Accelerating the Training of an LP-SVR over Large Datasets

Pablo Rivas[0000−0002−8690−0987]

Computer Science, Baylor University, Waco, Texas, USA
Pablo_Rivas@Baylor.edu

Abstract. This paper presents a learning speedup method based on the relationship between the support vectors and the within-class Mahalanobis distances among the training set. We explain how statistical properties of the data can be used to pre-rank the training set. Then we explain the relationship among the pre-ranked training set indices, convex hull indices, and the support vector indices. We also explain how this method has better efficiency than those approaches based on the convex hull, especially, at large-scale problems. At the end of the paper we conclude by explaining the findings of the experimental results over the speedup alternative.

Keywords: Support vector machines for regression · Linear programming · Big data sets.

1 Introduction

Support Vector Machines for Regression (SVRs) are popular option for learning to predicting a real-valued target given some training data. Their most common applications include predicting popularity of online videos [27], forecasting wind speed [25], energy consumption prediction [32], or forecasting rainfall [16]. SVRs can be posed in different ways motivated by different problems. For instance, $\epsilon-$SVR was an attempt to provide flexibility to an inflexible SVR solution by allowing the SVR to make mistakes; any mistakes beyond ϵ would start being penalized [4]. This parameter needs to be set up experimentally by observing the data and determining what amount of error can be tolerated in proportion to the observed target data. However, the value of ϵ is very likely to vary from problem to problem and its relationship with the support vectors is not intuitive; the $\nu-$SVR was introduced to solve this problem [26]. Its parameter ν provides the necessary meaning with respect to the support vectors (SVs) and is consistent from problem to problem; the $\nu-$SVR does not eliminate the parameter ϵ but removes the requirement of the user to provide it because it is introduced as part of the optimization problem and is calculated automatically. There are many other SVRs motivated by different issues. In this paper we will focus on the formulation of our earlier work [24], which aims to address large-scale training of a linear programming SVR (LP-SVR) based on the solution of small LP problems. Our formerly proposed algorithm has been mathematically proven

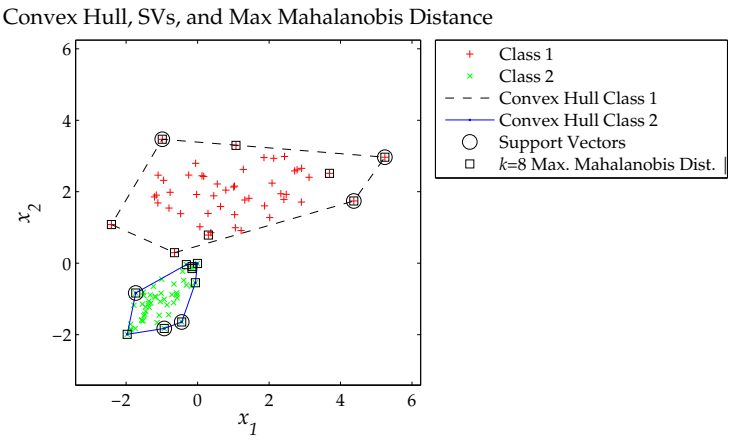

Fig. 1. Relationship between convex hull and maximum Mahalanobis distance for the two class problem. Here it is shown the original separable two class problem, class convex hull, support vectors, and $k = 8$ maximum Mahalanobis distance samples. Both SVs and Convex Hull match the $k = 8$ maximum Mahalanobis distance samples.

to converge, but the training process is still costly in proportion to the number of samples, N. This motivates the exploration of alternatives for speeding up the learning process. In this paper, we propose using a distance measure for ranking the training set and providing a sequence in which the data could be trained for speeding. To achieve the speed up, the data is ranked according to its distance to the class mean.

It is well known [1, 9, 11, 13, 19, 28, 29, 33] that in classification tasks, those data points closer to the decision boundaries (*i.e.*, convex hull of the data class) are more likely to be support vectors, as illustrated in Figure 1 with bold circles. Figure 1 depicts the relationship between convex hull and maximum Mahalanobis distance (MD) for an arbitrary two class problem. The figure shows the convex hull of each class, the SVs that define the separating hyperplane, and the eight samples having the maximum MD from their class center. Note that both SVs and convex hull match the eight samples with maximum MD.

Vapnik [28] improved the speed of his learning method by considering only those variables on the boundaries of the feasible region instead of considering all the data, which allowed computational tractability of some problems. Then Joachims [9] defined a heuristic approach to identify variables at boundaries based on Lagrange multiplier estimates. Later, Bennett, *et al.* [1] posed the problem of finding the optimal separating hyperplane using the distance between class convex hulls. A similar concept was followed by Keerthi, *et al.* [11] in 2002, by Osuna, *et al.* [19] in 2003, and by Zhenbing, *et al.* [13] in 2010. In 2013, Wang *et al.* [29], describe an approach to calculate distances to the convex-hull to improve a particular kind of model known as online SVMs. Gu *et al.* in 2018 [8], proposed another convex-hull approach by directly calculating it and ranking

data samples for naive versions of ℓ_1 and ℓ_2 SVMs. Most recently, in 2020, Chen *et al.* [3], provided an entire model based on Mahalanobis distances that are part of the learning process of a twin SVM formulation; such work has a similar motivation to our early work in 2011 [23]; however, they focus on an SVM model that incorporates sample selection into the learning process.

In this paper we introduce an approach to accelerate the training of an LP-SVR, specifically, which uses a probabilistic argument and can be applied to other flavors of SVRs. It is unique in that it aims for a specific flavor of SVR and that it can easily be extended to other models with relative training speed improvements. This paper is organized as follows: Section 2 introduces the background information for an LP-SVR, convex hull, and Mahalanobis distances. Section 3 describes our experimental results, while Section 4 addresses some computational concerns of the proposed approach. Conclusions are drawn in Section 5.

2 Within-Class Distances for Learning Speed Up

In the following discussion we assume that our training data $\mathcal{T}_\phi = \{\mathbf{x}_i, d_i\}_{i=1}^N$ has been taken to the kernel-induced feature space; $\mathbf{x} = [x_1, x_2, \ldots, x_M]$ is a feature vector in $\mathbb{R}^M$, and $d \in \mathbb{R}$ is the target output.

2.1 LP-SVR

In this research we are training the following LP-SVR (see [24] for details):

$$\begin{aligned} \min_{\boldsymbol{\alpha}^+,\boldsymbol{\alpha}^-,b^+,b^-,\boldsymbol{\xi},\mathbf{u}} \quad & \textstyle\sum_{i=1}^N \left(\alpha_i^+ + \alpha_i^- + 2C\xi_i\right) \\ \text{s.t.} \quad & \begin{cases} -\sum_{i=1}^N (\alpha_i^+ - \alpha_i^-) k(\mathbf{x}_j, \mathbf{x}_i) \ldots \\ \qquad -b^+ + b^- - \xi_j + u_j = \epsilon - d_j \\ \sum_{i=1}^N (\alpha_i^+ - \alpha_i^-) k(\mathbf{x}_j, \mathbf{x}_i) \ldots \\ \qquad +b^+ - b^- - \xi_j + u_j = \epsilon + d_j \\ \qquad \alpha_j^+, \alpha_j^-, b^+, b^-, \xi_j, u_j \geq 0 \end{cases} \\ \text{for} \quad & \qquad j = 1, 2, \ldots, N, \end{aligned} \tag{1}$$

where the kernel mapping $k(\mathbf{x}_i, \mathbf{x}_j) : \mathcal{X}^{(N\times M)\times(M\times N)} \mapsto \mathcal{H}^{N\times N}$. Then, assume that the slack variables ξ_i, ξ_i^* can be expressed as simply $2\xi_i$ (e.g. $\xi_i \xi_i^* = 0$). Then, let us introduce a slack variable $\mathbf{u}$ to get rid of the inequalities in the original SVR formulation [24]. As a consequence of these assumptions,

Problem (1) can be posed as the linear programming problem in its canonical form. To do so, one can define the following equalities:

$$\mathbf{A} = \begin{pmatrix} -\mathbf{K} & \mathbf{K} & -1 & 1 & -\mathbf{I} & \mathbf{I} \\ \mathbf{K} & -\mathbf{K} & 1 & -1 & -\mathbf{I} & \mathbf{I} \end{pmatrix}, \tag{2a}$$

$$\mathbf{b} = \begin{pmatrix} \mathbf{1}\epsilon - \mathbf{d} \\ \mathbf{1}\epsilon + \mathbf{d} \end{pmatrix}, \tag{2b}$$

$$\mathbf{z} = \left(\boldsymbol{\alpha}^+ \; \boldsymbol{\alpha}^- \; b^+ \; b^- \; \boldsymbol{\xi} \; \mathbf{u}\right)^T, \tag{2c}$$

$$\mathbf{c} = \left(\mathbf{1} \; \mathbf{1} \; 0 \; 0 \; \mathbf{2C} \; \mathbf{0}\right)^T, \tag{2d}$$

where $\mathbf{A} \in \mathbb{R}^{(2N)\times(4N+2)}$, $\mathbf{b} \in \mathbb{R}^{2N}$, $\mathbf{z}, \mathbf{c} \in \mathbb{R}^{4N+2}$. If we use the above equalities, then problem (1) is identical to the canonical form, and we can claim that the problem has been posed as an LP problem.

We claim that problem (1) is an original formulation for LP-SVR. In comparison with the $\nu-$LPR formulation by Smola, *et al.* [26] problem (1) (i) uses the canonical formulation, (ii) computes b, and $\mathbf{u}$ implicitly, (iii) does not compute ϵ implicitly, (iv) does not require the parameter ν, (v) promotes efficiency in the sense of using only one ξ, and (vi) is a lower dimensional problem.

In comparison with Mangasarian, *et al.* [14], problem (1) (i) uses the canonical formulation, (ii) computes b implicitly, (iii) does not compute ϵ implicitly, and (iv) does not require the parameter μ. By (iii) and (iv) we provide the experimenter with more control of the sparseness of the solution [31]. In this case sparseness means fewer number of support vectors.

Similarly, Problem (1) in comparison to Lu, *et al.* [18] our LP-SVR formulation (1) (i) uses the canonical formulation and (ii) computes b implicitly. By (ii) the linear program (LP) size is reduced by a factor of $N^2 + N$.

In comparison with the ℓ_1-norm LP-SVR formulation by Zhang, *et al.* [31] problem (1) does not require parameter δ and is more efficient in several ways: (i) uses only one ξ, (ii) avoids penalization of b, (iii) reduces computational efforts by forcing positivity in $\mathbf{u}$ which reduces the LP problem size by $2N^2 + 2N$, and (iv) is a smaller problem.

Using equalities (2a)-(2d), we can obtain the dual problem of (1) as follows:

$$\begin{aligned} \max_{\boldsymbol{\lambda}} \quad & \mathbf{b}^T\boldsymbol{\lambda} \\ \text{s.t.} \quad & \begin{cases} \mathbf{A}^T\boldsymbol{\lambda} + \mathbf{s} = \mathbf{c} \\ \qquad\qquad \mathbf{s} \geq \mathbf{0}, \end{cases} \end{aligned} \tag{3}$$

which is equivalent to the dual of a linear programming problem, where $\boldsymbol{\lambda}$ is a vector of dual variables defined over $\mathbb{R}^{2N}$, and $\mathbf{s}$ is a slack vector variable in $\mathbb{R}^{4N+2}$.

Similarly, for the primal (1) and dual (3), the KKT conditions are defined as follows:

$$\mathbf{A}^T\boldsymbol{\lambda} + \mathbf{s} = \mathbf{c}, \tag{4a}$$

$$\mathbf{A}\mathbf{z} = \mathbf{b}, \tag{4b}$$

$$z_i s_i = 0, \tag{4c}$$

$$(\mathbf{z}, \mathbf{s}) \geq \mathbf{0}, \tag{4d}$$

$$\text{for} \;\; i = 1, 2, \ldots, n,$$

where the equality $z_i s_i$ implies that one of both variables must be zero. This equality will be referred to as the *complementarity condition*. Note that the

KKT conditions depend on the variables $(\mathbf{z}, \boldsymbol{\lambda}, \mathbf{s})$, and if the set of solutions $(\mathbf{z}^*, \boldsymbol{\lambda}^*, \mathbf{s}^*)$ satisfy all the conditions, the problem is said to be solved. The set $(\mathbf{z}^*, \boldsymbol{\lambda}^*, \mathbf{s}^*)$ is known as a *primal-dual solution.*

2.2 Definitions

Definition 1 (Support Vectors) *Let $\mathcal{T} = \{\mathbf{x}_i, d_i\}_{i=1}^N$ be a training set; let $\mathbf{z}$ be a solution to problem (1); and let $d_j \equiv f(\mathbf{x}_j) = \sum_{i=1}^N \left(\alpha_i^+ - \alpha_i^-\right) k(\mathbf{x}_i, \mathbf{x}_j) + (b^+ - b^-)$ be the regression function for problem (1). Then,*

1. $\mathcal{V}_S = \{\mathbf{x}_i \ : \ d_i - \epsilon < f(\mathbf{x}_i) < d_i + \epsilon\}$ *defines the set of Saturated Support Vectors (SSVs).*
2. $\mathcal{V}_E = \{\mathbf{x}_i \ : \ f(\mathbf{x}_i) = d_i + \epsilon, \ \text{or} \ f(\mathbf{x}_i) = d_i - \epsilon\}$ *defines the set of Exact Support Vectors (ESVs).*
3. $\mathcal{V}_N = \{\mathbf{x}_i \ : \ f(\mathbf{x}_i) < d_i + \epsilon, \ \text{or} \ f(\mathbf{x}_i) > d_i - \epsilon\}$ *defines the set of Non-Support Vectors (NSVs).*
4. $\mathcal{V}_\alpha = \{\mathbf{x}_i \ : \ \alpha_i \neq 0\}$ *defines the set of Sparse Vectors (SPVs).*
5. $N = |\mathcal{V}_S| + |\mathcal{V}_E| + |\mathcal{V}_N|$.
6. $\mathcal{S} = \mathcal{V}_S \cup \mathcal{V}_E$, *means that the union of the SSVs and the ESVs is the set of Support Vectors (SVs).*
7. $\mathcal{A} = \{\alpha_i \ : \ \alpha_i \neq 0\}$ *denotes the set of Non-zero Coefficients of the decision function and of problem (1).*

2.3 Background

It is well known that SVR and support vector machines (SVM) formulations do not make assumptions about the probability distribution of the data. Nonetheless, each class ω_j, should have a conditional class distribution $p(\mathbf{x}|\omega_j)$, where $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^M$ is defined as an M-dimensional random variable which could be be estimated if enough data points were available. Estimating a multidimensional probability density function (PDF) is difficult but we could make some basic assumptions. First, we could assume that the data has a uni-modal distribution which implies that data-samples would cluster around the class mean and the further a point is from its mean, the lower its probability and could be expected to be located on the convex hull of the data sample we are analyzing. A more strict assumption would be to consider that the $p(\mathbf{x}|\omega_j)$ are multivariate Gaussian distributed. Under this assumption each $p(\mathbf{x}|\omega_j)$ could be modeled using only the sample mean $\boldsymbol{\mu}_{\mathbf{x}|\omega_j}$ and a covariance matrix $\boldsymbol{\Sigma}_{\mathbf{x}|\omega_j}$, that is $p(\mathbf{x}|\omega_j) \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{x}|\omega_j}, \boldsymbol{\Sigma}_{\mathbf{x}|\omega_j})$. It is also well known that we can use the squared Mahalanobis distance (MD)

$$D(\mathbf{x}_i) = (\mathbf{x}_i - \boldsymbol{\mu}_{\mathbf{x}|\omega_j})^T \boldsymbol{\Sigma}_{\mathbf{x}|\omega_j}^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_{\mathbf{x}|\omega_j}), \tag{5}$$

as a measure of the distance of a data point with respect to its mean.

Based on these assumptions we propose a method for finding the SV candidates by computing the $D(\mathbf{x}_i)$ for all $i = \{1, 2, \ldots, N\}$. Once all training vectors

are sorted by their MD to their respective mean, and saved into the sets $\mathcal{Z}_j$ for the j-th class, then we can form the initial working set $\mathcal{B}$ of size B_{ini} using the procedure described in Algorithm 1 (explained in the next section). We traverse elements of $\mathcal{Z}_{j,i}$ into to $\mathcal{B}$ until B_{ini} elements are added.

Algorithm 1 Mahalanobis Distance-Based Working-Set Selection for Large-Scale LP-SVR Training Speedup

Require: A training set $\mathcal{T}_\phi = \{\mathbf{x}_i, d_i\}_{i=1}^N$.
Require: A desired number of samples per class v.
for $j = 1$ to $|D|$ **do**
  Estimate parameters $(\boldsymbol{\mu}_{\mathbf{x}|\omega_j}, \boldsymbol{\Sigma}_{\mathbf{x}|\omega_j})$. ▷ Sample mean and variance.
  for $i = 1$ to N **do**
    Compute Mahalanobis distance $D(\mathbf{x}_i)_j$ with (5).
  end for
  Obtain indices $\mathcal{Z}_j$ corresponding to the sorted D_j. ▷ Descending order.
  for $i = 1$ to v **do**
    $\mathcal{Z}_{j,i} = \mathcal{Z}(i)_j$. ▷ In this case $B_{\text{ini}} = k \equiv v \times |\mathcal{D}|$.
  end for
end for
Ensure: Initial working set indices $\mathcal{B} \leftarrow \mathcal{Z}_{j,i}$.
Ensure: Initial fixed set indices $\mathcal{M} \leftarrow \{1, 2, \ldots, N\} \notin \mathcal{Z}_{j,i}$.

In this manner, the SVR could be trained faster if the first working-set $\mathcal{B}$ contains those k samples, thereby, speeding up the training process. A similar approach to ours is given by Zhou, *et al.* [33] in 2010; but again the authors approach is still based on class and subclass convex hulls, which makes it computationally expensive.

To explain the proposed approach, consider the following definitions: Let $\mathcal{D} = \{\omega_1, \omega_2, \ldots, \omega_j\}$ be the set of classes where j is the total number of classes. Let $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_j\}$ denote a set of indices, where $\mathcal{C}_j$ contains the indices of all those samples associated with the j-th class, $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset$ for all $i \neq j$, and $\mathcal{C} \equiv \{1, 2, \ldots, N\}$.

2.4 Within-Class Mahalanobis distance and Class-Convex Hull

To explain the ideas behind the procedure shown in Algorithm 1, we will be considering the case of all the samples $\mathbf{x}_i$ belonging to the j-th class, that is, all $i \in \mathcal{C}_j$. The same principles will apply to all classes.

One of the first steps is to estimate the parameters $(\boldsymbol{\mu}_{\mathbf{x}|\omega_j}, \boldsymbol{\Sigma}_{\mathbf{x}|\omega_j})$, *i.e.*, from observed events. Then the within-class MD from the i-th feature vector $\mathbf{x}_i$ to the center of the j-th class $\boldsymbol{\mu}_{\mathbf{x}|\omega_j}$ is defined as $D(\mathbf{x}_i)$ from (5). Next, we define $\mathcal{Z}_j$ as the set of indices corresponding to the ordered Mahalanobis distance samples of the $j-$th class computed with (5). The indices in $\mathcal{Z}_j$ correspond to ordered values in descending form, as shown in Figure 2.

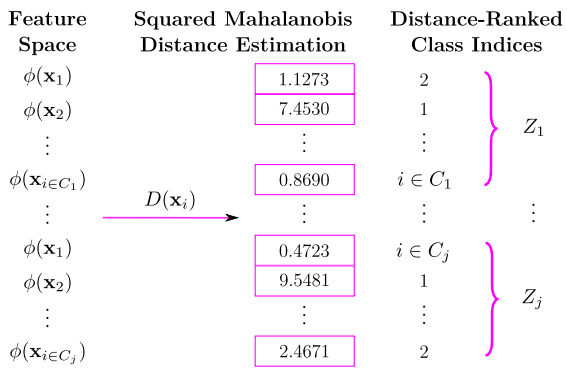

Fig. 2. Mahalanobis distance-ranking of class indices using feature vectors in either the input space or the kernel-induced feature space.

As mentioned before, we argument that the MD $D(\mathbf{x}_i)$ is related to the support vectors and the class convex hull (CCH), which is defined as follows:

$$\Theta(\omega_j) = \left\{ \sum_{i \in \mathcal{C}_j} \beta_i \mathbf{x}_i \ : \ i \in \mathcal{C}_j,\, \beta_i \in \mathbb{R},\, \beta_i \geq 0,\, \sum_{i \in \mathcal{C}_j} \beta_i = 1 \right\}, \tag{6}$$

where a number of $|\mathcal{C}_j|$ points in the form of $\sum_{i \in \mathcal{C}_j} \beta_i \mathbf{x}_i$ are the boundaries of the j-th class sample cloud. Then we can define the sets of indices corresponding to the convex hull of the j-th as $\mathcal{S} = \Theta(\omega_j)$. The algorithm that obtains the convex hull has complexity of $\mathcal{O}(N^{\frac{M}{2}})$, where M is the dimensionality of the feature vector. The complexity of the method proposed here has a complexity of $\mathcal{O}(L)$, where $L = \max\left[N \log N, \binom{M}{2}\right]$. This demonstrates that our model has lower complexity than those based on convex hulls. Now, we define a relationship between $\mathcal{Z}$, $\mathcal{S}$, and SV in Proposition 1.

Proposition 1 (SVs and Within-Class Distances) *Assume classes in $\mathcal{D}$ are linearly separable. Let $\mathcal{Z}_v = \{\mathcal{Z}(1), \mathcal{Z}(2), \ldots, \mathcal{Z}(v)\}$ denote the v maximum Mahalanobis distance indices. Similarly, let $\mathcal{Z}_{j,i} = \{\mathcal{Z}_{1,v}, \mathcal{Z}_{2,v}, \ldots, \mathcal{Z}_{j,v}\}$ be the set of v maximum Mahalanobis distance indices of all classes. Then*

1. *the maximum Mahalanobis distance samples indices contain the convex hull indices: $\mathcal{Z}_v \in \mathcal{S}$,*
2. *the maximum Mahalanobis distance samples indices contain the support vector indices: $\mathcal{Z}_{j,i} \in SV$,*

where v is an integer stating how many samples per class should be considered.

Proposition 1 states that the first v ranked MD indices $\mathcal{Z}_v$ contain the class convex hull indices $\mathcal{S}$ and, thus, contain the support vector indices. The integer v is bounded, $|\mathcal{D}| \leq v \leq |\mathcal{S}|$, and SV is as in Definition 1. Therefore, if the initial

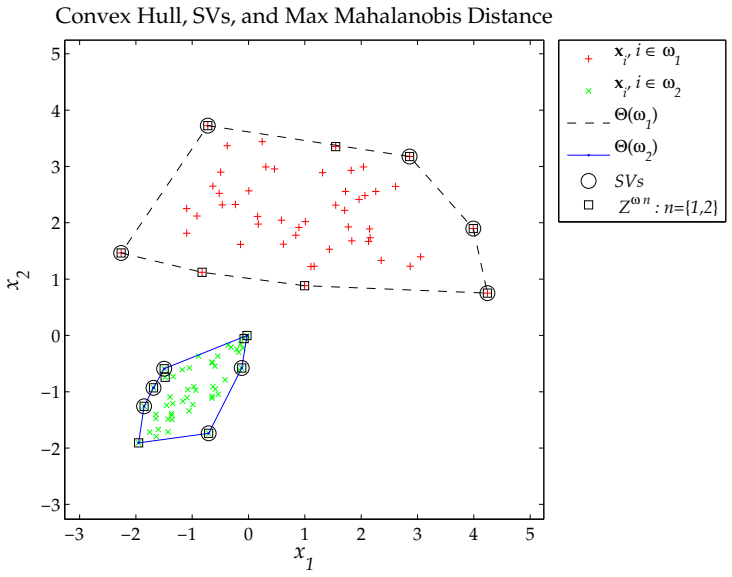

Fig. 3. Relationship between convex hull and maximum Mahalanobis distance for the two class problem. Here it is shown the original separable two class problem, class convex hull, support vectors, and k maximum Mahalanobis distance samples. Note that both SVs and Convex Hull match the k maximum Mahalanobis distance samples.

working-set is fixed to the indices in $\mathcal{Z}_{j,i}$, the training process will converge faster. This is mainly because if the support vectors are found at the very first iterations, the problem will be solved faster. Since $\mathcal{Z}_{j,i}$ is more likely (based on Mahalanobis distance information) to contain support vector indices, one can conclude that the training will be faster. We have found that a good value for v is the quotient between the initial working set size B_{ini} and the total number of classes: $v = \left\lceil \frac{B_{\text{ini}}}{|\mathcal{D}|} \right\rceil$. This choice of v was found empirically using the datasets discussed in the next section. This value v is used as input in Algorithm 1.

As an example, let us consider a case of a random variable $\mathbf{x} \in \mathbb{R}^2$. Then draw 50 samples that follows a multivariate normal distribution with parameters $\boldsymbol{\mu} = \begin{bmatrix} 1 & 2 \end{bmatrix}^T$ and $\boldsymbol{\Sigma} = \begin{bmatrix} 2 & 0 \; ; \; 0 & \frac{1}{2} \end{bmatrix}$ and assign these to class ω_1; draw 50 more from a multivariate copula [6] distribution with parameter $\rho = 0.8$, and assign these to class ω_2. The problem is shown in Figure 3 with the resulting convex hull. The associated support vectors are also shown in Figure 3. Let us remark that the Mahalanobis distance is associated with the spreadness of the class sample cloud, such that the most uncertain samples have the highest Mahalanobis distance, as shown in Figure 3. It is also important to remark that the highest Mahalanobis distance correspond to the lowest probability samples, which is also correlated to the support vectors as mentioned before. The next section addresses the speedup quantification and other numerical testing of Algorithm 1.

Table 1. Summary of the Dimensions and Properties of the Datasets.

Dataset	Classes	Features M	Training N	Reference
Ripley	2	2	250	[19]
Sonar	2	60	104	[7]
Wine	2	13	110	[12]
ADA	2	48	4,147	[9,21]
GINA	2	970	3,153	[5,21]
HIVA	2	1,617	3,845	[2]
NOVA	2	16,969	1,754	[2]
SYLVA	2	216	13,086	[2,5]
Iris	3	4	130	[15]
MODIS	4	4	374,566	[22]
Power Load	$\mathbb{R}$	8	35,064	[10]
Spiral	2	2	200	[30]
$f(x)$ =sinc(x)	$\mathbb{R}$	1	200	[20]
Synthetic S	3	2	3,000,000	–
Synthetic NS	3	2	3,000,000	–
$f(x)$ =sinc$(x) \times \pi$	$\mathbb{R}$	1	1 million	–

3 Experimental Results

To show the effectiveness and efficiency of the proposed algorithm, simulations were performed over different datasets. The summary of the properties of these datasets are shown in Table 1. Note that the simulations include classification in two and multiple classes, as well as regression problems. While the source of most of the datasets is referenced, the three synthetic datasets were generated as follows: "Synthetic S" is a non-linearly separable three-class problem whose classes are normally distributed; "Synthetic NS" is similar but the classes are non-separable; the "$f(x) = sinc(x) \times \pi$" consists of unevenly spaced points from the sinc function that are affected by multiplicative white Gaussian noise (WGN), making it very difficult to fit.

Here we analyze the speedup resulted of using the proposed sample selection algorithm using state of the art models and using benchmark datasets. The following paragraphs explain the results obtained.

3.1 Learning Speedup

Figure 4 depicts the behavior of the support vectors across iterations using the speedup strategy. The figure shows the number of support vectors, sparse support vectors, saturated support vectors, and exact support vectors. Note how the support vectors are found early in the learning process. If we compare Figure 4 on the left to the analysis on the right, we notice that most of the support vectors are found in earlier iterates, which is the goal of the strategy. We determined experimentally that in the average case, most of the sparse support vectors (SPV)

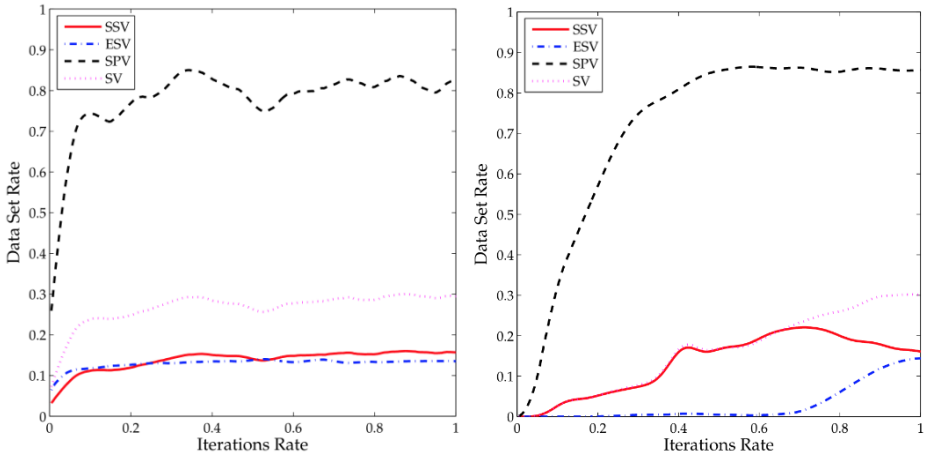

Fig. 4. Sample Selection. SVs as a Function of Iterations. SPV SSV ESV SV. (Left) With proposed sample selection. (Right) Without proposed sample selection.

are found in around 40% of the total iterations. In the other hand, Figure 4 tells that in around 20% of the total number of iterates.

Table 2 shows the total training time of the experiments without speedup. Compare against Table 3 that shows the total training time after using the speedup strategy. From these two tables we can observe that if strategy is used the total training time decreases, particularly as the problem size increases, as it was expected.

Besides a time reduction analysis, it is also important to observe if the total number of iterations is reduced if the speedup strategy is applied. Table 4 shows the total reduction of iterations, in percent, using the speedup strategy. Clearly, if the problem size is large, the reduction in number of iterations is also larger. A great benefit can be obtained of the speedup strategy, especially, if the class cloud is as close as possible to a multivariate Gaussian distribution within the kernel-induced feature space.

Another interesting thing to notice is that the percentage of iterations reduction seem to be superior in proportion to the learning time after speedup. This follows from noticing that, even if the support vectors are found at early iterations, still, the learning process has to perform several time-consuming decompositions and comparisons that might increment in size, specially, if the number of support vectors is large. However, since support vectors are found early at the learning process, there is no need to solve some of the subsequent sub-problems since the support vectors found will satisfy the KKT conditions of many of the sub-problems.

Table 2. Total training time without speedup (sec)

	Classifiers					
Dataset	LS SVM	LP-SVR	IncSVM	LSSVM	D. Reg. Trees	FFNN $[1, 20, 1]$
Ripley	9	16	8	4	–	4
Wine	6	4	6	3	–	5
ADA	75	174	–	4412	–	63
GINA	50	116	–	1403	–	48
HIVA	73	161	–	–	–	–
NOVA	25	47	258	–	–	–
SYLVA	247	495	–	–	–	190
Iris	6	4	3	3	–	3
Spiral	10	10	14	8	–	10
$f(x) = \text{sinc}(x)$	28	41	–	–	181	26
$f(x) = \text{sinc}(x) \times \pi$	9376	806	–	–	6933	602
Synthetic S	9349	794	–	–	–	–
Synthetic NS	9180	817	–	–	–	–
Avg.	2187	286	–	–	–	–

4 Computational Concerns

The primary concern of the speedup method is the computation of the covariance matrix $\boldsymbol{\Sigma}_{\mathbf{x}|\omega_j}$. In our implementation, the covariance matrix was estimated with the sample covariance matrix

$$\boldsymbol{\Sigma}_{\mathbf{x}|\omega_j} = \frac{1}{|\mathcal{C}_j| - 1} \sum_{i \in \mathcal{C}_j} (\mathbf{x}_i - \boldsymbol{\mu}_{\mathbf{x}|\omega_j})(\mathbf{x}_i - \boldsymbol{\mu}_{\mathbf{x}|\omega_j})^T \tag{7}$$

where $\mathbf{x}_i \in \mathbb{R}^M$ is some random vector with $|\mathcal{C}_j|$ realizations, and $\boldsymbol{\mu}_{\mathbf{x}|\omega_j} \equiv \mathcal{E}[\mathbf{x}_i]$ for all $i \in \mathcal{C}_j$. Clearly, $\boldsymbol{\Sigma}_{\mathbf{x}|\omega_j} \in \mathbb{R}^{M \times M}$, thus, problems with a very large number of variables *e.g.*, the NOVA dataset, cannot be resolved under current computational constraints. Therefore, some sort of feature reduction must be implemented to obtain the covariance matrix. In the case of the NOVA dataset, a large number of features are redundant or add no discriminant information and were eliminated without loss of generality.

Moreover, for the speedup process only, the kernel choice was also limited. The rule for selecting the kernel type is the following: If $N \geq 1000$ a polynomial kernel with degree $p = 1000$ is used, otherwise an RBF kernel is used. Since one of our goals is to deal with large-scale datasets most of the experiments used a polynomial kernel. The polynomial kernel is our second choice since it is known to be the second best after RBF kernels [17]. The degree of the polynomial kernel is directly related to the amount of data that can be efficiently handled for covariance matrix estimation purposes.

Table 3. Total training time with speedup (sec)

	Classifiers					
Dataset	LS SVM	LP-SVR	IncSVM	LSSVM	D. Reg. Trees	FFNN $[1, 20, 1]$
Ripley	8	9	7	4	–	4
Wine	1	2	4	3	–	5
ADA	71	74	–	3533	–	63
GINA	38	63	–	1191	–	48
HIVA	57	115	–	–	–	–
NOVA	17	19	234	–	–	–
SYLVA	246	230	–	–	–	190
Iris	2	2	2	2	–	3
Spiral	5	5	5	5	–	10
$f(x) = \text{sinc}(x)$	9	8	–	–	15	26
$f(x) = \text{sinc}(x) \times \pi$	5672	764	–	–	5597	602
Synthetic S	7038	467	–	–	–	–
Synthetic NS	8608	672	–	–	–	–
Avg.	1674	187	–	–	–	–

5 Conclusion

Within the context of kernel-induced feature space one can assume the data is (or is close to be) linearly separable and then compute the distances from each point to the center of the class cloud. This is done using the Mahalanobis distance. Since the support vectors (SVs) most likely lie on the class cloud boundaries or within the class convex hull, we can use the Mahalanobis distance to rank the training set, such that, the samples with the largest distances are used first as part of the working set. Experimental results suggest a reduction in the total training time, and a more dramatic decrease in the total iterations percentage. Results also suggest that, using the speedup strategy the support, the SVs are found early in the learning process. Furthermore, the speedup strategy was tested with other methods with similar results, suggesting that the proposed approach is not particular to LP-SVR but rather useful for other SV-based methods.

References

1. Bennett, K.P., Bredensteiner, E.J.: Duality and geometry in svm classifiers. In: ICML. vol. 2000, pp. 57–64 (2000)
2. Cawley, G.C.: Leave-one-out cross-validation based model selection criteria for weighted ls-svms. In: The 2006 ieee international joint conference on neural network proceedings. pp. 1661–1668. IEEE (2006)
3. Chen, X., Xiao, Y.: Geometric projection twin support vector machine for pattern classification. Multimedia Tools and Applications pp. 1–17 (2020)
4. Cherkassky, V., Ma, Y.: Practical selection of svm parameters and noise estimation for svm regression. Neural networks **17**(1), 113–126 (2004)

Table 4. Iterations reduction percentage after speedup.

	Classifiers					
Dataset	LS SVM	LP-SVR	IncSVM	LSSVM	D. Reg. Trees	FFNN $[1, 20, 1]$
Ripley	1	2	22	52	–	0
Wine	1	4	0	1	–	0
ADA	24	9	–	10	–	0
GINA	12	10	–	11	–	0
HIVA	39	9	–	–	–	–
NOVA	7	1	4	–	–	–
SYLVA	30	1	–	–	–	0
Iris	1	1	8	5	–	0
Spiral	15	12	4	7	–	0
$f(x) = \text{sinc}(x)$	5	13	–	–	1	0
$f(x) = \text{sinc}(x) \times \pi$	59	57	–	–	43	1
Synthetic S	23	51	–	–	–	–
Synthetic NS	27	37	–	–	–	–
Avg.	18.7	15.7	–	–	–	–

5. Collobert, R., Bengio, S.: Svmtorch: Support vector machines for large-scale regression problems. Journal of machine learning research **1**(Feb), 143–160 (2001)
6. Darsow, W.F., Nguyen, B., Olsen, E.T., et al.: Copulas and markov processes. Illinois journal of mathematics **36**(4), 600–642 (1992)
7. Gorman, R.P., Sejnowski, T.J.: Analysis of hidden units in a layered network trained to classify sonar targets. Neural networks **1**(1), 75–89 (1988)
8. Gu, X., Chung, F.l., Wang, S.: Fast convex-hull vector machine for training on large-scale ncrna data classification tasks. Knowledge-Based Systems **151**, 149–164 (2018)
9. Joachims, T.: Making large-scale svm learning practical. Tech. rep., Technical Report (1998)
10. Karsaz, A., Mashhadi, H.R., Mirsalehi, M.M.: Market clearing price and load forecasting using cooperative co-evolutionary approach. International Journal of Electrical Power & Energy Systems **32**(5), 408–415 (2010)
11. Keerthi, S.S., Shevade, S.K., Bhattacharyya, C., Murthy, K.R.: A fast iterative nearest point algorithm for support vector machine classifier design. IEEE transactions on neural networks **11**(1), 124–136 (2000)
12. Kinzett, D., Zhang, M., Johnston, M.: Using numerical simplification to control bloat in genetic programming. In: Asia-Pacific Conference on Simulated Evolution and Learning. pp. 493–502. Springer (2008)
13. Liu, Z., Liu, J., Pan, C., Wang, G.: A novel geometric approach to binary classification based on scaled convex hulls. IEEE transactions on neural networks **20**(7), 1215–1220 (2009)
14. Mangasarian, O.L., Musicant, D.R.: Large scale kernel regression via linear programming. Machine Learning **46**(1-3), 255–269 (2002)
15. McGarry, K.J., Wermter, S., MacIntyre, J.: Knowledge extraction from radial basis function networks and multilayer perceptrons. In: IJCNN'99. International Joint Conference on Neural Networks. Proceedings. vol. 4, pp. 2494–2497. IEEE (1999)

16. Mehr, A.D., Nourani, V., Khosrowshahi, V.K., Ghorbani, M.A.: A hybrid support vector regression–firefly model for monthly rainfall forecasting. International Journal of Environmental Science and Technology **16**(1), 335–346 (2019)
17. Mezghani, D.B.A., Boujelbene, S.Z., Ellouze, N.: Evaluation of svm kernels and conventional machine learning algorithms for speaker identification. International journal of Hybrid information technology **3**(3), 23–34 (2010)
18. Niu, B., Jin, Y., Lu, W., Li, G.: Predicting toxic action mechanisms of phenols using adaboost learner. Chemometrics and Intelligent Laboratory Systems **96**(1), 43–48 (2009)
19. Osuna, E., De Castro, O.: Convex hull in feature space for support vector machines. In: Ibero-American Conf. on Artificial Intelligence. pp. 411–419. Springer (2002)
20. Peng, X.: Tsvr: an efficient twin support vector machine for regression. Neural Networks **23**(3), 365–372 (2010)
21. Platt, J.C.: Using analytic qp and sparseness to speed training of support vector machines. In: Adv. in neural information processing systems. pp. 557–563 (1999)
22. Rivas-Perea, P.: Southwestern us and northwestern mexico dust storm modeling trough moderate resolution imaging spectroradiometer data: a machine learning perspective. Tech. rep., NASA/UMBC/GEST GSSP. (2009)
23. Rivas Perea, P.: Algorithms for training large-scale linear programming support vector regression and classification. The University of Texas at El Paso (2011)
24. Rivas-Perea, P., Cota-Ruiz, J.: An algorithm for training a large scale support vector machine for regression based on linear programming and decomposition methods. Pattern Recognition Letters **34**(4), 439–451 (2013)
25. Santamaría-Bonfil, G., Reyes-Ballesteros, A., Gershenson, C.: Wind speed forecasting for wind farms: A method based on support vector regression. Renewable Energy **85**, 790–809 (2016)
26. Smola, A., Scholkopf, B., Ratsch, G.: Linear programs for automatic accuracy control in regression. In: 1999 Ninth International Conference on Artificial Neural Networks ICANN 99.(Conf. Publ. No. 470). vol. 2, pp. 575–580. IET (1999)
27. Trzciński, T., Rokita, P.: Predicting popularity of online videos using support vector regression. IEEE Transactions on Multimedia **19**(11), 2561–2570 (2017)
28. Vapnik, V., Golowich, S., Smola, A.: Support vector method for function approximation, regression estimation, and signal processing. Advances in neural information processing systems **9**, 281–287 (1997)
29. Wang, D., Qiao, H., Zhang, B., Wang, M.: Online support vector machine based on convex hull vertices selection. IEEE Transactions on Neural Networks and Learning Systems **24**(4), 593–609 (2013)
30. Xu, Z., Huang, K., Zhu, J., King, I., Lyu, M.R.: A novel kernel-based maximum a posteriori classification method. Neural Networks **22**(7), 977–987 (2009)
31. Zhang, L., Zhou, W.: On the sparseness of 1-norm support vector machines. Neural Networks **23**(3), 373–385 (2010)
32. Zhong, H., Wang, J., Jia, H., Mu, Y., Lv, S.: Vector field-based svr for building energy consumption prediction. Applied Energy **242**, 403–414 (2019)
33. Zhou, X., Jiang, W., Tian, Y., Shi, Y.: Kernel subclass convex hull sample selection method for svm on face recognition. Neurocomputing **73**(10-12), 2234–2246 (2010)