Fast Algorithms for Least Squares SVMs

[1]Sachindra Joshi, [1]Jayadeva, [1]Ganesh Ramakrishnan, and [2]Suresh Chandra
[1]IBM India Research Laboratory
[2]Department of Mathematics, Indian Institute of Technology, Hauz Khas,
New Delhi - 110016. INDIA.

Abstract

Least Squares SVMs include the well known LSSVM and the Proximal SVM, which are normally trained by solving a system of linear equations, or through a matrix inversion. We examine a variant in which the objective function is similar to the Proximal SVM, while the constraints are those of LS-SVMs; we term this as a Relaxed Least Squares SVM (RLSSVM). The RLSSVM yields a simple dual formulation, for which we propose a fast sequential update algorithm; the update rules is about two to four times faster than conventional approaches on large datasets, while yielding similar error rates. Finally, we show that for a given LSSVM with any specified kernel matrix, there is an equivalent Relaxed SVM with the same solution. This opens up the possibility of developing alternate ways of solving the very popular LSSVM.

Keywords: Support Vector Machines, Classification, Regression, Function Approximation, Least Squares, Machine Learning, Classification, Numerical Methods, Linear Algebra

I. Introduction

The literature on neural networks is replete with powerful and efficient techniques for pattern classification and function approximation. The ideas on learning complexity, first investigated in the context of multilayer neural networks have now been encompassed in the area of Support Vector Machines (SVMs). SVMs have emerged in recent years as a powerful paradigm for pattern classification and regression [1-4]. SVMs emerged from research in statistical learning theory on how to regulate generalization in learning, and the tradeoff between structural complexity and empirical risk.

The classical maximum margin SVM classifier aims to minimize an upper bound on the generalization error through maximizing the margin between two disjoint half planes [1, 4]. This basically involves solving a quadratic programming problem that could be prohibitive on large data sets. To overcome this problem, Suykens and Vandewalle proposed the "least square SVM" (LSSVM) formulation [5]. The formulation considers equality constraints and adds an extra term to the cost function. As a result, the solution follows from directly solving a set of linear equations. The resulting system of equations is not positive definite, making it more difficult to solve. Some pre-conditioning is performed to the system of linear equations, so that more efficient numerical optimization methods could be applied. Therefore, the solution of the training procedure for LSSVM can be found by solving two sets of linear equations.

In this paper, we propose a new formulation for SVM called Relaxed LSSVM. The formulation is a variant of LSSVM and proximal SVM [6]. Solving relaxed LSSVM involves solving a system of linear equation that is guaranteed to be positive definite. Therefore, the solution of the training procedure for relaxed LSSVM can be obtained by solving a single system of linear equation in contrast to solving two systems of linear equations as required for LSSVM. This yields a speed up by a factor of 2-3 times over LSSVM on standard data sets.

Given a set of *M* patterns x^k, where $x^k = (x_1^k, x_2^k, \ldots, x_N^k)^T$, with corresponding labels $x^k \in \{-1, +1\}$, the LS-SVM determines a separating surface of the form $w^T\phi(x) + b = 0$ by solving a problem of the form

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

subject to

$$y_i\left[w^T\phi(P_i) + b\right] + q_i = 1, \; i = 1,2,\ldots,M \tag{2}$$

where $C > 0$ is a parameter.

Here, ϕ is a function that maps patterns from the input space into a higher dimensional feature space; q_k is the error variable associated with the *k*-th constraint. Proximal SVMs, proposed by Fung and Mangasarian in 2001 [6], minimize the objective function

$$\underset{q,b,w}{\text{Minimize}} \quad \frac{1}{2} w^T w + \frac{1}{2} b^2 + \frac{C}{2} q^T q \tag{3}$$

subject to the constraint

$$D\left(KK^T\right)Dw + eb + q = e. \tag{4}$$

Here, *e* is a vector of ones of appropriate dimension; *I* is the identity matrix; and *D* is a diagonal matrix whose entries are the class labels (± 1) of the patterns. The notation has been changed from that in [5], [6] to make it consistent with the rest of this paper. Note that the error variables q_i are not constrained to be non-negative, i.e. they can be of either sign. The solution to LS-SVM involves the solution of a system of linear equations, while proximal SVMs involve a matrix inversion.

The Relaxed LSSVM formulation has an objective function similar to the Proximal SVM and has LSSVM-type constraints. We derive simple update rules for each of them, which yield a speedup by factors of upto 2-3 times on a set of benchmark datasets, while providing similar error rates as the conventional formulations. While considerable research has been done on working set and decomposition methods for solving the classical L_1 norm SVM [7]-[15], there have been fewer, more recent attempts at developing SMO type fast update algorithms for Least Squares SVMs [16]-[17]. The update rules we propose in this work are motivated by recent work on the IDSA algorithm [18] for the L_1 norm SVM. Since the matrix in LSSVM is not positive definite, it has to be pre-conditioned [26]. On the other hand, Relaxed LSSVM yields a matrix in the quadratic objective function that is guaranteed to be positive definite. Relaxed SVM

therefore has the advantage of linear update rules, coupled with positive definite matrix. With respect to training time, we empirically show that Relaxed SVM scales much better with the size of the data set, when compared to Proximal SVM and LSSVM.

Finally, we show that the classical least squares SVM formulation (1)-(2) can be solved by solving a single Relaxed LSSVM with a modified kernel matrix. This opens up the route to alternative IDSA style algorithms for solving LS-SVMs.

The remainder of the paper is organized as follows. Section II discusses the LSSVM and Proximal SVM formulations. Section III is devoted to the Relaxed LSSVM formulation and an algorithm for its solution. Section IV deals with variants of the Relaxed LSSVM. Section V deals with experimental results. Section VI contains a discussion on how the Relaxed LSSVM is related with the classical LSSVM. Section VII contains concluding remarks. Appendix I presents an extension to the conventional LSSVM through a single unconstrained minimization.

II. Least Squares and Proximal SVMs

Suykens and Vandewalle proposed Least Squares SVMs (LS-SVMs) in 1998 [5], which solves the following optimization problem.

$$\underset{q,b,w}{\text{Minimize}} \quad \frac{1}{2} w^T w + \frac{C}{2} q^T q \tag{5}$$

subject to

$$y_i \left[w^T \phi(P_i) + b \right] + q_i = 1, \; i = 1,2,...,M \tag{6}$$

where $C > 0$ is a parameter. The first term on the R.H.S. of (9) is a for regularization, while 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 LS-SVM classifier parameters w and b may be determined by solving the following system of equations.

$$\begin{bmatrix} 0 & -y^T \\ y & K + \dfrac{I}{C} \end{bmatrix} \begin{bmatrix} b \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ e \end{bmatrix} \tag{7}$$

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

$$K_{ij} = \left[\phi(x^i) \right]^T \phi(x^i), \;\; i, j = 1, 2, ..., M. \tag{8}$$

As pointed out by Suykens and Vandewalle [5], the system of equations (7) can be solved by iterative methods. However, the matrix on the L.H.S. of (7) is not positive definite. By using appropriate transformations such as preconditioning (such as those given in [5], the system may be transformed into a positive definite one so that iterative methods such as conjugate gradient or successive over-relaxation may be applied.

Proximal SVMs, introduced by Fung and Mangasarian in 2001 [6], solve the following problem

$$\underset{q,b,w}{\text{Minimize}} \quad \frac{1}{2} w^T w + \frac{1}{2} b^2 + \frac{C}{2} q^T q \tag{9}$$

subject to the constraint

$$D\left(KK^T\right)Dw + eb + q = e. \tag{10}$$

Here, e is a vector of ones of appropriate dimension; I is the identity matrix; and D is a diagonal matrix whose entries are the class labels (± 1) of the patterns. Simplification of the K.K.T. conditions leads to

$$w = DK^T Dv, \; b = e^T Dv, \tag{11}$$

where

$$v = \left[\frac{I}{C} + D\left(KK^T + ee^T\right)D\right]^{-1} e. \tag{12}$$

The cost of solving a general system of linear equations or of inverting a matrix is the same, $O(M^3)$, where M is the size of the system or the order of the matrix. The aim of this paper is to suggest a more efficient route to training SVMs formulated in the least squares sense.

III. A Twist to Proximal SVMs

We first consider the problem

$$\underset{q,b,w}{\text{Minimize}} \quad \frac{1}{2} w^T w + \frac{A}{2} b^2 + \frac{C}{2} q^T q \tag{13}$$

subject to constraint (6). We refer to this problem as the Relaxed LSSVM; its objective function is in the spirit of the Proximal SVM when A = 1. However, note that for a general nonlinear kernel, the constraints are very **different** from those employed in Proximal SVMs, *viz.* (10). While the objective function is similar to Proximal SVMs, the constraints are those of LS-SVMs.

The Lagrangian for the problem (13) subject to constraints (6) is given by

$$L = \frac{1}{2}\left(w^T w + Ab^2\right) + \frac{C}{2} q^T q - \sum_{i=1}^{M} \lambda_k \left[1 - q_k - y_k \left[w^T \phi\left(x^k\right) + b\right]\right]. \tag{14}$$

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

$$\nabla_w L = 0 \Rightarrow w - \sum_{k=1}^{M} \lambda_k y_k \phi\left(x^k\right) = 0 \Rightarrow w = \sum_{k=1}^{M} \lambda_k y_k \phi\left(x^k\right). \tag{15}$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow Ab - \sum_{k=1}^{M} \lambda_k y_k = 0 \Rightarrow b = \frac{1}{A} \sum_{k=1}^{M} \lambda_k y_k. \tag{16}$$

$$\frac{\partial L}{\partial q_k} = 0 \Rightarrow Cq_k - \lambda_k = 0 \Rightarrow q_k = \frac{\lambda_k}{C}. \tag{17}$$

$$y_k \left[w^T \phi\left(x^k\right) + b\right] = 1 - q_k = 1 - \frac{\lambda_k}{C}. \tag{18}$$

From (15) and (16), we observe that

$$w^T\phi(x)+b=\sum_{k=1}^{M}\lambda_k y_k\left[\phi(x^k)\right]^T\phi(x)+\frac{1}{A}\sum_{k=1}^{M}\lambda_k y_k=\sum_{k=1}^{M}\lambda_k y_k\left[K(x^k,x)+\frac{1}{A}\right] \qquad (19)$$

where the kernel function K is defined in the usual manner.

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

$$\begin{aligned}&\text{Maximize } \sum_{i=1}^{M}\lambda_i-\frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M}Q_{ij}\lambda_i\lambda_j,\\ &\qquad\text{or equivalently,}\\ &\text{Minimize } \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M}Q_{ij}\lambda_i\lambda_j-\sum_{i=1}^{M}\lambda_i,\end{aligned} \qquad (20)$$

where

$$Q_{ij}=\begin{cases}K_{ij}y_i y_j+\dfrac{1}{A}, & i\neq j\\ K_{ii}+\dfrac{1}{A}+\dfrac{1}{C}, & i=j\end{cases}. \qquad (21)$$

Note that we have an *unconstrained* Quadratic Programming Problem, without even box constraints. For any positive definite kernel K, the matrix Q is guaranteed to be positive definite. The additional term $(1/C)$ in the diagonal entries of Q are expected to also contribute to faster convergence as compared to when K alone is used.

We now derive a sequential minimization procedure for determining the Lagrange multipliers λ_i, i.e. by updating one multiplier at a time. Without loss of generality, let λ_1 be the multiplier being updated. The objective function in (20) may be rewritten as a function of λ_1 only, as

$$E(\lambda_1)=\lambda_1+\sum_{j=2}^{M}\lambda_j-\frac{1}{2}\lambda_1^2Q_{11}-\lambda_1\sum_{j=2}^{M}\lambda_jQ_{1j}-\frac{1}{2}\sum_{i=2}^{M}\sum_{j=2}^{M}\lambda_i\lambda_jQ_{ij}, \qquad (22)$$

where we have assumed that Q is symmetric. For the new value of λ_1 to lie at an extremal point of $Q(\lambda_1)$, we have

$$\frac{\partial E}{\partial\lambda_1}=0\Rightarrow 1-\lambda_1^{new}Q_{11}-\sum_{j=2}^{M}Q_{1j}\lambda_j=0. \qquad (23)$$

For the extremal point to be a maximum, we require

$$\frac{\partial^2 E}{\partial\lambda_1^2}<0\Rightarrow Q_{11}>0. \qquad (24)$$

Note that this condition may be satisfied by matrices Q that are not necessarily positive definite, as required in the case of typical SVM learning algorithms [19]. From (23), we obtain

$$Q_{11}\lambda_1^{new}=1-\sum_{j=2}^{M}Q_{1j}\lambda_j. \qquad (25)$$

Note that the second term on the R.H.S. of (25) may be written as

$$\sum_{j=2}^{M}Q_{1j}\lambda_j^{old}=\left[-Q_{11}\lambda_1^{old}+\sum_{j=1}^{M}Q_{1j}\lambda_j^{old}\right]=-Q_{11}\lambda_1^{old}+\sum_{j=1}^{M}Q_{1j}\lambda_j^{old}. \qquad (26)$$

Defining

$$f(x)\equiv w^T\phi(x)+b, \qquad (27)$$

and from (19) and (21), we rewrite (26) as

$$\sum_{j=2}^{M} \lambda_j^{old} Q_{1j} = -Q_{11}\lambda_1^{old} + y_1 f^{old}(x^1). \quad (28)$$

Substituting from (28) into (25), we have

$$Q_{11}\lambda_1^{new} = Q_{11}\lambda_1^{old} + 1 - y_1 f^{old}(x^1). \quad (29)$$

which gives us the update rule

$$\lambda_1^{new} = \lambda_1^{old} + \frac{1 - y_1 f^{old}(x^1)}{Q_{11}}. \quad (30)$$

In general, the k-th multiplier is updated by using the rule

$$\lambda_k^{new} = \lambda_k^{old} + \frac{1 - y_k f^{old}(x^k)}{\left(K_{kk} + \frac{1}{A} + \frac{1}{C}\right)}. \quad (31)$$

We can now write the update algorithm for determining the solution to (13) constrained by (6), which we term as the 2SMO algorithm.

The 2SMO Algorithm for the Relaxed LSSVM

1. Pick a multiplier λ_k that violates (18), i.e. $f^{old}(x^k) \neq 1 - \frac{\lambda_k}{C} \Rightarrow \lambda_k \neq C(1 - f^{old}(x^k))$.
2. If all multipliers satisfy the K.K.T. conditions, then the minimum has been attained. Stop.
3. Update λ_k by using (31).
4. Go to Step 1.

The update rule updates one multiplier at a time, and the convergence of the 2SMO update rule (31) is linear in *M.* This also follows from the work of [20]. The update rule 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 in multiplier λ_k, which depends only on K_{ik}. Advantages in terms of a distributed or parallel implementation using O(*M*) processors may be a topic of future research.

It is also possible to simplify the above update algorithm by avoiding checks for the K.K.T. conditions, and continue updating multipliers until the change in their values falls below a specified tolerance. However, this proves to be less efficient.

IV. Variants of the Relaxed LSSVM

A simple variant of is obtained by changing the constraints to inequality ones, i.e. we consider the problem

$$\underset{q,b,w}{\text{Minimize}} \quad \frac{1}{2} w^T w + \frac{A}{2} b^2 + \frac{C}{2} q^T q \quad (32)$$

subject to the constraints

$$y_i \left[w^T \phi(P_i) + b \right] + q_i \geq 1, \; i = 1,2,..., M. \quad (33)$$

The change in the constraints to an inequality appears redundant, since if the L.H.S. is greater than 1, it can always be met as an equality by reducing the value of the error variable. However, in the dual formulation, the Lagrange multipliers are now constrained to be non-negative. The K.K.T. conditions specify that

$$\lambda_i = 0 \Rightarrow y_i \left[w^T \phi\left(x^i\right) + b \right] \geq 1, \tag{34}$$

$$\lambda_i \neq 0 \Rightarrow y_i \left[w^T \phi\left(x^i\right) + b \right] = 1 - \frac{\lambda_i}{C}. \tag{35}$$

Using (34) and (35), we can derive another update rule, which reduces the number of multipliers that need to be updated. Since the multipliers are bounded from one side and not from the other, we refer to this as the SeqsuiSMO algorithm.

The SesquiSMO Algorithm for the Relaxed LSSVM

1. Pick a multiplier λ_k that violates (34) or (35). If all multipliers satisfy the K.K.T. conditions, then the minimum has been attained. Stop.
2. Update λ_k by using (31).
3. Go to Step 1.

One of the desirable features of the classical SVM is that the multipliers are bounded from both below and above. This is useful when implementation needs to consider finite word length effects, e.g. in embedded systems. It also simplifies checks for terminating updates. In classical SVMs, the classifier can be very sensitive to a few training patterns that lie near the decision boundary; Least Squares SVMs offer a less sparse solution but are more robust to noise because the classifier depends on nearly all data patterns; small changes in individual patterns do not tend to perturb the classifier in a major way.

V. Experimental Results

The 2SMO and the SesquiSMO algorithms were implemented in C++ and run on a a dual 3.2GHz Xeon server with 4 GB RAM with the Linux OS. 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.

The performance of the algorithms was compared with that of the C implementation of LSSVM running on the same platform. We ran a series of experiments to study the effect of the parameter *A*, for a fixed value of C. Figures 1 and 2 show the effect on training time, of varying *A* for the kr-vs-kp and mushroom datasets, respectively. The plot indicates that training time montonically decreases with A, and 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 (31). 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 large value of A is a prudent choice.

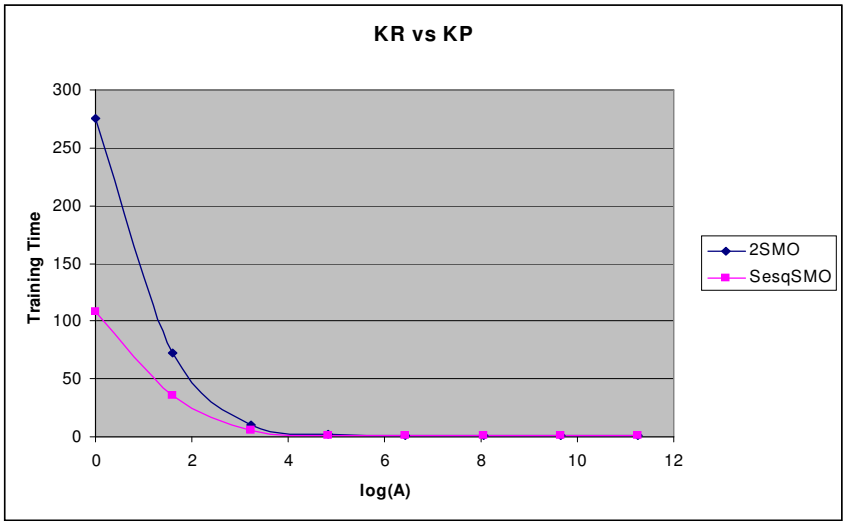

Fig. 1. Plot of training time vs. A for the kr-vs-kp dataset.

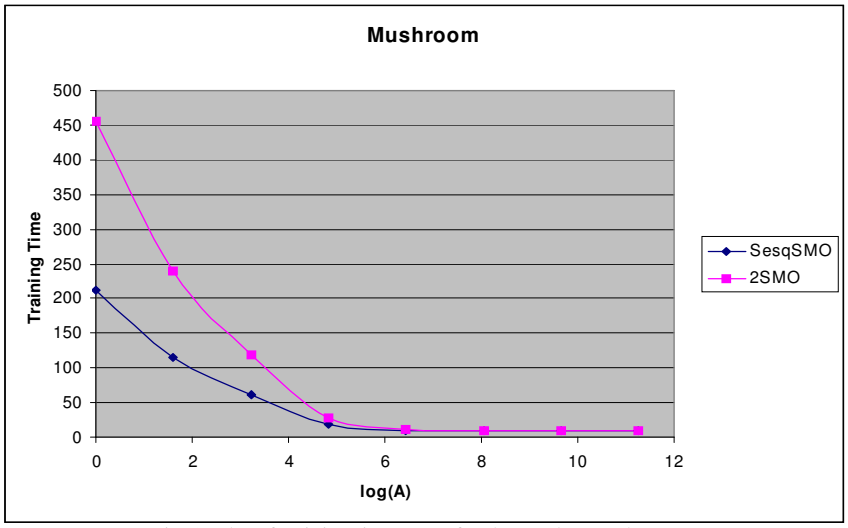

Fig. 2. Plot of training time vs. A for the mushroom dataset.

In order to understand how the algorithm scales with the number of training samples, we ran SesquiSMO, LSSVM and L2SMO on partitions of the mushroom dataset. No kernel caching was used in the case of SesquiSMO. The sizes of the training sets were chosen to be integral multiples of $1/12^{th}$ of the total size. Figure 3 shows the variation of the training times of SesquiSMO, LSSVM and L2SMO with subset size for the adult dataset. Least squares fits to the three curves are given by:

$t_{LSSVM} = 1.2409x^2 - 10.0613x + 22.4068$

$t_{2SMO} = 0.8053x^2 - 6.6341x + 14.8170$

$t_{SesquiSMO} = 0.6889x^2 - 5.6329x + 12.8290$

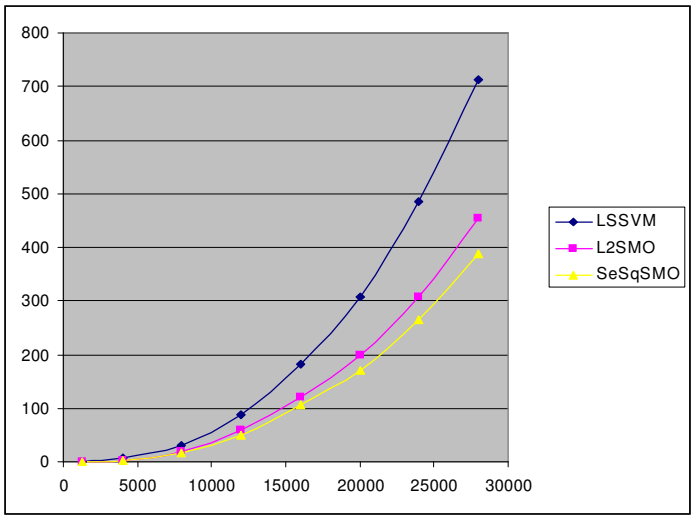

Fig. 3. Plot of variation of training times for SesquiSMO, LSSVM and L2SMO with increasing size of the adult dataset for training

In other words, with increase in the size of the dataset, the training time for SesquiSMO grows at a much slower rate than it does for LSSVM. This indicates that 2SMO and SesquiSMO scale better than LSSVM.

We next conducted a set of experiments on a number of binary classification datasets from the UCI repository. The datasets were picked to cover a wide range of number of features and instances. 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 largest subset of points that could be accomodated was limited by the memory requirements of LSSVM. The results on the kr-vs-kp and mushroom datasets therefore correspond to 50% of the complete datasets. The Table indicates the training times, accuracy, and the number of support vectors yielded by 2SMO, SeqquiSMO, and LSSVM on each of the datasets. Based on the results of the first experiment, we chose $A = 10^4$ for 2SMO. The experiments for all datasets 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.

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

DATASET	METHOD	SUPPORT VECTORS	ACCURACY	TRAINING TIME (S)
BREAST-CANCER (286,51)	LSSVM	215	67.02 ± 0.98	0.023
	2SMO	215	66.60 ± 1.14	0.009
	SesquiSMO	215	66.60 ± 1.14	0.006
BREAST-W (699,10)	LSSVM	359	85.53 ± 1.10	0.044
	2SMO	357	91.19 ± 0.88	0.020
	SesquiSMO	356	91.19 ± 0.88	0.021
CREDIT-G (1000,64)	LSSVM	783	69.78 ± 0.76	0.204
	2SMO	783	69.78 ± 0.76	0.157
	SesquiSMO	783	69.78 ± 0.76	0.151
HEART-C (302,23)	LSSVM	240	54.72 ± 1.16	0.019
	2SMO	240	54.72 ± 1.16	0.012
	SesquiSMO	240	54.72 ± 1.16	0.010
HEART-H (294,25)	LSSVM	233	66.35 ± 0.85	0.017
	2SMO	233	66.35 ± 0.85	0.011
	SesquiSMO	233	66.35 ± 0.85	0.010
HEART-STATLOG (270,14)	LSSVM	165	79.45±0.87	0.04
	2SMO	165	79.43±0.96	0.03
	SesquiSMO	135	78.34±1.10	0.009
HEPATITIS (155,30)	LSSVM	113	77.86±1.52	0.009
	2SMO	113	81.45±1.54	0.001
	SesquiSMO	109	81.45±1.54	0.001
IONOSPHERE (350,35)	LSSVM	230	93.12 ± 1.24	0.056
	2SMO	230	94.82 ± 0.88	0.017
	SesquiSMO	198	95.03 ± 0.78	0.011
PIMA-INDIAN	LSSVM	601	65.96 ± 0.95	0.100

(768,9)	2SMO	601	65.96 ± 0.95	0.073
	SesquiSMO	601	65.96 ± 0.95	0.069
SICK (3772,33)	LSSVM	2890	93.85 ± 0.17	3.123
	2SMO	2890	93.85 ± 0.17	2.093
	SesquiSMO	2890	93.85 ± 0.17	1.953
SONAR (208,61)	LSSVM	146	83.61 ± 1.49	0.018
	2SMO	146	82.50 ± 1.61	0.007
	SesquiSMO	138	81.62 ± 1.85	0.004
VOTE (435, 33)	LSSVM	222	72.71 ± 2.32	0.017
	2SMO	223	84.15 ± 1.16	0.006
	SesquiSMO	223	84.15 ± 1.16	0.006
KR-VS-KP	LSSVM	1582	89.86 ± 1.67	0.905
	2SMO	1582	92.62 ± 1.11	0.450
	SesquiSMO	1582	92.62 ± 1.11	0.435
MUSHROOM	LSSVM	4030	99.98 ± 0.02	7.005
	2SMO	4030	100 ± 0.00	4.27
	SesquiSMO	4030	100 ± 0.00	4.04

It can be observed that the training time of SesquiSMO is consistently lower than that of 2SMO and LSSVM. For larger datasets, SesquiSMO achieves higher speedup factors, roughly between 2 and 4. We note that all three algorithms converge to solutions with approximately the same number of support vectors, on all datasets. 2SMO therefore emerges as an attractive alternative to implementing least squares SVMs.

VI. Relating the The Relaxed LSSVM formulation with the Classical LSSVM

VI.I. Extension to the conventional LSSVM through Penalty Functions

We now discuss the connection between the classical LSSVM formulation and the relaxed SVM one. We also derive two simple update rules based on the 1SMO formulation.

Given an optimization problem of the form

$$\text{Min} \quad f(x), \tag{36}$$

subject to the constraints

$$h_j(x) = 0, \quad j = 1, 2, \ldots, L, \tag{37}$$

where $f(x)$ is convex and $h_j(x)$, j = 1, 2, ...L, are linear, the solution to (36)-(37) may be determined using the theory of Sequential Unconstrained Minimization Techniques (SUMTs). This is achieved by solving a sequence of optimization problems [21] of the form

$$\text{Min} \; E_p(x) = f(x) + \alpha_p \sum_{j=1}^{L} h_j^2(x), \tag{38}$$

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 $E_p(x)$. Denote the solution as x^{p*}.
3. If the constraints (37) 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 $x^{1*}, x^{2*}, \ldots x^{p*}, \ldots$, will converge to the solution of the original problem (36)-(37).

The above procedure, which is a restriction of Sequential Unconstrained Minimization Techniques to convex programming problems with equality constraints, allows us to extend the 2SMO algorithm to the classical LSSVM. To do this, we first note that the dual of the classical LSSVM (5)-(6) is given by

$$\underset{\lambda}{\text{Minimize}} \quad \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M} y_i y_j \lambda_i \lambda_j P_{ij} - \sum_{i=1}^{M} \lambda_i \tag{39}$$

subject to the constraints

$$\sum_{i=1}^{M} \lambda_i y_i = 0, \tag{40}$$

where

$$P_{ij} = \begin{cases} K_{ij}, & i \neq j \\ K_{ii} + \dfrac{1}{C}, & i = j \end{cases}. \tag{41}$$

The SUMT based procedure outlined above indicates that we need to solve a sequence of minimization problems of the form

$$\begin{aligned} \underset{\lambda}{\text{Minimize}} \quad E_p &= \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M} y_i y_j \lambda_i \lambda_j P_{ij} - \sum_{i=1}^{M} \lambda_i + \alpha_p \left\| \sum_{i=1}^{M} y_i \lambda_i \right\|^2 \\ &= \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M} y_i y_j \lambda_i \lambda_j \left(P_{ij} + 2\alpha_p\right) - \sum_{i=1}^{M} \lambda_i \end{aligned} \tag{42}$$

Note that the sequence of minima of (42) yields the solution to the classical SVM formulation (40)-(41) 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 LSSVM is now clear. Observe that (42) is identical to the relaxed SVM formulation of (20)-(21), with $\alpha_p = 1/2A$. Therefore, the solution to the classical SVM *cannot* be obtained by setting A=0 in (20)-(21), but by solving a sequence of problems with diminishing values of A, and with $A \rightarrow 0$ in the limit. Thus, the 2SMO algorithm needs to be incorporated into a loop in which A is progressively reduced to zero, and for each fixed value of A, steps 1-3 of the 2SMO algorithm are executed. This algorithm may be summarized as follows.

Simulating the LSSVM through the Relaxed LSSVM

1. Set A, and the factor ρ by which A will be changed in the sequence of sub-problems.
2. Run the 2SMO algorithm with the specified value of A until convergence is attained.
3. If $\lambda^T y$ is sufficiently close to 0, Stop. Otherwise, go to Step 4.
4. Update A as $A \leftarrow \rho A$. Increment the value of the number of SUMT iterations.
5. Go to Step 2.

We ran a SUMT based algorithm as described above on two datasets, *viz.,* mushroom and breast-cancer. For each experiment, 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 SUMT-iteration (comprising step 3). Figure 3 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

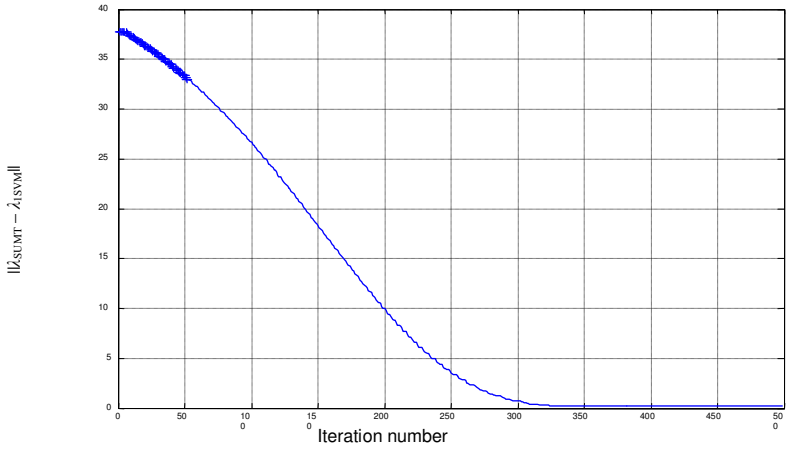

Fig. 4. Plot of $\|\lambda_{SUMT} - \lambda_{LSSVM}\|$ vs. iteration number for the SUMT based algorithm on the *sick* dataset.

iterations and quickly converges to 0. Figure 4 shows for the sick dataset, how the Lagrange multipliers for 2SMO converge to the solution obtained by LSSVM, as iterations progress. The plot demonstrates that the sequence of Relaxed LSSVM sub-problems converges to the solution of the classical SVM.

At this point, we remark that the offset b is given by (29) 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

$$\begin{aligned} & y_i\left[w^T\phi\left(x^i\right)+b\right]=1 \Rightarrow\ w^T\phi\left(x^i\right)+b=y_i \\ & \Rightarrow\ b=y_i-w^T\phi\left(x^i\right) \end{aligned} \tag{43}$$

since $y_i^2=1$. The value of b may be determined from any support vector, or by averaging the values obtained for different support vectors. The classical LSSVM solution may therefore be treated as the limiting case of a Relaxed SVM. Of course, this is only of academic interest, and is not a computationally attractive procedure, since a number of Relaxed LSSVM problems need to be solved to obtain the LSSVM solution.

We now change our focus to two different approaches for solving the classical LSSVM, and show that it is not necessary to find the solution to the classical SVM by solving a sequence of problems, but by solving a single unconstrained problem.

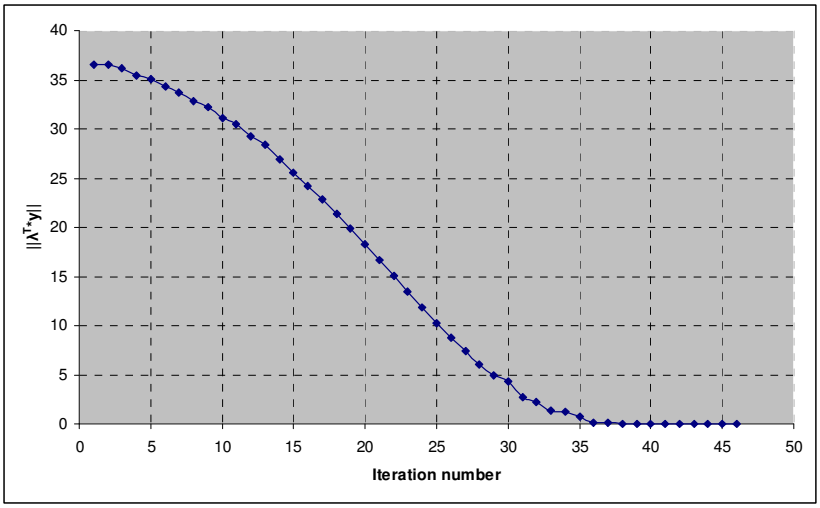

Fig. 5. Plot of variation of $\lambda^{\mathrm{T}} y$ with increasing iteration numbers for the SUMT based algorithm on the *ionosphere* dataset.

VII. Concluding Remarks

In this paper, we introduce a variant of the classical LSSVM, which we term as the Relaxed LSSVM. The Relaxed SVM differs from the conventional LSSVM in having the additional term $\frac{A}{2}b^2$ in the objective function; in this respect, it may be treated as a Least Squares Formulation with an objective function similar to the Proximal SVM [6] and constraints of the classical LSSVM. We show that the Relaxed SVM can be solved through its dual, which involves an unconstrained quadratic minimization problem. This leads to an efficient update rule for the multipliers, termed as 2SMO, in which individual multipliers are updated. The simplicity of the 2SMO Algorithm allows for several optimizations. For example, caching the values of K_{ii}^{-1} requires only O(M) storage, but reduces the computational cost significantly per iteration. We discuss some of the implementation aspects and demonstrate on a number of benchmark datasets that the update rules proposed in this work can be used to obtain marked in terms of convergence time.

The update rule also allows for the updates to be distributed over several processors, each responsible for updating a small subset (ideally a single) multiplier. This is of value in a distributed setting, where the dataset may be either collected over a network, or may be so large that it cannot be handled on a single machine. This may be of particular value in online learning scenarios on large distributed systems.

We show three ways in which a similar update rule can be developed for the classical LSSVM, which normally requires the minimization of a quadratic objective function subject to a linear constraint. Firstly, the theory of Sequential Unconstrained Minimization Techniques shows that the SVM can be solved through a sequence of Relaxed LSSVMs, in which the co-efficient A is successively reduced to zero. Secondly, by using exact penalization, we derive a new objective function. It may be observed that other rules may be obtained by using different exact penalty functions [24], where a differential equation is solved to obtain the multipliers. Thirdly, we show that the classical LSSVM can be reduced to an unconstrained quadratic minimization problem. Whether the solution of LSSVM through Relaxed SVMs is computationally advantageous is an interesting question for current investigation.

References

1. V. Vapnik, *Statistical Learning Theory*, Wiley, 1998.
2. N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and other kernel based learning methods*, Cambridge University Press, 2000.
3. P.S. Bradley and O.L.Mangasarian "Massive data discrimination via linear support vector machines", Optimization Methods and Software, **13**, pp.1-10, 2000.
4. C. Burges, "A tutorial on support vector machines for pattern recognition", *Data Mining and Knowledge Discovery*, **2**, No. 2, pp. 121-167, 1998.
5. Suykens J.A.K., Vandewalle J., "Least squares support vector machine classifiers'', *Neural Processing Letters*, **9**, No. 3, pp. 293-300, June 1999.
6. G. Fung and O.L. Mangasarian, "Proximal support vector machine classifiers", in D.Lee et al (Eds.), *Proc. KDD-2001: Knowledge Discovery and Data Mining*, San Francisco, CA, Association for Computing Machinery, New York, pp. 77-86, 2001.
7. E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. In *Proc. of IEEE NNSP'97*, Amelia Island, Florida, Sep. 24-26, 1997. Online at http://citeseer.ist.psu.edu/osuna97improved.html
8. J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Sch¨olkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1998.
9. J. Platt. Using sparseness and analytic QP to speed training of support vector machines. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*. MIT Press, 1999.
10. S.S. Keerthi et. al, "Improvements to Platt's SMO algorithm for SVM classifier design, *Neural Computation*," 13 pp. 637-649, 2001.
11. S.K. Shevade et. al, "Improvements to the SMO algorithm for SVM regression", *IEEE Transactions on Neural Networks*, 11 pp. 1188-1194, 2000.
12. T. Joachims, "Making large-scale support vector machine learning practical," in *Advances in Kernel Methods: Support Vector Machines*, B. Scholköpf, C. Burges, and A. Smola, Eds. Cambridge, MA: MIT Press, 1998.

13. Chih-Chung Chang and Chih-Jen Lin. LIBSVM: a library for support vector machines, 2001. Online at http://www.csie.ntu.edu.tw/□cjlin/libsvm
14. T. Glasmachers and C. Igel, "Maximum-Gain Working Set Selection for SVMs", Journal of Machine Learning Research 7 (2006) 1437–1466.
15. C.-J. Lin, "On the convergence of the decomposition method for support vector machines," *IEEE Trans. Neural Netw.*, vol. 12, no. 6, pp. 1288–1298, Nov. 2001.
16. X. Zeng and X.-W. Chen, "SMO-based pruning methods for sparse least squares support vector machines", *IEEE Transactions on Neural Networks*, **16**, No. 6, pp. 1541 - 1546, Nov. 2005.
17. S.S. Keerthi and S.K. Shevade, ""SMO algorithm for least squares SVM", Proc. International Joint Conference on Neural Networks 2003, Volume 3, pp. 2088-2093, 20-24 July 2003.
18. Kecman V., T.-M. Huang, M. Vogt, "Iterative Single Data Algorithm for Training Kernel Machines from Huge Data Sets: Theory and Performance", in *Support Vector Machines: Theory and Applications*, Ed. L. Wang, Series: Studies in Fuzziness and Soft Computing, Vol. 177, Springer-Verlag, pp. 255-274, 2005.
19. P.S. Sastry, "An Introduction to Support Vector Machines", *private communication.*
20. 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).
21. A.V. Fiacco and G.P. McCormick, *Nonlinear Programming: Sequential unconstrained minimization techniques*. Wiley and Sons, New York, 1968.
22. R. Fletcher, *Practical methods of optimization*, John Wiley, 1987.
23. R. Fletcher, "A Class of Methods for Nonlinear Programming with Termination and Convergence Properties", in J. Abadie (Ed.), *Methods for Nonlinear Programming*,
24. A. Bhaya and E. Kaszkurewicz, "Control Perspectives on Numerical Algorithms And Matrix Problems", Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2006.
25. L.V. Ferreira, E. Kaszkurewicz, and A. Bhaya, " Support vector classifiers via gradient systems with discontinuous righthand sides", *Neural Networks*, **19**, pp. 1612–1623, 2006.
26. J. A. K. Suykens, P. Van Dooren, B. De Moor, and J. Vandewalle. "Least squares support vector machine classifiers: a large scale algorithm", *European Conference on Circuit Theory and Design,* pp. 839-842, 1999.

Appendix I
Extension to the conventional LSSVM through a Single Unconstrained Minimization : Non-smooth penalty functions

In the case of convex programming problems, it is possible to use exact penalty functions [22], [23] that require the solution of a single unconstrained minimization, instead of a sequence of problems, as in the case of SUMTs. On the lines of the exact penalty function proposed in [21]-[25], we note that the solution to (39)-(40) can be determined by solving the following problem.

$$\underset{\lambda}{\text{Minimize}} \quad E_C = \frac{1}{2}\lambda^T DPD\lambda - e^T\lambda + \beta\left|y^T\lambda\right| \tag{44}$$

Following [24], [25], it can be shown, using a Control Lyapunov Function approach, that the gradient dynamical system

$$\dot{\lambda} = e - G\lambda - \beta y\,\mathrm{sgn}(r) \tag{45}$$

$$\dot{r} = y^T\dot{\lambda} \tag{46}$$

where $G = DPD$, $r = y^T\lambda$ and

$$\mathrm{sgn}(\theta) = \begin{cases} 1, & \theta > 0 \\ -1, & \theta < 0 \end{cases} \tag{47}$$

converges to a solution of the problem (39)-(40) for any choice of β, and starting from any initial λ.

27. Solution through Substitution

We once again consider the problem (39)-(40). The Lagrangian for this problem may be written as

$$L = \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M}\lambda_i\lambda_j G_{ij} - \sum_{i=1}^{M}\lambda_i - \gamma\sum_{i=1}^{M}\lambda_i y_i = \frac{1}{2}\lambda^T G\lambda - e^T\lambda - \gamma\left(\lambda^T y\right), \tag{48}$$

where

$$G = DPD, \tag{49}$$

D being a diagonal matrix containing class labels on its diagonal. A minimum of (39)-(40) requires that the K.K.T. conditions for a minimum of (48) are satisfied. In other words,

$$\nabla_\lambda L = 0 \;\Rightarrow\; DPD\lambda - e - y\gamma = 0 \;\Rightarrow\; y\gamma = DPD\lambda - e. \tag{50}$$

Pre-multiplying (50) by y^T, we have

$$\left(y^T y\right)\gamma = y^T DPD\lambda - y^T e \;\Rightarrow\; M\gamma = y^T DPD\lambda - y^T e \;\Rightarrow\; \gamma = \frac{1}{M}\left(y^T DPD\lambda - y^T e\right). \tag{51}$$

Substituting in (48), we obtain

$$\begin{aligned} L &= \frac{1}{2}\lambda^T DPD\lambda - e^T\lambda - \frac{1}{M}\lambda^T y\left(y^T DPD\lambda - y^T e\right) \\ &= \frac{1}{2}\lambda^T DPD\lambda - e^T\lambda - \frac{1}{M}\left(\lambda^T yy^T DPD\lambda - \lambda^T yy^T e\right) \\ &= \frac{1}{2}\lambda^T\left[DPD - \frac{2}{M}yy^T DPD\right]\lambda - \lambda^T\left[\left(I - \frac{1}{M}yy^T\right)e\right] \\ &= \frac{1}{2}\lambda^T W\lambda - d^T\lambda. \end{aligned} \tag{52}$$

where

$$W = \left(DPD - \frac{2}{M}yy^T DPD\right); \quad d = \left(I - \frac{1}{M}yy^T\right)e. \tag{53}$$

Therefore, the solution to (47)-(49) and the solution to the following problem have a one-to-one correspondence.

$$\underset{\lambda}{\text{Minimize}} \quad \frac{1}{2}\lambda^T W\lambda - d^T\lambda, \tag{54}$$

where *W* and *d* are given by (53). In order for us to develop a 2SMO style update algorithm for solving (54), we require *W* to be positive semi-definite; this is not ensured in general.

We modify the Lagrangian in (52) to

$$L = \frac{1}{2}\lambda^T\left[DPD - \frac{2}{M}yy^T DPD\right]\lambda - \lambda^T\left[\left(I - \frac{1}{M}yy^T\right)e\right] + \frac{\alpha}{2}\lambda^T\left[yy^T DPD\right]\lambda. \tag{55}$$

Note that the last term on the R.H.S. of (55) is positive semi-definite everywhere. This is because the matrix $yy^T DPD$ is positive definite, it being the product of two positive definite matrices. On the feasible surface, $y^T\lambda = 0$, and hence, the last term is zero, since it may be rewritten as $\frac{\alpha}{2}\left(\lambda^T y\right)\left(y^T DPD\right)\lambda$. We observe that (55) may be simplified to

$$\begin{aligned} L &= \frac{1}{2}\lambda^T\left[DPD + \left(\alpha - \frac{2}{M}\right)yy^T DPD\right]\lambda - \lambda^T\left[\left(I - \frac{1}{M}yy^T\right)e\right] \\ &= \frac{1}{2}\lambda^T\left[DPD + \left(\alpha - \frac{2}{M}\right)yy^T DPD\right]\lambda - d^T\lambda, \end{aligned} \tag{56}$$

where *d* is given by (53). The consequence is that *W* is modified to W_1, where

$$W_1 = DPD + \left(\alpha - \frac{2}{M}\right)yy^T DPD. \tag{57}$$

We observe that for sufficiently large α, the matrix W_1 on the L.H.S. of (57) will be positive semi-definite. We remark that the form of (54) can also be obtained by using the exact penalty function proposed by Fletcher [23] for equality constrained problems.

We further remark that the fact that $\lambda^T y = 0$ on the feasible surface may be utilized to allow the use of kernel matrices that are not necessarily positive definite. This may be done by modifying the Lagrangian in (56) to

$$\begin{aligned} L &= \frac{1}{2}\lambda^T\left[DPD + \left(\alpha - \frac{2}{M}\right)yy^T DPD\right]\lambda - d^T\lambda + + \frac{\beta}{2}\left\|\lambda^T y\right\|^2 \\ &= \frac{1}{2}\lambda^T\left[DPD + \left(\alpha - \frac{2}{M}\right)yy^T DPD\right]\lambda + \frac{\beta}{2}\left(\lambda^T yy^T\lambda\right) - d^T\lambda \\ &= \frac{1}{2}\lambda^T\left[DPD + \left(\alpha - \frac{2}{M}\right)yy^T DPD\right]\lambda + \frac{\beta}{2}\left(\lambda^T yy^T\lambda\right) - d^T\lambda \\ &= \frac{1}{2}\lambda^T\left[DPD + \left(\alpha - \frac{2}{M}\right)yy^T DPD + \beta yy^T\right]\lambda - d^T\lambda. \end{aligned} \tag{58}$$

The values of α and β need to be chosen so that the Hessian is always positive semi-definite. The possibility of using Non-Mercer kernels by choosing a larger value of β is worthy of further investigation, and bears a certain resemblance to the regularization term used in Ridge Regression and similar approaches. Larger values of α and β may also have a bearing on the convergence rate. We examine some of these aspects in a companion paper. However, a minor point of consideration is, that for larger values of α, the matrix $DPD + \left(\alpha - \frac{2}{M}\right)yy^T DPD$ is not necessarily symmetric. We consider the quadratic form

$$E = \frac{1}{2}\lambda^T R\lambda - d^T\lambda \tag{59}$$

and note that when R is not symmetric, it may be replaced by its symmetric component, that is $\frac{1}{2}\left(R + R^T\right)$. This is because

$$\lambda^T R\lambda = \frac{1}{2}\lambda^T\left(R + R^T\right)\lambda + \frac{1}{2}\lambda^T\left(R - R^T\right)\lambda, \tag{60}$$

but

$$\frac{1}{2}\lambda^T\left(R - R^T\right)\lambda = 0, \tag{61}$$

since the matrix $(R - R^T)$ is antisymmetric.

The results indicate that one could solve the LSSVM through a Relaxed LSSVM, and evolve update algorithms similar to the 2SMO or SesquiSMO. However, note that additional computations may be required to determine matrices W_1 and the vector d. These are likely to adversely affect any potential speedup. We defer further discussion on this aspect for the present.