Sparse Reductions for Fixed-Size Least Squares Support Vector Machines on Large Scale Data

Raghvendra Mall[1] and J.A.K. Suykens[1]

Department of Electral Engineering, ESAT-SCD, Katholieke Universiteit Leuven, Kasteelpark Arenberg,10 B-3001 Leuven, Belgium

Abstract. Fixed-Size Least Squares Support Vector Machines (FS-LS-SVM) is a powerful tool for solving large scale classification and regression problems. FS-LSSVM solves an over-determined system of M linear equations by using Nyström approximations on a set of prototype vectors (PVs) in the primal. This introduces sparsity in the model along with ability to scale for large datasets. But there exists no formal method for selection of the right value of M. In this paper, we investigate the sparsity-error trade-off by introducing a second level of sparsity after performing one iteration of FS-LSSVM. This helps to overcome the problem of selecting a right number of initial PVs as the final model is highly sparse and dependent on only a few appropriately selected prototype vectors (SV) is a subset of the PVs. The first proposed method performs an iterative approximation of L_0-norm which acts as a regularizer. The second method belongs to the category of threshold methods, where we set a window and select the SV set from correctly classified PVs closer and farther from the decision boundaries in the case of classification. For regression, we obtain the SV set by selecting the PVs with least minimum squared error (mse). Experiments on real world datasets from the UCI repository illustrate that highly sparse models are obtained without significant trade-off in error estimations scalable to large scale datasets.

1 Introduction

LSSVM [3] and SVM [4] are state of the art learning algorithms in classification and regression. The SVM model has inherent sparsity whereas the LSSVM model lacks sparsity. However, previous works like [1],[5],[6] address the problem of sparsity for LSSVM. One such approach was introduced in [7] and uses a *fixed-size least squares support vector machines*. The major benefit which we obtain from FS-LSSVM is its applicability to large scale datasets. It provides a solution to the LSSVM problem in the primal space resulting in a parametric model and sparse representation. The method uses an explicit expression for the feature map using the Nyström method [8],[9] as proposed in [16]. In [7], the authors obtain the initial set of prototype vectors (PVs) i.e. M vectors while maximizing the quadratic Rènyi entropy criterion leading to a sparse representation in the primal space. The error of FS-LSSVM model approximates to that of LSSVM for $M \ll N$. But this is not the sparsest solution and selecting an initial value of M is an existent problem. In [11], they try to overcome this problem by iteratively

building up a set of conjugate vectors of increasing cardinality to approximately solve the over-determined FS-LSSVM linear system. But if few iterations don't suffice to result in a good approximation then the cardinality will be M.

The L_0-norm counts the number of non-zero elements of a vector. It results in very sparse models by returning models of low complexity and acts as a regularizer. However, obtaining this L_0-norm is an NP-hard problem. Several approximations to it are discussed in [12]. In this paper, we modify the iterative sparsifying procedure introduced in [13] and used for LSSVM the technique as shown in [14] and reformulate it for the FS-LSSVM. We apply this formulation on FS-LSSVM because for large scale datasets like Magic Gamma, Adult and Slice Localization we are overwhelmed with memory $O(N^2)$ and computational time $O(N^3)$ constraints when applying the L_0-norm scheme directly on LSSVM [14] or SVM [13]. The second proposed method performs an iteration of FS-LSSVM and then based on a user-defined window selects a subset of PVs as SV. For classification, the selected vectors satisfy the property of being correctly classified and are either closer or farther from the decision boundary since they well determine the extent of the classes. But for regression, the SV set comprises those PVs which have the least mse and are best-fitted by the regressor. Once the SV set is determined we re-perform FS-LSSVM resulting in highly sparse models without significant trade-off in accuracy and scalable to large scale datasets.

The contribution of this work involves providing smaller solutions which use $M' < M$ PVs for FS-LSSVM, obtaining highly sparse models with guarantees of low complexity (L_0-norm of $\tilde{w}$) and overcoming the problem of memory and computational constraints faced by L_0-norm based approaches for LSSVM and SVM on large scale datasets. Sparseness enables exploiting memory and computationally efficiency, e.g. matrix multiplications and inversions. The solutions that we propose utilize the best of both FS-LSSVM and sparsity inducing measures on LSSVM and SVM resulting in highly sparse and scalable models. Table 1 provides a conceptual overview of LSSVM, FS-LSSVM and proposes Reduced FS-LSSVM along with the notations used in the rest of the paper. Figures 1 and 2 illustrate our proposed approaches on the Ripley and Motorcycle dataset.

	LSSVM		**FS-LSSVM**		**Reduced FS-LSSVM**
SV/Train	N/N		M/N		M'/M
Primal	w		$\tilde{w}$		$\boxed{w'}$
	$\phi(\cdot) \in \mathbb{R}^{N_h}$	Step 1	$\tilde{\phi}(\cdot) \in \mathbb{R}^{M}$	Step 2	$\boxed{\phi'(\cdot) \in \mathbb{R}^{M'}}$
Dual	α	$\rightarrow$	$\boxed{\tilde{\alpha}}$	$\rightarrow$	α'
	$K \in \mathbb{R}^{N \times N}$		$\boxed{\tilde{K} \in \mathbb{R}^{M \times M}}$		$K' \in \mathbb{R}^{M' \times M'}$

Table 1. For Reduced FS-LSSVM, we first perform FS-LSSVM in the Primal. Then a sparsifying procedure is performed in the Dual of FS-LSSVM as highlighted in the box in the middle resulting in a reduced SV set. Then FS-LSSVM is re-performed in the Primal as highlighted by the box on the right. We propose two sparsifying procedures namely L_0 reduced FS-LSSVM and Window reduced FS-LSSVM.

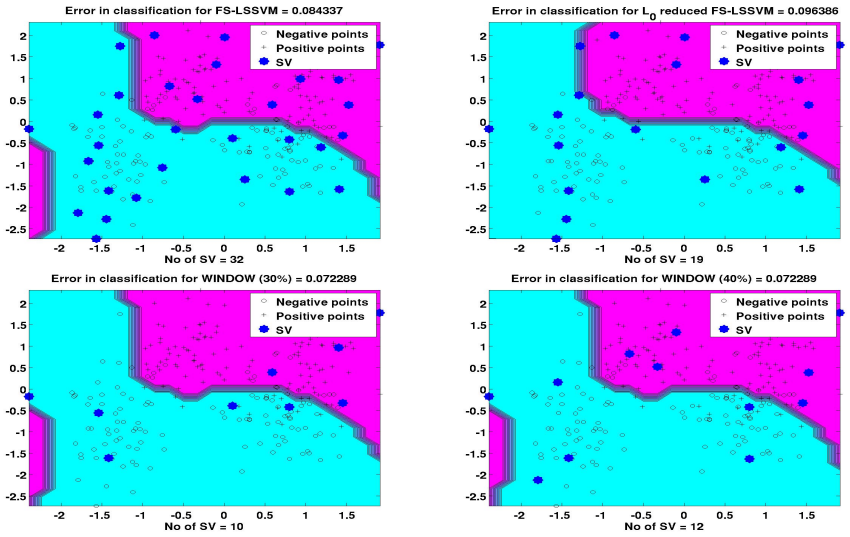

Fig. 1. Comparison of the best randomization result out of 10 randomizations for the proposed methods with FS-LSSVM for Ripley Classification data

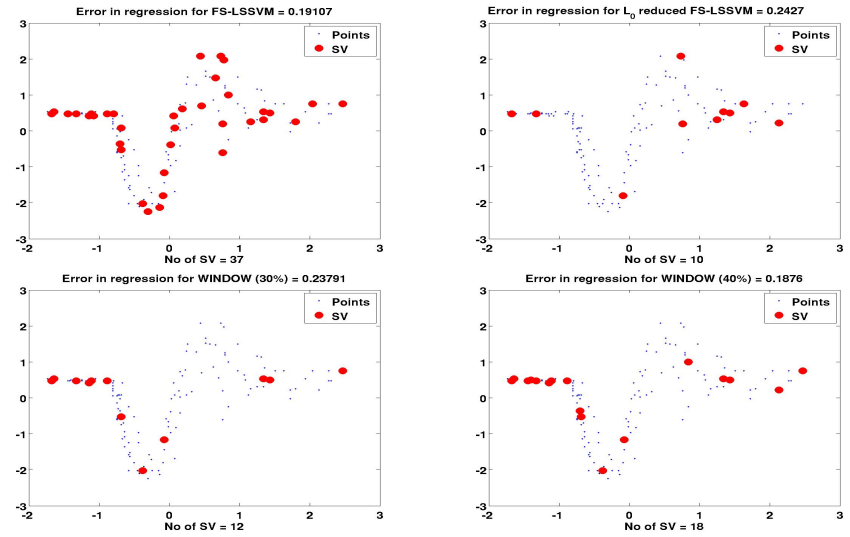

Fig. 2. Comparison of the best randomization result out of 10 randomizations for the proposed methods with FS-LSSVM for Motorcycle Regression data

2 L_0 reduced FS-LSSVM

Algorithm 1 gives a brief summary of the FS-LSSVM method. We first propose an approach using the L_0-norm to reduce $\tilde{w}$ and acting as a regularizer in the objective function. It tries to estimate the optimal subset of PVs leading to sparse

Algorithm 1: Fixed-Size LSSVM method

Data: $\mathbb{D}_n = \{(x_i, y_i) : x_i \in \mathbb{R}^d, y_i \in \{+1, -1\}$ for classification & $y_i \in \mathbb{R}$ for regression, $i = 1, \ldots, N\}$.
Determine the kernel bandwidth using the multivariate rule-of-thumb [17].
Given the number of PVs, perform prototype vector selection using quadratic Rènyi entropy criterion.
Determine the tuning parameters σ and γ performing fast v-fold cross validation as described in [10].
Given the optimal tuning parameters, get FS-LSSVM parameters $\tilde{w}$ & $\tilde{b}$.

solutions. For our formulation, the objective function is to minimize the error estimations of these prototype vectors regulated by L_0-norm of $\tilde{w}$. We modify the procedure described in [13], [14] and consider the following generalized primal problem:

$$\begin{aligned} \min_{\tilde{\alpha},\tilde{b},\tilde{e}} \quad & J(\tilde{\alpha},\tilde{e}) = \frac{1}{2}\sum_j \lambda_j \tilde{\alpha}_j^2 + \frac{\gamma}{2}\sum_i \tilde{e}_i^2 \\ \text{s.t.} \quad & \sum_j \tilde{\alpha}_j \tilde{K}_{ij} + \tilde{b} = y_i - \tilde{e}_i,\ i = 1, \ldots, M \end{aligned} \tag{1}$$

where $\tilde{w} \in \mathbb{R}^M$ and can be written as $\tilde{w} = \sum_j \tilde{\alpha}_j \tilde{\phi}(x_j)$. The regularization term is now not on $\| \tilde{w} \|^2$ but on $\| \tilde{\alpha} \|^2$. The regularization weights are given by the prefix λ_j coefficients. This formulation is similar to [14] with the difference that it is made applicable here to large scale datasets. These $\tilde{\alpha}_j$ are coefficients of linear combination of the features which result in $\tilde{w}$ vector. The set of PVs is represented as $\mathcal{S}_{PV}$. The $\tilde{e_i}$ are error estimates and are determined only for the vectors belonging to the set $\mathcal{S}_{PV}$. Thus, the training set comprises of the vectors belonging to the set $\mathcal{S}_{PV}$.

Introducing the coefficient β for Lagrangian $\mathcal{L}$ one obtains: $\partial\mathcal{L}/\partial\tilde{\alpha}_j = 0 \Rightarrow \tilde{\alpha}_j = \sum_j \beta_i K_{ij}/\lambda_j$, $\partial\mathcal{L}/\partial\tilde{b} = 0 \Rightarrow \sum_i \beta_i = 0$, $\partial\mathcal{L}/\partial\tilde{e}_i = 0 \Rightarrow \beta_i = \gamma\tilde{e}_i$, $\partial\mathcal{L}/\partial\beta_i = 0$ $\Rightarrow \sum_j \tilde{\alpha}_j K_{ij} + \tilde{b} = y_i - \tilde{e}_i$, $\forall i$.Combining the conditions $\partial\mathcal{L}/\partial\tilde{\alpha}_i = 0$, $\partial\mathcal{L}/\partial\tilde{e}_i = 0$ and $\partial\mathcal{L}/\partial\beta_i = 0$ and after little algebraic manipulation yields $\sum_k \beta_k H_{ik} + \tilde{b} = y_i$, with $H = \tilde{K}diag(\lambda)^{-1}\tilde{K} + I_M/\gamma$ and $\tilde{K}$ is a kernel matrix. The kernel matrix $\tilde{K}$ is defined as $\tilde{K}_{ij} = \tilde{\phi}(x_i)^\intercal \tilde{\phi}(x_j)$ where $x_i \in \mathcal{S}_{PV}, x_j \in \mathcal{S}_{PV}$, $\tilde{\phi}(x_i) \in \mathbb{R}^M$ and $H \in \mathbb{R}^{M \times M}$.

This, together with $\partial\mathcal{L}/\partial\tilde{b} = 0$, results in the linear system

$$\left[\begin{array}{c|c} 0 & 1_k^\intercal \\ \hline 1_k & H \end{array}\right] \left[\begin{array}{c} \tilde{b} \\ \hline \beta \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline y \end{array}\right] \tag{2}$$

The procedure to obtain sparseness involves iteratively solving the system (2) for different values of λ and is described in Algorithm 2. Considering the t^{th}

iteration, we can build the matrix $H^t = \tilde{K} diag(\lambda^t)^{-1} \tilde{K} + I_M/\gamma$ and solve the system of linear equations to obtain the value of β^t and $\tilde{b}^t$. From this solution we get $\tilde{\alpha}^{t+1}$ and most of its element tend to zero, the $diag(\lambda^{t+1})^{-1}$ will end up having many zeros along the diagonal due to the values allocated to λ^{t+1}. It was shown in [13] that as $t \to \infty$, $\tilde{\alpha}^t$ converges to a stationary point $\tilde{\alpha}^*$ and this model is guaranteed to be sparse and result in set SV. This iterative sparsifying procedure converges to a local minimum as the L_0-norm problem is NP-hard. Since this $\tilde{\alpha}^*$ depends on the initial choice of weights, we set them to the FS-LSSVM solution $\tilde{w}$, so as to avoid ending up in different local minimal solutions.

Algorithm 2: L_0 reduced FS-LSSVM method

Data: Solve FS-LSSVM to obtain initial $\tilde{w}$ & $\tilde{b}$
$\tilde{\alpha} = \tilde{w}(1:M)$
$\lambda_i \leftarrow \tilde{\alpha}_i, i = 1, \ldots, M$
while *convergence* **do**
  $H \leftarrow \tilde{K} diag(\lambda)^{-1} \tilde{K} + I_M/\gamma$
  Solve system (2) to give β and $\tilde{b}$
  $\tilde{\alpha} \leftarrow diag(\lambda)^{-1} \tilde{K} \beta$
  $\lambda_i \leftarrow 1/\tilde{\alpha}_i^2, i = 1, \ldots, M'$
Result: *indices* = find($|\tilde{\alpha}_i| > 0$)

The *convergence* of Algorithm 2 is assumed when the $\| \tilde{\alpha}^t - \tilde{\alpha}^{t+1} \| /M'$ is lower than 10^{-4} or when the number of iterations t exceeds 50. The result of the approach is the *indices* of those PVs for which $|\tilde{\alpha}_i| > 10^{-6}$. These *indices* provide the set of most appropriate prototype vectors (SV). The FS-LSSVM method (Algorithm 1) is re-perfomed using only this set SV. We are training only on the set $\mathcal{C}_{PV}$ and not on the entire training data because the H matrix becomes $N \times N$ matrix which cannot in memory for large scale datasets.

The time complexity of the proposed methods is bounded by solving the linear system of equations (2). An interesting observation is that the H matrix becomes sparser after each iteration. This is due to the fact that $diag(\lambda)^{-1} = diag(\tilde{\alpha}_1^2, \ldots, \tilde{\alpha}_M^2)$ and most of these $\tilde{\alpha}_i \to 0$. Thus the H matrix becomes sparser in each iteration such that after some iterations inverting H matrix is equivalent to inverting each element of the H matrix. The computation time is dominated by matrix multiplication to construct the H matrix. The H matrix construction can be formulated as multiplications of two matrices i.e. $P = \tilde{K} diag(\lambda)^{-1}$ and $H = P\tilde{K}$. The P matrix will become sparser as it multiplies the $\tilde{K}$ matrix with $diag(\lambda)^{-1}$. Let $\tilde{M}$ be the number of columns in P matrix with elements $\neq 0$. This number i.e. $\tilde{M}$ can be much less than M. Thus, for the L_0 reduced FS-LSSVM the time required for the sparsifying procedure is given by $\mathcal{O}(M^2\tilde{M})$ and the average memory requirement is $\mathcal{O}(M^2)$.

3 Window reduced FS-LSSVM

In [5], it was proposed to remove the support vectors with smaller $|\alpha_i|$ for the LSSVM method. But this approach doesn't greatly reduce the number of support vectors. In [6], the authors proposed to remove support vectors with the larger $y_i f(x_i)$ as they are farther from decision boundary and easiest to classify. But these support vectors are important as they determine the true extent of a class.

We propose window based SV selection method for both classification and regression. For classification, we select the vectors which are correctly classified and closer and farther from the decision boundary. An initial FS-LSSVM method determines the $\tilde{y}$ for the PVs. To find the reduced set SV, we first remove the prototype vectors which were misclassified ($\tilde{y} \neq y$) as shown in [2]. Then, we sort the estimated $f(x_j)$, $\forall j \in CorrectPV$ where $CorrectPV$ is the set of correctly classified PVs to obtain a sorted vector S. This sorted vector is divided into two halves one containing the sorted positive estimations ($\tilde{y}$) corresponding to positive class and the other containing sorted negative values ($\tilde{y}$) corresponding to negative class. The points closer to the decision boundary have smaller positive and smaller negative estimations ($|\tilde{y}|$) and the points farther from the decision boundary have the larger positive and larger negative estimations ($|\tilde{y}|$) as depicted in Figure 1. So these vectors corresponding to these estimations are selected. Selecting correctly classified vectors closer to decision boundary prevents over-fitting and selecting vectors farther from the decision boundary helps to identify the extent of the classes.

For regression, we select the prototype vectors which have least mse after one iteration of FS-LSSVM. We estimate the squared errors for the PVs and out of these prototype vectors select those vectors which have the least mse to form the set SV. They are the most appropriately estimated vectors as they have the least error and so are most helpful in estimating a generalization of the regression function. By selection of prototype vectors which have least mse we prevent selection of outliers as depicted in Figure 2. Finally, a FS-LSSVM regression is re-performed on this set SV.

The percentage of vectors selected from the initial set of prototype vectors is determined by the window. We experimented with various window size i.e. $(30, 40, 50)$ percent of the initial prototype vectors (PVs). For classification, we selected half of the window from the positive class and the other half from the negative class. In case the classes are not balanced and number of PVs in one class is less than half the window size then all the correctly classified vectors from those PVs are selected. In this case, we observe that the number of selected prototype vectors (SV) can be less than window size. The methodology to perform this Window reduced FS-LSSVM is presented in Algorithm 3.

This method result in better generalization error with smaller variance achieving sparsity. The trade-off with estimated error is not significant and in several cases it leads to better results as will be shown in the experimental results. As we increase the window size the variation in estimated error decreases and estimated error also decreases until the median of the estimated error becomes nearly constant as is depicted in Figure 3.

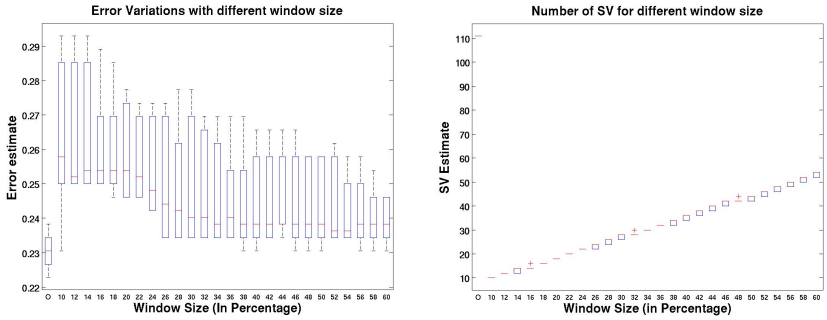

Fig. 3. Trends in error & SV with increasing window size for Diabetes dataset compared with the FS-LSSVM method represented here as 'O'

Algorithm 3: Window reduced FS-LSSVM

Data: $\mathbb{D}_n = \{(x_i, y_i) : x_i \in \mathbb{R}^d, y_i \in \{+1, -1\}$ for $Classification$ & $y_i \in \mathbb{R}$ for $Regression$, $i = 1, \ldots, N\}$.

Perform FS-LSSVM using the initial set of PVs of size M on training data.

if *Classification* **then**

$CorrectPV$ = Remove misclassified prototype vectors
$S = sort(f(x_i))\ \forall i \in CorrectPV$;
$A = S(:) > 0$; $B = S(:) < 0$;
$begin = windowsize/4$;
$endA = size(A) - windowsize/4$;
$endB = size(B) - windowsize/4$;
SV = $[A[begin\ \ endA]; B[begin\ \ endB]]$;

if *Regression* **then**

Estimate the squared error for the initially selected PVs
SV = Select the PVs with least mean squared error

Re-perform the FS-LSSVM method using the reduced set SV of size M' on training data

4 Computational Complexity and Experimental Results

4.1 Computational Complexity

The computation time of FS-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 $[\tilde{\phi}(x_1), \tilde{\phi}(x_2), \ldots, \tilde{\phi}(x_n)]^\intercal [\tilde{\phi}(x_1), \tilde{\phi}(x_2), \ldots, \tilde{\phi}(x_n)]$.

The computation time is $\mathcal{O}(NM^2)$ where N is dataset size as shown in [10]. We already presented the computation time for the iterative sparsifying procedure

for L_0 reduced FS-LSSVM. For this approach, the computation time $\mathcal{O}(M^3)$. So, it doesn't have an impact on the overall computational complexity as we will observe from the experimental results. In our experiments, we selected $M = \lceil k \times \sqrt{N} \rceil$ where $k \in \mathbb{N}$, the complexity of L_0 reduced FS-LSSVM can be re-written as $\mathcal{O}(k^2N^2)$. We experimented with various values of k and observed that after certain values of k, the change in estimated error becomes nearly irrelevant. In our experiments, we choose the value of k corresponding to the first instance after which the change in error estimations becomes negligible.

For the window based method, we have to run the FS-LSSVM once and based on window size obtain the set SV which is always less than PVs i.e. $M' \leq M$. The time-complexity for re-performing the FS-LSSVM on the set SV is $\mathcal{O}(M'^2N)$ where N is the size of the dataset. The overall time complexity of the approach is $\mathcal{O}(M^2N)$ required for Nyström approximation and the average memory requirement is $\mathcal{O}(NM)$.

4.2 Dataset Description

All the datasets on which the experiments were conducted are from UCI benchmark repository [15]. For classification, we experimented with Ripley (RIP), Breast-Cancer (BC), Diabetes (DIB), Spambase (SPAM), Magic Gamma (MGT) and Adult (ADU). The corresponding dataset size are 250,682,768,4061,19020,48-842 respectively. The corresponding k values for determining the initial number of prototype vector are 2,6,4,3,3 and 3 respectively. The datasets Motorcycle, Boston Housing, Concrete and Slice Localization are used for regression whose size is $111, 506, 1030, 53500$ and their k values are 6,5,6 and 3 respectively.

4.3 Experiments

All the 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 kernel matrix construction in all cases. As a pre-processing step, all records containing unknown values are removed from consideration. Input values have been normalized. We compare the performance of our proposed approaches with the normal FS-LSSVM classifier/regressor, L_0 LSSVM [14], SVM and ν-SVM. The last two methods are implemented in the LIBSVM software with default parameters. All methods use a cache size of 8 GB. Shrinking is applied in the SVM case. All comparisons are made on 10 randomizations of the methods.

The comparison is performed on an out-of-sample test set consisting of 1/3 of the data. The first 2/3 of the data is reserved for training and cross-validation. The tuning parameters σ and γ for the proposed FS-LSSVM methods and SVM methods are obtained by first determining good initial starting values using the method of coupled simulated annealing (CSA) in [18]. After that a derivative-free simplex search is performed. This extra step is a fine tuning procedure resulting in more optimal tuning parameters and better performance.

Table 2 provides a comparison of the mean estimated error $\pm$ its deviation, mean number of selected prototype vectors SV and a comparison of the

Test Classification(Error and Mean SV)												
	RIP		BC		DIB		SPAM		MGT		ADU	
Algorithm	Error	SV	Error	SV	Error	SV	Error	SV	Error	SV	Error	SV
FS-LSSVM	**0.0924 ± 0.01**	32	0.047 ± 0.004	155	0.233 ± 0.002	111	0.077 ± 0.002	204	**0.1361 ± 0.001**	414	0.146 ± 0.0003	664
L_0 LSSVM	0.183 ± 0.1	7	0.04 ± 0.011	17	0.248 ± 0.034	10	**0.0726 ± 0.005**	340	-	-	-	-
C-SVC	0.153 ± 0.04	81	0.054 ± 0.07	318	0.249 ± 0.0343	290	0.074 ± 0.0007	800	0.144 ± 0.0146	7000	0.1519(∗)	11,085
ν-SVC	0.1422 ± 0.03	86	0.061 ± 0.05	330	0.242 ± 0.0078	331	0.113 ± 0.0007	1525	0.156 ± 0.0142	7252	0.161(∗)	12,205
L_0 reduced FS-LSSVM	0.1044 ± 0.0085	17	**0.0304 ± 0.009**	19	0.239 ± 0.03	36	0.1144 ± 0.053	144	0.1469 ± 0.004	115	0.1519 ± 0.002	78
Window (30%)	0.095 ± 0.009	10	**0.0441 ± 0.0036**	46	**0.2234 ± 0.006**	26	0.1020 ± 0.002	60	0.1522 ± 0.002	90	**0.1449 ± 0.003**	154
Window (40%)	**0.091 ± 0.015**	12	**0.0432 ± 0.0028**	59	**0.2258 ± 0.008**	36	0.0967 ± 0.002	78	0.1481 ± 0.0013	110	**0.1447 ± 0.003**	183
Window (50%)	**0.091 ± 0.009**	16	**0.0432 ± 0.0019**	66	**0.2242 ± 0.005**	45	0.0955 ± 0.002	98	0.1468 ± 0.0007	130	**0.1446 ± 0.003**	213

Test Regression(Error and Mean SV)								
	Motorcycle		Boston Housing		Concrete		Slice Localization	
Algorithm	Error	SV	Error	SV	Error	SV	Error	SV
FS-LSSVM	**0.2027 ± 0.007**	37	**0.1182 ± 0.001**	112	**0.1187 ± 0.006**	192	**0.053 ± 0.0**	694
L_0 LSSVM	0.2537 ± 0.0723	9	0.16 ± 0.05	65	0.165 ± 0.07	215	-	-
ϵ-SVR	0.2571 ± 0.0743	69	0.158 ± 0.037	226	0.23 ± 0.02	670	0.1017(*)	13,012
ν-SVR	0.2427 ± 0.04	51	0.16 ± 0.04	226	0.22 ± 0.02	330	0.0921(*)	12,524
L_0 reduced FS-LSSVM	0.62 ± 0.29	5	0.1775 ± 0.057	48	0.2286 ± 0.03	43	0.1042 ± 0.06	495
Window (30%)	0.32 ± 0.03	12	0.1465 ± 0.02	34	0.2 ± 0.005	58	0.1313 ± 0.0008	208
Window (40%)	0.23 ± 0.03	15	0.1406 ± 0.006	46	0.1731 ± 0.006	78	0.116 ± 0.0005	278
Window (50%)	0.23 ± 0.02	21	**0.1297 ± 0.009**	56	0.1681 ± 0.0039	96	0.1044 ± 0.0006	347

Train & Test Classification(Computation Time)						
Algorithm	RIP	BC	DIB	SPAM	MGT	ADU
FS-LSSVM	4.346 ± 0.143	28.2 ± 0.691	15.02 ± 0.688	181.1 ± 7.75	3105.6 ± 68.6	23,565 ± 1748
L_0 LSSVM	5.12 ± 0.9	28.2 ± 5.4	39.1 ± 5.9	950 ± 78.5	-	-
C-SVC	13.7 ± 4	43.36 ± 3.37	24.8 ± 3.1	1010 ± 785	20,603 ± 396	139,730(*)
ν-SVC	13.6 ± 4.5	41.67 ± 2.436	23.23 ± 2.3	785 ± 22	35,299 ± 357	139,927(*)
L_0 reduced FS-LSSVM	4.349 ± 0.28	28.472 ± 1.07	15.381 ± 0.5	193.374 ± 8.35	3185.7 ± 100.2	22,485 ± 650
Window (30%)	4.28 ± 0.22	28.645 ± 1.45	14.5 ± 0.5	171.78 ± 4.82	3082.9 ± 48	22,734 ± 1520
Window (40%)	4.33 ± 0.27	27.958 ± 1.21	14.6 ± 0.75	172.62 ± 5.3	3052 ± 42	22,466 ± 1705
Window (50%)	4.36 ± 0.27	28.461 ± 1.3	14.255 ± 0.5	167.39 ± 2.31	3129 ± 84	22,621 ± 2040

Train & Test Regression(Computational Time)				
Algorithm	Motorcycle	Boston Housing	Concrete	Slice Localization
FS-LSSVM	3.42 ± 0.12	14.6 ± 0.27	55.43 ± 0.83	37,152 ± 1047
L_0 LSSVM	10.6 ± 1.75	158.86 ± 3.2	753 ± 35.5	-
ϵ-SVR	24.6 ± 4.75	63 ± 1	55.43 ± 0.83	252,438(*)
ν-SVR	25.5 ± 3.9	61 ± 1	168 ± 3	242,724(*)
L_0 reduced FS-LSSVM	3.426 ± 0.18	15.4 ± 0.35	131 ± 2	36,547 ± 1992
Window (30%)	3.52 ± 0.244	15.04 ± 0.17	56.75 ± 1.35	36,087 ± 1066
Window (40%)	3.425 ± 0.153	15.04 ± 0.27	55.67 ± 0.81	35,906 ± 1082
Window (50%)	3.5 ± 0.226	15.04 ± 0.15	55.06 ± 1.088	36,183 ± 1825

Table 2. Comparison of performance of different methods for UCI repository datasets. The '*' represents no cross-validation and performance on fixed tuning parameters due to computational burden and '-' represents cannot run due to memory constraints.

mean computation time ± its deviation for 10 randomizations of the proposed approaches with FS-LSSVM and SVM methods for various classification and regression data sets. Figure 4 represents the estimated error, run time and variations in number of selected prototype vectors for Adult (ADU) and Slice Localization (SL) datasets respectively.

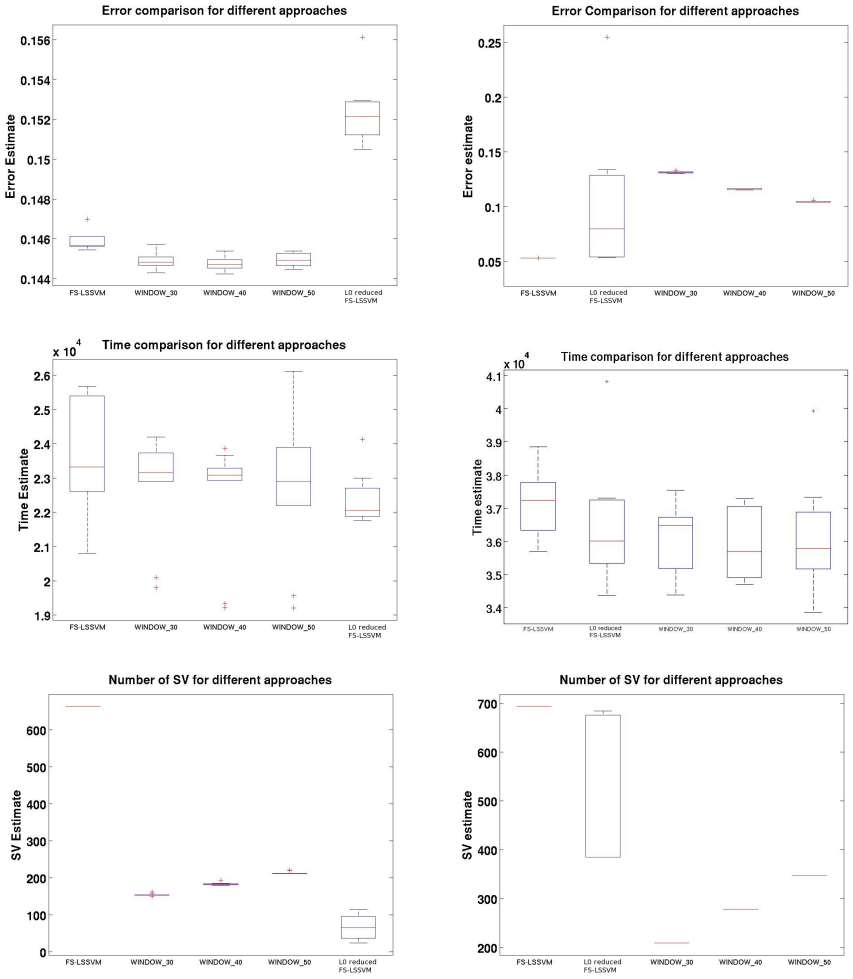

Fig. 4. Comparison of performance of proposed approaches with FS-LSSVM method for Adult & Slice Localization datasets

4.4 Performance Analysis

The proposed approaches i.e L_0 reduced FS-LSSVM and Window reduced FS-LSSVM method introduce more sparsity in comparison to FS-LSSVM and SVM

methods without significant trade-off for classification. For smaller datasets L_0 LSSVM produces extremely few support vectors but for datasets like SPAM, Boston Housing and Concrete it produces more support vectors. For some datasets like breast-cancer and diabetes, it can be seen from Table 2 that proposed approaches results in better error estimations than other methods with much smaller set SV. For datasets like SPAM and MGT, the trade-off in error is not significant considering the reduction in number of PVs (only 78 prototype vectors required by L_0 reduced FS-LSSVM for classifying nearly 20,000 points). From Figure 4, we observe the performance for Adult dataset. The window based methods result in lower error estimate using fewer but more appropriate SV. Thus the idea of selecting correctly classified points closer and farther from the decision boundary results in better determining the extent of the classes. These sparse solutions typically lead to better generalization of the classes. The mean time complexity for different randomizations is nearly the same.

For regression, as we are trying to estimate a continuous function, if we greatly reduce the PVs to estimate that function, then the estimated error would be higher. For datasets like boston housing and concrete, the estimated error by the proposed methods is more than FS-LSSVM method but the amount of sparsity introduced is quite significant. These methods result in reduced but more generalized regressor functions and the variation in the estimated error of window based approach is lesser as in comparison to L_0-norm based method. This is because in each randomization, the L_0-norm reduces to different number of prototype vectors whereas the reduced number of prototype vectors (SV) for window based method is fixed and is uninfluenced by variations caused by outliers as the SV have least mse. This can be observed for the Slice Localization dataset in Figure 4. For this dataset, L_0 reduced FS-LSSVM estimates lower error than window approach. This is because for this dense dataset, the L_0-norm based FS-LSSVM requires more SV (495) than window based method $(208, 278, 347)$ which signifies more vectors are required for better error estimation. The proposed models are of magnitude $(2 - 10)$x sparser than the FS-LSSVM method.

5 Conclusion

In this paper, we proposed two sparse reductions to FS-LSSVM namely L_0 reduced and Window reduced FS-LSSVM. These methods are highly suitable for mining large scale datasets overcoming the problems faced by L_0 LSSVM [14] and FS-LSSVM. We developed the L_0 reduced FS-LSSVM based on iteratively sparsifying L_0-norm training on the initial set of PVs. We also introduced a Window reduced FS-LSSVM trying to better determine the underlying structure of model by selection of more appropriate prototype vectors (SV). The resulting approaches are compared with normal FS-LSSVM, L_0 LSSVM and two kinds of SVM (C-SVM and ν-SVM from LIBSVM software) with promising performances and timing results using smaller and sparser models.

Acknowledgements This work was supported by Research Council KUL, ERC

AdG A-DATADRIVE-B, GOA/10/09MaNet, CoE EF/05/006, FWO G.0588.09, G.0377.12, SBO POM, IUAP P6/04 DYSCO, COST intelliCIS.

References

[1] Hoegaerts, L., Suykens, J.A.K., Vandewalle, J., De Moor B.: A comparison of pruning algorithms for sparse least squares support vector machines. In Proceedings of the 11^{th} International Conference on Neural Information Processings (ICONIP 2004), vol 3316,1247–1253, 2004.

[2] Geebelen, D., Suykens J.A.K., Vandewalle J.: Reducing the Number of Support Vectors of SVM classifiers using the Smoothed Seperable Case Approximation. IEEE Transactions on Neural Networks and Learning Systems,23(4), 682–688, 2012

[3] Suykens, J.A.K., Vandewalle J.: Least Squares Support Vector Machine Classifiers. Neural Processing Letters, vol 9(3), 293–300, 1999.

[4] Vapnik V.N.: The Nature of Statistical Learning Theory. Springer-Verlag, 1995.

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

[6] Li Y., Lin C., Zhang W.: Improved Sparse Least-Squares Support Vector Machine Classifiers. Neurocomputing, vol 69(13), 1655–1658, 2006

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

[8] Nyström E.J: Über die praktische Auflösung von Integralgleichungen mit Anwendungen auf Randwertaufgaben. Acta Mathematica, vol 54, 185–204, 1930.

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

[10] De Brabanter K., De Brabanter J., Suykens J.A.K., De Moor B: Optimized Fixed-Size Kernel Models for Large Data Sets. Computational Statistics & Data Analysis, vol 54(6), 1484–1504, 2010.

[11] Karsmakers P., Pelckmans K., De Brabanter K., Hamme H.V., Suykens J.A.K.: 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(1), 109–148, 2011.

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

[13] Huang, K., Zheng D., Sun J. et al: Sparse Learning for Support Vector Classification. Pattern Recognition Letters, 31(13), 1944–1951, 2010.

[14] Lopez J., De Brabanter K., Dorronsoro J.R., Suykens J.A.K.: Sparse LSSVMs with L_0-norm minimization. ESANN 2011, 189–194, 2011.

[15] Blake C.L, Merz C.J.: UCI repository of machine learning databases. `http://archive.ics.uci.edu/ml/datasets.html`, Irvine, CA, 1998.

[16] Williams C.K.I., Seeger M.: Using the Nyström method to speed up kernel machines. Advances in Neural Information Processing Systems, vol 13, 682–688, 2001.

[17] Scott D.W., Sain S.R.: Multi-dimensional Density Estimation. Data Mining and Computational Statistics, vol 23, 229–263, 2004.

[18] Xavier de Souza S., Suykens J.A.K., Vandewalle J., Bolle D.: Coupled Simulated Annealing for Continuous Global Optimization. IEEE Transactions on Systems, Man, and Cybertics - Part B, vol 40(2), 320–335, 2010.