Very Sparse LSSVM Reductions for Large-Scale Data

Raghvendra Mall and Johan A. K. Suykens, *Senior Member, IEEE*

***Abstract*— Least squares support vector machines (LSSVMs) have been widely applied for classification and regression with comparable performance with SVMs. The LSSVM model lacks sparsity and is unable to handle large-scale data due to computational and memory constraints. A primal fixed-size LSSVM (PFS-LSSVM) introduce sparsity using Nyström approximation with a set of prototype vectors (PVs). The PFS-LSSVM model solves an overdetermined system of linear equations in the primal. However, this solution is not the sparsest. We investigate the sparsity-error tradeoff by introducing a second level of sparsity. This is done by means of L_0-norm-based reductions by iteratively sparsifying LSSVM and PFS-LSSVM models. The exact choice of the cardinality for the initial PV set is not important then as the final model is highly sparse. The proposed method overcomes the problem of memory constraints and high computational costs resulting in highly sparse reductions to LSSVM models. The approximations of the two models allow to scale the models to large-scale datasets. Experiments on real-world classification and regression data sets from the UCI repository illustrate that these approaches achieve sparse models without a significant tradeoff in errors.**

***Index Terms*— L_0-norm, least squares support vector machine (LSSVM) classification and regression, reduced models, sparsity.**

I. Introduction

LEAST squares support vector machines (LSSVMs) were introduced in [2] and have become a state-of-the-art learning technique for classification and regression. In the LSSVM formulation, instead of solving a quadratic programming problem with inequality constraints as in the standard SVM [3], one has equality constraints and the L_2-loss function. This leads to an optimization problem whose solution in the dual is obtained by solving a system of linear equations.

Manuscript received June 17, 2013; revised March 11, 2014 and June 25, 2014; accepted June 25, 2014. Date of publication August 11, 2014; date of current version April 15, 2015. This work was supported in part by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013)/ERC AdG ADATADRIVE-B under Grant 290923, in part by the Research Council Katholieke Universitet Leuven under Grant GOA/10/09 MaNet, Grant CoE PFV/10/002 (OPTEC), and Grant BIL12/11T, in part by the Flemish Government under FWO Projects G.0377.12 (structured systems) and Project G.088114N (tensor based data similarity), in part by the IWT Project SBO POM under Grant 100031, in part by iMinds Medical Information Technologies SBO 2014 and Belgian Federal Science Policy Office under Grant IUAP P7/19 (dynamical systems, control and optimization, 2012–2017).

The authors are with the Stadius Centre for Dynamical Systems, Signal Processing and Data Analytics, Department of Electrical Engineering, Katholieke Universitet Leuven, Leuven 3001, Belgium (e-mail: rmall@esat.kuleuven.be; johan.suykens@esat.kuleuven.be).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2014.2333879

A drawback of LSSVM models is the lack of sparsity as usually all the data points become support vectors (SVs) as shown in [1]. Several works in the literature address this problem of lack of sparsity in the LSSVM model. They can be categorized as follows.

1) *Reduction Methods*: Training the model on the data set, pruning SVs, and selecting the rest for retraining the model.
2) *Direct Methods*: Enforcing sparsity from the beginning.

Some works in the first category are [4]–[10]. In [5], the authors provide an approximate SVM solution under the assumption that the classification problem is separable in the feature space. In [6] and [7], the proposed algorithm approximates the weight vector such that the distance to the original weight vector is minimized. Downs *et al.* [8] eliminate the SVs that are linearly dependent on other SVs. In [9] and [10], the authors work on a reduced set for optimization by preselecting a subset of data as SVs without emphasizing much on the selection methodology. Suykens *et al.* [4] prune the SVs which are farthest from the decision boundary. This is done recursively until the performance degrades. Another work [11] in this direction suggests to select the SVs closer to the decision boundary. However, these techniques cannot guarantee a large reduction in the number of SVs.

In the second category, the number of SVs referred to as prototype vectors (PVs) are fixed in advance. One such approach is introduced in [1] and is referred to as fixed-size least squares support vector machines (FS-LSSVM). It provides a solution to the LSSVM problem in the primal space resulting in a parametric model and a sparse representation. The method uses an explicit expression for the feature map using the Nyström method [12], [13]. The Nyström method is related to finding a low rank approximation to the given kernel matrix by choosing M rows or columns from the large $N \times N$ kernel matrix. In [1], the authors proposed searching for M rows or columns by maximizing the quadratic Rènyi entropy criterion. It was shown in [14] that the cross-validation error of primal FS-LSSVM (PFS-LSSVM) decreases with respect to the number of selected PVs until it does not change anymore and is heavily dependent on the initial set of PVs selected by quadratic Rènyi entropy. This point of saturation can be achieved for $M \ll N$ but this is not the sparsest solution. A sparse conjugate direction pursuit approach was developed in [15] where they iteratively build up a conjugate set of vectors of increasing cardinality to approximately solve the overdetermined PFS-LSSVM linear system. The approach works most efficiently when few iterations suffice for a good

2162-237X © 2014 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

approximation. However, when few iterations do not suffice for approximating the solution the cardinality will be M.

In recent years, the L_0-norm has been receiving increasing attention. The L_0-norm is the number of nonzero elements of a vector. So when the L_0-norm of a vector is minimized it results into the sparsest model. But this problem is NP-hard. Therefore, several approximations to it are discussed in [16] and [17], and so on. In this paper, we modify the iterative sparsification procedure introduced in [18] and [19]. The major drawbacks of the methods described in [18] and [19] are that these approaches cannot scale to very large-scale data sets because of memory ($N \times N$ kernel matrix) and computational [$O(N^3)$ time] constraints. We reformulate the iterative sparsification procedure for LSSVM and PFS-LSSVM methods to produce highly sparse models. These models can efficiently handle very large-scale data. We discuss two different initialization methods for which in a next step the sparsification step is applied.

1) *Initialization by PFS-LSSVM:* Sparsification of the PFS-LSSVM method leads to a highly sparse parametric model namely sparsified primal FS-LSSVM (SPFS-LSSVM).
2) *Initialization by Subsampled Dual LSSVM:* The subsampled dual LSSVM (SD-LSSVM) is a fast initialization to the LSSVM model solved in the dual. Its sparsification results into a highly sparse nonparametric model namely sparsified subsampled dual LSSVM (SSD-LSSVM).

We compare the proposed methods with state-of-the-art techniques including C-SVC, ν-SVC from the LIBSVM [21] software, Keerthi's method [22], L_0-norm-based method proposed by Lopez [19], and the L_0-reduced PFS-LSSVM method (SV_L_0-norm PFS-LSSVM) [20] on several benchmark data sets from the UCI repository [23]. In the following, we mention some motivations to obtain a sparse solution.

1) Sparseness can be exploited for having more memory and computationally efficient techniques, for example, in matrix multiplications and inversions.
2) Sparseness is essential for practical purposes such as scaling the algorithm to very large-scale data sets. Sparse solutions means fewer SVs and less time required for out-of-sample extensions.
3) By introducing two levels of sparsity, we overcome the problem of selection of the smallest cardinality (M) for the PV set faced by the PFS-LSSVM method.
4) The two level of sparsity allows scaling to large-scale data sets while having very sparse models.

We also investigate the sparsity versus error tradeoff.

This paper is organized as follows. A brief description of PFS-LSSVM and SD-LSSVM is given in Section II. The L_0-norm-based reductions to PFS-LSSVM, that is, SPFS-LSSVM and SD-LSSVM, that is, SSD-LSSVM are discussed in Section III. In Section IV, the different algorithms are successfully demonstrated on real-life data sets. We discuss about the sparsity versus error tradeoff in Section V. Section VI states the conclusion of this paper.

II. Initializations

In this paper, we consider two initializations. One is based on solving the LSSVM problem in the primal (PFS-LSSVM). The other is a fast initialization method solving a subsampled LSSVMs problem in the dual (SD-LSSVM).

A. Primal FS-LSSVM

1) Least Squares Support Vector Machine: We provide a brief summary of the LSSVMs methodology for classification and regression.

Given a sample of N data points $\{x_i, y_i\}$, $i = 1, \ldots, N$, where $x_i \in \mathbb{R}^d$ and $y_i \in \{+1, -1\}$ for classification and $y_i \in \mathbb{R}$ for regression, the LSSVM primal problem is formulated as follows:

$$\min_{w,b,e} \mathcal{J}(w, e) = \frac{1}{2} w^\intercal w + \frac{\gamma}{2} \sum_{i=1}^{N} e_i^2 \quad \text{such that} \quad w^\intercal \phi(x_i) + b = y_i - e_i, \quad i = 1, \ldots, N \tag{1}$$

where $\phi : \mathbb{R}^d \to \mathbb{R}^{n_h}$ is a feature map to a high-dimensional feature space, where n_h is the dimension of the feature space (which can be infinite dimensional), $e_i \in \mathbb{R}$ are the errors and $w \in \mathbb{R}^{n_h}$, $b \in \mathbb{R}$.

Using the coefficients α_i for the Lagrange multipliers, the solution to (1) can be obtained by the Karush-Kuhn-Tucker (KKT) [24] conditions for optimality. The result is given by the following linear system in the dual variables α_i:

$$\left[\begin{array}{c|c} 0 & 1_N^\intercal \\ \hline 1_N & \Omega + \frac{1}{\gamma} I_N \end{array}\right] \left[\begin{array}{c} b \\ \hline \alpha \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline y \end{array}\right] \tag{2}$$

with $y = (y_1, y_2, \ldots, y_N)^\intercal$, $1_N = (1, \ldots, 1)^\intercal$, $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_N)^\intercal$ and $\Omega_{kl} = \phi(x_k)^\intercal \phi(x_l) = K(x_k, x_l)$, for $k, l = 1, \ldots, N$ with K a Mercer kernel function. From the KKT conditions we get that $w = \sum_{i=1}^{N} \alpha_i \phi(x_i)$ and $\alpha_i = \gamma e_i$. The second condition causes the LSSVM to be nonsparse as whenever e_i is nonzero then $\alpha_i \neq 0$. Generally, in real-world scenarios the $e_i \neq 0, i = 1, \ldots, N$ for most data points. This leads to lack of sparsity in the LSSVM model.

2) Nyström Approximation and Primal Estimation: For large data sets it is often advantageous to solve the problem in the primal where the dimension of the parameter vector $w \in \mathbb{R}^d$ is smaller compared with $\alpha \in \mathbb{R}^N$. However, one needs an explicit expression for ϕ or the approximation of the nonlinear mapping $\hat{\phi} : \mathbb{R}^d \to \mathbb{R}^M$ based on a sampled set of PVs from the whole data set. In [14], the authors provide a method to select this subsample of size $M \ll N$ by maximizing the quadratic Rènyi entropy.

Williams and Seeger [25] use the Nyström method to compute the approximated feature map $\hat{\phi} : \mathbb{R}^d \to \mathbb{R}^M, i = 1, \ldots, M$ for a training point, or for any new point x^*, with $\hat{\phi} = (\hat{\phi}_1, \ldots, \hat{\phi}_M)^\intercal$, is given by

$$\hat{\phi}_i(x^*) = \frac{1}{\sqrt{\lambda_i^s}} \sum_{j=1}^{M} (u_i)_j K(z_j, x^*) \tag{3}$$

where λ_i^s and u_i are the eigenvalues and the eigenvectors of the kernel matrix $\bar{\Omega} \in \mathbb{R}^{M \times M}$ with $\bar{\Omega}_{ij} = K(z_i, z_j)$,

where z_i and z_j belong to the subsampled set $\mathcal{S}_{\text{PV}}$ which is a subset of the whole data set $\mathcal{D}$. The matrix $\bar{\Omega}$ relates to a subset of the big kernel matrix $\Omega \in \mathbb{R}^{N\times N}$. However, we should never calculate this big kernel matrix Ω in our proposed methodologies. The computation of the features corresponding to each point $x_i \in \mathcal{D}$ in matrix notation can be written as

$$\hat{\Phi} = \begin{bmatrix} \hat{\phi}_1(x_1) & \dots & \hat{\phi}_M(x_1) \\ \vdots & \ddots & \vdots \\ \hat{\phi}_1(x_N) & \dots & \hat{\phi}_M(x_N) \end{bmatrix}. \tag{4}$$

Solving (1) with the approximate feature matrix $\hat{\Phi} \in \mathbb{R}^{N\times M}$ in the primal as proposed in [1] results into solving the following linear system of equations:

$$\left[\begin{array}{c|c} \hat{\Phi}^\intercal\hat{\Phi} + \frac{1}{\gamma} I & \hat{\Phi}^\intercal 1_N \\ \hline 1_N^\intercal\hat{\Phi} & 1_N^\intercal 1_N \end{array}\right] \begin{bmatrix} \hat{w} \\ \hat{b} \end{bmatrix} = \begin{bmatrix} \hat{\Phi}^\intercal y \\ 1_N^\intercal y \end{bmatrix} \tag{5}$$

where $\hat{w} \in \mathbb{R}^M$, $\hat{b} \in \mathbb{R}$ are the model parameters in the primal space with $y \in \{+1, -1\}$ for classification and $y \in \mathbb{R}$ for regression.

3) Parameter Estimation for Very Large Data Sets: In [14], the authors propose a technique to obtain tuning parameters for very large-scale data sets. We use the same methodology to obtain the parameters of the model ($\hat{w}$ and $\hat{b}$) when the approximate feature matrix $\hat{\Phi}$ given by (4) cannot fit into memory. The basic concept is to decompose the feature matrix $\hat{\Phi}$ into a set of S blocks. Thus, $\hat{\Phi}$ is not required to be stored into memory completely. Let l_s, where $s = 1, \dots, S$, denote the number of rows in the sth block such that $\sum_{s=1}^{S} l_s = N$. The matrix $\hat{\Phi}$ can be described as

$$\hat{\Phi} = \begin{bmatrix} \hat{\Phi}_{[1]} \\ \vdots \\ \hat{\Phi}_{[S]} \end{bmatrix}$$

with $\hat{\Phi}_{[S]} \in \mathbb{R}^{l_s\times(M+1)}$ and the vector y is given by

$$y = \begin{bmatrix} y_{[1]} \\ \vdots \\ y_{[S]} \end{bmatrix}$$

with $y_{[S]} \in \mathbb{R}^{l_s}$. The matrix $\hat{\Phi}_{[S]}^\intercal\hat{\Phi}_{[S]}$ and the vector $\hat{\Phi}_{[S]}^\intercal y_{[S]}$ can be calculated in an updating scheme and stored efficiently in the memory since their sizes are $(M+1)\times(M+1)$ and $(M+1)\times 1$, respectively, provided that the size of each block, that is, l_s can fit into memory. Moreover, the following also holds:

$$\hat{\Phi}^\intercal\hat{\Phi} = \sum_{s=1}^{S} \hat{\Phi}_{[s]}^\intercal\hat{\Phi}_{[s]}, \qquad \hat{\Phi}^\intercal y = \sum_{s=1}^{S} \hat{\Phi}_{[s]}^\intercal y_{[s]}.$$

Algorithm 1 summarizes the overall idea.

Algorithm 1 PFS-LSSVM for Very Large-Scale Data [14]

Divide the training data $\mathcal{D}$ into approximately S equal blocks such that $\hat{\Phi}_{[s]}$ with $s = 1, \dots, S$, calculated using (4) can fit into memory.
Initialize matrix $A \in \mathbb{R}^{(M+1)\times(M+1)}$ and $c \in \mathbb{R}^{M+1}$.
for $s = 1$ *to* S **do**
  Calculate matrix $\hat{\Phi}_{[s]}$ for the sth block using Nyström approximation (4)
  $A \leftarrow A + \hat{\Phi}_{[s]}^\intercal\hat{\Phi}_{[s]}$
  $c \leftarrow c + \hat{\Phi}_{[s]}^\intercal y_{[s]}$
end
Set $A \leftarrow A + \frac{I_{M+1}}{\gamma}$
Solve the linear system (5) to obtain parameters $\hat{w}$,$\hat{b}$.

Algorithm 2 Primal FS-LSSVM Method

Data: $\mathcal{D} = \{(x_i, y_i) : x_i \in \mathbb{R}^d, y_i \in \{+1, -1\}$ for classification & $y_i \in \mathbb{R}$ for regression, $i = 1, \dots, N\}$.
1 Determine the kernel bandwidth using the multivariate rule-of-thumb.
2 Given the number of PV, perform PV selection by maximizing the quadratic Rènyi entropy.
3 Determine the learning parameters σ and γ performing fast υ-fold cross validation as described in [14].
4 **if** *the approximate feature matrix (4) can be stored into memory* **then**
5 Given the optimal learning parameters, obtain the PFS-LSSVM parameters $\hat{w}$ and $\hat{b}$ by solving the linear equation (5).
6 **else**
7 Use Algorithm 1 to obtain the PFS-LSSVM parameters $\hat{w}$ and $\hat{b}$.
8 **end**

B. Fast Initialization: SD-LSSVM

In this case, we propose a different approximation instead of the Nyström approximation and solve a subsampled LSSVM problem in the dual (SD-LSSVM). We first use the active subset selection method as described in [14] to obtain an initial set of PVs, that is, $\mathcal{S}_{\text{PV}}$. This set of points is obtained by maximizing the quadratic Rènyi entropy criterion, that is, approximate the information of the big $N \times N$ kernel matrix with a smaller $M \times M$ matrix and this can be considered as the set of representative points of the data set.

The assumption for the approximation in the proposed approach is that this set of PVs is sufficient to train an initial LSSVM model in the dual and is sufficient to obtain the tuning parameters σ and γ for the SD-LSSVM model. Here the major advantage is that it greatly reduces the computation time required for training and cross-validation ($O(M^3)$ in comparison to $O(NM^2)$ for PFS-LSSVM). This results in an approximate value for the tuning parameters close to the optimal values (for the entire training data set). However, as the training of the LSSVM model is performed in the dual, we no longer need explicit approximate feature maps and can have the original feature map of the form $\phi : \mathbb{R}^d \to \mathbb{R}^{n_h}$ where n_h denotes the dimension of the feature space which can be infinite dimensional.

Thus, the SD-LSSVM problem of training on the M PVs selected by Rènyi entropy is given by

$$\min_{\bar{w},\bar{b},\bar{e}} \mathcal{J}(\bar{w}, \bar{e}) = \frac{1}{2}\bar{w}^\intercal\bar{w} + \frac{\gamma}{2}\sum_{i=1}^{M} \bar{e}_i^2$$
$$\text{such that} \quad \bar{w}^\intercal\phi(z_i) + \bar{b} = y_i - \bar{e}_i, \quad i = 1, \dots, M \tag{6}$$

where $z_i \in \mathcal{S}_{\text{PV}}$ and $\mathcal{S}_{\text{PV}}$ is a subset of the whole data set $\mathcal{D}$.

III. Sparsifications

We propose two L_0-norm-reduced models starting from the initializations explained in Section II: one for the primal FS-LSSVM method namely the sparsified primal FS-LSSVM (SPFS-LSSVM) and the other one for the SD-LSSVM method namely sparsified subsampled dual LSSVM (SSD-LSSVM). Both models can handle very large-scale data efficiently.

A. L_0-Norm Reduced PFS-LSSVM—SPFS-LSSVM

In this section, we propose an approach using the L_0-norm to introduce a second level of sparsity resulting in a reduced set of PVs $\mathcal{S}_{\text{SV}}$ whose cardinality is M' and is giving a highly sparse solution. We modify the procedure described in [18] and [19]. The methodology used in [18] and [19] cannot be extended to large-scale data due to memory constraints ($O(N \times N)$) and computational costs ($O(N^3)$). The L_0-norm problem can be formulated as

$$\min_{\tilde{w},\tilde{b},\tilde{e}} \mathcal{J}(\tilde{w},\tilde{e}) = \|\tilde{w}\|_0 + \frac{\gamma}{2}\sum_{i=1}^{N}\tilde{e}_i^2 \quad \text{such that} \quad \tilde{w}^\intercal\hat{\phi}(x_i) + \tilde{b} = y_i - \tilde{e}_i, \quad i = 1,\ldots,N \quad (7)$$

where $\tilde{w}$, $\tilde{b}$ and $\tilde{e}_i, i = 1,\ldots,N$ are variables of the optimization problem and $\hat{\phi}$ is the explicit feature map as discussed in (3).

The weight vector $\tilde{w}$ can be approximated as a linear combination of the M PVs, that is, $\tilde{w} = \sum_{j=1}^{M}\tilde{\beta}_j\hat{\phi}(z_j)$ where $\tilde{\beta}_j \in \mathbb{R}$ which do not need to be the Lagrange multipliers. We apply the regularization weight λ_j on each of these $\tilde{\beta}_j$ to iteratively sparsify such that most of the $\tilde{\beta}_j$ move to zero leading to an approximate L_0-norm solution as shown in [18]. The L_0-norm problem can then be reformulated in terms of reweighting steps of the form

$$\min_{\tilde{\beta},\tilde{b},\tilde{e}} J(\tilde{\beta},\tilde{e}) = \frac{1}{2}\sum_{j=1}^{M}\lambda_j\tilde{\beta}_j^2 + \frac{\gamma}{2}\sum_{i=1}^{N}\tilde{e}_i^2 \quad \text{such that} \quad \sum_{j}^{M}\tilde{\beta}_j\hat{Q}_{ij} + \tilde{b} = y_i - \tilde{e}_i, \quad i = 1,\ldots,N. \quad (8)$$

The matrix $\hat{Q}$ is a rectangular matrix of size $N \times M$ and is defined by its elements $\hat{Q}_{ij} = \hat{\phi}(x_i)^\intercal\hat{\phi}(z_j)$ where $x_i \in \mathcal{D}$, $z_j \in \mathcal{S}_{\text{PV}}$. The set $\mathcal{S}_{\text{PV}}$ is a subset of the data set $\mathcal{D}$. This problem (8) is similar to the one formulated for SVMs in [18] and guarantees sparsity and convergence. It is well known that the L_p-norm problem is nonconvex for $0 < p < 1$. We obtain an approximate solution for $p \to 0$ by the iterative sparsification procedure and converge to a local minimum.

We propose to solve this problem in the primal which allows us to extend the sparsification procedure to large-scale data sets along with incorporating the information about the entire training data set $\mathcal{D}$. Thus, after eliminating the $\tilde{e}_i$ the optimization problem becomes

$$\min_{\tilde{\beta},\tilde{b}} J(\tilde{\beta},\tilde{b}) = \frac{1}{2}\sum_{j=1}^{M}\lambda_j\tilde{\beta}_j^2 + \frac{\gamma}{2}\sum_{i=1}^{N}\left(y_i - \left(\sum_{j=1}^{M}\tilde{\beta}_j\hat{Q}_{ij} + \tilde{b}\right)\right)^2. \quad (9)$$

Algorithm 3 SPFS-LSSVM Method

Data: Solve PFS-LSSVM (5) to obtain initial $\hat{w}$ and $\hat{b}$
$\tilde{\beta} = \hat{w}$
$\lambda_i \leftarrow \tilde{\beta}_i$, $i = 1,\ldots,M$
if *the $\hat{Q}$ matrix can be stored into memory* **then**
  Calculate $\hat{Q}^\intercal\hat{Q}$ once and store into memory.
else
  Divide into blocks for very large data sets computing $\hat{Q}_{[s]}^\intercal\hat{Q}_{[s]}$ in an additive updating scheme similar to procedure in Algorithm 1.
  Calculate once and store the $M \times M$ matrix into memory.
end
while *not convergence* **do**
  $H \leftarrow \hat{Q}^\intercal\hat{Q} + \text{diag}(\lambda)/\gamma$; Solve system (10) to give $\tilde{\beta}$ and $\tilde{b}$; $\lambda_i \leftarrow 1/\tilde{\beta}_i^2, i = 1,\ldots,M$;
end
Result: indices $= find(|\tilde{\beta}_i| > 0)$, $\beta' = \tilde{\beta}$(indices), $b' = \tilde{b}$.

The solution to (9) resembles the ridge regression solution (in case of zero bias term) and is obtained by solving

$$\left[\begin{array}{c|c} \hat{Q}^\intercal\hat{Q} + \frac{1}{\gamma}\text{diag}(\lambda) & \hat{Q}^\intercal 1_N \\ \hline 1_N^\intercal\hat{Q} & 1_N^\intercal 1_N \end{array}\right]\left[\begin{array}{c}\tilde{\beta}\\ \tilde{b}\end{array}\right] = \left[\begin{array}{c}\hat{Q}^\intercal y\\ 1_N^\intercal y\end{array}\right] \quad (10)$$

where $\text{diag}(\lambda)$ is a diagonal $M \times M$ matrix with diagonal elements λ_j. The iterative sparsification method is presented in Algorithm 3.

The procedure to obtain sparseness involves iteratively solving the system (10) for decreasing values of λ. Considering the tth iteration, we can build the matrix $H \leftarrow \hat{Q}^\intercal\hat{Q} + \text{diag}(\lambda)/\gamma$ from the in-memory matrix $\hat{Q}^\intercal\hat{Q}$ and the modified matrix $\text{diag}(\lambda^t)$, and solve the system of linear equations. From this solution, we get λ^{t+1} and the process is restarted. It was shown in [18] that as $t \to \infty$, $\tilde{\beta}_t$ converges to the L_0-norm solution asymptotically. This is shown in Algorithm 3. Because this β' depends on the initial choice of weights, we set them to the PFS-LSSVM solution $\hat{w}$ and $\hat{b}$ to avoid ending up in very different local minima. For this procedure, we need to calculate $\hat{Q}^\intercal\hat{Q}$ matrix just once and keep it into memory. The final predictive model is

$$f(x^*) = \sum_{i=1}^{M'}\beta_i'\hat{\phi}(z_i)^\intercal\hat{\phi}(x^*) + b'.$$

We use $f(x^*)$ for regression and $\text{sign}[f(x^*)]$ for classification. Table I provides a conceptual and notational overview of the steps involved in SPFS-LSSVM model.

B. L_0-Norm Reduced Subsampled Dual LSSVM

The SD-LSSVM performs a fast initialization on a subsample obtained by maximizing the quadratic Rènyi entropy. The SD-LSSVM model as described in Section II results in $\bar{\alpha}, i = 1,\ldots,M$ and $\bar{b}$ as a solution in the dual. However, we have not seen all the data in the training set for this case. Also, we do not know beforehand the ideal value of the cardinality M for the initial PV set. In general, we start with a large value of M for fixing the initial cardinality of the PV set. But we propose an iterative sparsification procedure for the SD-LSSVM model leading to an L_0-norm-based solution.

TABLE I

GIVEN THE DATA SET $\mathcal{D}$ WE FIRST PERFORM THE PRIMAL FS-LSSVM IN STEP 1. WE OBTAIN THE PV SET $\mathcal{S}_{\text{PV}}$, THE WEIGHT VECTOR $\hat{w}$ ALONG WITH THE EXPLICIT FEATURE MAP $\hat{\phi}: \mathbb{R}^d \to \mathbb{R}^M$. IN STEP 2 WE PERFORM THE SPFS-LSSVM, THAT IS, WE USE AN ITERATIVE SPARSIFYING L_0-NORM PROCEDURE IN THE PRIMAL WHERE $\tilde{w} = \sum_{j=1}^{M} \tilde{\beta}_j \hat{\phi}(z_j)$ AND REGULARIZATION WEIGHTS λ_j ARE APPLIED ON $\tilde{\beta}_j$. WE CONSTRUCT THE $\hat{Q}$ MATRIX WHICH HAS INFORMATION ABOUT THE ENTIRE TRAINING SET $\mathcal{D}$. AFTER THE SECOND REDUCTION WE OBTAIN THE SOLUTION VECTOR β' AND b'. THE SOLUTION VECTOR RELATES TO PVS OF THE HIGHLY SPARSE SOLUTION SET $\mathcal{S}_{\text{SV}}$

	Initial		1^{st} **Reduction**		2^{nd} **Reduction**
SV/Train	N/N		M/N		M'/N
Primal	w, $\mathcal{D}$	**Step 1** $\rightarrow$	$\hat{w}$, $\mathcal{S}_{\mathcal{PV}}$	**Step 2** $\rightarrow$	β', b', $\mathcal{S}_{\mathcal{SV}}$
	$\phi(x) \in \mathbb{R}^{n_h}$	**PFS-LSSVM**	$\hat{\phi}(x) \in \mathbb{R}^M$	**SPFS-LSSVM**	

This results in a set of reduced PVs $\mathcal{S}_{\text{SV}}$ whose cardinality is M' which can be much less than M. We use these reduced PVs along with the nonzero $\bar{\alpha}_i, i = 1, \ldots, M$ and $\bar{b}$ obtained as a result of the iterative sparsification procedure for the out-of-sample extensions (i.e., test data predictions). The L_0-norm problem for the SD-LSSVM model can then be formulated as

$$\min_{\bar{w},\bar{b},\bar{e}} \mathcal{J}(\bar{w},\bar{e}) = \|\bar{w}\|_0 + \frac{\gamma}{2}\sum_{i=1}^{N} \bar{e}_i^2$$
$$\text{such that} \quad \bar{w}^{\intercal}\phi(x_i) + \bar{b} = y_i - \bar{e}_i, \quad i = 1, \ldots, N \quad (11)$$

where $\bar{w}$, $\bar{b}$ and $\bar{e}_i, i = 1, \ldots, N$ are the variables of the optimization problem, and ϕ is the original feature map.

One of the KKT conditions of (6) is $\bar{w} = \sum_{i=1}^{M} \bar{\alpha}_i \phi(z_i)$ where $\bar{\alpha}_i$ are Lagrange dual variables. We apply the regularization weight λ_j on each of these $\bar{\alpha}_j$ to iteratively sparsify such that most of the $\bar{\alpha}_j$ move to zero leading to an approximate L_0-norm solution as shown in [18]. To handle large-scale data sets and obtain the nonzero $\bar{\alpha}_j$ leading to the reduced set of PVs $\mathcal{S}_{\text{SV}}$, we formulate the optimization problem by eliminating each $\bar{e}_i$. Thus, the optimization problem can be reformulated as

$$\min_{\bar{\alpha},\bar{b}} J(\bar{\alpha},\bar{b}) = \frac{1}{2}\sum_{j=1}^{M} \lambda_j \bar{\alpha}_j^2 + \frac{\gamma}{2}\sum_{i=1}^{N}\left(y_i - \left(\sum_{j=1}^{M} \bar{\alpha}_j Q_{ij} + \bar{b}\right)\right)^2. \quad (12)$$

The matrix Q is a rectangular matrix of size $N \times M$ and is defined by its elements $Q_{ij} = \phi(x_i)^{\intercal}\phi(z_j) = K(x_i, z_j)$ where $x_i \in \mathcal{D}$, $z_j \in \mathcal{S}_{\text{PV}}$. The variables x_j and z_j which are part of the PV set $\mathcal{S}_{\text{PV}}$ are used interchangeably. This marks the distinction between (12) and (9) where we use explicit approximate feature maps. The solution to (12) is obtained by solving

$$\left[\begin{array}{c|c} Q^{\intercal}Q + \frac{1}{\gamma}\text{diag}(\lambda) & Q^{\intercal}1_N \\ \hline 1_N^{\intercal}Q & 1_N^{\intercal}1_N \end{array}\right]\left[\begin{array}{c} \bar{\alpha} \\ \bar{b} \end{array}\right] = \left[\begin{array}{c} Q^{\intercal}y \\ 1_N^{\intercal}y \end{array}\right]. \quad (13)$$

Here $\text{diag}(\lambda)$ is a diagonal $M \times M$ matrix with diagonal elements λ_j. We train the SD-LSSVM only on the PV set (cardinality M) but we incorporate the information from the entire training data set (cardinality N) in the loss function while performing the iterative sparsification algorithm. This results into an improvement in performance as more information is incorporated in the model. The approach to perform an iterative sparsification procedure for the SD-LSSVM model is presented in Algorithm 4.

Algorithm 4 SSD-LSSVM Method

Data: Solve SD-LSSVM (6) on actively selected PV set to obtain the initial $\bar{\alpha}$ and $\bar{b}$
$\lambda_i \leftarrow \bar{\alpha}_i$, $i = 1, \ldots, M$
if *the Q matrix can be stored into memory* **then**
  Calculate $Q^{\intercal}Q$ once and store into memory.
else
  Divide into blocks for very large data sets computing $Q_{[s]}^{\intercal}Q_{[s]}$ in an additive updating scheme similar to procedure in Algorithm 1.
  Calculate once and store the $M \times M$ matrix into memory.
end
while *not convergence* **do**
  $H \leftarrow Q^{\intercal}Q + \text{diag}(\lambda)/\gamma$;
  Solve system (13) to give $\bar{\alpha}$ and $\bar{b}$;
  $\lambda_i \leftarrow 1/\bar{\alpha}_i^2, i = 1, \ldots, M$;
end
Result: indices $= find(|\bar{\alpha}_i| > 0)$, $\alpha' = \bar{\alpha}(\text{indices})$, $b' = \bar{b}$

TABLE II

GIVEN THE DATA SET $\mathcal{D}$ WE FIRST PERFORM THE SD-LSSVM AS A FAST INITIALIZATION IN STEP 1. WE OBTAIN THE DUAL LAGRANGE VARIABLES $\bar{\alpha}_i, i = 1, \ldots, M$. IN STEP 2, WE PERFORM THE SSD-LSSVM, THAT IS, WE USE AN ITERATIVE SPARSIFYING L_0-NORM PROCEDURE IN THE PRIMAL RESULTING IN A REDUCED SET OF VECTORS $\mathcal{S}_{\text{SV}}$. WE CONSTRUCT THE RECTANGULAR MATRIX Q WHICH INCORPORATES INFORMATION ABOUT THE ENTIRE TRAINING SET. AFTER THE SECOND REDUCTION, WE SELECT THE NONZERO $\bar{\alpha}_i$ AND $\bar{b}$ TO OBTAIN THE SOLUTION VECTOR α' AND b'

	Initial		1^{st} **Reduction**		2^{nd} **Reduction**
SV/Train	N/N		M/M		M'/N
Dual	α, $\mathcal{D}$	**Step 1** $\rightarrow$	$\bar{\alpha}$, $\mathcal{S}_{PV}$	**Step 2** $\rightarrow$	α', b', $\mathcal{S}_{SV}$
	$\phi(x) \in \mathbb{R}^{n_h}$	**SD-LSSVM**	$\phi(x) \in \mathbb{R}^{n_h}$	**SSD-LSSVM**	

The procedure to obtain sparseness works similarly as described in Section III-A. Once we obtain the indices corresponding to the nonzero $\bar{\alpha}_i$, we can obtain the reduced set of PVs SV corresponding to those nonzero $\bar{\alpha}_i$. We can then use α' and b' (as defined in Algorithm 4) along with these SV to perform the test predictions. The final predictive model is

$$f(x^*) = \sum_{i=1}^{M'} \alpha_i' K(z_i, x^*) + b'.$$

We use $f(x^*)$ for regression and $\text{sign}[f(x^*)]$ for classification. Table II provides a conceptual and notational overview of the steps involved in SSD-LSSVM model.

Figs. 1 and 2 compare the proposed SPFS-LSSVM and SSD-LSSVM methods with the normal PFS-LSSVM method for classification on the Ripley data set and for regression on the Boston Housing data. For the Boston Housing data set, we display the projection of all the data points on the first

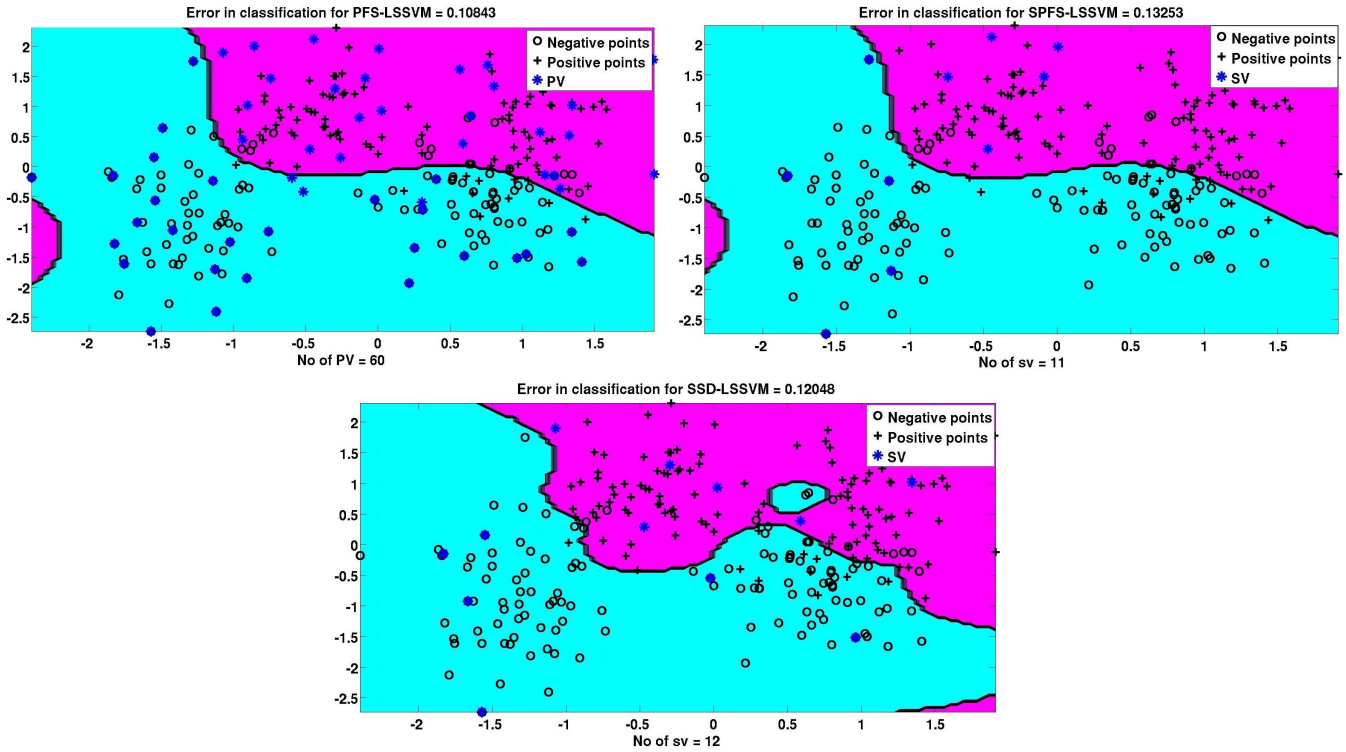


Fig. 1. Comparison of best results out of ten randomizations for the PFS-LSSVM method with the proposed approaches for the Ripley data set.

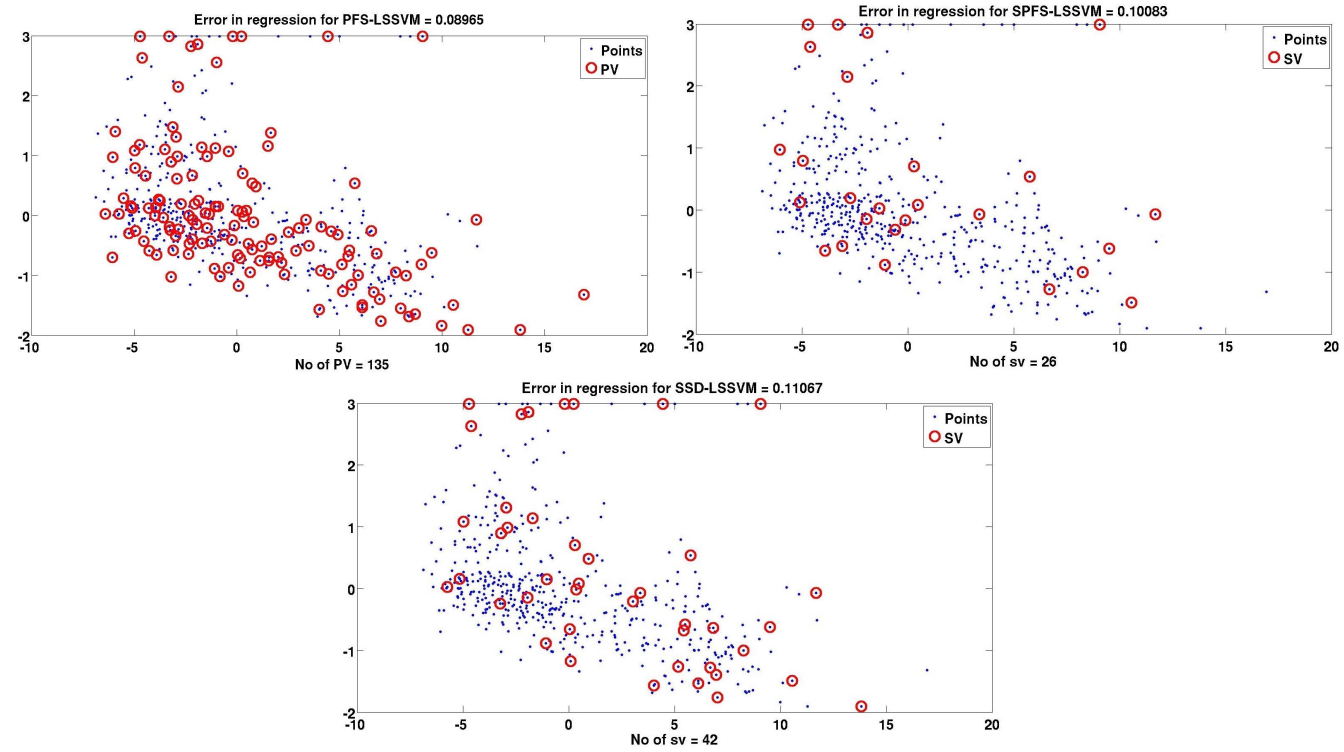


Fig. 2. Comparison of best results out of ten randomizations for the PFS-LSSVM method with the proposed approaches for the Boston Housing data set projected on the first eigenvector as the dimensions of the data set ($d > 3$). We use 337 training points whose projections are plotted for all the methods. This projection is only for visualization purposes.

eigenvector (for visualization purpose) along the x-axis while the estimator value is plotted along the y-axis since the data set has dimensions $d > 3$. From Fig. 1, we can observe that the PFS-LSSVM method results in a better decision boundary with lower prediction error. However, the cardinality of the SV set is much higher in comparison with the SPFS-LSSVM and

SSD-LSSVM methods. The proposed approaches result in a much sparser model without any significant tradeoff in error and have good decision boundaries.

From Fig. 2, we observe that the SPFS-LSSVM method results in better prediction errors than the SSD-LSSVM using a fewer number of PVs. In the SSD-LSSVM method during the training phase, we train just over the SV set, in comparison to the SPFS-LSSVM technique where we train over the entire training set. So, we might end up with an underdetermined model in the SSD-LSSVM case as less information is incorporated in the model. Fig. 2 also shows that the proposed methods select points with high and small predictor values as PVs for the set $\mathcal{S}_{\rm SV}$. However, the difference in errors between the proposed approaches is not very significant and when compared with the PFS-LSSVM method, the tradeoff in error with respect to the amount of sparsity gained is not significant.

IV. Computational Complexity and Experimental Results

The convergence of Algorithms 3 and 4 is assumed when the difference $\| \tilde{\beta}^t - \tilde{\beta}^{t+1} \| /M$ and $\| \bar{\alpha}^t - \bar{\alpha}^{t+1} \| /M$ is, respectively, lower than 10^{-4} or when the number of iterations t exceeds 50. The result of the two approaches is the indices of those SVs for which $|\tilde{\beta}_i| > 10^{-6}$ and $|\bar{\alpha}_i| > 10^{-6}$ which provides us the reduced set of PVs $\mathcal{S}_{\rm SV}$. We perform an analysis of the computation time required for the proposed approaches and are further described.

A. Computational Complexity

The computation time of the PFS-LSSVM method involves:

1) solving a linear system of size $M + 1$ where M is the number of PVs selected initially;
2) calculating the Nyström approximation and eigenvalue decomposition of the kernel matrix of size M once;
3) forming the matrix product $\Phi^\intercal \Phi$.

This computation time is $O(NM^2)$ where N is the data set size as shown in [14].

The computation time for the SPFS-LSSVM method comprises a v-fold cross-validation time ($O(vNM^2)$), time for matrix multiplication $\hat{Q}^\intercal \hat{Q}$ ($O(NM^2)$) once and iteratively solving a system of linear equations whose complexity is ($O(M^3)$). Since $M \ll N$, the overall complexity of SPFS-LSSVM is $O((v+1)NM^2)$.

The computation time for the SD-LSSVM method of training on M PVs is given by the time required to solve a system of linear equations with M variables and is equivalent to $O(M^3)$. For the proposed SSD-LSSVM approach, the computation time is $O(NM^2)$ for the matrix multiplication $Q^\intercal Q$ once and then iteratively solving a system of M linear equations ($O(M^3)$). Thus the overall time complexity of SSD-LSSVM method is much less in comparison with the SPFS-LSSVM technique. The major part of the computation time is required for cross validation to obtain the tuning parameters γ and σ for the proposed approaches.

Because there is no widely accepted approach for selecting an initial model selection value M, in our experiments we selected $M = \lceil k \times \sqrt{N} \rceil$ where $k \in \mathbb{N}$, the complexity of SPFS-LSSVM can be rewritten as $O(k^2N^2)$. The parameter k is a user-defined parameter and is not a tuning parameter. However, k should be chosen carefully such that the $N \times M$ matrices can fit into memory. For smaller data sets like Ripley, Breast-Cancer, Diabetes, Spambase, Magic Gamma, Boston Housing, Concrete, Slice Localization, and Adult, we experimented with different values of k. For each value of k, we obtained the optimal tuning parameters (σ and γ) along with the prediction errors. In Table III, we report the value of k corresponding to which we get the lowest classification and regression errors for these smaller data sets.

TABLE III
Classification and Regression Data Description and Initial Number of PVs Selected Such That $M = \lceil k \times \sqrt{N} \rceil$

Classification					
Dataset	N	Dims	N_{test}	PV	k
Ripley(RIP)	250	2	84	60	4
Breast-Cancer(BC)	682	10	227	155	6
Diabetes(DIB)	768	8	256	167	6
Spambase(SPAM)	4061	57	1354	204	3
Magic Gamma(MGT)	19020	11	6340	414	3
Adult(ADU)	48842	14	16281	664	3
Forest Cover(FC)	531012	54	177004	763	1
Regression					
Dataset	N	Dims	N_{test}	PVs	k
Boston Housing(BH)	506	14	169	135	6
Concrete(Con)	1030	9	344	192	6
Slice Localization(SLC)	53500	385	17834	694	3
Year Prediction(YP)	515345	90	171780	718	1

B. Data Set Description

All the data sets have been obtained from the UCI benchmark repository [23]. A brief description about the data sets is given in Table III.

C. Numerical Experiments

All experiments are performed on a PC machine with Intel Core i7 CPU and 8GB RAM under MATLAB 2008a. We use the RBF-kernel for the kernel matrix construction in all cases. As a preprocessing step, all records containing unknown values are removed from consideration. All given inputs are normalized to zero mean and unit variance. The codes for the proposed method and FS-LSSVM method are available at http://www.esat.kuleuven.be/sista/ADB/mall/softwareFS.php.

We compare the performance of our proposed approaches with methods including normal PFS-LSSVM classifier/regressor, SV_L_0-norm PFS-LSSVM proposed in [20], C-SVM and ν-SVM, L_0-norm method of Lopez [19], and Keerthi's method [22] for classification. The latter SVM and ν-SVM methods are implemented in the LIBSVM software. All methods use a cache size of 8 GB. Shrinking is applied in the SVM case. All comparisons are made on the same ten randomizations of the methods. The SV_L_0-norm PFS-LSSVM method tries to sparsify the PFS-LSSVM solution by an iterative sparsification procedure but its loss function only incorporates information about the set of PV vectors (M) while performing this operation. Thus, it results in solutions having more variations in error and more

TABLE IV

COMPARISON IN PERFORMANCE OF THE DIFFERENT METHODS FOR UCI REPOSITORY DATA SETS. *: NO CROSS-VALIDATION AND PERFORMANCE ON A FIXED VALUE OF TUNING PARAMETERS DUE TO COMPUTATIONAL BURDEN. -: METHOD CANNOT SCALE FOR THIS DATA SET

Test Classification(Error and Mean SV)														
	RIP		BC		DIB		SPAM		MGT		ADU		FC	
Algorithm	Error	SV	Error	SV	Error	SV	Error	SV	Error	SV	Error	SV	Error	SV
PFS-LSSVM [15]	**0.102 ± 0.008**	58	0.023 ± 0.004	153	**0.21 ± 0.008**	167	0.08 ± 0.002	204	**0.134 ± 0.001**	414	0.147 ± 0.0	664	0.2048 ± 0.026	763
C-SVC [22]	0.15 ± 0.07	81	0.0348 ± 0.02	414	0.332 ± 0.02	509	0.075 ± 0.067	800	0.144 ± 0.015	7000	0.151(∗)	11085(∗)	0.185(∗)	185000(∗)
ν-SVC [22]	0.167 ± 0.03	112	0.028 ± 0.011	398	0.338 ± 0.017	512	0.113 ± 0.07	1525	0.158 ± 0.014	7252	0.161(∗)	12205(∗)	184(∗)	165205(∗)
Lopez [20]	0.183 ± 0.1	7	0.04 ± 0.011	17	0.248 ± 0.034	10	0.0726 ± 0.005	340	-	-	-	-	-	-
Keerthi [23]	0.129 ± 0.033	26	0.0463 ± 0.037	30	0.228 ± 0.035	53	**0.07 ± 0.005**	175	0.135 ± 0.002	260	**0.145 ± 0.002**	494	0.2054(*)	762(*)
SV_L_0-norm [21]	0.1651 ± 0.13	15	0.023 ± 0.01	27	0.27 ± 0.05	39	0.127 ± 0.09	152	0.15 ± 0.009	75	0.149 ± 0.001	236	0.2346 ± 0.025	352
SPFS-LSSVM	0.147 ± 0.03	11	**0.021 ± 0.01**	26	0.273 ± 0.04	8	0.082 ± 0.002	169	0.141 ± 0.004	163	0.148 ± 0.0	464	0.2195 ± 0.03	505
SSD-LSSVM	0.13 ± 0.02	11	0.0256 ± 0.007	8	0.227 ± 0.02	6	0.0836 ± 0.005	63	0.135 ± 0.0	280	0.1488 ± 0.001	102	**0.1905 ± 0.009**	635

Test Regression(Error and Mean SV)								
	Boston Housing		Concrete		Slice Localization		Year Prediction	
Algorithm	Error	SV	Error	SV	Error	SV	Error	SV
PFS-LSSVM [15]	**0.133 ± 0.002**	113	**0.112 ± 0.006**	193	**0.0554 ± 0.0**	694	**0.402 ± 0.034**	718
ϵ-SVR [22]	0.16 ± 0.05	226	0.23 ± 0.02	670	0.102(*)	13012(*)	0.465(*)	192420(*)
ν-SVR [22]	0.16 ± 0.04	195	0.22 ± 0.02	330	0.092(*)	12524	0.472(*)	175250(*)
Lopez [20]	0.16 ± 0.05	65	0.165 ± 0.07	215	-	-	-	-
SV_L_0-norm [21]	0.192 ± 0.015	38	0.265 ± 0.02	25	0.156 ± 0.13	381	0.495 ± 0.03	422
SPFS-LSSVM	0.176 ± 0.03	50	0.2383 ± 0.04	46	0.058 ± 0.002	651	0.44 ± 0.05	595
SSD-LSSVM	0.189 ± 0.022	43	0.1645 ± 0.017	54	0.056 ± 0.0	577	0.42 ± 0.01	688

Train, Cross-validation & Test Classification(Computation Time)							
Algorithm	RIP	BC	DIB	SPAM	MGT	ADU	FC
PFS-LSSVM [15]	3.16 ± 0.18	27.36 ± 1.33	36.14 ± 2.4	182 ± 7	3142 ± 61	16523 ± 540	118500 ± 9294
C-SVC [22]	4.6 ± 0.83	14.76 ± 1	61 ± 75	1010 ± 53	20603 ± 396	13973(*)	58962(*)
ν-SVC [22]	5.67 ± 11	14.91 ± 1	83.5 ± 115	785 ± 22	13901 ± 189	13992.7(*)	53478(*)
Lopez [20]	5.12 ± 0.9	28.2 ± 5.4	39.1 ± 5.9	950 ± 78.5	-	-	-
Keerthi [23]	21.1 ± 0.6	74.34 ± 3.9	86.7 ± 5.12	1070 ± 37.7	16608 ± 397	20359 ± 762	9149.8(*)
SV_L_0-norm [21]	3.19 ± 0.16	27.384 ± 1.32	36.18 ± 2.4	182.2 ± 7	3143 ± 61	16527 ± 539	118540 ± 9267.9
SPFS-LSSVM	3.21 ± 0.16	27.42 ± 1.33	36.266 ± 2.4	182.3 ± 7	3155 ± 61	16588 ± 529	118570 ± 9247.6
SSD-LSSVM	**0.955 ± 0.05**	**8.67 ± 0.14**	**11.82 ± 0.14**	**81.5 ± 0.6**	**2192 ± 1.9**	**13482 ± 15.6**	**86460 ± 2.8**

Train, Cross-validation & Test Regression(Computational Time)				
Algorithm	Boston Housing	Concrete	Slice Localization	Year Prediction
PFS-LSSVM [15]	14.28 ± 0.2	58.41 ± 1.41	33769 ± 1571	248046 ± 15330
ϵ-SVR [22]	63 ± 1	168 ± 3	52438(*)	67231(*)
ν-SVR [22]	61 ± 1	131 ± 2	42724(*)	59899(*)
Lopez [20]	158.6 ± 3.2	753 ± 35.5	-	-
SV_L_0-norm [21]	14.58 ± 0.2	58.6 ± 1.48	33775 ± 1574	248178 ± 15262
SPFS-LSSVM	14.5 ± 0.2	58.76 ± 1.44	33790 ± 1571	248412 ± 15248
SSD-LSSVM	**3.75 ± 0.06**	**21.35 ± 0.137**	**25991 ± 11**	**86652 ± 24**

variations in the number of reduced PVs (SV). Details of the method are provided in [20]. The Lopez method cannot scale to large-scale data. Keerthi's method greedily finds a set of basis functions of a specified size using a Newton optimization technique. However, if the number of iterations required for the Newton method to converge increases, the time complexity increases and becomes even worse than the time required for the SVM methods as shown in Table IV.

For all the approaches, we use the method of coupled simulated annealing (CSA) as described in [33] to obtain the optimal tuning parameters namely γ, σ for PFS-LSSVM, LSSVM, SVM, Lopez, Keerthi, SV_L_0-norm PFS-LSSVM, SPFS-LSSVM, and SSD-LSSVM methods. We start by using five multiple random starters for each tuning parameter. For each combination of tuning parameters, we evaluate the cost. The cost is defined as the accuracy in the case of classification and mean squared error (mse) in the case of regression. This cost is obtained by performing one iteration of tenfold cross-validation of the corresponding method. These costs along with the combination of parameters are provided to CSA to obtain the optimal tuning parameters as illustrated in [33]. We fixed the ν parameter in ν-SVM to a value of 0.5 because if we tuned for ν then the method becomes computationally very expensive. Thus, the number of tenfold cross-validations performed for PFS-LSSVM, LSSVM, SVM, Lopez, Keerthi, and the two proposed approaches is 25 (5×5) for each randomization of these methods. The time reported in Table IV includes the time required for performing all these cross validations.

All comparisons are performed on an out-of-sample test set depicted as N_{test} in Table III consisting of $1/3$ of the data. The first $2/3$ of the data is reserved for training and cross validation. Several techniques [26]–[28] use cross validation for estimating the model parameters as it optimizes the bias-variance tradeoff. For each algorithm, the average set performances and sample standard deviations on the same ten randomizations are reported. Also the mean total time (training, cross validation, and testing) and the corresponding standard deviation, the mean number of PVs for each method is listed in Table IV.

Table IV provides a comparison of the mean estimated error, mean value of cardinality of prototype vectors (PV or SV, denoted in Table IV by SV), and a comparison of the mean run-time computations of the proposed approaches with Keerthi's, Lopez method, PFS-LSSVM and SVM methods for various classification and PFS-LSSVM and SVR methods

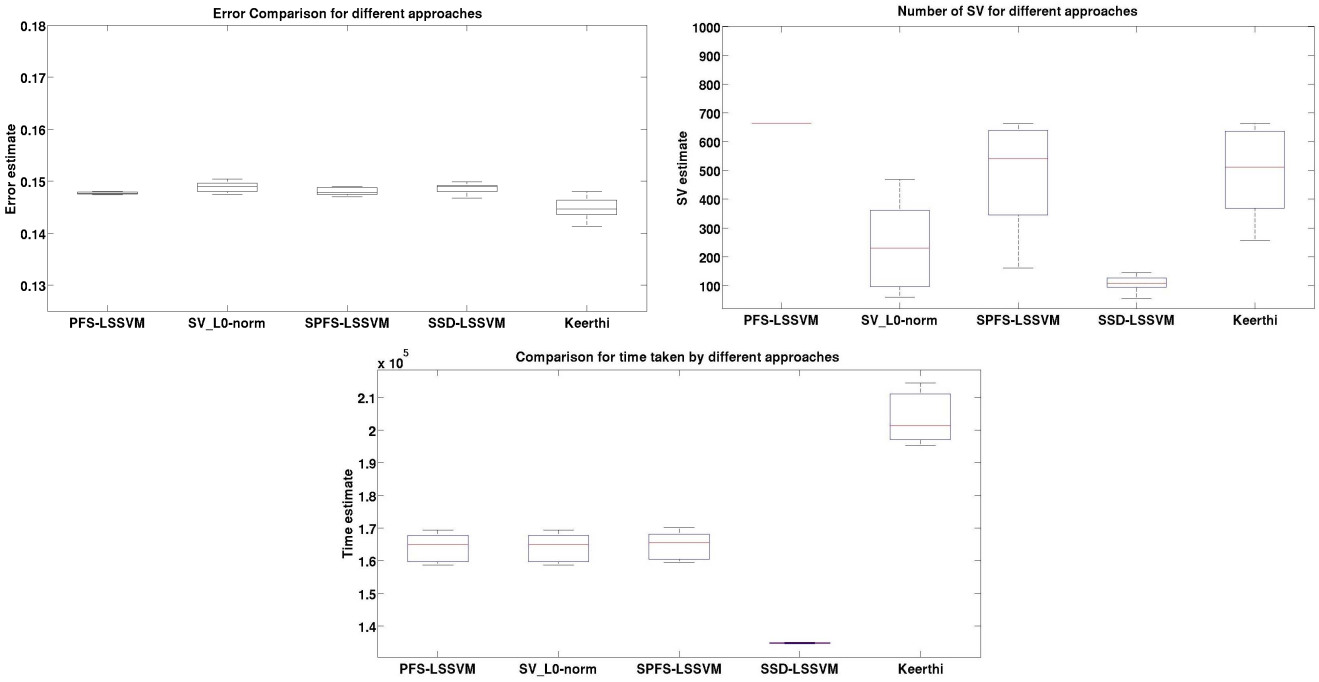


Fig. 3. Comparison of the proposed approaches with PFS-LSSVM, SV_L_0-norm PFS-LSSVM, and Keerthi's method for the Adult data set.

for different regression data sets. Fig. 3 shows the estimated error, run time performance, and variations in the number of PVs for Adult data set (ADU). From Fig. 3, we observe that Keerthi's method result in best prediction errors but requires the maximum computation time as well. The variations in the prediction errors by the proposed SPFS-LSSVM and SSD-LSSVM methods are insignificant and their estimated errors are comparable with that of the PFS-LSSVM method though they require a much smaller PVs set. An important observation was that the SSD-LSSVM method produces the maximum amount of sparsity and has the least computation time without significant tradeoff in error and thus making it highly suitable for very large-scale data sets. The SSD-LSSVM method works well because the initial set of PVs (PV) that it selects is only helping to construct a basis set where this basis set is maximizing the quadratic Rènyi entropy criterion. We also observe that the number of reduced PVs (SV) can vary a lot for L_0-norm-based methods as during each randomization we start with a different initialization and after performing the iterative sparsification procedure we end up in a different local minimum. This results in variations in the number of reduced SVs for these methods.

D. Performance Analysis

In case of classification data sets, the proposed approaches called SPFS-LSSVM and SSD-LSSVM work much better in comparison with SVM methods both in terms of sparsity and prediction errors as observed from Table IV. Their errors are comparable with those of PFS-LSSVM and Keerthi's method and in the case of the Forest Cover data set even better than PFS-LSSVM method. The amount of sparsity introduced is quite high for the proposed approaches, which is consistent with the fact that the L_0-norm leads to highly sparse solutions. However, an observation shows that the number of PVs is reduced to a maximum extent in most of the data sets for the SSD-LSSVM approach. The SPFS-LSSVM method results in best prediction error for Breast-Cancer (BC) data set while the SSD-LSSVM method gives best results for the very large-scale FC data set as highlighted in Table IV. The amount of sparsity introduced varies for different data sets. For example, for the BC data set the PFS-LSSVM method uses 33.65%, Keerthi's method uses 6.59%, SPFS-LSSVM and SSD-LSSVM method use 5.71% and 1.76% of the training data, respectively, as SV without significant tradeoff in errors.

From Fig. 4, we observe the performance for the Forest Cover (FC) data set. PFS-LSSVM uses 0.22% while SV_L_0-norm PFS-LSSVM uses 0.1%, SPFS-LSSVM uses 0.14% and SSD-LSSVM method uses 0.18% of the training data as SV. This suggests that for the FC data set the amount of sparsity achieved by the proposed approaches is much less in comparison with that for the BC data set. The SSD-LSSVM results have less variations, whereas the SV_L_0-norm PFS-LSSVM method has maximum variance in error predictions which can be attributed to the lack of information in the loss function that the technique was incorporating while performing the iterative sparsification procedure. This results in more variations in the number of SV obtained for the ten randomizations. The computation time for PFS-LSSVM, SV_L_0-norm PFS-LSSVM, SPFS-LSSVM are higher in comparison with that of SSD-LSSVM which follows the line of the reasoning provided in Section IV-A. Another important observation is that since SPFS-LSSVM and SSD-LSSVM result in much fewer number of PVs SV, the time required for out-of-sample predictions (testing time) for these methods is much less in comparison to the other methods.

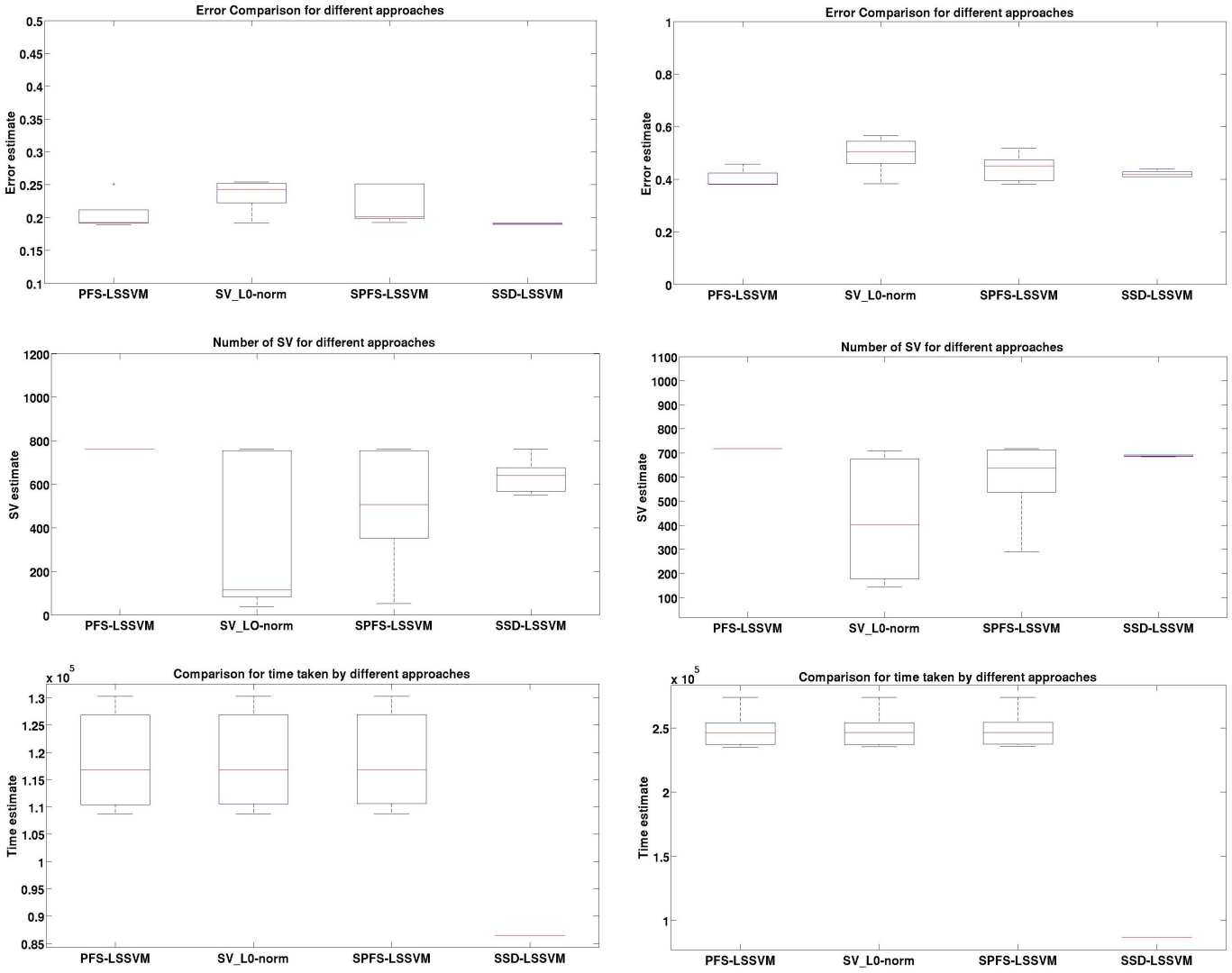


Fig. 4. Comparison of performance of the proposed approaches with PFS-LSSVM, SV_L_0-norm PFS-LSSVM method for FC & year prediction data sets.

For data sets like Boston Housing and Concrete, the estimated error by the proposed methods is larger than for the PFS-LSSVM method, but the amount of sparsity introduced is quite significant. This is because regression in general requires a large number of SVs as observed for the different SVR approaches in Table IV. From Table IV, we observe that for the proposed approaches the higher the mean value of SV, the lower is the estimated error. An exception is the case of Slice Localization data set which can be attributed to the nonparametric nature of SSD-LSSVM as we are not using an explicit feature map in this method and thus information about all the features (dimensions) are intact. From Fig. 4, we observe the performance of the various approaches for the large-scale Year Prediction data set. The PFS-LSSVM method has the least error estimates, but the SSD-LSSVM approach results in comparable prediction errors with a much lower computation cost. We also observe that the number of reduced PVs (SV) can vary a lot for the L_0-norm-based methods as during each randomization, we start with a different initialization and after performing the iterative sparsification procedure, we end up in a different local minimum.

V. Sparsity Versus Error Tradeoff

In this section, we perform additional experiments to illustrate the sparsity versus error tradeoff between PFS-LSSVM and SPFS-LSSVM techniques and between SD-LSSVM and SSD-LSSVM methods.

A. Additional Experiments

Table V provides a comparison between the error variations and the mean number of PVs for PFS-LSSVM and SPFS-LSSVM and SD-LSSVM and SSD-LSSVM techniques. For PFS-LSSVM and SD-LSSVM methods, we report the value of M as the number of PVs. For the proposed techniques we report the mean value of M' as the number of reduced PVs (SV). From Table V, we observe that the PFS-LSSVM performs the best with respect to error variations but the amount of sparsity introduced by the proposed techniques is quite significant without much tradeoff in error. An interesting observation is that the SD-LSSVM method results in poor prediction errors in comparison with the SSD-LSSVM method for most of the data sets. This is because the SD-LSSVM works with only M PV vectors while training, cross-validating,

TABLE V

ERROR AND MEAN SV COMPARISONS FOR PFS-LSSVM (INITIAL) AND SPFS-LSSVM (AFTER SPARSIFICATION), SD-LSSVM (INITIAL), AND SSD-LSSVM (AFTER SPARSIFICATION)

Classification								
	Intial Step		Final Step		Initial Step		Final Step	
Dataset	PFS-LSSVM		SPFS-LSSVM		SD-LSSVM		SSD-LSSVM	
	Error	PV	Error	SV	Error	PV	Error	SV
RIP	0.1 ± 0.01	58	0.15 ± 0.03	11	0.12 ± 0.03	58	0.13 ± 0.02	11
BC	0.02 ± 0.0	153	0.02 ± 0.01	26	0.02 ± 0.03	153	0.03 ± 0.01	8
DIB	0.2 ± 0.01	167	0.27 ± 0.04	8	0.22 ± 0.01	167	0.23 ± 0.02	6
SPAM	0.08 ± 0.0	204	0.082 ± 0.0	169	0.15 ± 0.01	204	0.083 ± 0.01	63
MGT	0.13 ± 0.0	414	0.14 ± 0.0	163	0.19 ± 0.01	414	0.135 ± 0.0	280
ADU	0.15 ± 0.0	664	0.15 ± 0.0	464	0.16 ± 0.0	664	0.15 ± 0.0	102
FC	0.20 ± 0.03	763	0.22 ± 0.03	505	0.26 ± 0.0	763	0.19 ± 0.01	635
Regression								
	Intial Step		Final Step		Initial Step		Final Step	
Dataset	PFS-LSSVM		SPFS-LSSVM		SD-LSSVM		SSD-LSSVM	
	Error	PV	Error	SV	Error	PV	Error	SV
BH	0.13 ± 0.0	113	0.18 ± 0.03	50	0.12 ± 0.0	113	0.19 ± 0.02	43
Con	0.11 ± 0.0	193	0.24 ± 0.04	46	0.16 ± 0.2	193	0.16 ± 0.02	54
SLC	0.06 ± 0.0	694	0.06 ± 0.0	651	0.08 ± 0.0	694	0.06 ± 0.0	577
YP	0.4 ± 0.03	718	0.44 ± 0.05	595	0.49 ± 0.0	718	0.42 ± 0.0	688

and testing. But the SSD-LSSVM uses the information about the entire training set N in its loss function while performing the iterative sparsification procedure. Thus, it results in a set of PVs (SV) which improves prediction errors particularly for large-scale data sets like Adult, Forest Covertype for classification, Slice Localization, and Year Prediction data sets for regression.

B. Tradeoff Analysis

Table VI illustrates the sparsity versus error tradeoff for the proposed L_0-norm-based reductions to PFS-LSSVM and SD-LSSVM methods for various classification and regression data sets of different sizes.

Sparsity is calculated as the fraction between the change in the number of PVs to the total number of PVs, that is, $(|\mathcal{S}_{\text{PV}}| - |\mathcal{S}_{\text{SV}}|)/|\mathcal{S}_{\text{PV}}|$. We use the metric fractional change in error for classification. For the PFS-LSSVM and SPFS-LSSVM it is defined as $(\text{err}_{\text{SPFS}} - \text{err}_{\text{PFS}})/\text{err}_{\text{PFS}}$ where err_{SPFS} is the error corresponding to SPFS-LSSVM and err_{PFS} is the error corresponding to PFS-LSSVM. Similarly, for the SD-LSSVM and SSD-LSSVM it is defined as $(\text{err}_{\text{SSD}} - \text{err}_{\text{SD}})/\text{err}_{\text{SD}}$ where err_{SSD} is the error corresponding to SSD-LSSVM and err_{SD} is the error corresponding to SD-LSSVM. For regression we report the mse. It is sufficient to investigate the change in error, that is, $\text{err}_{\text{SPFS}} - \text{err}_{\text{PFS}}$ and $\text{err}_{\text{SSD}} - \text{err}_{\text{SD}}$ with reference to the sparsity introduced by the proposed approaches.

From Table VI, we observe that for some data sets the fractional change in error or the change in error takes negative values. This means that by introducing sparsity the performance of the model has improved which is seen only for the case of the SD-LSSVM and SSD-LSSVM methods, because the SD-LSSVM model is built by incorporating information about only the M PVs selected by maximizing the quadratic Rènyi entropy. On the other hand, for the SSD-LSSVM during the sparsification process we incorporate the information about the entire training set (N data points) and thus it results in a better predictive model. This method works particularly well for large-scale data sets as can be observed from the results

TABLE VI

SPARSITY VERSUS ERROR TRADEOFF FOR PFS-LSSVM (INITIAL) AND SPFS-LSSVM (AFTER SPARSIFICATION), SD-LSSVM (INITIAL), AND SSD-LSSVM (AFTER SPARSIFICATION). NEGATIVE CHANGE IN ERROR MEANS THAT BY INTRODUCING SPARSITY THE PERFORMANCE OF THE MODEL HAS ACTUALLY IMPROVED OR REMAINED UNCHANGED. THESE RESULTS ARE HIGHLIGHTED. THIS IS EXPLAINED IN DETAIL IN SECTION V-B

Classification				
	Intial Step	Final Step	Initial Step	Final Step
Dataset	PFS-LSSVM & SPFS-LSSVM		SD-LSSVM & SSD-LSSVM	
	Sparsity	Fractional Change in Error	Sparsity	Fractional Change in Error
RIP	0.81	0.50	0.81	0.08
BC	0.63	0.00	0.95	0.50
DIB	0.95	0.35	0.96	0.05
SPAM	0.17	0.03	0.69	**-0.45**
MGT	0.61	0.08	0.32	**-0.29**
ADU	0.30	**0.00**	0.85	**-0.06**
FC	0.34	0.10	0.17	**-0.27**
Regression				
	Intial Step	Final Step	Initial Step	Final Step
Dataset	PFS-LSSVM & SPFS-LSSVM		SD-LSSVM & SSD-LSSVM	
	Sparsity	Change in Error	Sparsity	Change in Error
BH	0.56	0.05	0.62	0.07
Con	0.76	0.13	0.72	**0.00**
SLC	0.07	**0.00**	0.17	**-0.02**
YP	0.17	0.04	0.04	**-0.07**

corresponding to SPAM, MGT, ADU, FC, SLC, and YP data sets. In case of classification, the sparsity introduced for these data sets by the SD-LSSVM and SSD-LSSVM methods is quite substantial as shown in Table VI.

For the PFS-LSSVM and SPFS-LSSVM methods, if the amount of sparsity increases then the change in error also increases. This is reflected for data sets like RIP, DIB, and Con in Table VI. However, for large-scale data sets like MGT, ADU, FC, SLC, and YP a considerable amount of sparsity is introduced without much change in error estimation of the model as highlighted in Table VI.

The PFS-LSSVM can scale to large data sets but the resulting models are not the sparsest due to the problem of selection of smallest cardinality value (M) for the PV set. We overcome this problem with the proposed SPFS-LSSVM. The LSSVM model in general cannot scale for large-scale data sets as it creates and $N \times N$ kernel matrix. To overcome this problem we introduced the SD-LSSVM which trains on the smaller PV set. We further incorporate the information about large training set (N) while performing the L_0-norm on top of it. This results in the SSD-LSSVM. By doing this we prevent loss of information in the final model. Thus, the two levels of sparsity allow to scale to large-scale data sets while having very sparse models.

VI. CONCLUSION

In this paper, we proposed two techniques namely SPFS-LSSVM and SSD-LSSVM resulting in very sparse solutions for mining large-scale data sets. The problem of selection of smallest cardinality M for the PV set faced by the PFS-LSSVM method is solved by using an iterative sparsifying L_0-norm procedure resulting in very sparse solutions. The computation time required by one of the proposed approaches (SSD-LSSVM) is quite low owing to lower

training, cross-validation, and most importantly out-of-sample extension time because of fewer PVs. We compared the error predictions of proposed approaches with the recent approaches like Keerthi's method, SV_L_0-norm PFS-LSSVM, Lopez method, and traditional approaches like ν-SVM and C-SVM with good performance and sparser models. We also investigated the tradeoff between sparsity versus performance and showed that a significant amount of sparsity can be obtained without much change in error.

ACKNOWLEDGMENT

This paper reflects only the authors' views, the Union is not liable for any use that may be made of the contained information.

REFERENCES

[1] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle, *Least Squares Support Vector Machines*. Singapore: World Scientific, 2002.

[2] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Process. Lett.*, vol. 9, no. 3, pp. 293–300, 1999.

[3] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York, NY, USA: Springer-Verlag, 1995.

[4] J. A. K. Suykens, L. Lukas, and J. Vandewalle, "Sparse approximation using least squares support vector machines," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2000, pp. 757–760.

[5] D. Geebelen, J. A. K. Suykens, and J. Vandewalle, "Reducing the number of support vectors of SVM classifiers using the smoothed separable case approximation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 4, pp. 682–688, Apr. 2012.

[6] C. J. Burges, "Simplified support vector decision rules," in *Proc. 13th Int. Conf. Mach. Learn.*, 1996, pp. 71–77.

[7] B. Schölkopf *et al.*, "Input space versus feature space in kernel-based methods," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 10, no. 5, pp. 1000–1017, Sep. 1999.

[8] T. Downs, K. E. Gates, and A. Masters, "Exact simplification of support vector solutions," *J. Mach. Learn. Res.*, vol. 2, pp. 293–297, Dec. 2001.

[9] Y.-J. Lee and O. L. Mangasarian, "RSVM: Reduced support vector machines," in *Proc. 1st SIAM Int. Conf. Data Mining*, Apr. 2001, pp. 325–361.

[10] Y.-J. Lee and O. L. Mangasarian, "SSVM: A smooth support vector machine for classification," *Comput. Optim. Appl.*, vol. 20, no. 1, pp. 5–22, 2001.

[11] Y. Li, C. Lin, and W. Zhang, "Improved sparse least-squares support vector machine classifiers," *Neurocomputing*, vol. 69, no. 13, pp. 1655–1658, 2006.

[12] E. J. Nyström, "Über die praktische auflösung von integralgleichungen mit anwendungen auf randwertaufgaben," *Acta Math.*, vol. 54, no. 1, pp. 185–204, 1930.

[13] C. T. H. Baker, *The Numerical Treatment of Integral Equations*. Oxford, U.K.: Claredon Press, 1983.

[14] K. De Brabanter, J. De Brabanter, J. A. Suykens, and B. De Moor, "Optimized fixed-size kernel models for large data sets," *Comput. Statist. Data Anal.*, vol. 54, no. 6, pp. 1484–1504, 2010.

[15] P. Karsmakers, K. Pelckmans, K. De Brabanter, H. Van Hamme, and J. A. K. Suykens, "Sparse conjugate directions pursuit with application to fixed-size kernel models," *Mach. Learn.*, vol. 85, nos. 1–2, pp. 109–148, 2011.

[16] J. Weston, A. Elisseeff, B. Schölkopf, and M. Tipping, "Use of the zero norm with linear models and kernel methods," *J. Mach. Learn. Res.*, vol. 3, pp. 1439–1461, Mar. 2003.

[17] E. J. Candès, M. B. Wakin, and S. P. Boyd, "Enhancing sparsity by reweighted ℓ_1 minimization," *J. Fourier Anal. Appl.*, vol. 14, nos. 5–6, pp. 877–905, 2008.

[13] G. C. Cawley and N. L. Talbot, "Improved sparse least-squares support vector machines," *Neurocomputing*, vol. 48, nos. 1–4, pp. 1025–1031, 2002.

[18] K. Huang, D. Zheng, J. Sun, Y. Hotta, K. Fujimoto, and S. Naoi, "Sparse learning for support vector classification," *Pattern Recognit. Lett.*, vol. 31, no. 13, pp. 1944–1951, 2010.

[19] J. López, K. De Brabanter, J. R. Dorronsoro, and J. A. K. Suykens, "Sparse LSSVMs with L_0-norm minimization," in *Proc. Eur. Symp. Artif. Neural Netw., Comput. Intell. Mach. Learn. (ESANN)*, Apr. 2011, pp. 189–194.

[20] R. Mall and J. A. K. Suykens, "Sparse variations to fixed-size least squares support vector machines for large scale data," in *Proc. 17th Pacific-Asia Conf. Knowl. Discovery Data Mining (PAKDD)*, 2013.

[21] C.-C. Chang and C.-J. Lin, "LIBSVM : A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1–27, 2011.

[22] S. S. Keerthi, O. Chapelle, and D. DeCoste, "Building support vector machines with reduced classifier complexity," *J. Mach. Learn. Res.*, vol. 7, pp. 1493–1515, Jul. 2006.

[23] C. L. Blake and C. J. Merz. *UCI Repository of Machine Learning Databases* [Online]. Available: http://archive.ics.uci.edu/ml/datasets.html, accessed 2007.

[24] R. Fletcher, *Practical Methods of Optimization*. New York, NY, USA: Wiley, 1987.

[25] C. Williams and M. Seeger, "Using the Nyström method to speed up kernel machines," in *Advances in Neural Information Processing Systems*, vol. 13. Cambridge, MA, USA: MIT Press, 2001, pp. 682–688.

[26] M. Rudemo, "Empirical choice of histograms and kernel density estimators," *Scandin. J. Statist.*, vol. 9, no. 2, pp. 65–78, 1982.

[27] L. Du, Y. Yang, D. He, R. G. Harley, T. G. Habetler, and B. Lu, "Support vector machine based methods for non-intrusive identification of miscellaneous electric loads," in *Proc. 38th Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Montreal, QC, Canada, Oct. 2012, pp. 4866–4871.

[28] A. W. Bowman, "An alternative method of cross-validation for the smoothing of density estimates," *Biometrika*, vol. 71, no. 2, pp. 353–360, 1984.

[29] D. W. Scott and G. R. Terrel, "Biased and unbiased cross-validation in density estimation," *J. Amer. Statist. Assoc.*, vol. 82, pp. 1131–1146, Jan. 1987.

[30] C. C. Taylor, "Bootstrap choice of the smoothing parameter in kernel density estimation," *Biometrika*, vol. 76, no. 4, pp. 705–712, 1989.

[31] S. J. Sheather and M. C. Jones, "A reliable data-based bandwidth selection method for kernel density estimation," *J. Roy. Statist. Soc.*, vol. 53, no. 3, pp. 683–690, 1991.

[32] D. W. Scott and S. R. Sain, "Multi-dimensional density estimation," *Data Mining Comput. Statist.*, vol. 23, pp. 229–263, Aug. 2004.

[33] S. Xavier-de-Souza, J. A. K. Suykens, J. Vandewalle, and D. Bolle, "Coupled simulated annealing," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 40, no. 2, pp. 320–335, Apr. 2010.



Raghvendra Mall was born in Kolkata, India, in 1988. He received the B.Tech. and M.S. degrees in computer science from IIIT Hyderabad, Hyderabad, India, in 2010 and 2011, respectively. He is currently pursuing the Ph.D. degree with ESAT-STADIUS, Katholieke Universiteit Leuven, Leuven, Belgium, in 2012.

He is currently developing techniques to explore sparsity in large-scale machine learning problems. His current research interests include sampling and community detection in complex networks, exploring the role of sparsity in classification, clustering, and regression.



Johan A. K. Suykens (SM'05) received the M.S. degree in electro-mechanical engineering and the Ph.D. degree in applied sciences from Katholieke Universiteit Leuven (KU Leuven), Leuven, Belgium, in 1989 and 1995, respectively.

He was a Visiting Post-Doctoral Researcher with the University of California at Berkeley, Berkeley, CA, USA, in 1996. He has been a Post-Doctoral Researcher with the Fund for Scientific Research, FWO Flanders, Brussel, Belgium, and is currently a Professor with KU Leuven. He has authored the books *Artificial Neural Networks for Modelling and Control of Non-Linear Systems* (Kluwer Academic Publishers) and *Least Squares Support Vector Machines* (World Scientific), has co-authored the book *Cellular Neural Networks, Multi-Scroll Chaos and Synchronization* (World Scientific), and edited the books *Nonlinear Modeling: Advanced Black-Box Techniques* (Kluwer Academic Publishers), *Advances in Learning Theory: Methods, Models and Applications* (IOS Press), and *Regularization, Optimization, Kernels, and Support Vector Machines* (Chapman and Hall/CRC).