Building Support Vector Machines in the Context of Regularised Least Squares

Peng, J., Rafferty, K., & Ferguson, S. (2016). Building Support Vector Machines in the Context of Regularised Least Squares. Neurocomputing, 1. DOI: 10.1016/j.neucom.2016.03.087

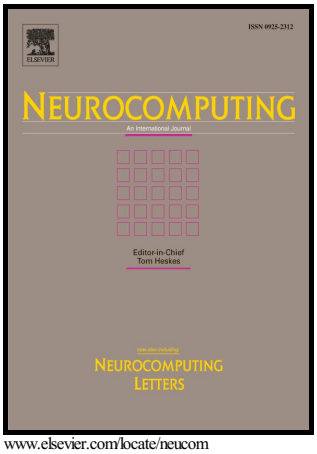
Published in:
Neurocomputing

Document Version:
Peer reviewed version

Queen's University Belfast - Research Portal:
Link to publication record in Queen's University Belfast Research Portal

Building support vector machines in the context of regularised least squares

Jian-Xun Peng, Karen Rafferty, Stuart Ferguson



www.elsevier.com/locate/neucom

PII: S0925-2312(16)30575-6
DOI: http://dx.doi.org/10.1016/j.neucom.2016.03.087
Reference: NEUCOM17187

To appear in: *Neurocomputing*

Received date: 19 August 2015
Revised date: 8 March 2016
Accepted date: 21 March 2016

Cite this article as: Jian-Xun Peng, Karen Rafferty and Stuart Ferguson, Building support vector machines in the context of regularised least squares *Neurocomputing,* http://dx.doi.org/10.1016/j.neucom.2016.03.087

Building Support Vector Machines in the Context of Regularised Least Squares

Jian-Xun Peng, Karen Rafferty, Stuart Ferguson

The School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, Ashby Building, Stranmillis Road, Belfast BT9 5AH, UK

Abstract

This paper formulates a linear kernel support vector machine (SVM) as a regularized least-squares (RLS) problem. By defining a set of indicator variables of the errors, the solution to the RLS problem is represented as an equation that relates the error vector to the indicator variables. Through partitioning the training set, the SVM weights and bias are expressed analytically using the support vectors. It is also shown how this approach naturally extends to Sums with nonlinear kernels whilst avoiding the need to make use of Lagrange multipliers and duality theory. A fast iterative solution algorithm based on Cholesky decomposition with permutation of the support vectors is suggested as a solution method. The properties of our SVM formulation are analyzed and compared with standard SVMs using a simple example that can be illustrated graphically. The correctness and behavior of our solution (merely derived in the primal context of RLS) is demonstrated using a set of public benchmarking problems for both linear and nonlinear SVMs.

Keywords: Data classification, support vector machines, regularized least squares, fast training algorithm, Cholesky decomposition

1. Introduction

Support vector machines (SVM) are a set of empirical data modeling techniques that is firmly grounded in the framework of *VC theory* [1], a specific approach to computational learning theory. The SVM was originally developed for binary data classification problems. Conceptually, a machine maps the input space to a so-called *feature space* through some non-linear mapping chosen *a priori*. The feature space is of higher dimension than the input space. In this feature space a linear decision surface is constructed based on the structural risk minimization (SRM) principal: an upper bound on the expected risk (the expectation of the test error for a machine on an unseen point) is minimized by maximization of the margin [2]. The margin refers to the distance between the two parallel hyperplanes that bound the training points of the two classes, respectively. The hyperplane that lies midway between the two bounding hyperplanes is called the decision hyperplane, and the training points that determine the two parallel bounding hyperplanes are referred to as the *support vectors*.

It was shown [2] that if the training vectors are separated without errors by an optimal hyperplane the expectation value of the probability of committing an error on a test example is bounded above by the ratio between the expectation value of the number of support vectors and the number of training vectors. Particularly, this bound does not explicitly contain the dimensionality of the feature space. It follows from this bound, that if the optimal hyperplane can be constructed from a small number of support vectors, relative to the training set size, then the generalization ability will be high even in an infinite dimensional feature space.

This optimal margin algorithm is generalized by [3] to non-separable data sets by the introduction of non-negative slack variables as a measurement of the misclassification errors in the statement of the optimization problem, and by using a structural objective function with a penalty term on the training errors. For a sufficiently large penalty parameter C, the hyperplane is chosen so that it minimizes the number of errors on the training set, while the rest of then training points are separated with maximal margin; if the training data can be separated without errors, then the hyperplane obtained in the procedure coincides with the optimal margin hyperplane.

Compared with traditional methods employed by conventional neural networks, the SRM principle has been shown to be superior because it not only minimizes the error on the training data [4], but also minimizes the capability of the model [5]. This equips SVM with a greater ability to generalize, which is the goal in statistical learning. Experimental studies have demonstrated the competitive performance of SVMs in a range of application fields [6][7][8][9].

Typically, constructing a SVM involves a constrained quadratic (or convex) optimization problem. In the majority of textbooks and articles introducing SVMs, instead of directly solving the primal problem, a dual of the problem is formulated using Lagrange multipliers [10][3][11][12]. There are two reasons for doing this [10]: a) duality theory provides a convenient way to deal with the constraints, and b) with the Lagrange reformulation of the problem, the training data will only appear (in the actual training and test phases) in the form of dot products between input vectors. This is a crucial property which allows us to generalize SVMs to the nonlinear case. In addi-

tion, most popular algorithms and existing toolboxes (for example, interior point method [13] and the sequential minimal optimization (SMO) [14] algorithm) formulate their solution in the dual. This gives the strong impression that this is the only possible way to construct a SVM, particularly for SVMs with nonlinear kernels. There has aslo been quite a lot of interest in studying systems that have particular properties, for example [15]'s work on sparse learning- and [16]'s sparse least squares. However, there has been increasing interest in constructing SVMs directly in the primal. [17] formulated a primal least-square version of the SVM, which had been originally proposed in the dual [18]. [19] applied conjugate gradient schemes to logistic regression for data classification. [20] proposed an algorithm for linear L1-SVMs that works by approximating the L1-loss function by a sequence of smooth modified logistic regression loss functions, this is followed by sequentially solving smooth primal modified logistic regression problems using nonlinear conjugate gradient methods. However, all the inequality constraints are replaced by equality constraints in least squares SVM. A particular drawback of that method is its inability to exploit the sparsity property of SVMs in which only the support vectors determine the final solution. To overcome this shortage of the least squares SVM, a pruning method was proposed based on the fact that support values reveal the relative importance of each of the training data points, where a small number of points, e.g., 5% in the training set [21], that have the smallest values in the sorted support vector values spectrum, are removed in each training loop, until some user-defined performance index degrades.

Some promising primal algorithms have also been studied for standard

linear SVMs [22][23][24], and implemented in toolboxes for linear SVMs, for example, in LIBLINEAR [25]. All of these algorithms are based on the fact that, for linear SVM, the feature space is the same as the input space, the normal vector to the separating hyperplane is thus explicitly presented in the linear SVM. However, for SVMs with nonlinear kernels, where some nonlinear map from the input space to the feature space exists, the map itself and many of its properties are unknown [26]. What is known is, a given kernel function involving a dot product in the feature space, a concept introduced by [27], thus the normal is not explicitly present in the final discriminative function of nonlinear SVMs again. This makes it difficult to apply primal solution algorithms in nonlinear kernel cases.

[28] showed that when the goal is to find an approximate solution, primal optimization is superior because it is focused on minimizing what we are directly interested in: the primal objective function. Motivated by this, a Newton method is applied to the primal problem for both linear and nonlinear cases. For the nonlinear case, the optimal solution to the SVM is expressed by a linear combination of the kernel functions evaluated in all the training points based on the *representer theorem* of [29]. Given this linear combination solution, and using the representing property of the kernel, the problem is thus converted into one of optimizing the linear coefficients in the combination. This requires the full kernel matrix to be invertible (positive definite), given that the full kernel matrix is a symmetric matrix formed by pair-wise point inner products or kernel evaluations on the full training set. An iterative technique, IRWLS [30] [31], based on re-weighed least squares produced the fastest algorithm of its time. The IRWLS approach was sub-

sequently proved to converge to the SVM solution [32], Since then there has been continued interest in primal and iterative least squares approaches to finding the best SVM solution [33] [34] [35]. Recently, particularly in the machine learning arena, recursive and weighted least squares has attracted interest in the context of twin support vectors, [36] provide an overview or nonparallel hyperplane algorithms and [37] and [38] illustrate recent work on twin support vector machines.

The goal of this paper is to show how a primal SVM algorithm can be constructed that removes some of the caveats on other formulations of primal solutions. Most notably: we use kernel matrices that need only be positive semi-definite and suggest a procedure that overcomes the lack-of-sparseness shortage of least squares SVMs. Those points without violations are not presented in the solution. Our SVM in this paper is different from the least squares SVM [18] in that our SVM is derived merely in the context of RLS, while the least squares SVM was originally derived in the dual. Secondly, the least squares SVM replaces the inequality constraints with equalities while the solution proposed in this paper minimizes the violations without that replacement.

Our formulation begins as a regularized least squares (RLS) problem as was done by [28]. LSSVM only needs to solve a linear equation set rather than dealing with a quadratic programming problem, by using equality constraints instead of inequality ones and a least squares loss function, which greatly reduces the computational complexity [39]. The training set is partitioned into two parts: the one includes those points that are bounded by the two class-bounding-hyperplanes, and another one includes those points that

are unbounded. The later is referred to as the *support vector set* hereafter. Accordingly, the error vector is partitioned into two parts. The main contribution of this paper is the derivation of the optimal solution with the use of only some matrix operations for the partitioned error vector and merely in the context of the RLS. Instead of giving the linear combination form of the optimal solution in advance as was done by [28], our optimal solution is derived and can be expressed as a linear combination of the inner products of the support vectors with an input point. This approach not only overcomes the drawback of the invertability requirement of the kernel matrix, but also makes it natural to generalize to cases of SVMs with nonlinear kernels.

In Section 2, an SVM with linear kernel is formulated as an unconstrained minimization problem with the L2-loss.

The main details of our approach is presented in sections 3 and 4. Firstlty section 3 expresses the solution to the problem as an equation with regard to the error vector and a set of indicator variables. Then In section 4, it is shown how the error vector may be partitioned into two parts and how the solution to the linear SVM is expressed as a linear combination of inner products of the support vectors with an input point. How the solution may be generalized to cases of nonlinear kernels is discussed in comparison with standard SVM. In section 5 an iterative algorithm to solve our SVM formulation is described, this is based on Cholesky decomposition, (an approach also favored by [40]) and offers the potential to contribute when it comes to develop a wider population of problems with nonlinear kernels. The accuracy of the method is examined in Section 6 by comparing the algorithm's output with that from some existing SVM software packages. Section 7 draws a few

conclusions about our algorithm.

2. Linear Support Vector Machines

This section briefly reviews the SVM and introduces the notation to be used in the paper.

Given a data set of N point-label pairs $\{(\mathbf{x}_k, y_k), k = 1, \ldots, N\}$, referred to as the training set, each point is represented in a row vector $\mathbf{x}_k \in \Re^{1\times n}$, to which a label either $+1$ or -1, i.e., $y_k \in \{+1, -1\}$, is attached. This means the points (or patterns in some context) fall into two categories. This is a binary data classification problem, where a classifier is to be found that can separate the points into two classes.

For general SVMs with nonlinear kernels, there is some nonlinear map from the input space to a high-dimensional feature space where the two parallel bounding hyperplanes and the separating hyperplane are defined. In the following we formulate the linear SVM for simplicity, but all the derivations and conclusions are applicable to nonlinear cases by replacing $\mathbf{x} \leftarrow \mathbf{f} = \phi(\mathbf{x})$, where ϕ denotes some nonlinear map from the input space to a feature space, $\mathbf{x}$ an input point and $\mathbf{f} = \phi(\mathbf{x})$ the corresponding map of $\mathbf{x}$ in the feature space.

The standard SVM with a linear kernel is given by the following primary quadratic program [3][28]:

$$\min_{\mathbf{w},b} J(\mathbf{w}, b) = \mathbf{w}^T\mathbf{w} + C\sum_{k=1}^{N} \xi_k^p \tag{1}$$

subject to

$$\begin{cases} \mathbf{x}_k\mathbf{w}+b & \geq & +1-\xi_k, \texttt{if} \quad y_k=+1 \\ \mathbf{x}_k\mathbf{w}+b & \leq & -1+\xi_k, \texttt{if} \quad y_k=-1 \\ \xi_k & \geq & 0, \quad k=1,\cdots,N \end{cases} \tag{2}$$

which is equivalent to

$$y_k(\mathbf{x}_k\mathbf{w}+b) \geq 1-\xi_k, \xi_k \geq 0, k=1,\cdots,N \tag{3}$$

where $C > 0$ is referred to as the *penalty parameter*; p is either 1 [10] or 2 [23], $J(\mathbf{w}, b)$ is the objective function to be minimized. This quadratic program defines two parallel hyperplanes in $\Re^n$ that have the same normal vector $\mathbf{w}$ (a n-dimensional column vector). The two hyperplanes are given by $\mathbf{x}_k\mathbf{w}+b=\pm 1$. The distance between the two parallel hyperplanes, referred to as the *margin*, is given by $2(\mathbf{w}^T\mathbf{w})^{-1/2}$. Hyperplane $\mathbf{x}_k\mathbf{w}+b=+1$ bounds the points of class $y_k=+1$, while $\mathbf{x}_k\mathbf{w}+b=-1$ bounds the points of class $y_k=-1$. For a linearly inseparable training data set, points that lie on the wrong side of the hyperplane bounding their category are called violates of the points. The magnitude of these violations ξ_k, $\xi_k \geq 0$, referred to as the *violations* in this paper, are penalized in the objective function. The violation is zero for any point of the training set that satisfies $y_k(\mathbf{x}_k\mathbf{w}+b) \geq 1$, meaning that this point is within the category boundary. The problem (1) is thus solved by maximizing the margin while minimize the loss (the sum of the violations or squared violations). The penalty parameter C is predefined to balance between the magnitude of then margin and the number of violations. The hyperplane:

$$f(\mathbf{x}) = \mathbf{x}\mathbf{w}+b=0 \tag{4}$$

lying midway between the two bounding hyperplanes is the (linear) separating boundary, which is the discriminant function of the classifier for predicting the category y of an unseen points $\mathbf{x} \in \Re^{1\times n}$ as follows:

$$y = \begin{cases} +1, & if \quad f(\mathbf{x}) > 0 \\ -1, & if \quad f(\mathbf{x}) < 0 \end{cases} \tag{5}$$

where $f(\mathbf{x})$ is the discriminant function of the SVM classifier given in (4).

The quadratic program (1) with constraints (3) can be reformulated as the following minimization problem without constraints [25][28]:

$$\min_{\mathbf{w},b} J(\mathbf{w}, b) = \mathbf{w}^T\mathbf{w} + C\sum_{k=1}^{N} L(\mathbf{w}, b; \mathbf{x}_k, y_k) \tag{6}$$

where the loss function L is properly chosen such that points that violate the constraints in (3) are penalized according to the violations, while points that satisfy the constraints are ignored. The two most commonly used loss functions are $L(\mathbf{w}, b; \mathbf{x}_k, y_k) = \max(0, 1 - y_k(\mathbf{x}\mathbf{w} + b))$ and

$$L(\mathbf{w}, b; \mathbf{x}_k, y_k) = \max(0, 1 - y_k(\mathbf{x}\mathbf{w} + b))^2, \tag{7}$$

respectively referred to as L1-loss (hinged loss) and L2-loss functions [25], respectively, where and hereafter function $\max(\cdot, \cdot)$ values the maximal one of the two input arguments to the function.

In this paper the L2-loss function is used, which is first-order continuously differentiable with respect to $\mathbf{w} \in \Re^n$ and $b \in \Re$. In this case, the problem specified by (6) is to minimize the sum of the squared violations of the training points with a regularization term $\mathbf{w}^T\mathbf{w}$. In this sense, problem (6) is thus viewed as a regularized least-squares (RLS) problem, That is: Generalize the objective (6) as $\sum_{i=1}^{n} |w_i|^{p_w} + C\sum_{k=1}^{N} L(\mathbf{w}, b; \mathbf{x}_k, y_k)^{p_h}$ with $p_w, p_h > 0$, which

is convex only when $p_w, p_h \geq 1$. We known that small p_h encourages sparsity in the dual variables (less kernels in the model), while small p_w encourages sparsity in the primal variables (less features involved in the model). Based on this fact, [41] proposed $p_w < 1$ for the purpose of feature selection (for smallest number of inputs to SVM), while standard SVM use $p_h = 1$ (the smallest value for convex problems) for sparse models (for smallest number support points/kernels).

Most methods in the literature that solve this unconstrained optimization problem or its dual are based on quadratic programming [42] or Newton search [43]. As the objective function (6) for the L2-SVM loss function is obviously convex with regard to $[\mathbf{w}, b] \in \Re^{n+1}$, there is a unique optimal solution to $\mathbf{w}$ and b that minimizes the objective function. Thus a vanishing gradient becomes a sufficient condition for a feasible solution to be globally optimal and [32] proves that iterative lease squares conveges to an SVM for hinge loss constraints.

Based on this, the solution for a linear SVM is derived in section 3, it is then extended to allow for nonlinear generalizations in section 4. These two sections constitute the details of our formulation. Later sections provide details of an iterative algorithm that uses the new formulation and some basic verifications of their efficacy, sec the following section.

3. Solution in the Primal

For the L2-SVM loss function (7), the first-order partial derivatives of $J(\mathbf{w}, b)$ in (6) with respect to $\mathbf{w}$ and b are given by

$$\frac{\partial J}{\partial \mathbf{w}} = 2\mathbf{w} + 2C\mathbf{P}^T\mathbf{S}(b\mathbf{y} + \mathbf{P}\mathbf{w} - \mathbf{1}) \tag{8}$$

and

$$\frac{\partial J}{\partial b} = 2C\mathbf{y}^T\mathbf{S}(b\mathbf{y} + \mathbf{P}\mathbf{w} - \mathbf{1}) \tag{9}$$

where $\mathbf{1}$ denotes a column of N unity elements, $\mathbf{y} = [y_1, \cdots, y_N]^T$, $\mathbf{P} = [\mathbf{p}_1^T, \cdots, \mathbf{p}_N^T]^T$ with $\mathbf{p}_k = y_k\mathbf{x}_k$ or simply $\mathbf{P} = diag(\mathbf{y})\mathbf{X}$. In this paper, $diag(\cdot)$ denotes the diagonal matrix formed with a given vector as the diagonal elements, and for convenience, the N n-dimensional training patterns are collected in an $N \times n$ matrix $\mathbf{X}$.

The matrix

$$\begin{aligned} &\mathbf{S}{=}diag(\mathbf{s}), \mathbf{s} = [s_1, \cdots, s_N]^T \\ &s_k{=}s(e_k) = \begin{cases} 0, & e_k \leq 0 \\ 1, & e_k > 0 \end{cases} \\ &e_k{=}1 - y_k(\mathbf{x}_k\mathbf{w} + b), k = 1, \cdots, N \end{aligned} \tag{10}$$

arises during the calculation of the partial derivatives (8) and (9) where the first-order differential of function $L(e) = \max(0, e)^2$ with regard to e is involved, given by

$$\frac{dL(e)}{de} = 2s(e)e = \begin{cases} 0, & e \leq 0 \\ 2e, & e > 0 \end{cases} \tag{11}$$

Here $e_k = 1 - y_k(\mathbf{x}_k\mathbf{w} + b)$ is referred to as the error for point $\mathbf{x}_k$, the indicator variables, $s_k, k = 1, \cdots, N$, identify which points give rise to the support vectors.

Equating the first-order partial derivatives (8) and (9), to zero results in solutions for $\mathbf{w}$ and b:

$$\begin{cases} \mathbf{w} &= C(\mathbf{I} + C\mathbf{P}^T\mathbf{S}\mathbf{B}\mathbf{P})^{-1}\mathbf{P}^T\mathbf{S}\mathbf{B}\mathbf{1} \\ b &= \frac{\mathbf{y}^T\mathbf{S}(\mathbf{1}-\mathbf{P}\mathbf{w})}{\mathbf{y}^T\mathbf{S}\mathbf{y}} \end{cases} \tag{12}$$

where (and hereafter) $\mathbf{I}$ denotes an identity matrix of proper size and

$$\mathbf{B} = \mathbf{I} - \frac{\mathbf{y}\mathbf{y}^T\mathbf{S}}{\mathbf{y}^T\mathbf{S}\mathbf{y}} \tag{13}$$

It is easy to verify that $\mathbf{B}\mathbf{B} = \mathbf{B}, \mathbf{B}$ is therefore idempotent, i.e. the eigenvalues of $\mathbf{B}$ are either 1 or zero. Noting that the second term is formed by left-multiplying a column vector to a row vector, which has one unity eigenvalue and $N-1$ zero eigenvalues, $\mathbf{B}$ is of rank $N-1$ and has $N-1$ unity and one zero eigenvalues. Vector $\mathbf{y}$ is obviously the eigenvector for the zero eigenvalue.

Noting the solution to b given in (12), define

$$\begin{aligned}\mathbf{e} &= \mathbf{1} - \mathbf{P}\mathbf{w} - b\mathbf{y} = \mathbf{B}(\mathbf{1} - \mathbf{P}\mathbf{w}) \\ &= \mathbf{B}[\mathbf{I} - C\mathbf{P}(\mathbf{I} + C\mathbf{P}^T\mathbf{S}\mathbf{B}\mathbf{P})^{-1}\mathbf{P}^T\mathbf{S}\mathbf{B}]\mathbf{1} \\ &= [\mathbf{I} - C\mathbf{B}\mathbf{P}(\mathbf{I} + C\mathbf{P}^T\mathbf{S}\mathbf{B}\mathbf{P})^{-1}\mathbf{P}^T\mathbf{S}]\mathbf{B}\mathbf{1}\end{aligned} \tag{14}$$

which is the column of the errors for the N training points, and referred to as the *error vector*. Using the matrix identity (15), which is a special case of the Woodbury matrix identity (19) [44]

$$(\mathbf{I} + \mathbf{A}\mathbf{B})^{-1} = \mathbf{I} - \mathbf{A}(\mathbf{I} + \mathbf{B}\mathbf{A})^{-1}\mathbf{B} \tag{15}$$

we have

$$\mathbf{e} = \mathbf{B}(\mathbf{I} + C\mathbf{P}\mathbf{P}^T\mathbf{S}\mathbf{B})^{-1}\mathbf{1} = (\mathbf{I} + C\mathbf{B}\mathbf{P}\mathbf{P}^T\mathbf{S})^{-1}\mathbf{B}\mathbf{1} \tag{16}$$

Note that the errors $e_k = 1 - y_k(\mathbf{x}_k\mathbf{w} + b)$ here are different from the violations ξ_k defined in (3), the relationship between them is $\xi_k = s_k e_k, k = 1, \cdots, N$.

If a classifier's discriminant function, $f(\mathbf{x}) = \mathbf{x}\mathbf{w} + b$, does not require the

bias term b, i.e. $b = 0$, simply let $\mathbf{B} = \mathbf{I}$, to give

$$\begin{cases} \mathbf{w} &= C(\mathbf{I} + C\mathbf{P}^T\mathbf{S}\mathbf{P})^{-1}\mathbf{P}^T\mathbf{S}\mathbf{1} \\ b &= 0 \end{cases} \tag{17}$$

with the corresponding error vector

$$\mathbf{e} = (\mathbf{I} + C\mathbf{P}\mathbf{P}^T\mathbf{S})^{-1}\mathbf{1} \tag{18}$$

Equation (16) represents a nonlinear relationship between $\mathbf{e}$ and the indicators $\mathbf{S}$, where $\mathbf{P} = diag(\mathbf{y})\mathbf{X}$ is given by the training data set standing constant; $\mathbf{S} = \mathbf{S}(\mathbf{e})$ and $\mathbf{B} = \mathbf{B}(\mathbf{e})$ are matrix functions of $\mathbf{e}$, which are defined in (10) and (13), respectively.

As mentioned before, the objective function (6) is convex with regard to $[\mathbf{w}, b] \in \Re^{n+1}$, thus there is a unique optimal solution to $\mathbf{w}$ and b. The sufficient and necessary condition for the global optimal solution is that the gradient vanishes, or equivalently, the equality (16) holds. Solving the optimization problem (6) is therefore equivalent to finding the solution to (16). This is further explained as follows.

Eequation (12) can be viewed as the solution to $\mathbf{w}$ and b for the regularized least-squares problem (6) with the L2-SVM loss for the subset of training points $\{(\mathbf{x}_k, y_k), k = 1, \cdots, N, s(e_k) = 1\}$. These points lie outside the bounding hyperplanes (i.e. $\mathbf{x}_k\mathbf{w} + b < 1$ for $y_k = +1$ and $\mathbf{x}_k\mathbf{w} + b > -1$ for $y_k = -1$) and their violations $e_k = 1 - y_k(\mathbf{x}_k\mathbf{w} + b) > 0$ are penalized by Ce_k^2 in the L2-SVM loss function (7). These training points determining the solution are referred to as the *support vectors*. The other points with $s_k = 0$ lie within the bounding hyperplanes satisfying $e_k = 1 - y_k(\mathbf{x}_k\mathbf{w} + b) \leq 0$ and have no violation to be penalized in the objective function (6), therefore

they need to be ignored. This is accomplished by multiplication with the corresponding $s_k = 0$ in $\mathbf{S}$, which achieves the same effect. As a result, the solution for $\mathbf{S}$ in equation (16) indicates a set of support vectors.

To this end, it should be noted that the definition (10) of the indicator function $s(e)$ at $e = 0$ (either 0 or 1) does not influence the first-order differential of $L(e)$ given in (11). It is demonstrated in the following that this definition has also no influence on the solutions of (12) and (16).

For case of the classifier without bias, we first investigate how the normal vector to the separating hyperplane $\mathbf{w}$ varies as the indicator for a point changes, say s_k for the k'th point $\mathbf{p}_k = y_k\mathbf{x}_k$. Let $\mathbf{I}_k$ denote the matrix with only the k'th diagonal entry being unity while all others being zero. Suppose a change δ_k in s_k is made. Substituting $\mathbf{P}^T(\mathbf{S} + \delta_k\mathbf{I}_k)\mathbf{P} = \mathbf{P}^T\mathbf{S}\mathbf{P} + \delta_k\mathbf{p}_k^T\mathbf{p}_k$ into (17), and applying the Woodbury matrix identity [45]

$$(\mathbf{A} + \mathbf{BCD})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{D}\mathbf{A}^{-1}\mathbf{B} + \mathbf{C}^{-1})^{-1}\mathbf{D}\mathbf{A}^{-1} \tag{19}$$

results in the variation in $\mathbf{w}$ as δ_k, which is given by:

$$\begin{aligned}\mathbf{w}(s_k + \delta_k) &= C(\mathbf{I} + C\mathbf{P}^T\mathbf{S}\mathbf{P} + C\delta_k\mathbf{p}_k^T\mathbf{p}_k)^{-1}(\mathbf{P}^T\mathbf{S}\mathbf{1} + \delta_k\mathbf{p}_k^T) \\ &= \mathbf{w}(s_k) + \frac{\delta_k e_k(s_k)(\mathbf{I} + C\mathbf{P}^T\mathbf{S}\mathbf{P})^{-1}\mathbf{p}_k^T}{\delta_k\mathbf{p}_k(\mathbf{I} + C\mathbf{P}^T\mathbf{S}\mathbf{P})^{-1}\mathbf{p}_k^T + 1}\end{aligned} \tag{20}$$

The corresponding error vector is given by

$$\mathbf{e}(s_k + \delta_k) = 1 - \mathbf{P}\mathbf{w}(s_k + \delta_k) = \mathbf{e}(s_k) - \frac{\delta_k e_k(s_k)\mathbf{P}(\mathbf{I} + C\mathbf{P}^T\mathbf{S}\mathbf{P})^{-1}\mathbf{p}_k^T}{\delta_k\mathbf{p}_k(\mathbf{I} + C\mathbf{P}^T\mathbf{S}\mathbf{P})^{-1}\mathbf{p}_k^T + 1} \tag{21}$$

Now suppose $e_k(s_k) = 0$, then any change δ_k to s_k will not affect either the normal vector $\mathbf{w}$ of the separating hyperplane or the error vector $\mathbf{e}$, thus

the objective function value will also be unaffected. This also reveals that the solution with L2-loss is more sensitive to outliers far away from the separating hyperplanes than to those near to the separating hyperplanes. In contrast, algorithms with L1-loss are more sensitive to outliers near to the separating hyperplanes than to those faraway from them.

In cases where the classifier has a bias, simply augment $\mathbf{P}$ and the identity $\mathbf{I}$ in (17) as follows,

$$\mathbf{P} \leftarrow [\mathbf{y}, \mathbf{P}], \mathbf{I} \leftarrow \begin{bmatrix} 0 & \mathbf{0}^T \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{22}$$

(where the bold type face $\mathbf{0}$ denotes a column of n 0's, the zero diagonal element in the augmented identity matrix means no regularization is applied on the bias b in problem (6)) and one can conclude the same results.

Particularly, the bias will also be unaffected by δ_k in case $e_k(s_k) = 0$, given that the augmented normal vector here includes the bias b as its first component.

4. Nonlinear Generalization

As previously discussed, existing literature on SVMs gives an impression that one has to represent the SVM in the dual in order to introduce nonlinear kernels (i.e., nonlinear generalization). This section reformulates the previous solution for linear SVM by partitioning the training points and correspondingly the error vector. This reformulation gives us a novel way to generalize the linear SVM to nonlinear cases. This generalization may be done merely in the primal RLS context.

4.1. Existing Nonlinear Generalizations

In standard SVM theory, the principal way of introducing a nonlinear kernel is to solve the dual problem of (1), where the classifier (4) can be expressed for any input $\mathbf{x} \in \Re^{1\times n}$ as a linear combination of the inner products of all the support vectors with the input $\mathbf{x}$, possibly with an additional bias b as follows:

$$f(\mathbf{x}) = b + \sum_{i=1}^{n_{SV}} y_i \theta_i \langle \mathbf{x}, \mathbf{x}_i \rangle \tag{23}$$

where $\mathbf{x}_i \in \Re^{1\times n}, i = 1, \cdots, n_{SV}$ denote a set of n_{SV} support vectors with associated labels y_i, respectively, $\langle \mathbf{x}, \mathbf{x}_i \rangle$ the inner product of $\mathbf{x}$ and $\mathbf{x}_i$. For linear SVMs, $\langle \mathbf{x}, \mathbf{x}_i \rangle = \mathbf{x}\mathbf{x}_i^T$, noting that input patterns are represented in row vectors in this paper. If a nonlinear kernel is defined on the n-dimensional input space, for example, the well-known Gaussian kernel:

$$\langle \mathbf{x}, \mathbf{x}_i \rangle = \exp(-\frac{1}{2\sigma^2}(\mathbf{x} - \mathbf{x}_i)(\mathbf{x} - \mathbf{x}_i)^T) \tag{24}$$

we can replace all the inner product expressions of input-pairs from the input space for linear SVMs with the defined nonlinear kernel (which represents the inner product of the maps of the input-pairs in the feature space), resulting in SVMs of nonlinear kernels.

For convenience, the inner products of two sets of points from the input space are denoted hereafter as follows:

$$\langle \mathbf{U}, \mathbf{V} \rangle = \begin{bmatrix} \langle \mathbf{u}_1, \mathbf{v}_1 \rangle & \cdots & \langle \mathbf{u}_1, \mathbf{v}_v \rangle \\ \vdots & & \vdots \\ \langle \mathbf{u}_u, \mathbf{v}_1 \rangle & \cdots & \langle \mathbf{u}_u, \mathbf{v}_v \rangle \end{bmatrix} \tag{25}$$

where $\mathbf{U}$ and $\mathbf{V}$ represent u and v n-dimensional points, respectively, $\mathbf{u}_i$ and

$\mathbf{v}_i$ denote the i'th rows (a point) of $\mathbf{U}$ and $\mathbf{V}$, respectively. For linear kernel, one has the inner product $\langle \mathbf{U}, \mathbf{V} \rangle = \mathbf{U}\mathbf{V}^T$.

Much interest has been directed to solving SVMs in the primal, so avoiding the need to apply any duality theory. However these only apply to the linear case and solve for the normal vector of the separating hyperplane ($\mathbf{w}$), while unable to work out the α's of standard SVMs, for example [25][28]. To extend linear SVM to cases with a nonlinear kernel in the primal, [28] applied in advance a representation theorem, which implies that the optimal normal vector to the separating hyperplane in the higher dimensional feature space can be written as a linear combination of kernel functions that are evaluated at the training samples. By substituting this linear combination for $\mathbf{w}$, the original RLS problem with regard to $\mathbf{w}$ and b is thus converted into an optimization problem to obtain coefficients for the linear combination of kernels. This optimization problem takes the form similar to (55) for all the training points (instead of the support vector set $\mathbf{X}_S$ in (55)) for the L2-SVM loss. To confirm the uniqueness of the solution, the Hessian matrix $2(C\mathbf{K}+\mathbf{KSK})$ must be invertible, the full kernel matrix $\mathbf{K} = \langle \mathbf{X}, \mathbf{X} \rangle$ is therefore assumed invertible.

4.2. Assumptions

In our approach, nonlinear kernels are introduced into SVMs without the need to call on duality theory, or by specifying in advance that the optimal normal vector is a linear combination form of kernel function evaluations. Instead, all derivations are matrix operations in the primal context of RLS. Formula that relate the bias b and the coefficient vector $\theta = [\theta_1, \cdots, \theta_{n_{SV}}]^T$ of the generalized SVM (23) with the support vectors and the associated labels

are obtained, where all support vectors are present only in the inner product form. As a result, it is natural to generalize all the steps in our method into cases for nonlinear SVMs simply by replacing the inner products of point-pairs from the input space with a nonlinear kernel function defined in the input space.

For this generalization, it is assumed that:

a) the full kernel matrix is symmetrical, and

b) any principal minor of the full kernel matrix is positive semi-definite.

These assumptions guarantee that the RLS problem has a unique optimal solution. In addition, matrix $\mathbf{I} + C\mathbf{P}\mathbf{P}^T\mathbf{S}\mathbf{B}$ is invertible in (16) for any $\mathbf{S}$ when the full inner product matrix $\mathbf{P}\mathbf{P}^T$ is replaced in a general form $diag(\mathbf{y})\langle\mathbf{X},\mathbf{X}\rangle diag(\mathbf{y})$ for nonlinear kernels. This will be discussed again later. However the full kernel matrix $\langle\mathbf{X},\mathbf{X}\rangle$ is not required to be invertible in this paper. For an obvious example, the linear case, $\langle\mathbf{X},\mathbf{X}\rangle = \mathbf{X}\mathbf{X}^T$, which is of dimension $N \times N$, is definitely not invertible if $n < N$, but it is certainly positive semi-definite.

In this paper, we use θ's to denote the coefficients of the SVM in (23) to distinguish it from the well known α's (the Lagrange multipliers) in the literature. The θ's here are defined in the primal RLS context of problem (6) and have no restrictions on their values. The α's for standard SVMs are introduced in the dual problem as the Lagrange multipliers and are constrained by $0 \leq \alpha_i \leq C$. Furthermore, only n_{SV} (the number of support vectors, although this is of course not known in advance) θ's are defined here, while N (the number of training points) α's are defined with each applied to a constraint in standard SVM (3).

4.3. Partition of the Training Patterns

Suppose a set of support vectors (SVs) is identified, partition matrix $\mathbf{P}$ and $\mathbf{y}$ into two parts respectively as follows:

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_S \\ \mathbf{P}_0 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} \mathbf{y}_S \\ \mathbf{y}_0 \end{bmatrix} \tag{26}$$

where $\mathbf{P}_S$ and $\mathbf{y}_S$ denote the rows of $\mathbf{P}$ and $\mathbf{y}$, respectively, corresponding to the SVs (for $s_k = 1$), while $\mathbf{P}_0$ and $\mathbf{y}_0$ collect all the other rows (for $s_k = 0$). Then we have the following block matrix:

$$\mathbf{B} = \mathbf{I} - \frac{1}{n_{SV}} \begin{bmatrix} \mathbf{y}_S \\ \mathbf{y}_0 \end{bmatrix} \left[\mathbf{y}_S^T, \mathbf{0}\right] = \begin{bmatrix} \mathbf{I}_S - n_{SV}^{-1}\mathbf{y}_S\mathbf{y}_S^T & \mathbf{0} \\ -n_{SV}^{-1}\mathbf{y}_0\mathbf{y}_S^T & \mathbf{I}_0 \end{bmatrix} \tag{27}$$

and it follows that

$$\mathbf{I} + C\mathbf{P}\mathbf{P}^T\mathbf{S}\mathbf{B} = \begin{bmatrix} \mathbf{I}_S + C\mathbf{P}_S\mathbf{P}_S^T(\mathbf{I}_S - n_{SV}^{-1}\mathbf{y}_S\mathbf{y}_S^T) & \mathbf{0} \\ C\mathbf{P}_0\mathbf{P}_S^T(\mathbf{I}_S - n_{SV}^{-1}\mathbf{y}_S\mathbf{y}_S^T) & \mathbf{I}_0 \end{bmatrix} \tag{28}$$

where the $\mathbf{0}$'s are matrices of zeros of proper size, $n_{SV} = \mathbf{y}^T\mathbf{S}\mathbf{y} = \mathbf{y}_S^T\mathbf{y}_S$ is the number of support vectors, $\mathbf{I}_S$ and $\mathbf{I}_0$ denote identity matrices of proper size.

Denote the following corresponding block matrix

$$(\mathbf{I} + C\mathbf{P}\mathbf{P}^T\mathbf{S}\mathbf{B})^{-1} = \begin{bmatrix} \mathbf{V}_S & \mathbf{Z} \\ \mathbf{V}_0 & \mathbf{U} \end{bmatrix} \tag{29}$$

It then follows from (28) and

$$(\mathbf{I} + C\mathbf{P}\mathbf{P}^T\mathbf{S}\mathbf{B}) \begin{bmatrix} \mathbf{V}_S & \mathbf{Z} \\ \mathbf{V}_0 & \mathbf{U} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_S & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_0 \end{bmatrix} \tag{30}$$

that

$$\left[\mathbf{I}_S + C\mathbf{P}_S\mathbf{P}_S^T(\mathbf{I}_S - n_{SV}^{-1}\mathbf{y}_S\mathbf{y}_S^T)\right]\mathbf{V}_S = \mathbf{I}_S \tag{31}$$

$$\left[\mathbf{I}_S + C\mathbf{P}_S\mathbf{P}_S^T(\mathbf{I}_S - n_{SV}^{-1}\mathbf{y}_S\mathbf{y}_S^T)\right]\mathbf{Z} = \mathbf{0} \tag{32}$$

$$C\mathbf{P}_0\mathbf{P}_S^T(\mathbf{I}_S - n_{SV}^{-1}\mathbf{y}_S\mathbf{y}_S^T)\mathbf{V}_S + \mathbf{V}_0 = \mathbf{0} \tag{33}$$

$$C\mathbf{P}_0\mathbf{P}_S^T(\mathbf{I}_S - n_{SV}^{-1}\mathbf{y}_S\mathbf{y}_S^T)\mathbf{Z} + \mathbf{U} = \mathbf{I}_0 \tag{34}$$

Note that $\mathbf{P}_S\mathbf{P}_S^T$ is always positive semi-definite and, $\mathbf{I}_S - n_{SV}^{-1}\mathbf{y}_S\mathbf{y}_S^T$ has non-negative (one zero and $n_{SV}-1$ unity) eigenvalues, see the comment following equation (13). Scalar $C > 0$ is given in advance. Matrix $\mathbf{I}_S + C\mathbf{P}_S\mathbf{P}_S^T(\mathbf{I}_S - n_{SV}^{-1}\mathbf{y}_S\mathbf{y}_S^T)$ is therefore of full rank and invertible if $n_{SV} > 0$, i.e. there is at least one support-vector. The inverse is given later in (46). This in turn confirms the invertibility of $\mathbf{I} + C\mathbf{P}\mathbf{P}^T\mathbf{S}\mathbf{B}$ in (29). Therefore (30)-(34) are reasonable. It follows from (31) and (33) that

$$\mathbf{V}_S = \left[\mathbf{I}_S + C\mathbf{P}_S\mathbf{P}_S^T(\mathbf{I}_S - n_{SV}^{-1}\mathbf{y}_S\mathbf{y}_S^T)\right]^{-1} \tag{35}$$

$$\mathbf{V}_0 = -C\mathbf{P}_0\mathbf{P}_S^T(\mathbf{I}_S - n_{SV}^{-1}\mathbf{y}_S\mathbf{y}_S^T)\mathbf{V}_S \tag{36}$$

Again, as $\mathbf{I}_S + C\mathbf{P}_S\mathbf{P}_S^T(\mathbf{I}_S - n_{SV}^{-1}\mathbf{y}_S\mathbf{y}_S^T)$ is invertible, from (32) and (34) there must be

$$\mathbf{Z} = \mathbf{0} \tag{37}$$

$$\mathbf{U} = \mathbf{I}_0 \tag{38}$$

Correspondingly, substituting (27), (29), (36), (37) and (38) into (16), the error vector defined in (14) is thus partitioned as follows

$$\mathbf{e} = \begin{bmatrix} \mathbf{e}_S \\ \mathbf{e}_0 \end{bmatrix} = \begin{bmatrix} \mathbf{I}_S - n_{SV}^{-1}\mathbf{y}_S\mathbf{y}_S^T & \mathbf{0} \\ -n_{SV}^{-1}\mathbf{y}_0\mathbf{y}_S^T & \mathbf{I}_0 \end{bmatrix} \begin{bmatrix} \mathbf{V}_S & \mathbf{0} \\ \mathbf{V}_0 & \mathbf{I}_0 \end{bmatrix} \begin{bmatrix} \mathbf{1}_S \\ \mathbf{1}_0 \end{bmatrix} \tag{39}$$

or

$$\mathbf{e} = \begin{bmatrix} (\mathbf{I}_S - n_{SV}^{-1}\mathbf{y}_S\mathbf{y}_S^T)\mathbf{V}_S & \mathbf{0} \\ -(n_{SV}^{-1}\mathbf{y}_0\mathbf{y}_S^T + C\mathbf{P}_0\mathbf{P}_S^T\mathbf{B}_S)\mathbf{V}_S & \mathbf{I}_0 \end{bmatrix} \begin{bmatrix} \mathbf{1}_S \\ \mathbf{1}_0 \end{bmatrix} \tag{40}$$

where and hereafter we denote $\mathbf{B}_S = \mathbf{I}_S - n_{SV}^{-1}\mathbf{y}_S\mathbf{y}_S^T$ for simplicity, $\mathbf{e}_S$ and $\mathbf{e}_0$ are the two parts for unity and zero indicators, respectively. Rewrite the partitioned error vector as

$$\begin{bmatrix} \mathbf{e}_S \\ \mathbf{e}_0 \end{bmatrix} = \begin{bmatrix} \mathbf{V}_S\mathbf{1}_S - n_{SV}^{-1}\mathbf{y}_S\mathbf{y}_S^T\mathbf{V}_S\mathbf{1}_S \\ \mathbf{1}_0 - C\mathbf{P}_0\mathbf{P}_S^T\mathbf{B}_S\mathbf{V}_S\mathbf{1}_S - n_{SV}^{-1}\mathbf{y}_0\mathbf{y}_S^T\mathbf{V}_S\mathbf{1}_S \end{bmatrix} \tag{41}$$

¿From matrix identity (15), we have

$$\mathbf{V}_S = \left[\mathbf{I}_S + C\mathbf{P}_S\mathbf{P}_S^T(\mathbf{I}_S - n_{SV}^{-1}\mathbf{y}_S\mathbf{y}_S^T) \right]^{-1}$$

or

$$\mathbf{V}_S = \mathbf{I}_S - C\mathbf{P}_S\mathbf{P}_S^T(\mathbf{I}_S - n_{SV}^{-1}\mathbf{y}_S\mathbf{y}_S^T)\mathbf{V}_S \tag{42}$$

Substituting (42) into the first term of $\mathbf{e}_S$ in (41), i.e. for the $\mathbf{V}_S$ in term $\mathbf{V}_S\mathbf{1}_S$, it follows that

$$\begin{bmatrix} \mathbf{e}_S \\ \mathbf{e}_0 \end{bmatrix} = \begin{bmatrix} \mathbf{1}_S - C\mathbf{P}_S\mathbf{P}_S^T\mathbf{B}_S\mathbf{V}_S\mathbf{1}_S - n_{SV}^{-1}\mathbf{y}_S\mathbf{y}_S^T\mathbf{V}_S\mathbf{1}_S \\ \mathbf{1}_0 - C\mathbf{P}_0\mathbf{P}_S^T\mathbf{B}_S\mathbf{V}_S\mathbf{1}_S - n_{SV}^{-1}\mathbf{y}_0\mathbf{y}_S^T\mathbf{V}_S\mathbf{1}_S \end{bmatrix} \tag{43}$$

Looking at the second component of equation (43) and comparing it with the definition of the error vector $\mathbf{e}$ in (14), the following equation holds throughout the RLS solution for $\mathbf{w}$ and b given in (12)

$$\mathbf{e}_0 = \mathbf{1}_0 - \mathbf{P}_0\mathbf{w} - \mathbf{y}_0 b \equiv \mathbf{1}_0 - C\mathbf{P}_0\mathbf{P}_S^T\mathbf{B}_S\mathbf{V}_S\mathbf{1}_S - n_{SV}^{-1}\mathbf{y}_0\mathbf{y}_S^T\mathbf{V}_S\mathbf{1}_S$$

where $\mathbf{V}_S$ is given in (35). Note again $\mathbf{B}_S = \mathbf{I}_S - n_{SV}^{-1}\mathbf{y}_S\mathbf{y}_S^T$. Both $\mathbf{V}_S$ and $\mathbf{B}_S$ only depend on the support vectors $\mathbf{P}_S$ and the associated labels $\mathbf{y}_S$, and

are independent from $\mathbf{P}_0$ and $\mathbf{y}_0$. From the previous derivation, (44) holds for any $\mathbf{P}_0$ and $\mathbf{y}_0$. So $\mathbf{w} = C\mathbf{P}_S^T(\mathbf{I}_S - n_{SV}^{-1}\mathbf{y}_S\mathbf{y}_S^T)\mathbf{V}_S\mathbf{1}_S$ must hold. Thus we can define

$$\begin{cases} \mathbf{w} &= \mathbf{P}_S^T\theta \\ \theta &= C(\mathbf{I}_S - n_{SV}^{-1}\mathbf{y}_S\mathbf{y}_S^T)\mathbf{V}_S\mathbf{1}_S \\ b &= n_{SV}^{-1}\mathbf{y}_S^T\mathbf{V}_S\mathbf{1}_S \end{cases} \tag{44}$$

It is obvious that θ and b only depends on the support vectors $\mathbf{P}_S$ and their associated labels $\mathbf{y}_S$. Note that the introduction of θ here is not subject to any constraint on it, such as the $0 \leq \alpha_i \leq C$ that would apply in the case of the Lagrange multiplier α_i's for standard SVMs [3][10]. It is simply based on the fact that the definition of θ in (44) enables the linear SVM (4) to take the form $f(\mathbf{x}) = \mathbf{x}\mathbf{P}_S^T\theta + b$ for any input $\mathbf{x}$, say any row from $\mathbf{P}_S$ or $\mathbf{P}_0$ as shown in the two component equations of (43). In this form, the SVM is a linear combination of inner products of the input $\mathbf{x}$ with $\mathbf{P}_S$. This makes it easy to generalize the linear SVM (4) into (23) for nonlinear kernels.

To simplify the computation and to analyze the properties of the solution to θ and b, (44) is further simplified as follows.

Using a matrix identity (15) again, we have

$$(\mathbf{I}_S + C\mathbf{P}_S\mathbf{P}_S^T)^{-1}\mathbf{P}_S\mathbf{P}_S^T = C^{-1}\mathbf{I}_S - C^{-1}(\mathbf{I}_S + C\mathbf{P}_S\mathbf{P}_S^T)^{-1} \tag{45}$$

Noting the number of support vectors $n_{SV} = \mathbf{y}_S^T\mathbf{y}_S$, it follows from (45) that

$$\mathbf{y}_S^T(\mathbf{I}_S + C\mathbf{P}_S\mathbf{P}_S^T)^{-1}\mathbf{P}_S\mathbf{P}_S^T\mathbf{y}_S - n_{SV}C^{-1} = -C^{-1}\mathbf{y}_S^T(\mathbf{I}_S + C\mathbf{P}_S\mathbf{P}_S^T)^{-1}\mathbf{y}_S$$

Along with matrix identity (19), $\mathbf{V}_S$ given in (35) is simplified as follows.

$$\begin{aligned}
\mathbf{V}_S &= [\mathbf{I}_S + C\mathbf{P}_S\mathbf{P}_S^T(\mathbf{I}_S - n_{SV}^{-1}\mathbf{y}_S\mathbf{y}_S^T)]^{-1} \\
&= \left[\mathbf{I}_S - \frac{(\mathbf{I}_S + C\mathbf{P}_S\mathbf{P}_S^T)^{-1}\mathbf{P}_S\mathbf{P}_S^T\mathbf{y}_S\mathbf{y}_S^T}{\mathbf{y}_S^T(\mathbf{I}_S + C\mathbf{P}_S\mathbf{P}_S^T)^{-1}\mathbf{P}_S\mathbf{P}_S^T\mathbf{y}_S - n_{SV}C^{-1}}\right](\mathbf{I}_S + C\mathbf{P}_S\mathbf{P}_S^T)^{-1} \\
&= \left[\mathbf{I}_S - \frac{\mathbf{y}_S\mathbf{y}_S^T - (\mathbf{I}_S + C\mathbf{P}_S\mathbf{P}_S^T)^{-1}\mathbf{y}_S\mathbf{y}_S^T}{\mathbf{y}_S^T(\mathbf{I}_S + C\mathbf{P}_S\mathbf{P}_S^T)^{-1}\mathbf{y}_S}\right](\mathbf{I}_S + C\mathbf{P}_S\mathbf{P}_S^T)^{-1} \qquad (46)
\end{aligned}$$

As $\mathbf{P}_S\mathbf{P}_S^T$ is symmetric and positive semi-definite and $C > 0$, $\mathbf{I}_S + C\mathbf{P}_S\mathbf{P}_S^T$ is always symmetric and positive definite and therefore $\mathbf{y}_S^T(\mathbf{I}_S + C\mathbf{P}_S\mathbf{P}_S^T)^{-1}\mathbf{y}_S > 0$ holds for any nonzero $\mathbf{y}_S$, given that $\mathbf{y}_S$ here is a column of 1's and -1's. Substituting (46) into (44) for θ and b, results in

$$\begin{cases}
\theta &= (C^{-1}\mathbf{I}_S + \mathbf{P}_S\mathbf{P}_S^T)^{-1}(\mathbf{1}_S - b\mathbf{y}_S) \\
b &= \frac{\mathbf{y}_S^T(C^{-1}\mathbf{I}_S + \mathbf{P}_S\mathbf{P}_S^T)^{-1}\mathbf{1}_S}{\mathbf{y}_S^T(C^{-1}\mathbf{I}_S + \mathbf{P}_S\mathbf{P}_S^T)^{-1}\mathbf{y}_S}
\end{cases} \qquad (47)$$

4.4. Nonlinear Generalization

¿From (47) it is obvious that only then support vectors ($\mathbf{P}_S$) and the associated labels ($\mathbf{y}_S$) are involved in the solution. Note the notation $\mathbf{P} = diag(\mathbf{y})\mathbf{X}$ introduced in (8) and (9), and that the labels in $\mathbf{y}$ are either 1 or -1. Denoting $\mathbf{Y}_S = diag(\mathbf{y}_S)$ as the diagonal matrix with $\mathbf{y}_S$ being the diagonals, there have $\mathbf{P}_S\mathbf{P}_S^T = \mathbf{Y}_S\mathbf{X}_S\mathbf{X}_S^T\mathbf{Y}_S$, $\mathbf{Y}_S\mathbf{y}_S = \mathbf{1}_S$, $\mathbf{Y}_S\mathbf{1}_S = \mathbf{y}_S$ and $\mathbf{Y}_S\mathbf{Y}_S = \mathbf{I}_S$. Rewriting (47) with the original data $\mathbf{Y}_S\mathbf{X}_S$ instead of $\mathbf{P}_S$, and making the substitution $\mathbf{X}_S\mathbf{X}_S^T \leftarrow \langle\mathbf{X}_S, \mathbf{X}_S\rangle$, results in the following solution for general SVMs with linear or nonlinear kernels:

$$\begin{cases}
\theta &= \mathbf{Y}_S(C^{-1}\mathbf{I}_S + \langle\mathbf{X}_S, \mathbf{X}_S\rangle)^{-1}(\mathbf{y}_S - b\mathbf{1}_S) \\
b &= \frac{\mathbf{1}_S^T(C^{-1}\mathbf{I}_S + \langle\mathbf{X}_S, \mathbf{X}_S\rangle)^{-1}\mathbf{y}_S}{\mathbf{1}_S^T(C^{-1}\mathbf{I}_S + \langle\mathbf{X}_S, \mathbf{X}_S\rangle)^{-1}\mathbf{1}_S}
\end{cases} \qquad (48)$$

Correspondingly, the normal vector $\mathbf{w}$ in (44) to the separating hyperplane (4) is rewritten (for then linear case) as

$$\mathbf{w} = \mathbf{X}_S^T \mathbf{Y}_S \theta = \mathbf{X}_S (C^{-1}\mathbf{I}_S + \langle \mathbf{X}_S, \mathbf{X}_S \rangle)^{-1} (\mathbf{y}_S - b\mathbf{1}_S)$$

and the classifier (4) is rewritten for generalized kernels as:

$$f(\mathbf{x}) = \langle \mathbf{x}, \mathbf{X}_S \rangle \mathbf{Y}_S \theta + b \tag{49}$$

For cases without bias, simply let $b = 0$ in (48), and thus

$$\begin{cases} \theta &= \mathbf{Y}_S (C^{-1}\mathbf{I}_S + \langle \mathbf{X}_S, \mathbf{X}_S \rangle)^{-1} \mathbf{y}_S \\ b &= 0 \end{cases} \tag{50}$$

4.5. Properties of the Solution

It is of interest to compare the properties of θ here with those of the α's for standard SVMs, which are well known in the literature, and are introduced as Lagrange multipliers and discussed in the context of duality theory. Firstly, noting $\mathbf{y}_S^T \mathbf{Y}_S = \mathbf{1}_S^T$, it is easily checked from (48) that, for cases with the bias b, the following equation holds

$$\sum_{s_i=1} y_i \theta_i = \mathbf{y}_S^T \theta = \mathbf{1}_S^T (C^{-1}\mathbf{I}_S + \langle \mathbf{X}_S, \mathbf{X}_S \rangle)^{-1} (\mathbf{y}_S - b\mathbf{1}_S) = 0 \tag{51}$$

where the summation over $s_i = 1$ means sum over all support vectors, the θ_i's are the components of θ. Secondly, from (49), it holds that

$$\mathbf{w} = \sum_{s_i=1} y_i \theta_i \mathbf{x}_i^T \tag{52}$$

In addition, by comparing θ in (44) with (41), $\mathbf{e}_S$ (the upper part of the matrix partition in (41)) an interesting relationship between $\mathbf{e}_S$ and θ is evident as follows:

$$\theta = C\mathbf{e}_S \tag{53}$$

This shows that the coefficient vector θ for a support vector machine is C times the value of the partial error vector for the support vectors. As discussed before, for the global solution, the errors for the support vectors are positive and therefore the following inequalities hold:

$$\theta_i > 0, i = 1, \cdots, N, s_i = 1 \tag{54}$$

Properties defined by equations (51), (52) and (54) are exactly the same as those of the α_i's as described in textbooks and articles on SVMs, see for example, [10][3].

With regard to property (53), it could be noted that a similar property is derived by [18] for least squares SVMs from the Lagrangian function, where the inequality constraints (3) are replaced by equalities. Because of this replacement, the sparseness is lost in such a way that points with negative errors (meaning that the points do not violate the constraints (3)) may be included in the resulted SVM because $\alpha_i = Ce_i < 0$ (instead of zero) [18][46]. The spareness of the SVMs obtained using our method is highlighted by example in section 6.

Of particular interest, as previously shown in (20) and (21) that for points with zero errors, the values of the corresponding indicators are irrelevant, any arbitrary choice (either 1 or 0) can be made and this will not influence the solution. This conclusion is more clearly confirmed by property (53): support vectors (if they are set) with vanishing errors have zero coefficients in the final SVM (49). Furthermore, it can be concluded that the coefficient of a support vector in the SVM (49) is proportional to the violation of the support vector in the corresponding constraint (3).

Finally, using (49) and replacement $\mathbf{X}_S\mathbf{X}_S^T \leftarrow \langle \mathbf{X}_S, \mathbf{X}_S \rangle$, the objective

function (6) with the L2-SVM loss function can be rewritten as

$$J(\theta, b) = \theta^T \mathbf{Y}_S \langle \mathbf{X}_S, \mathbf{X}_S \rangle \mathbf{Y}_S \theta + C\mathbf{e}^T \mathbf{S}(\mathbf{e})\mathbf{e} \tag{55}$$

where diagonal matrix $\mathbf{S}(\mathbf{e})$ is evaluated using (10) for the full error vector $\mathbf{e}$. The loss term $\mathbf{e}^T\mathbf{S}(\mathbf{e})\mathbf{e}$ is thus the sum of squares of all the positive errors (violations). For an algorithmic procedure, vector $\mathbf{w}$ in the feature space is generally numerically unavailable, as it is of indefinite or unknown dimensions in general cases, Equation (55) can be useful to monitor the value of the objective function during optimization.

Note again that this solution guarantees that the errors for support vectors are positive (with unity diagonal entries for $\mathbf{S}$) while for other points the errors are negative (with zero diagonal entries for $\mathbf{S}$). Based on (53), the minimal value of the objective function is thus given by

$$\begin{aligned} J(\hat{\theta}, \hat{b}) &= \hat{\theta}^T \mathbf{Y}_S (C^{-1}\mathbf{I}_S + \langle \mathbf{X}_S, \mathbf{X}_S \rangle) \mathbf{Y}_S \hat{\theta} \\ &= \mathbf{y}_S^T (C^{-1}\mathbf{I}_S + \langle \mathbf{X}_S, \mathbf{X}_S \rangle)^{-1} (\mathbf{y}_S - b\mathbf{1}_S) = \mathbf{1}_S^T \hat{\theta} = \sum_{i=1}^{n_{SV}} \hat{\theta}_i \end{aligned} \tag{56}$$

where $\hat{\theta}$ denotes the optimal solution to θ of components $\hat{\theta}_i$'s.

5. An Iterative Algorithm

As discussed in section 3, solving the optimization problem (6) subject to the L2-SVM loss (7) is equivalent to solving the nonlinear equation (16) for $\mathbf{S}$ and $\mathbf{e}$ subject to (10). However, this equation is difficult to solve analytically, given that $\mathbf{S} = \mathbf{S}(\mathbf{e})$ involves indicator functions of the error vector $\mathbf{e}$ which are not continuous. In this paper we simply demonstrate the effectiveness

of our formulation with an iterative algorithm to solve problem (6). The iteration scheme is constructed directly from (16) as follows:

$$\begin{cases} \mathbf{B}^{\tau)} &= \mathbf{I} - (\mathbf{y}^T\mathbf{S}^{\tau)}\mathbf{y})^{-1}\mathbf{y}\mathbf{y}^T\mathbf{S}^{\tau)} \\ \mathbf{e}^{\tau+1)} &= \varepsilon(\mathbf{S}^{\tau)}) = (\mathbf{I} + C\mathbf{B}^{\tau)}\mathbf{Y}\langle\mathbf{X},\mathbf{X}\rangle\mathbf{Y}\mathbf{S}^{\tau)})^{-1}\mathbf{B}^{\tau)}\mathbf{1} \\ \mathbf{S}^{\tau+1)} &= \mathbf{S}(\mathbf{e}^{\tau+1)}) \\ \mathbf{S}^{0)} &= \mathbf{I}, \tau = 0, 1, 2, \cdots \end{cases}$$

where $\mathbf{Y} = diag(\mathbf{y})$ and $\mathbf{Y}\langle\mathbf{X},\mathbf{X}\rangle\mathbf{Y}$ is the generalization of $\mathbf{P}\mathbf{P}^T$ for non-linear kernels. The notation $\varepsilon(\mathbf{S}^{\tau)}) = (\mathbf{I} + C\mathbf{B}^{\tau)}\mathbf{Y}\langle\mathbf{X},\mathbf{X}\rangle\mathbf{Y}\mathbf{S}^{\tau)})^{-1}\mathbf{B}^{\tau)}\mathbf{1}$ is to highlight that the right-hand side is a vector function of $\mathbf{S}$, and the partially parenthesized super subscripts for $\mathbf{S}$, $\mathbf{B}$ and $\mathbf{e}$ denote the iteration count. Given initially $\mathbf{S}^{0)} = \mathbf{I}$, iterate for $\tau = 0, 1, 2\cdots$ until the indicators $\mathbf{S}$ do not change, i.e. $\mathbf{S}^{\tau+1)} = \mathbf{S}^{\tau)}$. In this case, $\mathbf{S}$ indicates a set of support vectors as previously discussed: those training points having unity indicators are support vectors.

However directly implementing algorithm (57) involves the inversion of an $N \times N$ matrix for each iteration. To reduce the computation, we re-order the matrix according to the values of the indicators $\mathbf{S}^{\tau)}$. This allows us to partition the matrix when solving for θ and b using (48).

To see how this is done, we first (for convenience) rewrite the error vector (43) as:

$$\begin{bmatrix} \mathbf{e}_S^{\tau+1)} \\ \mathbf{e}_0^{\tau+1)} \end{bmatrix} = \begin{bmatrix} \theta^{\tau+1)}/C \\ \mathbf{1}_0 - \mathbf{Y}_0\langle\mathbf{X}_0,\mathbf{X}_S\rangle\mathbf{Y}_S\theta^{\tau+1)} - b^{\tau+1)}\mathbf{y}_0 \end{bmatrix} \tag{57}$$

where $\theta^{\tau+1)}$ and $b^{\tau+1)}$ are updated using (48) for $\mathbf{S} = \mathbf{S}^{\tau)}$. For any given set of indicators, say $\mathbf{S}^{\tau)}$, the training data points $\mathbf{X}$ and the associated labels $\mathbf{y}$

are partitioned into two parts: $\mathbf{X}_S$ with the associated labels $\mathbf{y}_S$ corresponds to points having unity indicators in $\mathbf{S}^{\tau)}$. $\mathbf{X}_0$ is associated with labels $\mathbf{y}_0$ corresponding to zero indicators in $\mathbf{S}^{\tau)}$. As discussed before, the necessary and sufficient condition for a given indicator $\mathbf{S}^{\tau)}$ to be optimal is that the partitioned error vector (44) satisfies $\mathbf{e}_S^{\tau+1)} > 0$ and $\mathbf{e}_0^{\tau+1)} \leq 0$ with $\theta^{\tau+1)}$ and $b^{\tau+1)}$ being determined using (48) for $\mathbf{S} = \mathbf{S}^{\tau)}$. The terms $\langle \mathbf{X}_S, \mathbf{X}_S \rangle$ and $\langle \mathbf{X}_0, \mathbf{X}_S \rangle$ are calculated in a predefined kernel.

Based on this, an iteration algorithm is constructed as follows:

Step 1) Initially set the indicators $s_i^{0)} = 1, i = 1, \cdots, N$ for a given set of N training points represented in $\mathbf{X}$ (with each row representing a point. Set the associated label $\mathbf{y}$ (a column of 1's and -1's), and iteration number $\tau = 0$.

Step 2) Partition the training data as follows: $\mathbf{X}_S^{\tau)}$ and $\mathbf{y}_S^{\tau)}$ collect those rows of $\mathbf{X}$ and $\mathbf{y}$, for $s_i^{\tau)} = 1, i = 1, \cdots, N$, respectively, while $\mathbf{X}_0^{\tau)}$ and $\mathbf{y}_0^{\tau)}$ collect rows for $s_i^{\tau)} = 0, i = 1, \cdots, N$.

Step 3) Compute $\theta_S^{\tau+1)}$ and $b^{\tau+1)}$ using (48) for $\mathbf{X}_S^{\tau)}$ and $\mathbf{y}_S^{\tau)}$. Compute the error vector using (49) for $\theta_S^{\tau+1)}$ and $b^{\tau+1)}$ as $e_i^{\tau+1)} = 1 - y_i \langle \mathbf{x}_i, \mathbf{X}_S^{\tau)} \rangle \mathbf{Y}_S^{\tau)} \theta^{\tau+1)} - b^{\tau+1)} y_i$ for $i = 1, \cdots, N$. Note that for those points with a unity indicator, the errors $e_S^{\tau+1)}$ can be computed simply by dividing $\theta^{\tau+1)}$ by C as shown in (57). Alternatively, directly take the value of $\theta^{\tau+1)}$ since only the signs of the errors are concerned here rather than their magnitudes.

Step 4) Update the indicators according to (10): for $i = 1, \cdots, N$, let $s_i^{\tau+1)} = 1$ if $e_i^{\tau+1)} > 0$ otherwise let $s_i^{\tau+1)} = 0$, where $e_i^{\tau+1)}$ is the i'th element of the updated error vector $\mathbf{e}^{\tau+1)}$.

Step 5) Check for termination. If $s_i^{\tau+1)} = s_i^{\tau)}$ for $i = 1, \cdots, N$, stop the itera-

tion, otherwise let $\tau \leftarrow \tau + 1$ and go to step 2.

In this algorithm, the inverse of an $n_{SV} \times n_{SV}$ matrix $C^{-1}\mathbf{I}_S + \langle \mathbf{X}_S, \mathbf{X}_S \rangle$ is required at each iteration, see (48). This is inefficient and impractical for large n_{SV}. For cases of SVMs with linear kernels and $n < n_{SV}$, this inverse can be computed as $(C^{-1}\mathbf{I}_S + \mathbf{X}_S\mathbf{X}_S^T)^{-1} = C\mathbf{I}_S - C\mathbf{X}_S(C^{-1}\mathbf{I} + \mathbf{X}_S^T\mathbf{X}_S)^{-1}\mathbf{X}_S^T$ by applying matrix identity (15), where $C^{-1}\mathbf{I}+\mathbf{X}_S^T\mathbf{X}_S$ is an $n \times n$ matrix with n being now only the number of dimensions of the input space. In this case, the computational complexity of this algorithm depends only on $\min(n, n_{SV})$.

5.1. Cholesky Decomposition

For cases of SVMs with nonlinear kernels or when $n \geq n_{SV}$, to avoid the inversion of a large matrix in (48), one can solve the following two linear systems

$$(C^{-1}\mathbf{I}_S + \langle \mathbf{X}_S, \mathbf{X}_S \rangle)\mathbf{u} = \mathbf{1}_S(C^{-1}\mathbf{I}_S + \langle \mathbf{X}_S, \mathbf{X}_S \rangle)\mathbf{v} = \mathbf{y}_S \tag{58}$$

for $\mathbf{u}$ and $\mathbf{v}$. Then the solution for θ and $\mathbf{b}$ in (48) is given by

$$\begin{cases} \theta &= \mathbf{Y}_S(\mathbf{v} - b\mathbf{u}) \\ b &= \mathbf{1}_S^T\mathbf{v}/\mathbf{1}_S^T\mathbf{u} \end{cases} \tag{59}$$

To see the importance of this, consider again that $(C^{-1}\mathbf{I}_S + \langle \mathbf{X}_S, \mathbf{X}_S \rangle)$ has real entries and is symmetric and positive definite, thus it can be uniquely decomposed as

$$C^{-1}\mathbf{I}_S + \langle \mathbf{X}_S, \mathbf{X}_S \rangle = \mathbf{L}\mathbf{D}\mathbf{L}^T \tag{60}$$

where $\mathbf{L}$ is a lower triangular matrix with unity diagonal entries, $\mathbf{D}$ is a diagonal matrix with strictly positive diagonal entries. The diagonals of $\mathbf{D}$

and entries of $\mathbf{L}$ below its diagonal are given recursively by

$$\begin{cases} D_j &= A_{j,j} - \sum_{k=1}^{j-1} L_{j,k}^2 D_k,\ j = 1, \cdots, n_{SV} \\ L_{i,j} &= \frac{1}{D_j}\left(A_{i,j} - \sum_{k=1}^{j-1} L_{i,k} L_{j,k} D_k\right),\ i > j \end{cases} \tag{61}$$

where $A_{i,j}$ and $L_{i,j}$ denote the entries (i, j) of $\mathbf{A} = C^{-1}\mathbf{I}_S + \langle \mathbf{X}_S, \mathbf{X}_S \rangle$ and $\mathbf{L}$, respectively, D_j the j'th diagonal entry of $\mathbf{D}$. This is a Cholesky decomposition [47], resulting in

$$\begin{cases} \mathbf{u} &= (C^{-1}\mathbf{I}_S + \langle \mathbf{X}_S, \mathbf{X}_S \rangle)^{-1}\mathbf{1}_S = (\mathbf{L}\mathbf{D}\mathbf{L}^T)^{-1}\mathbf{1}_S \\ \mathbf{v} &= (C^{-1}\mathbf{I}_S + \langle \mathbf{X}_S, \mathbf{X}_S \rangle)^{-1}\mathbf{y}_S = (\mathbf{L}\mathbf{D}\mathbf{L}^T)^{-1}\mathbf{y}_S \end{cases} \tag{62}$$

which can be computed by forward and backward substitution.

5.2. Permutation of The Support Vectors

Note that there are always some common vectors in the support vector sets of two sequential steps during the previous iteration process. Particularly in the later stage (near convergence), only a few of the support vector set are updated during each step. This fact makes it possible to improve the computational efficiency of the iteration by partially reusing the Cholesky factor matrices $\mathbf{L}$ and $\mathbf{D}$.

Suppose two sets of support vectors denoted as, $S^{1)} = \{s_1^{1)}, \cdots, s_{n_{SV1}}^{1)}\}$ and $S^{2)} = \{s_1^{2)}, \cdots, s_{n_{SV2}}^{2)}\}$ of n_{SV_1} and n_{SV_2} support vectors, respectively, with $1 \le s_j^{i)} \le N, i = 1, 2, j = 1, \cdots, n_{SV_i}$ denoting the index numbers of the support vectors (i.e., the row numbers in the training set $\mathbf{X}$). There are q common support vectors in the two sets. The set $S^{2)}$ has been sorted so that the first q support vectors, $s_1^{2)}, \cdots, s_q^{2)}$, are present in set $S^{1)}$. It can be arranged that the first q elements of $S^{1)}$ are identical to that of $S^{2)}$ by permuting the n_{SV_1} elements of $S^{1)}$. Corresponding to this permutation, the

Cholesky factor matrices need to be updated. Permutation of the support vectors in $S^{1)}$ and the corresponding update to the Cholesky factor matrices are investigated in the following.

Suppose the Cholesky factor matrices for support vectors $S^{1)}$ are computed, and denoted as $\mathbf{L}^{1)}$ and $\mathbf{D}^{1)}$, which satisfy $C^{-1}\mathbf{I}_{S^{1)}} + \langle \mathbf{X}_{S^{1)}}, \mathbf{X}_{S^{1)}} \rangle = \mathbf{L}^{1)}\mathbf{D}^{1)}\mathbf{L}^{1)\,T}$. Now investigate the effect of switching two consequent support vectors $s_p^{1)}$ and $s_{p+1}^{1)}$ in set $S^{1)}$, then, both the p'th and the $p+1$'th support vectors (rows) of $\mathbf{X}_{S^{1)}}$ are switched. For simplicity, denote matrix $\mathbf{A}^{1)} = C^{-1}\mathbf{I}_{S^{1)}} + \langle \mathbf{X}_{S^{1)}}, \mathbf{X}_{S^{1)}} \rangle$ and $\widehat{\mathbf{A}}^{1)} = C^{-1}\mathbf{I}_{S^{1)}} + \langle \mathbf{U}_m\mathbf{X}_{S^{1)}}, \mathbf{U}_m\mathbf{X}_{S^{1)}} \rangle$, where the $\mathbf{U}_m$ denotes the $n_{SV_1} \times n_{SV_1}$ permutation matrix that switches the p'th and the $p+1$'th rows of $\mathbf{X}_{S^{1)}}$, of which the entries $U_{p,p+1} = U_{p+1,p} = U_{ii} = 1, i = 1, \cdots, p-1, p+1, \cdots, n_{SV_1}$ while all the others are zero. Noting the symmetrical property of the kernel function, i.e., $\langle \mathbf{x}_i, \mathbf{x}_j \rangle = \langle \mathbf{x}_j, \mathbf{x}_i \rangle$, we have

$$\widehat{\mathbf{A}}^{1)} = \mathbf{U}_m\mathbf{A}^{1)}\mathbf{U}_m \tag{63}$$

Denote the Cholesky factor matrices of the permuted matrix $\widehat{\mathbf{A}}^{1)}$ as $\widehat{\mathbf{L}}^{1)}$ and $\widehat{\mathbf{D}}^{1)}$, i.e., $\widehat{\mathbf{A}}^{1)} = \widehat{\mathbf{L}}^{1)}\widehat{\mathbf{D}}^{1)}\widehat{\mathbf{L}}^{1)\,T}$. With relationship (63), it is not difficult to determine from (61) that only the entries in rows p and $p+1$ and columns p and $p+1$ of the Cholesky factor matrices are changed, while all other entries remain unchanged. In addition, $\widehat{\mathbf{L}}^{1)}$ and $\widehat{\mathbf{D}}^{1)}$ may be computed from $\mathbf{L}^{1)}$ and

$\mathbf{D}^{1)}$ as follows.

$$\begin{cases} \widehat{D}_p &= D_{p+1} + \mu L_{p+1,p} \\ \widehat{L}_{p+1,p} &= \widehat{D}_p^{-1}\mu \\ \widehat{D}_{p+1} &= D_p - \widehat{\mu}\widehat{L}_{p+1,p} \\ \widehat{L}_{p,j} &= L_{p+1,j}, \quad \widehat{L}_{p+1,j} = L_{p,j}, \quad j < p \\ \widehat{L}_{i,p} &= \widehat{D}_p^{-1}(D_{p+1}L_{i,p+1} + \mu L_{i,p}) \\ \widehat{L}_{i,p+1} &= \widehat{D}_{p+1}^{-1}(D_p L_{i,p} - \widehat{\mu}\widehat{L}_{i,p}), \quad i > p+1 \\ \mu &= D_p L_{p+1,p}, \quad \widehat{\mu} = \widehat{D}_p \widehat{L}_{p+1,p} \end{cases} \tag{64}$$

where for simplicity $L_{i,j}$ and D_p denote the entries of the original Cholesky factor matrices $\mathbf{L}^{1)}$ and $\mathbf{D}^{1)}$ (before switching support vectors p and $p+1$), $\widehat{L}_{i,j}$ and $\widehat{D}_p$ denote the corresponding updated entries of $\widehat{\mathbf{L}}^{1)}$ and $\widehat{\mathbf{D}}^{1)}$ (due to the switching).

The steps specified in (64) that update the Cholesky factor matrices in correspondence with the permutation of the support vectors can be performed without any evaluation of the kernel function. To compute the Cholesky factor matrices for the q common support vectors in $S^{1)}$ and $S^{2)}$, one can permute the support vectors in $S^{1)}$ by the same series of switching operations that are used to update the Cholesky factor matrices using (64). Other entries of rows $q+1$ to n_{SV_2} for the other $n_{SV_2} - q$ support vectors of $S^{2)}$ can then be computed using (61).

5.3. Complexity Analysis and Implementation

Looking at (64), corresponding to each switching operation of two sequential support vectors, say the p'th and the $p+1$'th of a support vector set of size n_{SV}, $8(n_{SV} - p)$ floating-point operations (FPOs, including addition/subtraction and multiplication/division and comparison operations of

floating-point numbers) are required to update the two columns of $\mathbf{L}$ and the two diagonal entries of $\mathbf{D}$, i.e., $2(n_{SV}-p)+1$ entries in total. The two rows p and $p+1$ of $\mathbf{L}$ from columns 1 to $p-1$ are switched (without using any FPOs), while all other entries are unchanged.

For a worst case example (which will never occur in practice), that revises the order of a set of support vectors of size n_{SV}, it would be necessary to iterate the switching operation for p from $n_{SV}-1$ to k and k from 1 to $n_{SV}-1$, requiring $\frac{4}{3}n_{SV}(n_{SV}-1)(n_{SV}+1)$ FPOs. By comparison, to fully compute the same Cholesky factor matrices using (61) requires $\frac{1}{2}n_{SV}(n_{SV}-1)(n_{SV}+2)$ FPOs. For this worst case scenario, the computational burden of using the switching technique (64), is about $\frac{8}{3}$ times of that of Cholesky decomposition (61). Note that computation of the kernel matrix for $C^{-1}\mathbf{I}_S + \langle \mathbf{X}_S, \mathbf{X}_S \rangle$ is not counted here, but it is required in the method that uses (61), and is of order $O(nn_{SV}^2)$ FPOs, where n is the number of dimensions of the training vectors.

When implementing this algorithm, for a given support vector set of size n_{SV}, only one memory buffer block is required to store the lower triangular part of matrix $\mathbf{A} = C^{-1}\mathbf{I}_S + \langle \mathbf{X}_S, \mathbf{X}_S \rangle$. Once $\mathbf{A}$ is computed (with its lower triangular part), the Cholesky factor matrices can be computed using (61), which overwrite $\mathbf{A}$ with the Cholesky factor $\mathbf{D}$ overwriting the diagonals. The unity diagonal entries of $\mathbf{L}$ needs not to be stored.

Using $\mathbf{L}$ and $\mathbf{D}$, vectors $\mathbf{u}$ and $\mathbf{v}$ (62) can be computed using forward and backward substitution, and from them b and θ using (59). By evaluating the errors, a new support vector set can be identified, see Step 3) of the iterative algorithm. To reuse the existing $\mathbf{L}$ and $\mathbf{D}$, re-sort the new support vector

set with the common support vectros at the start. Assuming q common support vectors are identified, permutate the original support vector set such that the q common support vectors are also at the beginning, and update $\mathbf{L}$ and $\mathbf{D}$ correspondingly. The top rows(1 to q) of the updated Cholesky factor matrices for the new support vector set is thus obtained for the q common support vectors. the bottom rows ($q+1$ and below) is irrelevant and the memory buffer can be used to store the corresponding part of matrix $C^{-1}\mathbf{I}_S + \langle \mathbf{X}_S, \mathbf{X}_S \rangle$ by overwriting. The bottom part of the updated $\mathbf{L}$ and $\mathbf{D}$ can then be computed using (61).

This technique of reusing the Cholesky factor matrices of the previous step can greatly reduce the computational burden in both factorizing using (61) and kernel function evaluations. The full Cholesky factorization (61) is only performed in the first iteration. In the later stage of the iteration, the support vector set is approaching to the solution, only a small part of it is changed in each iteration. In addition, this algorithm converges in a few iterations (often no more than 10 cycles.) Some problems might result in systems where it takes longer to converge but we do not explore the possibility of terminating the iterative process early in a trade off of accuracy versus speed of convergence. Combining previous analysis, the overall computational complexity is of order $O(n_{SV}^2(n_{SV}+N+n))$.

Obviously the amount of memory required to store the kernel matrix and the Cholesky factor matrices in this algorithm is $\frac{1}{2}n_{SVMax}^2$, where n_{SVMax} is the maximum size of the support vector sets during the iteration, which is unknown in advance. One can simply let $n_{SVMax} = N$. However this assumption is inefficient, even impractical, for large data set. According to the

previous analysis, the memory requirement increases quadratically (n_{SV}^2) and the computational complexity required to solve for b and θ using Cholesky decomposition increased cubically (n_{SV}^3) with the size of the support vector set.

To overcome this problem, n_{SVMax} can be set in advance at some value that is not smaller than the size of the final solution support vector set. That is, no more than n_{SVMax} support vectors are selected in each iteration. Instead of selecting all the training support vectors of positive error values, only n_{SVMax} at most will be selected in decreasing order of the errors. This may not only reduce the memory requirement, but also improve the computational efficiency, as the number of potential support vectors (of positive error values) during the iteration can be much greater than the size of the final solution, particularly for large data set in the initial stage of the iteration.

When starting with $n_{SVMax} < N$ the initial support vector set can be drawn from the training vector set using stratified sampling strategies on the two classes. Instead of simply taking all the training vectors as mentioned in algorithm Step 1) A uniform fraction of vectors are drawn from each of the two classes (strata), and thus the numbers of initial support vectors drawn from each class is proportional to the number of training vectors belonging to that class.

Since the iterative process will work towards a final set of support vectors there is no loss of accuracy over other methods. There are a few cases where it does take a larger number of iterations to arrive at a suitable support vector set. Our broad spectrum applicability, with its ability to converge in situations where other methods do not, mitigates the few cases where

convergence is slower. There is no winner takes all SVM technique and our approach demonstrates a method that offers some potential to obtain a support vector set when other techniques fail.

6. Simulation Examples

In this section, our SVM solution technique, derived in the RLS context (denoted as RLSSVM here after), is demonstrated for linear and nonlinear kernels. The proposed iterative algorithm has been implemented in C code and can be compared in terms of execution time with other algorithms coded in C or C++. The algorithm of [28] for solving (6) in the primal (referred to as Primal[1]), and two well-known packages, LIBSVM[2][48] and SVMLight[3][49] for SVM are employed for comparisons. These software packages are not the most recently available but they are exhaustively tested and known to produce accurate results. We test our algorithm here mainly for accuracy, robustness and range of applicability. Note that the LIBSVM implements an SMO algorithm to solve the constrained problem (1) [50], while SVMLight implements a chunking optimization method.

[1]MATLAB code downloaded at

http://www.kyb.tuebingen.mpg.de/bs/people/chapelle/primal/

[2]C++ code package downloaded at

http://www.csie.ntu.edu.tw/∼cjlin/libsvm

[3]C code package downloaded at

http://www.cs.cornell.edu/People/tj/svm_light/index.html

Table 1: Results of the four Algorithms on the Simple Example

	RLSSVM	**Primal**	**LIBSVM**	**SVMLight**
SVs	2,3	2,3	2,3	2,3
SVM-	0.9999	0.9999	1.0000	1.0000
coefficients	-0.9999	-0.9999	-1.0000	-1.0000
SVM bias	-0.9999	-0.9999	-1.0000	-1.0000
$\mathbf{w}^T\mathbf{w}$	1.9996	1.9996	2.0000	2.0000
L2-loss	1.9996e-8	1.9996e-8	0.0000	0.0000
Error vector	-9.9890e-2	-9.9890e-2	-1.0000e-1	-1.0000e-1
	9.9990e-5	9.9990e-5	0.0000e+0	0.0000e+0
	9.9990e-5	9.9990e-5	0.0000e+0	0.0000e+0
	-9.9890e-2	-9.9890e-2	-1.0000e-1	-1.0000e-1

6.1. A Simple Artificial Problem

Note that LSSVM is not maximal margin classification algorithm as no class-bounding hyperplane is clearly defined. The four maximal margin algorithms, i.e. RLSSVM, Primal, LIBSVM and SVMLight, were tested on a very simple binary classification problem to separate four 2-dimensional points using linear SVM. The four labeled points are given by $\mathbf{x}_1 = [1.1, 1]$, $y_1 = +1$; $\mathbf{x}_2 = [1, 1]$, $y_2 = +1$; $\mathbf{x}_3 = [0, 0]$, $y_3 = -1$; $\mathbf{x}_4 = [-0.1, 0]$, $y_4 = -1$. This set of four points is linearly separable with the solution obviously known. The linear solution SVM by SVMLIB is illustrated in Fig.1.

It should be noted that RLSSVM and Primal solve the unconstrained problem (6) with the L2-SVM loss function, while LIBSVM and SVMLight solve the dual of the constrained problem (1). Theoretically, algorithms solving the constrained problem (1) for separable data sets yield solutions independent of parameter C, because all the potential solutions that must satisfy the constraints have zero violations. However, algorithms solving the unconstrained problem (6) produce different solutions for different settings of C. The linear solutions of RLSSVM for different C are illustrated in Fig.2. It is evident in Fig.2 that small values for parameter C yield solutions with large margins. For sufficiently large C (approaching $+\infty$), the solution approaches the solution of Equation (1) [23][3].

The four maximal margin algorithms succeeded in identifying the two

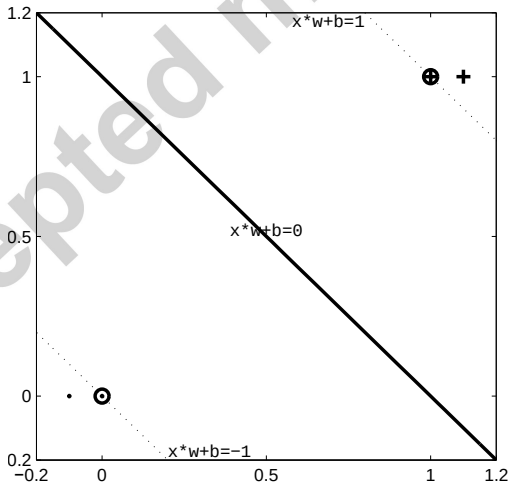


Figure 1: The solution of LIBSVM on a set of four 2-dimensional points. *Pluses* – points of class +1, *dots* – points of class -1, *dashed lines* – bounding planes, *solid lines* – the separating plane, and *circles* – support vectors.

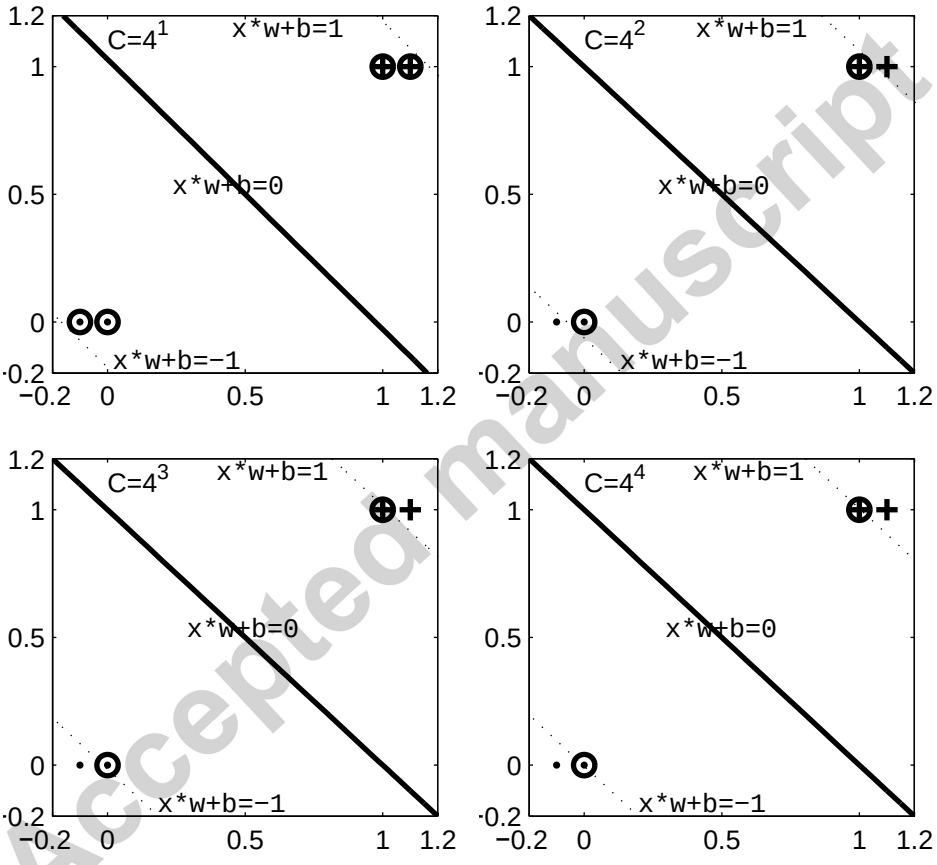


Figure 2: Solutions of RLSSVM for different penalty parameter values (C)

solution SVs. Table 1 compares the solutions produced by the four maximal margin algorithms for $C = 10,000$. In table 1, SVs indicates that the second and the third points are identified as the SVs by the four algorithms; the SVM coefficients are the coefficients present in the SVM expansion expression (i.e. $y_i\theta_i$'s for RLSSVM, nonzero β_i's for Primal and nonzero $y_i\alpha_i$'s for LIBSVM and SVMLight); $\mathbf{w}^T\mathbf{w}$ and L2-loss are the values of the two parts (the regularization term and the loss term without the penalty parameter C) of the objective function (6) for RLSSVM and Primal or (1) for LIBSVM and SVMLight; error vector lists all the errors (14) of an SVM for the four points. It can be checked that, for all the algorithms, the sum of the two SVM coefficients vanishes as shown in (51). For RLSSVM and Primal, θ_i's are C times the e_i's for the second and the third point, respectively, as shown in (53).

It can be seen in Table 1 that, both RLSSVM and Primal, and both LIBSVM and SVMLight approach the same solutions. Obviously, the L2-losses of the solutions of both LIBSVM and SVMLight are zero, meaning that the violations ξ_1, ξ_2, ξ_3 and ξ_4 of the four training points defined in (1) are all zero, while for both RLSSVM and Primal there are small L2-loss values meaning that those violations are not zero but very small. Parameter C is used to balance between those violations and the size of the margin (distance between the two bounding hyperplanes) as previously discussed and illustrated in Fig.2. Intuitively, more (at least not less) support vectors can be involved in SVMs produced by solving the unconstrained problem (6) than are involved in SVMs produced by solving the constrained problem (1).

6.2. Real World Problems

In the following, experimental results of five algorithms on five public benchmark problems are presented. Statistics of the the five benchmarking data sets are listed in Table 2. These datasets have been referred to numerous times in the literature, which makes them very suitable for testing our algorithms performance. Additional information about the datasets and their properties is given in our paper [51], no pre-processing was carried out on the data except to choose a diversity of applications from the examples available. For more information on the data sets one is referred to LIBSVM Data All the data sets employed in this work are available on-line. [4] The splice data set is from the Delve Datasets of The University of Toronto, Toronto, Ontario, Canada[5]. Both the adult and the web-page data sets are from the University of California at Irvine (UCI) machine learning repository [52].

[4]http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/
[5]http://www.cs.toronto.edu/∼delve/data/datasets.html

Table 2: Statistics of the five public benchmark problems

Data set	**Training set**	**Test set**	**Features**	**Source**
Splice	1,000	2,175	60	Delve
Adult	1,605	30,956	123	UCI [52]
Web-page	2,477	47,272	300	UCI [52]
Colon	30	32	2,000	[53]
Leukemia	38	34	7,129	[54]

The five algorithms are tested for linear kernel/basis and Gaussian radial kernel/basis, $k(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\gamma\|\mathbf{x}_1 - \mathbf{x}_2\|_2^2)$, on each training/test pair. Noting that LIBSVM and SVMLight use L1-loss while the other four use L2-loss. The penalty parameter C and the kernel parameter γ are tuned for each of the problems by a trial-and-error method: LIBSVM and Primal are tested on the training data set for 9 different C values for linear kernel, and 9×41 different (C, γ) pairs for Gaussian kernel. The 9 different C values are $\{10^{-4}, 10^{-3}, \cdots, 10^{4}\}$; and the 41 γ values are $\{10^{-5}, 10^{-4.8}, \cdots, 10^{3}\}$. The produced classifier models are tested on the test data set, the C and (C, γ) pair that yield best classification performance on the test data are chosen, see Table 3. In Table 3, the parameters for L1-loss are tuned using LIBSVM and are settings for both LIBSVM and LIBLight which employ L1-loss, while the parameters for L2-loss are tuned using Primal and are settings for Primal, RLSSVM and LSSVM which employ L2-loss.

To allow for comparison, ten training/test pairs are produced by randomly splitting each data set (for random training/test pairs of the same size as specified in Table 2), and the means and the standard deviations of the model size (#SVs), the running time (R.Time) which is included only for comparision and should not be considered in absolute terms, the training rate (T.Rate) and the testing rate (V.Rate) of the five algorithm over the ten random training/test set-pairs are compared in Table 4-8 respectively for the five data set. The training/testing rate here is defined as the percentage of correctly predicted labels for a algorithm on a training/testing set.

To compare the solutions produced by different algorithms, the penalty parameter C and kernel parameter γ are set to the same values for all the

Table 3: Tuned parameter settings for the benchmark problems

Data	**L1-loss**		**L2-loss**	
set	Linear	Gaussian	Linear	Gaussian
	(C)	(C, γ)	(C)	(C, γ)
Splice	10^{-2}	$10^{1}, 1.6e^{-2}$	10^{-3}	$10^{1}, 1.6e^{-2}$
Adult	10^{-1}	$10^{1}, 1.6e^{-2}$	10^{-1}	$10^{0}, 2.5e^{-2}$
Web-page	10^{1}	$10^{1}, 1.6e^{-2}$	10^{1}	$10^{1}, 2.5e^{-2}$
Colon	10^{-3}	$10^{2}, 1.6e^{-5}$	10^{-1}	$10^{2}, 2.5e^{-5}$
Leukemia	10^{-3}	$10^{2}, 1.0e^{-5}$	10^{-1}	$10^{3}, 1.0e^{-5}$

four algorithms, and tuned for each of the problems by a trial-and-error method. The maximal size of the support vector set is set at $n_{SVMax} = 1000$ for RLSSVM for all the experiments. Both linear and Gaussian kernels are tested for all the algorithms on each of the data sets.

In all of these experiments, the RLSSVM algorithm converges in no more that 10 iterations. Test results on the five benchmark problems show that all the four maximal margin algorithms provided solutions of competitive performance for both linear and Gaussian kernels with regard to the rates (both the means and the standard deviations) on test and training data sets. It is shown that RLSSVM (our proposed algorithm) delivered exactly the same results, in all the experiments, given that both Primal and RLSSVM employs the same L2-loss. Both LIBSVM and SVMLight using the same L1-loss delivered very similar solutions.

It can be observed that RLSSVM and Primal produced models of significantly more support vectors (less sparse) than both LIBSVM and SVMLight because both of them use L2-loss while LIBSVM and SVMLight use L1-loss. This is an obvious shortcoming of methods that employ L2-loss as discussed at the end of section 2. To overcome this problem, approximation techniques, for an example, that of [55] can be applied.

Table 4: Results on Splice data set

Method	**#SVs**	**R.Time (sec)**	**T.Rate(%)**	**V.Rate(%)**
Model of Linear kernel/basis				
RLSSVM	969.7, 7.1	0.341, 0.220	86.12, 0.79	83.33, 0.68
PRIMAL	969.7, 7.1	14.164, 0.489	86.12, 0.79	83.33, 0.68
LIBSVM	481.3, 12.0	0.930, 0.124	86.72, 0.61	83.74, 0.47
LIGHT	481.3, 11.7	0.966, 0.504	86.72, 0.61	83.74, 0.47
LSSVM	214.8, 34.1	0.086, 0.017	51.95, 1.17	51.89, 0.54
Model of Gaussian kernel/basis				
RLSSVM	743.2, 16.9	2.772, 0.128	99.98, 0.04	89.43, 0.49
PRIMAL	743.2, 16.9	12.163, 1.271	99.98, 0.04	89.43, 0.49
LIBSVM	695.4, 16.6	1.925, 0.123	99.98, 0.04	89.44, 0.45
LIGHT	699.0, 18.3	3.384, 0.170	99.98, 0.04	89.44, 0.45
LSSVM	905.0, 0.0	3.602, 0.141	51.95, 1.17	51.89, 0.54

Table 5: Results on Adult data set

Method	**#SVs**	**R.Time (sec)**	**T.Rate(%)**	**V.Rate(%)**
Model of linear kernel/basis				
RLSSVM	997.4, 29.3	1.264, 0.064	85.76, 0.63	84.22, 0.12
PRIMAL	997.4, 29.3	24.725, 1.324	85.76, 0.63	84.22, 0.12
LIBSVM	629.0, 19.6	0.937, 0.056	85.30, 0.71	83.89, 0.35
LIGHT	629.3, 19.3	1.565, 0.659	85.30, 0.71	83.89, 0.35
LSSVM	69.3, 19.2	0.084, 0.020	24.31, 0.94	24.07, 0.05
Model of Gaussian kernel/basis				
RLSSVM	1079.3, 25.0	12.055, 0.738	86.41, 0.55	84.20, 0.15
PRIMAL	1079.3, 25.0	32.705, 4.004	86.41, 0.55	84.20, 0.15
LIBSVM	623.4, 21.6	1.334, 0.108	87.36, 0.57	83.96, 0.17
LIGHT	623.4, 21.6	6.595, 0.267	87.36, 0.57	83.96, 0.17
LSSVM	136.2, 66.7	0.201, 0.115	24.31, 0.94	24.07, 0.05

The LSSVM performed much more unsatisfactorily than the other methods in all the test cases. This is probably due to some poorly chosen algorithm parameters, such as the step (by default 5%) and the tradeoff percent (by default 75%). Publicly available code usually requires additional tunings. An obvious advantage of the proposed RLSSVM over LSSVM is that there is no additional parameters (in addition to C and then kernel parameters) that

need to be tuned, given that the maximal number of the support vectors, n_{SVMax}, for RLSSVM will not influence the solution when it is large enough (not less than the number of support vectors of the solution), given again that the global optimal solution is unique. In this experiment, it is simply set at $n_{SVMax} = 1200$ for all the cases.

Tables 9, 10 and 11 present relative running times for the five meth-

Table 6: Results on Web-page data set

Method	**#SVs**	**R.Time (sec)**	**T.Rate(%)**	**V.Rate(%)**
Model of linear kernel/basis				
RLSSVM	524.4, 30.1	1.703, 0.034	99.34, 0.15	97.98, 0.12
PRIMAL	524.3, 30.0	3.753, 0.252	99.34, 0.15	97.98, 0.12
LIBSVM	172.6, 12.7	0.434, 0.170	99.18, 0.14	98.00, 0.13
LIGHT	173.7, 11.2	1.191, 0.361	99.18, 0.14	98.00, 0.13
LSSVM	745.0, 229.6	2.051, 1.117	11.62, 1.17	11.50, 0.81
Model of Gaussian kernel/basis				
RLSSVM	677.9, 38.3	13.288, 0.305	99.31, 0.13	98.03, 0.07
PRIMAL	677.9, 38.3	6.267, 0.470	99.31, 0.13	98.03, 0.07
LIBSVM	270.4, 30.1	0.408, 0.053	98.89, 0.27	97.95, 0.10
LIGHT	273.9, 33.6	2.433, 0.476	98.89, 0.27	97.95, 0.10
LSSVM	805.8, 312.7	2.098, 1.266	3.08, 0.28	2.97, 0.01

Table 7: Results on Colon data set

Method	**#SVs**	**R.Time (sec)**	**T.Rate(%)**	**V.Rate(%)**
Model of linear kernel/basis				
RLSSVM	22.5,2.8	0.061, 0.008	100.00, 0.00	82.81, 6.29
PRIMAL	22.5,2.8	0.064, 0.047	100.00, 0.00	82.81, 6.29
LIBSVM	22.7,2.9	0.091, 0.009	98.67, 1.63	85.94, 4.89
LIGHT	22.7,2.9	0.123, 0.024	98.67, 1.63	85.94, 4.89
LSSVM	24.6,1.8	0.039, 0.014	49.33, 6.11	50.31,14.69
Model of Gaussian kernel/basis				
RLSSVM	24.1, 2.7	0.055, 0.008	100.00, 0.00	83.13, 6.88
PRIMAL	24.1, 2.7	0.061, 0.009	100.00, 0.00	83.13, 6.88
LIBSVM	22.8, 2.9	0.089, 0.010	100.00, 0.00	83.13, 6.43
LIGHT	22.8, 2.9	0.150, 0.020	100.00, 0.00	83.13, 6.43
LSSVM	24.8, 1.3	0.039, 0.008	34.00, 8.14	36.88, 7.63

ods considered RLS(SVM), PRI(MAL), LIB(SVM) LIG(HT) LSS(VM). It should be noted that the proposed algorithm is essentially a batch algorithm. It is not suitable for large data sets (with large number of high-dimensional points) from the point of view of both execution time and memory requirements. This is why RLSSVM is much slower than Primal, LIBSVM, SVM-LIGHT and LSSVM on the web-page data set for Gaussian kernel, given

Table 8: Results on Leukemia data set

Method	**#SVs**	**R.Time (sec)**	**T.Rate(%)**	**V.Rate(%)**
Model of linear kernel/basis				
RLSSVM	30.6, 1.8	0.019, 0.006	100.00, 0.00	94.41, 5.95
PRIMAL	30.6, 1.8	0.020, 0.007	100.00, 0.00	94.41, 5.95
LIBSVM	30.6, 1.8	0.091, 0.006	100.00, 0.00	94.41, 5.95
LIGHT	30.6, 1.8	0.106, 0.014	100.00, 0.00	94.41, 5.95
LSSVM	31.6, 1.5	0.023, 0.008	56.32, 9.86	50.88, 7.33
Model of Gaussian kernel/basis				
RLSSVM	32.3, 1.8	0.017, 0.005	100.00, 0.00	93.82, 6.76
PRIMAL	32.3, 1.8	0.028, 0.006	100.00, 0.00	93.82, 6.76
LIBSVM	32.1, 2.0	0.092, 0.004	100.00, 0.00	93.53, 6.81
LIGHT	32.1, 2.0	0.108, 0.013	100.00, 0.00	93.53, 6.81
LSSVM	31.0, 1.8	0.027, 0.007	66.58, 4.09	63.82, 4.57

again that the running time of RLSSVM and memory requirement are of orders $O(n_{SV}^2(n_{SV}+N+n)$ and $O(n_{SV}^2)$, respectively, as discussed in section 5.3.

Hence, for problems of large data set sizes, a sequential implementation of RLSSVM is desirable. Since it is merely derived in the RLS context, it is straightforward to apply recursive least-squares in order to sequentialize the

Table 9: Relative running times for different C's on splice for linear kernel

C	RLS.	**PRI.**	**LIB.**	**LIG.**	**LSS.**
10^{-4}	0.062	0.843	0.437	0.109	0.047
10^{-3}	0.063	1.484	0.390	0.094	0.032
10^{-2}	0.062	1.422	0.328	0.172	0.031
10^{-1}	0.062	1.312	0.672	0.438	0.032
10^{+0}	0.063	0.890	8.343	7.297	0.046
10^{+1}	0.062	1.110	44.859	23.844	0.032
10^{+2}	0.062	1.109	154.500	102.094	0.047
10^{+3}	0.078	1.109	544.407	30.640	0.063
10^{+4}	0.062	1.109	1419.250	30.797	0.062

algorithm by using (20) and (21). It is also observed that the running time of LIBSVM is very sensitive to C, particularly for linear kernels. SVMLight is also sensitive to its settings but slightly less so than LIBSVM. The other four algorithms are much more stable with RLSSVM outperforming all the other algorithms in stability.

7. Conclusions

The work presented in this paper shows how the linear SVM can be formulated as a regularized least squares (RLS) problem. We have also shown how this naturally extends to generalized SVMs with nonlinear kernels. The

Table 10: Relative running times times for different C's on adult for linear kernel

C	RLS.	**PRI.**	**LIB.**	**LIG.**	**LSS.**
10^{-4}	0.282	4.860	0.344	0.422	0.000
10^{-3}	0.297	4.860	0.343	0.188	0.000
10^{-2}	0.313	8.937	0.359	0.204	0.140
10^{-1}	0.328	5.125	0.375	0.297	0.031
10^{+0}	0.328	5.938	0.812	1.937	0.078
10^{+1}	0.360	3.797	5.563	13.984	0.266
10^{+2}	0.422	4.797	44.672	54.453	0.250
10^{+3}	0.406	4.344	179.000	202.468	0.313
10^{+4}	0.390	4.359	615.125	13.094	0.406

novelty of this generalization is that it is done merely in the context of the primal RLS, neither the Lagrange multipliers, Karush-Kuhn-Tucker (KKT) conditions and duality theory, nor the application in advance of the *representer theorem* of Kimeldorf and Wahba is involved. A fast iterative algorithm for solving the SVM based Cholesky decomposition is proposed to allow for the formulation to be tested on some typical problems. The behavior of the solution (it's correctness and form) has been analyzed and is compared with the results of applying standard SVM solvers to the same problems.

With the least squares solution to the normal of the SVM separating hyperplane, the solution is expressed as an equation with regard to the error

Table 11: Relative running times times for different C's on web-page for linear kernel

C	RLS.	**PRI.**	**LIB.**	**LIG.**	**LSS.**
10^{-4}	1.172	13.703	0.234	9.750	0.125
10^{-3}	1.922	42.750	0.234	2.359	0.125
10^{-2}	1.656	18.609	0.250	1.125	0.141
10^{-1}	1.610	4.250	0.250	0.797	8.687
10^{+0}	1.703	3.578	0.328	1.328	17.453
10^{+1}	1.875	3.516	1.516	3.016	30.391
10^{+2}	2.703	4.610	2.484	5.079	33.328
10^{+3}	2.985	5.110	3.454	6.141	34.297
10^{+4}	3.109	6.640	8.250	26.750	32.797

vector and a set of indicator variables depending on the errors, referred to as the error equation. As the optimal solution is unique, solving the primal RLS problem is equivalent to solving the error equation. Corresponding to the bounded points and the support vectors which are unbounded, the error vector is partitioned into two parts. By applying some matrix operations on this error vector, the optimal solution is expressed as a linear combination of inner products of the support vectors with an input point.

Existing primal methods that invoke the representer theorem require the full kernel matrix (over the full training set) to be positive definite for a well-conditioned Hessian and the globally optimal solution to be unique. Of

course, many practical data sets may have repeated points, making the full kernel matrix only positive semi-definite. The solution proposed in this paper is simply derived in the primal RLS context without invoking the representer theorem, and only requiring the full kernel matrix to be positive semi-definite – this is guaranteed for any training data by kernel functions satisfying Mercer's condition. Also , the solution here is derived in the context of the RLS without replacing the inequality constraints with equalities, this overcomes the lack-of-sparseness disadvantage, those points without violations are not presented in the solution.

The fast (Cholesky decomposition based) successive substitution iterative algorithm is proposed directly from the error equation based on and using permutation of the support vectors. An experiment is presented on a simple artificial data set to demonstrate the properties of the solution.

The algorithm is then applied to three benchmark binary classification problems and compared with some well established approaches that solve the (unconstrained) primal RLS problem and two popular software packages, LIBSVM and SVMLight (which solve the dual of the standard constrained quadratic programming problem.)

The only caveat evident from the experimental tests is that SVMs formulated in unconstrained RLS problems with the L2-loss function, such as our proposed algorithm and other Primal approaches, involve more support vectors than standard SVMs.

This study of a variety of problem types is not intended to deliver a detailed statistically significant evaluation. Our broad spectrum of examples demonstrates the wide range of problems for which we can demonstrate good

convergence and we leave it to further research to perform larger trials upon which it would be meaningful to perform statistical analysis.

However, the test results demonstrate the accuracy, stability and computational simplicity of the method which should make it a very attractive new method for solving problems that are amenable to attack using a SVM classifier.

Acknowledgment

This work was part-funded by the European Commission under the Seventh Framework Programme: large-scale integrating project *HaptiMap*, FP7-ICT-224675.

[1] V. Vapnik, A. Lerner, Pattern recognition using generalized portrait method, Automation and Remote Control 24 (1963) 774–780.

[2] V. N. Vapnik, Estimation of Dependences Based on Empirical Data, Addendum 1, New York: Springer-Verlag, 1982.

[3] C. Cortes, V. Vapnik, Support vector networks, Machine Learning 20 (1995) 273 – 297.

[4] J. Zhou, L.Zhu, Principal minimax support vector machine for sufficient dimension reduction with contaminated data, Computational Statistics & Data Analysis 94 (2016) 33–48.

[5] S. R. Gunn, M. Brown, K. M. Bossley, Network performance assessment for neuro-fuzzy data modelling, in: X. Liu, P. Cohen, M. Berthold

(Eds.), Intelligent Data Analysis, vol. 1208 of Lecture Notes in Computer Science, 1997, pp. 313–323.

[6] A. Savio, M. Grana, Local activity features for computer aided diagnosis of schizophrenia on resting-state fmri, Neurocomputing 164 (2015) 154–161.

[7] D. Chyzhyk, M. Grana, Classification of schizophrenia patients on lattice computing resting-state fmri features, Neurocomputing 151 (1) (2015) 151–160.

[8] M.Z.Parvez, M.Paul, Epileptic seizure detection by analyzing eeg signals using different transformation techniques, Neurocomputing 145 (2014) 190–200.

[9] X.Chen, J.Yang, Q.Mao, F.Han, Regularized least squares fisher linear discriminant with applications to image recognition, Neurocomputing 122 (2013) 521–534.

[10] C. J. C. Burges, A tutorial on support vector machines for pattern recognition, Data Mining and Knowledge Discovery 2 (2) (1998) 121–167.

[11] V. N. Vapnik, Statistical Learning Theory, John Wiley & Sons, 1998.

[12] B. Schölkopf, A. J. Smola, Learning with kernels, MIT Press, Cambridge, MA, 2002.

[13] M. Ferris, T. Munson, Interior-point methods for massive support vector machines, SIAM Journal on Optimization 13 (2002) 783–804.

[14] J. C. Platt, Fast training of support vector machines using sequential minimal optimization, in: B. Schölkopf, C. J. C. Burges, A. J. Smola (Eds.), Advances in Kernel Methods - Support Vector Learning, MIT Press, Cambridge, MA, 1998.

[15] K. Huanga, D. Zhengb, J. Sunb, Y. Hottac, K. Fujimotoc, S. Naoic, Sparse learning for support vector classification, Pattern Recognition Letters 31 (2010) 1944–1951.

[16] Q. Li, X. Li, W. Ba, Sparse least squares support vector machine with l0-norm in primal space, 2015, pp. 2778–2783.

[17] B. G. Fung, O. Mangasarian, Proximal support vector machine classifiers, Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2001) 77–86.

[18] J. A. K. Suykens, J. Vandewalle, Least squares support vector machine classifiers, Neural Processing Letters 9 (3) (1999) 293–300.

[19] B. P. Komarek, Logistic regression for data mining and high-dimensional classification (Ph.d. thesis), Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, 2004.

[20] B. Zhang, R. Jin, Y. Yang, A. Hauptmann, Modified logistic regression: An approximation to svmand its applications in large-scale text categorization, Twentieth International Conference on Machine Learning (2003) 472–479.

[21] J. A. K. Suykens, L. Lukas, J. Vandewalle, Sparse least squares support vector machine classifiers, ESANN'2000 European Symposium on Artificial Neural Networks (2000) 37–42.

[22] O. L. Mangasarian, A finite newton method for classification, Optimization Methods and Software 17 (2002) 913–929.

[23] S. S. Keerthi, D. M. DeCoste, A modified finite newton method for fast solution of large scale linear svms, Journal of Machine Learning Research 6 (2005) 341–361.

[24] S. Sonnenburg, V. Franc, Coffin : A computational framework for linear svms, Proceedings of the 27th International Conference on Machine Learning (2010) 999–1006.

[25] R. Fan, K. W. Chang, C. J. Hsieh, X. R. Wang, C. J. Lin, Liblinear: A library for large linear classification, Journal of Machine Learning Research 9 (2008) 1871–1874.

[26] A. J. Smola, B. Schölkopf, From regularization operators to support vector kernels, Advances in Neural information processings systems 10 (1998) 343–349.

[27] M. A. Aizerman, E. M. Braverman, L. I. Rozonoer, Theoretical foundations of the potential function method in pattern recognition learning, Automation and Remote Control 25 (1964) 821–837.

[28] O. Chapelle, Training a support vector machine in the primal, Neural Computation 19 (5) (2007) 1155–1178.

[29] G. S. Kimeldorf, G. Wahba, A correspondence between bayesian estimation on stochastic processes and smoothing by splines, Annals of Mathematical Statistics 41 (1970) 495–502.

[30] F. Perez-Cruz, A. Navia-Vazquez, P. L. Alarcon-Diana, A. A. Rodrguez, A new training algorithm for support vector machines, in: Proceedings of the Fifth Bayona Workshop on Emerging Technologies in Telecommunications, Baiona, Spain, 1999, pp. 116–120.

[31] F. Perez-Cruz, A. Navia-Vazquez, P. L. Alarcon-Diana, A. A. Rodrguez, Svc-based equalizer for burst tdma transmissions, Signal Processing 81 (2001) 1681–1693.

[32] F. Perez-Cruz, C. Bousono-Calzon, A. Artes-Rodriguez, Convergence of the irwls procedure to the support vector machine solution, Neural Computing 17 (2005) 7–18.

[33] K. D. Brabanter, J. D. Brabanter, J. A. K. Suykens, B. D. Moor, Optimized fixed-size kernel models for large data sets, Computational Statistics and Data Analysis 54 (2010) 1481–1504.

[34] R. Mall, J. A. K. Suykens, Very sparse lssvm reductions for large scale data, IEEE Transactions on Neural Networks and Learning Systems 26 (2015) 1086–1097.

[35] J. M. Leski, Iteratively reweighted least squares classifier and its l1-and l1-regularized kernel versions, Bulletin of the Polish Academy of Sciences 58 (2010) 171–182.

[36] S. Dinga, X. Hua, J. Yu, An overview on nonparallel hyperplane support vector machine algorithms, Neural Computing and Applications 25 (2014) 975–982.

[37] S. Ding, X. Hua, Recursive least squares projection twin support vector machines for nonlinear classification, Neurocomputing 130 (2015) 3–9.

[38] X. Hua, S. Ding, Weighted least squares projection twin support vector machines with local information, Neurocomputing 160 (2015) 228–237.

[39] C.Chen, C.Yan, Y. Li, A robust weighted least squares support vector regression based on least trimmed squares, Neurocomputing 168 (2015) 941–946.

[40] S. Zhou, Sparse lssvm in primal using cholesky factorization for large-scale problems, IEEE Transactions on Neural Networks and Learning Systems 10.1109/TNNLS.2015.2424684.

[41] Z. Liu, S. Lin, M. T. Tan, Sparse support vector machines with l_p penalty for biomarker identification, IEEE/ACM Transactions on Computational Biology And Bioinformatics 7 (1) (2010) 100–107.

[42] K.-W. C. Cho-Jui Hsieh, S. S. K. Chih-Jen Lin, S. Sun-dararajan, A dual coordinate descent method for large-scale linear svm, Proceedings of the Twenty Fifth International Conference on Machine Learning (ICML).

[43] C.-J. Lin, R. C. Weng, S. S. Keerthi, Trust region newton method for large-scale logistic regression, Journal of Machine Learning Research 9 (2008) 627–650.

[44] S. R. Searle, Matrix Algebra Useful for Statistics, John Wiley and Sons, 1982.

[45] G. H. Golub, C. F. van Loan, Matrix Computations, The Johns Hopkins University Press, 3rd edition, Baltimore, 1996.

[46] T. van Gestel, J. A. K. Suykens, B. Baesens, G. D. S. Viaene, J. Vanthienen, B. D. Moor, J. Vandewalle, Benchmarking least squares support vector machine classifiers, Machine Learning (2004) 5–32.

[47] D. S. Watkins, Fundamentals of Matrix Computations, 2nd Edition, Wiley-Interscience, 2002.

[48] C.-C. Chang, C.-J. Lin, Libsvm: a library for support vector machines.

[49] T. Joachims, Making large-scale svm learning practical, in: B. Schölkopf, C. Burges, A. Smola (Eds.), Advances in Kernel Methods - Support Vector Learning, MIT Press, Cambridge, MA, 1999.

[50] R.-E. Fan, P.-H. Chen, C.-J. Lin, Working set selection using second order information for training svm, Journal of Machine Learning Research 6 (2005) 1889–1918.

[51] J. X. Peng, S. Ferguson, K. Rafferty, V. Stewart, A sequential algorithm for sparse support vector classifiers, Pattern Recognition 46 (2013) 1195–1280.

[52] C. L. Blake, C. J. Merz, UCI Repository of Machine Learning Databases, Univ. California, Dept. Inform. Comput. Sci., Irvine, CA, 1998.

[53] U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, A. J. Levine, Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays, Cell Biology 96 (1999) 6745–6750.

[54] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, E. S. Lander, Molecular classification of cancer: class discovery and class prediction by gene expression monitoring, Science 286 (5439) (1999) 531–537.

[55] S. S. Keerthi, O. Chapelle, D. DeCoste, Building support vector machines with reduced classifier complexity, Journal of Machine Learning Research 7 (2006) 1493–1515.