AIMS Mathematics

http://www.aimspress.com/journal/Math

AIMS Mathematics, 6(12): 13931–13953.
DOI:10.3934/math.2021806
Received: 28 June 2021
Accepted: 15 September 2021
Published: 27 September 2021

Research article

A new parallel data geometry analysis algorithm to select training data for support vector machine

Yunfeng Shi[1], **Shu Lv**[1,*] **and Kaibo Shi**[2,3]

[1] School of Mathematical Sciences, University of Electronic Science and Technology of China, Chengdu 611731, China

[2] Yangtze Delta Region Institute (Huzhou), University of Electronic Science and Technology of China, Huzhou 313001, China

[3] School of Electronic Information and Electrical Engineering Chengdu University, Sichuan Chengdu 610106, China

* **Correspondence:** Email: lvshu@uestc.edu.cn.

Abstract: Support vector machine (SVM) is one of the most powerful technologies of machine learning, which has been widely concerned because of its remarkable performance. However, when dealing with the classification problem of large-scale datasets, the high complexity of SVM model leads to low efficiency and become impractical. Due to the sparsity of SVM in the sample space, this paper presents a new parallel data geometry analysis (PDGA) algorithm to reduce the training set of SVM, which helps to improve the efficiency of SVM training. The PDGA introduce Mahalanobis distance to measure the distance from each sample to its centroid. And based on this, proposes a method that can identify non support vectors and outliers at the same time to help remove redundant data. When the training set is further reduced, cosine angle distance analysis method is proposed to determine whether the samples are redundant data, ensure that the valuable data are not removed. Different from the previous data geometry analysis methods, the PDGA algorithm is implemented in parallel, which greatly saving the computational cost. Experimental results on artificial dataset and 6 real datasets show that the algorithm can adapt to different sample distributions. Which significantly reduce the training time and memory requirements without sacrificing the classification accuracy, and its performance is obviously better than the other five competitive algorithms.

Keywords: support vector machine; sample reduction; geometry analysis; Mahalanobis distance; parallel

Mathematics Subject Classification: 68T09

1. Introduction

In the era of big data, information presents explosive growth. How to find valuable information from large-scale data shows great importance. In recent years, the research on classifier of large-scale data has become a hot research field [1, 2]. Many classifiers, such as Naive Bayes [3, 4], Artificial Neural Network [5], Decision Tree and Random Forest [6], Logical Regression [7], K-Nearest Neighbor [8] and Support Vector Machine [9] have been used for the classification of big data. Among these classifiers, support vector machine (SVM) [10–12] proposed by Vapnik has been widely used due to its solid theoretical basis. Based on VC dimension theory and minimum structural risk function, SVM aims at maximizing the margin between the two classes, and thus has strong generalization ability. Bhowmik et al., [13, 14] mentioned that SVM achieved better classification accuracy than other supervised learning methods. The research results [15, 16] showed that the average performance of SVM was even equal to that of deep learning. This is why SVM has been widely concerned in recent years.

However, it is impractical to use SVM for the classification of large-scale data, because the key to training SVM is to solve a quadratic programming problem (QP), which is dependent on the size of training dataset greatly. Theoretically, time complexity of training SVM is $O\left(m^3\right)$ and space complexity is $O\left(m^2\right)$ [17]. This results in slow training of SVM even on some medium-size datasets [12]. This brings great challenges to the classification of large-scale data by SVM [18]. In recent years, a number of methods have been developed to reduce the computational complexity of SVM. These algorithms can be classified into three categories including techniques which (i) decomposition of QP problems, (ii) parallel training and (iii) sample reduction.

(i) Decomposition of QP problems [19]: These methods decompose large QP problem into a series of small QP problems by making use of the sparsely characteristic [20] of SVM solutions. By solving the subproblem iteratively, the calculation of whole kernel matrix can be avoided, thus requirements of memory and solution time are reduced. Among these methods, the most famous one is the Sequence Minimum Optimization (SMO) algorithm [21, 22], which only needs to optimize two variables in each iteration and has a very fast convergence rate. LIBSVM [23], one of the most popular SVM implementations, is proposed based on SMO algorithm. With LIBSVM, the time complexity of SVM training was reduced to $O\left(m^2\right)$.

(ii) Parallel training: The basic idea of most SVM parallel training methods is to divide training set into a series of independent subsets, and then train SVM simultaneously on different processors, thus to filter out the support vectors (SVs) of each subset. Finally use these SVs to train the real SVM decision plane, as shown in [24, 25]. The parallelization training idea of Scholkopf et al., [26] is different. It approximates kernel matrix of SVM to a block diagonal matrix, so that the original optimization problem can be decomposed into hundreds of sub-problems that are easy to be solved in parallel.

(iii) Sample reduction: In most cases, the solution of SVM is determined by a small subset of training data, which are called support vectors (SVs) [27], and the number of SVs is much less than samples in whole training set. Sample reduction methods [28] reduce time and memory requirement of SVM by eliminating redundant data that are unlikely to be SVs and retaining only samples that may be SVs.

Among the three types of techniques, decomposition of QP problems are applied to either reduce the complexity of the underlying optimization problem, or to handle the optimization process more

efficiently. However, these approaches still induce the problem of high memory complexity in big data problems [29]. Parallel training methods [30] require multiple iterations in the process of filtering SVs, which introduce additional memory burden and reduce classification accuracy. Therefore, these two methods are not suitable for problems with large-scale datasets. In contrast, sample reduction techniques are considered to be the most straightforward and effective ways to use SVM for big data.

There exist five main groups of approaches for sample reduction, including data geometry analysis methods, neighborhood analysis methods, evolutionary methods, active learning methods and random sampling methods. These algorithms seems to have certain shortcomings and limitations. For example, data geometry analysis methods usually require clustering [31], which can be time-consuming and need to determine the number of clusters manually. Neighborhood analysis methods [32] usually need to calculate all distances between samples in training set, which has high time complexity and sensitive to noise. Genetic algorithm [33] is one of the important evolutionary methods for SVM sample reduction. However, it is a challenging to effectively determine the search space and ensure appropriate convergence speed, which is also the biggest disadvantage of genetic algorithm. Active learning algorithms can also be used to select training samples [34, 35], but these methods usually require multiple iterations and take a long time for large-scale datasets. Random sampling methods are fast, but that ignore the relationship between samples, which may lead to a lot of useful information not being extracted.

In this paper, a new parallel data geometry analysis(PDGA) algorithm was proposed to select training data for SVM, to solve the problems of long execution time and insufficient ability to maintain classification accuracy of existing algorithms. This method extracts redundant samples from training set that cannot be SVs, and ensures that potential SVs are not eliminated, to reduce the size of training set as much as possible. The properties of this algorithm are summarized as follows.

1). The non-SVs removal is more thorough, will not be affected by the correlation and dimensional differences of data different attributes, and will not increase the risk of SVs being removed by mistake. It is more stable and reliable;
2). Different from other geometric analysis methods, this algorithm processes the noise data in the process of sample reduction, which makes the training of SVM more noise resistant;
3). For the case that data distribution shape is not convex or the centroid is not in geometric center, this algorithm can also well extract the potential SVs located on the boundary;
4). This algorithm can be carried out in parallel, and the time complexity of proposed algorithm increases approximately linearly with sample size, which improves the efficiency of training SVM on large-scale datasets.

The rest of this paper is organized as follows. In Section 2, we provide a brief review of SVM and sample reduction algorithms. We introduce the basic theory of the proposed algorithm and analyze time complexity in Section 3. In Section 4, we experiment the proposed algorithm on artificial dataset and 6 real datasets, and compare it with the most competitive algorithms. Finally, we summarize the paper and discuss future work in Section 5.

2. A brief review of SVM and sample reduction algorithms

2.1. Support vector machine

SVM is a classical binary classification algorithm. Take the linear binary classification problem as an example, for dataset $\{(x_i, y_i) \mid x_i \in R^n, y_i \in \{-1, +1\}, i = 1, 2, \cdots, m.\}$, where x_i is a n-dimensional

vector, y_i is the category label of x_i. The essence of SVM is to find a hyperplane $\omega^T x + b = 0$ in a n-dimensional space that can separate different types of data as much as possible. According to the theory of SVM, the problem of finding the optimal hyperplane is transformed into a QP problem as follows:

$$\begin{aligned} &\min \varphi(\omega) = \tfrac{1}{2}\|\omega\|^2 \\ &\text{s.t. } y_i\left(\omega^T x_i + b\right) \geq 1, i = 1, 2, \cdots, m. \end{aligned} \tag{1}$$

If the training data is not linear separable, in order to tolerate some samples that do not meet the constraint, relaxation variables ξ need to be introduced, and the problem to be optimized is accordingly modified as

$$\begin{aligned} &\min \varphi(\omega, \xi) = \tfrac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{m}\xi_i \\ &\text{s.t.} \quad y_i\left(\omega^T x_i + b\right) \geq 1 - \xi_i, i = 1, 2, \cdots, m, \\ &\quad \xi_i \geq 0, i = 1, 2, \cdots, m, \end{aligned} \tag{2}$$

where $C > 0$, is used to control the cost of violating the constraint. By introducing Lagrange multipliers, the dual form of Eq 2 minimization problem is equivalent to

$$\begin{aligned} &\max L(\alpha) = \sum_{i=1}^{m}\alpha_i - \tfrac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j\left(x_i^T x_j\right) \\ &\text{s.t. } 0 \leq \alpha_i \leq C, i = 1, 2, \cdots, m, \\ &\sum_{i=1}^{m} y_i\alpha_i = 0, \end{aligned} \tag{3}$$

where α is a m-dimensional vector, and α_i is its component, which is the i-th Lagrange multiplier. By solving quadratic convex programming problem of Eq 3, the optimal decision function of SVM can be obtained as

$$f(x) = \operatorname{sgn}\left(\sum_{i=1}^{m}\alpha_i y_i\left(x_i^T x\right) + b\right), \tag{4}$$

where α_i corresponds to the solution of the QP problem, and b is the optimal offset of the hyperplane.

For nonlinear data in original space, SVM implicitly maps vectors to a high-dimensional feature space with the help of kernel function [8], and searches for the optimal hyperplane in this space, to solve the linearly unfractionable problem in original space. In the high-dimensional feature space mapped by kernel function, the optimal SVM hyperplane becomes

$$f(x) = \operatorname{sgn}\left(\sum_{i=1}^{m}\alpha_i y_i K\left(x_i, x\right) + b\right), \tag{5}$$

where $K\left(x_i, x_j\right) = \phi\left(x_i\right)^T \phi\left(x_j\right)$, $\phi : \mathrm{R}^n \to F$ is a mapping of vector x from the input space R^n to the eigenspace F. With the help of kernel function, we do not need to find specific mapping expression, only need to calculate inner product of two eigenvectors $\phi\left(x_i\right)$ and $\phi\left(x_j\right)$ by kernel function.

The key to training SVM is to solve the QP problem of Eq 3, whose time complexity is $O\left(m^3\right)$ and memory complexity is $O\left(m^2\right)$ [17], where m is the number of samples in training set. In fact, there is only a small part of $\alpha_i > 0$ by solving Eq 3, and the corresponding samples x_i are called support vectors (SVs), which can actually define the optimal hyperplane. Most of $\alpha_i = 0$, and the corresponding samples are non-support vectors (non-SVs), which make no contribution to the optimal hyperplane of SVM training, but waste memory and increase training time. In other words, the hyperplane trained by SVs alone is completely equivalent to the hyperplan trained by all data. In recent years, in order to accelerate the training speed of SVM, there are many methods devoted to

extracting SVs or removing non-SVs from training set. Such methods are called sample reduction [36] or instance selection [37].

2.2. Sample reduction for SVM

Simple random sampling methods (SRS) [38] are the most direct ways to reduce training samples. These methods randomly extract a certain proportion of samples from training set to train SVM, to improve the training efficiency of SVM. Among them, uniform random sampling [39] may be the most robust method to reduce training samples. However, these methods are not reliable, because they ignores the structural information between samples, which lead to uncertain results [40].

The neighborhood analysis methods are based on the fact, that only samples belonging to different categories and located near the hyperplane are likely to become SVs [41]. The purpose of neighborhood analysis is to extract data point pairs that are close to each other and belong to different categories. Compressed Nearest Neighbor algorithm (CNN) [42] is one of the neighborhood analysis methods. However, CNN needs to calculate all distances between samples, the time complexity is $O\left(m^2\right)$, and CNN is not noise tolerant. Shin et al., [43] proposed a method to select samples of overlapping regions near decision boundaries, because SVs generally exists in different categories of overlapping regions. Jiantao et al., [44] proposed a fuzzy clustering method to select samples existing in overlapping regions. However, these two approaches do not perform well when the class distributions do not overlap.

There are few evolutionary methods to reduce training set for SVM, and genetic algorithm [33] is one of the most important methods. It evolves a group of solutions (individuals), and the fitness of individuals corresponds to the classification accuracy of SVM. However, it is a challenging problem of how to effectively determine search interval of genetic algorithm and speed up convergence rate. Pighetti et al., [45] enhanced genetic evolution by using a locality sensitive hashing method, which found the nearest vector in training set for all generated vectors in the optimization process and applied it to solve the multi-classification problem. Although this method is novel, it did not provide a stop criterion for the optimization of multi-classification tasks.

Active learning methods [46] are a group of semi supervised learning approaches, which can also be used to reduce the size of SVM training set. They initially divide the training set into a labeled dataset and an unlabeled dataset (corresponding to SVs and non-SVs). Firstly, the classifier is trained with the tagged set. According to the pre-set criteria, the selected unlabeled samples are labeled and added to the tagged dataset. Then, the classifier is retrained with the updated tagged set, and the iteration continues until the predefined conditions are satisfied. Wang et al., [47] proposed an active learning algorithm for SVM sample selection based on neighborhood entropy measure. On the basis of tagged samples, they introduced the concept of neighborhood entropy to analyze the uncertainty of untagged samples. This model improves the classification ability of SVM. However, active learning algorithms have shortcoming in the whole dataset could be scanned for many times, thus the convergence time may be very long.

Data geometry analysis methods [48] exploit the information about training set structure to extract potential SVs. These methods generally calculate the distance from each vector to its centroid of the category to determine whether the vector is located on the interior or the boundary. Samples on the boundary have a high probability of becoming SVs. These methods can be roughly classified into two groups, the first encompasses clustering-based techniques, whereas the second contains non-clustering algorithms.

Clustering-based algorithms have been intensively studied for selecting refined training sets, Wang

and Shi [49] proposed an SR-DSA algorithm, which first uses agglomerative hierarchical clustering (AHC) to cluster each category separately. Then calculate the Mahalanobis distance from the vector of each cluster to its centroid, and delete the "inner points" which are very close to the centroid. Finally, remove "exterior points" that are distant from the opposite class. This method can maintain the classification accuracy well while reducing training samples, but it has two disadvantages, one is the time complexity of clustering is relatively high, the other is both the number of clustering and the proportion of retained samples need to be determined artificially, which depends on experience heavily. These are also common disadvantages of clustering based methods.

In the non-clustering-based methods, the SE algorithm proposed by Liu et al., [50] is a very competitive technique, which can adapt to the case where the centroid is not located in the geometric center of this class due to uneven distribution of instances in vector space. However, too many parameters introduced by this method artificially may lead to too many iterations of sample reduction and increase the time complexity. In addition, this algorithm uses Euclidean distance to measure the distance from each sample to its centroid, which cannot reflect the structural information of the sample distribution well, and is not applicable to samples with large differences in the distribution range of each attribute.

Data geometry analysis methods are a fast and direct sample reduction techniques, which have attracted the attention of many researchers. In general, it is very difficult to extract the SVs directly from training set, mainstream geometric analysis methods usually extract redundant samples that are unlikely to become SVs. However, most existing geometric analysis methods have the limitations of high time complexity or insufficient removal ability of non-SVs. We proposed a new approach to reduce the SVM training samples. Firstly, both outliers and non-SVs near the centroid are removed based on Mahalanobis distance, and then the non-SVs near the boundary are further removed by cosine angle distance analysis, to eliminate redundant data to the greatest extent. Since this method can be implemented in parallel, the efficiency of sample reduction was greatly improved.

3. Sample reduction of parallel data geometric analysis for support vector machine

Some algorithms compare the Euclidean distances from different samples to the centroid to determine whether the samples are located in the interior or the boundary of this class, which is not stable. Because Euclidean distance is easily affected by the dimensionality of different attributes, without considering the overall distribution of training set. If the centroid is not located in the geometric center of this class, that may cause some SVs at the boundary are discarded. As shown in Figure 1, points A, B and C are all located on the boundary of class distribution and are likely to become SVs. They should be retained with equal probability in the sample reduction process. If compare their Euclidean distance to the centroid O, the probability of discarding point A is greater than that of point B when the radius of the reducing sphere increases, because the data is more distributed in the direction of e_1. Point B has a higher probability of being discarded than C, because the centroid is not located in the geometric center of this class, resulting in point C being farther away from the centroid, which is obviously unreasonable.

Compared with Euclidean distance, Mahalanobis distance is a more reasonable distance measure, because it is calculated by the first and second order statistical information of samples, which implies the structural information of samples. Mahalanobis distance can eliminate the influence of dimensional difference and linear correlation of different attributes of samples.

In our algorithm, Mahalanobis distance is introduced to measure the distance from each sample to

its centroid. On this basis, we propose a method to identify outlier points and non-SVs simultaneously. This section first introduces the theory of Mahalanobis distance, presents a criterion for identifying outlier points and non-SVs simultaneously. Then explains how to calculate the cosine angle distance between samples. Finally summarizes the implementation steps of our algorithm, and analyzes the time complexity.

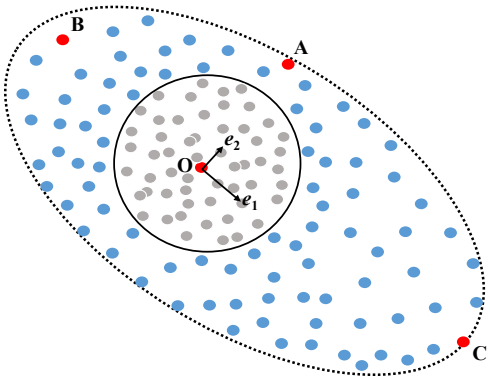


Figure 1. Redundant samples are deleted based on the data geometry analysis method. Note: Point O represents the centroid of this class, and the samples in the spherical region are discarded.

3.1. Mahalanobis distance

Let X be a $m \times n$ sample matrix, containing m random observations $x_i, i = 1, 2, \cdots, m$. n represents the number of features of each sample. The definition of Mahalanobis distance is

$$d^2(x_i, X) = (x_i - \mu)\Sigma^{-1}(x_i - \mu)^T, \tag{6}$$

where $d^2(x_i, X)$ is the square of Mahalanobis distance from sample x_i to population X, μ is the mean vector of sample matrix X, $\mu = \frac{1}{m}\overrightarrow{1}^T X$, $\overrightarrow{1}$ denotes a m-dimensional all one vector. Σ is the covariance matrix of sample matrix X, $\Sigma = \frac{1}{m}X^T X - \frac{1}{m^2}X^T\overrightarrow{1}^T\overrightarrow{1}X$. Since Σ is a real symmetric and positive semidefinite matrix, it can be orthogonally similar diagonalized, that is, there exists an orthogonal matrix P, which makes the

$$\Sigma = P\Lambda P^T, \tag{7}$$

where $P = [e_1, e_2, \cdots, e_n]$ satisfy $e_i^T e_k = 0, i \neq k, i, k = 1, 2, \cdots, n$. Λ is a diagonal matrix, the elements λ_i on the diagonal are the eigenvalues of Σ, and $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$. The Mahalanobis distance in Eq 6 is rewritten as

$$d^2(x_i, X) = (x_i - \mu)P\Lambda^{-1}P^T(x_i - \mu)^T. \tag{8}$$

If the first d eigenvalues greater than 0 are selected, accordingly, the orthogonal matrix $P = [e_1, e_2, \cdots, e_d]$, let $z_i = (x_i - \mu)P$, then Eq 8 becomes

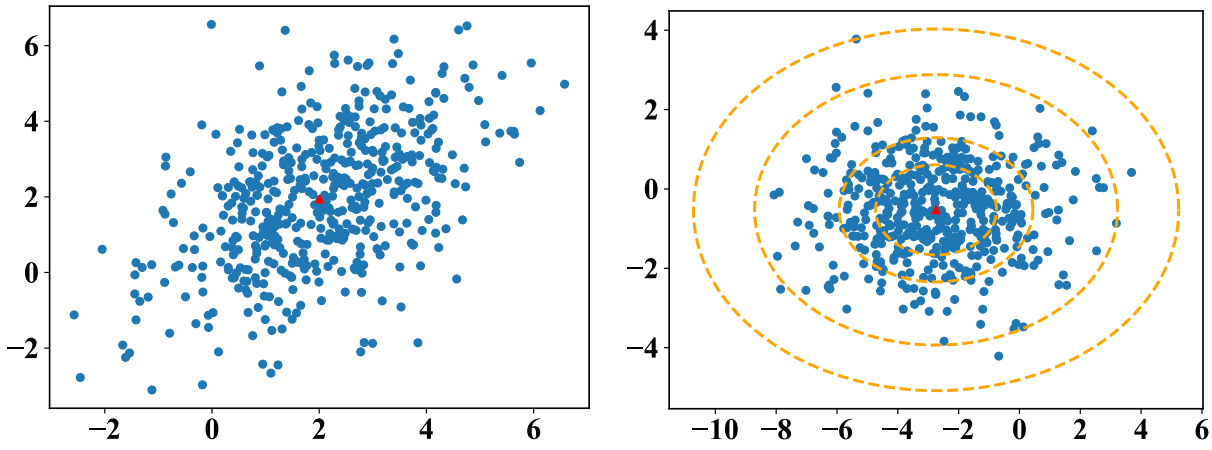
$$d^2(x_i, X) = z_i\Lambda^{-1}z_i^T = \sum_{j=1}^{d}\frac{z_{ij}^2}{\lambda_j}. \tag{9}$$

It can be seen that the orthogonal matrix P projects x_i to another d-dimensional space, and the attributes of the projected samples are orthogonal to each other, eliminate the influence of dimensional difference and linear correlation of different attributes of X. We may as well abbreviate

the Mahalanobis distance from x_i to the X as d_i and further transform Eq 9 to obtain

$$\frac{z_{i1}^2}{(d_i\sigma_1)^2} + \frac{z_{i2}^2}{(d_i\sigma_2)^2} + \cdots + \frac{z_{id}^2}{(d_i\sigma_d)^2} = 1, \tag{10}$$

where $\lambda_j = \sigma_j^2$, it also represents the sample variance of $(X - \mu)P$ on the j-th dimension. Under the measure of Mahalanobis distance, the projection of sample x_i to x_iP must be located on the hyperellipsoid with centroid μP and axis $d_i\sigma_1, d_i\sigma_2, \cdots, d_i\sigma_d$. For different samples in X, $\sigma_1, \sigma_2, \cdots, \sigma_d$ is fixed, so d_i can also be regarded as the generalized radius of the hyperellipsoid. As shown in Figure 2, where Figure 2(a) is the distribution of original dataset, and Figure 2(b) is the distribution of the projected dataset. The data are distributed as hyperellipsoids, and Mahalanobis distances from data points on the same hyperellipsoid to the centroid are equal. Mahalanobis distance can better adapt to the shape of data distribution.



(a) Original data distribution shape

(b) Data distribution under Mahalanobis distance metric

Figure 2. The effect of Mahalanobis distance on data conversion.

3.2. Recognition of Non-SVs and multivariate outliers

The closer a sample is to its centroid, the less likely it is to be a SV [49], so we can determine whether a sample is redundant data by calculating the Mahalanobis distance from the sample to its centroid. However, it should be noted that with the removal of non-SVs, the influence of some outliers on the classification plane will gradually increase, resulting in over-fitting of outliers, which will greatly reduce the classification accuracy. Therefore, it is necessary to remove outliers before reducing dataset.

Among some popular multivariate outlier detection algorithms, outlier detection based on Mahalanobis distance is an important method. Inspired by Leys et al., [51], we propose a method based on Mahalanobis distance that can simultaneously identify outliers and non-SVs. The details of the method are as follows.

Given a training dataset $D = \{(x_i, y_i) \mid x_i \in R^n, y_i \in \{1, 2, \cdots, C\}, i = 1, 2, \cdots, m.\}$, x_i is a sample of D, y_i is the class label of x_i. Assuming that each sample can only belong to one category, then D can be expressed as $D = D_1 \cup D_2 \cup \cdots \cup D_C$ according to different labels, where $D_1, D_2, \cdots, D_C$ represent C subsets of different categories. Correspondingly, $X_1, X_2, \cdots, X_C$ are used to represent the sample matrix, where the size of X_c is $m_c \times n$, m_c represents the number of samples in D_c, and n represents the number of features of each sample. Let μ_c be the centroid of X_c, and Σ_c be the covariance matrix

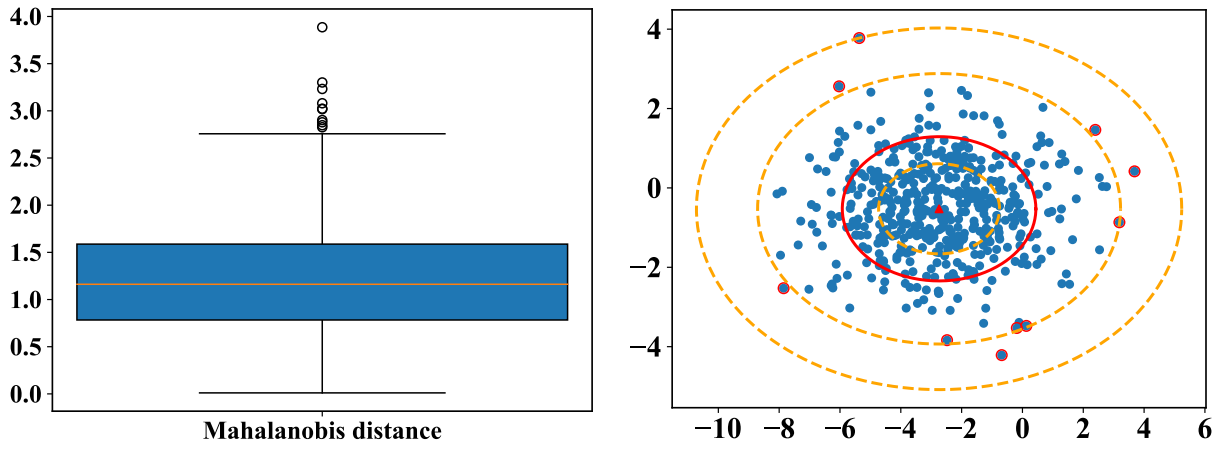
of X_c. For a data point $(x_i, y_i) \in D_c, x_i \in R^n, y_i = c$, the Mahalanobis distance between x_i and the population X_c can be calculated according to Eqs 7 and 8. After sorting by Mahalanobis distance, the dataset is represented by $D_c = \{sx_1, sx_2, \cdots, sx_{m_c}\}$. To simplify notation, we'll still use d_i to represent the Mahalanobis distance from sx_i to X_c and satisfie $d_1 \leq d_2 \leq \cdots \leq d_{m_c}$.

We introduce the concept of quantile to identify outliers. The Mahalanobis distance from each point to its centroid is taken as the measure of the degree of outlier. The outliers are defined as

$$sx_{\text{outlier}} : d_{\text{outlier}} > d_{[0.75m_c]} + whis\left(d_{[0.75m_c]} - d_{[0.25m_c]}\right), \tag{11}$$

where $[*]$ represents the integer part of $*$, and whis is a threshold, indicating how much we tolerate outliers. In general, sx_{outlier} represents outliers when whis is 1.5, and extreme outliers when whis is 3. The corresponding Mahalanobis distance of each point can be represented in the box diagram, as shown in Figure 3(a). The points beyond the upper whisk are regarded as outliers, which correspond to the red circle data in Figure 3(b). The number of outliers removed is represented by δ_c.

Meanwhile, we removed points which close to centroid from dataset, as shown in the points inside the red solid line in Figure 3(b). The number of samples to be removed is $\text{num}_c = \xi m_c - \delta_c$, where ξ is a parameter given by user to control the removal proportion. The dataset after removal becomes $D'_c = \{sx_{num_c+1}, sx_{num_c+2}, \cdots, sx_{m_c-\delta_c}\}$, that is, $m'_c = (1-\xi)m_c$. Whether to remove internal redundant points or outliers, it is necessary to calculate the Mahalanobis distance from samples to its centroid. The identification of outliers and non-SVs based on Mahalanobis distance not only helps to remove the noise data steadily, to maintain or even improve the classification accuracy, but also does not need to increase the training time.



(a) Mahalanobis distance box diagram of outliers

(b) Outliers and non-SVs that should be removed

Figure 3. Simultaneously identifying outliers and non-SVs based on Mahalanobis distance.

3.3. Cosine angle distance analysis

For the class distribution is non-convex, if the redundant samples are removed by constructing a reduction sphere, some useful samples may be deleted [50]. In our algorithm, we introduce an artificially given variable ξ to control the percentage of outliers and internal non-SVs being removed, this parameter should be chosen carefully, but this may lead to incomplete extraction of redundant samples. In order to solve this problem, we propose a method of cosine angle distance analysis between samples to determine whether the remaining samples are on the boundary. The basic idea

is to judge whether there are samples on the line between the outermost sample and the centroid. If there are samples, it means that these samples cannot be located on the boundary of class distribution and are unlikely to become SVs, as shown in Figure 4.

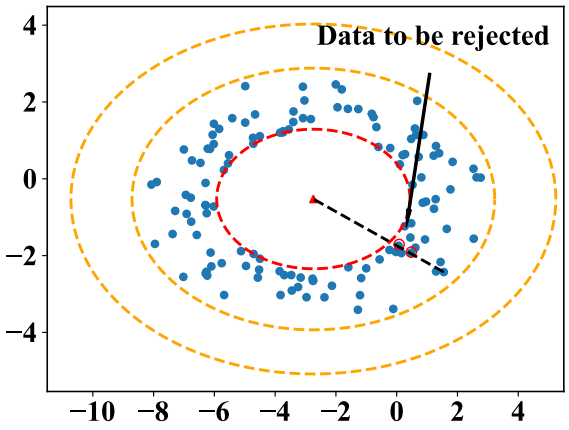


Figure 4. The samples on the line between the centroid and the outermost point should be removed as redundant samples.

After outliers and internal redundant data are deleted based on Mahalanobis distance, the training set D_c labeled with class c is reduced to D_c', then, the formula for calculating the cosine angle distance between samples in D_c' is

$$\theta_{ij} = \cos^{-1}\left(\frac{(x_i - \mu_c)\left(x_j - \mu_c\right)^T}{\|x_i - \mu_c\|\left\|x_j - \mu_c\right\|}\right), \tag{12}$$

where x_i, x_j are any two different training samples in D_c', μ_c is the centroid of the training set D_c, θ_{ij} represents the angle between vector $x_i - \mu_c$ and $x_j - \mu_c$ and $\cos^{-1}(.)$ represents the inverse function of cosine value. If $\theta_{ij} = 0$ and $d_i > d_j$ (d_i, d_j are the Mahalanobis distances from x_i, x_j to the centroid), it means that x_j is located between the line connected by x_i and μ_c, thus cannot be located on the boundary of class distribution. Therefore, it should be removed from the training set. To simplify the calculation, let's directly calculate the cosine distance between the vector $x_i - \mu_c$ and $x_j - \mu_c$, where $\theta_{ij} = 0$ is equivalent to $\cos\theta_{ij} = 1$,

$$\cos\theta_{ij} = \frac{(x_i - \mu_c)\left(x_j - \mu_c\right)^T}{\sqrt{(x_i - \mu_c)(x_i - \mu_c)^T}\sqrt{\left(x_j - \mu_c\right)\left(x_j - \mu_c\right)^T}}. \tag{13}$$

After cosine angle analysis, the sample points left are shown in Figure 5. The cosine angle analysis can not only minimize the training samples, but also ensure that SVs located on the boundary will not be deleted.

3.4. Sample reduction algorithm

The sample reduction of parallel data geometric analysis algorithm (PDGA) is summarized as follows and gives the pseudo code of the algorithm.

Step1. Computing Mahalanobis distance.
For training dataset D_c with class label c, the Mahalanobis distances from all the samples to the population are calculated by using Eqs 7 and 8.

Step2. Remove non-SVs and outliers.
Sorting samples in D_c based on the Mahalanobis distances from small to large. Then, given the whis and ξ to determine how many outliers and non-SVs to delete. After the outliers and non-SVs are removed, the dataset becomes D'_c.

Step3. Cosine angle distance analysis, delete the internal non-SVs which close to the boundary.
Calculating the cosine distance between any two samples in D'_c based on Eq 13. If there is $\cos\theta_{ij} = 1$, and $d_i > d_j$, then remove x_j from D'_c.

Step4. Repeat Step 3 until all points does not meet the condition, and the training set is updated to D''_c.

Step5. Repeat Step 1 to Step 4, until all categories of training set are processed.

Step6. The reduced training dataset $D'' = D''_1 \cup D''_2 \cup \cdots \cup D''_C$ is obtained and used to train SVM.

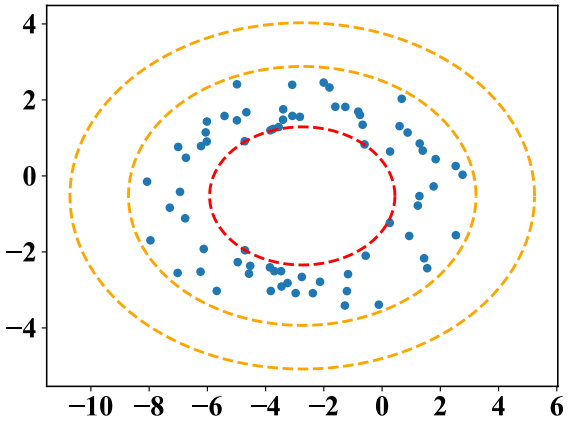


Figure 5. After cosine angle analysis, the boundary points which are not collinear with each other are left as training samples.

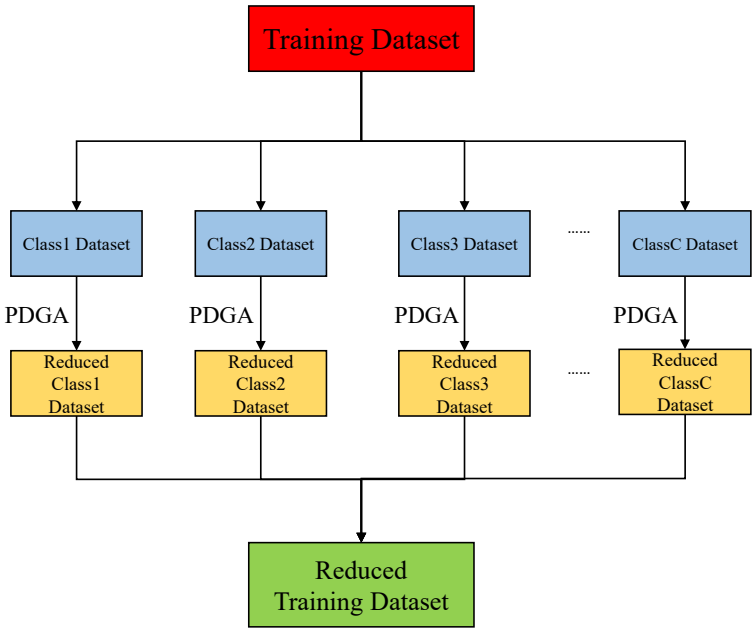


Figure 6. Parallel processing mode of PDGA algorithm.

It is noted that this algorithm processes training set of each category separately, which has two significant advantages. On the one hand, it is easy to apply the algorithm to multi-classification datasets. On the other hand, the whole training set can be processed in parallel according to different class label, which can significantly accelerate the training speed on large-scale datasets. The parallel processing pattern for this algorithm is shown in Figure 6.

The pseudo code of the algorithm is shown in Algorithm 1. In the parallel processing mode, the input dataset is allocated to different CPUs according to categories for processing at the same time. This mode is more efficient than previous data geometric structure analysis methods.

Algorithm 1 A new sample reduction of parallel data geometric analysis algorithm (PDGA).

Input: Training dataset D_c, whis, ξ;
Output: Reduced training dataset D_c'';

Compute $\mu_c = \frac{1}{m_c}\overrightarrow{1}_c^T X_c$, $\Sigma_c = \frac{1}{m_c}X_c^T X_c - \frac{1}{m_c^2}X_c^T\overrightarrow{1}^T\overrightarrow{1}X_c$, $\Sigma_c = P_c\Lambda_c P_c^T$;
for $i = 1$ to m_c **do**
    $d_i = \sqrt{(x_i - \mu_c)\, P_c\Lambda_c^{-1}P_c^T\,(x_i - \mu_c)^T}$;
Sorting data according to Mahalanobis distance:$\{sx_1, sx_2, \cdots, sx_{m_c}\}$ satisfy $d_1 \le d_2 \le \cdots \le d_{m_c}$;
$D_c' = \{sx_1, sx_2, \cdots, sx_{m_c}\}$, $\delta_c = 0$
for $i = 1$ to m_c **do**
    if $d_i > d_{[0.75m_c]} + whis\,(d_{[0.75m_c]} - d_{[0.25m_c]})$ **then**
        $D_c' \leftarrow D_c'\backslash\{sx_i\}$, $\delta_c \leftarrow \delta_c + 1$
$num_c = \xi m_c - \delta_c$, $D_c'' = D_c'\backslash\{sx_1, sx_2, \cdots, sx_{num_c}\}$.
for $k = 1$ to m_c'' **do**
    for $j = 1$ to m_c'' **do**
        compute $cos\theta_{kj}$ according to Eq 13;
        if $cos\theta_{kj} = 1$ **then**
            if $d_k > d_j$ **then**
                $D_c'' \leftarrow D_c''\backslash\left\{sx_j\right\}$
            else
                $D_c'' \leftarrow D_c''\backslash\{sx_k\}$
Output D_c''.

3.5. Complexity analysis

The time consumed by this algorithm is mainly in the following two stages.

(1) The PDGA sample reduction algorithm is used for each subset of different categories, and the time spent is mainly concentrated on calculating the Mahalanobis distance and cosine distance of the subset. Taking subset D_c as an example, we need to decompose the covariance matrix Σ_c with time complexity of $O(n^3)$, and the time complexity of calculating the Mahalanobis distance and remove outliers and non-SVs is $O(m_c)$. When further extracting redundant data, the cosine distances between samples need to be calculated with the time complexity of $O\left(m_c'\right)^2$.

(2) Training SVM with reduced dataset, the time complexity is $O\left(\sum_{c=1}^{C} m_c''\right)^2$ (Time complexity of SVM implementation based on LIBSVM).

The total time complexity of the algorithm is $\max_c \left\{O\left(n^3 + m_c + (m_c')^2\right)\right\} + O\left(\sum_{c=1}^{C} m_c''\right)^2$. For large-scale datasets, since $n \ll m$, $m_c'' \ll m_c' \ll m_c \ll m$, the execution time of this algorithm is far less than the time $O(m^2)$ needed to directly train SVM with all samples.

4. Experiments

In order to verify the rationality and performance of the algorithm, we have carried out experiments on a artificial dataset and six real datasets, and compared with several other competitive algorithms. The abbreviations of the mainstream algorithms used for comparison are shown in Table 1.

Table 1. Abbreviations of the competitive comparison algorithms.

Abbreviation	Describe	Reference
LIBSVM	Directly use all the data of the original training set to train SVM	[23]
LSSVM	Least squares support vector machines	[52]
SR-DSA	The sample reduction by data structure analysis algorithm	[49]
SE	Shell extraction	[50]
PSCC	Pre-Selection sample based on class centroid	[53]

The contents to be explored in our experiments are (i) the ability of reduction algorithms to maintain classification accuracy; (ii) The total time of algorithms execution (including processing datasets and training SVM with reduced datasets). The experiments are performed on a PC with Intel(R) Pentium(R) CPU G2030 at 3.0 GHz 8 GB RAM, Windows 10, 64 bit Operating System under Python3.

4.1. Datasets

The experiments were carried out on a artificial dataset, 4 low-dimensional UCI real datasets and 2 high-dimensional text datasets. The following is a brief introduction to those datasets.

Artificial dataset includes 3000 samples subject to two-dimensional extreme value distribution, which is used to simulate the data that the centroid is not in the geometric center and the distribution range of each attribute is quite different. It contains 3 categories, each category contains 1000 samples. The number and distribution of datasets used for testing remain the same as that of training sets.

The 4 low-dimensional UCI datasets are described below. (i) Dermatology is used to determine the type of Erythematous-Squamous disease, which contains 6 classes and 34 attributes. (ii) Letter is a database of character image features, which is used to identify the letter. (iii) Mushroom dataset includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family. Each species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. This latter class was combined with the poisonous one. (iv) Adult dataset is from the UCI machine learning database repository. The goal of this data is to predict whether a household has an income greater than \$50,000. The dataset contains 48,842 samples. Each datum has 14 attributes.

The 2 high-dimensional datasets are 20-Newsgroups and Reuters-21578. 20-Newsgroups contains 19,997 messages from 20 kinds of news. Reuters-21578 is a group of 1987 reuters news. For text datasets, first remove the texts with multiple labels, delete the empty documents, then remove the

category of less than 30 texts, and finally use the Word2vec method to convert each text into a 300-dimensional word vector.

In this paper, SVM was implemented based on LIBSVM. For each dataset, parameter tuning is carried out by using grid search first, to find the optimal parameters on each dataset. Before the experiment, the kernel type (-t), loss function (-c) and gamma function (-g) of LIBSVM were adjusted, where -t traverses 0, 1, 2; -c traversal $2^{-7}, 2^{-6}, \cdots, 2^{6}, 2^{7}$; -g traversal $10^{-1}, 10^{-2}, \cdots 10^{-5}$. In order to obtain the parameters corresponding to the best classification performance, 225 parameter optimization experiments were carried out on each dataset. The information and parameter setting details of each dataset are shown in Table 2.

Table 2. Dataset information and parameter setting.

Dataset	# samples	# features	# classes	-t	-c	-g
Artificial dataset	3000	2	3	2	2^6	10^{-1}
Dermatology	358	34	6	0	2^7	10^{-4}
Letter	20000	16	26	1	2^1	10^{-2}
Mushroom	8124	22	2	2	2^4	10^{-1}
Adult	48842	14	2	2	2^4	10^{-1}
20-NewsGroup	18828	300	20	2	2^6	10^{-1}
Reuters	9931	300	30	2	2^6	10^{-1}

We use F_1 score to measure the accuracy of SVM. F_1 score is the harmonic average of precision and recall, and is defined as

$$F_1 = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

In order to test the stability of each algorithm, the five-fold cross validation was adopted on real datasets of UCI and 2 high-dimensional text datasets, it means that, the training set was divided into five parts randomly, one of which was selected for testing and the remaining four subsets were used for training each time, to avoid the impact of classification performance due to random division of the training set. It is noted that our algorithm and several other algorithms can control the parameters to determine the number of samples to be retained, except for the LIBSVM and LSSVM. In order to explore the influence of parameter setting on the experimental results, the experiment studies the classification accuracy and the change of training time under different sample retention rates. The sample retention rate R is defined as the ratio of reduced sample subset size to the training set size, that is

$$R = \frac{\text{\# samples in subset}}{\text{\# samples in original training set}}.$$

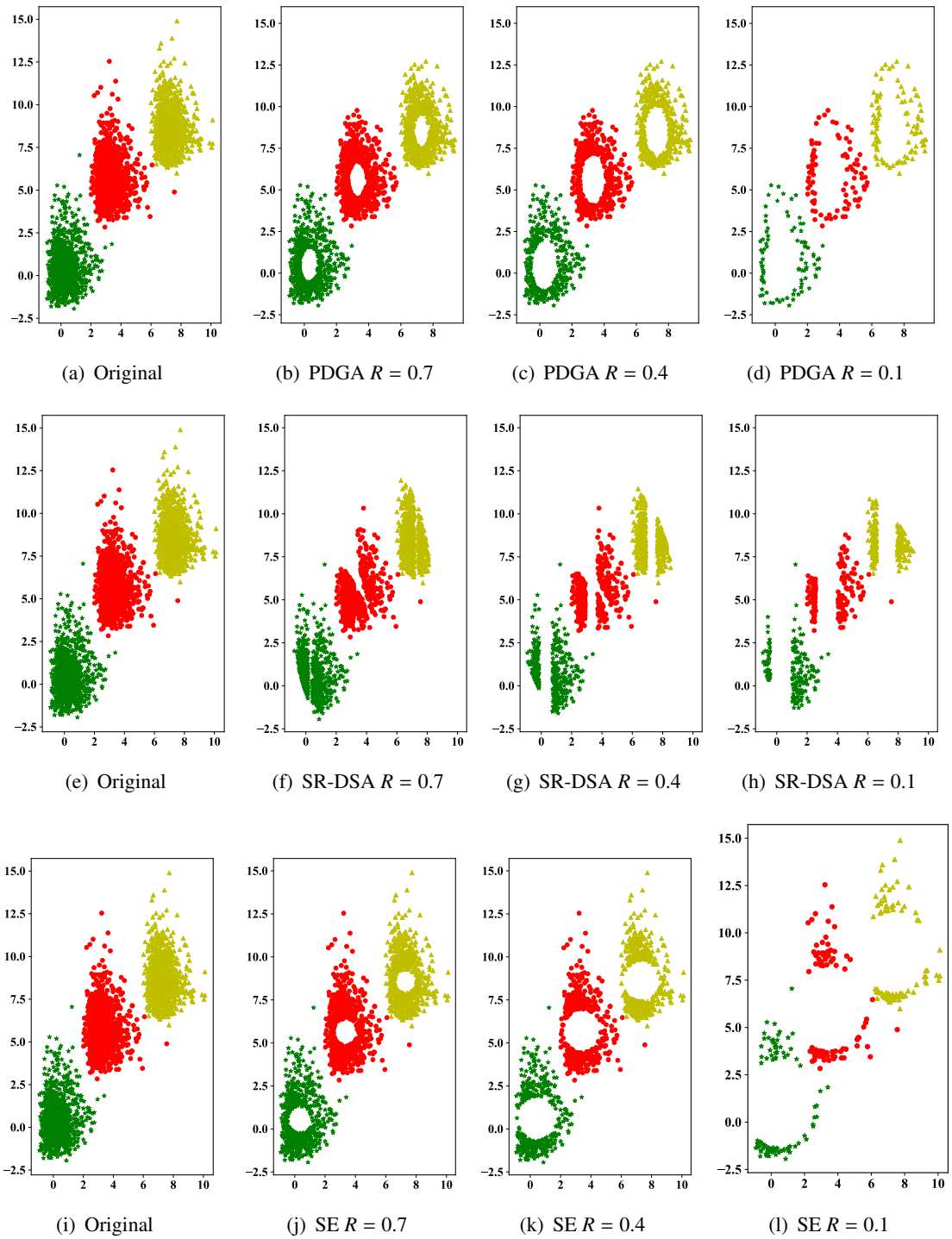
It should be noted that in order to keep the same samples with other algorithms, PDGA algorithm does not carry out cosine distance analysis when R is greater than 0.3, only when R is less than 0.3, and we set whis = 1.5.

4.2. Experiments on artificial dataset

The experiments first carried out on artificial dataset, which were used to visually demonstrate the process of reducing the size of training set by the proposed algorithm. The experiment is divided as

(i) reduce the training set to get the sample subset; (ii) train SVM with sample subset to get decision plane.

Among the total 6 algorithms, our algorithm and SR-DSA, SE are to retain the samples on the boundary of data distribution. In order to show the process of extracting redundant data intuitively, we set the sample retention rate R to 0.7, 0.4, and 0.1 respectively to observe the ability of these algorithm to retain boundary samples. The results are shown in Figure 7.



(a) Original (b) PDGA R = 0.7 (c) PDGA R = 0.4 (d) PDGA R = 0.1

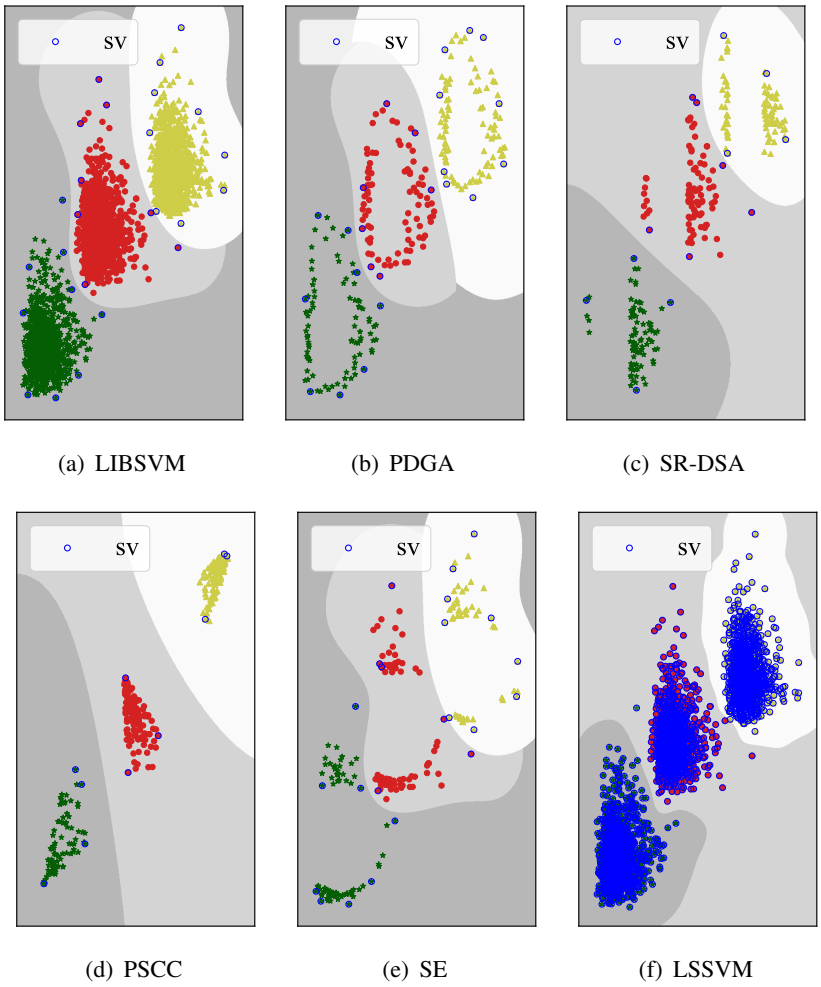
(e) Original (f) SR-DSA R = 0.7 (g) SR-DSA R = 0.4 (h) SR-DSA R = 0.1

(i) Original (j) SE R = 0.7 (k) SE R = 0.4 (l) SE R = 0.1

Figure 7. Sample subsets with different retention rates.

It can be seen from Figure 7 that our algorithm can accurately retain the samples on the boundary under different retention rates R. For SR-DSA algorithm, the internal data deletion is not complete. When the sample retention rate is very small, SE algorithm has the risk of mistakenly deleting SVs.

In order to explore the accuracy of training SVM with sample subsets obtained by different algorithms, we conducted another group of experiments on artificial dataset. In this group of experiments, our algorithm controls R at about 0.3 by adjusting parameter ξ, and further reduces the size of training set by cosine angle distance analysis. Finally, the sample subset only accounts for 0.09 of the original dataset. Other algorithms also try to keep the same dataset size as our algorithm by controlling the parameters. Finally, we use the obtained sample subset to train SVM, and draw the classification hyperplane and SVs. The results are shown in Figure 8.



(a) LIBSVM (b) PDGA (c) SR-DSA

(d) PSCC (e) SE (f) LSSVM

Figure 8. Performance comparison of several accelerate algorithms on 2-dimensional artificial datasets.

Figure 8(a) is the decision plane trained by all training samples. It can be seen from Figure 8(b) that the PDGA algorithm well preserves the boundary points of each class. Figure 8(c) is the result of SR-DSA algorithm, it does not perform well for the samples whose centroid is not in the geometric center, and the classification plane is quite different from the real plane. Figure 8(d) is the result of PSCC algorithm. It is worth noting that this algorithm is designed based on the cosine distance from

each class of samples to the centroid of different classes of samples, and almost loses most of SVs in original training set. The processing result of SE algorithm is shown in Figure 8(e). The algorithm is based on Euclidean distance to delete redundant samples, which is not effective for datasets with large differences in the distribution range of each dimension. It removes some SVs mistakenly on the boundary. We have also drawn the classification plane of LSSVM, as shown in Figure 8(f). Due to the lack of sparsity of LSSVM, every sample will become its SV.

We counted the number of mis-classified samples of each algorithm on the testing set, and the experimental results are shown in Table 3. It is noted that the classification accuracy of PDGA algorithm is higher than LIBSVM when R is only 0.09, because noise data is removed. While the classification accuracy of PSCC algorithm is seriously reduced.

Table 3. Performance comparison of several algorithms on artificial datasets.

Algorithms	# reserved samples	# mis-classified
PDGA	264	**3**
LIBSVM	3000	4
LSSVM	3000	4
SR-DSA	252	32
PSCC	282	50
SE	255	4

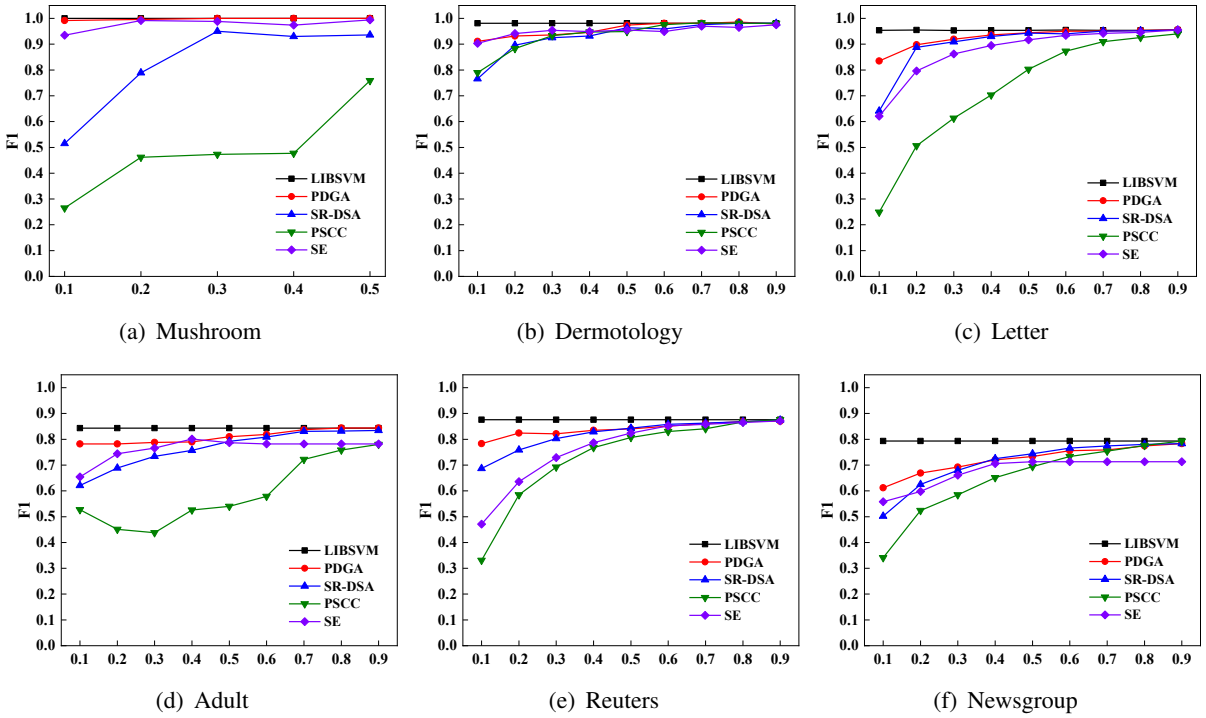
4.3. Experiments on real datasets

On real datasets, we will study the ability of the algorithm to maintain classification accuracy and the total time of each algorithm running. In order to facilitate comparison, we adjust the parameters of several algorithms to observe the classification accuracy and running time of several algorithms at the same level of sample retention R. Since the SR-DSA algorithm based on clustering can reduce the amount of data by at least half in Mushroom dataset, thus R of all algorithms traverse $0.5, 0.4, \cdots, 0.1$. On the other datasets, let R traverse $0.9, 0.8, \cdots, 0.1$. Since LSSVM is an algorithm that can avoid solving QP problems, it cannot control how many samples are retained, so we do not compare it with sample reduction algorithms here.

Firstly, the ability of each algorithm to maintain the classification accuracy in the process of reducing samples is compared. Experimental results on several real datasets are shown in Figure 9. The horizontal axis represents the sample retention rate R, and the vertical axis represents the change of F_1 score on the testing set. It should be noted that the LIBSVM shown in the figure is always the result of training with all data.

It is easy to see from Figure 9(a) that the PDGA algorithm has outstanding performance on Mushroom dataset. When R drops to 0.1, the classification accuracy of PDGA algorithm is consistent with that of the LIBSVM trained with all the samples, while the accuracy of SE algorithm decreases by 0.1, the SR-DSA by 0.5, and the PSCC algorithm by about 0.75.

Figure 9(b) shows that when R is higher than 0.5, the classification accuracy of several algorithms on Dermatology dataset is almost unaffected. However, when $R < 0.5$, the classification accuracy of several algorithms has a downward trend, but PDGA algorithm and SE algorithm can still maintain the classification accuracy above 0.9 when $R = 0.1$. SR-DSA drops to about 0.8 and PSCC to about 0.75.



(a) Mushroom (b) Dermotology (c) Letter

(d) Adult (e) Reuters (f) Newsgroup

Figure 9. Comparison of F_1 score of several SVM sample reduction algorithms on real datasets.

Figure 9(c) shows the experimental results on Letter dataset. When $R \geq 0.5$, the classification performance of PDGA, SR-DSA and SE algorithms are very close to that of LIBSVM. When $R \geq 0.2$, the F_1 score of PDGA and SR-DSA algorithms are not significantly different, and the F_1 score is maintained at about 0.9. When $R = 0.1$, the F_1 score of several algorithms all showed a certain decrease, but PDGA still ranked first, with its F_1 score above 0.83, while the other algorithms were reduced to below 0.7.

On Adult dataset (see Figure 9(d)), the change trend of PDGA was relatively slow. When $R \geq 0.5$, the F_1 score of PDGA and SR-DSA was not affected much, while when R decreased to 0.4, the F_1 score of PDGA algorithm decreased slightly. When R continues to decrease, the F_1 score of PDGA does not change much. However, the F_1 score of other algorithms continue to decline, finally droped below 0.7.

Figure 9(e) and 9(f) are the results of all algorithms on 2 high-dimensional text datasets. On Reuters dataset, when $R \geq 0.5$, the F_1 score of PDGA and SR-DSA are almost unaffected. On 20-Newsgroups dataset, F_1 score of PDGA and SR-DSA are almost unaffected when $R \geq 0.6$. When $R < 0.5$, the F_1 score of several algorithms decrease rapidly, but PDGA algorithm decrease the least. On Reuters dataset, the F_1 score of PDGA only decreases by about 0.1 when $R = 0.1$, but other algorithms drop at least 0.2. On 20-Newsgroups dataset, the PDGA algorithm decreases by about 0.2 when $R = 0.1$, while the other algorithms decrease by at least 0.25.

From the experimental results on several datasets, PDGA algorithm maintain ability of classification accuracy is better than that of several other algorithms. In most datasets, PDGA algorithm can reduce the sample size by about 50%, while maintain almost no impact on the classification accuracy. Even as R dropped to 0.1, PDGA algorithm can still obtain a better classification accuracy than several other algorithms. However, PSCC algorithm performs poorly on

these datasets.

We also compared the running time of several algorithms, including the time spent on the sample reduction process and on training SVM. The experimental results are shown in Figure 10.

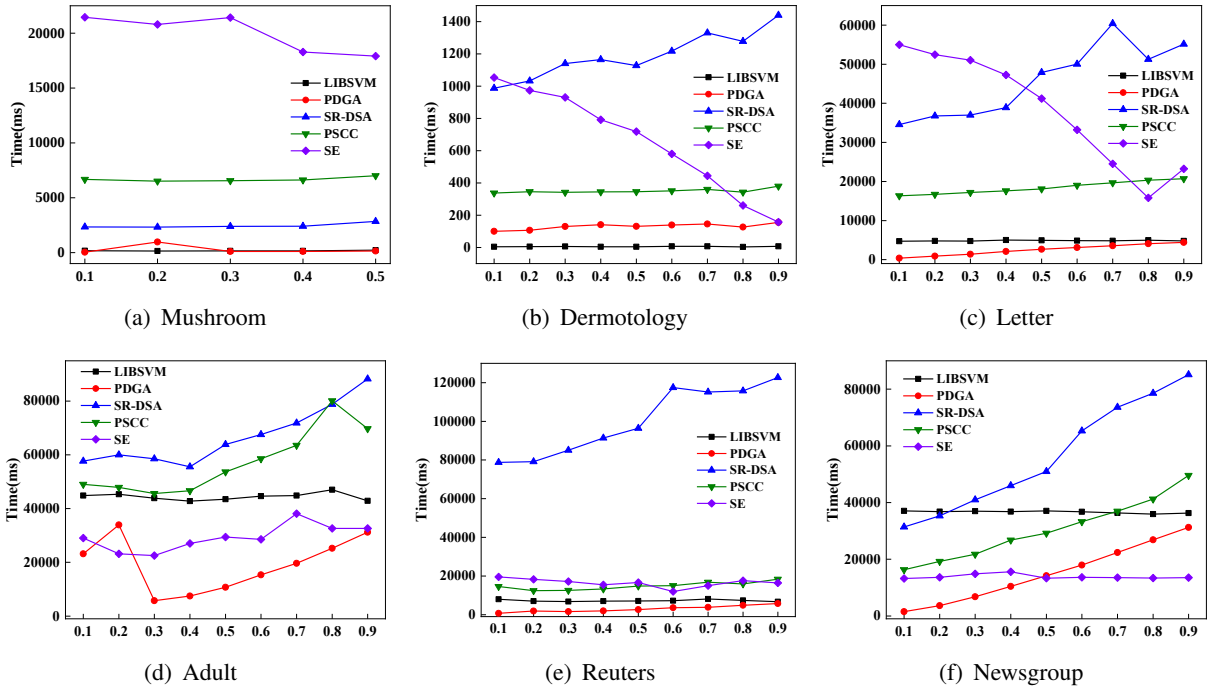


Figure 10. The total running time of several algorithms.

As can be seen from Figure 10(b), for Dermatology dataset with the minimum sample size (only about 286 training samples), it is faster to train SVM directly with all the data. Figure 10(a) shows that the running time of PDGA and LIBSVM is almost the same. However, with the increase of sample size and sample dimension, the advantage of sample reduction algorithm becomes prominent. On several datasets with a sample size of over 8000, the total time consumed by PDGA is less than that of LIBSVM. In particular, almost all sample reduction algorithms consume less time than LIBSVM in the high-dimensional and large-scale 20-Newsgroups dataset as the R decreases. This also reflects the advantage of SVM with sample reduction on large-scale datasets.

In order to illustrate whether LSSVM avoiding solving QP problem is a fast algorithm, we also calculate LSSVM classification accuracy and calculation time under the same parameter setting. For PDGA, we count the maximum sample size that PDGA can reduce when the decrease degree of F_1 score is less than 0.02, and calculate the proportion of PDGA running time to LIBSVM running time. The results are shown in Table 4.

It can be seen from Table 4 that although LSSVM can avoid solving the QP problem, the training time is much longer than LIBSVM due to the lack of sparsity, so it is not feasible on large-scale datasets. For PDGA, on Dermatology dataset, PDGA takes more time than LIBSVM. On Letter dataset, PDGA can remove 50% of the training samples, which takes only 33.54% of LIBSVM time, and the classification accuracy is only reduced by 0.019. On Mushroom dataset, PDGA algorithm can reduce 90% of the samples, while the classification accuracy decreased by only 0.008, and the running time of the algorithm only accounts for 20.69% of LIBSVM. On Adult dataset, the running time of PDGA only accounts for 45.59% of LIBSVM, while the F_1 score only decreases by 0.012, which almost has no impact on the classification accuracy. The experimental results on 2 high-dimensional

text datasets also show that PDGA algorithm can save nearly 40%–60% of LIBSVM time without significantly reducing classification accuracy.

Table 4. The time saving of PDGA algorithm without affecting the classification accuracy.

Datasets	LIBSVM		LSSVM		PDGA			
	F_1	Time(ms)	F_1	Time(ms)	F_1	Time(ms)	R	Rate of running time
Dermatology	0.986	16	0.945	179	0.986	32	0.4	–
Letter	0.962	6884	0.931	5428835	0.943	2310	0.5	33.54%
Mushroom	1	174	1	212950	0.992	36	0.1	20.69%
Adult	0.843	76462	0.847	733497	0.831	34859	0.7	45.59%
Reuters	0.876	6219	0.88	7810798	0.859	2299	0.6	35.84%
20-Newsgroups	0.793	31754	0.8	4480719	0.78	18398	0.8	57.94%

5. Conclusions and future work

This paper proposed a parallel data geometry analysis method (PDGA) for SVM sample reduction to extract the data geometry structure of SVM training set. It can minimize the number of samples without reducing the classification accuracy significantly, thus improving the training speed of SVM. The experimental results on artificial dataset shows that PDGA can retain the potential SVs well on all kinds of boundaries in training set, even for datasets with irregular geometric distribution. The experimental results on 6 real datasets show that PDGA is always the best compared with the same kind of algorithms in maintaining classification accuracy, and the time consumed is also very small. In the best case, more than 90% samples can be reduced without affecting the classification accuracy.

In the future work, we will try to extend the algorithm on high-dimensional and large-scale datasets, to simultaneously reduce redundant samples and invalid features, thus achieve effective expansion.

Acknowledgments

The authors would like to express the sincere appreciation to the editor and reviewers for their helpful comments in improving the presentation and quality of the paper.

Conflict of interest

The authors declare no conflict of interest in this paper.

References

1. J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua, A.Lopez, A comprehensive survey on support vector machine classification: Applications, challenges and trends, *Neurocomputing*, **408** (2020), 189–215.
2. Y. Wang, Z. Wang, Q. H. Hu, Y. C. Zhou, H. L. Su, Hierarchical semantic risk minimization for large-scale classification, *IEEE T. Cybernetics*, 2021, DOI: 10.1109/TCYB.2021.3059631.
3. S. H. Alizadeh, A. Hediehloo, N. S. Harzevili, Multi independent latent component extension of naive bayes classifier, *Knowl. Based Syst.*, **213** (2021), 106646.

4. L. X. Jiang, C. Q. Li, S. S. Wang, L. G. Zhang, Deep feature weighting for naive bayes and its application to text classification, *Eng. Appl. Artif. Intel.*, **52** (2016), 26–39.
5. R. J. Prokop, A. P. Reeves, A survey of moment-based techniques for unoccluded object representation and recognition, *CVGIP*, **54** (1992), 438–460.
6. A. Trabelsi, Z. Elouedi, E. Lefevre, Decision tree classifiers for evidential attribute values and class labels, *Fuzzy Set. Syst.*, **366** (2019), 46–62.
7. F. C. Pampel, *Logistic regression: A primer*, Sage publications, 2020.
8. P. Skryjomski, B. Krawczyk, A. Cano, Speeding up k-Nearest Neighbors classifier for large-scale multi-label learning on GPUs, *Neurocomputing*, **354** (2019), 10–19.
9. V. Vapnik, R. Izmailov, Reinforced SVM method and memorization mechanisms, *Pattern Recogn.*, **119** (2021), 108018.
10. V. N. Vapnik, *Statistical learning theory*, New York: Wiley, 1998.
11. C. J. C. Burges, A tutorial on support vector machines for pattern recognition, *Data Min. Knowl. Disc.*, **2** (1998), 121–167.
12. N. Cristianini, J. Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*, Cambridge: Cambridge University Press, 2000.
13. T. K. Bhowmik, P. Ghanty, A. Roy, S. K. Parui, SVM-based hierarchical architectures for handwritten bangla character recognition, *Int. J. Doc. Anal. Recog.*, **12** (2009), 97–108.
14. X. P. Liang, L. Zhu, D. S. Huang, Multi-task ranking SVM for image cosegmentation, *Neurocomputing*, **247** (2017), 126–136.
15. Y. S. Chen, Z. H. Lin, X. Zhao, G. Wang, Y. F. Gu, Deep learning-based classification of hyperspectral data, *IEEE J.-STARS*, **7** (2014), 2094–2107.
16. P. Liu, K.-K. R. Choo, L. Z. Wang, F. Huang, SVM or deep learning? A comparative study on remote sensing image classification, *Soft Comput.*, **21** (2017), 7053–7065.
17. J. Nalepa, M. Kawulok, Adaptive memetic algorithm enhanced with data geometry analysis to select training data for SVMs, *Neurocomputing*, **185** (2016), 113–132.
18. J. F. Qiu, Q. H. Wu, G. R. Ding, Y. H. Xu, S. Feng, A survey of machine learning for big data processing, *EURASIP J. Adv. Sig. Pr.*, **2016** (2016), 67.
19. T. Joachims, *Making large-scale SVM learning practical*, Technical Reports, 1998.
20. Y. F. Ma, X. Liang, G. Sheng, J. T. Kwok, M. L. Wang, G. S. Li, Noniterative sparse LS-SVM based on globally representative point selection, *IEEE T. Neur. Net. Lear.*, **32** (2021), 788–798.
21. J. C. Platt, Sequential minimal optimization: A fast algorithm for training support vector machines, 1998. Available form: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr-98-14.pdf.
22. G. Galvan, M. Lapucci, C. J. Lin, M. Sciandrone, A two-level decomposition framework exploiting first and second order information for SVM training problems, *J. Mach. Learn. Res.*, **22** (2021), 1–38.
23. C. C. Chang, C. J. Lin, LIBSVM: A library for support vector machines, *ACM T. Intel. Syst. Tec.*, **2** (2011), 27.

24. H. P. Graf, E. Cosatto, L. Bottou, I. Durdanovic, V. Vapnik, Parallel support vector machines: The cascade SVM, In: *Advances in Neural Information Processing Systems*, **17** (2004), 521–528.
25. B. L. Lu, K. A. Wang, Y. M. Wen, Comparison of parallel and cascade methods for training support vector machines on large-scale problems, In: *Proceedings of the Third International Conference on Machine Learning and Cybernetics*, **5** (2004), 3056–3061.
26. B. Scholkopf, A. J. Smola, *Learning with Kernels: Support vector machines, regularization, optimization, and beyond*, Cambridge, USA: MIT Press, 2001.
27. F. Cheng, J. B. Chen, J. F. Qiu, L. Zhang, A subregion division based multi-objective evolutionary algorithm for SVM training set selection, *Neurocomputing*, **394** (2020), 70–83
28. J. Nalepa, M. Kawulok, Selecting training sets for support vector machines: A review, *Artif. Intell. Rev.*, **52** (2019), 857–900.
29. L. Guo, S. Boukir, Fast data selection for SVM training using ensemble margin, *Pattern Recogn. Lett.*, **51** (2015), 112–119.
30. Y. Q. Lin, F. J. Lv, S. H. Zhu, M. Yang, T. Cour, K. Yu, et al., Large-scale image classification: fast feature extraction and SVM training, *CVPR*, **2011** (2011), 1689–1696.
31. A. Lyhyaoui, M. Martinez, I. Mora, M. Vazquez, J. L. Sancho, A. R. Figueiras-Vidal, Sample selection via clustering to construct support vector-like classifiers, *IEEE T. Neural Networ.*, **10** (1999), 1474–1481.
32. G. W. Gates, The reduced nearest neighbor rule, *IEEE T. Inform. Theory*, **18** (1972), 431–433.
33. M. Kawulok, J. Nalepa, Support vector machines training data selection using a genetic algorithm, In: *Structural, dyntactic, and dtatistical pattern recognition*, Springer, Berlin, Heidelberg, 2012.
34. D. R. Musicant, A. Feinberg, Active set support vector regression, *IEEE T. Neural Networ.*, **15** (2004), 268–275.
35. F. Alamdar, S. Ghane, A. Amiri, On-line twin independent support vector machines, *Neurocomputing*, **186** (2016), 8–21.
36. D. R. Wilson, T. R. Martinez, Reduction techniques for instance-based learning algorithms, *Mach. Learn.*, **38** (2000), 257–286.
37. M. Ryu, K. Lee, Selection of support vector candidates using relative support distance for sustainability in large-scale support vector machines, *Appl. Sci.*, **10** (2020), 6979.
38. J. Balcázar, Y. Dai, O. Watanabe, A random sampling technique for training support vector machines, In: *Algorithmic learning theory*, 2001, 119–134.
39. F. Zhu, J. Yang, N. Ye, C. Gao, G. B. Li, T. M. Yin, Neighbors' distribution property and sample reduction for support vector machines, *Appl. Soft. Comput.*, **16** (2014), 201–209.
40. X. O. Li, J. Cervantes, W. Yu, Fast classification for large data sets via random selection clustering and support vector machines, *Intell. Data Anal.*, **16** (2012), 897–914.
41. S. Abe, T. Inoue, Fast training of support vector machines by extracting boundary data, In: *International Conference on Artificial Neural Networks*, Springer, Berlin, Heidelberg, 2001, 308–313.
42. P. Hart, The condensed nearest neighbor rule (corresp.), *IEEE T. Inform. Theory*, **14** (1968), 515–516.

43. H. Shin, S. Cho, Neighborhood property–based pattern selection for support vector machines, *Neural Comput.*, **19** (2007), 816–855.
44. J. T. Xia, M. Y. He, Y. Y. Wang, Y. Feng, A fast training algorithm for support vector machine via boundary sample selection, In: *International Conference on Neural Networks and Signal Processing*, 2003, 20–22.
45. R. Pighetti, D. Pallez, F. Precioso, Improving SVM training sample selection using multi-objective evolutionary algorithm and LSH, In: *2015 IEEE Symposium Series on Computational Intelligence*, 2015, 1383–1390.
46. J. Kremer, K. S. Pedersen, C. Igel, Active learning with support vector machines, In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, **4** (2014), 313–326.
47. R. Wang, S. Kwong, Sample selection based on maximum entropy for support vector machines, In: *2010 International Conference on Machine Learning and Cybernetics*, 2010, 1390–1395.
48. W. J. Wang, Z. B. Xu, A heuristic training for support vector regression, *Neurocomputing*, **61** (2004), 259–275.
49. D. F. Wang, L. Shi, Selecting valuable training samples for SVMs via data structure analysis, *Neurocomputing*, **71** (2008), 2772–2781.
50. C. Liu, W. Y. Wang, M. Wang, F. M. Lv, M. Konan, An efficient instance selection algorithm to reconstruct training set for support vector machine, *Knowl. Based Syst.*, **116** (2017), 58–73.
51. C. Leys, O. Klein, Y. Dominicy, C. Ley, Detecting multivariate outliers: Use a robust variant of the mahalanobis distance, *J. Exp. Soc. Psychol.*, **74** (2018), 150–156.
52. J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, J. Vandewalle, Least squares support vector machines, *World Scientific*, 2002.
53. L. Yu, W. D. Yi, D. K. He, y. Lin, Fast reduction for large-scale training data set, *J. Southwest Jiaotong Univ.*, **42** (2007), 460–468.

AIMS Press

© 2021 the Author(s), licensee AIMS Press. This is an open access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0)