Primal Space Sparse Kernel Partial Least Squares Regression for Large Scale Problems

Special Session paper "Least Squares Support Vector Machines"

L. Hoegaerts, J.A.K. Suykens, J. Vandewalle, B. De Moor
Katholieke Universiteit Leuven
Department of Electrical Engineering, ESAT-SCD-SISTA
Kasteelpark Arenberg 10, B-3001 Leuven (Heverlee), Belgium
Phone: +32/16/32 85 40 - Fax: +32/16/32 19 70
Email: {luc.hoegaerts,johan.suykens}@esat.kuleuven.ac.be

***Abstract*— Kernel based methods suffer from exceeding time and memory requirements when applied on large datasets since the involved optimization problems typically scale polynomially in the number of data samples. As a remedy we propose both working on a reduced set (for fast evaluation) and at the same time keeping the number of model parameters small (for fast training). Departing from the Nyström based feature approximation we describe fixed-size LS-SVM in the context of primal space least squares regression, to extend it with a supervised counterpart, sparse kernel Partial Least Squares. The model is illustrated on a large scale example.**

I. Introduction

Over the last years one can see many learning algorithms being transferred to a kernel representation [1], [2]. The benefit lies in the fact that nonlinearity can be allowed, while avoiding to solve a nonlinear optimization problem. In this paper we focus on least squares regression models in the kernel context. By means of a nonlinear map into a Reproducing Kernel Hilbert Space (RKHS) [3] the data are projected to a high-dimensional space.

Kernel methods typically operate in this RKHS. The high-dimensionality which could pose problems for proper parameter estimation is circumvented by the kernel trick, which brings the dimensionality to the number of training instances n and at the same time allows excellent performance in classification and regression tasks. Yet, for large datasets this dimensionality in n means a serious bottleneck, since the corresponding training methods scale polynomially in n. Downsizing the system in dimensions to size $m \ll n$ is therefore needed. On the one hand, low-rank approximations (e.g. [4], [5]) offer reduction to smaller $m \times m$ matrices. On the other hand reduced set methods work with a thin tall $n \times m$ system (e.g. [6], [7], [8], [3], [9]). This work connects at the latter perspective.

We deal with least squares (LS) regression in the feature space. The estimation of the regression coefficients in the RKHS should be performed in a (potentially) high-dimensional basis. Most kernel LS based methods circumvent this by the kernel trick or a functional formulation. Here, we propose to explicitly work with an approximation of the feature map, via the Nyström method [10]. This we show to result in a model formulation of fixed-size LS-SVMs [2], which combines regression in the primal space with a small number of regression parameters and exhibits a sparse representation (unlike estimation in the dual space as done in Gaussian Processes [4], [11]).

Closer inspection of the FS-LSSVM model leads us to propose a supervised counterpart to yield approximate features. Therefore we consider kernel Partial Least Squares (KPLS), which fits naturally in a primal-dual optimization class of kernel machines [2]. By employing then the KPLS eigenbasis based on a fixed subset of features we are able to effectively accommodate this method with a sparse representation in a natural and efficient manner (different from [12], where sparsification is induced via a multi-step adaptation of the algorithm with extra computational burdens). In doing so we open at the same time the possibility for KPLS be applicable to large scale data sets.

This paper is organized as follows. In Section II we present some minimal background on kernel methods in relation to reproducing kernel Hilbert spaces. In Section III we start from the Nyström approximation for features to arrive at FS-LSSVM with sparse kernel representation and limited number of regression parameters. In Section IV we deal with KPLS in its primal-dual optimization formulation and formulate its sparse kernel version. In Section V we illustrate the sparse KPLS on a large data set application.

II. Reproducing Kernel Hilbert Space

The central idea for kernel algorithms within the learning theory context is to change the representation of a data point into a higher-dimensional mapping in a reproducing kernel Hilbert space (RKHS) H_k by means of a kernel function. When appropriately chosen, the kernel function with arguments in the original space corresponds to a dot product with arguments in the RKHS. This allows to circumvent working explicitly with the new representation as long as one can express the computation in the RKHS as inner products.

Assume data $\{\{(\mathbf{x}_i, y_i)\}_{i=1}^n \in \mathbb{R}^p \times \mathbb{R}\}$ have been given. A kernel k provides a similarity measure between pairs of data points

$$k : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R} : (\mathbf{x}_i, \mathbf{x}_j) \mapsto k(\mathbf{x}_i, \mathbf{x}_j). \tag{1}$$

Once a kernel is chosen, one can associate to each $\mathbf{x} \in \mathbb{R}^p$ a mapping $\varphi : \mathbb{R}^p \to H_k : \mathbf{x} \mapsto k(\mathbf{x}, \cdot)$, which can be

evaluated at $\mathbf{x}'$ to give $\varphi(\mathbf{x})(\mathbf{x}') = k(\mathbf{x}, \mathbf{x}')$. The set of these mappings can be extended by including all possible finite linear combinations, adjoining the limits and constructing an inner product

$$\varphi(\mathbf{x})(\mathbf{x}') = \langle k(\mathbf{x}, \cdot), k(\mathbf{x}', \cdot) \rangle = k(\mathbf{x}, \mathbf{x}') \tag{2}$$

based on the chosen kernel. One obtains a RKHS on X under the condition that the kernel is positive definite [13]. Remark that the kernel function coincides with an inner product function and that it reproduces itself. It is also called a representer of f at $\mathbf{x}$ because $f(\mathbf{x}) = \langle f, k(\mathbf{x}, \cdot) \rangle$.

The resulting RKHS has then the property that every evaluation operator and norm of any element in H_k is bounded [3]. This makes the elements of a RKHS well-suited to interpolate pointwise known functions that must be smooth. In the context of regularization one tries to do this by minimizing a pointwise cost function $c(\cdot)$ over data and monotonic smoothness function $g(\cdot)$:

$$\min_{f \in H_k} c(\mathbf{x}_i, y_i, f(\mathbf{x}_i)) + g(\|f\|). \tag{3}$$

The representer theorem [14] states that the solution is constrained to the subspace spanned by the mapped data points:

$$f(\mathbf{x}) = \sum_{i=1}^{n} w_i \varphi(\mathbf{x}_i)(\mathbf{x}) = \sum_{i=1}^{n} w_i k(\mathbf{x}_i, \mathbf{x}). \tag{4}$$

Thus the solution can always be expanded as a linear function of dot products. The w_i coefficients are typically found by solving the optimization problem that involves the specific regularization functional.

The Mercer-Hilbert-Schmidt theorem reveals more about the nature of φ by stating that for each positive definite kernel there exists an orthonormal set $\{\phi_i\}_{i=1}^{d}$ with non-negative λ_i such that we have the following spectral decomposition:

$$k(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{d} \lambda_i \phi_i(\mathbf{x}) \phi_i(\mathbf{x}') = \langle \varphi(\mathbf{x}), \varphi(\mathbf{x}') \rangle, \tag{5}$$

where $d \leq \infty$ is the dimension of of the RKHS, and λ_i and ϕ_i are the eigenvalues and eigenvectors of the kernel operator, defined by the integral equation

$$\int k(\mathbf{x}, \mathbf{x}') \phi_i(\mathbf{x}) p(\mathbf{x}) \mathrm{dx} = \lambda_i \phi_i(\mathbf{x}'), \tag{6}$$

on $L_2(\mathbb{R}^p)$. This allows to formulate the inner product in terms of expansion coefficients:

$$\langle f, g \rangle = \langle \sum_{i=1}^{d} a_i \phi_i, \sum_{j=1}^{d} b_j \phi_j \rangle = \sum_{i=1}^{d} \frac{a_i b_i}{\lambda_i}, \tag{7}$$

where $\langle \phi_i, \phi_j \rangle = \delta_{ij} / \lambda_i$. A proper scaling of the basis vectors ϕ_i with factor $\sqrt{\lambda_i}$ will transform the inner product to its most simple canonical form of an Euclidean dot product so that $k(\mathbf{x}, \mathbf{x}') = \varphi(\mathbf{x})^T \varphi(\mathbf{x}')$. Furthermore one can express each feature $\varphi(\mathbf{x})$ in this basis so that the φ mapping can be identified with a $d \times 1$ *feature* vector

$$\varphi(\mathbf{x}) = [\sqrt{\lambda_1}\phi_1(\mathbf{x}) \quad \sqrt{\lambda_2}\phi_2(\mathbf{x}) \quad ... \quad \sqrt{\lambda_d}\phi_d(\mathbf{x})]^T. \tag{8}$$

For regression purposes, just like one builds up a $n \times p$ data matrix X from the $\mathbf{x}_i$ inputs, one constructs with the mapped data points $\varphi(\mathbf{x})$, an $n \times d$ *feature matrix* :

$$\Phi = \begin{pmatrix} \sqrt{\lambda_1}\phi_1(\mathbf{x}_1) & \sqrt{\lambda_2}\phi_2(\mathbf{x}_1) & \dots & \sqrt{\lambda_n}\phi_d(\mathbf{x}_1) \\ \sqrt{\lambda_1}\phi_1(\mathbf{x}_2) & \sqrt{\lambda_2}\phi_2(\mathbf{x}_2) & \dots & \sqrt{\lambda_n}\phi_d(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \sqrt{\lambda_1}\phi_1(\mathbf{x}_n) & \sqrt{\lambda_2}\phi_2(\mathbf{x}_n) & \dots & \sqrt{\lambda_n}\phi_d(\mathbf{x}_n) \end{pmatrix}. \tag{9}$$

It is a difficulty that in general the elements of this feature matrix Φ are unknown because the explicit expression for φ is not available. The common approach to deal with this issue is then to make use of the so-called kernel trick [1]. The need for a direct expression is typically avoided by an ad hoc substitution such that scalar products are formed and consequently kernels come into play. Thus, the element $\varphi(\mathbf{x})$ may be not explicitly known, but the projection on any other $\varphi(\mathbf{x}')$ is simply $k(\mathbf{x}, \mathbf{x}')$, and thus the unknown elements are eliminated.

Here we follow another approach in which we start from an approximate explicit expression for $\varphi(\mathbf{x})$, via the Nyström method. We will show how it leads to a least squares regression model, FS-LSSVM, with a downsized number of parameters and a sparse kernel expansion. Such parsimonious models are highly attractive when dealing with large data sets.

III. The Nyström approximation

The Nyström method dates back from the late 1920's [10]. It offers an approximate solution to integral equations [15], [16]. It became recently of interest in the kernel machine learning community via Gaussian processes [4], [11].

A. Approximation of eigenfunctions

One can discretize the integral of eq. (6) on a finite set of evaluation points $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$, with simple equal weighting, yielding a system of equations:

$$\frac{1}{n} \sum_{j=1}^{n} k(\mathbf{x}, \mathbf{x}_j) \phi_i(\mathbf{x}_j) = \lambda_i^{(n)} \phi_i^{(n)}(\mathbf{x}), \tag{10}$$

which can be structured as a matrix eigenvalue problem:

$$K U_n = U_n \Lambda_n, \tag{11}$$

where $K_{ij} = \langle \varphi(\mathbf{x}_i), \varphi(\mathbf{x}_j) \rangle = k(\mathbf{x}_i, \mathbf{x}_j)$ are the elements of the kernel Gram matrix, U_n a $n \times n$ matrix of eigenvectors of K and Λ_n a $n \times n$ diagonal matrix of nonnegative eigenvalues in non-increasing order. (We emphasized the dimensions of matrices by a subscript and for vectors it is indicated in the superscript, between brackets to avoid confusion with powers). Expression (10) delivers direct approximations of the eigenvalues and eigenfunctions for the $\{\mathbf{x}_j\}_{j=1}^{n}$ points:

$$\phi_i(\mathbf{x}_j) \approx \sqrt{n}\ \mathbf{u}_{ji}^{(n)}, \tag{12}$$

$$\lambda_i \approx \frac{1}{n} \lambda_i^{(n)}. \tag{13}$$

It was a key observation of Nyström to backsubstitute these approximations in (10) to obtain an approximate of an eigenfunction evaluation in new points $\mathbf{x}'$:

$$\phi_i(\mathbf{x}') = \frac{\sqrt{n}}{\lambda_i^{(n)}} \sum_{j=1}^{n} k(\mathbf{x}', \mathbf{x}_j)\mathbf{u}_{ji}^{(n)} = \frac{\sqrt{n}}{\lambda_i^{(n)}} \left(\mathbf{k}^{(n)}(\mathbf{x}')\right)^T \mathbf{u}_i^{(n)}, \tag{14}$$

where we denote

$$\left(\mathbf{k}^{(n)}(\mathbf{x})\right)^T = (\Phi\varphi(\mathbf{x}))^T = [k(\mathbf{x}_1, \mathbf{x}), k(\mathbf{x}_2, \mathbf{x}), \ldots, k(\mathbf{x}_n, \mathbf{x})]. \tag{15}$$

As can be seen from (8), the entries of features in a RKHS match the eigenfunctions, apart from a scaling factor, so an approximate eigenfunction yields us an approximate feature.

B. Feature approximation based on the complete training data set

Maximally n approximate eigenfunctions ϕ_i can be computed, since we have only information in the given data points. The Nyström approximation allows to obtain an explicit expression for the entries of the approximated feature vector:

$$\varphi_i(\mathbf{x}) = \sqrt{\lambda_i}\ \phi_i(\mathbf{x}) \tag{16}$$

$$= \left(\lambda_i^{(n)}\right)^{-1/2} \sum_{j=1}^{n} k(\mathbf{x}, \mathbf{x}_j)\mathbf{u}_{ji}^{(n)} \tag{17}$$

$$= \left(\lambda_i^{(n)}\right)^{-1/2} \left(\mathbf{k}^{(n)}(\mathbf{x})\right)^T \mathbf{u}_i^{(n)}. \tag{18}$$

The approximation to a feature vector becomes a $n \times 1$ vector:

$$\varphi(\mathbf{x}) \approx \Lambda_n^{-1/2} U_n^T \mathbf{k}^{(n)}(\mathbf{x}). \tag{19}$$

In [4], [11] this approximation was recognized as implicitly present in the projection of features in the eigenspace produced by Kernel principal component analysis (KPCA), introduced in [17].

At the same time we obtain also an $n \times n$ matrix approximation Φ_n of the $n \times d$ *regressor matrix* (9) :

$$\Phi = \begin{pmatrix} \varphi^T(\mathbf{x}_1) \\ \varphi^T(\mathbf{x}_2) \\ \vdots \\ \varphi^T(\mathbf{x}_n) \end{pmatrix} \approx K U_n \Lambda_n^{-1/2} =: \Phi_n. \tag{20}$$

If we use this matrix in a least squares setting, we obtain a linear model in feature space. More specifically we are then performing PCR, which has the advantage of reducing the number of parameters if enough components are left out, but a sparse kernel expansion is not obtained. In next section we show that this extra requirement can be added.

C. Feature approximation based on a training data subset

In order to introduce parsimony, this technique can be employed to yield a simple approximation of the features for a predetermined subset of $m \ll n$ training points. These are then used on their turn as an interpolative, but quite accurate, formula for the features in the remaining training points. A likewise m-approximation can be made:

$$\lambda_i \approx \frac{1}{m}\lambda_i^{(m)}, \tag{21}$$

$$\phi_i(\mathbf{x}) \approx \frac{\sqrt{m}}{\lambda_i^{(m)}} \sum_{j=1}^{m} k(\mathbf{x}, \mathbf{x}_j)\ u_{ji} = \frac{\sqrt{m}}{\lambda_i^{(m)}} \left(\mathbf{k}^{(m)}(\mathbf{x})\right)^T \mathbf{u}_i. \tag{22}$$

Again one can obtain an explicit expression for the entries of the approximated feature vector:

$$\varphi_i(\mathbf{x}) = \sqrt{\lambda_i}\ \phi_i(\mathbf{x}) \tag{23}$$

$$\approx \left(\lambda_i^{(m)}\right)^{-1/2} \sum_{j=1}^{m} k(\mathbf{x}_j, \mathbf{x})\ u_{ji}. \tag{24}$$

The approximation to the feature vector now becomes a $m \times 1$ vector:

$$\varphi(\mathbf{x}) \approx \Lambda_m^{-1/2} U_m^T \mathbf{k}^{(m)}(\mathbf{x}). \tag{25}$$

It yields the following approximation of the feature matrix

$$\Phi_n = K_{nm} U_m \Lambda_m^{-1/2}. \tag{26}$$

If this approximation is directly used in a LS model we obtain

$$f(\mathbf{x}) = \left(\mathbf{w}^{(s)}\right)^T \varphi(\mathbf{x}) + b \tag{27}$$

$$= \sum_{j=1}^{s} w_j \left(\sum_{i=1}^{m} \left(\lambda_j^{(m)}\right)^{-1/2} u_{ij}^{(m)} k(\mathbf{x}_i, \mathbf{x})\right) + b \tag{28}$$

$$= \sum_{i=1}^{m} \alpha_i\ k(\mathbf{x}_i, \mathbf{x}) + b \tag{29}$$

where s is the number of retained principal components and

$$\alpha_i = \sum_{j=1}^{s} w_j \left(\lambda_j^{(m)}\right)^{-1/2} u_{ij}^{(m)}. \tag{30}$$

This model (in conjunction with an entropy-based selection procedure) has been named Fixed-Size LS-SVM and proposed in coauthors' work [2]. It delivers a *sparse kernel expansion* and at the same time the number of parameters of $\mathbf{w}$ is kept smaller or equal to m. A model that combines these two benefits has distinct advantage, because it allows a considerable computational speedup during evaluation in new inputs and training.

In [18] the preceding two least squares models are described in a general framework of subspace regression. Formally the restriction of the LS regression coefficients can be seen as confining the regression to a subspace, and the size of the kernel expansion is a matter of choosing a reduced set of basis vectors in which the subspace is to be expressed. It can formally be shown that a lot of well-known least squares kernel methods can be described whithin this framework.

The subspace in case of FS-LS-SVM is the PCA basis of the m features, the eigenspace of the variance matrix of Φ_m. With this particular choice of subspace one obtains new, transformed regressors, that are subsequently used for a

regression. The idea we apply now is to include the case where also information of the dependent variable is involved in the construction of important regressors. The immediate and most wellknown candidate for this goal is Partial Least Squares, which we will treat in next section.

IV. Kernel Partial Least Squares with sparse kernel expansion

In this section we first give a short summary of Partial Least Squares. We also show that next to the supervised aspect, this method in fact belongs to a general class of kernel machines with a primal-dual optimization formulation [2]. As such KPLS receives a proper foundation and we can then employ it on a reduced set to extend KPLS with a sparse kernel expansion.

A. Partial Least Squares

Partial Least Squares (PLS) [19] is a multivariate technique that delivers an optimal basis in x-space for y onto x regression. Reduction to a certain subset of the basis introduces a bias, but reduces the variance.

In general, PLS is based on a maximization of the covariance between successive linear combinations in x and y space, $\langle \mathbf{v}, \mathbf{x} \rangle$ and $\langle \mathbf{w}, \mathbf{y} \rangle$, where coefficient vectors are normed to unity and constrained to be orthogonal in x space:

$$\max_{\mathbf{v},\mathbf{w}} \text{cov } (\mathbf{v^T x}, \mathbf{w^T y}) = \mathbf{v^T} C_{xy} \mathbf{w} \tag{31}$$

subject to $\|\mathbf{v}\| = 1 = \|\mathbf{w}\|$ and $V^T V = I_p$, with $C_{xy} = X^T Y$ the sample covariance matrix. Solutions can be obtained by using Lagrange multipliers, which leads to solving the following system

$$C_{xy}\mathbf{w} = \lambda \mathbf{v} \tag{32}$$

$$C_{yx}\mathbf{v} = \lambda \mathbf{w}. \tag{33}$$

As a least squares cost function, PLS turns out to be a sum of the least squares formulation of each of the above methods:

$$J(\mathbf{v}, \mathbf{w}) = \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{v}\mathbf{v}^T \mathbf{x}_i\|^2 + \|\mathbf{v}^T \mathbf{x}_i - \mathbf{w}^T \mathbf{y}_i\|^2 + \|\mathbf{y}_i - \mathbf{w}\mathbf{w}^T \mathbf{y}_i\|^2 \tag{34}$$

subject to the constraints.

B. KPLS in primal-dual optimization context

Again we explicitly avoid an ad hoc introduction of kernels through substitution but we follow instead a more principled approach starting from the PLS least squares formulation (34). This primal cost criterion aims at optimizing the coefficient vectors $\mathbf{v}$ and $\mathbf{w}$, searching simultaneously for the maximal projection of a data point in x space, maximal covariation with the corresponding projection of the point in y space, and maximal projection of a data point in y space. Since the coefficients could become arbitrarily large, these are typically constrained or at least regularized.

By simplifying expression (34) we obtain $\langle \mathbf{v}^T \mathbf{x}, \mathbf{w}^T \mathbf{y} \rangle$ and by adding (soft) regularization, we arrive at the following primal form problem:

$$\max_{\mathbf{v},\mathbf{w}} J_{PLS}(\mathbf{v}, \mathbf{w}, e, r) = \gamma \sum_{i=1}^{n} e_i r_i - \frac{1}{2}\mathbf{v}^T \mathbf{v} - \frac{1}{2}\mathbf{w}^T \mathbf{w} \tag{35}$$

such that $e_i = \mathbf{v}^T \varphi(\mathbf{x}_i)$ and $r_i = \mathbf{w}^T \varphi(\mathbf{y}_i)$ for $i = 1, \ldots, n$ with hyperparameter $\gamma \in \mathbb{R}_0^+$. Introducing a_i, b_i as Lagrange multiplier parameters, the Lagrangian is written as

$$L(\mathbf{v}, \mathbf{w}, e, r; \mathbf{a}, \mathbf{b}) = \gamma \sum_{i=1}^{n} e_i r_i - \frac{1}{2}\mathbf{v}^T \mathbf{v} - \frac{1}{2}\mathbf{w}^T \mathbf{w} - \sum_{i=1}^{n} a_i (e_i - \mathbf{v}^T \varphi(\mathbf{x}_i)) - \sum_{i=1}^{n} b_i (r_i - \mathbf{w}^T \varphi(\mathbf{y}_i)). \tag{36}$$

A given optimization problem has a corresponding dual formulation. The number of constraints in the original problem becomes the number of variables in the dual problem. One optimizes the Lagrangian subject to the following optimality conditions:

$$\frac{\partial L}{\partial \mathbf{v}} = 0 \rightarrow \mathbf{v} = \sum_{i=1}^{n} a_i \varphi(\mathbf{x_i}) = \Phi^T \mathbf{a} \tag{37}$$

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \rightarrow \mathbf{w} = \sum_{i=1}^{n} b_i \varphi(\mathbf{y}_i) = \Phi^T \mathbf{b} \tag{38}$$

$$\frac{\partial L}{\partial e_i} = 0 \rightarrow \gamma r_i = a_i \quad i = 1, \ldots, n \tag{39}$$

$$\frac{\partial L}{\partial r_i} = 0 \rightarrow \gamma e_i = b_i \quad i = 1, \ldots, n \tag{40}$$

$$\frac{\partial L}{\partial a_i} = 0 \rightarrow e_i = \mathbf{v}^T \varphi(\mathbf{x}_i) \quad i = 1, \ldots, n \tag{41}$$

$$\frac{\partial L}{\partial b_i} = 0 \rightarrow r_i = \mathbf{w}^T \varphi(\mathbf{y}_i) \quad i = 1, \ldots, n. \tag{42}$$

By elimination of the variables $\mathbf{e}, \mathbf{r}, \mathbf{v}, \mathbf{w}$ and defining $\lambda = 1/\gamma$, we can simplify the dual problem further, which results into the very system

$$\begin{aligned} K_{yy}\mathbf{b} &= \lambda \mathbf{a} \\ K_{xx}\mathbf{a} &= \lambda \mathbf{b} \end{aligned} \tag{43}$$

where $[K_{xx}]_{ij} = \varphi(\mathbf{x}_i)^T \varphi(\mathbf{x}_j)$ and $[K_{yy}]_{ij} = \varphi(\mathbf{y}_i)^T \varphi(\mathbf{y}_j)$ are the kernel Gram matrices. We point out that numerous variants have been formulated to the basic PLS problem [20], [21]. Even this KPLS solution can be considered in the optimization context as a special case of the 'regularized' KCCA variant which was proposed within the context of primal-dual LS-SVM formulations in [2]

$$\begin{aligned} K_{yy}\ \mathbf{b} &= \lambda\ (\nu_1 K_{xx} + I)\ \mathbf{a} \\ K_{xx}\ \mathbf{a} &= \lambda\ (\nu_2 K_{yy} + I)\ \mathbf{b}, \end{aligned} \tag{44}$$

If one chooses the parameters $\nu_1 = \nu_2 = 0$. The 'regular' KCCA was originally reported by [22] and [23], in an independent component analysis (ICA) context, without proper regularization.

C. Sparse KPLS regression

Remark it is exactly eq. (37) that explicitly shows that the KPLS eigenbasis is expressed in the basis spanned by the set of features and bears naturally the connection with the subspace regression framework of [18]. In fact: any set of vectors that one can express in the basis of the features could be used as a subspace and thus many more methods next to KPLS could be subjected to sparsification.

As such it only remains to choose a reduced set of feature vectors to induce the sparse kernel expansion. This gives analogously rise to an expression like eq. (27), but with the KPLS based eigenvectors instead:

$$f(\mathbf{x}) = \left(\mathbf{w}^{(s)}\right)^T \varphi(\mathbf{x}) + b \tag{45}$$

$$= \sum_{j=1}^{s} w_j \left(\sum_{i=1}^{m} a_{ij}^{(m)} k(\mathbf{x}_i, \mathbf{x}) \right) + b \tag{46}$$

$$= \sum_{i=1}^{m} \alpha_i \; k(\mathbf{x}_i, \mathbf{x}) + b, \tag{47}$$

where s is the number of retained principal components and

$$\alpha_i = \sum_{j=1}^{s} w_j a_{ij}^{(m)}. \tag{48}$$

In Figure (1) we show an illustration of the first three approximated features with sparse KPLS on a simple sinc data example. Figure (2) shows on the same example that often with a very small subset very good approximations can be made, with performance close to the one of methods that make use of information on the full training set.

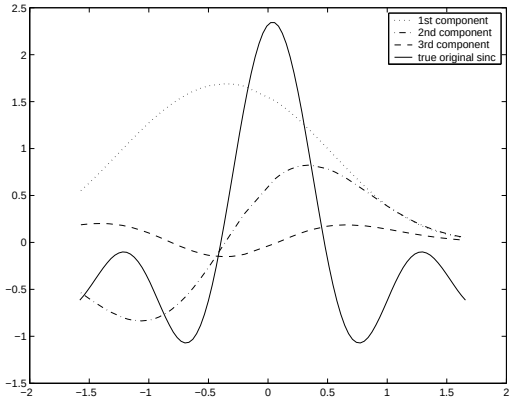


Fig. 1. Visualisation of the features of the LS PLS-SVD subspace regression model on a sinc artificial data set sample. The subset consists of 5 points. It are linear combinations of these clearly nonlinear features that will fit the original curve.

V. Large Scale Example

With a large scale data set example we demonstrate that the sparse kernel framework indeed can manage where other methods fail. From the computational side our approach achieves overall a much smaller $O(nm)$ memory cost, compared to the typical $O(n^2)$ and a computational complexity of $O(nm^3)$ compared to the typical $O(n^2)$.

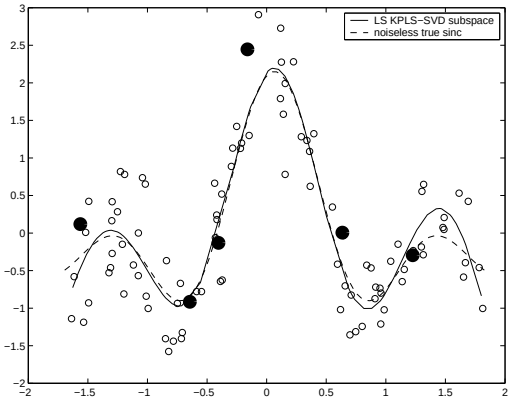


Fig. 2. Visualization of the LS sparse KPLS regression model, modified naturally to a sparse representation, on a sinc artificial data set sample with added noise (dots). The subset consists of 5 points, marked with a full dot on the figure.

The ADULT UCI data set [24] consists of 45222 cases having 14 input variables. The aim is to classify if the income of a person is greater than 50K based on several census parameters, such as age, education, marital status, etc. We standardized the data to zero mean and unit variance. We picked at random a training set of size $n = 33000$ and a test set of size $t = 12222$. We used the common Gaussian kernel

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{h^2}\right). \tag{49}$$

with width parameter $h^2 = 29.97$ derived from preliminary experiments. We performed 10-fold cross-validation to estimate the optimal number of components. After cross-validation the best model was evaluated on the independent test set. The randomization trials were repeated 20 times, and this for subset sizes over different values in the range $[25, 1000]$. Training and evaluation time per model is typically of the order of minutes for a modest number of components.

In Figure (3) we show the averaged misclassification rate (in percent) versus subset size m on the test set. For reference we included the solutions of a state-of-the-art regression solver, the standard LS-SVM for regression, using the LS-SVMlab software from http://www.esat.kuleuven.ac.be/sista/lssvmlab/. The LS-SVM was trained and cross-validated each time only on the subset data of size m, because it is not possible to take into account the information of the entire training set. Qualitatively both the FS-LSSVM and sparse KPLS outperform LS-SVM (on subset) with a considerable amount of at least 4%. A Wilcoxon rank sum test [25] at a significance level of $\alpha = 0.01$ shows no difference of the mean values between the two sparse methods relative to the standard deviations. Other tests on smaller benchmark datasets all confirm that KPLS equals FS-LSSVM in performance.

It is remarkable that already with a subset size of 75 a favorable classification is obtained. The lowest rate was reached at a subset size of $m = 400$, after which we found no

further improvements. compared with results from literature, a correct test set classification rate of $84.7(\pm 0.3)\%$ ranks among the best of results delivered by other current state-of-the-art classifiers [2]. The use of other kernels, like the polynomial or the sigmoidal kernel, did not produce such good results as the Gaussian kernel.

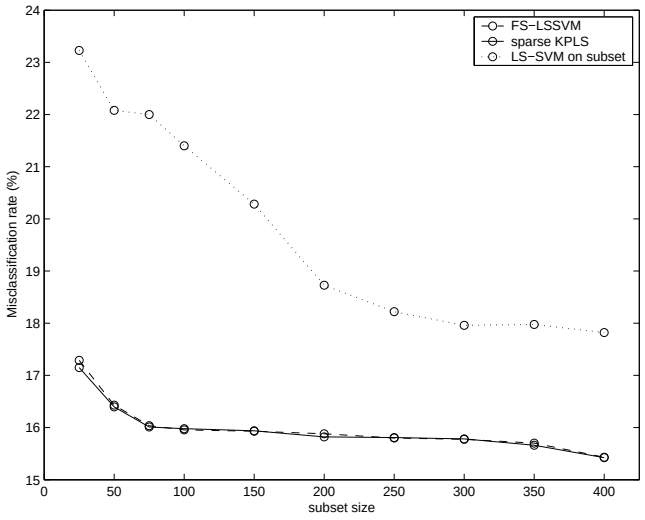


Fig. 3. Averaged misclassification rate (in %) on the independent test set versus subset size m for FS-LSSVM, sparse KPLS and LS-SVM on the subset. Again very few basis vectors are necessary to reach favorable performance at a low computational cost.

VI. Conclusions

In large scale applications a model must be able to process a large number of datapoints. Most kernel methods are simply not adequate for this task. In this paper we discussed a (Nyström based) Fixed-Size variant of the LS-SVM, which operates in primal space and has two important advantages: a small number of regression coefficients (which allows a fast training) and a sparse kernel expansion (which allows fast evaluation). By considering the regression model as restricted in the parameters and because kernel Partial Least Squares Kernel delivers components as linear combinations of features we were able to formulate a version with a sparse kernel representation. In an example we show that the approximation of the features with sparse KPLS equals the ones obtained through FS-LSSVM and that this sparse KPLS variant is effectively capable of dealing with a large dataset.

Acknowledgment

This research work was carried out at the ESAT laboratory of the Katholieke Universiteit Leuven. It is supported by grants from several funding agencies and sources: Research Council KU Leuven: Concerted Research Action GOA-Mefisto 666 (Mathematical Engineering), IDO (IOTA Oncology, Genetic networks), PhD/postdoc & fellow grants; Flemish Government: Fund for Scientific Research Flanders (PhD/postdoc grants, projects G.0407.02 (support vector machines), G.0256.97 (subspace), G.0115.01 (bio-i & microarrays), G.0240.99 (multilinear algebra), G.0197.02 (power islands), research communities ICCoS, ANMMM), AWI (Bil. Int. Collaboration Hungary/ Poland), IWT (Soft4s (softsensors), STWW-Genprom (gene promotor prediction), GBOU-McKnow (Knowledge management algorithms), Eureka-Impact (MPC-control), Eureka-FLiTE (flutter modeling), PhD grants); Belgian Federal Government: DWTC (IUAP IV-02 (1996-2001) and IUAP V-10-29 (2002-2006) (2002-2006): Dynamical Systems & Control: Computation, Identification & Modelling), Program Sustainable Development PODO-II (CP/40: Sustainibility effects of Traffic Management Systems); Direct contract research: Verhaert, Electrabel, Elia, Data4s, IPCOS. LH is a PhD student supported by the Flemish Institute for the Promotion of Scientific and Technological Research in the Industry (IWT). JS is an associate professor at KU Leuven. JVDW and BDM are full professors at KU Leuven, Belgium.

References

[1] B. Schölkopf and A. Smola, *Learning with kernels*. MIT Press, 2002.
[2] J. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle, *Least Squares Support Vector Machines*. World Scientific, Singapore, 2002.
[3] G. Wahba, *Spline Models for Observational Data*, ser. CBMS-NSF Regional Conference Series in Applied Mathematics. Philadelphia, PA: SIAM, 1990, vol. 59.
[4] C. Williams, C. Rasmussen, A. Schwaighofer, and V. Tresp, "Observations on the Nyström method for Gaussian processes," Institute for Adaptive and Neural Computation, Division of Informatics, University of Edinburgh, Tech. Rep., 2002.
[5] S. Fine and K. Scheinberg, "Efficient SVM training using low-rank kernel representations," *Journal of Machine Learning Research*, vol. 2, no. 2, pp. 243–264, 2002.
[6] K.-M. Lin and C.-J. Lin, "A study on reduced support vector machines," *Neural Networks, IEEE Transactions on*, vol. 14, no. 6, pp. 1449 – 1459, Nov 2003.
[7] Y. Lee and O. L. Mangasarian, "RSVM: Reduced support vector machines," First SIAM International Conference on Data Mining, Chicago, April 2001, technical Report 00-07, July, 2000.
[8] A. J. Smola and B. Schölkopf, "Sparse greedy matrix approximation for machine learning," in *Proc. 17th International Conf. on Machine Learning*. Morgan Kaufmann, San Francisco, CA, 2000, pp. 911–918.
[9] R. Rifkin, G. Yeo, and T. Poggio, *Advances in Learning Theory: Methods, Models and Applications, Eds. Suykens, Horvath, Basu, Micchelli, and Vandewalle*, ser. NATO Science Series III: Computer and Systems Sciences. Amsterdam: IOS Press, 2003, vol. 190.
[10] E. J. Nyström, "Uber die praktische auflosung von linearen integralgleichungen mit anwendungen auf randwertaufgaben der potentialtheorie," *Commentationes PhysicoMathematica*, vol. 4, no. 15, pp. 1–52, 1928.
[11] C. Williams and M. Seeger, "Using the Nyström method to speed up kernel machines," in *Advances in Neural Information Processing Systems, eds. T. K. Leen, T. G. Diettrich, V. Tresp*, vol. 13. MIT Press, 2001.
[12] M. Momma and K. Bennett, "Sparse kernel partial least squares regression," in *Proceedings of Conference on Learning Theory (COLT 2003)*, 2003, pp. 216–230.
[13] N. Aronszajn, "Theory of reproducing kernels," *Transactions of the American mathematical society*, vol. 686, pp. 337–404, 1950.
[14] G. Kimeldorf and G. Wahba, "Tchebycheffian spline functions," *J. Math. Ana. Applic.*, vol. 33, pp. 82–95, 1971.
[15] C. Baker, *The numerical treatment of integral equations*. Oxford Clarendon Press, 1977.
[16] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes: The Art of Scientific Computing*, 1st ed. Cambridge (UK) and New York: Cambridge University Press, 1986.
[17] B. Schölkopf, A. Smola, and K. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10(5), pp. 1299–1319, 1998.
[18] L. Hoegaerts, J. Suykens, J. Vandewalle, and B. De Moor, "Subset based least squares subspace regression in rkhs," *Technical Report 03-171, ESAT-SCD-SISTA, K.U.Leuven (Leuven, Belgium), november 2003, (submitted)*, 2004.
[19] H. Wold, "Estimation of principal components and related models by iterative least squares," in *Multivariate Analysis, ed. P.R. Krishnaiah*. New York: Academic Press, 1966, pp. 391–420.
[20] R. Rosipal and L. J. Trejo, "Kernel partial least squares regression in reproducing kernel hilbert space," *Journal of Machine Learning Research*, vol. 2, pp. 97–123, Dec 2001.
[21] Bennett and M. J. Embrechts, *Advances in Learning Theory: Methods, Models and Applications, J.A.K. Suykens, G. Horvath, S. Basu, C. Micchelli, J. Vandewalle (Eds.)*. NATO Science Series III: Computer & Systems Sciences,IOS Press Amsterdam, 2003, vol. 190, ch. An Optimization Perspective on Partial Least Squares, pp. 227–250.
[22] P. L. Lai and C. Fyfe, "Kernel and nonlinear canonical correlation analysis," *International Journal of Neural Systems*, vol. 10, no. 5, pp. 365–377, 2000.
[23] F. Bach and M. Jordan, "Kernel independent component analysis," *Journal of Machine Learning Research*, vol. 3, pp. 1–48, 2002.
[24] C. Blake and C. Merz, "Uci repository of machine learning databases," 1998. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html
[25] F. Ramsey and D. Schafer, *The Statistical Sleuth: A Course in Methods of Data Analysis*. Belmont, CA: Wadsworth Publishing Company, 1997.