indicates that we need to solve a sequence of minimization problems $(p = 1,2,\ldots)$ of the form

$$\begin{aligned} &\min_{\lambda} \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} y_i y_j \lambda_i \lambda_j P_{ij} - \sum_{i=1}^{M} \lambda_i + \alpha_p \left| \sum_{i=1}^{M} y_i \lambda_i \right|^2 \\ &= \min_{\lambda} \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} y_i y_j \lambda_i \lambda_j (P_{ij} + \alpha_p) - \sum_{i=1}^{M} \lambda_i \end{aligned} \tag{24}$$

Note that the sequence of minima of (24), $(p = 1,2,\ldots)$, subject to (19), yields the solution to the classical LSSVM formulation in the limit $p \to \infty$.

The relaxed LSSVM solves only one problem in the sequence, i.e. it minimizes the objective function

$$\frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{M} y_i y_j \lambda_i \lambda_j (P_{ij} + \alpha_p) - \sum_{i=1}^{M} \lambda_i \tag{25}$$

Since the q_i's are unconstrained, the box constraints are absent in this case. The problem (25) is a QPP without the usual linear constraint.

4. Solving relaxed-SVM and relaxed-LSSVM

We proposed the relaxed-SVM (RSVM) and relaxed-LSSVM (RLSSVM) formulations in Section 2.1. The duals of the RSVM and RLSSVM are very similar to each other and can be written in a unified way as follows:

$$\min_{\lambda} \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} y_i y_j \lambda_i \lambda_j (Q_{ij}) - \sum_{i=1}^{M} \lambda_i \tag{26}$$

where $Q_{ij} = K_{ij} + \alpha_p$ in case of RSVM and $Q_{ij} = P_{ij} + \alpha_p$ in case of RLSSVM, where P_{ij} is given by (20). Box constraints are present in the case of the RSVM only. Since the linear constraint of the form (7) is absent from these formulations, we propose to update one multiplier at a time. To this end, we develop the 1SMO algorithm.

4.1. The 1SMO algorithm

We consider the following optimization problem:

$$\frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} y_i y_j \lambda_i \lambda_j (Q_{ij}) - \sum_{i=1}^{M} \lambda_i \tag{27}$$

Note that the box constraints

$$0 \leq \lambda_i \leq C, \quad i = 1,2,\ldots,M \tag{28}$$

where present are considered during the update of the multipliers. Without loss of generality, let λ_1 be the multiplier that is to be updated. The objective function in (27) may be rewritten as a function of λ_1 only as

$$Q(\lambda_1) = -\lambda_1 - \sum_{j=2}^{M} \lambda_j + \lambda_1 y_1 \sum_{j=2}^{M} \lambda_j y_j (Q_{1j}) + \frac{1}{2}\lambda_1^2 (Q_{11}) + \frac{1}{2}\sum_{i=2}^{M}\sum_{j=2}^{M} y_i y_j \lambda_i \lambda_j (Q_{ij}) \tag{29}$$

We assume Q to be symmetric and use $y_1^2 = 1$ for (29). For the new value of λ_1 to lie at an extremal point of $Q(\lambda_1)$, we have $\partial Q/\partial\lambda_1 = 0$:

$$\Rightarrow 1 - \lambda_1^{new}(Q_{11}) - y_1 \sum_{j=2}^{M} \lambda_j y_j (K_{1j}) = 0 \tag{30}$$

For the extremal point to be a minimum we require

$$\frac{\partial^2 Q}{\partial \lambda_1^2} > 0 \Rightarrow (Q_{11}) > 0 \tag{31}$$

From (30) we obtain

$$\lambda_1^{new}(Q_{11}) = 1 - y_1 \sum_{j=2}^{M} \lambda_j y_j (Q_{1j}) \tag{32}$$

Defining $f(x) = w^T\phi(x) + b$, (32) can be written as

$$\lambda_1^{new}(Q_{11}) = \lambda_1^{old}(Q_{11}) + 1 - y_1 f^{old}(x^1) \tag{33}$$

which gives us the following update rule:

$$\lambda_k^{new} = \lambda_k^{old} + \frac{1 - y_k f^{old}(x^k)}{(Q_{kk})} \tag{34}$$

where we have written the rule for any k in place of 1. The box constraints (23) are considered during the update; if any multiplier crosses the bounds, it is held at the boundary value. The update rule (34) translates to

$$\lambda_k^{new} = \lambda_k^{old} + \frac{1 - y_k f^{old}(x^k)}{(K_{kk} + \alpha_p)} \tag{35}$$

in the case of the relaxed SVM, and to

$$\lambda_k^{new} = \lambda_k^{old} + \frac{1 - y_k f^{old}(x^k)}{(P_{kk} + \alpha_p)} \tag{36}$$

in the case of the relaxed LSSVM, where P_{kk} is given by (20). Since the update rule updates one multiplier at a time, updating pairs of multipliers as in the case of the SMO [13] is no longer necessary. This leads to the 1SMO-RSVM algorithm for relaxed SVM and 1SMO-RLSSVM algorithm for relaxed LSSVM, which are summarized in Figs. 1 and 2 respectively. These algorithms serially update all the multipliers until convergence is achieved.

In the following section, we discuss the primal formulations that lead to the relaxed duals discussed above. In deriving the dual formulations from their primal versions, we find that the use of KKT conditions can be used to make the multiplier updates more efficient, since multipliers that do not violate the KKT conditions do not need to be modified. In the process, we obtain a more efficient version of the 1SMO algorithm, that we employ in all of our experiments.

```
Procedure 1SMO_RSVM_for_the_Relaxed_SVM()
do
    iterate = false
        for each multiplier $\lambda_k$ do
            get $\lambda_k^{new}$ using (35)
            if $\lambda_k^{new} < 0$
                $\lambda_k^{new} = 0$
            if $\lambda_k^{new} > C$
                $\lambda_k^{new} = C$
            if ($|\lambda_k - \lambda_k^{new}| \geq \epsilon$)
                iterate = true
    done
while (iterate)
```

Fig. 1. The 1SMO-RSVM algorithm for the relaxed SVM.

```
Procedure 1SMO_RLSSVM_for_the_Relaxed_LSSVM()
do
    iterate = false
        for each multiplier $\lambda_k$ do
            get $\lambda_k^{new}$ using (36)
            if ($|\lambda_k - \lambda_k^{new}| \geq \epsilon$)
                iterate = true
    done
while (iterate)
```

Fig. 2. The 1SMO-RLSSVM algorithm for the relaxed LSSVM.

5. Discussion on the SUMT-based reformulations

5.1. The relaxed SVM primal

We derived the relaxed SVM and the relaxed LSSVM by using the dual formulations. In this section, we obtain similar constructions starting from the primal formulations. Consider the following optimization problem in its primal form:

$$\min_{q,w} \frac{1}{2} w^T w + \frac{A}{2} b^2 + Ce^T q \tag{37}$$

subject to constraints

$$y_k[w^T\phi(x^k) + b] \geq 1 - q_k$$

$$q_k \geq 0, \quad k = 1, 2, \ldots, M \tag{38}$$

This differs from the classical SVM formulation because of addition of $(A/2)b^2$ in the objective function. The constraints are the same as those in the classical SVM formulation. Here, A is a constant whose choice we discuss in the sequel. Note that for $A=1$, we obtain the formulation of [11], which is also related to the Kernel Adatron [5]. The Lagrangian for the problem (37)–(38) is given by

$$L = \frac{1}{2}(w^T w + Ab^2) + Ce^T q - \sum_{k=1}^{M} \beta_k y_k - \sum_{k=1}^{M} \lambda_k [1 - q_k - y_k[w^T\phi(x^k) + b]] \tag{39}$$

The KKT optimality conditions are given by

$$\nabla_w L = 0 \Rightarrow w = \sum_{k=1}^{M} \lambda_k y_k \phi(x^k) \tag{40}$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow b = \frac{1}{A}\sum_{k=1}^{M} \lambda_k y_k \tag{41}$$

$$\frac{\partial L}{\partial q_k} = 0 \Rightarrow C - \lambda_k - \beta_k = 0 \Rightarrow \lambda_k + \beta_k = C \tag{42}$$

From (41) and (42), we observe that

$$w^T\phi(x) + b = \sum_{k=1}^{M} \lambda_k y_k \left[K(x^k, x) + \frac{1}{A}\right] \tag{43}$$

where the kernel K is defined in the usual manner.

With a little algebra, we obtain the dual problem as

$$\min_{\lambda} \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{M} y_i y_j \lambda_i \lambda_j \left(K_{ij} + \frac{1}{A}\right) - \sum_{i=1}^{M} \lambda_i \tag{44}$$

subject to the constraints

$$0 \le \lambda_i \le C, \quad i = 1,2,\ldots,M \tag{45}$$

Observe that (44)–(45) is identical to the relaxed SVM formulation of (22)–(23), with $\alpha_p = 1/2A$. It is tempting to relate the dual formulations to a conventional SVM by considering the limit as $A \to \infty$. However, we see from the earlier discussion on SUMTs that this limit must be obtained by considering the sequence, and cannot be obtained simply by setting $1/2A$ to zero.

5.2. The relaxed LSSVM primal

Consider the optimization problem

$$\min_{q,w} \frac{1}{2} w^T w + \frac{A}{2} b^2 + C q^T q \tag{46}$$

subject to the constraints

$$y_i[w^T \phi(x^i) + b] + q_i = 1, \quad i = 1,2,\ldots M \tag{47}$$

The Lagrangian for the problem (46)–(47) is given by

$$L = \frac{1}{2}(w^T w + Ab^2) + C q^T q - \sum_{k=1}^{M} \lambda_k [1 - q_k - y_k [w^T \phi(x^k) + b]] \tag{48}$$

The KKT optimality conditions are given by

$$\nabla_w L = 0 \Rightarrow w = \sum_{k=1}^{M} \lambda_k y_k \phi(x^k) \tag{49}$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow b = \frac{1}{A} \sum_{k=1}^{M} \lambda_k y_k \tag{50}$$

$$\frac{\partial L}{\partial q_k} = 0 \Rightarrow q_k = \frac{\lambda_k}{C} \tag{51}$$

From (50) and (51), we observe that

$$w^T \phi(x) + b = \sum_{k=1}^{M} \lambda_k y_k \left[K(x^k, x) + \frac{1}{A} + \frac{1}{C} \right] \tag{52}$$

where the kernel K is defined in the usual manner.

On simplifying, we obtain the dual problem as

$$\frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{M} y_i y_j \lambda_i \lambda_j \left(P_{ij} + \frac{1}{A} \right) - \sum_{i=1}^{M} \lambda_i \tag{53}$$

where P_{ij} is given by (20).

Observe that (53) is identical to the relaxed SVM formulation of (25), with $\alpha_p = 1/2A$. Once again, we note that the limiting case when $1/A$ tends to zero cannot be obtained in a single step, and the sequence of minima must be considered in order to reach the minimum of the original LSSVM solution. In the sequel, we illustrate this through a simple experiment.

6. Experimental evaluation

6.1. Experimental setup

In order to evaluate the effectiveness of the proposed methods, we conducted four different experiments. In our first experiment we observe the effect of the parameter "A" on the performance of the 1SMO algorithm. In the second experiment, we compare the performance of 1SMO-RSVM with two state-of-the art implementations of SVM, viz. LibSVM and *SVM^{Light}*, on 13 benchmark datasets from the UCI repository.[1] In the third experiment, we compare the performance of 1SMO-RLSSVM and LSSVM on the same 13 benchmark datasets. In the fourth and the final experiment we examine how solutions to classical SVM and LSSVM problems can be obtained using the SUMT based methods.

All our algorithms were implemented in C++. The experiments were performed on a dual 3.2 GHz Xeon server with 4 GB RAM. We used the RBF kernel in all our experiments, with the value of the exponent (gamma) set to 1. The value of the slack parameter C was also chosen to be 1. Unless otherwise mentioned, in all our experiments, the kernel entries were computed on a need basis and cached for further use. All results are reported by following the standard 10-fold cross-validation methodology.

6.2. The effect of A on the performance of 1SMO

In order to understand the effect of the parameter A, we varied it from 1 to 10^4 and observed its effect on the performance of 1SMO-RSVM as well as on 1SMO-RLSSVM. While accuracy is unaffected by changes in the value of A, the training time reduces drastically with increasing values of A. Fig. 3 shows how training time for 1SMO-RSVM varies as a function of log(A) for the mushroom and kr-vs-kp datasets. The y-coordinate has been normalized with respect to the largest training time for each dataset, i.e. by dividing the training time with that for $A = 1$. The plot indicates that the rate of decrease of training time varies inversely with the value of A; the lower the value of A, the greater is the rate at which the training time decreases. The training time saturates beyond a sufficiently large value of A (10^4). This behavior may be understood from (35). A larger value of A corresponds to a larger step size, and the algorithm converges faster, leading to a lower value of training time. The rate of change of the step size is larger for smaller values of A. This explains why the curve has a much larger slope for lower values of A. Therefore, a sufficiently large value of A is a prudent choice for 1SMO-RSVM. We observe similar behavior for 1SMO-RLSSVM.

6.3. Comparison between 1SMO-RSVM, LibSVM and SVM^{Light}

We next conducted a set of experiments on 15 different two-class datasets from the UCI repository. The datasets were picked to cover a wide range of number of features and instances. The number of instances varied from 57 to 8124 and the number of features varied from 9 to 126.

The first column of Table 1 presents the number of instances and features for each dataset in the corresponding order as comma-separated values along with the name of the dataset. The table indicates the training times, accuracy, and number of support vectors yielded by 1SMO, LibSVM and SVMLight on each

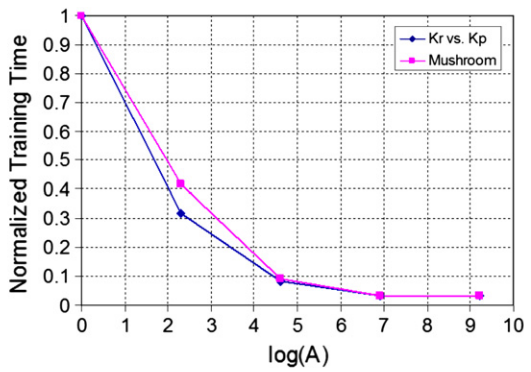


Fig. 3. Plot of variation of training time with increasing value of log(A) for the mushroom and kr-vs-kp datasets. The training times are averaged over 10 folds.

[1] http://archive.ics.uci.edu/ml/datasets.html

Table 1
Comparison between training times for 1SMO-RSVM, LibSVM and SVMLight. In nearly all cases, the three algorithms find solutions with the same number of support vectors, and show the same generalization performance.

Dataset	Method	Training time	# S. V.	ACC ± std
Breast-Cancer (286,51)	LibSVM	0.015	215	67.02 ± 0.98
	SVM^{Light}	0.016	215	67.02 ± 0.98
	1SMO	**0.006**	215	72.09 ± 1.32
Breast-W (699,10)	LibSVM	0.051	353	83.87 ± 1.12
	SVM^{Light}	0.054	353	83.87 ± 1.12
	1SMO	**0.021**	353	93.25 ± 0.8
Credit-G (1000,64)	LibSVM	0.389	783	69.78 ± 0.76
	SVM^{Light}	0.274	783	69.78 ± 0.76
	1SMO	**0.157**	783	69.78 ± 0.76
Heart-C (302,23)	LibSVM	0.03	240	54.72 ± 1.16
	SVM^{Light}	0.028	240	54.72 ± 1.16
	1SMO	**0.013**	240	54.72 ± 1.16
Heart-H (294,25)	LibSVM	0.028	233	66.35 ± 0.85
	SVM^{Light}	0.026	233	66.35 ± 0.85
	1SMO	**0.011**	233	66.35 ± 0.85
Heart-Statlog (270,14)	LibSVM	0.023	214	58.24 ± 2.42
	SVM^{Light}	0.019	214	58.24 ± 2.42
	1SMO	**0.008**	214	58.24 ± 2.42
Hepatitis (155,30)	LibSVM	0.009	123	79.81 ± 1.9
	SVM^{Light}	0.008	123	79.81 ± 1.9
	1SMO	**0.003**	123	79.81 ± 1.9
Ionosphere (350,35)	LibSVM	0.021	167	93.14 ± 0.95
	SVM^{Light}	0.029	167	93.14 ± 0.95
	1SMO	**0.011**	189	93.05 ± 0.93
Kr Vs Kp (3196,41)	LibSVM	3.48	2488	96.52 ± 0.28
	SVM^{Light}	2.76	2488	96.52 ± 0.28
	1SMO	**1.06**	2488	97.05 ± 0.19
Mushroom (8124,126)	LibSVM	19.62	6340	100 ± 0
	SVM^{Light}	18.27	6340	100 ± 0
	1SMO	**9.85**	6340	100 ± 0
Pima-Indian (768,9)	LibSVM	0.17	601	65.96 ± 0.95
	SVM^{Light}	0.14	601	65.96 ± 0.95
	1SMO	**0.07**	601	65.96 ± 0.95
Sick (3772,33)	LibSVM	4.94	2890	93.85 ± 0.17
	SVM^{Light}	3.93	2890	93.85 ± 0.17
	1SMO	**1.97**	2890	93.85 ± 0.17
Sonar (208,61)	LibSVM	0.011	134	78.55 ± 1.48
	SVM^{Light}	0.016	134	78.55 ± 1.48
	1SMO	**0.005**	134	78.35 ± 1.74

of the 15 datasets. Based on the results of the first experiment, we chose $A = 10^4$ for 1SMO-RSVM.

Table 1 summarizes the results. In nearly all cases, the three algorithms find solutions with the same number of support vectors, and show the same generalization performance. It can be observed that the training time of 1SMO-RSVM is consistently lower than the training time of *LibSVM* and *SVM*^{Light}. For the larger datasets, 1SMO-RSVM achieves higher speedup factors, roughly between 2 and 4. For each dataset, the lowest training time is marked in bold-face in Table 1. We note that all three algorithms converge to solutions with approximately the same number of support vectors, on all datasets. The accuracy of 1SMO-RSVM on every dataset is either equal to or slightly higher than the accuracies achieved by *LibSVM* and *SVM*^{Light}. 1SMO-RSVM therefore emerges as an attractive alternative to the widely used SMO approach for training SVMs.

The slight discrepancy between number of support vectors and accuracy could be attributed to the difference in the ways the value of b is computed in the case of the classical SVM and the relaxed SVM.

6.4. Comparison between 1SMO-LSSVM and LSSVM

We next compare the performance of 1SMO-LSSVM and LSSVM algorithms on the same 13 datasets. Table 2 shows the dataset, the method compared, training time, number of support vectors and accuracy achieved along with standard deviation in the respective columns. It is clear from the comparison as made in the table, that 1SMO-RLSSVM is 2–3 times faster than LSSVM.

6.5. Convergence of the SUMT solution

In our final experiment, we solve the classical SVM problem by solving a sequence of relaxed-SVM problems. Each of the relaxed-

Table 2
Comparison between training times for 1SMO-LSSVM and LSSVM. In nearly all cases, the two algorithms find solutions with the same number of support vectors, and show the same generalization performance.

Dataset	Method	Training time (s)	# S.V.	ACC ± std
Breast-Cancer (286,51)	LSSVM	0.023	215	67.02 ± 0.98
	1SMO	**0.009**	215	66.60 ± 1.14
Breast-W (699,10)	LSSVM	0.044	359	85.53 ± 1.10
	1SMO	**0.020**	357	91.19 ± 0.88
Credit-G (1000,64)	LSSVM	0.204	783	69.78 ± 0.76
	1SMO	**0.157**	783	69.78 ± 0.76
Heart-C (302,23)	LSSVM	0.019	240	54.72 ± 1.16
	1SMO	**0.012**	240	54.72 ± 1.16
Heart-H (294,25)	LSSVM	0.017	233	66.35 ± 0.85
	1SMO	**0.011**	233	66.35 ± 0.85
Heart-Statlog (270,14)	LSSVM	0.04	165	79.45 ± 0.87
	1SMO	**0.03**	165	79.43 ± 0.96
Hepatitis (155,30)	LSSVM	0.009	113	77.86 ± 1.52
	1SMO	**0.001**	113	81.45 ± 1.54
Ionosphere (350,35)	LSSVM	0.056	230	93.12 ± 1.24
	1SMO	**0.017**	230	94.82 ± 0.88
Kr Vs Kp (3196,41)	LSSVM	0.905	1582	89.86 ± 1.67
	1SMO	**0.450**	1582	92.62 ± 1.11
Mushroom (8124,126)	LSSVM	7.05	4030	99.98 ± 0.02
	1SMO	**4.27**	4030	100 ± 0
Pima-Indian (768,9)	LSSVM	0.1	601	65.96 ± 0.95
	1SMO	**0.073**	601	65.96 ± 0.95
Sick (3772,33)	LSSVM	3.13	2890	93.85 ± 0.17
	1SMO	**2.09**	2890	93.85 ± 0.17
Sonar (208,61)	LSSVM	0.018	146	83.61 ± 1.49
	1SMO	**0.007**	146	82.50 ± 1.61

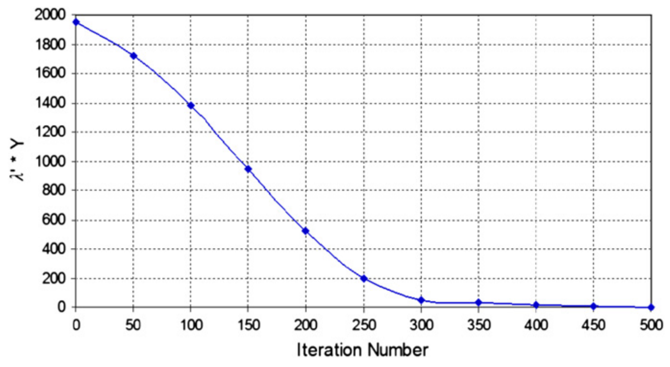


Fig. 4. Plot of variation of $\lambda^T y$ with increasing iteration numbers for the SUMT based algorithm on the sick dataset.

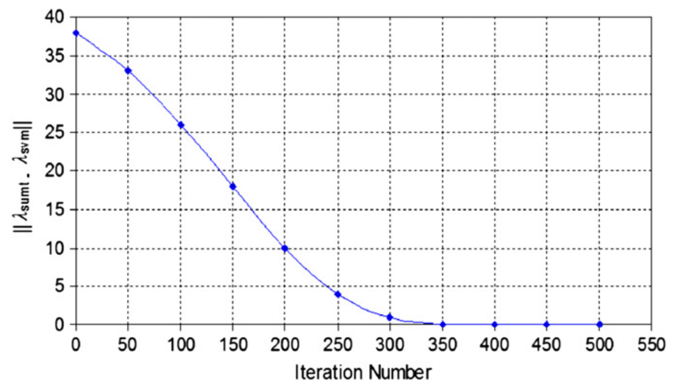


Fig. 5. Plot of $\|\lambda_{SUMT} - \lambda_{SVM}\|$ vs. iteration number for the SUMT based algorithm on the sick dataset.

SVM problem is solved using the 1SMO-RSVM algorithm. We use the *sick* dataset from the UCI repository. We initialize A with a value of 10^4 and successively reduce A by a factor of $\rho = 0.9$. We observe the value of $\lambda^T y$ after each iteration. Fig. 4 shows how $\lambda^T y$ reduces as a function of the number of SUMT-iterations. As expected, the value of $\lambda^T y$ decreases with successive iterations and quickly converges to 0.

Fig. 5 shows how the Lagrange multipliers for 1SMO-RSVM converge to the solution obtained by SVM^{Light}, as iterations progress. The plot demonstrates that the sequence of relaxed SVM sub-problems converges to the solution of the classical SVM. We observe similar behaviour on all the datasets for relaxed-SVM as well as for relaxed-LSSVM problems.

7. Concluding remarks

In this paper, we used Sequential Unconstrained Minimization Techniques (SUMTs) to show that the classical SVM and LSSVM formulations can be solved through a sequence of unconstrained optimization problems using the dual formulations. We showed that solving an appropriate problem in that sequence can yield competitive solutions with large reductions in training times. This led us to alternative approaches termed as the relaxed SVM and the relaxed LSSVM methods. Both the relaxed-SVM as well as relaxed-LSSVM formulations turn out to be variants of the original SVM and LSSVM formulation in the primal. Some of these variants have been reported in the literature in the past, albeit not through the framework introduced in this paper. The relaxed-SVM formulation is related to the kernel adatron machine [17]. The relaxed-LSSVM problem has similarities with proximal SVM [6]. There are many other flavours of SVM, that use other notions of class separability. Relaxed versions of these offer similar advantages. Additionally, kernel optimization can be speeded up by using relaxed variants.

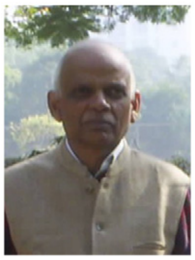
References

[1] P.S. Bradley, O.L. Mangasarian, Massive data discrimination via linear support vector machines, Optim. Methods Software 13 (1) (2000) 1–10.
[2] C. Burges, A tutorial on support vector machines for pattern recognition, Data Min. Knowl. Discovery 2 (2) (1998) 121–167.
[3] N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines and Other Kernel based Learning Methods, Cambridge University Press, 2000.
[4] A.V. Fiacco, G.P. McCormick, Nonlinear Programming: Sequential Unconstrained Minimization Techniques, Wiley and Sons, New York, 1968.
[5] T.T. Friess, N. Cristianini, C. Campbell, The kernel adatron algorithm: a fast and simple learning procedure for support vector machines, in: Proceedings of the Fifteenth International Conference on Machine Learning (ICML), 1998, pp. 188–196.
[6] G. Fung, O.L. Mangasarian, Proximal support vector machine classifiers, in: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '01, ACM, New York, NY, USA, 2001, pp. 77–86.
[7] T. Glasmachers, C. Igel, Maximum-gain working set selection for SVMs, J. Mach. Learn. Res. 7 (December) (2006) 1437–1466.
[8] T. Joachims, Making large-scale support vector machine learning practical, in: B. Schölkopf, C. Burges, A. Smola (Eds.), Advances in Kernel Methods—Support Vector Learning, MIT Press, Cambridge, MA, USA, 1999, pp. 168–184.
[9] S.S. Keerthi, S.K. Shevade, SMO algorithm for least-squares SVM formulations, Neural Comput. 15 (2) (2003) 487–507.
[10] S.S. Keerthi, S.K. Shevade, C. Bhattacharyya, K.R.K. Murthy, Improvements to Platt's SMO algorithm for SVM classifier design, Neural Comput. 13 (3) (2001) 637–649.
[11] O.L. Mangasarian, D.R. Musicant, Successive overrelaxation for support vector machines, IEEE Trans. Neural Networks 10 (5) (1998) 1032–1037.
[12] E. Osuna, R. Freund, F. Girosi, An improved training algorithm for support vector machines, Proceedings of the 1997 IEEE Workshop on Neural Networks for Signal Processing (NNSP), Amelia Island, FL, USA, vol. VII, 1997, pp. 276–285.
[13] J.C. Platt, Fast training of support vector machines using sequential minimal optimization, in: B. Schölkopf, C. Burges, A. Smola (Eds.), Advances in Kernel Methods—Support Vector Learning, MIT Press, Cambridge, MA, USA, 1999, pp. 185–208.
[14] J.C. Platt, Using analytic QP and sparseness to speed training of support vector machines, Proceedings of the 1998 Conference on Advances in Neural Information Processing Systems, vol. II, MIT Press, Cambridge, MA, USA, 1999, pp. 557–563.
[15] S.K. Shevade, S.S. Keerthi, C. Bhattacharyya, K.R.K. Murthy, Improvements to the SMO algorithm for SVM regression, IEEE Trans. Neural Networks 11 (5) (2000) 1188–1194.
[16] J.A.K. Suykens, J. Vandewalle, Least squares support vector machine classifiers, Neural Process. Lett. 9 (3) (1999) 293–300.
[17] M. Vogt, V. Kecman, T.-M. Huang, Iterative single data algorithm for training kernel machines from huge data sets: theory and performance, Support Vector Machines: Theory and Applications, vol. 177, Springer Verlag, 2005, pp. 255–274.
[18] V. Vapnik, Statistical Learning Theory, Wiley, 1998.
[19] X. Zeng, X.-W. Chen, SMO-based pruning methods for sparse least squares support vector machines, IEEE Trans. Neural Networks 16 (6) (2005) 1541–1546.

Sachindra Joshi is a research staff member in IBM Research at New Delhi since 2000. His research interests lie in the general area of machine learning, statistical relational learning, text mining and natural language processing. He has published over 25 papers in the refereed journals and international conferences and also has over 20 patents filed in these and related areas. He has successfully applied machine learning techniques in the design and development of systems aimed at utilizing unstructured information in business processes at IBM. Prior to joining IBM research, Sachindra completed Masters in Computer Science and Engineering from Indian Institute of Technology Bombay, where he was the recipient of the gold medal award for ranking first in the department. He is also pursuing Ph.D. at IIT Delhi since 2007.

Jayadeva obtained his B.Tech and Ph.D. degrees from the Department of Electrical Engineering, IIT Delhi. He is currently a Professor in the same department. His research interests include Machine Learning, Optimization, and VLSI. He has been a speaker on the IEEE Computer Society Distinguished Visitor Programme, a recipient of the Young Engineer Award from the Indian National Academy of Engineering, the Young Scientist Award from the Indian National Science Academy, and the BOYSCAST Fellowship from the Department of Science and Technology, Government of India. He was a URSI Young Scientist at the General Assembly in Lille, France (1996), and received the Sir J.C. Bose Young Scientist title from the Indian Council of the URSI. He visited the Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology in 1997 as a BOYSCAST fellow. He spent a sabbatical year as a visiting Researcher at IBM India Research Laboratory, from July 2006. One of his papers in Neurocomputing was listed on the Top25 hotlist; he is a recipient of best paper awards from the IETE Journal of Research, and three other conference papers. He holds a US Patent on A/D conversion. His research interests include Machine Learning, Optimization, and VLSI. He has served on the Steering and Program Committees of several international conferences. He is the co-author of a book with Profs. Suresh Chandra and Aparna Mehra, entitled "Numerical Optimization with Applications".

Ganesh Ramakrishnan is an Assistant Professor at the Computer Science and Engineering Department at IIT Bombay. His main areas of interest are Relational Learning, Feature Induction and Text Mining. He has published over 40 papers in leading international conferences and journals including ICML, WWW, KDD, CIKM, etc. Currently, he is co-authoring a book "Handbook of Relational Learning" to be published by CRC Press, USA. Ganesh did his B.Tech. (in 2000) and Ph.D. (in 2005) at the Department of Computer Science and Engineering, IIT Bombay.



Suresh Chandra received the M.S. degree in Mathematical Statistics from Lucknow University, and his Ph.D. from the Indian Institute of Technology, Kanpur. He is currently a Professor at the Department of Mathematics, Indian Institute of Technology, Delhi, India. He has authored and co-authored more than 100 publications in refereed journals and international conferences and coauthored two books, one on Fuzzy Mathematical Programming and Fuzzy Matrix Games and the other on Principles of Optimization Theory. His research interests include numerical optimization, mathematical programming, generalized convexity, fuzzy optimization, fuzzy games, neural networks, machine learning and financial mathematics. He is a Member of the Editorial Board for the International Journal of Management and Systems, and the Journal of Decision Sciences. Also he is a Senior Member of the Operational Research Society of India, and a Member of the International Working Group on Generalized Convexity and Applications.