Training Sparse Least Squares Support Vector Machines by the QR Decomposition


Published in:
Neural Networks

Document Version:
Peer reviewed version

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

Publisher rights
Copyright 2017 Elsevier. This manuscript is distributed under a Creative Commons Attribution-NonCommercial-NoDerivs License (https://creativecommons.org/licenses/by-nc-nd/4.0/), which permits distribution and reproduction for non-commercial purposes, provided the author and source are cited.

General rights
Copyright for the publications made accessible via the Queen's University Belfast Research Portal is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The Research Portal is Queen's institutional repository that provides access to Queen's research output. Every effort has been made to ensure that content in the Research Portal does not infringe any person's rights, or applicable UK laws. If you discover content in the Research Portal that you believe breaches copyright or violates any law, please contact openaccess@qub.ac.uk.

Open Access
This research has been made openly available by Queen's academics and its Open Research team. We would love to hear how access to this research benefits you. – Share your feedback with us: http://go.qub.ac.uk/oa-feedback
Accepted Manuscript

Training sparse least squares support vector machines by the QR decomposition

Xiao-Lei Xia

PII: S0893-6080(18)30207-7
DOI: https://doi.org/10.1016/j.neunet.2018.07.008
Reference: NN 3991

To appear in: Neural Networks

Received date: 2 November 2017
Revised date: 7 July 2018
Accepted date: 11 July 2018

Please cite this article as: Xia, X.-L., Training sparse least squares support vector machines by the QR decomposition. Neural Networks (2018), https://doi.org/10.1016/j.neunet.2018.07.008

This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.
Training Sparse Least Squares Support Vector Machines by the QR Decomposition

Xiao-Lei Xiaa,1,*

*Centre for Cancer Research and Cell Biology, School of Medicine, Dentistry and Biomedical Sciences, Queen’s University Belfast, BT9 7BL, UK

Abstract

The solution of an LS-SVM has suffered from the problem of non-sparseness. The paper proposed to apply the KMP algorithm, with the number of support vectors as the regularization parameter, to tackle the non-sparseness problem of LS-SVMs. The idea of the kernel matching pursuit (KMP) algorithm was first revisited from the perspective of the QR decomposition of the kernel matrix on the training set. Strategies are further developed to select those support vectors which minimize the leave-one-out cross validation error of the resultant sparse LS-SVM model. It is demonstrated that the LOOCV of the sparse LS-SVM can be computed accurately and efficiently. Experimental results on benchmark datasets showed that, compared to the SVM and variants sparse LS-SVM models, the proposed sparse LS-SVM models developed upon KMP algorithms maintained comparable performance in terms of both accuracy and sparsity.

Keywords: least-squares support vector machines, kernel matching pursuit, QR decomposition, sparseness

1. Introduction

Support Vector Machines (SVM) [22] are a family of algorithms for classification and regression and have enjoyed widespread applications to various pattern recognition tasks since its introduction two decades ago. The standard SVM follows the structural risk minimization principle and is formulated as a quadratic programming (QP) problem subject to inequality constraints. As opposed to the SVM, the least-squares SVM (LS-SVM) which is an important variant of the SVM, adopts equality constraints [20]. Gestel et al. gave a Bayesian perspective on the formulation of the LS-SVM and also investigated its connection with Gaussian Process and kernel fisher discriminant analysis [8]. On the other
hand, the LS-SVM can be viewed as a regularization method in a reproducing kernel Hilbert space (RKHS) [11].

The training of the LS-SVM can be reduced to a system of linear equations which can be addressed by a variety of methods. Suykens et al. transformed the training of an LS-SVM into two linear systems with an identical positive definite coefficient matrix, followed by the application of the conjugate gradient (CG) method [19]. Chu et al. proposed an alternative linear system to which the CG method can be directly applied but with less computational cost [5]. Starting from the dual form of the LS-SVM, Keerthi and Shevade proposed the sequential minimal optimization (SMO) algorithm which optimized a pair of Lagrangian multipliers at each iteration [12]. Bo et al. provided an alternative strategy for the selection of the Lagrangian multiplier pair [1].

An issue with these training algorithms is that the resultant LS-SVM decision function is parameterized by a large number of training samples, or support vectors. This problem which has been referred to non-sparseness of the LS-SVM has received considerable research attention. A straightforward strategy to approach the problem was the removal of support vectors whose Lagrangian multipliers were of small absolute values [18]. Zeng and Chen proposed to remove those support vectors whose absence caused the least perturbation to the dual form [25]. de Kruif and de Vries factored in the change of every Lagrangian multiplier arising from pruning a training sample and recommended the deletion of the one introducing the minimal approximation error [6, 14]. An alternative strategy to build a sparse LS-SVM select, iteratively, into the solution samples which established a good approximation of the other samples in the feature space [2]. Jiao et al. developed a sparse LS-SVM model in which only one training sample was selected into the decision function at an iteration until the target output for each sample has been well approximated [11]. The Kernel matching pursuit (KMP) algorithm constructed the solution in a greed manner and at each iteration, selected the sample which led to the maximal drop in the sum of squared error loss [23, 16]. Because the KMP algorithm facilitates a direct control of the sparsity of the solution, it has been applied to ease the non-sparseness of the LS-SVM [24]. KMP was closely related to the orthogonal least squares (OLS) method in the field of nonlinear model identification [4]. Both methods realized QR decomposition [21] of the coefficient matrix associated with training an LS-SVM, factoring it into an orthogonal matrix and a triangular matrix. Sparse LS-SVM models proposed by Zhou employed the low rank representation for the kernel matrix by pivoted Cholesky decomposition [26]. On the other hand fixed-size LS-SVM applies Nyström method in order to find low rank estimation of the kernel matrix and techniques have been introduced to minimize the the L0 norm of the solution [15].

This paper first revisited the idea of the KMP algorithm from the perspective of the QR decomposition of the kernel matrix. Strategies were then proposed of building sparse LS-SVM models based on the KMP algorithm: 1) the number of support vectors was introduced as the regularization parameter tuning off the trade-off between the empirical risk and the generalization ability and the KMP algorithm was then applied; 2) in contrast to the classical KMP algorithm...
selecting support vectors so that the sum of squared error for the associated linear system is minimized, an alternative scheme was proposed which selected support vectors that minimize the leave-one-out cross validation (LOOCV) error of the resultant LS-SVM model. It was demonstrated the the LOOCV error rate can be accurately computed for the proposed KMP algorithm. Furthermore, techniques were discussed which makes the computation of the LOOCV error rate efficient.

The rest of the paper is organized as follows. Section 2 briefly reviews the formulation of the LS-SVM and demonstrates that training an LS-SVM can be reduced to a linear system whose coefficient matrix is positive-definite. The classical KMP algorithm is revisited in the context of the QR decomposition of the kernel matrix in Section 3. Section 4 proposed variants of the KMP algorithms and demonstrates that the LOOCV error rates of the proposed model can be calculated accurately and efficiently. Section 6 compares performances of the proposed sparse LS-SVM models based on the KMP algorithm, the standard SVM and a number of sparse LS-SVM models on public datasets for machine learning datasets. The paper concludes in Section 7.

2. The Least Squares SVM and its training

For the classification of samples \((x_1, y_1), \ldots, (x_\ell, y_\ell)\), where \(x \in \mathbb{R}^d\) and \(y \in \{-1, 1\}\), a Least-Squares SVM seeks the optimal separating hyperplane parameterized by \((w, b)\) which are the solution to the following optimization problem:

\[
\min_{w, b, \xi} \frac{1}{2} w^T w + \frac{1}{2} C \sum_{i=1}^\ell \xi_i^2 \tag{1}
\]

s.t. \(w^T \Phi(x_i) + b = y_i - \xi_i\)

where \(i = 1, \ldots, \ell\) and \(C\) is the regularization parameter indicating the tradeoff between the training cost and the generalization ability. \(\Phi\) is the function which maps training data into the feature space.

Introducing the Lagrange multipliers \(\alpha_i\) \((i = 1, \ldots, \ell)\) for each of the equality constraints gives rise to the following Lagrangian:

\[
\mathcal{L}(w, b, \xi, \alpha) = \frac{1}{2} w^T w + \frac{1}{2} C \sum_{i=1}^\ell \xi_i^2 - \sum_{i=1}^\ell \alpha_i [w^T \Phi(x_i) + b - y_i + \xi_i] \tag{2}
\]

Due to the equality constraints, \(\alpha_i\) can either be positive or negative according to the Karush-Kuhn-Tucker (KKT) conditions [7]. The optimality of (2)
requires:
\[ \frac{\partial L}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{\ell} \alpha_i \Phi(x_i) \] (3)
\[ \frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{i=1}^{\ell} \alpha_i = 0 \] (4)
\[ \frac{\partial L}{\partial \xi_i} = 0 \Rightarrow \alpha_i = C\xi_i, \quad i = 1, \ldots, \ell \] (5)
\[ \frac{\partial L}{\partial \alpha_i} = 0 \Rightarrow w^T \Phi(x_i) + b = y_i - \xi_i, \quad i = 1, \ldots, \ell \] (6)

The linear equations can be represented as:

\[ \begin{bmatrix} K + C^{-1}I & \vec{1} \\ \vec{1}^T & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ b \end{bmatrix} = \begin{bmatrix} y \\ 0 \end{bmatrix} \] (7)

where \( y = [y_1, \ldots, y_\ell]^T \), \( \alpha = [\alpha_1, \ldots, \alpha_\ell]^T \) and \( \vec{1} \) is a \( \ell \)-dimensional vector of ones and \( I \) is an identity matrix of appropriate rank. The matrix \( K \) is a square matrix of \( \ell \) rows and \( K_{ij} = \Phi(x_i)^T \Phi(x_j) = K(x_i, x_j) \). Equation (7) can actually be transformed into a single linear system of order \( \ell - 1 \), which has been discussed in Appendix.

3. Kernel Matching Pursuit (KMP) for Sparsity Improvement

An alternative solution to sparsity improvement of LS-SVM is the application of the Kernel Matching Pursuit (KMP) algorithm. Since the KMP algorithm have been described in detail in [23, 24], the paper revisited the idea of the KMP from the perspective of QR decomposition.

3.1. QR decomposition for Solving Linear Systems

Denoting the coefficient matrix of Equation (7) as \( H \). The QR decomposition of the matrix \( H \), implemented by the Gram–Schmidt process, factorizes \( H \) into an orthogonal matrix, denoted as \( P = \begin{bmatrix} p_1, \ldots, p_\ell \end{bmatrix} \) and an upper triangle, denoted as \( A \). The coefficient matrix is updated iteratively and the \( i \)-iteration identifies the \( i \)-th orthonormal basis \( p_i \) for the matrix \( P \).

The general workflow of the QR decomposition of \( H \) is described as follows. Initially, set \( H_0 = H \). Representing the coefficient matrix at the \( i \)-iteration as \( H_{i-1} \) and \( p_i \) is selected as the \( i \)-th column of \( H_{i-1} \). The coefficient matrix for the \( (i + 1) \)-iteration, denoted as \( H_i \), is obtained:

\[ H_i = \left( I - \frac{p_i p_i^T}{p_i^T p_i} \right) H_{i-1} \] (8)
\( \mathbf{p}_{i+1} \) is identified as the \((i + 1)\)-th column of the current coefficient matrix of \( \mathbf{H}_i \). For a matrix whose column rank is \( \ell + 1 \) as the coefficient matrix \( \mathbf{H} \) of Equation (7), an orthogonal matrix of \( \ell + 1 \) columns, represented by \( \mathbf{P} \) can be generated after \( \ell \) iterations.

Meanwhile, the upper triangle \( \mathbf{A} \) is obtained as

\[
\mathbf{A} = \mathbf{P}^T \mathbf{H}
\]  

(9)

Equation (7) is thus transformed into:

\[
\mathbf{P} \mathbf{A} \mathbf{\beta} = \mathbf{r}
\]  

(10)

where \( \mathbf{r} = \begin{bmatrix} y \\ 0 \end{bmatrix} \) and \( \mathbf{\beta} = \begin{bmatrix} \alpha \\ b \end{bmatrix} \). Multiplying both sides of Equation (10) by \( \mathbf{P}^T \) yields:

\[
\mathbf{A} \mathbf{\beta} = \mathbf{P}^T \mathbf{r}
\]  

(11)

The solution can then be obtained by the back substitution algorithm.

3.2. Sparsity Control of the Solution

In order to enhance the sparsity of the decision function, the KMP algorithm selects, iteratively, only a subset of the \((\ell + 1)\) columns of \( \mathbf{H} \).

At the \((n+1)\)-th iteration, \((n)\) basis functions have been selected, producing a matrix of \( \mathbf{H}_n \) which is factorized into an orthogonal matrix of \( \mathbf{P}_n \) and an upper triangle \( \mathbf{A}_n \). It holds that \( \mathbf{H}_n = \mathbf{P}_n \mathbf{A}_n \).

Denoting the solution to the nonlinear system of \( \mathbf{H}_n \mathbf{\beta} = \mathbf{r} \) as \( \mathbf{\beta}_n \) and it holds that:

\[
\mathbf{\beta}_n = (\mathbf{H}_n^T \mathbf{H}_n)^{-1} \mathbf{H}_n^T \mathbf{r} = \mathbf{A}_n^{-1} \mathbf{P}^T_n \mathbf{r}
\]  

(12)

The associated sum of squared error, denoted as \( e_n \), can be computed

\[
e_n = \mathbf{\beta}_n^T \mathbf{H}_n^T \mathbf{H}_n \mathbf{\beta}_n
= \mathbf{r}^T \mathbf{H}_n (\mathbf{H}_n^T \mathbf{H}_n)^{-1} \mathbf{H}_n^T \mathbf{r}
= \mathbf{r}^T \mathbf{P}_n (\mathbf{P}_n^T \mathbf{P}_n)^{-1} \mathbf{P}_n^T \mathbf{r}
= ||\mathbf{P}_n^T \mathbf{r}||^2
\]  

(13)

The KMP algorithm seeks the minimization of the sum of squared error. Thus, the subsequent \((n + 1)\)-th iteration selects the basis function whose orthonormal basis \( \mathbf{p}_{n+1} \) results in the smallest \( (\mathbf{p}_{n+1}^T \mathbf{r})^2 \).

4. Building Sparse LS-SVMs by the Application of KMP

An usual train of thought is the application of the KMP algorithm to Equation (7) directly. But it is worth attention that by forcing subsets of \( \alpha_i \) in Equation (1) to be zeros, the optimality condition of \( \alpha_i = C \xi_i \) is violated.
Without introducing the variables \( \xi_i \) and the representer theorem of \( w = \sum_{i=1}^{n} \alpha_i \Phi(x_i) \) where \( n \) is the non-zero \( \alpha \)s, the linear system corresponding to a sparse LS-SVM is

\[
K_n \alpha + b = y
\]  

(14)

where \( K_n \) is a matrix of \( n \) columns from the full kernel matrix on the training set.

The empirical risk of the learning model is \((K_n \alpha + b - y)^T (K_n \alpha + b - y)\), which, from Equation (12), is a function of the selected columns. Furthermore, the empirical risk monotonically decreases to the growth in the number of columns. Meanwhile, because that \( \alpha = (K_n^T K_n)^{-1} K_n^T (y - b \bar{1}_n) \), the term indicative of the generalization ability, i.e., \( \alpha^T K_n \alpha \) is also a function of the selected columns. Thus, the selected columns can be seen the parameters tuning the trade-off between the empirical risk and the generalization ability of the sparse LS-SVM. The sparse LS-SVM trained by the application of the KMP algorithm to Equation (14) is referred to as “KMP-SSE” henceforth. Because the “KMP-SSE” algorithm selects columns from the full kernel matrix so that the sum of squared error for the linear system is minimized.

Nevertheless, for classification, the objective is the optimization of the generalization performance for LS-SVMs. A widely-used metric for the generalization performance is the leave-one-out cross validation (LOOCV) error. This paper proposed to select columns from the full kernel matrix, iteratively, which minimizes the LOOCV error. In the subsequent subsections, it is demonstrated that: 1) the LOOCV error pertaining to a specific setting on \( K_n \), can be accurately computed; 2) the LOOCV error can be efficiently calculated, due to the QR decomposition of \( K_n \).

### 4.1. Accurate Computation of the LOOCV Error

Both cases of \( b = 0 \) and \( b \neq 0 \) are discussed.

#### 4.1.1. \( b = 0 \)

Assuming that \( n \) columns from the full kernel matrix have been selected and formed the matrix \( K_n \), the solution to from Equation (14) is:

\[
\alpha_n = (K_n^T K_n)^{-1} K_n^T y
\]  

(15)

For the \( i \)-th \( (1 \leq i \leq \ell) \) training sample, the decision value output by the model parameterized by \( (\alpha_n, K_n) \), denoted as \( y'_i \), is

\[
y'_i = z'^T \alpha_n = z'^T Q^{-1} K_n^T y
\]  

(16)

and the difference between the actual target value and the predicted value is represented as \( r_i \):

\[
y_i - y'_i = \delta_i
\]  

(17)

During the LOOCV procedure, the removal of the \( i \)-th \( (1 \leq i \leq \ell) \) training sample corresponds to the removal of the \( i \)-th row from \( K_n \) which is represented as

\[
6
\]
as row vector $z^T$. The target vector is updated accordingly to be $K_n^T y - y_i z$. The solution to the updated linear system, denoted as $\alpha_n^{-i}$ can be computed:

$$
\alpha_n^{-i} = (K_n^T K_n - zz^T)^{-1}(K_n^T y - y_i z)
= (Q - zz^T)^{-1}(K_n^T y - y_i z)
= \left((Q^{-1} + \frac{Q^{-1} zz^T Q^{-1}}{1 - z^T Q^{-1} z})\right) (K_n^T y - y_i z)
$$

where $Q = K_n^T K_n$.

By incorporating Equation (16) and Equation (17), it can be further deduced that,

$$
\alpha_n^{-i} = Q^{-1} K_n^T y - y_i Q^{-1} z + \frac{Q^{-1} z}{1 - z^T Q^{-1} z}(y_i - \delta_i) - \frac{z^T Q^{-1} z}{1 - z^T Q^{-1} z}(y_i Q^{-1} z)
= \alpha_n - y_i Q^{-1} z + \frac{Q^{-1} z}{1 - z^T Q^{-1} z}(y_i - \delta_i) - \frac{z^T Q^{-1} z}{1 - z^T Q^{-1} z}(y_i Q^{-1} z)
= \alpha_n - \delta_i Q^{-1} z
\frac{Q^{-1} z}{1 - z^T Q^{-1} z}
$$

Meanwhile, it holds that:

$$
y_i z^T \alpha_n^{-i} = y_i \left(z^T \alpha_n - \frac{z^T Q^{-1} z}{1 - z^T Q^{-1} z}\delta_i\right)
= y_i \left(y_i - \delta_i - \frac{z^T Q^{-1} z}{1 - z^T Q^{-1} z}\delta_i\right)
= 1 - \frac{y_i \delta_i}{1 - z^T Q^{-1} z}
= 1 - \frac{1 - y_i \delta_i}{1 - z^T Q^{-1} z}
$$

Thus the prediction error of the omitted $i$-th sample, is $I((y_i z^T \alpha_n^{-i}) > 0)$ where $I(\cdot)$ is the indicator function. And the overall LOOCV error for the model parameterized by $(K_n, \alpha)$ is $\sum_{i=1}^n I((y_i z^T \alpha_n^{-i}) > 0)$.

4.1.2. $b \neq 0$

The linear pertaining to $n$ selected columns of basis function is:

$$
\begin{bmatrix}
K_n & \mathbf{T}_n \\
\mathbf{T}_n^T & 0
\end{bmatrix}
\begin{bmatrix}
\alpha_n \\
b
\end{bmatrix}
= \begin{bmatrix}
y \\
0
\end{bmatrix}
$$

$$
H_n^T \beta_n = r
$$

where $H_n = \begin{bmatrix}
K_n & \mathbf{T}_n \\
\mathbf{T}_n^T & 0
\end{bmatrix}$, $\beta_n = \begin{bmatrix}
\alpha_n \\
b
\end{bmatrix}$ and $r = \begin{bmatrix}
y \\
0
\end{bmatrix}$.
The removal of \( i \)-th training samples corresponds to the removal of the \( i \)-th row from the matrix \( H_n \), which is represented as \( q^T = [z^T \ 1] \). The target vector is updated \( H_n^r r - y_i q \) accordingly. The removal of \( i \)-th training samples results in a new learning model parameterized by \( (\beta_{n}^{-i}, K_n) \) and:

\[
\beta_{n}^{-i} = (H_n^T H_n - qq^T)^{-1}(H_n^T r - y_i q)
\]
\[
= (\Psi - qq^T)^{-1}(H_n^T r - y_i q)
\]
\[
= \left( \Psi^{-1} + \frac{\Psi^{-1}qq^T}{1 - q^T \Psi^{-1}q} \right)(H_n^T r - y_i q)
\]
\[
= \Psi^{-1}H_n^r y - y_i \Psi^{-1}q + \frac{\Psi^{-1}q}{1 - q^T \Psi^{-1}q}(y_i - \delta_i) - \frac{q^T \Psi^{-1}q}{1 - q^T \Psi^{-1}q}(y_i \Psi^{-1}q)
\]
\[
= \beta_n - y_i \Psi^{-1}q + \frac{\Psi^{-1}q}{1 - q^T \Psi^{-1}q}(y_i - \delta_i) - \frac{q^T \Psi^{-1}q}{1 - q^T \Psi^{-1}q}(y_i \Psi^{-1}q)
\]
\[
= \beta_n - \frac{\delta_i \Psi^{-1}q}{1 - q^T \Psi^{-1}q}
\]

where \( \Psi = H_n^T H_n \) and \( \delta_i = y_i - q^T \beta_n \).

\[
y_i q^T \beta_{n}^{-i} = y_i \left( \frac{q^T \beta_n - \frac{q^T \Psi^{-1}q}{1 - q^T \Psi^{-1}q} \delta_i}{1 - \frac{q^T \Psi^{-1}q}{1 - q^T \Psi^{-1}q}} \right)
\]
\[
= 1 - \frac{y_i \delta_i}{1 - q^T \Psi^{-1}q}
\]

The overall LOOCV error for the model parameterized by \( (H_n, \beta) \) is \( \sum_{i=1}^{n} I(y_i q^T \beta_{n}^{-i}) > 0 \). Cawley and Talbot reported similar results of the LOOCV error estimation for sparse LS-SVMs [2]. Nevertheless, in their sparse LS-SVM model, a regularization parameter was introduced to tune the tradeoff between empirical cost the generalization ability.

4.2. Efficient Computation of the LOOCV Error

Assuming that \( n \) columns from the full kernel matrix have been selected and formed the matrix \( K_n \). From the remaining \( (\ell - n) \) columns, the \((i + 1)\)-iteration identifies and selects the column \( h \) with which the resultant matrix \( [K_n \ h] \) leads to the minimal LOOCV error. Taking the case of \( b = 0 \) for example, it is described next how the \( z^T Q^{-1}z \) and the \( y_i \) of Equation (19) can be computed efficiently, for each choice of \( h \).

As described Section 3.1, the application of QR decomposition of \( K_n \) produces an orthogonal matrix \( P_n \) of \( n \) columns and an triangle \( A_n \) of size \( n \times n \). With the addition of \( h \) to \( K_n \) as the new column, \( P_n \) and \( A_n \) each are appended with a column. Denoting the new column for \( A_n \) as \([b^T \ 1/d]^T\) where \( b \) is a \( n \)-dimensional vector and \( d \) a constant:

\[
A_{n+1} = \begin{bmatrix}
A_n & b \\
0 & 1/d
\end{bmatrix}
\]
It follows that:
\[
A_{n+1}^{-1} = \begin{bmatrix} A_n^{-1} & -dA_n^{-1}b \\ 0 & d \end{bmatrix}
\]  
(24)

And it holds that
\[
(K_n^T K_n)^{-1} = A_n^{-1}(A_n^{-1})^T
\]  
(25)

As \( K_n \) expands into \( K_{n+1} = [K_n \ h] \), the \( i \)-th row of \( K_n \) associated with the \( i \)-th training sample, denoted as \( z^T \) is appended with the \( i \)-th element of \( h \) which is represented by \( a \). Therefore,
\[
[z^T \ a](K_{n+1}^T K_{n+1})^{-1}\begin{bmatrix} z \\ a \end{bmatrix} = [z^T \ a]A_{n+1}^{-1}(A_{n+1}^{-1})^T\begin{bmatrix} z \\ a \end{bmatrix} = z^T A_n^{-1}(A_n^{-1})^T z + d^2 (a - z^T A_n^{-1} b)^2
\]
\[
= z^T (K_n^T K_n)^{-1} z + d^2 (a - z^T A_n^{-1} b)^2
\]

For each of the \( \ell \) training samples, by keeping in the memory the value of \( z^T (K_n^T K_n)^{-1} z \) as well as the \( n \)-dimensional vector \( z^T A_n^{-1} \), the time cost for computing \( z^T (K_{n+1}^T K_{n+1})^{-1} z \) is reduced to primarily the computation of \( z^T A_n^{-1} b \).

Meanwhile, the predicted values for sample \( i \) given matrix \( K_{n+1} \) is also changed to \( [z^T \ a](K_{n+1}^T K_{n+1})^{-1} K_{n+1}^T y \) and:
\[
[z^T \ a](K_{n+1}^T K_{n+1})^{-1} K_{n+1}^T y = [z^T \ a](K_{n+1}^T K_{n+1})^{-1} \begin{bmatrix} K_n^T \\ h^T \end{bmatrix} y
\]
\[
= [z^T \ a]A_{n+1}^{-1}(A_{n+1}^{-1})^T \begin{bmatrix} K_n^T \\ h^T \end{bmatrix} y
\]
\[
= z^T A_n^{-1}(A_n^{-1})^T K_n^T y + d^2 (a - z^T A_n^{-1} b) [h^T y - b (A_n^{-1})^T K_n^T y]
\]
\[
= z^T K_n^3 (K_n^{-1})^T K_n^T y + d^2 (a - z^T A_n^{-1} b) [h^T y - b (A_n^{-1})^T K_n^T y]
\]

For each of the \( \ell \) training samples, by keeping in the memory the old predicted value of \( z^T (K_n^T K_n)^{-1} K_n^T y \) as well as the \( n \)-dimensional vector \( (A_n^{-1})^T K_n^T y \), the time cost for computing the \( y_i \) with respect to the addition of \( h \) into \( K_n \) is reduced to primarily the computation of \( b(A_n^{-1})^T K_n^T y \) and \( h^T y \).

The proposed sparse LS-SVMs by selection of those support vectors which minimize the LOO error rate were referred to as “KMP-LOO” and “KMP-LOOwb” for the case of \( b = 0 \) and \( b \neq 0 \) respectively. The time complexity for the proposed algorithms of “KMP-SSE”, “KMP-LOO” and “KMP-LOOwb” are uniformly \( O(n\ell^2) \) where \( n \) is the number of selected column. Their memory requirement are uniformly \( O(\ell^2) \).
5. Experiments

The proposed algorithms of “KMP-SSE”, “KMP-LOO” and “KMP-LOOwb’ were implemented in Matlab and compared against the standard SVM algorithm, a sparse SVM model called “spSVM” [13], fixed-sized LS-SVM and $L_0$ norm LS-SVM [15], a sparse LS-SVM resulting from low-rank approximation of the kernel matrix call “P-LSSVM” [26].

The Gaussian Radial Basis Function (RBF), in the form of $K(x, x_j) = \exp(-\lambda \|x - x_j\|^2)$, was used as the kernel function. 21 value settings on both $\lambda$ and $C$, which were 2 to the power of $\{-10, -9, \ldots, 9, 10\}$, were evaluated and the optimal settings were the one resulting in the best Leave-One-Out (LOO) cross-validation accuracy. All experiments were run on a Pentium 4 3.2GHz processor under Windows XP with 2 GB of RAM.

5.0.1. The Checkerboard Problem

Experiments were first performed on the well-known checkerboard dataset [9]. The dataset contains 1,000 data points of two classes which forms a pattern similar to the chess board, respectively highlighted by red circles and blue asterisks in Figure 1.

Figure 2 illustrated the checkerboard pattern recognized by the LS-SVM and SVM algorithms. The decision functions of the LS-SVM was parameterized by 1000 support vectors (SVs) while that of the SVM 70 SVs. The optimal settings on the parameters of $(C, \lambda)$ for the LS-SVM and the SVM method were respectively $(2^{-3}, 2^6)$ and $(2^{-10}, 2^3)$.

Figure 3 demonstrated the performances of “KMP-SSE”, “KMP-LOO” and “KMP-LOOwb’”. Figure 3 (a) and (b) illustrated the results of “KMP-SSE” where the number of support vectors were 50 and 60 respectively. Figure 3 (c) and (d) illustrated the results of “KMP-LOO” where the number of support vectors were 50 and 60 respectively. Figure 3 (e) and (f) illustrated the results
Figure 2: the checkerboard pattern recognized by: (a) the standard LS-SVM method where $C = 2^{-3}$ and $\lambda = 2^6$; (b) the standard SVM method where $C = 2^{-10}$ and $\lambda = 2^9$. of “KMP-LOOwb” where the number of support vectors were 50 and 60 respectively. The optimal value for the kernel parameter $\lambda$ was uniformly $2^6$. It can be seen that the KMP-LOO performed better than KMP-SSE and KMP-LOOwb.

5.0.2. Binary Benchmark Problems

Experiments were performed on the following benchmark datasets for binary classification: Banana, Image and Splice which are accessible at http://theoval.cmp.uea.ac.uk/matlab/#benchmarks/. One adult dataset from UCI machine learning repository was also used for performance evaluation of various algorithms.

For the Banana dataset, each of the 20 realizations had 400 training samples and 4900 testing samples. The optimal of the spSVM method setting on $(C, \lambda)$ was $(2^6, 2)$. For the p-LSSVM method, the optimal parameter setting on $(C, \lambda)$ was $(2^{-8}, 2)$. For the KMP methods, the optimal value on $\lambda$ was $2^{-1}$. The parameters for the FS-LSSVM and $L_0$ norm LS-SVM were automatically tuned by the LS-SVM toolbox available at: https://www.esat.kuleuven.be/sista/lssvmlab/

Figure 4 plotted the mean accuracy over the 20 runs across 400 different settings of the number of support vectors (SVs) which was growing from 1 sequentially to 400. It was worth attention that P-LSSVM, KMP-SSE, KMP-LOO and KMP-LOOwb terminated after selecting approximately 111 SVs, 61 SVs, 63 SVs and 61 SVs respectively. This was because these methods ensured the linear independency among the selected basis functions and thus the number of SVs was bounded by the column rank of the kernel matrix.

P-LSSVM can only have up to 112 due to linear dependencies between basis functions. The best accuracy for KMP-SSE, KMP-LOO and KMP-LOOwb were achieved at the settings of 4, 24 and 8 SVs respectively. The best performances for KMP-based methods were comparable to the other methods. With 9 SVs, the accuracy of KMP-LOOwb was 0.8925. With 11 SVs, the accuracy of KMP-LOO reached 0.8916. With 8 SVs, the accuracy of KMP achieved the highest
Figure 3: the checkerboard pattern recognized by the LS-SVM trained by KMP-based methods (a) KMP-SSE: 50 support vectors with $\lambda = 2^6$; (b) KMP-SSE: 60 support vectors with $\lambda = 2^6$; (c) KMP-LOO: 50 support vectors with $\lambda = 2^6$; (d) KMP-LOO: 60 support vectors with $\lambda = 2^6$; (e) KMP-LOOwb: 50 support vectors with $\lambda = 2^6$; (f) KMP-LOOwb: 60 support vectors with $\lambda = 2^6$. 
accuracy of 0.8933 while the highest accuracy with spSVM was 0.8923, requiring 23 SVs. These facts suggested that KMP based methods could give very sparse solution while maintaining outstanding generalization performances.

After selecting around 10 SVs, the prediction accuracies of KMP based methods dropped sharply. This was due to the fact the number of SVs was used as the regularization parameter. The increase in the number of SVs reduced the empirical risk but could potentially cause the degradation of the generalization performances.

For the Splice dataset, each of the 20 realizations had 1000 training samples and 1991 testing samples. The optimal of the spSVM method setting on \((C, \lambda)\) was \((2^2, 2^{-6})\). For the p-LSSVM method, the optimal parameter setting on \((C, \lambda)\) was \((2^{-6}, 2^{-8})\). For the KMP methods, the optimal value on \(\lambda\) was \(2^{-6}\). Figure 5 plotted the mean accuracy over the 20 runs as the number of SVs grew sequentially from 1 to 400. It can be seen that the KMP-SSE method performed better than the rest. Meanwhile, all the methods performed similarly with the number of SVs approaching 400.

For the Image dataset, each of the 20 realizations had 1300 training samples and 786 testing samples. The optimal of the spSVM method setting on \((C, \lambda)\) was \((2^4, 2^{-1})\). For the p-LSSVM method, the optimal parameter setting on \((C, \lambda)\) was \((2^{-7}, 2^{-1})\). For the KMP methods, the optimal value on \(\lambda\) was \(2^{-1}\). Figure 6 plotted the mean accuracy over the 20 runs as a function of the number of SVMs which was growing from 1 sequentially to 400. The KMP-SSE method remained the most accurate method across the 400 settings on the number of SVMs. Meanwhile, KMP-LOO was the second best performing when the number of SVMs was within the range of [1, 9].

For the Adult dataset, each of the 20 realizations had 6000 training samples and 786 testing samples. the optimal of the spSVM method setting on \((C, \lambda)\) was \((2^4, 2^{-1})\). For the p-LSSVM method, the optimal parameter setting on \((C, \lambda)\) was \((2^{-7}, 2^{-1})\). For the KMP methods, the optimal value on \(\lambda\) was \(2^{-1}\). Figure 6 plotted the mean accuracy over the 20 runs as a function of the number of SVMs which was growing from 1 sequentially to 400. The KMP-SSE method remained the most accurate method across the 400 settings on the number of SVMs. Meanwhile, KMP-LOO was the second best performing when the number of SVMs was within the range of [1, 9].

For the Adult dataset, each of the 20 realizations had 6000 training samples
Figure 5: the classification accuracy on the splice dataset of different sparse LS-SVM models as a function of the number of support vectors.

Figure 6: the classification accuracy on the image dataset of different sparse LS-SVM models as a function of the number of support vectors.
Figure 7: the classification accuracy on the adult dataset of different sparse LS-SVM models as a function of the number of support vectors.

and 26561 testing samples. The optimal of the spSVM method setting on \((C, \lambda)\) was \(2^2, 2^{-6}\). For the P-LSSVM method, the optimal parameter setting on \((C, \lambda)\) was \(2^{-1}, 2^{-6}\). For the KMP methods, the optimal value on \(\lambda\) was \(2^{-6}\). Figure 7 plotted the mean accuracy over the 20 runs as the number of SVs grew sequentially from 1 to 400. When the number of SVs was within \([1, 13]\), the methods of KMP-SSE, KMP-LOO and KMP-LOOwb outperformed the other methods. When the number of SVs was within the value range of \([14, 45]\), the KMP-SSE method was the one exhibiting the best performance. With the number of SVs going over 45, the spSVM method became the one possessing the highest classification accuracies. Furthermore, the KMP-SSE algorithm gave the worst performance when the number of SVs went over 200. When the number of SVs was smaller than 100, both KMP-LOO and KMP-LOOwb maintained similar performances to FS-LSSVM, \(L_0\) norm LS-SVM and pLSSVM.

5.0.3. Multiclass Problems

Experiments were also conducted on DNA dataset which was of three classes. The dataset was downloadable from the webpage for the LIBSVM toolbox [3]. The one-vs-rest learning strategy for multiclass SVMs was adopted [17]. At each node which accommodated a binary SVM, the class which possessed the largest number of training samples was regarded as the positive class and all the remaining samples were grouped as the negative class.

For FS-LSSVM and \(L_0\) norm LS-SVM, the optimal values for the parameters of \((C, \lambda)\) were \(2^{-5}, 2^{-6}\) and \(2^3, 2^{-6}\). For spSVM, the optimal parameter setting was \(2^2, 2^{-6}\). For P-LSSVM, the optimal parameter setting was \(2^{-3}, 2^{-6}\). For KMP methods, the optimal parameter setting was \(\lambda = 2^{-6}\). The training set for the binary classifier at the top has 2548 samples, composed of 80% of the
Figure 8: the accuracy corresponding to the 4225 value settings of the number of SVs for the two binary classifiers for different sparse LS-SVM algorithm. Graph (g) plotted the best accuracy for each method.
samples from each of the three classes. 65 different settings on the number of SVs was evaluated, which was 25 to 2265 at a step of 35. The training set for the binary classifier at the bottom was composed of 1225 samples from two classes. The number of SVs varied from 12 to 1100 at a step of 17, which were altogether 65 different settings. It produced altogether 4225 (= 65 × 65) settings for the parameter pair of (#sv1, #sv2) where #sv1 is the number of SVs for the top binary classifier and #sv2 for the bottom one. Interestingly, for the proposed KMP based methods, this corresponded to 4225 different settings on the pair of regularization parameters associated with the pair of binary classifiers. While traditional one-vs-rest and one-vs-one multiclass SVMs adopted a uniform value for the regularization parameters of all the binary classifiers [10], the proposed KMP-based multiclass LS-SVMs allows for different parameter settings on different binary LS-SVMs. 20 realizations of the DNA dataset were generated and the prediction accuracy henceforth referred to the mean accuracy over the 20 random realizations.

In Figure 8, subplots (a)–(g) plotted the accuracy for the 4225 settings on (#sv1, #sv2) for various sparse LS-SVM models. The y-axis of each subplot indicated the prediction accuracy and the x-axis ranged from 1 to 4225. In each subplot, the legend of each plot showed the associated sparse LS-SVM algorithm and the accuracy for each one of the 4225 settings was indicated by a circle in blue. Neighborhoods that featured a large number of blue circles exhibited a shade of dark blue. The subplot (h) plotted the best accuracy for the sparse LS-SVM, as well as one-vs-rest SVM whose legend label was “ovrSVM” and one-vs-one SVM whose legend label was “ovoSVM”. The settings on (#sv1, #sv2) for the best accuracy of KMP-SSE, KMP-LOO, KMP-LOOwb, spSVM, FS-LSSVM, L0 norm LS-SVM, and P-LSSVM were (2195, 1066), (2230, 1100), (2265, 1083), (1740, 726), (2230, 1100), (2265, 1032) and (2160, 913) respectively. It can be clearly observed that the proposed KMP-based methods achieved highest prediction accuracies, with only P-LSSVM and one-vs-rest SVM demonstrating comparable accuracies.

6. Discussions and Conclusions

Alternatively, the proposed sparse LS-SVM model can be viewed as solutions to the following optimization problem:

\[
\min_{w, b, \xi} \|w\|_0 + \frac{1}{2} C \sum_{i=1}^{\ell} \xi_i^2
\]

s.t. \( w^T \Phi(x_i) + b = y_i - \xi_i \quad \forall i \)

With the requirement of \( \|w\|_0 = n \) and the application of \( w = \sum_{i=1}^{n} \alpha_i \Phi(x_i) \), Equation (26) was reduced to Equation (20).

Equation (26) was similar to the one used by Mall and Suykens [15]. But it was further proposed to represent \( \|w\|_0 \) by \( \sum_{i=1}^{n} \lambda_i \alpha_i \) where \( \lambda_i \) was the “weight” introduced for the Lagrangian multiplier \( \alpha_i \).
On the other hand, Zhou used the rank-$n$ Nyström approximation to represent the kernel matrix denoted as $K$ [26]:

$$K \approx K_{MB} K_{BM}^{-1} (27)$$

where $|M| = \ell$ containing indexes of all the samples and $|B| = n$ indexes of the $n$ support vectors. It was then proposed to select, iteratively, the $n$ support vectors which minimized the trace norm of $(K - K_{MB} K_{BM}^{-1})$. An alternative strategy was to select, at each iteration, a support vector that caused the least perturbation to the dual objective function.

In contrast, the proposed algorithms in this article selected support vectors which minimized either the sum of squared error or the LOOCV error. Nevertheless, both the proposed algorithms and Zhou’s method ensured the linear independencies between selected support vectors.

Empirically, on binary problems, the proposed algorithms suggested advantages in building extreme sparse LS-SVMs while maintaining outstanding generalization performances. For multiclass problems, the proposed methods used different number of support vectors for different binary problems that the multiclass problem was decomposed into. This was equivalent to different regularization parameters for different binary classifiers, which was different from the tradition strategy of adopting a uniform regularization parameter for all the binary classifiers. This was likely to contribute to improvement of prediction accuracy, as suggested by experimental results on the DNA dataset.

7. Appendix

First, the matrix $H$ is partitioned:

$$H = \begin{bmatrix} \Pi & h \\ h^T & h_\ell \end{bmatrix} \quad (28)$$

where the matrix $\Pi \in \mathbb{R}^{(\ell-1) \times (\ell-1)}$, the vector $h \in \mathbb{R}^{\ell-1}$ and $h_\ell \in \mathbb{R}$. Consequently, $\alpha = \begin{bmatrix} \alpha \cr \alpha_\ell \end{bmatrix}$ and $y = \begin{bmatrix} y \cr y_\ell \end{bmatrix}$ can be accordingly partitioned a vector of $(\ell - 1)$ dimensions and a constant pertaining to the last training sample.

It holds that

$$h^T \alpha + \alpha_\ell h_\ell = y_\ell - b \quad (29)$$

$$\overrightarrow{1}^T \alpha + \alpha_\ell = 0 \quad (30)$$

The above two equations give:

$$b = y_\ell - h^T \alpha + h_\ell \overrightarrow{1}^T \alpha \quad (31)$$

Meanwhile, it satisfies that

$$\Pi \alpha - h \overrightarrow{1}^T \alpha + b \overrightarrow{1} = y \quad (32)$$
Substituting Equation (31) into Equation (32), it follows that:
\[
(H - h T - \overrightarrow{1} h + h \ell \overrightarrow{1} T) \alpha = \overrightarrow{y} - y \ell \overrightarrow{1}
\]
(33)

Denoting the \((\ell - 1) \times (\ell - 1)\) coefficient matrix for Equation (32) as \(G\), it has entries
\[
G_{ij} = (H_{ij} - h_i - h_j + h \ell)
\]
where \(H_{ij}\) are the entries of the matrix \(H\), \(h_i\) and \(h_j\) are respectively the \(i\)-th and the \(j\)-th entry of the vector \(h\).

It can be seen that:
\[
G_{ij} = H_{ij} - h_i - h_j + h \ell
\]
(34)
\[
= \left[\Phi(x_i) - \Phi(x_j)\right]^T \left[\Phi(x_i) - \Phi(x_j)\right] + 2 \gamma \delta_{ij}
\]
where \(\gamma = C^{-1}, \delta_{ij} = 1\) when \(i = j\) and 0 otherwise.

For any non-zero column vector \(v\) of \((\ell - 1)\) dimensions, it holds that
\[
v^T G v = \sum_{i,j=1}^{\ell} v_i v_j G_{ij}
\]
\[
= \sum_{i,j=1}^{\ell} v_i v_j \left[\Phi(x_i) - \Phi(x_j)\right]^T \left[\Phi(x_i) - \Phi(x_j)\right] + 2 \gamma \sum_{i,j=1}^{\ell} v_i v_j \delta_{ij}
\]
\[
= \|\sum_{i=1}^{\ell} v_i [\Phi(x_i) - \Phi(x_\ell)]\|^2 + 2 \gamma \sum_{i=1}^{\ell} v_i^2 > 0
\]

References


