# Sparse, Robust Least Squares Support Vector Machines for the Classification of noisy datasets

Iwein Vranckx[a,*], Joachim Schreurs[b], Bart De Ketelaere[c], Mia Hubert[a], Johan Suykens[b]

[a] *KU Leuven, Department of Mathematics and LStat, Celestijnenlaan 200B, BE-3001 Heverlee, Belgium*
[b] *KU Leuven, ESAT-STADIUS, Kasteelpark Arenberg 10, BE-3001 Heverlee, Belgium*
[c] *KU Leuven, Division of Mechatronics, Biostatistics and Sensors, Kasteelpark Arenberg 30, BE-3001 Heverlee, Belgium*

**Abstract**

(Abstract uit weighted LS-SVM, ter voorbeeld). Least squares support vector machines (LS-SVM) is an SVM version which involves equality instead of inequality constraints and works with a least squares cost function. In this way, the solution follows from a linear Karush–Kuhn–Tucker system instead of a quadratic programming problem. However, sparseness is lost in the LS-SVM case and the estimation of the support values is only optimal in the case of a Gaussian distribution of the error variables. In this paper, we discuss a method which can overcome these two drawbacks. We show how to obtain robust estimates for regression by applying a weighted version of LS-SVM. We also discuss a sparse approximation procedure for weighted and unweighted LS-SVM. It is basically a pruning method which is able to do pruning based upon the physical meaning of the sorted support values....

The methods of this paper are illustrated for RBF kernels and demonstrate how to obtain robust estimates with selection of an appropriate number of hidden units, in the case of outliers or non-Gaussian error distributions with heavy tails

*Keywords:* Robust support vector machines, Non-linear outlier detection, Support vector pruning, Sparse LS-SVM

*Corresponding author
*URL:* `wis.kuleuven.be/stat/robust` (Iwein Vranckx),
`iwein.vranckx@kuleuven.be` (Iwein Vranckx)

## 1. Introduction

The least-squares support vector machines (LS-SVM) is powerful method for solving pattern recognition and regression problems [12]. The LS-SVM maps the data to a higher dimensional space in which one constructs an optimal separating hyperplane. It has been applied to many real-world problems such as optimal control [13], financial time series prediction [14], system identification [6], electrical load forecasting [4] and many others. However the LS-SVM has two main disadvantages. It is sensitive to outliers which have large support values resulting from the solution to the linear system. The second disadvantage is that the solution lacks sparseness, which is essential for real-time prediction with big-data.

To reduce the influence of outliers, Suykens et al. [10] proposed the weighted LS-SVM. By iteratively assigning small weights to outliers and retraining, the method diminishes the effect of extreme values. Another solution was proposed by Yang et. al [16], where a truncated loss function is used in the objective which is solved by a concave-convex procedure and the newton algorithm. A third solution is suggested by Debruyne et. al [3], which proposes a weighted LS-SVM classification where weights are equal to spatial rank with respect to the other feature vectors in the group.

In comparison to the standard Support Vector Machines (SVM), the LS-SVM only requires solving a linear system, but it lacks sparseness in the number of solution terms. To solve this problem, Suykens et. al [11] propose a method that iteratively prunes the datapoints with lowest support values and retrains. Yang et.al [15] propose a one-step compressive pruning strategy to reduce the number of support vectors. Fixed-size LS-SVM sparsifies the LS-SVM by selecting important point or landmark points based on the quadratic Renyi Entropy criterion [12]. However the landmark points are fixed in advance and don't take into account information of the classification itself, which could lead to sub-optimal results. This is in contrast to the sparse LS-SVM [11], which chooses datapoints based on the impact on the classification boundary. A comparison of different pruning methods can be found in [7].

In this paper, we propose a method to solve the two problems at once. Our mean contributions consist off:

1. Kernel Concentration steps for outlier detection

2. Soft reweighting based on Stahel-Donoho outlyingness
3. A new Pruning strategy based on Entropy and determinantal point processes (DPP)

Other methods that try to tackle both problems at once are found in [2], where a primal formulation of the LS-SVM with a truncated loss is introduced, sparseness is obtained by the Nyström approximation. A second method is the weighted LS-SVM of Suykens et. al [10].

NOG AANPASSEN The remainder of this paper is organized as follows. In section 2 we introduce our sparse robust least squares support vector machine. In section 3 we propose our robust outlier detection routine, followed by our support vector sparseness routine (??). The extensive simulation results of both theoretical and real, industrial datasets listed in section 4 conform the robustness, prediction time speed-up and improved classifier efficiency of our proposed method. Finally, our main findings and suggestions for further research are summarized in the conclusion, section 5.

## 2. LS-SVM for classification

A binary classifier's goal is to learn a prediction model that assigns the correct label $y \in \{+1, -1\}$ to an unseen test sample. Restated more formally, we seek an SVM-based classifier that fits an optimal hyperplane between the two classes, where the hyperplane maximizes the distance to the closest point(s) of either class. This margin $||w||^{-1}$ maximization leads directly to a classifier with good generalisation properties, i.e: it will result in good classification performance on (unseen) test data, for example compared with density based estimators.

Given an $p$-variate trainingsset $x \in \mathbb{R}^p$ and the binary class label $y_i \in [+1, -1]$ for observation $x_i$ with index $i$, the following constrained optimization must be solved to obtain the LS-SVM representation of a support vector machine:

$$min\ J(w, b, e_i) = \frac{1}{2} w^T w + \frac{\gamma}{2} \sum_{i=1}^{n} e_i^2 \tag{1}$$

...subject to the equality constraints:

$$y_i[w^T \varphi(x_i) + b] = 1 - e_i \tag{2}$$

Here $w^T\varphi(x_i) + b$ is the hyperplane based decision function of the classifier with corresponding parameters $w$ and $b$, where the scalar $\gamma$ is used to control the over/under-fitting trade-off. Finally, $\varphi(.)$ is the transformation from input space $\mathbb{R}^p$ to the kernel feature space, abbreviated as $\mathcal{H}$.

The specified constraint states that every given multivariate sample should be lie on the correct side of hyperplane. Stated differently, the classifier should predict the class label of each sample correctly, where each observation $x_i$ has an corresponding error $e_i$ due to the constraint equality sign. This, in turn, implies that a LS-SVM loses its support vector sparseness compared to ordinary SVM's.

This constrained optimization problem is solved by the optimality conditions of its Lagrangian $\alpha$ as follows:

$$L(w, b, e; \alpha) = J(w, b, e) - \sum_{i=1}^{n} \alpha_i (y_i[w^T\varphi(x_i) + b] - 1 + e_i) \tag{3}$$

As a direct consequence of the equality sign in the given constraint the specified Lagrangian is now solvable trough a linear system of equations:

$$\frac{\partial L}{\partial w} = 0 \rightarrow w = \sum_{i=1}^{n} \alpha_i y_i \varphi(x_i) \tag{4}$$

$$\frac{\partial L}{\partial b} = 0 \rightarrow \alpha^T y = 0 \tag{5}$$

$$\frac{\partial L}{\partial e} = 0 \rightarrow \alpha = \gamma e \tag{6}$$

$$\frac{\partial L}{\partial \alpha_i} = 0 \rightarrow y_i[w^T\varphi(x_i) + b] = 1 - e_i \tag{7}$$

Combining the first and last equation and defining $\Omega(i, j)$ for two observations with index $i$ and $j$ as:

$$\Omega(i, j) = y_i y_j \varphi(x_i)^T \varphi(x_j) \tag{8}$$

$$= y_i y_j K(x_i, x_j) \tag{9}$$

Yields the following set of linear equations, where the kernel matrix $K$ transforms observations to the high dimensional kernel feature space $\mathcal{H}$.

$$\begin{bmatrix} 0 & y^T \\ y & \Omega + \gamma^{-1}\mathbf{I} \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{1} \end{bmatrix} \tag{10}$$

The least squares solution of the aforementioned system of equations is then used to obtain the Lagrange coefficients $\alpha$ and offset $b$, where the decision function used for test set prediction is defined as:

$$\hat{y}(x) = \sum_{1}^{n} \alpha_i y_i K(x, x_j) + b \tag{11}$$

Or, written in a more convenient matrix notation, short-writing $\beta_i = (\alpha_i \; y_i)^T$

$$\hat{y}(x) = \beta K + b\mathbf{1} \tag{12}$$

$\hat{y}_i$ is the predicted output of the ith training data point. Note that $\alpha$ satisfies the linear constraint $\sum_{i=1}^{n} \alpha_i e_i = 1$, and that the derivation for least squares support vector regression follows the same reasoning (DOES IT??)

## 3. Proposed method

### *3.1. Concentration steps in kernel feature space*

Having introduced the LS-SVM model, our attention first goes to the detection of outliers in the dataset. Modern multivariate robust statistical methods are frequently based on the Minimum Covariance Determinant (MCD) method, first introduced in (Rousseeuw, 1984, 1985). This Mahalanobis distance based estimator can withstand a substantial amount of trainingset contamination (up to 50%), and is nowadays (also) employed as the a refinement step of an initial estimator. This two-step mechanism inherits the robustness of the MCD, but offers an improved statistical efficiency [see e.g. REF: DetS, detMM]. Deterministic variants and well know PCA-based applications are described in [detMCD, RobPCA] respectively.

Given a trainingset $X$ of $n$ observations in $p$ dimensions, the MCD-objective is to find the $h$ observations whose sample covariance matrix has the lowest possible determinant, where the amount of regular observations $h < n$ is specified before the algorithm starts. The MCD-estimate of location $c_h$ is then the average of these $h$ points (the $h$-subset), whereas the scatter estimate is a multiple of their covariance matrix $\hat{\Sigma}_h$.

In order to find the MCD-estimate, the FastMCD-algorithm (Rousseeuw and Van Driessen, 1999) uses so-called concentration steps (C-steps). These iterative algorithm steps result in a decreasing covariance matrix determinant: a goodness-of-fit metric [BLABLABLA]. Based on its robustness properties, simplicity and proven convergence, we propose to modify the algorithm in way that it can work in kernel feature space, where it is used to for the detection of non-linear outliers.

To enable the C-steps methodology $\mathcal{H}$, we integrate the mapping function $\phi(.)$ in the required algorithm formula's:

$$c_h = \frac{1}{h} \sum_{i=1}^{h} \phi(x_i) \tag{13}$$

Likewise, the Mahalanobis distance in $\mathcal{H}$ is defined as:

$$||\phi(x) - c_h||^2_{\Sigma_h} = (\phi(x) - c_h)\Sigma_h^{-1}(\phi(x) - c_h) \tag{14}$$

Where the biased covariance matrix of the $h$-subset is calculated as:

$$\Sigma_h = \frac{1}{h}\sum_{i=1}^{h}(\phi(x_i) - c_h)(\phi(x_i) - c_h)^T \tag{15}$$

As we do not explicit know the mapping function $\phi(.)$, equation 14 cannot be calculated, a problem circumvented by the singular value decomposition of the covariance matrix:

$$\Sigma_h = V^T D V \tag{16}$$

Where $V$ is the matrix of eigenvectors $v^k$ and $D$ resembles the diagonal matrix of eigenvalues $\lambda^k$ for $k = 1, 2, \ldots, h$. The relation between eigenvalues and eigenvectors follows the identity:

$$\Sigma_h v^k = \lambda^k v^k \tag{17}$$

From the definition of equation 15, it can be seen that each eigenvector is a linear combination of the observations $\phi(x_i)$ in kernel feature space:

$$v^k = \sum_{1}^{n} \alpha_i^k(\phi(x_i) - c_n) \tag{18}$$

If we substitute the above in equation 17, it follows that:

$$n\lambda^k \alpha^k = \tilde{K}\alpha^k \tag{19}$$

Here, $\tilde{K}$ denotes the symmetric kernel matrix of $h$ rows, centered in $\mathcal{H}$. The weights $\alpha$ are determined by solving the eigendecomposition problem.
By refactoring $\Sigma^{-1}$ as $V^T D^{-1/2} D^{-1/2} V$ and exploiting reductant operations in the Mahalanobis distance formula, [REF: NADER] proofs that equation 14 can be calculated in $\mathcal{H}$ as:

$$||\phi(x) - c_n||_\Sigma^2 = \sum_{k=1}^{n}(\lambda^k)^{-1}\Big(\sum_{k=1}^{n}\alpha^k[...]\Big)^2 \tag{20}$$

Finally, the centered kernel matrix of the $h$-subset $\tilde{K}_h$ is calculated as [REF: K-PCA]:

$$\tilde{K}_h = ... \tag{21}$$

Where the centering of the kernel matrix of $n$ observations, given the $h$-subset reduces to:

$$\tilde{K}_h = ... \tag{22}$$

*3.2. The proposed C-step in kernel feature space*

We first standardize each dataset observation by $z_i = (x_i - \hat{\mu}_{mcd})/\hat{\sigma}_{mcd}$ where $\hat{\mu}_{mcd}$ and $\hat{\sigma}_{mcd}$ are the univariate location and scale estimations of the univariate MCD[REF]. This reduces the impact of different feature scales. Next, the standardized dataset is split in two subsets according to its class label $y \in \{+1, -1\}$. The following procedure is then applied per subset to find the regular observation list:

Step 1 Calculate the spatial median in $\mathcal{H}$, as introduced in [MICHIEL]. The spatial median can be written as a linear combination of the feature vectors. Denote $\gamma = (\gamma_1, \ldots, \gamma_n)^T$ the vector of coefficients, where it is know that $\sum_{i=1}^{n} \phi(x_i)\gamma_i$ equals the spatial median in feature space. Initialise $\gamma = (1/n, 1/n, \ldots)$ and recursively compute the normalized coefficients $\gamma = w/\sum_{i=1}^{n} w_i$, where $w_i$ for the observation with index $i$ is calculated as:

$$w_i = \frac{1}{\sqrt{K_{i,i} - 2\gamma^T K_{\cdot,i} + \gamma^T K \gamma}} \tag{23}$$

We apply fifteen iterations to obtain a good approximation.

Step 2 Find the $h$ observations with closest Euclidean distance to the spacial median, measured in $\mathcal{H}$, by [REF: The Kernel Trick for Distances]:

$$d_i = K(i,i) - 2\sum_{j=1}^{n} sum(K(i,j)gamma(j) + gamma' * K * gamma; \tag{24}$$

$$dist = diag(K) - 2 * sum(K. * repmat(gamma', n, 1), 2) + gamma' * K * gamma; \tag{25}$$

Step 4 Apply kernel C-step for a fixed amount of iterations as follows:

1. For each observation $x$, calculate the corresponding Mahalanobis distance, defined as:

$$||x - c_h||^2_{\Sigma_h} = (x - c_h)\Sigma_h^{-1}(x - c_h) \tag{26}$$

2. Sort the Mahalanobis distances in ascending order. Redefine the $h$-subset with the $h$ observations with lowest distance.

3. Obtain a new estimation of location and scatter:

$$c_h = \frac{1}{h}\sum_{i=1}^{h} x_i \tag{27}$$

$$\Sigma_h = \frac{1}{h}\sum_{i=1}^{h}(\phi(x_i) - c_h)(\phi(x_i) - c_h)^T \tag{28}$$

4. Re-iterate until a specified number of steps or until no convergence is obtained:

$$\det(\hat{\Sigma}_h(t)) = \det(\hat{\Sigma}_h(t-1)) \tag{29}$$

Step 5 Determine the final support vector candidate mask. Assign a binary weight to each observation $i$:

$$w_i = \begin{cases} 1, & \text{if } rd^2(i) \leq max(\chi^2_{0.975,p}, rd^2(h - subset)) \\ 0, & \text{otherwise} \end{cases} \tag{30}$$

The different steps of this algorithm are shown in figure [REF], where it can be seen that [WAT]. The final outcome, weights for each observation, are kept in memory to ...;

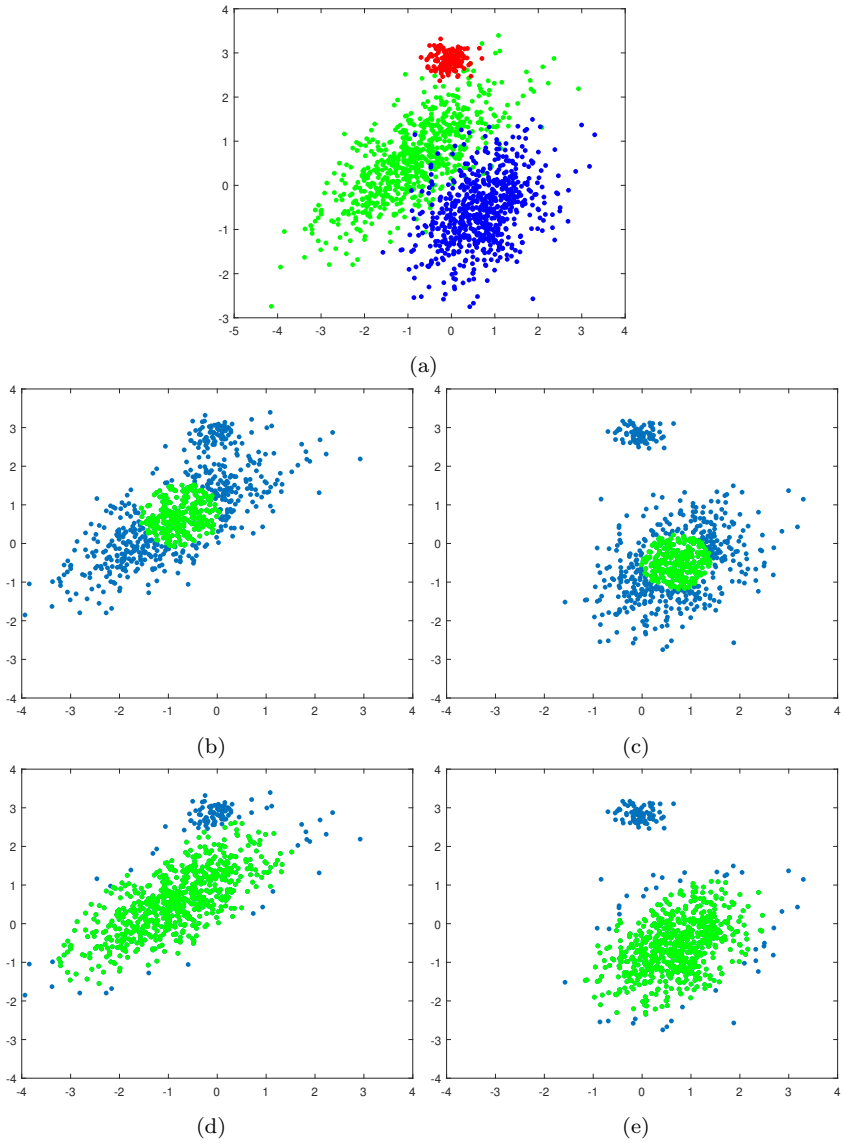


Figure 1: Figure a depicts a normally distributed classification problem, where both classes (blue, green) contain 10% disjoint outliers from the same material group, shown in red. After splitting the dataset by their class labels [STEP2?], the $h_{init}$ closest points from the spatial median are used as initial subsets for the C-step algorithm [STEP3?], here shown as green points in figure b, c respectively. After iterative execution we obtain the re-weighted results [STAP5?] (figures d and e). Support vectors are then determined from the detected regular observation list by a landmark selection algorithm (section [REF]).

## 4. Imposing sparseness

### *4.1. Selection of the best support vectors*

Having established the data model and optimized classification accuracy for contaminated datasets, let us return to the problem at hand: the development of a pruning strategy. This implies that prediction formula should be partitioned in a *relevant* (the support vectors or landmark points, found by the pruning algorithm) - and *irrelevant* part. An overview of existing pruning methods can be found in Hoegaerts et. al [7]. A first approach was suggest by Suykens et. al [11], where sparseness is imposed by pruning support values from the sorted support value spectrum resulting from the solution to the linear system. The motivation comes from the fact that LS-SVM support values are proportional to the errors at datapoints. Thus omitting the points that contribute least to the training error. An example of the support value spectrum can be seen on Figure **??**. The values with a high $|\alpha_k|$ reside close to the decision boundary and are thus important to classify correctly.

However blindly taking the points with largest support value spectrum could lead to sub-optimal results. Firstly, when outliers are present, you want to be certain that these are not chosen. Secondly, points are chosen independently towards each other. This results in clumping of landmarks. The first problem is addressed by running a kernelized minimum covariance determinant, which detects and omits potential outliers.

The second problem is solved by introducing a "region of interest" (ROI) for each class $Y = [+1, -1]$, which consist of the points with the $\beta$ percentage highest value of $Y\alpha_i y_i$, where $\alpha_i$ and $y_i$ are the support value and class label of training point $x_i$. In contrast to the proposed pruning strategy by Suykens et. al [11], the sign of alphas is taken in consideration when determining the ROI. Potential landmark points are now points with large importance and that predict the right class. This is easily seen from the prediction equation (12), where the contribution of a training point in the prediction a test point $z$ towards class Y is proportional to $Y\alpha_i y_i$.

The ROI represents a subset with points important for the decision boundary. Using a sampling algorithm that promotes diversity, $h$ landmark points are selected to represent the ROI such that no clumping is possible.

(a) Example bad Alpha spectrum

(b) Example good SV selection

*4.1.1. Entropy*

Selection of the landmark points is based on quadratic Renyi entropy [5] and the fixed size LS-SVM [12]. The landmarks points are chosen to maximize the quadratic Renyi Entropy:

$$H_R = -\log \int p(x)^2 dx. \tag{31}$$

The quadratic Renyi Entropy is approximated by the following equation[5]:

$$\int \hat{p}(x)^2 dx = \frac{1}{N^2} 1_v^{\mathrm{T}} \Omega 1_v, \tag{32}$$

where $1_v = [1; 1; ...; 1]$ and a normalized kernel is assumed with respect to density estimation. In the fixed-size LS-SVM approach, one chooses a fixed working set of size $M$ which is initialized randomly. Afterwards, random points are swapped in and out. If the entropy increases, the swapped point is accepted, otherwise the original subset is kept. This process continues until the change in entropy is small or a fixed number of iterations is reached.

In contrast to the original fixed-size LS-SVM formulation, which is used to find a representative subset for the Nyström approximation [12]. We propose to first determine to region of interest, which includes the points that have the highest contribution to the robust LS-SVM model. On this reduced dataset, the fixed-size LS-SVM algorithm is used to select the $h$ landmark points. This ensures that the ROI is well approximated and there is no clumping effect present. The improved sv selection for the toy-problem is visible on Figure **??**.

**Remark.** *It is important to first omit outliers, before continuing with the entropy procedure. Large contributions to the entropy will come from elements that have little or no structure [5].*

—-TODO—— MORE EXPLANATIONS DPP

*4.1.2. Determinantal point process*

Selection of landmark points is based on determinantal point processes (DPPs) [9]. DPPs are particularly interesting for set selection problems where diversity is preferred. A point process $\mathcal{P}$ on a ground set $\mathcal{Y} = 1, 2, ..., N$ is a probability measure over point patterns of $\mathcal{C}$ , which are finite subsets of $\mathcal{Y}$. When $\mathcal{C}$ is a random subset, drawn according to the DPP $\mathcal{P}$, we have:

$$\mathcal{P}(C \subseteq \mathcal{Y}) = \det(K_{\mathcal{C}}), \tag{33}$$

where $K \preceq I$ is a positive symmetric semidefinite matrix with all eigenvalues smaller then or equal to 1, containing the index elements of $\mathcal{Y}$. $K_{\mathcal{C}} = [K_{i,j}]_{i,j\in\mathcal{C}}$ contains the selected rows and columns of $K$ and $\det(K_\emptyset) = 1$. From equation (33) follows:

$$\mathcal{P}(i \in \mathcal{Y}) = K_{i,i} \tag{34}$$

$$\mathcal{P}(i,j \in \mathcal{Y}) = K_{i,i}K_{j,j} - K_{i,j}K_{j,i} \tag{35}$$

$$= \mathcal{P}(i \in \mathcal{Y})\mathcal{P}(j \in \mathcal{Y}) - K_{i,j}^2. \tag{36}$$

The diagonal elements of the kernel matrix give the marginal probability of inclusion for individual elements of $\mathcal{Y}$, whereas the off-diagonal elements determine the (anti-)correlation between pairs of elements. Thus for large values of $K_{i,j}$, or a high similarity, points are unlikely the appear together.

In our case, we would like to build a DPP based on the kernel matrix $K$, which is done using L-ensembles [1]. The probability of observing a subset $C \subseteq \mathcal{Y}$ is now equal to:

$$\Pr(\mathcal{C}) = \frac{\det(K_{\mathcal{C}})}{\det(K+I)}, \tag{37}$$

where $I$ is the identity matrix, and $K$ a positive semidefinite matrix indexed by the elements of $\mathcal{Y}$. In contrast to equation (33), $K$ only has to be positive semidefinite, while the eigenvalues previously where bounded above. When conditioning on a fixed cardinality $k = |\mathcal{C}|$, one gets the k-DPP [8].

The landmark selection algorithm consists of the following steps: We propose to first determine to region of interest, which includes the points that have the highest contribution to the robust LS-SVM model. On this reduced dataset, a $k$-DPP [8], where $k = h$, is used to sample the $h$ landmark points. This ensures that the ROI is well approximated and there is no clumping effect present. The improved sv selection for the toy-problem is visible on Figure **??**.

**Remark.** *It is important to first omit outliers, before continuing with the DPP sampling. In order to promote diversity, points that have a high similarity have a low chance of being sampled together (see equation* (36)*). Consequently outliers, that have a low similarity to any other point in the dataset, have a high chance of being sampled.*

*4.2. Information transfer*

The information contained in the prune candidates can then be transferred to the support vectors - an idea originally introduced in [???]. Starting from equation 12 with the appropriate matrix dimensions:

$$\hat{y}_{(1,n_2)} = \beta_{(1,n_1)} K_{(n_1,n_2)} + b_{(1,1)}\ \mathbf{1}_{(1,n2)} \tag{38}$$

Introducing sparse matrices, we could partition this expression in a (non) support vector part, denoted by the subscript $S$ and $N$ respectively.

$$\hat{y}_{(1,n_2)} = \beta_{S(1,n_1)} K_{S(n_1,n_2)} + \beta_{N(1,n_1)} K_{N(n_1,n_2)} + b_{(1,1)}\ \mathbf{1}_{(1,n2)} \tag{39}$$

Next, one transfers the information of $\beta_N K_N$ using the update $\Delta\beta$:

$$\Delta\beta K_S = \beta_N K_N \tag{40}$$

$$\Delta\beta = K_S^{\dagger} \beta_N K_N \tag{41}$$

We now have obtained an explicit expression for the update of our Lagrange multipliers.

$$\hat{\beta}_S = \beta_S + \Delta\beta \tag{42}$$

If we omit all zero rows in the matrices above we obtain a compressed matrix of size $m_1$ rows in $n_2$ dimensions. Here, $m_1$ equals the (a priori, before the training stage) defined number of support vectors. The classier prediction equation finally boils down to:

$$\hat{y}_{(1,n_2)} = \beta_{(1,m)} K_{(m,n_2)} + b_{(1,1)}\ \mathbf{1}_{(1,n2)} \tag{43}$$

Which is implemented using equation **??** given the knowledge that $n_1 = m$ - the number of a priori defined support vectors.

The only problem that remains is the section of the most relevant support vectors....

*Algorithm 3— Sparse K-MCD based LS-SVM.* summary all steps algorithm

## 5. Experiments

*5.1. Simulation results*

*5.1.1. Linear example*

Two Gaussians close, the reverse labels behind at large distance. See Robustified LS-SVM [3]

*5.1.2. Non-Linear example*

Yin-Yang, Two spiral dataset or Cross Dataset [16]

*5.1.3. UCI*

Robust: Banana, Celveland heart, Glass, Heartstatlog, liver disorder, monk PIMA, ripley, Transfusion, Vehicle [16]

Robust + sparse: Splice, Pendigits (choose two digits 3 vs 4), Satimage (1 vs 6), USPS (1 vs 2), Mushrooms, Protein (1 vs 2).[2] Outliers = 30 % samples that where far away from decision hyperplane, then randomly sample 1/3 of them and flip labels. datasets are in https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/

Robust: Wine dataset + Linear, Glass, Biscuit Dough, Alon colon cancer, Hepatocellular carcinoma dataset [3]. However mostly linear

*5.2. Industrial data results*

## 6. Conclusions and future work


## Acknowledgements

We thank Johan Speybrouck for providing us the industrial datasets and Tim Wynants for his feedback throughout this project. We also acknowledge the financial support of the VLAIO, grant HBC.2016.0208, for making this industrial research possible.


## References


[1] Alexei Borodin. Determinantal point processes. *arXiv preprint arXiv:0911.1153*, 2009.

[2] Li Chen and Shuisheng Zhou. Sparse algorithm for robust lssvm in primal space. *Neurocomputing*, 275:2880–2891, 2018.

[3] Michiel Debruyne, Sven Serneels, and Tim Verdonck. Robustified least squares support vector classification. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 23(9):479–486, 2009.

[4] Marcelo Espinoza, Johan AK Suykens, and Bart De Moor. Fixed-size least squares support vector machines: A large scale application in electrical load forecasting. *Computational Management Science*, 3(2):113–129, 2006.

[5] Mark Girolami. Orthogonal series density estimation and the kernel eigenvalue problem. *Neural computation*, 14(3):669–688, 2002.

[6] Ivan Goethals, Kristiaan Pelckmans, Johan AK Suykens, and Bart De Moor. Identification of mimo hammerstein models using least squares support vector machines. *Automatica*, 41(7):1263–1272, 2005.

[7] Luc Hoegaerts, Johan AK Suykens, Joos Vandewalle, and Bart De Moor. A comparison of pruning algorithms for sparse least squares support vector machines. In *International Conference on Neural Information Processing*, pages 1247–1253. Springer, 2004.

[8] Alex Kulesza and Ben Taskar. k-dpps: Fixed-size determinantal point processes. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1193–1200, 2011.

[9] Alex Kulesza, Ben Taskar, et al. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012.

[10] Johan AK Suykens, Jos De Brabanter, Lukas Lukas, and Joos Vandewalle. Weighted least squares support vector machines: robustness and sparse approximation. *Neurocomputing*, 48(1-4):85–105, 2002.

[11] Johan AK Suykens, Lukas Lukas, and Joos Vandewalle. Sparse approximation using least squares support vector machines. In *2000 IEEE International Symposium on Circuits and Systems. Emerging Technologies for the 21st Century. Proceedings (IEEE Cat No. 00CH36353)*, volume 2, pages 757–760. IEEE, 2000.

[12] Johan AK Suykens, Tony Van Gestel, and Jos De Brabanter. *Least Squares Support Vector Machines*. World Scientific, 2002.

[13] Johan AK Suykens, Joos Vandewalle, and Bart De Moor. Optimal control by least squares support vector machines. *Neural networks*, 14(1):23–35, 2001.

[14] Tony Van Gestel, Johan AK Suykens, D-E Baestaens, Annemie Lambrechts, Gert Lanckriet, Bruno Vandaele, Bart De Moor, and Joos Vandewalle. Financial time series prediction using least squares support

vector machines within the evidence framework. *IEEE Transactions on neural networks*, 12(4):809–821, 2001.

[15] Lixia Yang, Shuyuan Yang, Rui Zhang, and Honghong Jin. Sparse least square support vector machine via coupled compressive pruning. *Neurocomputing*, 131:77–86, 2014.

[16] Xiaowei Yang, Liangjun Tan, and Lifang He. A robust least squares support vector machine for regression and classification with noise. *Neurocomputing*, 140:41–52, 2014.