Ping Y, Tian YJ, Zhou YJ *et al.* Convex decomposition based cluster labeling method for support vector clustering. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 27(2): 428–442 Mar. 2012. DOI 10.1007/s11390-012-1232-1

Convex Decomposition Based Cluster Labeling Method for Support Vector Clustering

Yuan Ping[1,3] (平　源), *Member*, *CCF*, *ACM*, Ying-Jie Tian[2] (田英杰), *Member*, *CCF*, *ACM*
Ya-Jian Zhou[1] (周亚建), *Member*, *CCF*, *ACM*, and Yi-Xian Yang[1] (杨义先)

[1] *Information Security Center, Beijing University of Posts and Telecommunications, Beijing 100876, China*

[2] *Graduate University of Chinese Academy of Sciences, Beijing 100190, China*

[3] *Department of Computer Science and Technology, Xuchang University, Xuchang 461000, China*

E-mail: pingyuan@bupt.edu.cn; tianyingjie1213@163.com; {yajian, yxyang}@bupt.edu.cn

Received July 15, 2011; revised November 21, 2011.

Abstract Support vector clustering (SVC) is an important boundary-based clustering algorithm in multiple applications for its capability of handling arbitrary cluster shapes. However, SVC's popularity is degraded by its highly intensive time complexity and poor label performance. To overcome such problems, we present a novel efficient and robust convex decomposition based cluster labeling (CDCL) method based on the topological property of dataset. The CDCL decomposes the implicit cluster into convex hulls and each one is comprised by a subset of support vectors (SVs). According to a robust algorithm applied in the nearest neighboring convex hulls, the adjacency matrix of convex hulls is built up for finding the connected components; and the remaining data points would be assigned the label of the nearest convex hull appropriately. The approach's validation is guaranteed by geometric proofs. Time complexity analysis and comparative experiments suggest that CDCL improves both the efficiency and clustering quality significantly.

Keywords support vector clustering, convex decomposition, convex hull, geometric

1 Introduction

Clustering focuses on forming natural groupings of data points that maximize intra-cluster similarity and minimize inter-cluster similarity. It has been used for decades in image processing, pattern recognition, and now in instance-based learning, etc. Among the previous studies[1], the support vector clustering (SVC)[2-4], inspired by the support vector machines (SVMs)[5], and its variants[6-15] are recently emerged algorithms to characterize the support of a high-dimensional distribution. Actually, the SVC is a boundary-based clustering algorithm for its ability to generate cluster boundaries of arbitrary shape.

Assuming that there are N points $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ in a dataset $\mathcal{X}$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ in data space with $i \in [1, N]$. The procedure of clustering these data points by the SVC consists in general of two main phases: SVM training to estimate a support function and cluster labeling to assign each data point to its corresponding cluster. With a classic Gaussian kernel, the appropriate support vectors (SVs) are collected by solving a dual problem[16] which is related to the kernel width q and the penalty term C. Since the SVs locate on the surface of the hypersphere (with radius R), we can check the distances from m ($m \ll N$) segmers sampled on the line segment connecting any two points to the center α of the hypersphere to verify if they belong to one cluster.

Apparently, besides solving the dual problem, the labeling step with time complexity of $O(N^2m)$ takes most of the computation time for the entire SVC process[17-18]. Thus, it is crucial to accelerate the labeling speed, especially for large-scale problems. To achieve this objective, some insightful strategies have been done to replace the complete graph (CG) strategy[19], such as the support vector graph (SVG)[2], proximity graph of delaunay (DD)[6], minimum spanning tree (MST), k-nearest neighbor (kNN)[20-21], divide- and conquer-based methods[22-23],

Regular Paper

This work was supported by the National Natural Science Foundation of China under Grant No. 60972077 and partially under Grant No. 70921061, the National Science and Technology Major Program under Grant No. 2010ZX03003-003-01, the Natural Science Foundation of Beijing under Grant No. 9092009, and the Fundamental Research Funds for the Central Universities under Grant No. 2011RC0212.

cell growth based method[24], maximal margin clustering (MMC)[25-26], cone cluster labeling (CCL)[10,27], and equilibrium based approaches[7-9,17-18,28-29]. However, these approaches either reach lower cost with higher error or improve clustering results with the price of large increase of cost. Neither of them consider improving the accuracy as well as reducing the time complexity by decreasing both the number of points N and the sample rate m in the labeling step. If we get both N' ($\ll N$) and m' ($\ll m$), then a significantly faster cluster labeling method with time complexity $O({N'}^2 m') \ll O(N^2 m)$ could be achieved.

Consider the previous problems, aiming at achieving improvements on both efficiency and accuracy, a convex decomposition based cluster labeling (CDCL) algorithm is proposed in this paper. In the design and implementation of the CDCL algorithm, our paper's main contributions lie in two aspects: 1) In order to reduce the account of sampled point pairs, we provide a framework to partition the whole set of SVs into a number of non-overlapping subsets for constructing convex hulls. The vertices of each convex hull correspond to a subset of SVs. Since the SVs are the boundary points, the absolute majority of inner points are enclosed by these convex hulls separately. Therefore the judgement for the connected components can be transferred to check the connectivity of any two nearest neighboring convex hulls (NNCHs) by a few points. 2) To avoid checking the weak connected components or irregular shaped components incorrectly, we also have studied the crucial factors dominating the connectivity of two NNCHs, and give a novel definition of quasi-support vectors (QSVs). More importantly, regarding for the function of QSVs, an effective sampling sequence is found to reduce the sample rate significantly (approximately less than 2 in average) without sacrificing the clustering quality. Time complexity analysis and comparative experiments with the state-of-the-art methods suggest that the CDCL improves both the efficiency and clustering quality significantly.

The remainder of this paper is arranged as follows. We briefly review the most work on their principle, merit and limitations that related with SVC in Section 2. Section 3 formally introduces the framework of the CDCL as well as its geometric model. Section 4 discusses the time complexity of the proposed labeling algorithm in comparison of the state-of-the-art methods. Compared with the traditional ones, Section 5 details the evaluations with respect to accuracy and efficiency. Finally, the last section draws conclusions for this study with the future work.

2 Related Work

As a time-consuming work, the cluster labeling has attracted most of attentions, which first computes the adjacency matrix, and then labels the whole data points in terms of the adjacency matrix[2,19]. Originally, Ben-Hur *et al.*[2] proposed the CG to make adjacency matrix between all the point pairs, which has a highly intensive complexity ($O(N^2 m)$). To reduce the complexity of CG, proximity graph (PG) methods, i.e., DD, MST and kNN, are proposed by [6]. They constructed an appropriate PG to model a dataset, in which vertices are data points and edges connect point pairs to model their proximity and adjacency. Though the PG methods can significantly reduce the cluster labeling time (usually $O(N)$ or $O(N \log N)$), they fail frequently in labeling the clusters correctly and their time complexity grows exponentially as the dimension increases[17].

To overcome the difficulties of the PG based approaches, Lee *et al.*[7-8] presented a robust labeling method namely reduced complete graph (R-CG) on a topological property of a trained kernel radius function. Although this algorithm has very low time complexity of $O(lN \log N)$ (l is the times of iteration), only stable equilibrium points (SEPs) employed to represent dataset and to find the connected components usually lead to relatively high error on irregular shaped datasets[14]. In fact, since all the points are taken into account, the R-CG also has pricey computation for l is usually greater than 20. To further improve the accuracy, [9] thus developed equilibrium based support vector clustering (E-SVC) which introduces a transition point between two neighboring SEPs to check the connectivity of the corresponding basin cells. Later, a weighted graph and attractors[29] were constructed to make the algorithm more robust. Unfortunately, too much time consumed by seeking the transition points or constructing attractors even causes them run rather slower than R-CG.

By adopting a multi-sphere structure, [24, 30] enriched the labeling algorithm by cell growth which is able to find the cluster prototype and obtain the natural grade of membership in partition. Practically, similar to [12], too much time is consumed by both the iterative constructing centroids and surfaces and searching the nearest neighbors when the cells are growing up. Meanwhile, maximal margin based framework[26] has emerged for clustering. However, it is problematic for time-consuming since the complexity is related to the dimensionality of data. Different in approach, a recent insightful paper of [17] developed a fast support vector clustering (FSVC) algorithm which constructs an amount of small balls using the whole dataset and looks for stable equilibrium vectors (SEVs) as prototypes from the centers of balls separately. Then finding the connected components could be done by sampling the line segment connecting the SEV pairs corresponding to small balls. However, based on the assumption of circular distribution of clusters, both constructing and intermediate merging balls, similar to k-means, spectral clustering, etc.[31-32], suffer from instabilities, either because they are cast as non-convex optimization problems, or because they rely on hard threshold of distances.

Other achieved papers[10,27,33] presented the CCL method which constructs cone-shaped neighborhoods for SVs, and then observes the intersecting situation among neighborhoods to generate the adjacency matrix of SVs: two SVs belong to the same cluster if their cones are intersected. In geometric, all of the SVs have identical distance from itself to the center of hypersphere, such that verifying if the intersected cones exist will be imposed by q seriously. Therefore, a q list is required by the CCL. Thus the disadvantage of the CCL lies in expensive cost, which is used in constructing cones, observing intersecting situations, searching parameters. Furthermore, the heavily overlapping and slowly shrinks support vector cones for producing small number of clusters of the CCL may not fit for dataset of multi-classes and high-dimensional. Actually, it is the regardlessness of difference between outliers and noise points that leads to the requirement of q list. Thus Ping *et al.*[14] recommended a noise elimination algorithm which could distinguish the noise points from the outliers without affecting the clusters' profiles.

3 Convex Decomposition Based Cluster Labeling

3.1 Preliminary

This preliminary follows closely the derivation of [7-9, 18]. Since an appropriate width q of the kernel function is set to obtain the minimal hypersphere approximate covering, the solution can be considered as a gradient dynamical system (2) associated with the trained kernel function $R^2(\boldsymbol{x})$.

$$R^2(\boldsymbol{x}) = \| \varPhi(\boldsymbol{x}) - \alpha\|^2, \tag{1}$$

$$\frac{\partial x}{\partial t} = -\nabla R^2(\boldsymbol{x}). \tag{2}$$

In (1), $\varPhi(\cdot)$ is the nonlinear mapping function[2]. There exists a unique solution (or trajectory) $x(\cdot) : \mathbb{R} \rightarrow \mathbb{R}^n$ for each condition $\boldsymbol{x}(0) = \boldsymbol{x}_0$ is guaranteed since it is twice differentiable and the norm of $\nabla R^2(\boldsymbol{x})$ is bounded. Here the $\bar{\boldsymbol{x}}$ satisfying the equation $\nabla R^2(\bar{\boldsymbol{x}}) = 0$ is called equilibrum vector and is called an SEV if all the eigenvalues of its corresponding Jacobian matrix, $J_R(\bar{\boldsymbol{x}}) \equiv \nabla^2 R^2(\bar{\boldsymbol{x}})$, are positive.

At the point of convergence, the SEVs are helpful for constructing the convex hulls in this paper. We furthermore use the SVs to construct convex hulls and find the connected convex hulls for facilitating the cluster labeling. Moreover, in Appendix we strictly prove Theorems 1∼2, 4 and Lemmas 1∼3 of the following subsections.

3.2 Convexity of Clusters in Feature Space

As the prior knowledge of the proposed CDCL algorithm, the convexity of clusters is investigated in this subsection. In order to facilitate the presentation of the proposed strategy, we take no account of both noise points (eliminated by the algorithm in [14]) and outliers (eliminated by the penalty term C).

Definition 1 (Hyper Convex Polyhedron). *In feature space, the hyper convex polyhedron is a convex polyhedron whose cross sections always satisfy the property of convexity*[34].

Theorem 1. *Based on the solution of the minimum enclosing ball* (*MEB*) *problem, a support hyper convex polyhedron* (*SHCP, denoted by H*) *can be defined and constructed by all the SVs which lie on the hypersphere acting as vertices.*

Lemma 1. *Any support vector $\boldsymbol{v}_i$ ($\boldsymbol{v}_i \in \mathcal{V}$), in feature space, could not be represented by the convex combination of the others in $\mathcal{V} \backslash \boldsymbol{v}_i$.*

Theorem 2. *Employed as vertices, in feature space, the SVs whose account is greater than* 3 *can construct a hyper convex polyhedron.*

Since each cluster is enclosed by a contour across a number of SVs, in feature, the SVs surrounding a cluster can certainly construct a hyper convex polyhedron. Apparently, the aforementioned convexity will be preserved.

3.3 Convex Decomposition for Support Hyper Convex Polyhedron

3.3.1 Analysis of Separability

Lemma 2 (Decomposition). *An SHCP can be decomposed into multiple hyper convex polyhedrons, and for each one, the vertices are a subset of SVs. Furthermore, there will be no intersection existing in any two vertex sets.*

Lemma 3. *The SHCP could be decomposed into a number of non-overlapping hyper convex polyhedrons whose vertices are SVs. If the distance between any two vertices $\varPhi^{-1}(\boldsymbol{v}_i), \varPhi^{-1}(\boldsymbol{v}_j)$ ($i \neq j$) is lower than $\mathcal{D} = 2\sqrt{-\frac{\ln(\sqrt{1-R^2})}{q}}$, they should be assigned with the same label. $\varPhi^{-1}(\cdot)$ maps the data point from feature space back to data space.*

3.3.2 Decomposing SHCP into Convex Hulls

Definition 2 (Convex Hull). $\forall i \in [1, N_{V_i}]$, $V_i \subseteq R^d$, *the convex hull of V_i is defined as*

$$CH(V_i) = \left\{ \boldsymbol{x} | \boldsymbol{x} = \sum_{1 \leqslant \boldsymbol{k} \leqslant |N_i|} \lambda_k \boldsymbol{x}_k, \right.$$
$$\left. \sum_{1 \leqslant k \leqslant |N_i|} \lambda_k = 1, \boldsymbol{x}_k \in V_i, \lambda_k \geqslant 0, \lambda_k \in \mathbb{R} \right\}.$$

Theorem 3. *In feature space, the inner points of support hyper convex polyhedron (IPSHCP) denoted by $\mathcal{I}$, is defined by the data points that locate in SHCP and could be represented by the convex combination of $\mathcal{V}$. Let $\mathcal{S}$ denote the clustered data without outliers, $\mathcal{I}$ is the subset of $\mathcal{S} \backslash \mathcal{V}$. While the SHCP is decomposed into a number of sub-SHCPs, with the notation of H_i ($i \in [1, N_{\mathcal{V}}]$), $\mathcal{I}$ can also be partitioned into $\mathcal{I}_i$ which could be represented by V_i. Moreover, all the data points in $\mathcal{I}_i$ locate in the convex hull constructed by vertices of V_i while they are mapped back to data space. These convex hulls, denoted by $CH(\mathcal{V}) = \{CH(V_1) \cup CH(V_2) \cup \cdots \cup CH(V_{N_{\mathcal{V}}})\}$, are built up by vertex sets and inner points mapped back from $\mathcal{V}$ and $\mathcal{I}$ by means of $\Phi^{-1}(\cdot)$ respectively.*

Proof. This is omitted for that it is obvious and can be easily inferred from the definition of SVs and Lemmas 2∼3. For more intuitive understanding of the IPSHCP, we scatter these data by means of green points in Fig.1. □

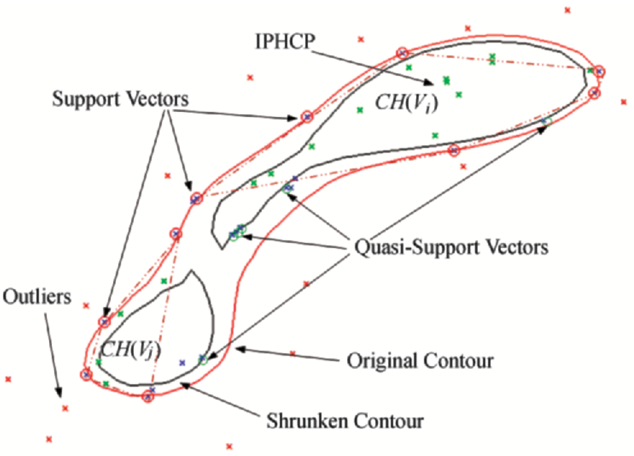

Fig.1. Figure for Theorem 3. A partial figure of *ring*[7] clustered by CG[2] algorithm with $q = 2$, $C = 0.1$. The regions encircled with dash-dot line, denoted by $CH(V_i)$ and $CH(V_j)$, are two convex hulls constructed by the current SVs circled by red circle. Without SVs shared by each other, the two components which enclose $CH(V_i)$ and $CH(V_j)$ respectively are overlapping (connected) in the original contour (red curve) while non-overlapping in the shrunken contour (black curve). On the shrunken contour, the QSVs are marked by green circle.

Theorem 4. *In feature space, there might be some points locating in the hypersphere but outside the SHCP. And, while being mapped back to the data space, these points inevitably locate outside all the partitioned convex hulls.*

According to Lemmas 2∼3, these decomposed convex hulls are non-overlapping in data space. Apparently, because of their distance to the center α is lower than R, as well as their particular locations, these data points locating between two nearest but non-overlapping convex hulls might play an important role in finding the connected convex hulls. Before detailing how these points perform, we prefer to give a definition first. For the sake of readability, in the following descriptions, we use $\boldsymbol{v}_i$ ($\boldsymbol{v}_i \in \mathcal{V}$) and $\boldsymbol{x}_i$ ($\boldsymbol{x}_i \in \mathcal{I}$) to represent either the SVs and IPSHCP in feature space, or vertices of convex hulls and inner points of these convex hulls in data space, respectively.

Definition 3 (Quasi-Support Vector). *For a dataset $\mathcal{X}$ with outliers $\mathcal{X}_O$ specified by appropriate kernel width q and penalty term C, the data point which satisfies Theorem 4 is in the set of $\mathcal{X}_{\rm QSV}$ ($= \mathcal{X} \backslash (\mathcal{S} \cup \mathcal{X}_O)$) and is defined as quasi-support vector (QSV).*

Although the distances from the QSVs to the center α are lower than R, the QSVs could become SVs while q is tuned appropriately. The relationship between SV and QSV is depicted by Fig.1. We notice that the current SVs are these points lying on the original contour and are marked with red circles. However, if the contour are tighten by means of an increased q, the QSVs marked with green circles, which locate outside of $CH(V_i)$ and $CH(V_j)$ in the red contour are being new SVs on the shrunken contour. In that case, the current SVs will be outliers. From Theorem 4, it is the sparse and discrete distribution of data points that causes the non-overlapping convex hulls seldom to enclose all of the data points either in feature space or in data space. An overlapping tolerance way of partitioning the vertices for convex hulls can inhibit the QSVs, however, more redundant outliers might be introduced to interfere the cluster labeling. Therefore, in this study, we prefer to partition the SHCP into a number of non-overlapping convex hulls and explore a fresh way to employ the QSVs for facilitating the cluster labeling.

3.3.3 Connectivity Analysis Among Convex Hulls

Consider whether the QSVs is locating between two nearest neighboring convex hulls or not, four crucial scenarios illustrated by Fig.2 might help to find the connected components.

- A number of QSVs exist between two NNCHs (see Fig.2(a)). The red contour shows that the two components enclosing $CH(V_i)$ and $CH(V_j)$ respectively are connected. It is the QSVs that enhance the connectivity; we therefore call this relationship of the two components are "*strong connection*".

- The QSVs exist between two nearest but disconnected neighboring convex hulls (see Fig.2(b)). Yet, there is still a tendency of the two components to be connected, i.e., relaxing the contour by a finely decreased q. Therefore, this relationship is called "*conditional disconnection*".

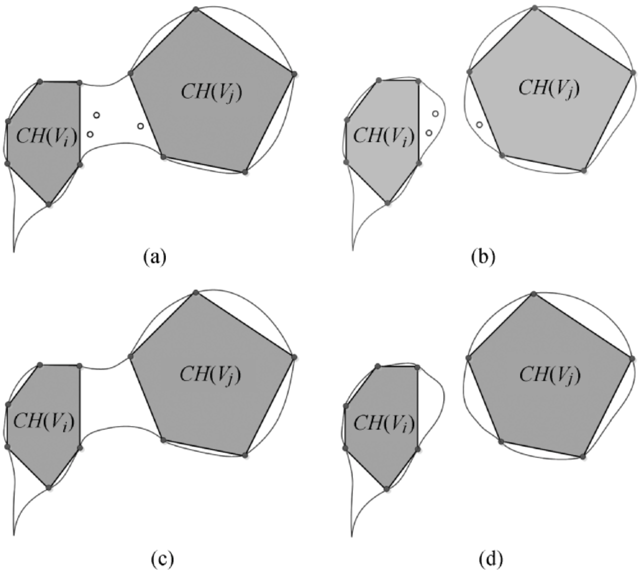

Fig.2. Four crucial scenarios with or without QSVs locating between two convex hulls, i.e., $CH(V_i)$ and $CH(V_j)$. The inner points are omitted in the grey region of the convex hulls. (a) QSVs between connected components. (b) QSVs between disconnected components. (c) Connected components without QSVs. (d) Disconnected components without QSVs.

- No QSVs exist between two NNCHs (see Fig.2(c)). However, an appropriate q could make the two components be connected. Actually, a little change for q can let them dispatch from each other. So, this could be recognized as "*weak connection*".

- No QSVs exist between two NNCHs and the corresponding two components are obvious disconnected (see Fig.2(d)). In this case, the relationship is "*disconnection*".

In Fig.2, $CH(V_i)$ and $CH(V_j)$ are two convex hulls whose vertices are the SVs highlighted by red dots and which are enclosed by red contour with different q. $CH(V_i)$ and $CH(V_j)$ are connected in Figs. 2(a)~2(b) while disconnected in Figs. 2(c)~2(d). The QSVs locating between $CH(V_i)$ and $CH(V_j)$ are marked by hollow circle.

According to the aforementioned analysis, we notice that the QSVs expand the contour outside the convex hulls, thus the four sub-figures can be obtained by different q. Practically, the QSVs can increase the connection probability of two nearest neighboring convex hulls. Consider these scenarios, an instinctive suggest would be judging for the connected components by means of finding the key QSVs and analyzing their capabilities of expanding contour of the corresponding convex hulls. The expanded contours which are overlapping with or without QSVs should be connected, otherwise disconnected. However, it is hard to check out the QSVs between two NNCHs. Fortunately, it is noteworthy that only the QSVs which locate between two NNCHs can enhance the connection probability of two corresponding components. Since the overlapping region is critical for both clustering and classificaiton[35], an alternative strategy is proposed by the authors for finding the connected components, i.e., to sample the line segments crossing the dense region of QSVs. This difficult work thus can be transferred to checking the connectivity of two NNCHs without obtaining the exact QSVs.

To achieve this goal, suppose that the dimensionality of any data point $\boldsymbol{x}_i$ is d, some essential definitions are promoted here.

Definition 4. *The distance between any two data points* $\boldsymbol{x}_i, \boldsymbol{x}_j (\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}; i, j \in [1, N])$ *is defined by*

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j)^{\mathrm{T}}(\boldsymbol{x}_i - \boldsymbol{x}_j)}. \tag{3}$$

Definition 5. (Quasi-Distance of Convex Hulls). *The quasi-distance between convex hulls* $CH(V_i)$ *and* $CH(V_j)$*, denoted by* d_q*, is defined by the distance between their nearest vertices. It is calculated as*

$$\begin{aligned} & d_q(CH(V_i), CH(V_j)) = \min d(\boldsymbol{v}_{ik}, \boldsymbol{v}_{jl}) \\ \text{s.t.} \quad & i, j \in [1, N_V],\ k \in [1, N_i],\ l \in [1, N_j], \\ & \boldsymbol{v}_{ik} \subseteq V_i, \boldsymbol{v}_{jl} \subseteq V_j \end{aligned}$$

where V_i, V_j *are the vertex sets of* $CH(V_i)$ *and* $CH(V_j)$ *whose account are* N_i *and* N_j *respectively.* N_V *is the number of non-overlapping convex hulls.*

Theorem 5. *With the dynamic system of* (2)*, the SHCP can be partitioned into a number of hyper convex polyhedrons,* $\{H_1, H_2, \ldots, H_{N_V}\}$*, which satisfy* (7)*. While they are mapped back to data space, the corresponding convex hulls,* $\{CH(V_1), CH(V_2), \ldots, CH(V_{N_V})\}$*, which are non-overlapping can be obtained.*

Proof. Proved by [7-9, 29], each SV reaches a unique local minimum position with respect to the dynamic system (2). Along with Lemma 2 and Theorem 3, in data space, these convex hulls are non-overlapping. □

3.4 Assigning Labels to Convex Hulls

3.4.1 Group the SVs by Index of Convex Hulls

Since the SVs of a convex hull will converge to one SEV with respect to system (2), we use the index of convex hulls to group the SVs according to function ConstructConvexHullsbySVs($\mathcal{V}$) which is explained by Algorithm 1. In lines 5~16, all the SVs (N_V in total) are employed to locate SEVs, and then they will be grouped by the index of SEVs to construct different convex hulls. Following Theorem 5, Algorithm 1 will actually return a unique partition of the SHCP.

```
Algorithm 1. ConstructConvexHullsbySVs($\mathcal{V}$)
Input: the collection of support vectors $\mathcal{V}$
Output: a set of convex hulls (denoted by $S_{CH}$) with
        their SEVs, vertices and indexes
1  $N_{SV} \leftarrow |\mathcal{V}|, j \leftarrow 1, k \leftarrow 1$
2  $S_{CH}.sev[] = \varnothing$ //the array of SEVs for the decom-
   posed convex hulls
3  $S_{CH}.vertex[] = \varnothing$ //the set of vertices for decomposed
   convex hulls
4  $S_{CH}.lbl[] = 0$ //the $S_{CH}.lbl[j]$ corresponds to the label
   (index) of set $S_{CH}.vertex[j]$
5  for $i = 1$ to $N_{SV}$
6     set $\boldsymbol{x}_0 \leftarrow \boldsymbol{v}_i \in \mathcal{V}$
7     Numerically integrate (2) forward with an initial
      point $\boldsymbol{x}_0$ to locate an SEV $\boldsymbol{x}_i^*$
8     if $\boldsymbol{x}_i^* \notin S_{CH}.sev$
9     then $S_{CH}.sev[j] \leftarrow \boldsymbol{x}_i^*$
10       $S_{CH}.vertex[j] \leftarrow S_{CH}.vertex[j] \cup \{\boldsymbol{v}_i\}$
11       $S_{CH}.lbl[j] \leftarrow j$
12       $j \leftarrow j+1$
13    else find $\boldsymbol{x}_i^* == S_{CH}.sev[k]$
14        $S_{CH}.vertex[k] \leftarrow S_{CH}.vertex[k] \cup \{\boldsymbol{v}_i\}$ //group
   $\boldsymbol{v}_i$ with the existed index
15    end
16 end
```

3.4.2 Merging and Labeling Convex Hulls

Since only the SEVs employed in finding the connected components frequently failed in labeling clusters correctly[14,17], we explore a new way which employs a small number of SVs with much fewer samples to achieve improvements on both efficiency and accuracy.

Consider a division of the SHCP, to avoid redundant and meaningless sample checks, finding the connected components should be taken in the NNCHs not in all the pairwise combinations. Similar to [36], the intuition of the proposed method can be considered as one kind of hierarchical clustering from bottom to top. From bottom to top, we recursively check the connectivity of two NNCHs to get the final result. Furthermore, by the transferred way, only the nearest vertices from the convex hull pairs are chosen for sampling. Therefore, the computational requirement is significantly reduced as the size of sampled points are greatly decreased.

In order to analyze the connectivity of two neighboring convex hulls, taking $CH(V_i)$ and $CH(V_j)$ for example, the concept of the proposed method is depicted by Fig.3 while the corresponding pseudocode of function MergeLabelConvexHulls(S_{CH}) is explained by Algorithm 2. In lines 5~7, firstly, we find two nearest neighboring vertices $\boldsymbol{v}_{ik}(\boldsymbol{v}_{ik} \in V_i)$ and $\boldsymbol{v}_{jl}(\boldsymbol{v}_{jl} \in V_j)$. Then, for $\boldsymbol{v}_{ik}$, we retrieve the second nearest neighboring vertex $\boldsymbol{v}_{jl'}$ from V_j in $CH(V_j)$. The same operation is done synchronously to get two neighboring vertices $\boldsymbol{v}_{ik}$ and $\boldsymbol{v}_{ik'}$ for $\boldsymbol{v}_{jl}$. As shown in Fig.3, sampling the line segments connecting either $\boldsymbol{v}_{ik}$ and $\boldsymbol{v}_{jl}$ or $\boldsymbol{v}_{ik'}$ and $\boldsymbol{v}_{jl'}$ gets incorrect results; yet too many repetitive computations consumed by sampling the other SV pairs.

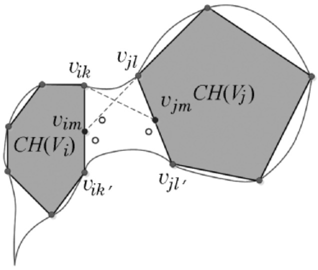

Fig.3. Sampling way towards checking the connectivity of two nearest neighboring convex hulls $CH(V_i), CH(V_j)$. $\boldsymbol{v}_{im}$ locates on the center of the line segment connecting $\boldsymbol{v}_{ik}$ and $\boldsymbol{v}_{ik'}$, while $\boldsymbol{v}_{jm}$ is the center point of line $\boldsymbol{v}_{jl}\boldsymbol{v}_{jl'}$. The quasi-distance between $CH(V_i)$ and $CH(V_j)$ is $d_q(CH(V_i), CH(V_j)) = d(\boldsymbol{v}_{ik}, \boldsymbol{v}_{jl})$. Therefore, two dash line segments should be sampled for this judgement.

```
Algorithm 2. MergeLabelConvexHulls($S_{CH}$)
Input: the set of convex hulls returned by
       ConstructConvexHullsbySVs($\mathcal{V}$)
Output: the labels of the convex hulls
1  $N_{CH} \leftarrow |S_{CH}.sev|$ //the number of convex hulls
2  set $\boldsymbol{A}_{ij} = 0$ for $i, j = 1, 2, \ldots, N_{CH}$
   //adjacency matrix
3  for $i = 1$ to $N_{CH}$
4     for $j \leftarrow j+1$ to $N_{CH}$
5        $\boldsymbol{v}_{ik}, \boldsymbol{v}_{jl} = \arg\min d_q(S_{CH}.vertex[i],$
         $S_{CH}.vertex[j])$,
6        $\boldsymbol{v}_{ik'} = \arg\min d(S_{CH}.vertex[i] \backslash \boldsymbol{v}_{ik}, \boldsymbol{v}_{jl})$
7        $\boldsymbol{v}_{il'} = \arg\min d(\boldsymbol{v}_{ik}, S_{CH}.vertex[j] \backslash \boldsymbol{v}_{il})$
      //$\boldsymbol{v}_{ik}, \boldsymbol{v}_{ik'} \in S_{CH}.vertex[i], \boldsymbol{v}_{jl}, \boldsymbol{v}_{jl'} \in S_{CH}.$
      $vertex[j]$
8        $\boldsymbol{A}_{ij}$ =CheckConnforConvexHulls($\boldsymbol{v}_{ik}, \boldsymbol{v}_{ik'}, \boldsymbol{v}_{jl}, \boldsymbol{v}_{jl'}$)
9     end
10 end
11 if $S_{CH}.vertex[i]$ and $S_{CH}.vertex[j]$ in the same con-
   nected components of the graph
12   induced by $\boldsymbol{A}$
13 then set $S_{CH}.lbl[i] = S_{CH}.lbl[j]$
   //assign the same cluster index
```

In line 8, to generate the adjacency matrix, we respectively sample the points on line segments of $\overline{\boldsymbol{v}_{ik}\boldsymbol{v}_{jm}}$ and $\overline{\boldsymbol{v}_{jl}\boldsymbol{v}_{im}}$ where $\boldsymbol{v}_{jm} = \frac{1}{2}(\boldsymbol{v}_{jl} + \boldsymbol{v}_{jl'})$ and $\boldsymbol{v}_{jl}$, $\boldsymbol{v}_{im} = \frac{1}{2}(\boldsymbol{v}_{ik} + \boldsymbol{v}_{ik'})$. Then it guarantees that the line segments can cross the overlapping region in which the QSVs may locate.

Due to the special form of the contour, in Algorithm 3, we present a two-way alternate sampling method CheckConnforConvexHulls$(\boldsymbol{v}_{ik}, \boldsymbol{v}_{ik'}, \boldsymbol{v}_{jl}, \boldsymbol{v}_{jl'})$ to verify if outliers exist. Once an outlier is found on the line segment, $CH(V_i)$ and $CH(V_j)$ are disconnected, otherwise they are connected. The difference between the conventional sampling strategy and the proposed strategy also includes the sample sequence implemented by the subfunction CheckLine$(\boldsymbol{x}_1, \boldsymbol{x}_2)$. It inherits the advantages of the binary search and the breadth-first traversal algorithms. Suppose that L_s is 10, the sample sequence generated by CheckLine$(\boldsymbol{x}_1, \boldsymbol{x}_2)$ can be illustrated by Fig.4. Theoretically, since the QSVs are able to expand the corresponding contour, the closer from a sample point to both sides of the convex hulls, the more impossible that the sample point is an outlier. Therefore, we recommend a nonlinear sample sequence like {5, 3, 7, 2, 4, 6, 8, 1, 9} to replace the conventional linear sequence as $\{1, 2, \ldots, 9\}$. Then in the case of Fig.2(d), the best situation is that one sample point is sufficient for giving the final decision correctly. The proposed algorithm, certainly, might consistently outperform the traditional algorithms which ignore the contributions from QSVs in saving computational time to find the connected components.

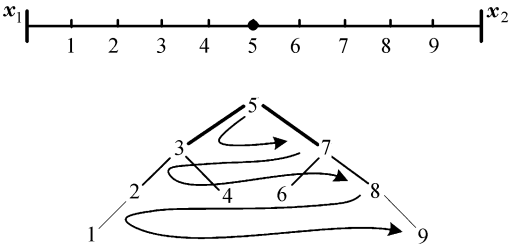

Fig.4. Sample sequence generated by the proposed function CheckLine$(\boldsymbol{x}_1, \boldsymbol{x}_2)$ while $L_s = 10$. Between points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, the exact sample sequence is {5, 3, 7, 2, 4, 6, 8, 1, 9}.

Algorithm 3. CheckConnforConvexHulls$(\boldsymbol{v}_{ik}, \boldsymbol{v}_{ik'}, \boldsymbol{v}_{jl}, \boldsymbol{v}_{jl'})$

```
Input: two point pairs extracted from the convex hulls
       $CH(V_i)$ and $CH(V_j)$ respectively
Output: adjacency matrix $\boldsymbol{A}_{ij}$
1  $a_{im} \leftarrow 0, a_{jm} \leftarrow 0$
2  $\boldsymbol{v}_{im} = \frac{1}{2}(\boldsymbol{v}_{ik} + \boldsymbol{v}_{ik'})$, $\boldsymbol{v}_{km} = \frac{1}{2}(\boldsymbol{v}_{kl} + \boldsymbol{v}_{kl'})$
3  $a_{im}$ = CheckLine$(\boldsymbol{v}_{im}, \boldsymbol{v}_{jl})$
4  if ($a_{im} == 1$) then $\boldsymbol{A}_{ij} = 1$ //connected
5  else
6    if(($a_{jm}$ = CheckLine$(\boldsymbol{v}_{km}, \boldsymbol{v}_{ik})$) == 1) then $\boldsymbol{A}_{ij} = 1$
       //connceted
7    else $\boldsymbol{A}_{ij} = 0$ //disconnected
8  end
9
10 function CheckLine$(\boldsymbol{x}_1, \boldsymbol{x}_2)$
11 $i \leftarrow 1, \lambda \leftarrow 0, N_S \leftarrow L_s$ //$L_s$ is the number of segmers,
   even number is recommended
12 $S_{\rm P} = \{1, N_S\}$ //the set of past sample points in order
13 $S_{\rm L} = |S.p|$ //the number of past sample points
14 while ($|S_{\rm L}| < N_S$)
15     for $i = 1$ to $|S_{\rm L}| - 1$
16       $j \leftarrow$ floor$(\frac{1}{2}(S_{\rm P}[i] + S_{\rm P}[i+1]))$
         //round down to the closet integer
17       if $j \in S_{\rm P}$ then continue
18       else $\lambda \leftarrow \frac{j}{N_S}$
19         if $R(\lambda \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2) > R$ then return 0,
           break //disconnected
20       $S.l \leftarrow S_{\rm L} + 1, S_{\rm P}$ = sort$(S_{\rm P} \cup \{j\})$ //in sequence
21       end
22     end
23 end
24 return 1 //connected
```

If there are M convex hulls with N_{SV} SVs, to merge convex hulls, a collection of $2 \times C_M^2 = P_M^2$ point pairs are required for sampling by the proposed method. Unfortunately, for algorithms in [10, 14, 17, 27], $C_{N_{SV}}^2$ point pairs should be taken into account. For example, if N_{SV} is 20 for 4 convex hulls, the point pairs required by the proposed algorithm is $P_4^2 = 12$, whereas the traditional algorithms need exactly $C_{20}^2 = 190$ pairs. Instinctively, the CDCL will take much less time in merging and labeling convex hulls than the traditional ones, especially in dealing with large scale but low dimensional data. In contrast with the high-dimensional data, the profile of the low-dimensional data can be stroked by much fewer SVs.

3.5 Assign the Remaining Data Points

Algorithm 4 shows the CDCL method. In line 1, for the given q value, it collects the SVs set $\mathcal{V}$ by solving the dual problem. Then in line 2, the SHCP constructed by $\mathcal{V}$ is decomposed into convex hulls S_{CH} according to

Algorithm 4. *CDCL*$(\mathcal{X}, q, C)$

```
Input: the dataset $\mathcal{X}$, Gaussian kernel width $q$ and the
       penalty term $C$
Output: clustering labels for all the data points
1  collect $\mathcal{V}$ for $q$ by solving dual problem
2  $S_{CH} \leftarrow$ ConstructConvexHullsbySV$(\mathcal{V})$
3  $\{S_{CH}, \boldsymbol{A}\} \leftarrow$ MergeLabelConvexHulls$(S_{CH})$
4  Labels $\leftarrow$ FindConnComponents$(\boldsymbol{A})$
5  for each $\boldsymbol{x} \in \mathcal{X} \backslash \mathcal{V}$
6    $inx \leftarrow$ find the nearest SV from $\boldsymbol{x}$
7    Labels$[\boldsymbol{x}] \leftarrow$ Labels$[v_{inx}]$
8  end
9  return Labels
```

the function ConstructConvexHullsbySV($\mathcal{V}$). In line 3, the adjacency matrix $\boldsymbol{A}$ is obtained by MergeLabelConvexHulls(S_{CH}). The connected convex hulls, namely cluster, can be immediately found by means of any standard algorithm, e.g., the depth first search (DFS). Therefore, the output of FindConnComponents($\boldsymbol{A}$) in line 4 is an array with size N_{CH} which contains the cluster labels. Finally, similar to [10, 17, 27], the remaining data points are separately assigned with the labels of their nearest SVs.

4 Time Complexity Analysis

To analyze the time complexity of the proposed method, let N be the number of data points in a dataset, N_{SV} be the number of SVs, l be the average number of iterations for each data point to locate its corresponding local minimum via the steepest decent process[7] and m be the sample rate for finding the connected components. Apparently, the time cost of constructing convex hulls by SVs (Algorithm 1) and merging the extracted convex hulls (Algorithm 2) are dependent on the N_{SV} and the actual number of convex hulls (denoted by N_{CH}) which is usually much lower than N_{SV}. Cluster assignment of the convex hulls from adjacency matrix by DFS algorithm and cluster matching for each takes $O(N_{CH})$ and $O(N)$ of elementary calculation respectively, which are ignorable in practice. Thus the time complexity of the CDCL is $O(lN_{SV} + 2mN_{CH})$. Compared with the state-of-the-art algorithms, the time complexity of these algorithms are listed in Table 1.

Table 1. Time Complexity of Labeling Approaches

Index	Method	Time Complexity
1	CG	$O(mN^2)$
2	DD	$O(N \log N + mf(N))$
3	kNN	$O(N \log N + mkN)$
4	MST	$O(N \log N + mN)$
5	R-CG	$O(lN + mN_{CH}^2)$
6	E-SVC	$O(lN + mN_{CH}^2 + 2lN_{CH})$
7	CCL	$O(N_{SV}^2)$
8	FSVC	$O(lN_b + \gamma N^2)$
9	CDCL	$O(lN_{SV} + 2mN_{CH})$

In Table 1, $f(N)$ is a function of N for the DD algorithm, k is the number of nearest neighbors ($k = 4$ is preferred) used by the kNN. Controlled by parameter ρ in [17], N_b is the number of small balls extracted from the dataset and γ ranges from $\frac{1}{N}$ to 1. Since the number of edges in DD is linear in size N for 2D datasets, it is even more than quadratic in size for the high-dimensional datasets, thus $f(N)$ implies that the DD would take much more time than most of the others. When we move to high-dimensional datasets, kNN and MST are attractive, since the number of edges of graphs constructed remains linear in N. However, they fail in accuracy frequently that will be summarized in the next section. It is worth mentioning that we omit the size of parameter list required by the CCL algorithm, therefore, the actual time consumed by the CCL usually exceeds the observed time complexity. Another attractive algorithm is FSVC, the cost is almost linear in number of small balls if $\gamma = 1$. Unfortunately, similar to the instability converge of k-means[32], FSVC is intrinsically unstable for partitioning data points into small balls by selecting instance randomly. Among these algorithms, since the N_{CH} is usually much smaller than N_{SV}, CDCL can be considered as linear in size of SVs, especially for a small m. As discussed in Section 1, one of our objectives is to decrease m for efficiency. Therefore, as shown in Table 1, CDCL can be expected to produce competitive results in most cases.

5 Experimental Evaluation

5.1 Datasets and Experimental Settings

To demonstrate the effectiveness and performance of the proposed method, we compare it with the state-of-the-art methods listed in Table 1 on various datasets: *ring*, *sunflowers*, *five-Gaussians*, *orange* and *twocircles* which are widely used in the literatures[15,17,37-38] and *iris*, *wisconsin*, *wine*, *zoo* and *movement_libras* which are from UCI repository[39]. All the datasets are described in Table 2. Two series of simulations are conducted with Core dual 2.66 GHz and 3 GB memory size machine.

Table 2. Description of the Benchmark Datasets

Datasets	Dataset Description		
	Dims	Size	No. Classes
Sunflowers	2	200	9
Orange	2	140	9
Twocircles	2	300	2
Five-Gaussians	2	1 000	5
Iris	4	150	3
Wisconsin	9	683	2
Wine	13	178	3
Zoo	16	101	7
Movement_libras	90	360	15

The purpose of the first series of simulations is to check whether the proposed method improves the adaptability of finding the connected components, while the shape of dataset is irregular, e.g., weak connectivity. Since the shape is generally affected by the parameters, including q and C, an effective method should adapt wide range of parameters to achieve an expected performance. To evaluate both the efficiency and accuracy, we conduct the second series of experiments on the ten datasets and use adjusted rand index (ARI, denoted by RI_{adj})[1,40] which is a widely used similarity measure between two data partitions where both true labels and predicted cluster labels are given.

5.2 Adaptability for Connectivity of Convex Hulls

Although R-CG is one of the fastest algorithms, it performs ineffectively while processing the case of weak connected components[14]. To check the adaptability of the proposed method, we use *ring* to construct a number of weak components and compare its performance with R-CG. The results are illustrated by Fig.5. Obviously, since three line segments cross outside the contour (the red bold line in Fig.5(a)), R-CG incorrectly find the connected components. On the contrary, by employing the proposed method, the corresponding lines in Fig.5(b) results in correct clusters.

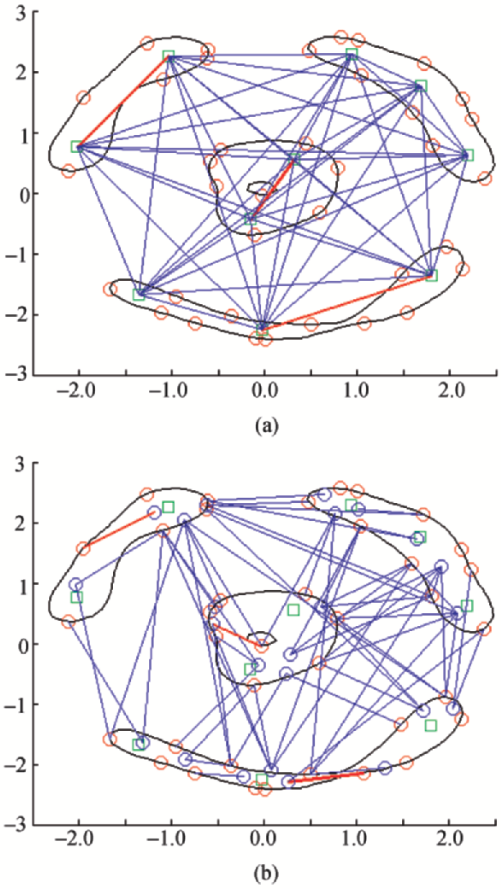

Fig.5. Comparison test on sampling strategy between R-CG and CDCL, where $q = 2$, $C = 0.1$ while the inner points and outliers are removed. The SVs are denoted by red circles while the SEVs and the generated points are marked by green rectangles and blue circles respectively. (a) Sampling strategy of R-CG. (b) Sampling strategy of CDCL.

The *orange* is used to verify the adaptability and parameter sensitivity of the proposed method. Comparison is conducted between CCL and CDCL, which is presented in Fig.6. Since the radius of the hypersphere is controlled by q, performance of CCL is almost independent of C. Furthermore, we notice that the proposed method not only outperforms CCL in their best performance, but also shows its stability and satisfaction of reaching an expected accuracy with much wider parameter combinations. In evidence, these comparisons suggest that CDCL outperforms the traditional methods in capability of generalization.

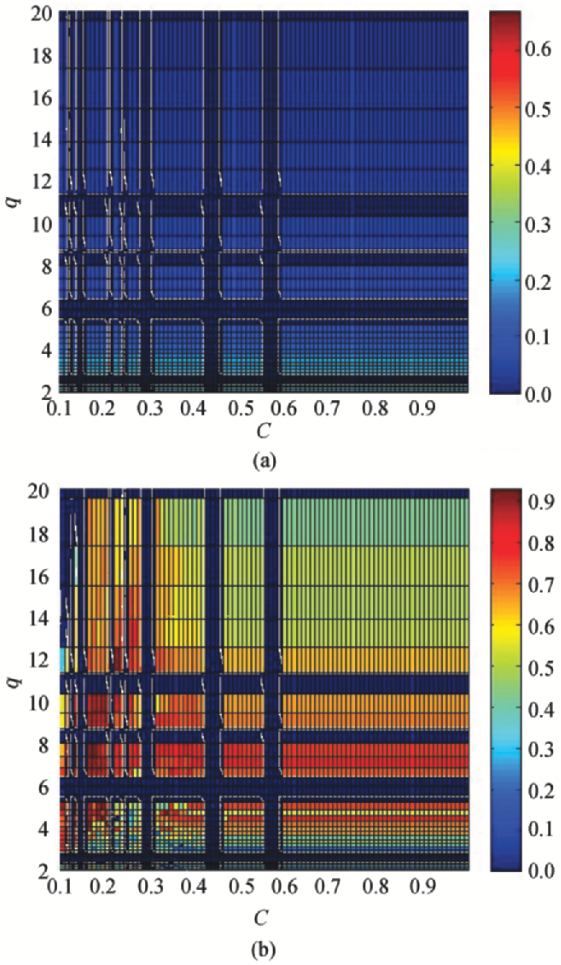

Fig.6. Comparison of ARI measures of CCL and CDCL. X-axis represents the range of C from 0.1 to 1, and Y-axis the range of q from 2 to 20. The actual values of RI_{adj} are stroked by different colors which can be found at the right color bar. (a) ARI measures of CCL. (b) ARI measures of CDCL.

5.3 Comparisons of Benchmark Datasets

To achieve full comparisons, we include cluster labeling time (consisting of convex decomposition, merging convex hulls and labeling the remaining data), the number of clusters and RI_{adj} achieved by the evaluated algorithms separately and list the results in Table 3. Rank of each item highlighted by boldface with superscript is given depending on its performance followed by corresponding rank (from 1 to 3).

For *orange*, *twocircles* and *wisconsin*, CDCL spends the least time in cluster labeling. More than this, it almost achieves the first three rank in each evaluation whereas performance of the others are unstable. In most cases, kNN, CCL and FSVC perform well. But, the cluster labeling time of these algorithms increase dramatically as the dimensionality and size of a dataset increase. Another time-consuming algorithm is E-SVC, which fails in handling the 90-dimensional dataset *movement_libras*. It is worthy to note that DD hardly clusters such high-dimensional data since it cannot construct the graph; therefore, the DD is only suitable to deal with low-dimensional data. Intuitively, we expect an effective algorithm could achieve a high accuracy with a corresponding number of clusters close to the exact number of classes. In Table 3, N_R denotes the number of clusters got by these algorithms while N_C is the exact number of classes summarized in Table 2. A good clustering algorithm should avoid splitting the same group of data into different clusters[41], thus we highlighted the number of clusters by boldface, which is closest to the real number of classes. Obviously, N_R found by the proposed method is almost the same with N_C, which outperforms the others significantly. More importantly, in terms of the ARI measure, performance of CDCL is the best for most of the datasets (namely *sunflowers*, *orange*, *twocircles*, *five-Gaussians*, *iris*, *wine* and *movement_libras*), where as CCL gives better result for *wisconsin* and both of R-CG and E-SVC get the best result for *zoo*. Nevertheless for these cases, the proposed CDCL obtains rank 2. Consider the column 4 of Table 3, analysis of Subsection 5.2 and the description of datasets (see Table 2), we can find that the proposed CDCL is superior to the others in clustering data of weak connectivity. However, FSVC frequently fails in processing datasets of high-

Table 3. Benchmark Results of Cluster Labeling Time, Number of Clusters (N_R), Number of Classes (N_C) and Accuracy

Datasets	Method	Cluster Labeling Time (s)	$N_R/(N_C)$	RI_{adj}
sunflowers	CG	3.09168	**9**/(9)	**1.00000**^{1}
	DD	**0.66696**^{3}	**9**/(9)	**1.00000**^{1}
	kNN	**0.57213**^{2}	**9**/(9)	**1.00000**^{1}
	MST	2.59106	10/(9)	**0.99412**^{2}
	R-CG	1.48508	**9**/(9)	**1.00000**^{1}
	E-SVC	106.24486	10/(9)	**0.98662**^{3}
	CCL	**0.16530**^{1}	10/(9)	0.86759
	FSVC	0.67017	**9**/(9)	0.98070
	CDCL	0.75678	**9**/(9)	**1.00000**^{1}
orange	CG	1.28655	21/(9)	0.74781
	DD	0.40709	15/(9)	0.83076
	kNN	**0.27903**^{3}	11/(9)	**0.91439**^{3}
	MST	0.77973	12/(9)	0.85503
	R-CG	1.05923	**9**/(9)	**0.92825**^{2}
	E-SVC	15.25184	**9**/(9)	0.90898
	CCL	**0.20506**^{2}	5/(9)	0.49927
	FSVC	0.77569	**9**/(9)	0.88476
	CDCL	**0.17326**^{1}	**9**/(9)	**0.92830**^{1}
twocircles	CG	9.96133	**2**/(2)	**1.0000**^{1}
	DD	11.53812	**2**/(2)	**1.0000**^{1}
	kNN	2.33760	4/(2)	0.69679
	MST	17.47976	8/(2)	0.59935
	R-CG	2.86713	4/(2)	0.67695
	E-SVC	28.40459	4/(2)	**0.73547**^{3}
	CCL	**1.04268**^{3}	**2**/(2)	**0.76193**^{2}
	FSVC	**0.73254**^{2}	19/(2)	0.14022
	CDCL	**0.66909**^{1}	**2**/(2)	**1.0000**^{1}
five-Gaussians	CG	86.38930	17/(5)	0.47118
	DD	22.56921	27/(5)	0.61487
	kNN	71.11776	23/(5)	0.67808
	MST	3279.69089	**5**/(5)	0.72182
	R-CG	**10.89263**^{3}	17/(5)	**0.86934**^{2}
	E-SVC	1079.60133	18/(5)	**0.85854**^{3}
	CCL	413.66390	19/(5)	0.00032
	FSVC	**1.04425**^{1}	17/(5)	0.71373
	CDCL	**4.81953**^{2}	11/(5)	**0.88074**^{1}

(to be continued on next page)

Table 3. Benchmark Results of Cluster Labeling Time, Number of Clusters (N_R), Number of Classes (N_C) and Accuracy (continued)

Datasets	Method	Cluster Labeling Time (s)	$N_R/(N_C)$	RI_{adj}
iris	CG	1.11913	11/(3)	0.61780
	DD	2.85248	35/(3)	0.58334
	*k*NN	**0.35029**^{1}	9/(3)	0.64143
	MST	1.04676	6/(3)	**0.79457**^{3}
	R-CG	3.05768	16/(3)	0.73737
	E-SVC	4.73092	2/(3)	0.56812
	CCL	**0.72480**^{3}	**3**/(3)	**0.88579**^{2}
	FSVC	1.05975	5/(3)	0.56196
	CDCL	**0.65785**^{2}	**3**/(3)	**0.92218**^{1}
wisconsin	CG	319.57929	8/(2)	0.77930
	DD	–	–	–
	*k*NN	31.59080	9/(2)	0.76243
	MST	111.14570	11/(2)	0.66311
	R-CG	**22.21045**^{3}	13/(2)	**0.80345**^{3}
	E-SVC	443.19526	46/(2)	0.13441
	CCL	23.69021	**2**/(2)	**0.90763**^{1}
	FSVC	**13.01807**^{2}	153/(2)	0.66871
	CDCL	**1.28192**^{1}	**2**/(2)	**0.86850**^{2}
wine	CG	2.97916	31/(3)	0.59121
	DD	–	–	–
	*k*NN	**0.46556**^{1}	14/(3)	0.56323
	MST	0.79843	15/(3)	0.49680
	R-CG	7.78199	20/(3)	0.79281
	E-SVC	36.70322	12/(3)	0.41586
	CCL	**0.63475**^{2}	**3**/(3)	**0.81902**^{2}
	FSVC	7.60204	15/(3)	**0.80419**^{3}
	CDCL	**0.79507**^{3}	**3**/(3)	**0.89613**^{1}
zoo	CG	0.60476	**8**/(7)	**0.93421**^{3}
	DD	–	–	–
	*k*NN	**0.24880**^{1}	**8**/(7)	**0.93421**^{3}
	MST	**0.36386**^{2}	**8**/(7)	**0.93421**^{3}
	R-CG	3.80894	**8**/(7)	**0.95702**^{1}
	E-SVC	19.83797	**8**/(7)	**0.95702**^{1}
	CCL	**0.51522**^{3}	12/(7)	0.83426
	FSVC	2.65616	10/(7)	0.86791
	CDCL	2.81722	**6**/(7)	**0.94691**^{2}
movement_libras	CG	**15.08310**^{3}	139/(15)	0.24218
	DD	–	–	–
	*k*NN	**7.19747**^{2}	90/(15)	**0.26661**^{2}
	MST	41.00582	92/(15)	**0.24872**^{3}
	R-CG	252.84331	138/(15)	0.23559
	E-SVC	–	–	–
	CCL	**1.02772**^{1}	230/(15)	0.08988
	FSVC	226.06705	213/(15)	0.14206
	CDCL	78.53238	**37**/(15)	**0.33195**^{1}

Note: "–" means not available.

dimensionality or irregular shape, e.g., *twocircles*, *iris* and *movement_libras*, and E-SVC performs ineffectively for high-dimensional datasets while CCL fails in labeling the datasets with many samples, for instance, *five-Gaussians*. Because, to guarantee $R \leqslant 1$, the constraints in solving the dual problem required by CCL[10,27] is much more strict than the other algorithms.

Since the objective of the sampling strategy is to reduce the sample rate m required by finding the connected components, another interest of our experiments is to verify if the implemented Algorithm 3 works as we expected. Compared with CG which employs the same strategy with the others, we illustrate the average sample rate required by CDCL and CG respectively in Fig.7. By employing the proposed strategy, the average sample rate retains lower than 2, whereas CG almost requires twice. Particularly, for *wisconsin*, the ratio of CG and CDCL reaches 11.12. So the proposed CDCL gets an evident improvement of reducing the average sample rate to the others.

As the aforementioned discussions, it is clear that CDCL which is independent of shape and dimensionality and insensitive to the parameters is quite available for improving both efficiency and accuracy. It should be pointed out that the results in Table 3 were obtained from the programs available. Different implementation of the algorithms may result in somewhat different speed. In particular, we did not attempt to optimize CDCL program for faster speed at all. Therefore, Table 3 can only be used as an indication of the relative speed of the algorithms.

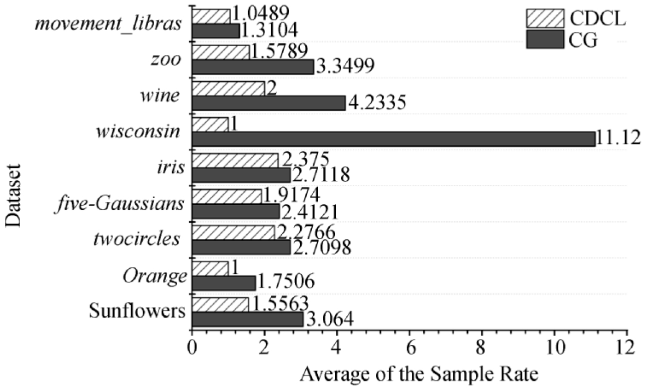

Fig.7. Comparison of the sample rate between CG and CDCL algorithms.

6 Conclusions and Future Work

In this paper, a novel efficient and robust cluster labeling algorithm, namely CDCL, based on convex decomposition is proposed to improve both efficiency and accuracy of support-based clustering. Differing from the traditional cluster labeling algorithms, we consider to improve the accuracy as well as to reduce the time complexity by decreasing both the number of points N and the sample rate m. Starting from a series of analysis of the support vector domain description, we find that the SVs can be employed to construct an SHCP, which can be further decomposed into a number of non-overlapping convex hulls. The work of finding the connected components, therefore, can be done between any two NNCHs. So the number of convex hulls, denoted by N_{CH}, which would substitute N is a relatively much smaller value. Furthermore, after analyzing the relationship of neighboring convex hulls, a newly defined concept of QSV is suggested to dominate the connectivity of two NNCHs. To guarantee an expected accuracy, therefore, we sample the line segments which not only connect two point pairs of two nearest neighboring convex hulls respectively, but also cross the QSVs with the maximal probability. Practically, this strategy is proven to overcome the weak connectivity effectively. To make a further improvement, a new sampling strategy is developed to avoid too much redundant and meaningless operations which can significantly decrease the sample rate and even requires only once. Then, the time complexity of CDCL achieves $O(lN_{SV} + 2mN_{CH})$. Since N_{CH} is much smaller than N_{SV} and m is consistently less than 2, the time complexity is practically linear in size of SVs. Experimental results show that the proposed CDCL method significantly reduces the time consumption while obtaining higher accuracy than the traditional algorithms.

Since the size of SVs could be affected by the distribution of data, e.g., even or uneven distribution, how to control the size of SVs while obtains high quality profiles of clusters might be an open issue for further improvements on both efficiency and accuracy. At the same time, the convex decomposition method reported in this paper is envisaged to provide a systematic approach to exemplar selection, accurate behavior description, event presentation, etc. All of these potential applications are worthy of further investigation.

References

[1] Xu R, Wunsch D C. Hierarchical clustering. In *Clustering*, Hoboken: John Wiley & Sons, 2008.

[2] Ben-Hur A, Horn D, Siegelmann H T, Vapnik V N. Support vector clustering. *Journal of Machine Learning Research*, 2001, 2: 125-137.

[3] Schölkopf B, Platt J C, Shawe-Taylor J C, Smola A J, Williamson R C. Estimating the support of a high-dimensional distribution. *Neural Computation*, 2001, 13(7): 1443-1472.

[4] Tax D M J, Duin R P W. Support vector domain description. *Pattern Recognition Letters*, 1999, 20(11-13): 1191-1199.

[5] Burges C J C. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 1998, 2(2): 121-167.

[6] Yang J H, Estivill-Castro V, Chalup S K. Support vector clustering through proximity graph modelling. In *Proc. the 9th International Conference on Neural Information Processing* (*ICONIP 2002*), Orchid Country Club, Singapore, Nov. 18-22, 2002, pp.898-903.

[7] Lee J, Lee D. An improved cluster labeling method for support vector clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2005, 27(3): 461-464.

[8] Lee J, Lee D. Dynamic characterization of cluster structures for robust and inductive support vector clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2006, 28(11): 1869-1874.

[9] Lee D, Lee J. Equilibrium-based support vector machine for semisupervised classification. *IEEE Transactions on Neural Networks*, 2007, 18(2): 578-583.

[10] Lee S-H, Daniels K M. Cone cluster labeling for support vector clustering. In *Proc. the 6th SIAM Conference on Data Mining*, Bethesda, Maryland, Apr. 20-22, 2006, pp.484-488.

[11] Varma C M B S, Asharaf S, Murty M N. Rough core vector clustering. In *Proc. the 2nd International Conference on Pattern Recognition and Machine Intelligence*, Kolkata, India, Dec. 18-22, 2007, pp.304-310.

[12] Hsieh T W, Taur J S, Tao C W, Kung S Y. A kernel-based core growing clustering method. *International Journal of Intelligent Systems*, 2009, 24(4): 441-458.

[13] Ling P, Zhou C G, Zhou X. Improved support vector clustering. *Eng. Applicat. Artificial Intelligence*, 2010, 23(4): 552-559.

[14] Ping Y, Zhou Y J, Yang Y X. A novel scheme for accelerating support vector clustering. *Computing and Infomatics*, 2012, 31(2): in press.

[15] Jung J H, Kim N, Lee J. Dynamic pattern denoising method using multi-basin system with kernels. *Pattern Recognition*, 2011, 44(8): 1698-1707.

[16] Vapnik V. The Nature of Statistical Learning Theory. New York: Springer-Verlag, 1995.

[17] Jung K H, Lee D, Lee J. Fast support-based clustering method for large-scale problems. *Pattern Recognition*, 2010, 43(5): 1975-1983.

[18] Lee D, Lee J. Dynamic dissimilarity measure for support-based clustering. *IEEE Transactions on Knowledge and Data Engineering*, 2010, 22(6): 900-905.

[19] Ben-Hur A, Horn D, Siegelmann H T, Vapnik V. A support vector cluster method. In *Proc. the 15th International Conference on Pattern Recognition*, Barcelona, Spain, Sep. 3-7, 2000, pp.724-727.

[20] Estivill-Castro V, Lee I. AMOEBA: Hierarchical clustering based on spatial proximity using Delaunay diagram. In *Proc. the 9th Int. Symposium on Spatial Data Handling*, Beijing, China, Aug. 10-12, 2000, pp.7a.26-7a.41.

[21] Estivill-Castro V, Lee I, Murray A T. Criteria on proximity graphs for boundary extraction and spatial clustering. In *Proc. the 5th Pacific-Asia Conference on Knowledge Discovery and Data Mining* (*PAKDD 2001*), Hong Kong, China, Apr. 16-18, 2001, pp.348-357.

[22] Ban T, Abe S. Spatially chunking support vector clustering algorithm. In *Proc. International Joint Conference on Neural Networks*, Budapest, Hungary, July 25-29, 2004, pp.413-418.

[23] Puma-Villanueva W J, Bezerra G B, Lima C A M, Zuben F J V. Improving support vector clustering with ensembles. In *Proc. International Joint Conference on Neural Networks*, Montreal, Quebec, Canada, Jul. 31-Aug. 4, 2005, pp.13-15.

[24] Chiang J H, Hao P Y. A new kernel-based fuzzy clustering approach: Support vector clustering with cell growing. *IEEE Transactions on Fuzzy Systems*, 2003, 11(4): 518-527.

[25] Wang F, Zhao B, Zhang C S. Linear time maximum margin clustering. *IEEE Transactions on Neural Networks*, 2010, 21(2): 319-332.

[26] Peng J M, Mukherjee L, Singh V, Schuurmans D, Xu L L. An efficient algorithm for maximal margin clustering. *Journal of Global Optimization*, 2011, 2: 1-15. doi:10.1007/s10898-011-9691-4.

[27] Lee S H, Daniels K. Gaussian kernel width selection and fast cluster labeling for support vector clustering. University of Massachusetts, Lowell, Technical Report No. 2005-009, 2005.

[28] Lee C H, Yang H C. Construction of supervised and unsupervised learning systems for multilingual text categorization. *Expert Systems with Applications*, 2009, 36(2): 2400-2410.

[29] Lee D, Jung K H, Lee J. Constructing sparse kernel machines using attractors. *IEEE Transactions on Neural Networks*, 2009, 20(4): 721-729.

[30] Hao P Y, Chiang J H, Tu Y K. Hierarchically SVM classification based on support vector clustering method and its application to document categorization. *Expert Systems with Applications*, 2007, 33(3): 627-635.

[31] Hocking T D, Joulin A, Bach F, Vert J P. Clusterpath: An algorithm for clustering using convex fusion penalties. In *Proc. the 28th International Conference on Machine Learning* (*ICML*), Bellevue, WA, USA, Jun. 28-Jul. 2, 2011, pp.1-8.

[32] Shamir O, Tishby N. Stability and model selection in k-means clustering. *Machine Learning*, 2010, 80(2-3): 213-243.

[33] Lee S H, Daniels K. Gaussian kernel width generator for support vector clustering. In *Proc. International Conference on Bioinformatics and Its Applications*, Fort Lauderdale, Florida, USA, Dec. 16-19, 2004, pp.151-162.

[34] Boyd S, Vandenberghe L. Convex Optimization. Cambridge, 7th edition, New York: Cambridge University Press, 2009.

[35] Li Y H, Maguire L. Selecting critical patterns based on local geometrical and statistical information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011, 33(6): 1189-1201.

[36] Kpotufe S, von Luxburg U. Pruning nearest neighbor cluster trees. In *Proc. the 28th International Conference on Machine Learning* (*ICML 2011*), Bellevue, USA, Jun. 28-Jul. 2, 2011, pp.225-232.

[37] Kim H C, Lee J. Clustering based on gaussian processes. *Neural Computation*, 2007, 19(11): 3088-3107.

[38] Camastra F, Verri A. A novel kernel method for clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2005, 27(5): 801-805.

[39] Frank A, Asuncion A. UCI machine learning repository. http://archive.ics.uci.edu/ml, 2010.

[40] Hubert L, Arabie P. Comparing partitions. *Journal of Classification*, 1985, 2(1): 193-218.

[41] Wu M, Scholkopf B. A local learning approach for clustering. In *Proc. the 20th Annual Conference on Advances Neural Information Processing Systems* (*NIPS 2007*), Vancouver, B.C., Canada, Dec. 4-7, 2006, pp.1529-1536.

Yuan Ping received his M.S. degree in mathematics from the Henan University, Henan in 2008, and his B.S. degree in electronics and information engineering from Southwest University, Chongqing in 2003. He is currently a Ph.D. candidate at Beijing University of Posts and Telecommunications, Beijing, China. His current research interests include pattern classification, machine learning, text categorization, security protocol and operating system security.

Ying-Jie Tian received his Ph.D. degree in management science from China Agricultural University. Currently he is an associate professor of Graduate University of Chinese Academy of Sciences. His research interests include data mining, machine learning, SVM theory, optimization theory and method, pattern analysis.

Ya-Jian Zhou was born in China in 1971. He received his Ph.D. degree in 2003 in communications engineering from Xidian University at Xi'an. He received his M.S. degree in 1996 and B.S. degree in 1993, both from the Department of Material Science and Engineering of Beijing University of Aeronautics and Astronautics. He is currently a lecturer in School of Computer Science and Technology of Beijing University of Posts and Telecommunications. His main research interests include mobile communications, security of wireless networks, security of databases, cryptography theory and its application, etc. He is now the project leader of two projects from the National Natural Science Foundation of China and the National High Technology Research and Development 863 Program of China, respectively. Meanwhile, he is in charge of a project from Beijing Municipal Natural Science Foundation. He has published about 20 technical papers and two books.

Yi-Xian Yang received his Ph.D. degree in signal and information processing from Beijing University of Posts and Telecommunications in 1988. He is now the director of National Engineering Laboratory for Disaster Backup and Recovery, the director of Key Laboratory of Network and Information Attack & Defence Technology of MOE, the director of Information Security Center, and the Deputy Director of State Key Laboratory of Networking and Switching Technology. Meanwhile, he is a Changjiang Scholars Distinguished Professor and the editor-in-chief of "Journal on Communications" and "The Journal of China University of Posts and Telecommunications". His current research interests include network and information security theory and applications, network-based computer application technology, coding theory and technology.

Appendix

Proof of Theorem 1. Consider the analysis in Section 1, in feature space, the collection of SVs, $\mathcal{V} = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_{N_{SV}}\}$, has a radius R to the center α. Therefore any point x sampled on the line segment connecting two SVs locates in the hypershpere with $R(x) \leqslant R$. Following the principle of MEB[4], the hypersphere is convex. If there is a finite set $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_{N_{SV}}\}$ with cardinality N_{SV} satisfying $\lambda_i \geqslant 0$ ($i \in [1, N_{SV}]$) and $\sum_{i=1}^{N_{SV}} = 1$, then, the convex combination of $\mathcal{V}$ meets

$$R\Big(\sum_{i=1}^{N_{SV}} \lambda_i \boldsymbol{v}_i\Big) \leqslant \lambda_1 R(\boldsymbol{v}_1) + \cdots + \lambda_{N_{SV}} R(\boldsymbol{v}_{N_{SV}}) = \Big(\sum_{i=1}^{N_{SV}} \lambda_i\Big) R(\boldsymbol{v}_i) = R. \tag{A1}$$

Therefore, Theorem 1 is proved. Furthermore, this hypersphere is a special case of hyper convex polyhedron if $N_{SV} \to \infty$. □

Proof of Lemma 1. If a support vector $\boldsymbol{v}_i$ ($i \in [1, N_{SV}]$) is the convex combination of vertices in $\mathcal{V}' = \mathcal{V} \backslash \boldsymbol{v}_i$, i.e., $\sum_{j=1, j\neq i}^{N_{SV}} \lambda_j \boldsymbol{v}_j$, then the value of $R(\boldsymbol{v}_i)$ is lower than R. Actually, it conflicts with $R(\boldsymbol{v}_i) = R$. □

Proof of Theorem 2. In evident, every point $\boldsymbol{x}_s$ sampled on the line segment of any two SVs satisfies $R(\boldsymbol{x}_s) < R$. That means the convexity is always existing in the convex combination of SVs. Since a sphere should be comprised by at least 3 points, Theorem 2 is proved. □

Proof. of Lemma 2. Suppose that the vertex set of the SHCP is $\mathcal{V}$, following Theorem 2, a subset $V_i (V_i \subseteq \mathcal{V})$ with N_{V_i} ($\geqslant 3$) vertices constructs a hyper convex polyhedron H_i. Thus a kind of partition for $\mathcal{V}$ corresponding to (A2) whose number is N_V can be obtained. Although there may be vertices shared by two neighboring hyper convex polyhedrons, e.g., H_i and H_j ($i \neq j$), without considering the intersecting line segment, their inner data points cannot be expressed by the other's vertex set. That means (A3) can be inferred by (A2).

$$\mathcal{V} = V_1 \cup V_2 \cup \cdots \cup V_{N_V}, \quad \forall i, j \in [1, N_V],\ i \neq j,\ V_i \cap V_j \neq \varnothing \tag{A2}$$

$$H \supseteq (H_1 \cup H_2 \cup \cdots \cup H_{N_V}), \quad \forall i, j \in [1, N_V], i \neq j, H_i \cap H_j = \varnothing \tag{A3}$$

□

Proof of Lemma 3. This lemma can be expressed following the principle of minimal hypersphere approximate covering in [10, 27]. As depicted in Fig.A1, in feature space, all the data points are mapped into the hypersphere S with center $\boldsymbol{\alpha}$. Simultaneously, for the gaussian kernel, $K(\boldsymbol{x}, \boldsymbol{x}) = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}) \rangle = ||\Phi(\boldsymbol{x})||^2 = 1$, it means all the data points locate on the surface of a unit ball B with center $\boldsymbol{O}$. Thus data actually spread on the cap-like surface intersected by S and B[13]. In Fig.A1, $\boldsymbol{v}_1$, $\boldsymbol{v}_2$ and $\boldsymbol{v}_3$ are samples taken from the set

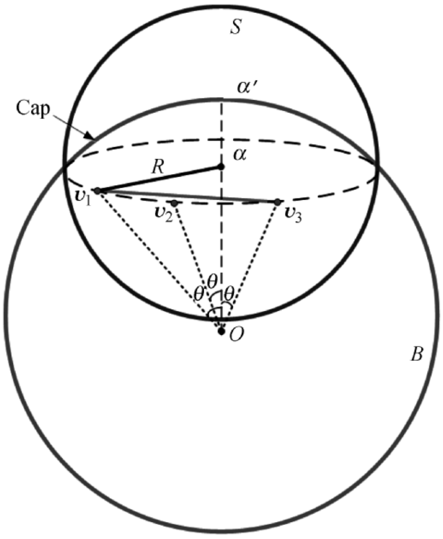

Fig.A1. Figure for Lemma 3. If the distance between $\boldsymbol{v}_1$ and $\boldsymbol{v}_3$ is lower than $\mathcal{D}$, in the hypersphere S, then they should be assigned with the same label.

$\mathcal{V}$. $\boldsymbol{\alpha}'$ is projection of $\boldsymbol{\alpha}$ onto the surface B. Apparently, there is $\angle(\boldsymbol{v}_1\boldsymbol{O}\boldsymbol{\alpha}') = \angle(\boldsymbol{v}_2\boldsymbol{O}\boldsymbol{\alpha}') = \angle(\boldsymbol{v}_3\boldsymbol{O}\boldsymbol{\alpha}')$. For any two points in $\mathcal{V}$, i.e., $\boldsymbol{v}_1$ and $\boldsymbol{v}_3$, they generate two cones CN_1, CN_2 with $\boldsymbol{v}_1, \boldsymbol{v}_2$ as the vertices, $\boldsymbol{O}\boldsymbol{v}_1, \boldsymbol{O}\boldsymbol{v}_2$ as axes, and $\angle(\boldsymbol{v}_1\boldsymbol{O}\boldsymbol{\alpha}'), \angle(\boldsymbol{v}_2\boldsymbol{O}\boldsymbol{\alpha}')$ as the basic angle respectively. If CN_1 and CN_2 are intersected, it means $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ belong to the same cluster in data space. From the Lemmas 2.2∼2.4 in [10], any data point $\boldsymbol{x}$, in data space, is in the cluster of $\boldsymbol{v}_i$ ($i \in [1, N_{SV}]$) while their distance is lower than the radius $r = \sqrt{-\frac{\ln(\sqrt{1-R^2})}{q}}$. Therefore, if two clusters are connected and could be assigned with the same label, the maximal distance between two nearest vertices from each cluster is twice of the radius r. So, for any two hyper convex polyhedrons, the distance $2 \times r$ between two vertices of them, $2 \times \sqrt{-\frac{\ln(\sqrt{1-R^2})}{q}}$, is the upper bound if they are connected. □

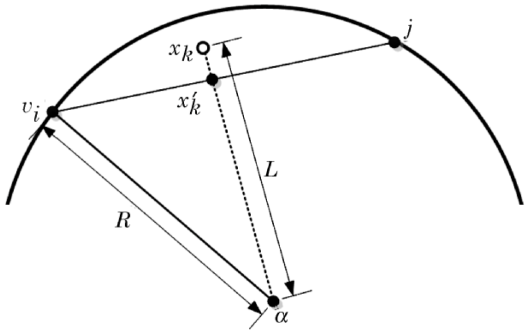

Fig.A2. Figure for Theorem 4. It is a cross section from hypersphere S. $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ are two SVs in $\mathcal{V}$, x_k is a data point with distance $L(L < R)$ to the center. In sight of the SHCP, x_k is an outlier locating outside the boundary $\overline{\boldsymbol{v}_i\boldsymbol{v}_j}$.

Proof. of Theorem 4. As depicted in Fig.A2, the cross section is a part of the SHCP. On the one hand, due to the number of SVs is finite, $\mathcal{S}$ and SHCP are not completely overlapping. Therefore, there might be some data points, e.g., $\boldsymbol{x}_k$, locates in the non-overlapping regions.

On the other hand, a data point $\boldsymbol{x}_k$ and a hyper convex hull which has a line segment $\overline{\boldsymbol{v}_i\boldsymbol{v}_j}$ which is the nearest to $\boldsymbol{x}_k$ are taken into account for this proof (see Fig.A2). Suppose that the distance from $\boldsymbol{x}_k$ to any inner point, e.g., the center α (every inner point is suitable), is greater than to the corresponding point $\boldsymbol{x}_k{}'$ sampled on $\overline{\boldsymbol{v}_i\boldsymbol{v}_j}$. Thus we get an inequality described by

$$\begin{aligned} & R^2(\Phi^{-1}(\boldsymbol{x}_k)) > R^2(\Phi^{-1}(\boldsymbol{x}_k{}')) \\ \Longrightarrow & ||\boldsymbol{x}_k - \alpha||^2 > ||\boldsymbol{x}_k{}' - \alpha||^2. \end{aligned} \quad \text{(A4)}$$

Then, using (A4) and the Gaussian kernel, the following inequality can be deduced.

$$\begin{aligned} & K(\Phi^{-1}(\boldsymbol{\alpha}), \Phi^{-1}(\boldsymbol{x}_k)) < K(\Phi^{-1}(\boldsymbol{\alpha}), \Phi^{-1}(\boldsymbol{x}_k{}')) \\ \Longrightarrow & e^{-q||\Phi^{-1}(\boldsymbol{\alpha})-\Phi^{-1}(\boldsymbol{x}_k)||^2} < e^{-q||\Phi^{-1}(\boldsymbol{\alpha})-\Phi^{-1}(\boldsymbol{x}_k{}')||^2} \\ \Longrightarrow & ||\Phi^{-1}(\boldsymbol{\alpha}) - \Phi^{-1}(\boldsymbol{x}_k)||^2 > ||\Phi^{-1}(\boldsymbol{\alpha}) - \Phi^{-1}(\boldsymbol{x}_k{}')||^2. \end{aligned} \quad \text{(A5)}$$

Obviously, these data points locate outside the corresponding convex hulls while they are mapped back to data space. □