Fast Update Rules for a Relaxed SVM Formulation

Abstract

The Sequential Minimal Optimization algorithm and its variants are well known techniques for the fast training of Support Vector Machines. SMO solves a linearly constrained quadratic programming problem by updating pairs of Lagrange multipliers. We propose a variant of the classical SVM formulation, which we term as the Relaxed SVM. We develop a learning algorithm, termed as 1SMO in the sequel, that allows individual Lagrange multipliers in the dual formulation to be updated, and is linearly convergent. On selected benchmark datasets, the Relaxed SVM trained with 1SMO is 2-3 times faster than LibSVM and SVMLight, while comparing very favourably in terms of error rate. On larger datasets, where the kernel cannot be cached, the speedup is even higher. We next extend the proposed 1SMO algorithm to the solution of the classical SVM, through a sequence of Relaxed SVM sub-problems.

Keywords: Support Vector Machines, Sequential Minimal Optimization, Machine Learning, Classification, Function Approximation.

1. Introduction

Over the last decade or more, Support Vector Machines (SVMs) have emerged as a popular and powerful paradigm for pattern classification and function approximation (Vapnik, 1998; Cristianini & Shawe-Taylor, 2000; Bradley & Mangasarian, 2000; Burges, 1998). SVMs emerged from research in statistical learning theory on how to regulate generalization in learning, and the tradeoff between structural complexity and empirical risk. SVM classifiers assign data samples to one of two half-planes, either in the pattern space or in a higher-dimensional feature space. One of the most popular SVM classifiers is the "maximum margin" one, that aims to minimize an upper bound on the generalization error through maximizing the margin between two disjoint half planes (Vapnik, 1998; Burges, 1998). The best known algorithm for training SVMs is Platt's SMO (1998), that updates pairs of Lagrange multipliers at a time to find a solution to the dual formulation.

In this paper, we propose a modification of the classical SVM formulation, termed as the *Relaxed SVM* in the sequel. We derive an update rule for determining the Lagrange multipliers in the dual formulation that allows multipliers to be updated individually, instead of in pairs. The proposed update rule, termed as one-SMO (1SMO) is simple, faster, more scaleable, and linearly convergent. At the same time, the generalization performance provided by the Relaxed SVM is the same or better than that of the classical SVM. We also show that the solution to the classical SVM can be obtained as the limit of a sequence of Relaxed SVM sub-problems.

Consider a training set consisting of M patterns $x^1, x^2, \ldots, x^M$, where $x^i = \left(x_1^i, x_2^i, ..., x_N^i\right)^T$ is a point in $\Re^N$. The class label of the i-th pattern is denoted by $y_i \in \{-1, 1\}$. Non-linearly separable problems are often solved by mapping the input data samples x^i to a higher dimensional feature space $\phi(x^i)$. The classical maximum margin SVM classifier aims to find a hyperplane of the form

$$w^T \phi(x) + b = 0,$$

that separates patterns of the two classes. The variables w and b are determined by solving the optimization problem

$$\underset{q,w}{\text{Minimize}} \quad \frac{1}{2} w^T w + \mathrm{C} e^T q \tag{1}$$

subject to the constraints

$$\begin{aligned} & y_k \left[w^T \phi\left(x^k\right) + b \right] \geq 1 - q_k, \\ & q_k \geq 0, \ k = 1, 2, ..., M, \end{aligned} \tag{2}$$

where e is a vector of ones of dimension M. The solution to (1)-(2) yields the soft margin classifier, so termed because the distance or margin between the separating hyperplane $w^T\phi(x) + b = 0$ and the image of the k-th pattern x^k may be reduced from 1 by an amount q_k . The solution to (1)-(2) is usually determined by considering the dual problem, which is given by

$$\underset{\lambda'}{\text{Minimize}} \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} y_i y_j \lambda_i \lambda_j K_{ij} - \sum_{\mathrm{i}=1}^{\mathrm{M}} \lambda_i, \tag{3}$$

subject to the linear equality constraint

$$\sum_{i=1}^{M} \lambda_i y_i = 0, \tag{4}$$

and bound/box constraints on the dual variables

$$0 \leq \lambda_i \leq C, \ i = 1, 2, ..., M. \tag{5}$$

Appearing in *Proceedings of the 24dthInternational Conference on Machine Learning*, Corvallis, OR, 2007.

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

When the sample set size is large, the computational and memory costs of solving the constrained Quadratic Programming Problem (QPP) can be prohibitive. Therefore, a lot of effort has focused on how to efficiently solve (3)-(5). Osuna et. al (1997) showed that the QPP could be solved through the solution of a series of subproblems of a smaller size. Platt's SMO (1998) is a special type of decomposition method for SVMs, that solves a problem of size two at each step. Improvements to the basic SMO have been suggested in (Platt, 1999; Keerthi et. al, 2001; Shevade et. al, 2000; Joachims, 1998; Chang & Lin, 2001; Glasmachers and Igel, 2006), which largely focus on how to choose a good working set (pair of multipliers) at each stage, so that maximum progress towards the optimal solution is made. Proofs of convergence of decomposition methods may be found in Lin (2001). A rigorous proof of SMO convergence was provided in (Takahashi & Nishi, 2005). Active set methods (Vogt & Kecman, 2005) have also been proposed for the efficient computation of the Lagrange multipliers. The recent seminal work of Joachims (2006) in training linear SVMs in linear time must be pointed out, though our present investigation is more relevant to nonlinear kernels.

The proposed Relaxed SVM solves the optimization problem

$$\underset{q,w}{\text{Minimize}} \quad \frac{1}{2} w^T w + \frac{A}{2} b^2 + \mathrm{C} e^T q \tag{6}$$

subject to constraints (5). This formulation is related to approaches such as the Kernel Adatron, among others (Friess et. al, 1998; Navone & Down, 2001; Mangasarian & Musicant, 1999); however, all such prior work may be shown to correspond to the special case A = 1. In the sequel, we show that in fact this is a very poor choice of the value of A.

The dual formulation for the Relaxed SVM has no linear equality constraint, but only box constraints. We use this to derive an update rule, termed in the sequel as one-SMO (1SMO), that allows individual multipliers to be updated, without having to consider pairs. The results of this paper show that choosing large values of A yields substantial improvements in training time. On several selected benchmark datasets from the UCI Machine Learning repository, the proposed 1SMO update rule converges 2-3 times faster than state-of-the-art SMO implementations such as SVM$^{\text{Light}}$ (Joachims, 1999) and LibSVM (Chang & Lin, 2001). We also show that on very large datasets, where the kernel cannot be pre-computed and cached, the 1SMO update rule offers larger speedups, since the number of kernel computation calls is reduced in view of the fewer iterations required to compute the multipliers. On a well known dataset with 8000 points, the speedup is by a factor of 5. In all cases, the performance of the Relaxed SVM is either better than, or the same as that of SVM$^{\text{Light}}$ and LIBSVM.

The classical SVM corresponds to the case when A = 0, but cannot be solved by merely solving the limiting case of (6). In Section 3, we show how a 1SMO type algorithm can be obtained for the classical SVM by solving a sequence of related relaxed SVM sub-problems. This approach to solving the classical SVM may have advantages when dealing with very large datasets, where the kernel cannot be cached.

The remainder of this paper is organized as follows. In Section 2, we describe the Relaxed SVM and derive the 1SMO update rule for determining the Lagrange multipliers in the dual formulation. In Section 3, we show how the Relaxed SVM is related to the classical SVM formulation. We also describe how the solution to a classical SVM may be determined by solving a sequence of Relaxed SVM sub-problems. Section 4 is devoted to experimental results. Section 5 contains concluding remarks.

2. The Relaxed SVM formulation

2.1 The Classical SVM and the SMO

The Karush-Kuhn-Tucker optimality conditions for the classical SVM formulation (1)-(2) may be obtained in the usual way, and require that for a solution to be feasible, the multipliers λ_i must satisfy (4), (5), and meet the following requirements.

$$\lambda_i = 0 \Rightarrow y_i\left[w^T \phi\left(x^i\right) + b\right] \geq 1 \tag{7}$$

$$\lambda_i = C \Rightarrow y_i\left[w^T \phi\left(x^i\right) + b\right] \leq 1 \tag{8}$$

$$0 \leq \lambda_i \leq C \Rightarrow y_i\left[w^T \phi\left(x^i\right) + b\right] = 1 \tag{9}$$

Platt's SMO solves a series of subproblems of size 2 by updating two multipliers, say λ_1 and λ_2 such that constraints (4) and (5) are always satisfied. Note that the updates are done so that the state λ always lies in the feasible region. SMO and its variants use a hierarchy of heuristics to choose the multipliers to be updated at each step. The main requirement is that all pairs of multipliers are chosen repeatedly. It is also assumed that for any pair (k, l), the kernel matrix satisfies the requirement

$K_{kk} + K_{ll} - 2K_{kl} > 0$. This condition is satisfied by any positive definite kernel.

2.2 The Relaxed SVM

The relaxed SVM is obtained by solving the following optimization problem

$$\underset{q,w}{\text{Minimize}} \quad \frac{1}{2} w^T w + \frac{A}{2} b^2 + \mathrm{C} e^T q \tag{10}$$

subject to the constraints

$$\begin{aligned} & y_k \left[w^T \phi\left(x^k\right) + b \right] \geq 1 - q_k, \\ & q_k \geq 0, \ k = 1, 2, ..., M. \end{aligned} \tag{11}$$

Here, A is a constant whose choice we will discuss at length in the sequel. Note that for $A = 1$, we obtain the formulation of (Mangasarian & Musicant, 1999), which is also related to the Kernel Adatron. The Lagrangian for the problem (10)-(11) is given by

$$\begin{aligned} L = \frac{1}{2}\left(w^T w + A b^2\right) + C e^T q - \sum_{k=1}^{M} \beta_k y_k - \\ \sum_{k=1}^{M} \lambda_k \left[1 - q_k - y_k \left[w^T \phi\left(x^k\right) + b \right] \right]. \end{aligned} \tag{12}$$

The K.K.T. optimality conditions are given by

$$\nabla_w L = 0 \Rightarrow w = \sum_{k=1}^{M} \lambda_k y_k \phi\left(x^k\right). \tag{13}$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow b = \frac{1}{A} \sum_{k=1}^{M} \lambda_k y_k. \tag{14}$$

$$\frac{\partial L}{\partial q_k} = 0 \Rightarrow C - \lambda_k - \beta_k = 0 \Rightarrow \lambda_k + \beta_k = C. \tag{15}$$

From (13) and (14), we observe that

$$w^T \phi(x) + b = \sum_{k=1}^{M} \lambda_k y_k \left[K\left(x^k, x\right) + \frac{1}{A} \right], \tag{16}$$

where the kernel K is defined in the usual manner. The dual problem minimizes the objective function

$$\frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} y_i y_j \lambda_i \lambda_j \left(K_{ij} + \frac{1}{A} \right) - \sum_{i=1}^{M} \lambda_i \tag{17}$$

subject to the constraints (5). Note that the problem (17) subject to constraints (5) is a QPP with only bound constraints and without the usual linear constraint. It is tempting to relate (10) to a conventional SVM by considering the limit as $A \to 0$. However, we defer this discussion to the sequel. We will first develop the 1SMO update algorithm for the relaxed SVM.

2.2 Updating the Mutlipliers in a Relaxed SVM

Since the linear constraint of the form (4) that couples the multipliers is absent, we propose to update one multiplier at a time. Without loss of generality, let λ_1 be the multiplier being updated. The objective function in (17) may be rewritten as a function of λ_1 only, as

$$\begin{aligned} Q(\lambda_1) = -\lambda_1 - \sum_{j=2}^{M} \lambda_j + \lambda_1 y_1 \sum_{j=2}^{M} \lambda_j y_j \left(K_{1j} + \frac{1}{A} \right) + \\ \frac{1}{2} \lambda_1^2 \left(K_{11} + \frac{1}{A} \right) + \frac{1}{2} \sum_{i=2}^{M} \sum_{j=2}^{M} y_i y_j \lambda_i \lambda_j \left(K_{ij} + \frac{1}{A} \right), \end{aligned} \tag{18}$$

assuming K to be symmetric, and since $y_1^2 = 1$. For the new value of λ_1 to lie at an extremal point of $Q(\lambda_1)$, we have $\frac{\partial Q}{\partial \lambda_1} = 0$,

$$\Rightarrow 1 - \lambda_1^{new} \left(K_{11} + \frac{1}{A} \right) - y_1 \sum_{j=2}^{M} \lambda_j y_j \left(K_{1j} + \frac{1}{A} \right) = 0. \tag{19}$$

For the extremal point to be a minimum, we require

$$\frac{\partial^2 Q}{\partial \lambda_1^2} > 0 \Rightarrow \left(K_{11} + \frac{1}{A} \right) > 0. \tag{20}$$

Note that this condition may be satisfied by matrices K that are not necessarily positive definite, as required in the case of SMO type update algorithms. From (19), we obtain

$$\lambda_1^{new} \left(K_{11} + \frac{1}{A} \right) = 1 - y_1 \sum_{j=2}^{M} \lambda_j y_j \left(K_{1j} + \frac{1}{A} \right). \tag{21}$$

Defining $f(x) \equiv w^T \phi(x) + b$, (21) may be written as

$$\lambda_1^{new} \left(K_{11} + \frac{1}{A} \right) = \lambda_1^{old} \left(K_{11} + \frac{1}{A} \right) + 1 - y_1 f^{old}\left(x^1\right). \tag{22}$$

which gives us the update rule

$$\lambda_k^{new} = \lambda_k^{old} + \frac{1 - y_k f^{old}\left(x^k\right)}{\left(K_{kk} + \frac{1}{A} \right)}. \tag{23}$$

where we have written the rule for any λ_k in place of λ_1. Since the update rule updates one multiplier at a time, updating pairs of multipliers is no longer necessary. Figure 1 describes a possible update algorithm for determining the Lagrange multipliers. The algorithm is a co-ordinate descent method (CDM), and is linearly convergent, which follows from the work of Luo and Tseng (1993).

1. 1.For each multiplier λ_k, update λ_k by using (23). If $\lambda_k < 0$, set λ_k to 0. If $\lambda_k > C$, set $\lambda_k = C$.
2. If the values of all multipliers are unchanged (within a specified tolerance ε), Stop; Otherwise, go to Step 1.

Figure 1: A simple update algorithm (CDM)

As mentioned before, the special case $A = 1$ corresponds to the Kernel Adatron and the SOR based approach to its solution (Mangasarian & Musicant, 1999). Experimental

results in Section V indicate that this choice of A is almost always a bad one. In the sequel, we also show how the classical SVM can be related to the relaxed SVM.

The update algorithm described in Fig.1 serially updates all multipliers. On a large dataset, this leads to unnecessary computation for many multipliers that will not change at all. A better update algorithm may therefore be obtained by using information about the primal to choose which multipliers to update. This leads to the 1SMO update algorithm, which is summarized in Fig. 2.

The update rule (23) is attractive from many viewpoints. The new value of $f(x^i)$, denoted by $f^{new}(x^i)$, may be computed by computing the incremental change due to the update of multiplier λ_k, which depends only on K_{ik}. The denominator for each k can be computed in advance and stored, avoiding one division per multiplier per iteration. Advantages in terms of a distributed or parallel implementation using $O(M)$ nodes may be a topic of future research.

1. Pick a multiplier λ_k that violates (7), (8), or (9). Note that $f(x) \equiv w^T\phi(x)+b$ may be computed by using (16). If all multipliers satisfy the K.K.T. conditions, then the minimum has been attained - Stop. Othewise, proceed to Step 2.
2. Update λ_k by using (23). If $\lambda_k < 0$, set λ_k to 0. If $\lambda_k > C$, set $\lambda_k = C$.
3. Go to Step 1.

Fig.ure 2: Outline of the 1SMO Algorithm

3. Extension to conventional SVM

We now discuss the connection between the classical SVM formulation and the relaxed SVM one. Given an optimization problem of the form

$$\text{Min} \quad f(x), \text{ subject to } h_j(x) = 0, \ j = 1, 2, \ldots, L, \qquad (24)$$

where $f(x)$ is convex and $h_j(x)$, $j = 1, 2, \ldots, L$, are linear, the solution to (24) may be determined using the theory of Sequential Unconstrained Minimization Techniques (SUMTs). One approach involves solving a sequence of optimization problems (Fiacco & McCormick, 1968) of the form

$$\text{Min} \quad E_p(x) = f(x) + \alpha_p \sum_{j=1}^{L} h_j^2(x), \qquad (25)$$

The procedure may be outlined as follows

1. Set p = 0. Choose the value of the co-efficient α_0, and an initial state x^0.
2. Find the minimum of E_p (denoted as x^{p^*}).
3. If the constraints 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 $\rightarrow \infty$, the sequence of minima x^{1^*}, x^{2^*}, … x^{p^*}, …, will converge to the solution of the original problem (24).

The above procedure, which is a restriction of Sequential Unconstrained Minimization Techniques to convex programming problems with equality constraints, allows us to extend the 1SMO algorithm to the classical SVM. To do this, we consider the optimization problem (3) subject to the constraints (4)-(5). We note that our updates always ensure that constraints (5) 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 above indicates that we need to solve a sequence of minimization problems (p = 1, 2, …) involving the following objective

$$\begin{aligned} &\frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M} y_i y_j \lambda_i \lambda_j K_{ij} - \sum_{i=1}^{M} \lambda_i + \alpha_p \left|\sum_{i=1}^{M} y_i \lambda_i\right|^2 \\ &= \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M} y_i y_j \lambda_i \lambda_j \left(K_{ij} + 2\alpha_p\right) - \sum_{i=1}^{M} \lambda_i \end{aligned} \qquad (26)$$

Note that the sequence of minima of (26) subject to (5) yields the solution to the classical SVM formulation (3)-(5) in the limit $p \rightarrow \infty$, in which case, we also see that $\alpha_p \rightarrow \infty$. The connection between the relaxed SVM and the classical SVM is now clear. Observe that the objective function of (26) is identical to that of the relaxed SVM formulation of (10), with $\alpha_p = 2/A$.

Therefore, the solution to the classical SVM cannot be obtained by setting A=0 in (10), but by solving a sequence of problems with diminishing values of A, and with $A \rightarrow 0$ in the limit. The overall procedure is outlined in Fig. 3.

1. Set A, and the factor ρ by which A will be changed in the sequence of sub-problems.
2. Pick a multiplier λ_k that violates (7), (8), or (9). If all multipliers satisfy the K.K.T. conditions, then the minimum has been attained. Go to Step 4.
3. Update λ_k by using (23). If $\lambda_k < 0$, set λ_k to 0. If $\lambda_k > C$, set $\lambda_k = C$.
4. If $\lambda^T y$ is sufficiently close to 0, Stop. Otherwise, go to Step 5.
5. Update A as $A \leftarrow \rho A$.
6. Go to Step 2.

Figure 3:. The 1SMO Algorithm for the Classical SVM

At this point, we remark that the offset b is given by (14) for any nonzero value of A. However, when A = 0, the

expression for b is a ratio of two quantities that are zero. This may be interpreted by noting that in the classical SVM, the value of b is indeterminate, since it may be determined by considering any of the support vectors, for which

$$y_i\left[w^T\phi\left(x^i\right)+b\right]=1 \Rightarrow w^T\phi\left(x^i\right)+b=y_i \\ \Rightarrow b=y_i-w^T\varphi\left(x^i\right) \quad (27)$$

The value of b may be determined from any support vector, or by averaging the values obtained for different support vectors. The classical SVM solution may therefore be treated as the limit of a sequence of relaxed SVMs.

4. Experimental Evaluation

4.1 Dataset and Experimental Setup

In order to evaluate the effectiveness of the proposed methods, we conducted five different experiments. In our first experiment, we compare the performance of the two candidate training algorithms, *viz.*, CDM and 1SMO. The second experiment was designed to observe the effect of the parameter `A' on the performance of the 1SMO algorithm outlined in Fig. 2. The third experiment compares the performance of 1SMO with two state-of-the art implementations of SVM, *viz.* LibSVM and SVM$^{\text{Light}}$, on 15 benchmark datasets. The fourth experiment was performed to study the scalability of 1SMO with respect to the number of training instances, in comparison with LibSVM and SVM$^{\text{Light}}$. The convergence of the SUMT based algorithm outlined in Fig. 3 is studied in our fifth and final experiment.

All our algorithms were implemented in C++. The experiments were performed on a dual 3.2GHz 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.

4.2 Experimental Results

4.2.1 Comparison between CDM and 1SMO

In our first experiment, we compared the performance of our CDM and 1SMO training algorithms on all the 15 datasets. We observed that 1SMO and CDM give the same accuracy and number of support vectors on all datasets. However, 1SMO is faster than CDM by 5-15% on all datasets – the difference being more pronounced for larger datasets. Table 1 reports the training time of the two algorithms on some of the larger datasets. This observation makes 1SMO a natural choice for all subsequent experiments. Throughout the remainder of the paper, the experiments therefore refer to the performance of the 1SMO algorithm.

Table 1: Comparison between the training times of the CDM and 1SMO algorithms on six large datasets. While the two algorithms give the same accuracies and number of support vectors, 1SMO is consistently faster than CDM..

	Training Time		
Dataset	(CDM)	(1SMO)	% Gain
Adult	913.85	840.65	08.01
Mushroom	11.976	9.850	17.75
Ionosphere	0.013	0.011	15.38
Kr-vs-kp	1.188	1.060	10.77
Pima-india	0.072	0.070	02.78

4.2.2 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, on each dataset. While accuracy is unaffected by changes in the value of A, the training time reduces drastically with increasing values of A. Figure 4 shows how training time 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

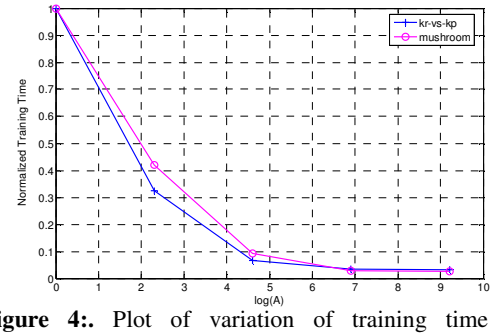

Figure 4:. Plot of variation of training time with increasing value of log(A) for the mushroom and *kr-vs-kp* datasets. The training times are averaged over 10 folds.

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 equation (23). 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.

4.2.3 Comparison between 1SMO, LibSVM and SVM^{Light}

We next conducted a set of experiments on fifteen 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 48789 and the number of features varied from 9 to 126. The first column of Table 2 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 SVM^{Light} on each of the 15 datasets. Based on the results of the first experiment, we chose $A = 10^4$ for 1SMO. Table 2 summarizes the results. The experiments for all datasets except the Adult dataset were conducted with kernel caching. In nearly all cases, the three algorithms find solutions with the same number of support vectors, and show the same generalization performance.

In the case of the adult dataset, the size of the required cache was over 4 GB. Since different SVM implementations might possibly use different cache replacement policies, kernel caching was avoided altogether in this specific case to allow for a fair comparison across all methods.

It can be observed that the training time of 1SMO is consistently lower than the training time of LibSVM and SVM^{Light}. For the larger datasets, 1SMO achieves higher speedup factors, roughly between 2 and 4. For each dataset, the lowest training time is marked in bold-face in Table 2. We note that all three algorithms converge to solutions with approximately the same number of support vectors, on all datasets. The accuracy of 1SMO on every dataset is either equal to or slightly higher than the accuracies achieved by LibSVM and SVM^{Light}. 1SMO 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.

4.2.4 Scalability of 1SMO

SVM training algorithms require the computation of elements of the kernel matrix, whose size is quadratic in the number of data-points. Most implementations typically start with all multipliers initialized to 0. However, many Lagrange multipliers remain zero and hence the kernel rows for the corresponding datapoints never need to be computed. Therefore, kernel entries are computed only if required. To avoid re-computation, kernel entries are cached as they are computed. For large datasets, caching all kernel entries could be prohibitive. One of the key bottlenecks in the scalability of SVM training algorithms is therefore the requirement of a

Table 2: Comparison between training times for 1SMO, LibSVM and SVM^{Light}. 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)	SUPPORT VECTORS	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	83.8± 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	69.7±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	54.7± 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	79.8±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
LABOR (57,30)	LibSVM	**0.001**	44	71.71± 3.95
	SVM^{Light}	0.003	44	71.71± 3.95
	1SMO	**0.001**	44	82.88± 5.33
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	78.54± 1.48
	1SMO	**0.005**	134	78.35± 1.74
ADULT (48789,106)	LibSVM	4552	38042	76.15± 0.10
	SVM^{Light}	3689	38042	76.15± 0.10
	1SMO	**1111**	38042	76.15± 0.10

sufficiently large kernel cache. Algorithms that can perform well on large datasets with relatively small kernel cache sizes could also be amenable to parallelization with lower communication overheads.

In order to compare the scalability of 1SMO with that of LibSVM and SVM^{Light}, we conducted an experiment on a large dataset *without* using any kernel caching. We used subsets of the Adult dataset, whose sizes were integral multiples of 1/12th of the total size. In the case of LibSVM and SVM^{Light} , the cache size was limited to 5MB, below which it was not possible to run them for the larger splits. In the case of 1SMO, caching was avoided altogether. Figure 5 shows the variation of the training time of 1SMO, LibSVM and SVM^{Light} with subset size for the a*dult* dataset. Least squares fits to the three curves are given by:

$t_{LIBSVM} = 1.7\text{x}10^{-6}x^2 + 1.04\text{x}10^{-2}x - 37.0575$
$t_{SVMLight} = 1.4\text{x}10^{-6}x^2 + 1.2\text{x}10^{-3}x - 17.7995$
$t_{1SMO} = 5\text{x}10^{-7}x^2 + 7.12\text{x}10^{-4}x - 7.8225$

In other words, with increase in the size of the dataset, the training time for 1SMO grows at a much slower rate than it does for SVM^{Light} and LibSVM, which grow quadratically with the number of training instances. This indicates that 1SMO scales better than LibSVM and SVM^{Light}.

4.2.5 Convergence of the SUMT solution

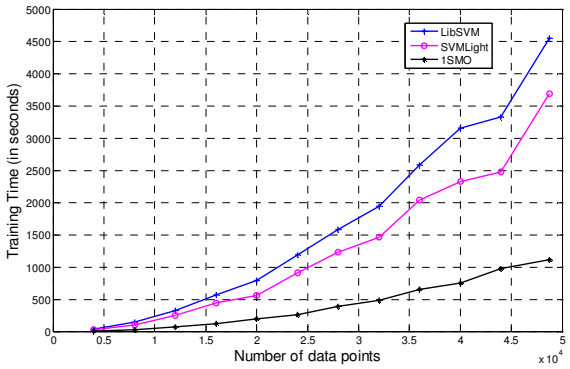

Figure 5: Variation of training time with for the three implementations, plotted as a function of number of data points sampled from the adult data set. While training times for SVM^{Light} and LibSVM grow almost quadratically with the size of the data sets, the training time for 1SMO grows at a much lower rate.

In our final experiment, we run the SUMT algorithm on two datasets, *viz.,* mushroom and breast-cancer. For each experiment, we initialize A with a value of 10^4 successively reduce A by a factor of ρ=*0.9*, as described in Fig, 3. We observe the value of $\lambda^T y$ after each SUMT-iteration (comprising steps 2-5 in Fig. 3). Figure 6 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.

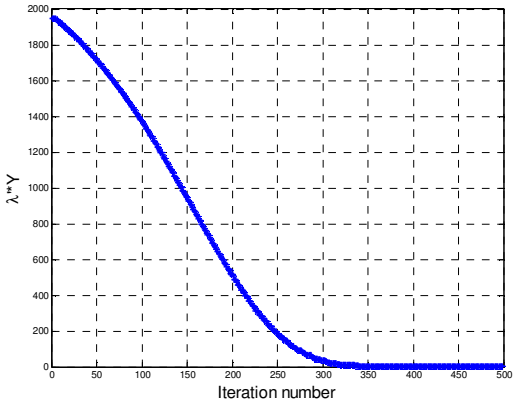

Figure 6: Plot of variation of $\lambda^T y$ with increasing iteration numbers for the SUMT based algorithm on the *sick* dataset.

Figure 7 shows how the Lagrange multipliers for 1SMO 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.

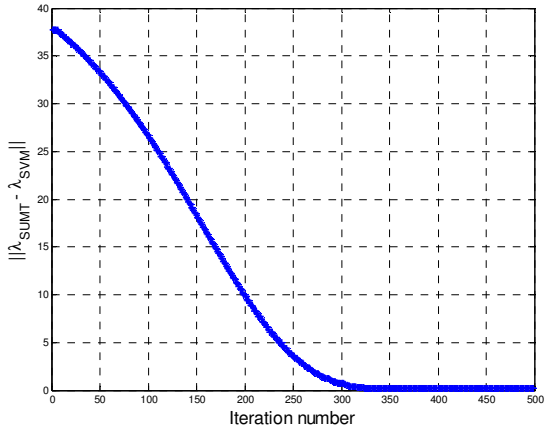

Figure 7: Plot of $\|\lambda_{SUMT} - \lambda_{1SVM}\|$ vs. iteration number for the SUMT based algorithm on the *sick* dataset.

5. Conclusions

In this paper, we have proposed a linearly convergent update rule for training a modified SVM formulation, which we term as the Relaxed SVM. On a selection of well known benchmark datasets, the Relaxed SVM with the proposed update rule yields speedups by a factor of 2-over well known SVM implementations, while providing very competitive generalization performance. The speedups on very large datasets, where caching the entire

kernel is infeasible, are even higher owing to the fewer iterations required to determine the Lagrange multipliers. We also show how the Relaxed SVM can be extended to solve the classical SVM formulation.

1SMO thus emerges as an attractive alternative to the widely used SMO algorithm for training SVMs. It also appears to be more amenable to a parallel implementation, which is worthy of further investigation for large datasets or embedded applications.

References

Bradley, P. S., & Mangasarian, O. L., (2000). Massive data discrimination via linear support vector machines, *Optimization Methods and Software*, *13*, (pp. 1-10).

Burges C. (1998). A tutorial on support vector machines for pattern recognition, *Data Mining and Knowledge Discovery*, *2*, (pp. 121-167).

Chang, C. C., & Lin, C. J. (2001). *LIBSVM: a library for support vector machines*, Online at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Cristianini, N., & Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and other kernel based learning methods*, Cambridge University Press.

Fiacco, A. V., & McCormick, G. P. (1968). *Nonlinear Programming: Sequential unconstrained minimization techniques*, Wiley and Sons, New York.

Glasmachers, T., & Igel, C. (2006). Maximum-Gain Working Set Selection for SVMs, *Journal of Machine Learning Research, **7***, (pp. 1437–1466).

Joachims, T. (1999). Making Large-Scale SVM Learning Practical, In Schölkopf, B., Burges, C., & Smola, A., (Eds.), *Advances in Kernel Methods - Support Vector Learning*, MIT-Press.

Joachims, T. (2006). Training Linear SVMs in Linear Time, *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*, ACM.

Friess, T. T., Cristianini, N. C., & Campbell, C. (1998). The kernel adatron algorithm: a fast and simple learning procedure for support vector machines, *Proceedings of 15th International Conference on Machine Learning*.

Keerthi, S. S., Shevade, S. K., Bhattacharyya, C. & Murthy, K. R. K. (2001). Improvements to Platt's SMO algorithm for SVM classifier design, *Neural Computation*, ***13***, (pp. 637-649).

Lin, C. J. (2001). On the convergence of the decomposition method for support vector machines, *IEEE Trans. Neural Netw.*, *12*, No. 6, (pp. 1288–1298).

Luo, Z. Q., & Tseng, P. (1993). Error bounds and convergence analysis of feasible descent methods: A general approach, *Annals of. Operations Research*, *46*, (pp. 157–178).

Mangasarian, O. L., & Musicant, D. R. (1999). Successive overrelaxation for support vector machines, *IEEE Transactions on Neural Networks*, *10,* No. 5, (pp. 1032-1037).

Navone, H. D., & Down, T. (2001). Variations on a Kernel-Adatron Theme, *Proceedings of the VII International Congress on Information Engineering*, (pp. 562-572).

Osuna, E., Freund, R., & Girosi, F. (1997). An improved training algorithm for support vector machines. *Proceedings of IEEE NNSP'97*, Amelia Island, Florida.

Platt, J. C., Fast training of support vector machines using sequential minimal optimization. *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1998.

Platt, J. (1999). Using sparseness and analytic QP to speed training of support vector machines. *Advances in Neural Information Processing Systems 11*. MIT Press.

Shevade, S. K., Keerthi, S. S., Bhattacharyya, C. & Murthy, K. R. K. (2001). "Improvements to the SMO algorithm for SVM regression", *IEEE Transactions on Neural Networks*, **11**, pp. 1188-1194, 2000.

Takahashi, N., & Nishi, T. (2005). Rigorous Proof of Termination of SMO Algorithm for Support Vector Machines, *IEEE Transactions on Neural Networks*, *16*, No.3, (pp.774-776).

Vogt, M., & Kecman, V. (2005). Active-Set Methods for Support Vector Machines, In Wang, L., (Ed.), Support Vector Machines: Theory and Applications, (pp. 133-158), *Studies in Fuzziness and Soft Computing, 177,* Springer-Verlag, Berlin, Heidelberg.

Vapnik, V. (1998). *Statistical Learning Theory*, Wiley.