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 KarushKuhnTucker 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 [14]. 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 [15], financial time series prediction [17], 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. A 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. [12] 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 [19], where a truncated loss function is used in the objective that 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 class.

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 [13] propose a method that iteratively prunes the datapoints with lowest support values and retrains. Yang et.al [18] 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 [14]. 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 [13], 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 focus on tackling both problems at once. Other methods are found in [1], 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 [12]. We propose a robust LS-SVM based on the Minimum covariance determinant (MCD) [8], which is known to be a highly robust estimator of multivariate location and scatter. However, in contrast to the original formulation, the location and scatter is determined in feature space using the kernel trick. This results in non-linear outlier detection. Afterwards extreme sparsity is obtained by determinantal point processes (DPP) [10], which are capable of selecting diverse samples. To summarize, our mean contributions consist off:

1. Kernel concentration steps for outlier detection.
2. A new Pruning strategy based on entropy and determinantal point processes

The remainder of this paper is organized as follows. A brief overview of LS-SVM is given in section 2. In section 3 we propose our robust outlier detection routine, followed by our support vector sparseness routine in section 4. The extensive simulation results of both theoretical and real, industrial datasets listed in section 5 confirm 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 6.

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 $i = 1, .., n$, the following constrained optimization must be solved to obtain the LS-SVM representation of a support vector machine:

$$\min\ J(w,b,e_i) = \frac{1}{2}w^T w + \frac{\gamma}{2}\sum_{i=1}^{n} e_i^2$$
$$\text{such that } y_i[w^T\varphi(x_i)+b] = 1-e_i,\ \ i=1,..,n$$

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 γ 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 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 α as follows:

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

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

$$\begin{aligned}
\frac{\partial L}{\partial w} &= 0 \rightarrow w = \sum_{i=1}^{n}\alpha_i y_i \varphi(x_i)\\
\frac{\partial L}{\partial b} &= 0 \rightarrow \sum_{i=1}^{n}\alpha_i y_i = 0\\
\frac{\partial L}{\partial e_i} &= 0 \rightarrow \alpha_i = \gamma e_i,\ \ i=1,..,n\\
\frac{\partial L}{\partial \alpha_i} &= 0 \rightarrow y_i[w^T\varphi(x_i)+b] = 1-e_i,\ \ i=1,..,n
\end{aligned}$$

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

$$\begin{aligned}
\Omega(i,j) &= y_i y_j \varphi(x_i)^T\varphi(x_j)\\
&= y_i y_j K(x_i,x_j)
\end{aligned}$$

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

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

where $\mathbf{y} = [y_1; ...; y_n]$, $\mathbf{1} = [1; ...; 1]$ and $\mathbf{I}$ the identity matrix. The least squares solution of the aforementioned system of equations is then used to obtain the Lagrange coefficients α and offset b, where the decision function used for test set prediction is defined as:

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

Note that α 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 where the sign is omitted.

3. Proposed method

We first start by laying out the main methodology of our proposed robust, sparse LS-SVM algorithm.

1. Outlier detection by kernel Csteps.
2. Train the robust LS-SVM by solving the system of equations.
3. Determine a region of interest (ROI), which is a reduction of the original dataset, that consist of points close to the decision boundary. This subset represent potential good support vector values.
4. Reduce the ROI even more by sampling the desired amount of support vector values using a method that promotes diversity:
    (a) Entropy
    (b) DPP
5. Transfer the information of non-selected support vectors to selected ones.

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: a noteworthy difference compared to off-theshelve robust statistical algorithms as normality is always assumed.

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

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

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

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

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

$$\Sigma_h = \frac{1}{h}\sum_{i=1}^{h}(\phi(x_i) - c_h)(\phi(x_i) - c_h)^T. \tag{3}$$

As we do not explicit know the mapping function $\phi(.)$, equation (2) cannot be calculated, a problem circumvented by the eigendecomposition of the covariance matrix:

$$\Sigma_h = V^T D V.$$

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

$$\Sigma_h v^k = \lambda^k v^k \tag{4}$$

From the definition of equation (3), it can be seen that each eigenvector is a linear combination of the observations $\phi(x_i)$ in kernel feature space:

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

If we substitute the above in equation (4), it follows that:

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

Here, $\tilde{K}$ denotes the symmetric kernel matrix of h rows, centred in $\mathcal{H}$. The weights α are determined by solving the eigendecomposition problem.
By refactoring Σ^{-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 (2) can be calculated in $\mathcal{H}$ as:

$$||\phi(x) - c_n||_{\Sigma}^2 = \sum_{k=1}^{n} (\lambda^k)^{-1} \Big(\sum_{k=1}^{n} \alpha^k \tilde{k}(x_i, x) \Big)^2$$

Finally, the centered kernel matrix $\tilde{K}$ is calculated as [11]:

$$\begin{aligned} \tilde{K}_{ij} &= \left(\phi(x_i) - \frac{1}{n}\sum_{m=1}^{n}\phi(x_m)\right) \cdot \left(\phi(x_j) - \frac{1}{n}\sum_{k=1}^{n}\phi(x_k)\right) \\ &= K_{ij} - \frac{1}{n}\sum_{m=1}^{n} K_{mj} - \frac{1}{n}\sum_{k=1}^{n} K_{ik} + \frac{1}{n^2}\sum_{m=1}^{n}\sum_{k=1}^{n} K_{mk} \\ &= \Big(K - 1_n K - K 1_n + 1_n K 1_n\Big)_{ij}, \end{aligned}$$

where $\tilde{K} \in \mathbb{R}^{n\times n}$ and $1_n \in \mathbb{R}^{n\times n}$ with all entries set to $1/n$. The centering of the kernel matrix of t observations, given the original trainingset follows a similar derivation:

$$\tilde{K}_t = K_t - 1_t K - K_t 1_n + 1_t K 1_n,$$

with kernel matrix $\tilde{K}_t \in \mathbb{R}^{t\times n}$ and the matrix $1_t \in \mathbb{R}^{t\times n}$ with all entries set to $1/n$.

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 $y \in \{+1, -1\}$ and our proposed procedure is applied (per subset) to derive at an outlier free support vector candidate set.

Step 1 Calculate the spatial median in $\mathcal{H}$, as introduced in [2]. 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 $\sum_{i=1}^{n} \phi(x_i)\gamma_i$

equals the spatial median in feature space. Initialise $\gamma = (1/n, 1/n, \dots)$ 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:

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

Ten iterations are usually enough to obtain a good approximation.

Step 2 Calculate the Euclidean distance for each observation to the spatial median:

$$\begin{aligned} d_i &= ||\phi(x_i) - \sum_{j=1}^{N} \gamma_j \phi(x_j)||^2 \\ &= ||\phi(x_i)||^2 + ||\sum_{j=1}^{N} \gamma_j \phi(x_j)||^2 - 2 < \phi(x_i), \sum_{j=1}^{N} \gamma_j \phi(x_j) > \\ &= K(x_i, x_i) + \sum_{j=1}^{N} \sum_{k=1}^{N} \gamma_j \gamma_k K(x_j, x_k) - 2 \sum_{j=1}^{N} \gamma_j K(x_i, x_j). \end{aligned}$$

Step 3 Sort these distances in ascending order. Define the initial h-subset by the h_{init} observations with the lowest distance.

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

1. Calculate the kernel matrix K_h for the given h-subset of observations.
2. Center K_h in $\mathcal{H}$ by equation 3.1, obtaining $\tilde{K}_h$
3. Perform the singular value decomposition on $\tilde{K}_h$ to obtain the eigenvectors α and eigenvalue vector L_d.
4. Normalize each eigenvector α_i by its corresponding eigenvalue, equation [...]
5. Calculate the kernel matrix for the given h-subset of observations in n features.
6. Center the kernel matrix by equation [REF..] and obtain the malalanobis distances with equation [REF].
7. Sort these Mahalanobis distances in ascending order and redefine the h-subset as the h observations with smallest distance.

8. If the new h-subset is equal to the previous one, exit the C-step loop.

Step 5 If the h-subset size is lower then h_{end}, increase the h-subset size by (say) 5% and go back to step 4.

Step 6 Determine the final support vector candidate mask. Assign a binary weight to each observation i:

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

The different steps of this algorithm are shown in Figure 1, for a linear and non-linear kernel. Given an initial estimate of the h-subset, the C-step converge to the outlier free dataset. The final outcome, weights for each observation, are used to train a robust LS-SVM that only consists of regular observations.

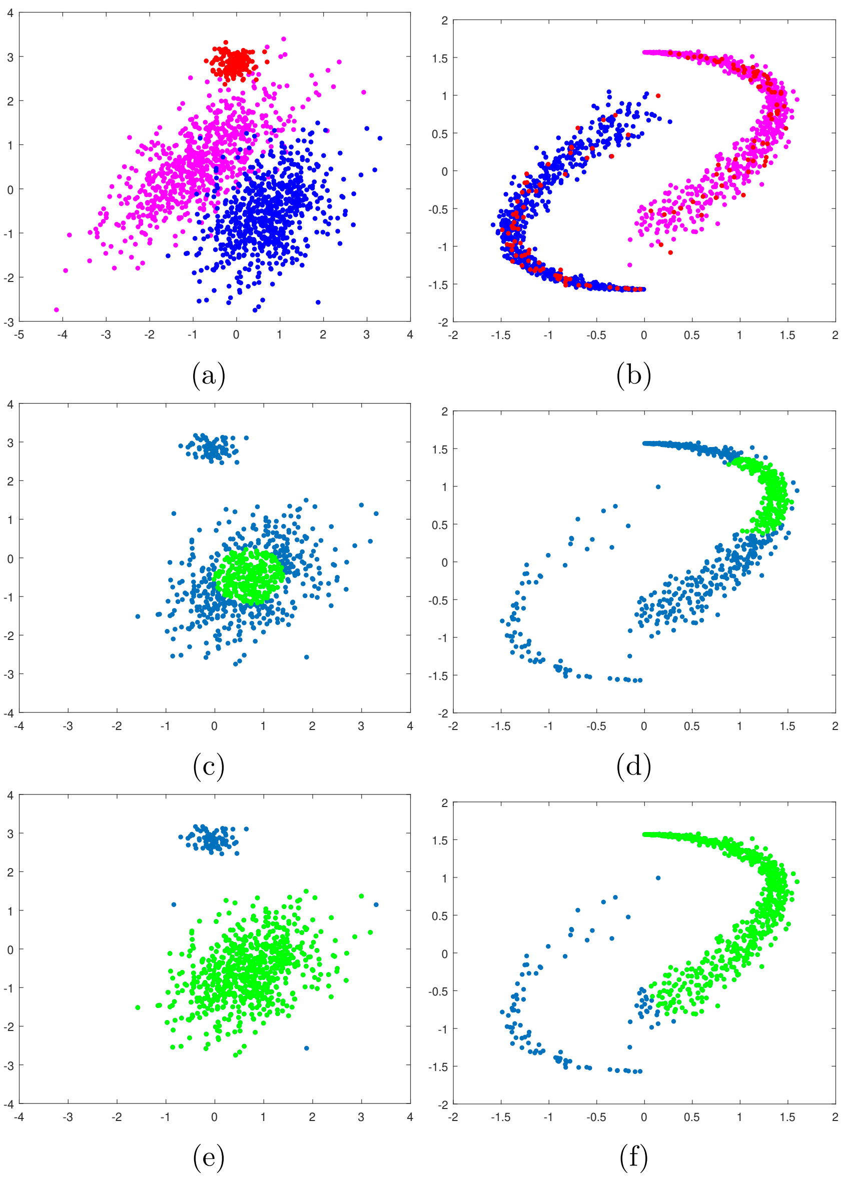

Figure 1: Figure a depicts a normal distributed, binary classification problem. A linear kernel will be used in the outlier detection of this dataset. Figure b shows a non-linear classification problem, where an RBF kernel with bandwidth $\sigma = INVULLEN$ is used. Both classes (blue, green) contain 10% outliers from, shown in red. After splitting the dataset by their class labels, the h_{init} closest points from the spatial median are used as initial subsets for the C-step algorithm [STEP3], depicted in green for the first class of both datamodels (Figures c, d). After iterative kernel C-steps, we obtain the re-weighted results [STAP6], shown in Figures e and f. Support vectors are then determined from the detected regular observation list by a landmark selection algorithm (section 4).

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 [13], 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. 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 subset for each class $y \in \{+1, -1\}$ called the "region of interest" (ROI), which consist of points close to the decision boundary. Determining the ROI is done for each class and consists out of two steps:

1. Determine the subset which consists of all points belonging to the class and are not considered outliers by the kernel MCD procedure.
2. Reduce the subset even more by the selecting the β percentage[1] datapoints with highest value α_i, where α_i is the support value.

In contrast to the proposed pruning strategy by Suykens et. al [13], 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 (1), where the contribution of a training point in the prediction a test point x_t towards class $y \in \{+1, -1\}$ is proportional to $y\, w_i$ with $w_i = \alpha_i y_i$.

[1]In the experiments, a reduction rate $\beta = 0.25$ is used.

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

4.1.1. Entropy

Selection of the landmark points is based on quadratic Renyi entropy [5] and the fixed size LS-SVM [14]. The landmarks points are chosen to maximize the quadratic Renyi Entropy:

$$H_R = -\log \int p(x)^2 dx. \tag{5}$$

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

$$\int \hat{p}(x)^2 dx = \frac{1}{N^2} 1_v^{\mathrm{T}} \Omega 1_v, \tag{6}$$

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 [14]. We propose to first determine to ROI, 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 l landmark points. This ensures that the ROI is well approximated and there is no clumping effect present. The downsides of the entropy criteria are that a relatively large number of iterations are necessary to get a diversified selection, secondly only the Gaussian kernel can be used [5].

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].*

Given the ROI of candidate support vectors for a class $y \in \{+1, -1\}$, the entropy based selection algorithm is repeated for every class [14]:

1. Choose a working set of size l by randomly selection support vector out of the ROI.
2. Randomly select a point out of the ROI and replace it by a random point in the working set.
3. Calculate the quadratic Renyi entropy for the new working set using equations (5) and (6).
4. If the entropy increases, keep the swapped point, otherwise return to the original working set.
5. Stop if the change in entropy is small or a maximum number of iterations is reached. Otherwise go back to step 2.

4.1.2. Determinantal point process

Selection of landmark points is based on determinantal point processes (DPPs) [10]. 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:

$$\mathcal{P}(C \subseteq \mathcal{Y}) = \det(K_{\mathcal{C}}), \tag{7}$$

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 (7) follows:

$$\begin{aligned} \mathcal{P}(i \in \mathcal{Y}) &= K_{i,i} \\ \mathcal{P}(i, j \in \mathcal{Y}) &= K_{i,i}K_{j,j} - K_{i,j}K_{j,i} \\ &= \mathcal{P}(i \in \mathcal{Y})\mathcal{P}(j \in \mathcal{Y}) - K_{i,j}^2. \end{aligned} \tag{8}$$

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. Which is essential to promote diversity,

In our case, we would like to build a DPP based on the kernel matrix K, which is done using L-ensembles [10]. The probability of observing a subset $C \subseteq \mathcal{Y}$ is now equal to:

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

where I is the identity matrix, and K a positive semidefinite matrix indexed by the elements of $\mathcal{Y}$. In contrast to equation (7), 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 [9]. MEER UITLEG K-DPP

The landmark selection algorithm consists of the following steps: We propose to first determine the ROI, which includes the points that have the highest contribution to the robust LS-SVM model. On this reduced dataset, a k-DPP [9], where $k = l$, is used to sample the l landmark points. This ensures that the ROI is well approximated and there is no clumping effect present.

Remark. *It is important to first omit outliers, before continuing with the k-DPP sampling. In order to promote diversity, points that have a high similarity have a low chance of being sampled together (see equation* (8)*). Consequently outliers, that have a low similarity to any other point in the dataset, have a high chance of being sampled.*

Given the ROI of candidate support vectors for a class $y \in \{+1, -1\}$, the k-DPP selection algorithm is repeated for every class to select l landmark points [9]:

1. Calculate the eigenvector/value pairs $\{(\mathbf{v}_r, \lambda_r)\}$ from the L-ensemble matrix $L = K_{ROI}(I - K_{ROI})^{-1}$, where $K_{ROI} \in \mathbb{R}^{N_{ROI} \times N_{ROI}}$ is the kernel matrix containing the points in the ROI.
2. Iterate over all components of the eigenvector/value pairs and include the component r if $u \sim U[0,1] < \lambda_r \frac{e_{k-1}^{r-1}}{e_k^r}$, where k is number of components selected so far, iterator $r = N_{ROI}, ..., 1$ and k elementary polynomial $e_k^r = \sum_{\substack{J \subseteq \{1,...,r\}\} \\ |J|=k}} \prod_{n \in J} \lambda_n$. Continue to the next step if you have selected l components. The matrix V is now equal to the eigenvectors of included components.
3. Select a landmark point x_i from ROI with $\Pr(x_i) = \frac{1}{|V|} \sum_{\mathbf{v} \in V} (\mathbf{v}^{\mathrm{T}} \mathbf{e}_i)^2$. The vector $\mathbf{e}_i$ is the ith standard basis, which is all zeros except for a one in the ith position.
4. Recalculate $V \leftarrow V_{\perp}$, an orthonormal basis for the subspace of V orthogonal to $\mathbf{e}_i$. Afterwards go back to step 3 until l landmark points are selected.

4.2. Information transfer

The information contained in the not selected support vectors can then be transferred to the selected support vectors, an idea originally introduced in [16]. Starting from equation (1) for the outlierfree dataset, we get in matrix notation:

$$\hat{\mathbf{y}} = \operatorname{sign}(K\beta + b\mathbf{1}),$$

with $\hat{\mathbf{y}} = [\hat{y}_1; ...; \hat{y}_n]$ the prediction vector and $\beta = [\alpha_i y_i, ..., \alpha_i y_{n_h}]^{\mathrm{T}}$, where n_h is the number of points in the outlierfree dataset. We can partition this expression in the selected landmark part and non-selected support vectors:

$$\hat{\mathbf{y}} = \operatorname{sign}(K_l\beta_l + K_{nl}\beta_{nl} + b\mathbf{1}),$$

where K_l is the matrix whose columns are selected from K using the landmark selection while the other columns of K form matrix K_{nl}, and β_l is a column vector whose elements are the multipliers corresponding to columns in K_l while the other multipliers compose the vector β_{nl}. Next, one transfers the information of $\beta_{nl}K_{nl}$ using the update $\Delta\beta$:

$$\begin{aligned} \Delta\beta K_l &= K_{nl}\beta_{nl} \\ \Delta\beta &= K_l^{\dagger} K_{nl}\beta_{nl}. \end{aligned}$$

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

$$\hat{\beta}_l = \beta_l + \Delta\beta.$$

The classier prediction equation finally boils down to:

$$\hat{y}(x) = \operatorname{sign}(K_l\hat{\beta}_l + b\mathbf{1}).$$

Remark. *As the outlier detection and landmark selection is done on each class separately. The proposed method is easily applicable in a multi-class classification setting.*

Algorithm 3— summary selection of landmark points.

The following procedure is applied on the robust trained LS-SVM and outlierfree dataset:

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 [19]

5.1.3. UCI

Robust: Banana, Celveland heart, Glass, Heartstatlog, liver disorder, monk PIMA, ripley, Transfusion, Vehicle [19]

Robust + sparse: Splice, Pendigits (choose two digits 3 vs 4), Satimage (1 vs 6), USPS (1 vs 2), Mushrooms, Protein (1 vs 2).[1] 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/lib-svmtools/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] Li Chen and Shuisheng Zhou. Sparse algorithm for robust lssvm in primal space. *Neurocomputing*, 275:2880–2891, 2018.

[2] Michiel Debruyne, Mia Hubert, and Johan Van Horebeek. Detecting influential observations in kernel pca. *Computational Statistics & Data Analysis*, 54(12):3007–3019, 2010.

[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] Mia Hubert and Michiel Debruyne. Minimum covariance determinant. *Wiley interdisciplinary reviews: Computational statistics*, 2(1):36–43, 2010.

[9] 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.

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

[11] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

[12] 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.

[13] 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.

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

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

[16] Shaohui Tao, Dezhao Chen, and Weixiang Zhao. Fast pruning algorithm for multi-output ls-svm and its application in chemical pattern classification. *Chemometrics and intelligent Laboratory systems*, 96(1):63–69, 2009.

[17] 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.

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

[19] 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.