Very Sparse LSSVM Reductions for Large Scale Data

Raghvendra Mall and Johan A.K. Suykens

***Abstract*—Least Squares Support Vector Machines (LSSVM) have been widely applied for classification and regression with comparable performance to 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) was previously proposed in [1] to introduce sparsity using Nyström approximation with a set of prototype vectors (PV). The PFS-LSSVM model solves an over-determined system of linear equations in the primal. However, this solution is not the sparsest. We investigate the sparsity-error trade-off 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 datasets from the UCI repository illustrate that these approaches achieve sparse models without a significant trade-off in errors.**

***Index Terms*—L_0-norm, reduced models, LSSVM classification & regression, sparsity**

I. Introduction

Least Squares Support Vector Machines (LSSVM) 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.

A drawback of LSSVM models is the lack of sparsity as usually all the data points become support vectors (SV) 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:

1) *Reduction methods*: - Training the model on the dataset, pruning support vectors and selecting the rest for retraining the model.
2) *Direct methods*: - Enforcing sparsity from the beginning.

Some works in the first category are [4], [5], [6], [7], [8], [11], [9] and [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. The authors of [8] eliminate the support vectors that are linearly dependent on other support vectors. In [9] and [10], the authors work on a reduced set for optimization by pre-selecting a subset of data as support vectors without emphasizing much on the selection methodology. The authors of [11] prune the support vectors which are farthest from the decision boundary. This is done recursively until the performance degrades. Another work [12] in this direction suggests to select the support vectors closer to the decision boundary. However, these techniques cannot guarantee a large reduction in the number of support vectors.

In the second category, the number of support vectors 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 [13] and [14]. 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 [15] 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 [16] where they iteratively build up a conjugate set of vectors of increasing cardinality to approximately solve the over-determined PFS-LSSVM linear system. The approach works most efficiently when few iterations suffice for a good approximation. However, when few iterations don't 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 non-zero 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 [17] and [18] etc. In this paper, we modify the iterative sparsification procedure introduced in [19] and [20]. The major drawbacks of the methods described in [19] and [20] are that these approaches cannot scale to very large scale datasets due to memory ($N \times N$ kernel matrix) and computational ($O(N^3)$

Raghvendra Mall, Department of Electrical Engineering, ESAT-STADIUS (SCD), KU Leuven, B-3001 Leuven, Belgium, email: rmall@esat.kuleuven.be.

Johan A.K. Suykens, Department of Electrical Engineering, ESAT-STADIUS (SCD), KU Leuven, B-3001 Leuven, Belgium, email:johan.suykens@esat.kuleuven.be.

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:

- Initialization by Primal Fixed-Size LSSVM: Sparsification of the primal fixed-size LSSVM (PFS-LSSVM) method leads to a highly sparse parametric model namely sparsified primal FS-LSSVM (SPFS-LSSVM).
- 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 non-parametric 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 [22] software, Keerthi's method [23], L_0-norm based method proposed by Lopez [20] and the L_0-reduced PFS-LSSVM method (SV_L_0-norm PFS-LSSVM) [21] on several benchmark datasets from the UCI repository [24]. Below we mention some motivations to obtain a sparse solution:

- Sparseness can be exploited for having more memory and computationally efficient techniques, e.g. in matrix multiplications and inversions.
- Sparseness is essential for practical purposes such as scaling the algorithm to very large scale datasets. Sparse solutions means fewer support vectors and less time required for out-of-sample extensions.
- 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.
- The two level of sparsity allows scaling to large scale datasets while having very sparse models.

We also investigate the sparsity versus error trade-off.

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, i.e., SPFS-LSSVM and SD-LSSVM, i.e., SSD-LSSVM are discussed in Section III. In Section IV, the different algorithms are successfully demonstrated on real-life datasets. We discuss about the sparsity versus error trade-off in Section V. Section VI states the conclusion of the paper.

II. Initializations

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

A. Primal FS-LSSVM

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

Given a sample of N data points $\{x_i, y_i\}$, $i = 1, ..., N$, 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:

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

where $\phi : \mathbb{R}^d \to \mathbb{R}^{n_h}$ is a feature map to a high dimensional feature space, where n_h denotes 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) [25] 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 non-sparse as whenever e_i is non-zero 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 datasets 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 to $\alpha \in \mathbb{R}^N$. 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 prototype vectors (PV) from the whole dataset. In [15], the authors provide a method to select this subsample of size $M \ll N$ by maximizing the Quadratic Rènyi entropy.

Williams and Seeger [26] uses 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 denote 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 $\mathcal{S}_{PV}$ which is a subset of the whole dataset $\mathcal{D}$. 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:

$$\hat{\Phi} = \left[\begin{array}{ccc} \hat{\phi}_1(x_1) & \ldots & \hat{\phi}_M(x_1) \\ \vdots & \ddots & \vdots \\ \hat{\phi}_1(x_N) & \ldots & \hat{\phi}_M(x_N) \end{array}\right]. \tag{4}$$

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:

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

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 Datasets: In [15], the authors propose a technique to obtain tuning parameters for very large scale datasets. We utilize 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 $s = 1, \ldots, S$, denote the number of rows in the s^{th} block such that $\sum_{s=1}^{S} l_s = N$. The matrix $\hat{\Phi}$ can be described as:

$$\hat{\Phi} = \left[\begin{array}{c} \hat{\Phi}_{[1]} \\ \vdots \\ \hat{\Phi}_{[S]} \end{array}\right],$$

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

$$y = \left[\begin{array}{c} y_{[1]} \\ \vdots \\ y_{[S]} \end{array}\right],$$

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, i.e., l_s can fit into memory. Moreover, the following also holds:

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

Algorithm 1 summarizes the overall idea.

Algorithm 1: PFS-LSSVM for very large scale data [15]

Divide the training data $\mathcal{D}$ into approximately S equal blocks such that $\hat{\Phi}_{[s]}$ with $s = 1, \ldots, S$, 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 s^{th} 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, $i = 1, \ldots, N\}$.
Determine the kernel bandwidth using the multivariate rule-of-thumb.
Given the number of PV, perform prototype vector selection by maximizing the quadratic Rènyi entropy.
Determine the learning parameters σ and γ performing fast v-fold cross validation as described in [15].
if *the approximate feature matrix (4) can be stored into memory* **then**
  Given the optimal learning parameters, obtain the PFS-LSSVM parameters $\hat{w}$ and $\hat{b}$ by solving the linear equation (5).
else
  Use Algorithm 1 to obtain the PFS-LSSVM parameters $\hat{w}$ and $\hat{b}$.
end

B. Fast Initialization: Subsampled Dual 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 [15] to obtain an initial set of prototype vectors PV, i.e., $\mathcal{S}_{PV}$. This set of points is obtained by maximizing the quadratic Rènyi entropy criterion, i.e., approximate the information of the big $N \times N$ kernel matrix by means of a smaller $M \times M$ matrix and this can be considered as the set of representative points of the dataset.

The assumption for the approximation in the proposed approach is that this set of prototype vectors 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 dataset). 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 prototype vectors selected by Rènyi entropy is given by:

$$\begin{aligned} \min_{\bar{w},\bar{b},\bar{e}} \quad & \mathcal{J}(\bar{w},\bar{e}) = \frac{1}{2}\bar{w}^\intercal\bar{w} + \frac{\gamma}{2}\sum_{i=1}^{M}\bar{e}_i^2 \\ \text{s.t.} \quad & \bar{w}^\intercal\phi(z_i) + \bar{b} = y_i - \bar{e}_i, i = 1, \ldots, M, \end{aligned} \tag{6}$$

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

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 prototype vectors $\mathcal{S}_{SV}$ whose cardinality is M' and is giving a highly sparse solution. We modify the procedure described in [19] and [20]. The methodology used in [19] and [20] 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:

$$\begin{aligned} \min_{\tilde{w},\tilde{b},\tilde{e}} \quad & \mathcal{J}(\tilde{w},\tilde{e}) = \|\tilde{w}\|_0 + \frac{\gamma}{2}\sum_{i=1}^{N}\tilde{e}_i^2 \\ \text{s.t.} \quad & \tilde{w}^\intercal\hat{\phi}(x_i) + \tilde{b} = y_i - \tilde{e}_i, i = 1,\ldots,N, \end{aligned} \tag{7}$$

where $\tilde{w}$, $\tilde{b}$ and $\tilde{e}_i, i = 1,\ldots,N$ are the 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 prototype vectors, i.e., $\tilde{w} = \sum_{j=1}^{M}\tilde{\beta}_j\hat{\phi}(z_j)$ where $\tilde{\beta}_j \in \mathbb{R}$ which don't 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 [19]. The L_0-norm problem can then be re-formulated in terms of reweighting steps of the form:

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

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 $x_i \in \mathcal{D}$, $z_j \in \mathcal{S}_{PV}$. The set $\mathcal{S}_{PV}$ is a subset of the dataset $\mathcal{D}$. This problem (8) is similar to the one formulated for SVMs in [19] and guarantees sparsity and convergence. It is well known that the L_p-norm problem is non-convex 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 datasets along with incorporating the information about the entire training dataset $\mathcal{D}$. Thus, after eliminating the $\tilde{e}_i$ the optimization problem becomes:

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

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

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

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

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 datasets 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**
  $H \leftarrow \hat{Q}^\intercal\hat{Q} + diag(\lambda)/\gamma$;
  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 procedure to obtain sparseness involves iteratively solving the system (10) for decreasing values of λ. Considering the t^{th} iteration, we can build the matrix $H \leftarrow \hat{Q}^\intercal\hat{Q} + diag(\lambda)/\gamma$ from the in-memory matrix $\hat{Q}^\intercal\hat{Q}$ and the modified matrix $diag(\lambda^t)$ and solve the system of linear equations. From this solution we get λ^{t+1} and the process is restarted. It was shown in [19] that as $t \to \infty$, $\tilde{\beta}_t$ converges to the L_0-norm solution asymptotically. This is shown in Algorithm 3. Since, 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 sign$[f(x^*)]$ for classification. Table I provides a conceptual and notational overview of the steps involved in SPFS-LSSVM model.

	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**	

TABLE I: Given the dataset $\mathcal{D}$ we first perform the primal FS-LSSVM in Step 1. We obtain the prototype vector set $\mathcal{S}_{PV}$, 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, i.e., 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 2^{nd} reduction we obtain the solution vector β' and b'. The solution vector relates to prototype vectors of the highly sparse solution set $\mathcal{S}_{SV}$.

B. L_0-norm reduced subsampled dual LSSVM - SSD-LSSVM

The subsampled dual LSSVM (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 don't 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. This results in a set of reduced prototype vectors $\mathcal{S}_{SV}$ whose cardinality is M' which can be much less than M. We use these reduced prototype vectors along with the non-zero $\bar{\alpha}_i, i = 1, \ldots, M$ 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:

$$\begin{aligned} \min_{\bar{w},\bar{b},\bar{e}} \quad & \mathcal{J}(\bar{w},\bar{e}) = \|\bar{w}\|_0 + \frac{\gamma}{2}\sum_{i=1}^{N}\bar{e}_i^2 \\ \text{s.t.} \quad & \bar{w}^\intercal\phi(x_i) + \bar{b} = y_i - \bar{e}_i, i = 1, \ldots, N, \end{aligned} \tag{11}$$

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 [19]. In order to handle large scale datasets and obtain the non-zero $\bar{\alpha}_j$ leading to the reduced set of prototype vectors $\mathcal{S}_{SV}$, we formulate the optimization problem by eliminating each $\bar{e}_i$. Thus, the optimization problem can be reformulated as:

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

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 $x_i \in \mathcal{D}$, $z_j \in \mathcal{S}_{PV}$. The variables x_j and z_j which are part of the PV set $\mathcal{S}_{PV}$ 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:

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

Here $diag(\lambda)$ 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 dataset (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 datasets 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**
  $H \leftarrow Q^\intercal Q + diag(\lambda)/\gamma$;
  Solve system (13) to give $\bar{\alpha}$ and $\bar{b}$;
  $\lambda_i \leftarrow 1/\bar{\alpha}_i^2, i = 1, \ldots, M$;
end
Result: $indices = find(|\bar{\alpha}_i| > 0)$, $\alpha' = \bar{\alpha}(indices), b' = \bar{b}$

The procedure to obtain sparseness works similarly as described in Section III-A. Once we obtain the $indices$ corresponding to the non-zero $\bar{\alpha}_i$, we can obtain the reduced set of prototype vectors SV corresponding to those non-zero $\bar{\alpha}_i$. 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 sign$[f(x^*)]$ for classification. Table II provides a conceptual and notational overview of the steps involved in SSD-LSSVM model.

	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**	

TABLE II: Given the dataset $\mathcal{D}$ 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, i.e., we use an iterative sparsifying L_0-norm procedure in the primal resulting in a reduced set of vectors $\mathcal{S}_{SV}$. We construct the rectangular matrix Q which incorporates information about the entire training set. After the 2^{nd} reduction we select the non-zero $\bar{\alpha}_i$ and $\bar{b}$ to obtain the solution vector α' and b'.

Figure 1 and 2 compare the proposed SPFS-LSSVM method and SSD-LSSVM method with the normal PFS-LSSVM method for classification on the Ripley dataset and for regression on the Boston Housing data. For the Boston Housing dataset we display the projection of all the data points on the first eigenvector (for visualization purpose) along the $x\text{-}axis$ while the estimator value is plotted along the $y\text{-}axis$ since the dataset has dimensions $d > 3$. From Figure 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 trade-off in error and have good decision boundaries.

From Figure 2, we observe that the SPFS-LSSVM method results in better prediction errors than the SSD-LSSVM using a fewer number of prototype vectors. 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 under-determined model in the SSD-LSSVM case as less information is incorporated in the model. Figure 2 also shows that the proposed methods select points with high and small predictor values as prototype vectors for the set $\mathcal{S}_{SV}$. However, the difference in errors between the proposed approaches is not very significant and when compared to the PFS-LSSVM method, the trade-off in error with respect to the amount of sparsity gained is not significant.

IV. Computational Complexity & Experimental Results

The *convergence* of Algorithm 3 and 4 is assumed when the difference $\parallel \tilde{\beta}^t - \tilde{\beta}^{t+1} \parallel /M$ and $\parallel \bar{\alpha}^t - \bar{\alpha}^{t+1} \parallel /M$ respectively is 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 prototype vectors $\mathcal{S}_{SV}$. We perform an analysis of the computation time required for the proposed approaches and are futher described.

A. Computational Complexity

The computation time of the PFS-LSSVM method involves:

- Solving a linear system of size $M+1$ where M is the number of prototype vectors selected initially (PV).
- Calculating the Nyström approximation and eigenvalue decomposition of the kernel matrix of size M once.
- Forming the matrix product $\Phi^\intercal\Phi$.

This computation time is $O(NM^2)$ where N is the dataset size as shown in [15].

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 prototype vectors 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.

Since 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 re-written 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 datasets 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 datasets.

B. Dataset Description

All the datasets have been obtained from the UCI benchmark repository [24]. A brief description about the datasets is given in Table III.

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

TABLE III: Classification & Regression Data Description and initial number of prototype vectors (PV) selected such that $M = \lceil k \times \sqrt{N} \rceil$.

C. Numerical Experiments

All experiments are performed on a PC machine with Intel Core i7 CPU and 8 GB RAM under Matlab 2008a. We use the RBF-kernel for the kernel matrix construction in all cases. As a pre-processing 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 [21], C-SVM and ν-SVM, L_0-norm method of Lopez [20] and Keerthi's method [23] 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 10 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 variations in the

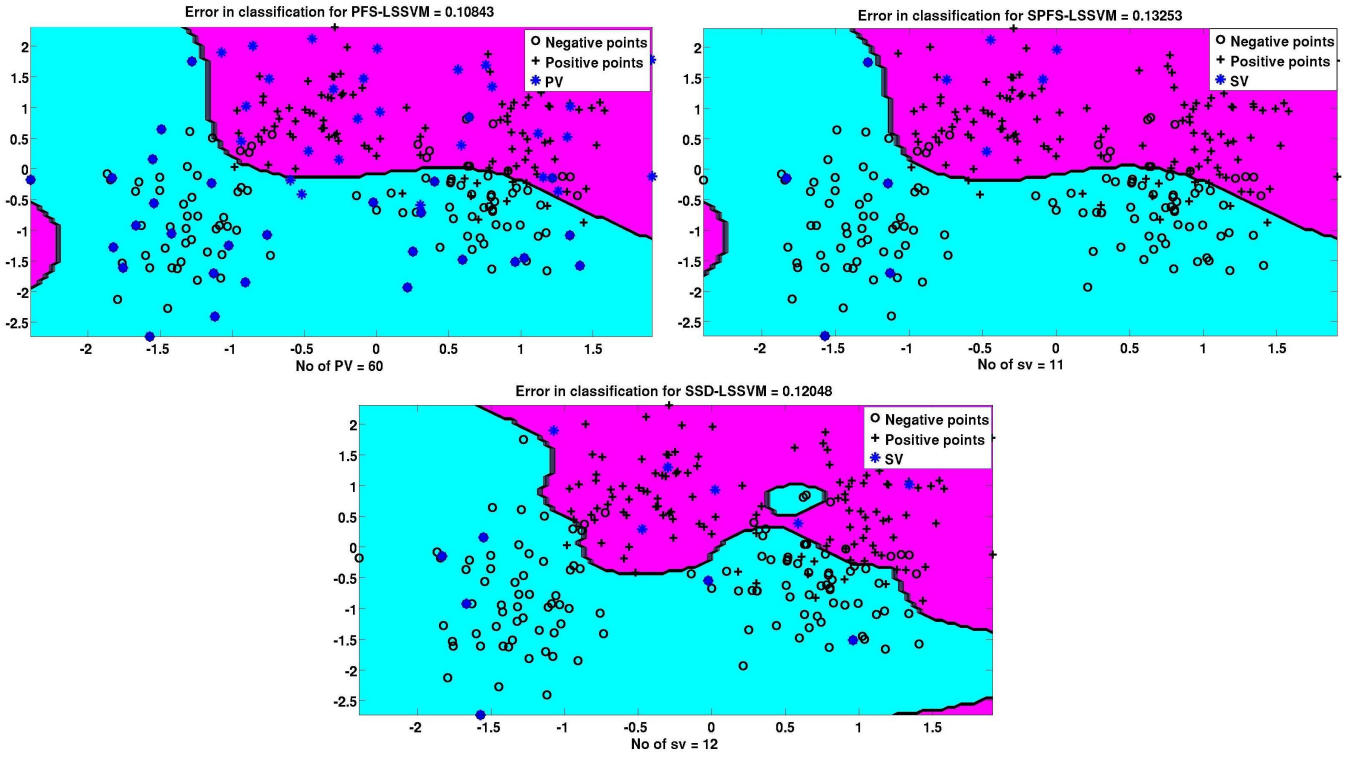

Fig. 1: Comparison of best results out of 10 randomizations for the PFS-LSSVM method with the proposed approaches for the Ripley dataset.

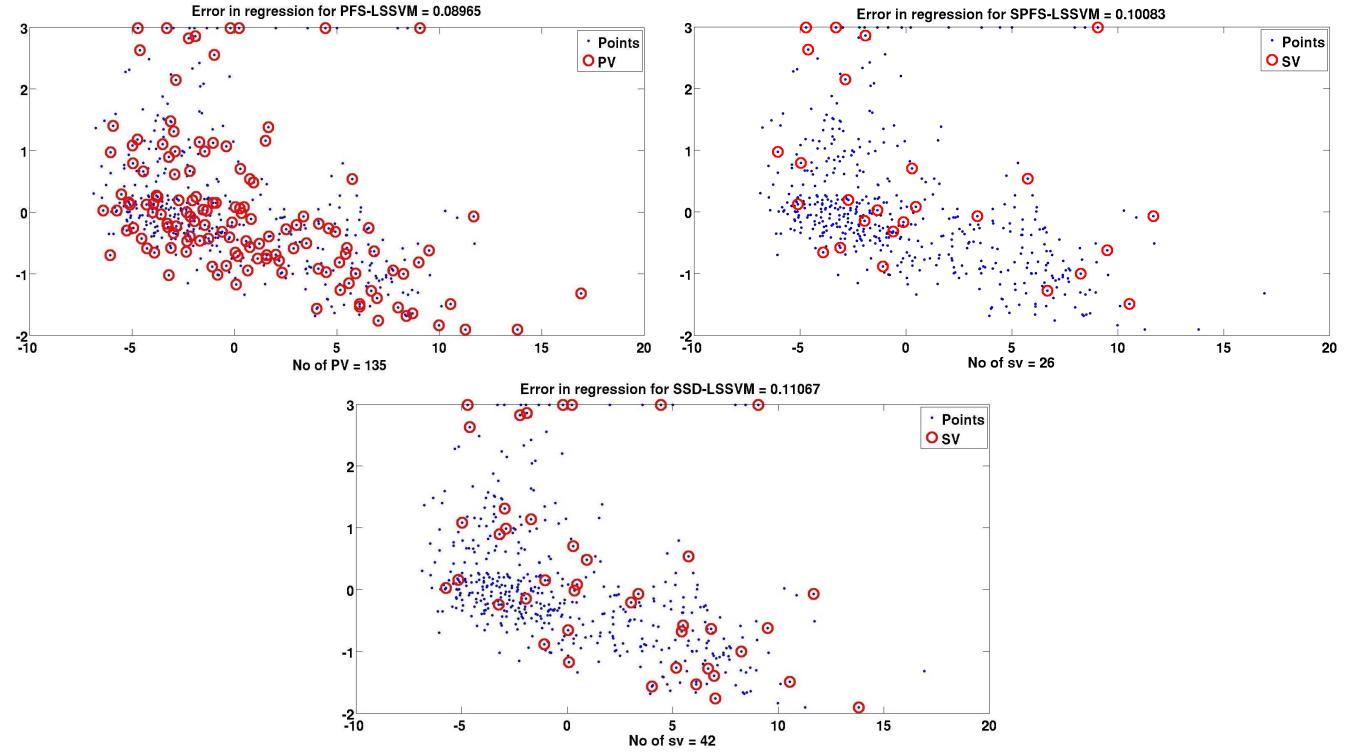

Fig. 2: Comparison of best results out of 10 randomizations for the PFS-LSSVM method with the proposed approaches for the Boston Housing dataset projected on the first eigenvector as the dimensions of the dataset ($d > 3$). We use 337 training points whose projections are plotted for all the methods. This projection is only for visualization purposes.

number of reduced prototype vectors (SV). Details of the method are provided in [21]. 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 [34] 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 5 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 10-fold 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 [34]. 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 10-fold 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 [27], [28], [29] use cross-validation for estimating the model parameters as it optimizes the bias-variance trade-off. For each algorithm, the average set performances and sample standard deviations on the same 10 randomizations are reported. Also the mean total time (training, cross-validation and testing) and the corresponding standard deviation, the mean number of prototype vectors for each method is depicted 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 for different regression datasets. Figure 3 represents the estimated error, run time performance and variations in the number of prototype vectors for Adult dataset (ADU). From Figure 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 method are insignificant and their estimated errors are comparable to that of the PFS-LSSVM method though they require a much smaller prototype vectors set. An important observation was that the SSD-LSSVM method produces the maximum amount of sparsity and has the least computation time without significant trade-off in error and thus making it highly suitable for very large scale datasets. The SSD-LSSVM method works well because the initial set of prototype vectors (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 prototype vectors (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 support vectors (SV) for these methods.

D. Performance Analysis

In case of classification datasets, 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 to those of PFS-LSSVM and Keerthi's method and in the case of the Forest Cover dataset 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 prototype vectors is reduced to a maximum extent in most of the datasets for the SSD-LSSVM approach. The SPFS-LSSVM method results in best prediction error for Breast-Cancer(BC) dataset while the SSD-LSSVM method gives best results for the very large scale FC dataset as highlighted in Table IV. The amount of sparsity introduced varies for different datasets. For example, for the BC dataset 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 trade-off in errors.

From Figure 4, we observe the performance for the Forest Cover (FC) dataset. 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 dataset the amount of sparsity achieved by the proposed approaches is much less in comparison to that for the BC dataset. The SSD-LSSVM results have less variations while 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 10 randomizations. The computation time for PFS-LSSVM, SV_L_0-norm PFS-LSSVM, SPFS-LSSVM are higher in comparison to 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 prototype vectors SV, the time required for out-of-sample predictions (testing time) for these methods is much less in comparison to the other methods.

For datasets 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

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**

TABLE IV: Comparison in performance of the different methods for UCI repository datasets. The (*) represents no cross-validation and performance on a fixed value of tuning parameters due to computational burden and (-) represents the method cannot scale for this dataset.

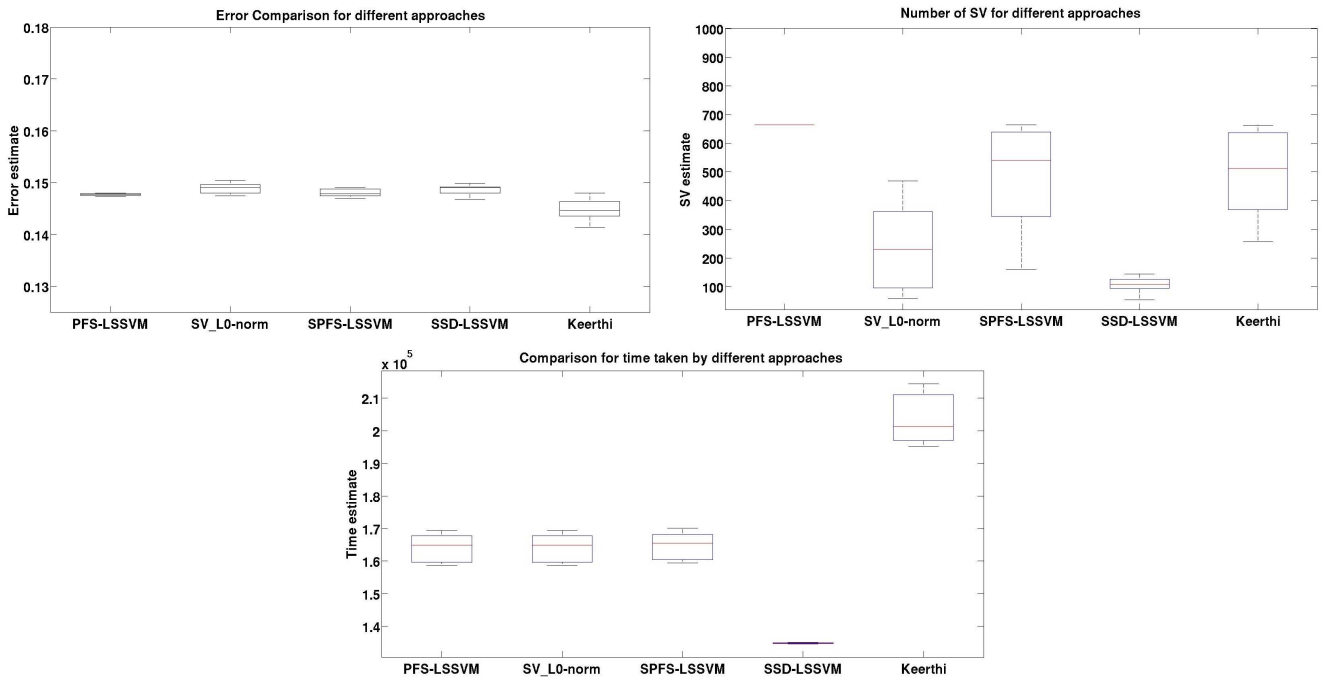

Fig. 3: Comparison of the proposed approaches with PFS-LSSVM, SV_L_0-norm PFS-LSSVM and Keerthi's method for the Adult dataset.

quite significant. This is because regression in general requires a large number of support vectors 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 dataset which can be attributed to the non-parametric 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 Figure 4, we observe the performance of the various approaches for the large scale Year Prediction dataset. 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 prototype vectors (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 Trade-off

In this section, we perform additional experiments to illustrate the sparsity versus error trade-off 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 prototype vectors 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 prototype vectors PV. For the proposed techniques we report the mean value of M' as the number of reduced prototype vectors 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 trade-off 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 datasets. This is because the SD-LSSVM works with only M PV vectors while training, cross-validating 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 prototype vectors SV which improves prediction errors particularly for large scale datasets like Adult, Forest Covertype for classification, Slice Localization and Year Prediction datasets for regression.

B. Trade-off Analysis

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

Sparsity is calculated as the fraction between the change in the number of prototype vectors to the total number of prototype vectors, i.e., $\frac{|\mathcal{S}_{PV}|-|\mathcal{S}_{SV}|}{|\mathcal{S}_{PV}|}$. We use the metric fractional change in error for classification. For the PFS-LSSVM and SPFS-LSSVM it is defined as $(err_{SPFS} - err_{PFS})/err_{PFS}$ 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

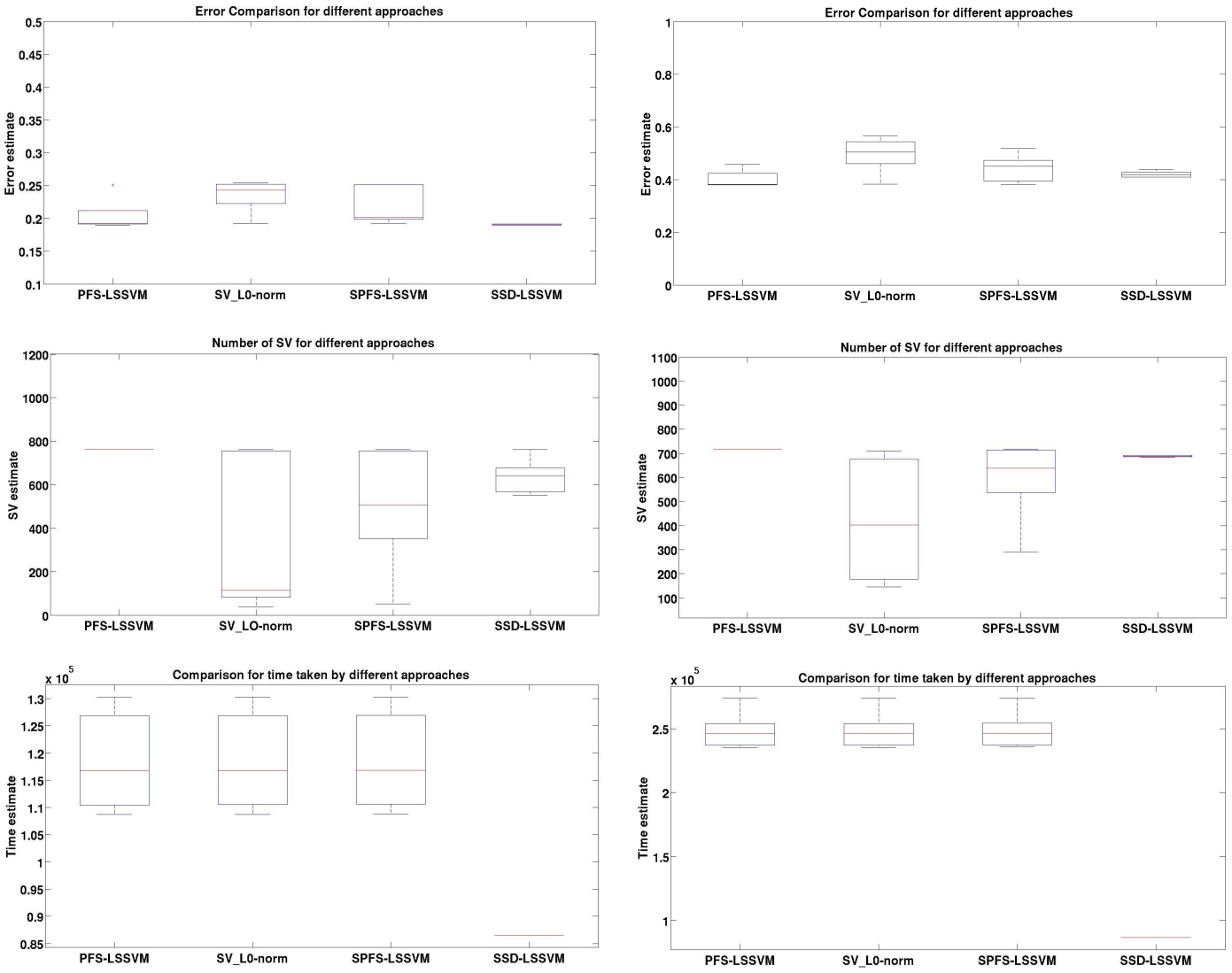

Fig. 4: Comparison of performance of the proposed approaches with PFS-LSSVM, SV_L_0-norm PFS-LSSVM method for FC & Year Prediction datasets.

as $(err_{SSD} - err_{SD})/err_{SD}$ 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 mean squared error (mse). It is sufficient to investigate the change in error, i.e., $err_{SPFS} - err_{PFS}$ and $err_{SSD} - err_{SD}$ w.r.t. the sparsity introduced by the proposed approaches.

From Table VI we observe that for some datasets 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. This is seen only for the case of the SD-LSSVM and SSD-LSSVM methods. The reason for this improvement is that the subsampled dual LSSVM (SD-LSSVM) model is built by incorporating information about only the M prototype vectors selected by maximizing the quadratic Rènyi entropy. On the other hand for the sparsified subsampled dual LSSVM (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 datasets as can be observed from the results corresponding to SPAM, MGT, ADU, FC, SLC and YP datasets. In case of classification, the sparsity introduced for these datasets 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 datasets like RIP, DIB and Con in Table VI. However, for large scale datasets 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 datasets 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 by the means of the proposed SPFS-LSSVM. The LSSVM model in general cannot scale for large scale

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

TABLE V: Error & 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	
	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**

TABLE VI: Sparsity versus Error trade-off 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.

datasets 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 allows to scale to large scale datasets 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 datasets. 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 due to lower training, cross-validation and most importantly out-of-sample extension time because of fewer prototype vectors. 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 trade-off between sparsity versus performance and showed that a significant amount of sparsity can be obtained without much change in error.

Acknowledgments

EU:The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC AdG A-DATADRIVE-B (290923). This paper reflects only the authors' views, the Union is not liable for any use that may be made of the contained information. Research Council KUL: GOA/10/09 MaNet, CoE PFV/10/002 (OPTEC), BIL12/11T; PhD/Postdoc grants, Flemish Government: FWO: projects: G.0377.12 (Structured systems), G.088114N (Tensor based data similarity); PhD/Postdoc grants IWT: projects: SBO POM (100031); PhD/Postdoc grants, iMinds Medical Information Technologies SBO 2014 and Belgian Federal Science Policy Office: IUAP P7/19 (DYSCO, Dynamical systems, control and optimization, 2012-2017)

References

[1] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor and J. Vandewalle. Least Squares Support Vector Machines. World Scientific Publishing Co, Pte, Ltd (Singapore), (ISBN: 981-238-151-1), 2002.

[2] J. A. K. Suykens and J. Vandewalle. "Least Squares Support Vector Machine Classifiers". Neural Processing Letters, Vol 9, No 3, pp: 293-300, 1999.

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

[4] J. A. K. Suykens, L. Lukas and J. Vandewalle. "Sparse approximation using least squares support vector machines". IEEE International Symposium on Circuits and Systems ISCAS 2000, pp. 757-760, 2000.

[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 Transactions on Neural Networks and Learning Systems, Vol 23, No 4, pp: 682-688, April 2012.

[6] C. J. C Burges. "Simplified support vector decision rules". In Proceedings of 13^{th} International Conference on Machine Learning, pp: 71-77, 1996.

[7] B. Schölkopf, S. Mika, C. J. C. Burges, P. Knirsch, K. Müller, G. Rätsch and A. J. Smola. "Input space verus feature space in kernel-based methods". IEEE Transactions on Neural Networks and Learning Systems, Vol 10, No 5, pp: 1000-1017, September 1999.

[8] T. Downs, K. E. Gates and A. Masters. "Exact Simplification of Support Vector Solutions". Journal of Machine Learning Research, Vol 2, pp: 293-297, December 2001.

[9] Y. J. Lee and O. L. Mangasarian. "RSVM: Reduced Support Vector Machines". In Proceedings of the 1^{st} SIAM International Conference on Data Mining, 2001.

[10] Y. J. Lee and O. L. Mangasarian. "SSVM: A smooth support vector machine for classification". Computational Optimization and Applications, Vol 20, No 1, pp: 5-22, 2001.

[11] J. A. K. Suykens, L. Lukas and J. Vandewalle. "Sparse approximation using Least Squares Support Vector Machines". In Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS 2000), pp: 757-760, 2000.

[12] Y. Li, C. Lin and W. Zhang. "Improved Sparse Least-Squares Support Vector Machine Classifiers". Neurocomputing, Vol 69, No 13, pp: 1655-1658, 2006.

[13] E. J. Nyström. "Über die praktische Auflösung von Integralgleichungen mit Anwendungen auf Randwertaufgaben". Acta Mathematica 54, pp: 185-204, 1930.

[14] C. T. H. Baker. "The Numerical Treatment of Integral Equations". Oxford Claredon Press, 1983.

[15] K. De Brabanter, J. De Brabanter, J. A. K. Suykens and Bart De Moor. "Optimized Fixed-Size Kernel Models for Large Data Sets". Computational Statistics & Data Analysis, Vol 54, No 6, pp: 1484-1504, 2010.

[16] 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 methods". Machine Learning, Special Issue on Model Selection and Optimization in Machine Learning, Vol 85, No 1, pp: 109-148, 2011

[17] J. Weston, A. Elisseeff, B. Schölkopf and M. Tipping. "Use of the Zero Norm with Linear Models and Kernel Methods". Journal of Machine Learning Research, Vol 3, pp: 1439-1461, 2003.

[18] E. J. Candes, M. B. Wakin and S. Boyd. "Enhancing Sparsity by Reweighted l1 Minimization". Journal of Fourier Analysis and Applications, Vol 14, No 5, pp: 877-905, special issue on sparsity, 2008.

[13] G. C. Cawley and N. L. C. Talbot. "Improved sparse least-squares support vector machines". Neurocomputing, Vol 48, No 1-4, pp: 1025-1031, 2002.

[19] K. Huang, D. Zheng, J. Sun, Y. Hotta, K. Fujimoto and S. Naoi. "Sparse Learning for Support Vector Classification". Pattern Recognition Letters, Vol 31, No 13, pp: 1944-1951, 2010.

[20] J. Lopez, K. De Brabanter, J. R. Dorronsoro and J. A. K. Suykens. "Sparse LSSVMs with L_0-norm minimization". In Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2011), pp: 189-194, Belgium, 2011.

[21] R. Mall and J. A. K. Suykens. "Sparse Variations to Fixed-Size Least Squares Support Vector Machines for Large Scale Data". In the The 17^{th} Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD2013). ftp://ftp.esat.kuleuven.be/SISTA/rmall/PAKDD_ms.pdf

[22] C. C. Chang and C. J. Lin. "LIBSVM : a library for support vector machines", ACM Transactions on Intelligent Systems and Technology, Vol 2, No 27, pp.1-27, 2011.

[23] S. S. Keerthi, O. Chapelle and D. DeCoste. "Building support vector machines with reduced classifier complexity". Journal of Machine Learning Research, Vol 7, pp: 1493-1515, July 2006.

[24] C. L. Blake and C. J. Merz. "UCI repository of machine learning databases". http://archive.ics.uci.edu/ml/datasets.html , Irvine, CA.

[25] R. Fletcher. Practical methods of Optimization. John Wiley & Sons, 1987.

[26] C. K. I. Williams and M. Seeger. "Using the Nyström method to speed up kernel machines". Advances in Neural Information Processing Systems, Vol 13, pp: 682-688, 2001.

[27] M. Rudemo. "Empirical choice of histograms and kernel density estimators". Scandinavian Journal of Statistics, Vol 9, pp: 65-78, 1982.

[28] L. Du, Y. Yang, D. He, R. G. Harley, T. G. Habetler and B. Lu. "Support Vector Machines Based Methods For Non-Intrusive Identification of Miscellaneous Electric Loads", 38^{th} Annual Conference of the IEEE Industrial Electronics Society (IECON 2012), October 25-28, Montreal, Quebec, Canada.

[29] A. W. Bowman. "An alternative method of cross-validation for smoothing of density estimators". Biometrika, Vol 71, pp: 353-360, 1984.

[30] D. W. Scott and G. R. Terrel. "Biased and unbiased cross-validation in density estimation". Journal of American Statistical Association, Vol 82, pp: 1131-1146, 1987.

[31] C. C. Taylor. "Bootstrap choice of the smoothing parameter in kernel density estimation". Biometrika, Vol 76, pp: 705-712, 1989.

[32] S. J. Sheather and M. C. Jones. "A reliable data-based bandwidth selection method for kernel density estimation". Journal of the Royal Statistical Society, Vol 53, pp: 683-690, 1991.

[33] D. W. Scott and S. R. Sain. "Multi-dimensional Density Estimation". Data Mining and Computational Statistics, Vol 23, pp: 229-263, 2004.

[34] S. Xavier de Souza, J. A. K. Suykens, J. Vandewalle and D. Bolle. "Coupled Simulated Annealing for Continuous Global Optimization". IEEE Transactions on Systems, Man, and Cybertics - Part B, Vol 40, No 2, pp: 320-335, 2010.

Johan A.K. Suykens received the degree in Electro-Mechanical Engineering and the Ph.D. degree in Applied Sciences from the KU Leuven, in 1989 and 1995, respectively. In 1996 he has been a Visiting Postdoctoral Researcher at the University of California, Berkeley. He has been a Postdoctoral Researcher with the Fund for Scientific Research FWO Flanders and is currently a Professor with KU Leuven. He is author of the books "Artificial Neural Networks for Modelling and Control of Non-linear Systems" (Kluwer Academic Publishers) and "Least Squares Support Vector Machines" (World Scientific), co-author of the book "Cellular Neural Networks, Multi-Scroll Chaos and Synchronization" (World Scientific) and editor of 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). He is a Senior IEEE member and has served as associate editor for the IEEE Transactions on Circuits and Systems (1997-1999 and 2004-2007) and for the IEEE Transactions on Neural Networks (1998-2009). He received an IEEE Signal Processing Society 1999 Best Paper Award and several Best Paper Awards at International Conferences. He is a recipient of the International Neural Networks Society INNS 2000 Young Investigator Award for significant contributions in the field of neural networks. He has been awarded an ERC Advanced Grant 2011.

Raghvendra Mall was born in Kolkata, India on March 5th, 1988. He received his BTech degree in 2010 and MS by Research in Computer Science in 2011 from IIIT-Hyderabad, India. He is started his doctorate at ESAT-STADIUS, KU Leuven, Leuven, Belgium in 2012 under the guidance of Prof. Johan A.K. Suykens. 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.