Polynomial componentwise LS-SVM: fast variable selection using low rank updates

Fabian Ojeda, Tillmann Falck, Bart De Moor and Johan A.K. Suykens

***Abstract*— This paper describes a Least Squares Support Vector Machines (LS-SVM) approach to estimate additive models as a sum of non-linear components. In particular, this work discuses the *low rank* matrix modifications for componentwise polynomial kernels, which allow the factors of the modified kernel-matrix to be directly updated. The main concept refers to the use of a valid explicit *feature map* for polynomial kernels in an additive setting. By exploiting the structure of such feature map the model parameters of the classification/regression problem can be easily modified and updated when new variables are added. Therefore, the low rank updates constitute an algorithmic tool to efficiently obtain the model parameters once the system has been altered in some minimal sense. Such strategy allows, for instance, the development of algorithms for sequential variable ranking in high dimensional settings, while non-linearity is provided by the polynomial feature map. Moreover relevant variables can be robustly ranked using the closed form of the leave-one-out (LOO) error estimator, obtained as a by-product of the low rank modifications.**

I. Introduction

Instead of defining the feature space explicitly, kernel based methods indirectly specify the feature space using a positive-definite kernel function $\kappa : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, that evaluates the inner product between pairs of vectors $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathbb{R}^d,\ i, j = 1, \ldots, n$. The kernel function and the high dimensional (possibly infinite) feature map are related by means of Mercer's theorem $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$, with $\phi(\cdot) : \mathbb{R}^d \to \mathbb{R}^{d_h}$ [1]. The main concept in this work however, relies on the use of an explicit finite dimensional *feature map* $\phi(\cdot)$ for additive polynomial kernels. By exploiting the structure of such feature map, the model parameters of the least squares classification/regression problem can be easily modified and updated when a new variable enters the current model. Such strategy allows, for instance, the construction of efficient algorithms for sequential variable ranking in high dimensional settings.

The support vector machine (SVM) method was initially introduced to solve binary classification problems. Starting from statistical learning theory, the solution to the SVM method is given by a convex quadratic programming (QP) problem [2]. Least squares support vector machine (LS-SVM) classifiers consist of a modified version to the standard SVM and the solution results from a set of linear equations [3]. Successful applications using these two techniques can be extensively found in the machine learning and data mining domains. Particularly, the recursive feature elimination (RFE) algorithm introduced in [4], has been widely acknowledged in the variable selection context. In such procedure, variables are successively eliminated during the training of a sequence of SVM classifiers. The algorithms presented in [5] and [6] make use of gradient procedures on generalization error bounds, while in [7] and [8] direct optimization methods aim at solving the joint classification - variable selection problem in one single formulation.

This study uses LS-SVM as core classifiers and in particular the componentwise LS-SVM proposed in [9], [10]. The use of low rank modifications in the LS-SVM structure was previously presented in [11], leading to efficient algorithms for sequential variable ranking wherein model parameters were updated instead of recalculated. However the analysis was restricted to the use of linear kernels. The presented paper advances precisely in this direction by providing a methodology to incorporate non-linearities using componentwise polynomial kernels. This raises the question on how the factors of the modified kernel-matrix can be efficiently updated instead of recomputed. Within the model specification three purposes are recognized: (i) variables are sequentially ranked using componentwise LS-SVM, (ii) efficient model updating for linear and componentwise polynomial kernels, (iii) model interpretation with the imposed additive structure [12].

This manuscript is organized as follows. Section II reviews the basics of Least Squares Support Vector Machines, while section III summarizes the extension to the componentwise LS-SVM model. Section IV outlines the explicit feature map for polynomial kernels and shows how low rank matrices result from the imposed additive structure. To update the componentwise LS-SVM parameters, methods based on the Cholesky decomposition are given in section V. Experimental results of the proposed methodology are presented in section VII.

II. Least squares support vector machines

Given $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n,\ \boldsymbol{x}_i \in \mathbb{R}^d,\ y_i \in \mathbb{R}$, the input-output training data, the regularized cost function for least-squares support vector machines (LS-SVM) is [3]

$$\begin{aligned} \min_{\boldsymbol{w},b,e} \mathcal{J}(\boldsymbol{w}, e) \quad &= \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \gamma\frac{1}{2}\sum_{i=1}^n e_i^2 \\ \text{s.t.} \quad & y_i = \boldsymbol{w}^T\phi(\boldsymbol{x}_i) + b + e_i,\ i = 1, \ldots, n \end{aligned} \tag{1}$$

where $\gamma > 0$ is a regularization parameter and, $e_i \in \mathbb{R}$, denote the error residuals of the outputs y_i. Instead of defining the feature space explicitly, $\phi(\cdot) : \mathbb{R}^d \to \mathbb{R}^{d_h}$ which might be

The authors are with the Department of Electrical Engineering ESAT-SCD-SISTA, Katholieke Universiteit Leuven, Kaastelpark Arenberg 10, B-3001 Leuven, Belgium (email: {fabian.ojeda, tillmann.falck, bart.demoor, johan.suykens} @esat.kuleuven.be).

infinite dimensional, kernel based methods specify the feature space indirectly by the use of a positive-definite kernel function $\kappa : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$. The kernel function evaluates the inner product between pairs of vectors $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$, for given $\boldsymbol{x}_i, \boldsymbol{x}_j,\ i, j = 1, \ldots, n$. The dual problem for the LS-SVM classifier is then given as [3]:

$$\left[\begin{array}{c|c} \boldsymbol{\Omega} + \gamma^{-1}\boldsymbol{I}_n & \boldsymbol{1} \\ \hline \boldsymbol{1}^T & 0 \end{array}\right] \left[\begin{array}{c} \boldsymbol{\alpha} \\ \hline b \end{array}\right] = \left[\begin{array}{c} \boldsymbol{y} \\ \hline 0 \end{array}\right]. \tag{2}$$

In (2), $\boldsymbol{y} = [y_1; \ldots; y_n]$ and $\boldsymbol{\alpha} = [\alpha_1; \ldots; \alpha_n]$ and $\boldsymbol{\Omega} \in \mathbb{R}^{n \times n}$, with $\boldsymbol{\Omega}_{ij} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The prediction for a data point $\boldsymbol{x}$ is given by

$$f(\boldsymbol{x}) = \operatorname{sign}\left(\sum_{i=1}^{n} \alpha_i \kappa(\boldsymbol{x}, \boldsymbol{x}_i) + b\right). \tag{3}$$

The optimal parameters, $\boldsymbol{\alpha}, b$ are determined by the system of linear equations in (2). Examples of widely used positive definite kernel functions include

$$\text{Linear :} \qquad \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^T \boldsymbol{x}_j \tag{4}$$

$$\text{Polynomial :} \qquad \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\tau + \boldsymbol{x}_i^T \boldsymbol{x}_j)^p \tag{5}$$

$$\text{Gaussian :} \qquad \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\eta ||\boldsymbol{x}_i - \boldsymbol{x}_j||_2^2\right) \tag{6}$$

where $p \in \mathbb{Z}^+$, and $\eta, \tau \geq 0$ are the corresponding *kernel parameters*. Since the kernel functions are required to be positive-(semi)definite, they give rise to Gram matrices over the set of input data points.

Remark 1: The elements of the Gram matrices associated to the kernel functions mentioned above are in general $\boldsymbol{\Omega}_{ij} = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j) = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$, with $i, j = 1, \ldots, n$. By denoting the matrix $\Phi = [\phi(\boldsymbol{x}_1), \ldots, \phi(\boldsymbol{x}_i), \ldots, \phi(\boldsymbol{x}_n)] \in \mathbb{R}^{d_h \times n}$, the Gram matrix $\boldsymbol{\Omega} \in \mathbb{R}^{n \times n}$, becomes

$$\boldsymbol{\Omega} = \left[\begin{array}{c} \phi(\boldsymbol{x}_1)^T \\ \vdots \\ \phi(\boldsymbol{x}_n)^T \end{array}\right] [\phi(\boldsymbol{x}_1), \ldots, \phi(\boldsymbol{x}_n)] = \Phi^T \Phi\ . \tag{7}$$

III. Componentwise support vector machines

This section considers the class of models that are additive in every input. Let the superscript for x^k, denote the kth component of an input vector $\boldsymbol{x} \in \mathbb{R}^d$, for all $k = 1, \ldots, d$. To represent the collection of a single component the following notation is used $\boldsymbol{x}^k = [x_1^k, \ldots, x_i^k, \ldots, x_n^k]^T$. Following the LS-SVM formulation given in the previous section, the componentwise regression/classification model takes the form $f(\boldsymbol{x}) = \sum_{k=1}^{d} \boldsymbol{w}_k^T \phi_k(x^k) + b$, where for the kth component $\phi_k(\cdot) : \mathbb{R} \to \mathbb{R}^{d_h}$ is a possibly infinite dimensional mapping and $\boldsymbol{w}_k \in \mathbb{R}^{d_h}$ the corresponding vector of model parameters. Therefore, the penalized sum of squares function, in contrast to that in Eq. (1), becomes [9]

$$\begin{aligned} \min_{\boldsymbol{w}_k, b, e} \mathcal{J}(\boldsymbol{w}_k, e) =& \frac{1}{2} \sum_{k=1}^{d} \boldsymbol{w}_k^T \boldsymbol{w}_k + \gamma \frac{1}{2} \sum_{i=1}^{n} e_i^2 \\ \text{s.t.} \quad y_i =& \sum_{k=1}^{d} \boldsymbol{w}_k^T \phi_k(x_i^k) + b + e_i,\ i = 1, \ldots, n. \end{aligned} \tag{8}$$

Along the lines of standard LS-SVM, by taking the conditions for optimality and application of the kernel trick $\kappa^k(x_i^k, x_j^k) = \phi_k(x_i^k)^T \phi_k(x_j^k)$, the following set of linear equations is obtained

$$\left[\begin{array}{c|c} \boldsymbol{\Omega}^d + \gamma^{-1}\boldsymbol{I}_n & \boldsymbol{1} \\ \hline \boldsymbol{1}^T & 0 \end{array}\right] \left[\begin{array}{c} \boldsymbol{\alpha} \\ \hline b \end{array}\right] = \left[\begin{array}{c} \boldsymbol{y} \\ \hline 0 \end{array}\right], \tag{9}$$

where as before $\boldsymbol{\Omega}^d = \sum_{k=1}^{d} \boldsymbol{\Omega}^k \in \mathbb{R}^{n \times n}$, and $\boldsymbol{\Omega}_{ij}^k = \kappa^k(x_i^k, x_j^k)$, is the kernel evaluation at the kth component between the ith and the jth data points. Note that the main difference between the dual models in (2) and (9), is in fact expressed solely in terms of the used kernels [10]. Componentwise models of this type have been also used in the context of survival models [13]. A new data point $\boldsymbol{x} \in \mathbb{R}^d$ can be evaluated as

$$f(\boldsymbol{x}) = \operatorname{sign}\left(\sum_{i=1}^{n} \alpha_i \sum_{k=1}^{d} \kappa^k(\boldsymbol{x}^k, \boldsymbol{x}_i^k) + b\right). \tag{10}$$

IV. Polynomial updates

In this section, a more detailed description of the type of kernels considered in this work is given. The key idea is to view the overall kernel matrix as the sum of individual contributions of kernels when applied individually on every input. These kernels are denoted as *componentwise* kernels and in the case of polynomial functions, their structure provides with low rank kernel matrices whose rank, for an univariate input, is determined by the chosen polynomial degree. Starting from the linear kernel (4), the associated Gram matrix $\boldsymbol{\Omega}$ in (7) can alternatively be written in the form of outer products, that is

$$\boldsymbol{\Omega} = \left[\boldsymbol{x}^1, \ldots, \boldsymbol{x}^d\right] \left[\begin{array}{c} \boldsymbol{x}^1 \\ \vdots \\ \boldsymbol{x}^d \end{array}\right]^T = \sum_{k=1}^{d} \boldsymbol{x}^k {\boldsymbol{x}^k}^T = \boldsymbol{X}\boldsymbol{X}^T, \tag{11}$$

with $\boldsymbol{X} = \left[\boldsymbol{x}^1, \ldots, \boldsymbol{x}^d\right]$. However, for more general kernels such as the polynomial or Gaussian kernel, the Gram matrix cannot be simply computed as the sum of outer products. Therefore, the family of kernels that are additive in every input is instead considered. That is

$$\kappa^d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{k=1}^{d} \kappa^k(x_i^k, x_j^k). \tag{12}$$

In this context, the individual kernels are defined on a single univariate input and hence functions of two scalar variables. The elements of the corresponding Gram matrix are $\boldsymbol{\Omega}_{ij}^k = \kappa(x_i^k, x_j^k)$, that is, the kernel evaluated in the kth component between the ith and the jth data points. The overall Gram matrix in this additive setting then takes the form [9]

$$\boldsymbol{\Omega}^d = \sum_{k=1}^{d} \boldsymbol{\Omega}^k. \tag{13}$$

It is clear that for the linear kernel $\boldsymbol{\Omega}^d = \boldsymbol{\Omega}$, that is the outer product definition given above in (11) and consequently also equivalent to that in (7).

For the polynomial kernel of degree p with $\tau = 1$ (cf. Eq(5)), $\kappa_p : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, a valid explicit feature map $\phi_p(\cdot)$ for a given z is [14], [15]:

$$\phi_p(z) = \left[1, \sqrt{\binom{p}{1}} z, \ldots, \sqrt{\binom{p}{p-1}} z^{p-1}, z^p\right]^T \tag{14}$$

with $\phi_p(\cdot) : \mathbb{R} \to \mathbb{R}^{p+1}$. The Gram matrix for the additive polynomial kernel is then

$$\boldsymbol{\Omega}_p^d = \sum_{k=1}^{d} \boldsymbol{\Omega}_p^k \qquad \text{with } \boldsymbol{\Omega}_p^k = \sum_{l=0}^{p} (\phi_l \circ \boldsymbol{x}^k)(\phi_l \circ \boldsymbol{x}^k)^T \tag{15}$$

where $(\phi_l \circ \boldsymbol{x}^k)$ corresponds to the elementwise application of the function $\phi_l = \sqrt{\binom{p}{l}} z^l$ to the elements of vector $\boldsymbol{x}^k$. In matrix notation it reads

$$\boldsymbol{\Omega}_p^k = \Phi_p^k \Phi_p^{k\,T} \tag{16}$$

where $\Phi_p^k = \left[(\phi_0 \circ \boldsymbol{x}^k), \ldots, (\phi_p \circ \boldsymbol{x}^k)\right]$ is a $n \times (p+1)$ matrix. Since for a Gram matrix

$$\text{rank}\left(\Phi_p^k\right) = \text{rank}\left(\Phi_p^k \Phi_p^{k\,T}\right) = \text{rank}\left(\boldsymbol{\Omega}_p^k\right) \tag{17}$$

holds, then for the additive model based on the polynomial kernel, the rank of $\boldsymbol{\Omega}_p^k$ for an input variable $\boldsymbol{x}^k$ is at most $p+1$. As a consequence, the polynomial kernel matrix $\boldsymbol{\Omega}_p^d$ for all inputs, $k = 1, \ldots, d$, and constant degree p can be written as the sum of d rank-$(p+1)$ matrices

$$\text{rank}\left(\boldsymbol{\Omega}_p^d\right) = \text{rank}\left(\sum_{k=1}^{d} \boldsymbol{\Omega}_p^k\right) \leq \sum_{k=1}^{d} \text{rank}\left(\boldsymbol{\Omega}_p^k\right). \tag{18}$$

The use of these componentwise (additive) models for input variable ranking is presented in the following sections. Moreover efficient algorithms are proposed along the lines of the earlier work in [11].

Remark 2: Rank-one linear kernel. Setting $\tau = 0$ and $p = 1$ in the inhomogeneous polynomial kernel (cf. (5)) gives the linear kernel and therefore the rank of the mapping $\Phi(\boldsymbol{x}^k)$ for a single variable $\boldsymbol{x}^k$ equals 1 [11].

Remark 3: Rank-one homogeneous polynomial kernel. For a given $p \in \mathbb{Z}^+$ and setting $\tau = 0$, the inhomogeneous polynomial kernel (cf. (5)) reduces to the homogeneous kernel [1]. The rank of the matrix $\Phi_p(\boldsymbol{x}^k)$ for a single variable $\boldsymbol{x}^k$ reduces also to 1, for the lower order monomials receive a zero weighting coefficient.

V. Low rank modifications

The system of linear equations in (9) can be alternatively factorized into a positive-definite system and hence the Cholesky decomposition can be used to obtain its solution [16]. In order to obtain the model parameters $\boldsymbol{\alpha}, b$, the solution of the two triangular systems $\boldsymbol{L}\boldsymbol{L}^T\boldsymbol{\chi} = \boldsymbol{y}$ and $\boldsymbol{L}\boldsymbol{L}^T\boldsymbol{\nu} = \mathbf{1}$, is required. The matrix $\boldsymbol{L}$ denotes the Cholesky factor of the positive-definite matrix $\boldsymbol{L}\boldsymbol{L}^T = \boldsymbol{\Omega} + \gamma^{-1}\boldsymbol{I}_n$. The model parameters in (2) or (9) are then given by

$$b = \mathbf{1}^T\boldsymbol{\chi}(\mathbf{1}^T\boldsymbol{\nu})^{-1}, \qquad \boldsymbol{\alpha} = \boldsymbol{\chi} - b\boldsymbol{\nu}. \tag{19}$$

Adding a new variable $\boldsymbol{x}^k$ to the system of linear equations represents, in the simplest case, a *rank-one* modification to the Cholesky factor $\boldsymbol{L}$ [17]

$$\tilde{\boldsymbol{L}}\tilde{\boldsymbol{L}}^T = \boldsymbol{L}\boldsymbol{L}^T + \boldsymbol{x}^k\boldsymbol{x}^{k\,T}. \tag{20}$$

In the same line if the polynomial mapping $\phi_p(\cdot)$ of the variable $\boldsymbol{x}^k$ is considered, then according to (16) and (17), the *rank-$(p+1)$* modification of the Cholesky factor reads

$$\tilde{\boldsymbol{L}}\tilde{\boldsymbol{L}}^T = \boldsymbol{L}\boldsymbol{L}^T + \Phi_p^k\Phi_p^{k\,T}. \tag{21}$$

Low rank updates of this form can always be done by applying $(p+1)$ rank-one updates sequentially with the columns of Φ_p^k. For the special *rank-one* case denote $\boldsymbol{u}$ as the variable $\boldsymbol{x}^k$ in the linear kernel, or a single column of Φ_p^k in the polynomial kernel. The modified Cholesky factor is obtained as follows

$$\begin{aligned} \tilde{\boldsymbol{L}}\tilde{\boldsymbol{L}}^T &= \boldsymbol{L}\boldsymbol{L}^T + \boldsymbol{u}\boldsymbol{u}^T \\ &= \boldsymbol{L}(\boldsymbol{I} + \boldsymbol{q}\boldsymbol{q}^T)\boldsymbol{L}^T \\ &= \boldsymbol{L}\bar{\boldsymbol{L}}\bar{\boldsymbol{L}}^T\boldsymbol{L}^T \end{aligned} \tag{22}$$

where $\boldsymbol{q}$ is the solution to the triangular system $\boldsymbol{L}\boldsymbol{q} = \boldsymbol{u}$, while the Cholesky factor of the elementary matrix $\boldsymbol{I} + \boldsymbol{q}\boldsymbol{q}^T$ is denoted by $\bar{\boldsymbol{L}}$. Since the product of two lower-triangular matrices is also a lower-triangular matrix, then the matrix $\boldsymbol{L}\bar{\boldsymbol{L}}$ is consequently the triangular factor required. The modified matrix $\tilde{\boldsymbol{L}}$ can be directly computed from $\boldsymbol{L}$ [17], which is simpler and cheaper to compute than starting from scratch. Therefore, the updated model parameters can be obtained from:

$$\tilde{b} = \mathbf{1}^T\tilde{\boldsymbol{\chi}}(\mathbf{1}^T\tilde{\boldsymbol{\nu}})^{-1}, \qquad \tilde{\boldsymbol{\alpha}} = \tilde{\boldsymbol{\chi}} - \tilde{b}\tilde{\boldsymbol{\nu}} \tag{23}$$

where $\tilde{\boldsymbol{L}}\tilde{\boldsymbol{L}}^T\tilde{\boldsymbol{\chi}} = \boldsymbol{y}$ and $\tilde{\boldsymbol{L}}\tilde{\boldsymbol{L}}^T\tilde{\boldsymbol{\nu}} = \mathbf{1}$. For a positive update $\tilde{\boldsymbol{L}}$ remains positive-definite ($\tilde{\boldsymbol{L}} \succ \mathbf{0}$) as long as $\boldsymbol{L}$ is well-conditioned. On the other hand, negative downdates break down if $\boldsymbol{L}\boldsymbol{L}^T - \boldsymbol{u}\boldsymbol{u}^T$ is not positive definite. This may lead to numerical errors and instability [18] [16]. This paper concerns only the use of positive updates. Similar versions of the algorithms presented here can be derived using stable downdating procedures [17].

VI. Efficient leave-one-out computation

This section elaborates on the computation of the leave-one-out (LOO) estimator as a by-product of the modified system of equations when a new variable enters the model. This motivates the use of a robust, yet efficient estimator to rank important variables in a sequential fashion. This is similar to wrapper algorithms [19] and stagewise/stepwise regression [20]. The system of linear equations in (2), can be represented in the form of an arbitrary block structured matrix $\boldsymbol{A}$, that is

$$\left[\begin{array}{c|c} \boldsymbol{H} & \mathbf{1} \\ \hline \mathbf{1}^T & 0 \end{array}\right] = \left[\begin{array}{c|c} \boldsymbol{A}_{11} & \boldsymbol{a}_{12} \\ \hline \boldsymbol{a}_{12}^T & a_{22} \end{array}\right] = \boldsymbol{A} \tag{24}$$

where $\boldsymbol{a}_{12}$, $\boldsymbol{a}_{21}$ are column vectors and a_{22} is a scalar, and $\boldsymbol{H} = \boldsymbol{\Omega} + \gamma^{-1}\boldsymbol{I}_n$. According to [21], by exploiting the block structure on (2), the residual of the leave-one-out prediction for the i-th training pattern can be expressed in closed form. Given

the inverse of $\boldsymbol{A}$ and the model parameters $\boldsymbol{\alpha}$ are known for the full training set, the residual of the leave-one-out prediction for the i-th training pattern is [21]:

$$r_i^{(-i)} = y_i - \hat{y}_i^{(-i)} = \frac{\alpha_i}{(\boldsymbol{A}^{-1})_{ii}}. \tag{25}$$

The leave-one-out estimate of the sum of squares error, or PRESS statistic, is then given by:

$$\text{error}_{\text{press}} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i^{(-i)}\right)^2. \tag{26}$$

Since $\boldsymbol{\alpha}$ is obtained via explicit inversion of $\boldsymbol{A}$, the leave-one-out error can be computed using the information already available from the training phase. This constitutes a more efficient alternative than n times solving the system of linear equations in (2). The computational complexity of this leave-one-out method reduces to $\mathcal{O}(n)$. Instead of directly using the inverse of $\boldsymbol{A}$, the Cholesky factor $\boldsymbol{L}$ of $\boldsymbol{H}$ comprises a numerically more stable approach to obtain the residuals $r_i^{(-i)}$ (see [21] for details).

VII. Experimental results

In this section, experimental results illustrate the use of low rank updates when using componentswise LS-SVM classifiers and polynomial kernels. This approach overcomes, and reduces the high computational cost incurred during variable selection. Synthetic data and the benchmark dataset collected by [6] are employed to compare the usefulness of the methodology hereby exposed. The implementation of the proposed approach is outlined in Algorithm 1. In step 9, the Matlab routine `cholupdate` or the code fragment in [17, pg. 43] can be used to update the Cholesky factor.

A. Toy Data

A synthetic toy data set drawn from an uniform distribution is generated according to the model $y_i = 10\,\text{sinc}(x_i^1) + 20(x_i^2 - 0.5)^2 + 10x_i^3 + 5x_i^4 + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0,1)$ [12]. A sample with $n = 100$ data points and $d = 500$ variables is generated to test the algorithm computational efficiency. For several degrees of the polynomial kernel ($p = 1, \ldots, 10$) the total time needed to compute 50 updates is measured. In Figure 1 each update corresponds to a new variable $\boldsymbol{x}^j,\ j = 1, \ldots, d$, added to set of linear equations. In addition, comparisons with respect to the time to compute the inverse from scratch (dotted line) are shown. A first alternative for efficiency is the use of the eigendecomposition method (dashed line) for the componentwise additive polynomial kernel. The kernel matrix eigendecomposition for each variable is computed from where a direct *rank* $(p+1)$ update is performed to obtain the inverse. The second method (solid line) corresponds to the main methodology introduced in this work, that is, the explicit feature map of the polynomial kernel for a single variable. Using this product form factorization one can also directly apply a rank $(p+1)$ update to compute the inverse. In all these cases the regularization parameter γ and the kernel parameter τ were fixed to a value of 1. Figure 2 presents the average

Algorithm 1 Variable ranking with componentwise polynomial kernels and low rank updates.

Input: Training set $\{\boldsymbol{x}_i, y_i\}_{i=1}^n, \boldsymbol{x}_i \in \mathbb{R}^d, y_i \in \mathbb{R}$, regularization and polynomial kernel hyperparameters (γ, p, τ), number of variables to rank T. **Output:** Ranked variables $\mathcal{S}$, ranking criteria J.

1: Initialize set of active variables $\mathcal{A} = \{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^d\}$.
2: Initialize set of ranked variables $\mathcal{S} = \emptyset$.
3: Initialize Cholesky factor $\boldsymbol{L}_0 = \text{diag}(\sqrt{1/\gamma}, \ldots, \sqrt{1/\gamma}) \in \mathbb{R}^{n\times n}$.
4: **for** $l = 1$ to T **do**
5: **for** $k = 1$ to $|\mathcal{A}|$ **do**
6: $\boldsymbol{L}_k \leftarrow \boldsymbol{L}_0$
7: Compute for $\boldsymbol{x}^k \in \mathbb{R}^{n\times 1}$, the feature map $\Phi_p \in \mathbb{R}^{n\times p+1}$ using Eq. (14).
8: **for** $r = 1$ to $p+1$ **do** {For every column of Φ_p}
9: Update Cholesky factor $\boldsymbol{L}_k \leftarrow \text{cholupdate}(\boldsymbol{L}_k, \Phi_p(:, r))$.
10: **end for**
11: Solve for $\boldsymbol{\alpha}$ using Eq. (19).
12: Compute $\text{error}_{\text{press}}(k)$ using Eq. (25) and Eq. (26).
13: **end for**
14: Select variable $\boldsymbol{x}^c$ such that $c = \text{argmin}_{k\in\{1,\ldots,|\mathcal{A}|\}} \text{error}_{\text{press}}(k)$.
15: Ranking cost $J(l) = \text{error}_{\text{press}}(c)$.
16: Ranked variables $\mathcal{S} \leftarrow \mathcal{S} \cup \boldsymbol{x}^c$.
17: Active variables $\mathcal{A} \leftarrow \mathcal{A} \setminus \boldsymbol{x}^c$.
18: Set Cholesky factor for next iteration $\boldsymbol{L}_0 \leftarrow \boldsymbol{L}_c$.
19: **end for**
20: **return** $\mathcal{S}, J$.

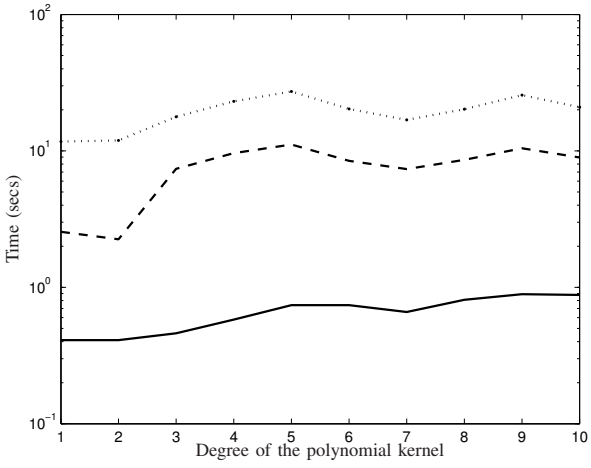

Fig. 1. Computational time for the updates of the regularized Gram matrix when varying the degree of the componentwise additive polynomial kernel.

LOO PRESS with respect to the number of variables ranked and the degree of the polynomial.

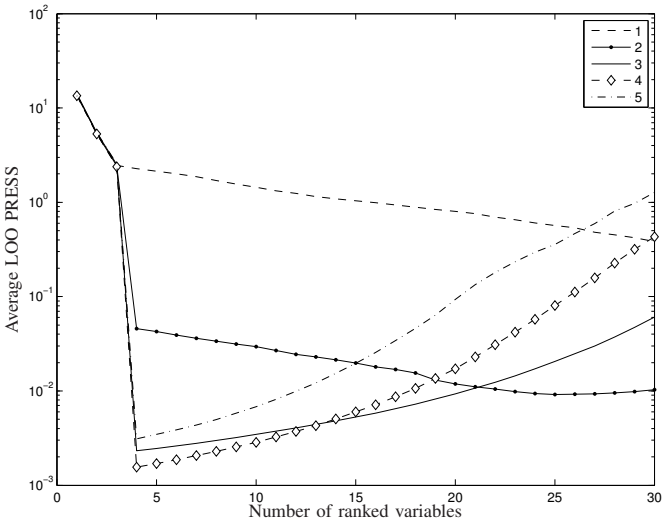

Fig. 2. Leave-one-out PRESS for the synthetic data when different degrees in the componentwise polynomial kernel ($p = 1, \ldots, 5$) are tested. In all but two of the different models ($p = 1, 2$) the minimum averaged LOO PRESS value is attained at 4 variables, which in turn corresponds to the true number of components. Moreover, the polynomial kernel of degree 1 retrieves one spurious variable.

In a second experiment a number of 100 different training and test set partitions are simulated. The size of the sample n is varied from 10 to 100, and it is randomly chosen from the interval $[0, 1]^{100}$, thus $d = 100$. The degree of the polynomial kernel is varied $p = 1, 2, 3$ while $\tau \in \{0, 1\}$. In order to avoid some bias in the sequential forward algorithm, the 4 relevant variables are placed at random positions. In this setting, the goal is to check the capacity of the algorithm to retrieve the relevant variables giving them the highest rank. The regularization parameter γ is fixed to a value of 1 and the results are presented in Table I. In the cases that τ is fixed to zero and the polynomial degree is higher, the lower order terms are discarded. The resulting non-linear mapping may not obey order restrictions, that is linear terms might be dropped while corresponding higher order terms are selected into the model. In fact the best entry for $\tau = 0$ corresponds to that of the linear model [11].

B. Model selection

The final performance of any learning strategy might be heavily influenced by the hyperparameter selection, i.e. a correct choice for (γ, τ). In this work, the hyperparameters are tuned as to minimize the PRESS statistic. For each realization of the data sets, the pair (γ^*, τ^*) minimizing the leave-one-out PRESS statistic is chosen. That pair is then used during the variable selection procedure. The model selection is performed independently for each realization of the data set, such that the standard errors would reflect the variability of the training algorithm and the model selection with respect to changes in the sampling of the data.

C. Real world data

In this section a selection of eleven UCI public domain benchmark data sets used in [6] are considered to evaluate the usefulness of the methodology here exposed. The use of low rank updates/downdates is demonstrated for linear and additive polynomial kernels. The ranking criterion for the variables is based on the PRESS statistic, which is derived from the efficient leave-one-out estimator [21], [22], [11]. Results obtained using standard LS-SVM and LS-SVM with automatic relevance determination (ARD) are taken from [22]. For standard SVM with different bounds, results were collected (though not complete) from previous studies in [23], [5], [24]. Results for each method are presented in the form of the mean error rate over test data for 100 realizations of each dataset (20 in the case of the image and splice datasets), along with the associated standard deviation of the error. Table II summarizes the results obtained, while Figure 3 illustrates the mechanism selecting the number of variables based on the LOO PRESS criterion.

VIII. Conclusions

The work presented in [4] proposes to solve the SVM algorithm only once with the full set of variables. Although the ranking criterion used there depends on the weight norm in the cost function, the support vectors $\boldsymbol{\alpha}$ remain fixed throughout the whole variable ranking. Moreover, even thought such strategy reduces efficiently the computational complexity of the approach from n-QP problems to merely one, it does violate however the optimality conditions in the optimization problem. In this work, the core learning algorithm is the (componentwise) LS-SVM classifier that can be efficiently trained by solving just one system of linear equations. The main concept presented here, however concerns the the use of a valid explicit *feature map* for polynomial kernels in an additive construction. By exploiting the structure of the feature map, it is demonstrated how the model parameters of the classification/regression problem can be easily modified and updated when new variables enter the current model, while still providing non-linear modeling capabilities. This is achieved by the use of low rank updates which in turn constitutes an algorithmic tool for the design of sequential variable ranking algorithms in high dimensional settings. Moreover relevant variables can be robustly ranked using the closed form of the leave-one-out (LOO) error estimator directly obtained as a by-product of the low rank modifications.

Acknowledgment

The authors would like to thank Kristiaan Pelckmans and Vanya Van Belle for helpful discussions over the componentwise LS-SVM. This work was supported by grants and projects for the Research Council KUL (GOA AMBioRICS, GOA MaNet, CoE EF/05/006 Optimization in Engineering(OPTEC), IOF-SCORES4CHEM, several PhD/postdoc & fellow grants), Flemish Government FWO: PhD/postdoc grants, projects G.0452.04 (new quantum algorithms), G.0499.04 (Statistics), G.0211.05 (Nonlinear), G.0226.06 (cooperative systems and optimization), G.0321.06 (Tensors), G.0302.07

TABLE I

SYNTHETIC PROBLEM. AVERAGE GENERALIZATION PERFORMANCE (WITH STANDARD DEVIATION) OVER 100 PARTITIONS OF THE DATA SET. THE CAPACITY TO RECOVER THE FOUR RELEVANT COMPONENTS IN THIS NON-LINEAR PROBLEM IS EXPRESSED AS THE AVERAGE MISSING RATE OVER 100 PARTITIONS. A VALUE CLOSE TO 0 REPRESENTS THAT ALL FOUR RELEVANT VARIABLES ARE SELECTED.

	Algorithm	Training set size					
		10	20	30	40	50	100
$\tau = 0$	LowR-lin	44.64 ± 23.77 3.30	$\mathbf{9.85 \pm 12.05}$ **1.56**	3.96 ± 2.15 1.01	3.61 ± 2.94 1.03	3.11 ± 0.27 0.97	2.72 ± 0.12 0.99
	LowR-poly2	106.64 ± 147.30 3.75	60.66 ± 25.21 3.71	44.58 ± 12.23 3.56	37.96 ± 7.61 3.47	34.50 ± 5.75 3.50	26.28 ± 2.80 3.07
	LowR-poly3	113.52 ± 182.46 3.45	35.75 ± 19.06 2.80	16.87 ± 10.50 1.90	10.69 ± 3.10 1.48	9.19 ± 1.74 1.23	7.34 ± 0.78 1.03
$\tau = 1$	LowR-lin	$\mathbf{40.97 \pm 24.21}$ **3.30**	10.44 ± 12.96 1.60	4.44 ± 3.78 1.14	3.25 ± 0.36 0.98	3.07 ± 0.26 1.00	2.73 ± 0.17 0.95
	LowR-poly2	75.45 ± 89.61 3.50	21.04 ± 26.87 1.72	$\mathbf{1.14 \pm 5.76}$ **0.15**	0.19 ± 0.93 0.04	$\mathbf{0.05 \pm 0.01}$ **0.00**	0.05 ± 0.00 0.00
	LowR-poly3	120.00 ± 221.74 3.68	40.70 ± 39.84 2.41	7.11 ± 15.05 0.72	$\mathbf{0.06 \pm 0.57}$ **0.01**	0.05 ± 0.47 0.01	$\mathbf{0.00 \pm 0.00}$ **0.00**

TABLE II

BENCHMARK DATA: ERROR RATES OF THE COMPARED METHODS. THE RESULTS OF EACH METHOD ARE PRESENTED IN THE FORM OF THE MEAN ERROR RATE AVERAGED OVER 100 TEST DATA REALIZATIONS. X: RESULTS NOT FOUND OR OTHERWISE NOT REPORTED.

	LS-SVM[22]		LowR LS-SVM			SVM		
Kernel	RBF		Linear[11]	Polynomial (15)		RBF		
Algorithm	LS-SVM	LS-SVM ARD	LowR-lin	LowR-poly2	LowR-poly3	$R\|\|\boldsymbol{w}\|\|$[5]	Span[5]	$\|\|\boldsymbol{w}\|\|$[24]
Breast	26.73 ± 0.47	29.08 ± 0.41	27.88 ± 5.14	33.34 ± 4.82	34.06 ± 4.90	26.84 ± 4.71	25.59 ± 4.18	x
Diabetes	23.34 ± 0.17	24.35 ± 0.19	23.55 ± 1.71	25.46 ± 2.20	25.45 ± 1.93	23.25 ± 1.70	23.19 ± 1.67	28.50
Flare	34.22 ± 0.17	34.39 ± 0.19	33.49 ± 1.70	33.82 ± 1.65	33.31 ± 1.78	x	x	x
German	23.55 ± 0.22	26.10 ± 0.26	24.97 ± 2.17	28.29 ± 2.69	28.89 ± 2.59	x	x	x
Heart	16.64 ± 0.36	23.65 ± 0.35	16.41 ± 3.14	16.83 ± 3.25	17.85 ± 3.72	15.92 ± 3.18	16.13 ± 3.11	27.00
Image	3.00 ± 0.16	1.96 ± 0.11	18.44 ± 0.72	10.56 ± 1.03	7.99 ± 0.78	x	x	x
Ringnorm	1.61 ± 0.01	2.11 ± 0.04	25.43 ± 0.55	5.48 ± 0.54	6.18 ± 0.53	x	x	8.40
Splice	10.97 ± 0.16	5.86 ± 0.18	16.13 ± 0.66	10.55 ± 0.41	6.91 ± 0.50	x	x	x
Thyroid	4.68 ± 0.23	4.68 ± 0.20	20.24 ± 5.05	18.57 ± 4.61	11.87 ± 4.13	4.62 ± 2.03	4.56 ± 1.97	x
Twonorm	2.84 ± 0.02	5.18 ± 0.07	2.53 ± 0.18	2.67 ± 0.21	3.43 ± 0.27	x	x	9.30
Waveform	9.79 ± 0.04	13.56 ± 0.14	14.69 ± 0.10	11.13 ± 0.57	11.41 ± 0.56	x	x	x

(SVM/Kernel), G.0320.08 (convex MPC), G.0558.08 (Robust MHE), G.0557.08 (Glycemia2), G.0588.09 (Brain-machine) research communities (ICCoS, ANMMM, MLDM); G.0377.09 (Mechatronics MPC). IWT: PhD Grants, McKnow-E, Eureka-Flite+, SBO LeCoPro, SBO Climaqs, POM. Belgian Federal Science Policy Office: IUAP P6/04 (DYSCO, Dynamical systems, control and optimization, 2007-2011); EU: ERNSI; FP7-HD-MPC (INFSO-ICT-223854), COST intelliCIS, EMBOCOM. Contract Research: AMINAL. Helmholtz: viCERP, ACCM, Bauknecht, Hoerbiger. Bart De Moor is a full professor at the Katholieke Universiteit Leuven, Belgium. Johan Suykens is a professor at the Katholieke Universiteit Leuven, Belgium. The scientific responsibility is assumed by its authors.

REFERENCES

[1] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

[2] V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

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

[4] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46, 2002.

[5] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1):131–159, 2002.

[6] G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for AdaBoost. *Machine Learning*, 42(3):287–320, 2001.

[7] J. Neumann, C. Schnörr, and G. Steidl. Combined svm-based feature selection and classification. *Mach. Learn.*, 61(1-3):129–150, 2005.

[8] J. Weston, A. Elisseeff, B. Schölkopf, and M. Tipping. The use of

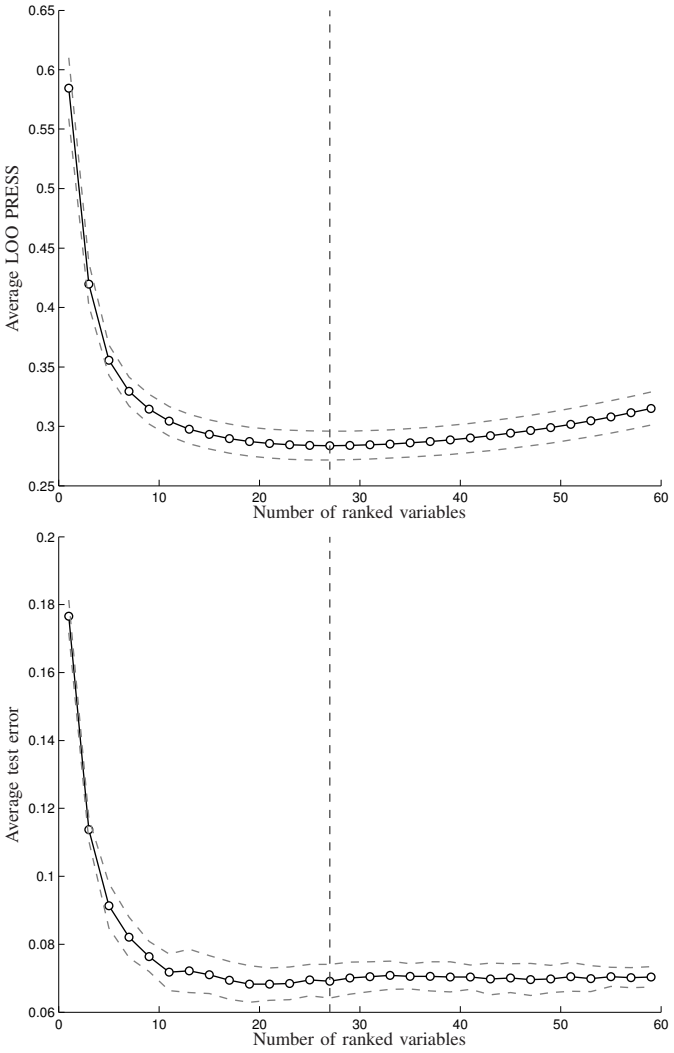

Fig. 3. Selecting the number of ranked variables in the Splice data set based on the averaged LOO PRESS. A number of 27 variables out of 60 attain the lowest LOO PRESS averaged over 100 partitions. The degree of the polynomial kernel is set to a value of 3, whit the hyperparameters (τ, γ) tuned as described in the section over model selection. Top: Average LOO PRESS over 100 realizations. Bottom: Average test error over 100 realizations. Dashed lines represent the standard error deviation. The vertical dashed line points the model that achieves the lowest LOO PRESS and its corresponding test error performance.

zero-norm with linear models and kernel methods. *Journal of Machine Learning Research.*, 2002.

[9] K. Pelckmans, I. Goethals, J. De Brabanter, J.A.K. Suykens, and B. De Moor. *Componentwise least squares support vector machines*, chapter in Support Vector Machines: Theory and Applications, pages 77–98. Springer, 2005.

[10] K. Pelckmans, J.A.K. Suykens, and B. De Moor. Building sparse representations and structure determination on LS-SVM substrates. *Neurocomputing*, (64):137–159, 2005.

[11] F. Ojeda, J.A.K. Suykens, and B. De Moor. Low rank updated LS-SVM classifiers for fast variable selection. *Neural Networks*, 21(2-3):437–449, 2008.

[12] T. Hastie and R. Tibshirani. *Generalized Additive Models*. Chapman and Hall, New York, 1990.

[13] V. Van Belle, K. Pelckmans J.A.K Suykens, and S. Van Huffel. Additive survival least squares support vector machines. *Statistics in Medicine*, (2):296–308, January 2010.

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

[15] B. Schölkopf, S. Mika, C. J. C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, and A. J. Smola. Input space versus feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 10:1000–1017, 1999.

[16] G. H. Golub and C. F. Van Loan. *Matrix computations*. The Jonhs Hopkins University Press, 2nd. edition, 1989.

[17] P.E. Gill, W. Murray, and M.H. Wright. *Practical Optimization*. Academic Press, 1981.

[18] J.J. Dongarra, J.R. Bunch, C.B. Moler, and G.W. Stewart. LINPACK Users' Guide. *SIAM, Philadelphia*, 1979.

[19] R. Kohavi and G. John. Wrappers for feature selection. *Artificial Intelligence*, 97:273–324, 1997.

[20] A. Miller. *Subset Selection in Regression*. Chapman and Hall, 2002.

[21] G. C. Cawley. Leave-one-out cross-validation based model selection criteria for weighted LS-SVMs. In *Proc of the International Joint Conference on Computational Neural Networks (IJCNN'06)*, 2006.

[22] G. C. Cawley and N. L. C. Talbot. Preventing over-fitting in model selection via bayesian regularisation of the hyper-parameters. *Journal of Machine Learning Research*, 8:841–861, 2007.

[23] A. Rakotomamonjy. Variable selection using SVM based criteria. *J. Mach. Learn. Res.*, 3:1357–1370, 2003.

[24] G. Claeskens, C. Croux, and J. Van Kerckhoven. An information criterion for variable selection in support vector machines. *J. Mach. Learn. Res.*, 9:541–558, 2008.