Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 597521, 12 pages
http://dx.doi.org/10.1155/2013/597521

Research Article

Integrating the Supervised Information into Unsupervised Learning

Ping Ling,[1] Nan Jiang,[2] and Xiangsheng Rong[3]

[1] *School of Computer Science and Technology, Jiangsu Normal University, Xuzhou 221116, China*
[2] *School of Business, Jiangsu Normal University, Xuzhou 221116, China*
[3] *Training Department, Air Force Logistics of P. L. A, Xuzhou 221000, China*

Correspondence should be addressed to Ping Ling; lingicehan@aliyun.com

Received 2 August 2013; Accepted 25 September 2013

Academic Editor: Hao Shen

This paper presents an assembling unsupervised learning framework that adopts the information coming from the supervised learning process and gives the corresponding implementation algorithm. The algorithm consists of two phases: extracting and clustering data representatives (DRs) firstly to obtain labeled training data and then classifying non-DRs based on labeled DRs. The implementation algorithm is called SDSN since it employs the tuning-scaled Support vector domain description to collect DRs, uses spectrum-based method to cluster DRs, and adopts the nearest neighbor classifier to label non-DRs. The validation of the clustering procedure of the first-phase is analyzed theoretically. A new metric is defined data dependently in the second phase to allow the nearest neighbor classifier to work with the informed information. A fast training approach for DRs' extraction is provided to bring more efficiency. Experimental results on synthetic and real datasets verify that the proposed idea is of correctness and performance and SDSN exhibits higher popularity in practice over the traditional pure clustering procedure.

1. Introduction

Among diverse issues of data mining, clustering is a fundamental one owing to its ability to provide the data group information before fulfilling other further mining tasks. Clustering pursues data partition that holds maximum cohesion and minimum coupling [1]. It is an intensively researched field, and a number of branches of clustering methods appeared, such as partition-based clustering, hierarchical clustering, density-based clustering, grid-based clustering, boundary-detecting clustering, and model-based clustering [1, 2]. These existing approaches are purely unsupervised learning procedures, which usually draw clustering models from the whole unlabeled dataset with statistics or geometric mechanisms. In other words, the data are probed only once to learn cluster structures. Based on that conventional clustering way, we try to learn discriminative information from data twice; with intention to obtain more accurate clusters. In this paper, we summarize that idea as an assembling clustering idea that consists of two phases. The first phase fulfills the first exploration on the whole data by extracting data representatives (DRs). The rough group information is revealed by DRs. Then, in the second phase, the discriminative information is explored through clustering DRs. We train a classifier based on the labeled DRs, and the discriminative information learned in the second phase is carried by the classifier. If viewing the label information obtained in the second phase as the supervised knowledge, the previous idea actually conducts a procedure that integrates the supervised knowledge into the unsupervised learning process.

Our main idea is implemented by a concrete algorithm, SDSN, which involves support vector technique [3], Spectrum analysis [4–8], and the nearest neighbor classifier. The extraction of DRs of SDSN is achieved by the support vector domain description (SVDD) that is equipped with a tuned scale parameter of Gaussian Kernel. DRs clustering task is fulfilled by a spectrum-analysis-based method. Nearest neighbor classifier (NN) accomplishes label assignment under the guidance of a new metric.

The contributions of SDSN are as follows. (1) The scale parameter of Kernel function used by SVDD is self-tuned, which helps SVDD to generate the semisupport vectors that

can act as DRs. (2) In feature space spanned by Gaussian Kernel function, geometric properties of the semisupport vectors and the hyperspheres are explored to guarantee the validation of the first-phase clustering method. (3) The definition of new metric is derived from the cluster structures of DRs, so it makes NN classifier ready to address various-shaped datasets. (4) A fast training approach of SDSN is given to speed up the extraction process of DRs. The fast training approach is based on Schrödinger equation of quantum mechanism [9] to yield an effective learning list of data batches that are friendly to adjusting model incrementally.

Experiments are conducted to compare SDSN with some state-of-the-art clustering methods, say the purely unsupervised clustering methods. Empirical results show that SDSN is of the better or competitive behaviors than its peers. That verifies the validation and the performance of our two-phase clustering idea. Besides, seen in another opinion, SDSN can be thought of as a type of support vector clustering algorithm. So, experiments are conducted to compare SDSN with the popular clustering methods based on support vector technique. Empirical evidence indicates the improvement of SDSN over its counterparts in efficiency and performance.

In the following description, Section 2 illustrates the two-phase clustering idea. Sections 3 and 4 give the details of SDSN's first phase, that is, extracting DRs and clustering them. Therein the geometric properties of hyper spheres in the feature space spanned by Gaussian Kernel are proposed and proofed to guarantee the validation of the clustering method of the first phase. Section 5 describes the label assignment method of the second phase, where a cluster-structure-dependent metric is defined to facilitate the classifier's job. The fast training approach is proposed in Section 6. The experimental analysis is given in Section 7. Section 8 is a conclusion.

2. Two-Step Clustering Idea

The essential of two-step clustering idea is to learn discriminative information from data twice by integrating supervised information into the unsupervised learning process. In details, it discovers DRs, learns the label information from DRs, and trains the classifier based on labelled DRs. Then the remaining unlabelled data are classified by the classifier.

Compared with purely unsupervised clustering process that detects clusters from all unlabelled data only once, the proposed idea learns discriminative information of clusters twice. The first learning is during the process of DRs collection. Cluster distribution information is investigated through finding the DRs that are located on the boundaries of dense regions. Secondly, when DRs are clustered, grouping information is probed once again, and this information is carried by the classifier trained on the labelled DRs. That, in turn, benefits the resulted classifier to hold the strong discriminative ability to label other data.

3. Extracting DRs of the First Phase

In the first phase, DRs that hold the ability to describe the dataset sketch are collected. Among existing data reduction approaches, there are two main types. The first type specifies DRs through random sampling; the second type finds DRs based on probability distribution information. Clearly the former incurs randomness and consequently leads to the unexpected results. The latter often produces unpleasant results since the exact probability distribution of real dataset is usually unknown.

Focused on previous difficulties, we exploit SVDD to produce support vectors, which serve as DRs. The reason of choosing SVDD to search DRs is that it is of boundary-detecting mechanism, which fosters SVDD the ability to handle arbitrary-shaped clusters. Furthermore the employment of Kernel function of SVDD enhances its generalization ability. However, the support vectors produced by SVDD only describe the cluster boundaries, without providing the information of inner regions of clusters. Therefore, we propose the tuning-scaled SVDD to yield the support vectors that can describe the dataset sketch. Besides, we give a fast training method for tuning-scaled SVDD to facilitate the efficiency. Now classical SVDD is introduced in brief.

3.1. SVDD. Given the n-dimensional dataset X, $X = \{x_1, \ldots, x_N\}$. SVDD process is expressed by the following optimization problem:

$$\begin{aligned} \min_{R,\xi} \quad & R^2 + C\sum_i \xi_i, \\ \text{s.t.} \quad & \left\|\Phi(x_i) - a\right\|^2 \le R^2 + \xi_i, \quad \xi_i \ge 0. \end{aligned} \tag{1}$$

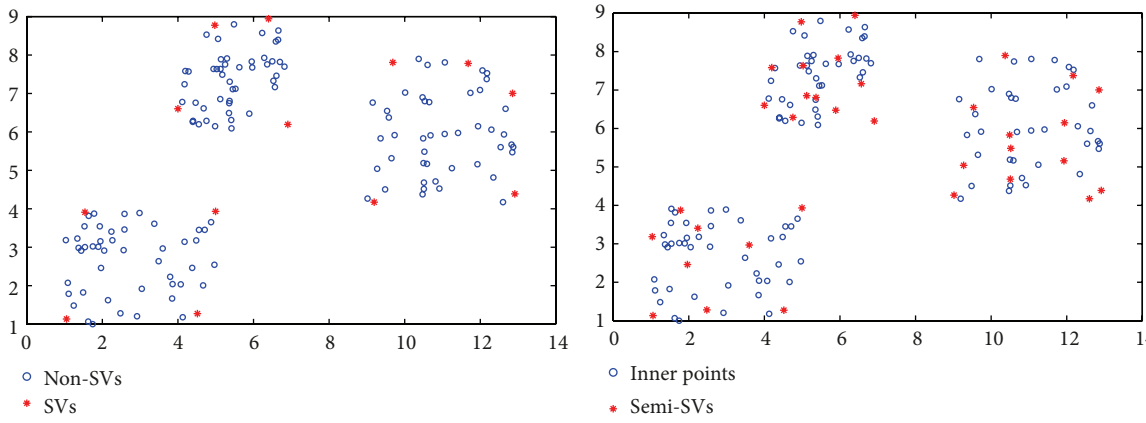
Φ is the nonlinear map from the input space to the feature space, ξ_i is the slack variable, a is the center of hypersphere, R is the radius of the hyper sphere, and C is the penalty parameter to tradeoff R and ξ_i. Transfer it to the Lagrange function, then to the Wolfe dual. Introduce Kernel function $K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$, and we have

$$\begin{aligned} \min_{\beta} \quad & \sum_{i,j} \beta_i \beta_j K(x_i, x_j) - \sum_i \beta_i K(x_i, x_i), \\ \text{s.t.} \quad & 0 \le \beta_i \le C, \quad \sum_i \beta_i = 1. \end{aligned} \tag{2}$$

Points with $\beta_i = 0$ are inner-boundary points. Points with $0 < \beta_i < C$ are SVs, and they actually formulate cluster contours. K is often set as the Gaussian Kernel function $K(x, y) = \exp(-q\|x - y\|^2)$, where q is the scale parameter.

3.2. Tuning-Scaled SVDD. The original SVs are only located on cluster boundaries. However, in this paper, for SVDD, we tune the scale parameter of Gaussian Kernel data adaptively, which enhances the difference among data points, and consequently more SVs are produced to describe both cluster contours and contours of small inner-cluster regions with high density. So, we name these SVs produced by tuning-scaled SVDD as the semisupport vectors (semi-SVs). Obviously, these semi-SVs develop a sketch of dataset, and they are qualified to work as DRs.

In details, for x, define its scale factor as $\sigma_x = \|x - x_r\|$. x_r is the rth furthest point to x. σ_x reflects the local distribution



(a) (b)

Figure 1: SVDD with $q = 1.86$ (a) and tuning-scaled SVDD (b).

density information of x's neighborhood. If $\|x - x_r\| \leq \|y - y_r\|$, it means that the neighborhood of x is denser than that of y. To measure Kernel affinity between x and y, their scale factors are combined in the way that $q_{xy} = 1/\sigma_x\sigma_y$, which will serve as the scale parameter. Then it results in the modified Gaussian Kernel

$$K(x, y) = \exp\left(-q_{xy}\|x - y\|^2\right) = \exp\left(-\frac{\|x - y\|^2}{\sigma_x \cdot \sigma_y}\right) = \exp\left(-\frac{\|x - y\|^2}{\|x - x_r\| \cdot \|y - y_r\|}\right). \quad (3)$$

Therein, r is specified as the max gap in the list of distances from x to other points. Say r is the index of the point with the max gap with x. Steps of the specification are as follows: (1) compute $\text{gap}(i) = \max_j\{[\|x_i - x_j\| - \|x_i - x_{j-1}\|]/\|x_i - x_j\|\}$, where the list $\{\|x_i - x_j\|\}$ $(j = 1 \cdots N)$ is sorted in the ascending order; (2) consider the average of $\text{gap}(i)$ as r value: $r = \text{ave}\{\text{gap}(i)\}$.

Two figures in Figure 1 show the SVs and the semi-SVs produced by SVDD and tuning-scaled SVDD. Obviously, SVs produced by classical SVDD only describe cluster boundaries. But semi-SVs produced by tuning-scaled SVDD are located on both cluster boundaries and important inner-cluster positions where sharp changes of density happen. Therefore, these semi-SVs give a good sketch description of dataset and can be regarded as qualified DRs.

4. Clustering DRs of the First Phase

4.1. Clustering Method. In the first phase, after extracting DRs, we cluster DRs with a spectrum-analysis-based method. The clustering method has the following steps. (1) Compute affinity matrix K of DRs according to Gaussian Kernel and denote the entries of K as K_{ij}: $K_{ij} = K(x_i, x_j)$. (2) Normalize K into K': $K' = \Lambda^{-1/2}K\Lambda^{-1/2}$, where $\Lambda = \text{diag}(\Lambda_i) = \text{diag}(\sum_j K_{ij})$. (3) Take top g eigenvectors as columns to form spectrum matrix. (4) Perform K-means on rows of spectrum matrix, with the cluster number being initialized as g. (5) Label x_i as the ith row's cluster membership. Therein, g is specified by the number of eigenvalues that are larger than 1 [10]. The previous clustering idea is rooted from spectrum analysis [4]. The validation of this algorithm is supported by the geometric properties of Gaussian Kernel feature space, which is discussed next.

4.2. Geometric Properties of Hyperspheres in Gaussian Kernel Feature Space. In feature space, after the tuning-scaled SVDD is conducted, data are mapped into the hyper sphere. Suppose this sphere being S, with the center being a. Since $K(x, x) = \langle\Phi(x), \Phi(x)\rangle = \exp(-q\|x - x\|^2) = 1$, it means that all data are located on the surface of the unit ball of the feature space. Denote the unit ball as B, with the center as the original O of the feature space. Then data actually spread on the cap-like surface intersected by S and B. For semi-SVs, they are located on the surface of S and B simultaneously; therefore, they appear on the rim of the cap. Denote the center of the cap as a'. This section presents that the geometric property of semi-SVs and two properties of hyper spheres in the feature space. Before giving the theoretical proof, some existing statements of Gaussian Kernel feature space [11] are mentioned first.

Denote V as the set of SVs. In [11], for any $v_i, v_j \in V$, in feature space, they have the same angle with Oa'; that is $\angle(\Phi(v_i)Oa') = \angle(\Phi(v_j)Oa')$. For any $v \in V$, it generates a cone with $\Phi(v)$ as the vertex, $O\Phi(v)$ vector as axis, and $\angle(\Phi(v)Oa')$ as the basic angle. That cone corresponds to a small sphere with v as the center and $\|v - \Phi^{-1}(a')\|$ as the radius. Whether a point within v's cone decides whether it has the same label with v. Those statements are summarized as the following lemma. See literature [11] to find more details.

Lemma 1. *For any $x \in X$ and $v \in V$, let $\theta = \angle(\Phi(v)Oa')$; there is:*

$$\angle(\Phi(v)O\Phi(x)) < \theta \Leftrightarrow \|v - x\| < \|v - \Phi^{-1}(a')\| \Leftrightarrow \Phi(x) \text{ is within } \Phi(v)\text{'s cone} \Leftrightarrow x \text{ is within } v\text{'s small sphere} \Leftrightarrow x \text{ and } v \text{ have the same cluster label.}$$

Although we name those points with nonzero support values as semi-SVs, they are also of the identical properties as SVs. Therefore, based on the previous lemma, we have the following geometric property of semi-SVs in feature space [12].

Corollary 2. *In feature space of Gaussian Kernel, semi-SVs are collected in terms of clusters on the intersection hyperline of S and B.*

In this study, we give two geometric properties about the hyper spheres produced by SVDD. Those two properties are expected to extend the proposed algorithm to the multi-classification issue in the future work.

Proposition 3. *If SVDD procedure is conducted on each class with the same Kernel scale, it ensures that multiple hyper spheres exist in the same feature space.*

Proof. For kernel-based algorithms, the kernel function defines a non-linear map Φ by $K(x, y) = \langle \Phi(x), \Phi(y) \rangle$. For Gaussian kernels, the scale is the only parameter to determine in the kernel functions and consequently the only factor to determine the feature space. As long as the scale remains the same, the spanned feature space is also the same. With the same scale, SVDD procedures conducted on each class will yield multiple hyper spheres in the same feature space. □

Proposition 4. *The maps of the same data created by different kernel-based methods are the same, provided that the Gaussian kernels have the same scale parameter.*

Proof. According to Proposition 3, the kernel scale is the only factor to determine the feature space where kernel works. As long as the kernel scale keeps the same, the nonlinear map Φ does not change; therefore, the image of the given data x mapped by Φ does not change. Of course, different kernel-based methods have different optimization objectives, so their optimization results are different. Take SVM and SVDD as examples. For a given x, with the same Gaussian kernel scale, SVM and SVDD will employ the same image, $\Phi(x)$, in their computation. As they have different optimization objectives, different solutions will be obtained. SVDD tries to find data located around the boundaries and SVM lets those data located around the margin area be equipped with non-zero support values to formulate the discriminative plane. □

Based on the previous two propositions, the next job is to probe the further geometric properties of SVs and try to investigate the relationship between geometric positions of SVs and the hyper spheres, with intention to develop the wise discrimination approach. That formulates the future work direction to address multiclassification issue by conducting SVDD procedures on each class and building the new labeling approach.

4.3. Validation of the Clustering Method. The validation of the clustering method is interpreted as follows.

Firstly, SVs are grouped in terms of clusters on the intersection circle, which means that their distribution in feature space has regular directions. Then angle information is a good criterion to detect their clusters. Angle distance is used in the way of Cosine value. Because SVs are on the surface of the unit ball, Cosine value is used in the way of inner product: $\langle \Phi(x_i), \Phi(x_j) \rangle$. In feature space, $K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$, so the first-phase method employs Kernel matrix as the affinity matrix.

Secondly, according to spectrum analysis idea, if we eigendecompose the affinity matrix of one dataset, the inherent distribution information of the dataset can be revealed. This type of distribution information is described by eigenvectors because these vectors describe the distribution directions. For our algorithm, the eigendecomposition on the kernel matrix yields the eigenvectors that describe the distribution directions based on angle information. The subsequent clustering operations are well expected to produce correct clusters.

5. Second Phase: Label Assignment

With labeled DRs in hand, we label data using NN classifier. Like any a distance-based classifier, NN is faced with two scenarios of clusters: the first is convex-shaped clusters, where spatial information can reflect cluster structures; the other is nonconvex clusters. In the later scenario, the Euclidean metric often fails and the cluster structures make importance in clusters detection. That can be illustrated in Figure 2.

In Figure 2(a), cluster2 and cluster3 are convex shaped, so Euclidean distance meaning is consistent with cluster membership. However, the cluster1 is string shaped, and Euclidean metric cannot provide accurate clusters. So in Figure 2(b), for x, y of cluster1, for z of the Cluster2, according to cluster structure, ideal members of the same cluster should have less distance than that of members of different clusters; that is, there should be $\|x - y\| < \|x - z\|$. But, with Euclidean metric, there is $\|x - y\| = 0.5356$, $\|x - z\| = 0.2239$; and there is $\|x - y\| > \|x - z\|$. To solve this problem, cluster shapes should be taken into consideration when the metric is defined. The next section touches that topic.

5.1. New Metric Definition. The key of new metric is to investigate multireachable paths between two points including the path of straight line segment that directly connects two points, and the indirect paths that consist of multiline segments. The final distance is determined by the shortest path with minima path length. In computing length of indirect paths, neighboring information is exploited, and the neighboring information spreads along paths so that the message of global cluster structures is accumulated gradually and integrated into metric formulation.

Given dataset $X = \{x_1, x_2, \ldots x_N\}$, we view it as a complete undirected graph, with graph node as data point and edge weight as distance of a pair of points. Inspired by [13], we firstly define edge weight that directly connects x and y as follows:

$$W(x, y) = \eta_{xy} \left[\exp\left(\|x - y\|^2 \right) - 1 \right]. \quad (4)$$

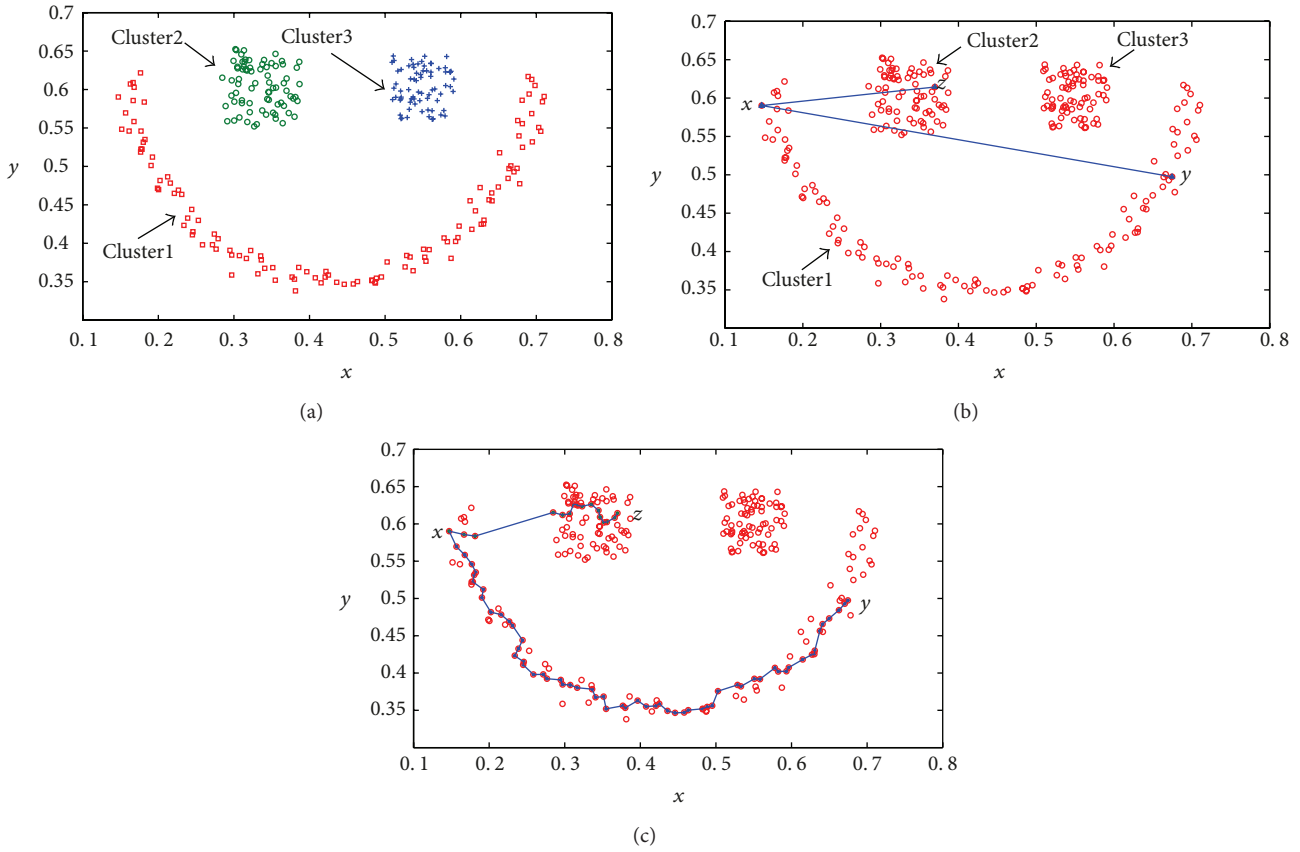


FIGURE 2: Dataset1 (a), the shortest path of Euclidean (b), and the shortest paths of $\|\cdot\|_*$ (c).

Here, Euclidean distance is rescaled by the exponential mechanism with intention to reduce the distance between similar points while enlarging distance between distinct points. That actually strengthens cluster boundaries. η_{xy} is the relative density ratio, defined as

$$\eta_{xy} = 2 - \frac{\min\{\mathrm{AD}(x), \mathrm{AD}(y)\}}{\max\{\mathrm{AD}(x), \mathrm{AD}(y)\}}, \tag{5}$$

with $\mathrm{AD}(x) = \mathrm{ave}_{u\in\mathrm{Nei}(x)}\{\|x-u\|\}$.

$\mathrm{Nei}(x)$ represents x's neighborhood. $\mathrm{AD}(x)$ is the average distance of $\mathrm{Nei}(x)$, which reveals the density information of $\mathrm{Nei}(x)$. If $\mathrm{AD}(x)$ is close to $\mathrm{AD}(y)$, then η_{xy} tends to 1, and $W(x,y)$ tends to $\exp(\|x-y\|^2) - 1$. If $\mathrm{AD}(x)$ is sharply different from $\mathrm{AD}(y)$, then η_{xy} tends to 2, and $W(x,y)$ tends to $2[\exp(\|x-y\|^2) - 1]$. The employment of η_{xy} avoids the distorted distance value caused by the unbalance of local distributions between two points. Therein, $\mathrm{Nei}(x)$ consists of some top nearest points to x. $\mathrm{Nei}(x)$ size is set as 5% of the whole dataset size.

Then the path length is defined. For a path p outgoing from x and ingoing to y, $p = \{x = z_1, z_2, \ldots z_m = y\}$, its length is defined as

$$\mathrm{PL}(p) = \sum_{i=1}^{m-1} W(z_i, z_{i+1}). \tag{6}$$

Then for x, y, their cluster-structure-dependent distance is the length of the shortest path. Consider

$$\|x-y\|_* = \min_{p\in\mathrm{path}[x,y]}\{\mathrm{PL}(p)\}. \tag{7}$$

Therein, $\mathrm{path}[x,y]$ is the set of all paths connecting two points. The new metric searches the shortest path among multipaths, so it breaks the limit of Euclidean metric, which takes the straight path that connects two points directly as the shortest path. Since the distance between similar points is small, it is absolutely possible that an indirect path that crosses over a series similar points is shorter than a direct path that crosses distinct points. Still, look at the example of Figure 2(b). With the new metric, there is $\|x-z\|_* = 0.0135$, $\|x-y\|_* = 0.0126$; consequently, $\|x-z\|_* > \|x-y\|_*$. Their shortest paths are illustrated in Figure 2(c). Those distance values are consistent with cluster membership. It shows that $\|\cdot\|_*$ does address the influence brought by nonconvex distribution of clusters.

5.2. Shortest Distance Computation. We now give the computation steps of the new metric. Different from classical Dijstra algorithm [13], in this paper, we care for the paths between pairs of points and compute the shortest distance values concerned with these paths simultaneously; that is, for the point pair (s,t), the distance of paths outgoing from s is computed, and the distance of paths ingoing at t is computed

PairShortD(s, t)
(1) $Ds[1:N] = \infty$; $Dt[1:N] = -\infty$; $sLeft = X$; $tLeft = X$;
(2) while ($NotEmpty(sLeft)$)
(3) { $u = \min_{j \in sLeft}\{Ds(j)\}$; $sLeft = sLeft - \{u\}$;
(4) for ($i = 1$; $i <= N$; i ++)
(5) if ($Ds(i) > Ds(u) + W(u, i)$)
(6) $Ds(i) = Ds(u) + W(u, i)$; }
(7) while ($NotEmpty(tLeft)$)
(8) { $u = \max_{j \in tLeft}\{Dt(j)\}$; $tLeft = tLeft - \{u\}$;
(9) for ($i = 1$; $i \le N$; i ++)
(10) if ($Dt(i) < Dt(u) - W(i, u)$)
(11) $Dt(i) = Dt(u) - W(i, u)$; }
(12) Fill Ds into *ShortD*'s s^{th} row and s^{th} column;
(13) Fill Dt into *ShortD*'s t^{th} row and t^{th} column.

Algorithm 1

at the same time. Denote the shortest distance over the whole dataset as matrix *ShortD*, then the previous computation fills the sth row and the tth column of *ShortD*. When it comes to the undirected graph, the entries of the sth column and the tth row get values simultaneously.

For the N-sized dataset X, we compute the distance of pairs of points in the following order: $(x_1, x_2), (x_3, x_4), (x_5, x_6), \ldots, (x_{N-1}, x_N)$. After all distances of paths between the prvious pairs of points are obtained, the whole *ShortD* can be specified. That idea forms an iterative procedure of the shortest distance computation as follows.

ShortDistance Computation

(1) for ($i = 1$; $i < N$; $i = i + 2$);
(2) {*PairShortD*$(i, i + 1)$;}.

Therein, procedure *PairShortD* computes distance values of point pair (x_i, x_{i+1}), $(i = 1 \cdots N - 1)$. Its details are listed in Algorithm 1.

In *PairShortD*, *Ds* and *Dt* are N-sized arraye, and they keep track of the latest shortest distance with respect to s and t, respectively. They are updated all the time. *sLeft* and *tLeft* preserve the points to be explored. Note that *Dt* uses minus value to show that it is the length of ingoing path that has the opposite direction with the outgoing path. The employment of minus value in *Dt* makes *PairShortD* procedure able to be applied in the directed graph.

5.3. Test New Metric. To check the performance of $\|\cdot\|^*$, it is compared with Euclidean metric and $\|\cdot\|_0$ in synthetic datasets, as shown in Figures 3, 4, 5, and 6. $\|\cdot\|_0$ is defined as follows without using relative density ratio η_{xy}:

$$W_0(x, y) = \left[\exp\left(\|x - y\|^2\right) - 1\right]. \quad (8)$$

Three metrics are introduced into K-means method to observe clustering results. Clearly, Euclidean metric fails in dataset2 and dataset3 due to its rigid dependence on the sole path. $\|\cdot\|_0$ works well in dataset2 but fails in dataset3. The outperformance of $\|\cdot\|_*$ over $\|\cdot\|_0$ shows the importance of η_{xy}. That can be explained clearer by the shortest paths produced by three metrics, as shown in Figure 5(b) and Figure 6. Obviously, in dataset3, the shortest path yielded by $\|\cdot\|_0$ is attracted by the dense cluster, while the path yielded by $\|\cdot\|_*$ can cross the same cluster region.

6. Fast Training Approach

The DRs' extraction process of SDSN is essentially a quadratic optimization problem, which consumes huge cost. So, we give a fast training approach based on incremental learning to facilitate the efficiency.

Like the common incremental method, our approach sets a working set and divides remainder data into batches. Those batches will be enriched into the working set incrementally. The novel in our approach is that the batches are sorted in an order that benefits the incremental development of the problem solution. For SVDD optimization problem, the solution is the support values $\{\beta_i\}$. Therefore, we select those data that could do significance on solution development to form the initial working set and specify the batches according to their importance to the model formulation. Those data batches will be used to update the working set in turn. The reason to consider the importance of data lies in the fact that if the important data are processed firstly, they can assist to build a promising solution script that readily accepts the following gentle adjusting. Yet, if important data are inputted late, sharp adjusting operations might happen when the ending batches are enriched into the working set. That could bring undesired affect to the formulation of final solution. Based on this thinking, we give the method to generate the sorted list of data batches based on Schrödinger equation [9], whose brief introduction is given firstly.

6.1. Schrödinger Equation. Schrödinger equation (SE) [9] comes from quantum mechanics of physics field. It is written as

$$H \cdot \Psi(x) = \left(-\frac{\sigma^2}{2}\nabla^2 + P(x)\right)\Psi(x) = e \cdot \Psi(x). \quad (9)$$

In quantum mechanics, SE describes the law of energy conservation of a particle, where e is the energy, $P(x)$ is the Schrödinger potential, and ∇ is the Laplacian. $\Psi(x)$ corresponds to the state of a quantum system, and $\Psi(x)$ can be explained as the wave function of one particle. Conventionally, given $P(x)$, the equation is solved to find solution $\Psi(x)$, so particle orbits are obtained.

To apply SE in machine learning, $\Psi(x)$ is regarded as probability distribution of dataset, and then the maxima of $\Psi(x)$ associate with cluster centers. Instead of looking for $\Psi(x)$'s maxima, potential $P(x)$'s minima are probed because under physical meaning the latter's minima correspond to the former's maxima. The advantage of investigating $P(x)$ rather than $\Psi(x)$ lies in the robustness of $P(x)$ towards the variation of parameter σ. As long as σ is within a moderate range, $P(x)$'s minima can reflect cluster centers correctly. That forms

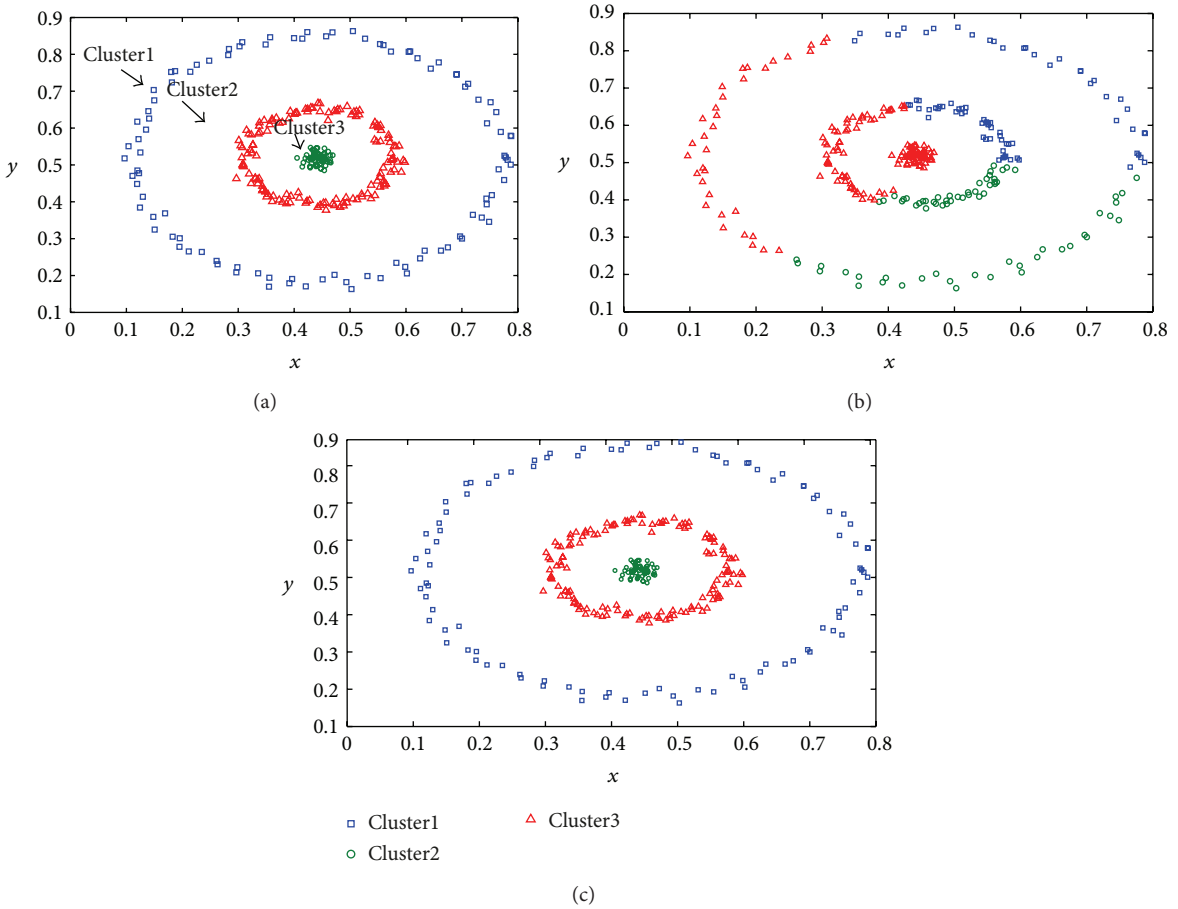


FIGURE 3: Dataset2 (a), clusters of Euclidean (b), and clusters of $\|\cdot\|_0$, $\|\cdot\|_*$ (c).

an inverse fashion of SE solving process. For a $\Psi(x)$, $P(x)$ is solved as

$$P(x) = e + \frac{\sigma^2}{2\Psi(x)}\nabla^2\Psi(x). \tag{10}$$

Usually, $\Psi(x)$ is simulated by the empirical distribution

$$\Psi(x) = \sum_i \exp\left(-\frac{\|x - x_i\|^2}{2\sigma^2}\right). \tag{11}$$

Then (10) is rewritten as

$$P(x) = e - \frac{n}{2} + \frac{1}{2\sigma^2}\cdot\frac{\sum_i \|x - x_i\|^2 \exp\left(-\|x - x_i\|^2/2\sigma^2\right)}{\sum_i \exp\left(-\|x - x_i\|^2/2\sigma^2\right)}. \tag{12}$$

Let us furthermore require $\min P(x) = 0$. That sets the value of e as

$$e = -\min\frac{\sigma^2}{2\Psi(x)}\nabla^2\Psi(x). \tag{13}$$

If we restrict the computation of $P(x)$ within the finite dataset, there is

$$P(x_i) = e - \frac{n}{2} + \frac{1}{2\sigma^2}\cdot\frac{\sum_i \|x_i - x_j\|^2 \exp\left(-\|x_i - x_j\|^2/2\sigma^2\right)}{\sum_i \exp\left(-\|x_i - x_j\|^2/2\sigma^2\right)}. \tag{14}$$

From geometry meaning, $P(x)$'s minima tell cluster centers, so $P(x)$'s maximum indicates the boundary information of clusters. That inspires us to use Schrödinger equation to define reduction strategy.

6.2. Fast Training Approach. It has been known that $P(x_i)$ tells the possibility of x_i how much it gets close to cluster boundaries. The bigger $P(x_i)$ is, the more possibly x_i gets close to boundaries, and the more possible it is an SV. This means that x_i makes importance in formulating solution $\{\beta_i\}$ and x_i should be added into working set early. According to $P(x_i)$ value, all data are sorted as $\{x_{(1)}, \ldots, x_{(N)}\}$, subject to $\{P(x_{(1)}) \geq P(x_{(2)}) \geq \cdots \geq P(x_{(N)})\}$. Specify the size of working set as M and the size of batch as B, with $B < M$. M and B can be determined by storage requirement. Dataset

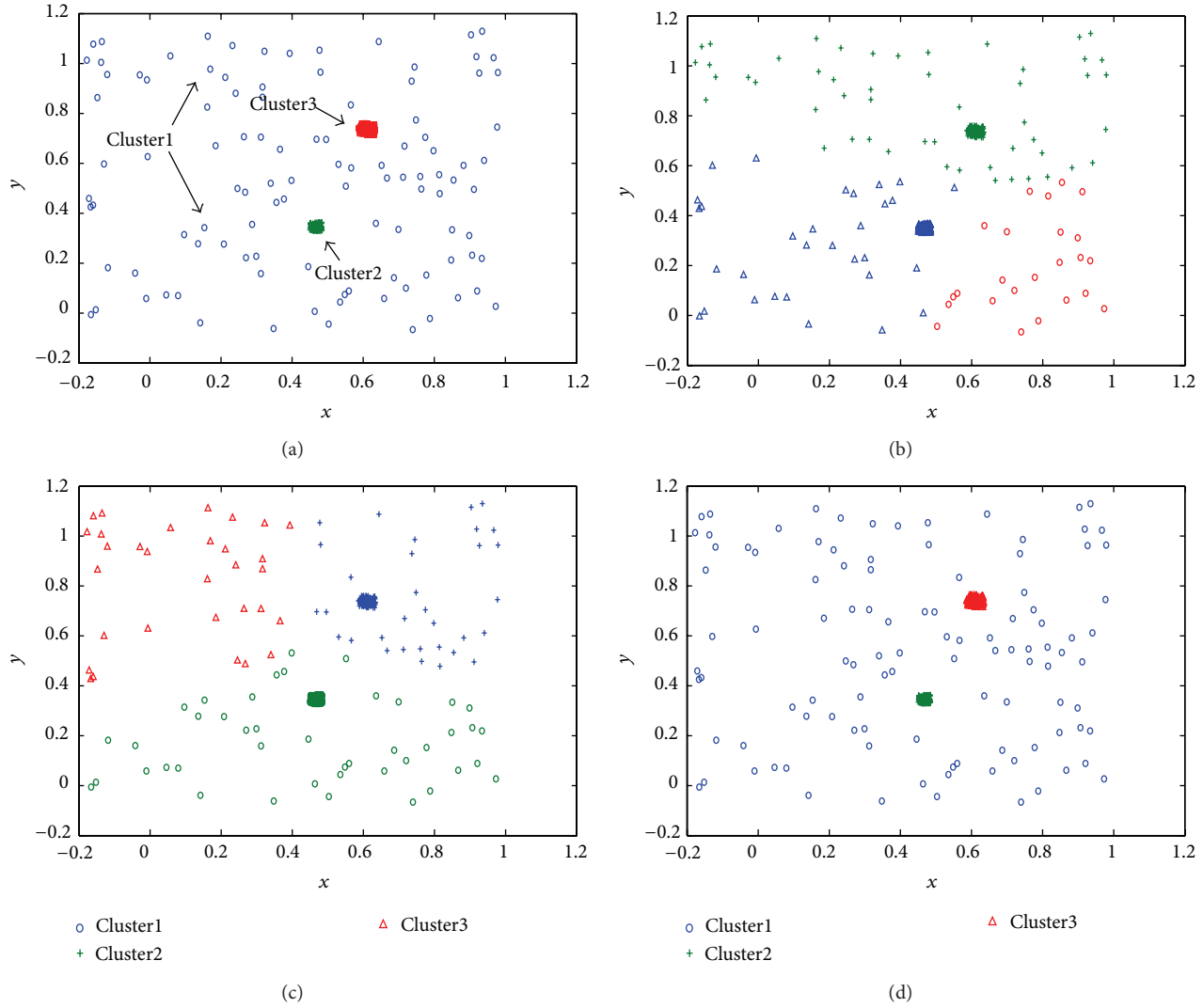


FIGURE 4: Dataset3 (a), clusters of Euclidean (b), clusters produced by $\|\cdot\|_0$ (c), and clusters produced by $\|\cdot\|_*$ (d).

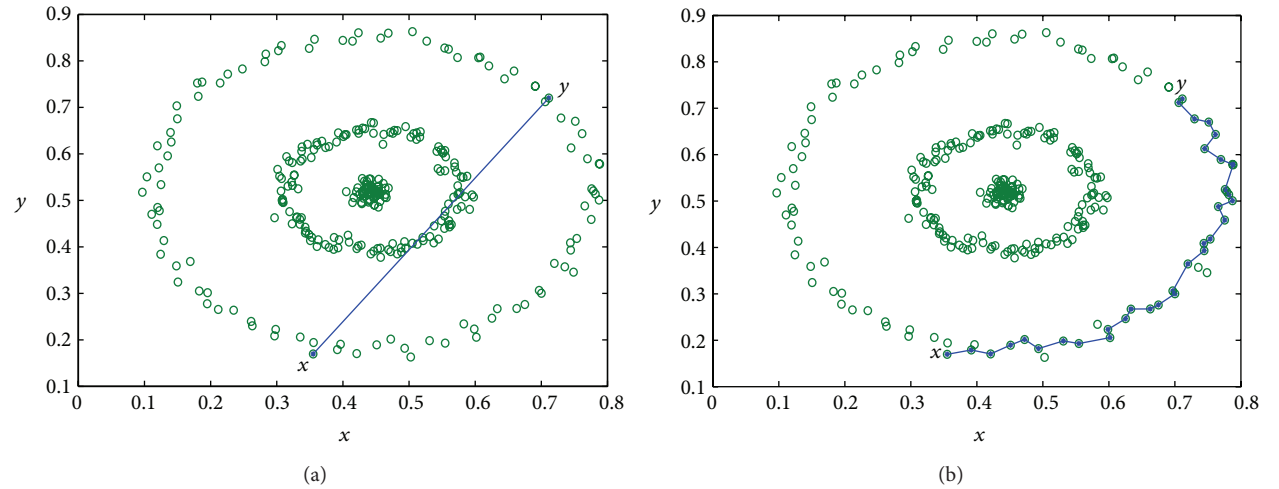


FIGURE 5: The shortest path produced by Euclidean (a) and the shortest path produced by $\|\cdot\|_0$ and $\|\cdot\|_*$ (b).

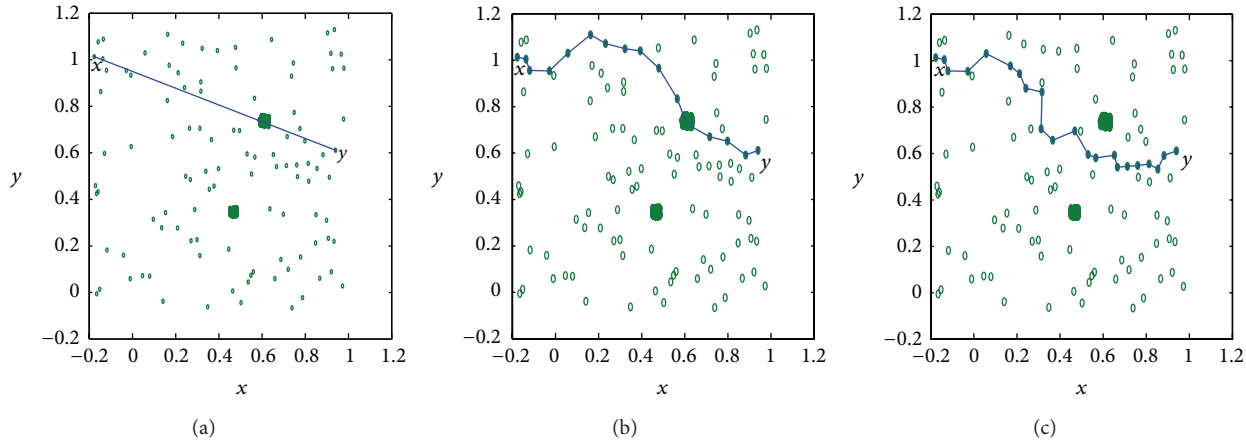


Figure 6: The shortest path produced by Euclidean (a), the shortest path produced by $\|\cdot\|_0$ (b), and the shortest path produced by $\|\cdot\|_*$ (c).

X is divided into 1 working set and T batches, $U_0, U_1, \ldots, U_T$, and there is

$$U_0 = \{x_{(1)}, \ldots, x_{(M)}\}, U_1 = \{x_{(M+1)}, \ldots, x_{(M+B)}\}, \ldots, \quad U_T = \{x_{(M+(T-1)B)}, \ldots, x_{(M+TB)}\}. \tag{15}$$

In our fast training approach, U_0 is the initial working set. The initial solution $\{\beta_i\}$ is obtained on U_0. Then sort the data of working set according to their support values $\{\beta_i\}$ in the ascending order. Replace the top B data of the sorted working set with U_1, so as to formulate the updated working set. The new solution is computed on the new working set again. Repeat the previous procedure until all data batches are input.

6.3. Test Fast Training Approach. Now check the quality of fast training approach. In experiments, SVDD optimization process is conducted in three ways: (1) nonincremental training, (2) the incremental training, and (3) the proposed fast training. In case (2), dataset is divided into batches randomly. For two incremental approaches, the working set size M is 20% of dataset size, and the batch size is the 10% of working set size. Experiments aim to compare (1) versus (2) and (1) versus (3), respectively. The criterion is the number of shared SVs between (1) versus (2) and between (1) versus (3). The number of SVs and the number of shared SVs are recorded in column "no. of SV" and "no. of shared" of Table 1.

Datasets are taken from UCI benchmark machine learning repository. In Table 1, sizes of datasets are indicated in the bracket following the names of datasets. Pay attention to letter dataset. It has 20000 data covering 26 letters from "a" to "z". Here, 260 random samples are selected from classes "a" and "b", respectively, to form the 520-sized experimental subset.

From Table 1, it is easy to find that the number of SVs generated by incremental approaches is less than that of nonincremental approach. Between (2) and (3), (3) has shared more SVs with approach (1) than (2). So, the proposed approach holds better fast training ability than the usual method.

Table 1: Performance comparison of fast training approaches.

Dataset (size)	(1) no. of SVs	(2) no. of SVs	(2) no. of shared	(3) no. of SVs	(3) no of shared
Wine (178)	22	21	18	21	20
Liver (345)	43	40	39	41	40
Monk3 (432)	102	96	92	95	95
Letter (520)	115	115	102	112	108

Besides, we record the time consumption of three processes in Table 2 to check the speed of our fast training method. According to Table 2, the obvious improvement of the proposed fast training method over the other two processes in efficiency is found.

7. Experiments

7.1. SDSN versus SVC and SVC Variants. Since SDSN is designed based on support vector clustering method, experiments will be done to compare the performance of SDSN with support vector clustering (SVC) as well as SVC variants firstly. To be self-included, here SVC and some variants are introduced in brief. SVC consists of optimization piece and labeling piece. The original labeling procedure is named as complete graph (CG). CG identifies clusters by constructing a complete graph and taking connected components of graph as clusters [3]. The adjacency matrix A is

$$A_{ij} = \begin{cases} 1 & \forall y,\ y \in \text{line}\left[x_i, x_j\right],\ R(y) \le R \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

Therein, line$[x_i, x_j]$ is the line segment between x_i and x_j. $R(y)$ is the distance between $\phi(y)$ and the center of hypersphere in the feature space. R is the radius of the hypersphere. Clusters are defined as the connected components of the graph induced by A. Clearly, randomness encoded in CG degrades clustering accuracy. Moreover, the number of sampling points creates the dilemma between clustering quality and time cost.

Table 2: Time comparison of fast training approaches (seconds).

Methods	Wine	Liver	Monk3	Letter
(1)	2.32	68.1	165.2	112.7
(2)	0.52	1.58	6.72	3.93
(3)	0.10	0.74	3.70	1.29

Table 3: Clustering accuracy comparison (%).

	CG1	CG2	CG3	SVG	PG	GD	CCL	SDSN
(1)	84.9	85.2	85.8	81.4	—	86.5	86.6	86.6
(2)	82.8	83.2	83.9	80.3	—	80.0	84.5	84.3
(3)	78.6	79.1	79.6	78	—	78.7	79.6	79.5
(4)	66.3	67.5	67.8	65	—	68	68.3	68.5
(5)	66.3	67.2	67.8	65.2	—	65	69	68.6
Wine	94.8	94.8	95.5	93	94.3	95	96.6	95
Liver	70.0	70.3	70.5	69.1	68.1	70.0	71.3	71.3
Monk3	96.6	96.6	97.1	95.8	95.3	96.1	97.1	97.3
Letter	87.2	87.2	87.8	87.0	86.9	85.3	88.5	88.2

Variants of SVC focus on revising its labeling process. support vector graph (SVG) [3] is an example that computes the adjacent matrix and connected components only with respect to SVs. So, it obtains dramatic time reduction but simultaneously sees a decrease in clustering accuracy. Proximity Graph (PG) [14, 15] also computes adjacent matrix among SVs, but it takes some simulation approach to derive connected components. Usually, PG employs Delaunay triangulation (DT) [15] as the simulation approaches. PG consumes less time than SVG and produces poorer result than SVG. Another method, gradient descendent (GD) [16], builds adjacent matrix and connected components based on stable equilibrium points (SEPs). Each SEP represents some data within its neighborhood. Data is labeled the same membership as its representative SEP. All these methods are all encoded with randomness, and they gain the improvement of time cost at the price of the decrease of clustering quality. Recently, cone cluster labeling (CCL) is given in [10] to do label assignment based on geometric observation. CCL overcomes randomness and achieves better performance than other methods.

Besides the previous datasets, news groups dataset is tested. This dataset contains about 20,000 articles divided into 20 news groups. They are named as NG1, NG2, ..., NG20. The usual *tf.idf* weighting schema is exploited to express documents, andsome words that appear very few times are deleted. Normalize each document vector to make them have the unit length. For empirical ease, some classes are sampled randomly to form six experimental sets: (1) NG1, NG2, NG8 (180); (2) NG2, NG7, NG9, NG14 (220); (3) NG1, NG3, NG5, NG6 (260); (4) NG17, NG18, NG19 (220); (5) NG12, NG13, NG14, NG15 (200). The number in the bracket indicates the number of random samples of each class. Clustering accuracies are recorded in Table 3. Note that PG has no result in news group, because DT used by PG is infeasible in high-dimensional data space. In computing adjacent matrix, CG runs with 3 different sampling numbers: CG1 with sampling 15 data; CG2 with sampling 23 data; CG3 with sampling 30 data. The aim is to observe the influence of sampling number on clustering accuracy. For SVG, PG, and GD, their sampling number is set as 15.

From Table 3, it is clear that for CG family, with sampling number increasing, clustering accuracy goes up, which is natural since higher sampling number can reduce randomness. Of course, this improvement is at the price of the huge increase of time cost. Note that sometimes the difference between three CG methods is subtle. One reason is that sometimes data distribution is bound to make influences on clustering results no matter how many samples are selected. The other reason lies in the randomness coming from random sampling, which always leads to unexpected results. To some extent, CG's performance is controlled by sampling number. If the sampling number gets to the infinity, CG would ideally present good result. SVG is a rough version of CG1, so its work is not as good as CG1. PG follows SVG due to its approximation operations. There is clear instability in GD's behaviors since it does well in datasets (1), (3) and (4), while it does poorly in (2), (5), and letter. Contributing factor for that is GD's susceptibility on data distribution and dimensionality. Only in some certain datasets, GD can generate qualified SEPs to present accurate neighborhood information and consequently to produce good results. Compared with these methods, CCL and SDSN show clear advantage due to their removing of randomness. Basically, CCL achieves the top place by doing the best job in most datasets. However, CCL's good behavior is at the price of expensive cost that is spent on investigating Kernel scale parameter list. SDSN's behaviors follow CCL by producing the optimal result in 4 of 9 datasets and doing well in the remainder datasets except Wine dataset. The reason of SDSN's failure is the fact that Wine data has 178 points but covers 13 dimensions, which makes the neighborhood information uninformed, and thus the tuning strategy cannot play its role. Generally speaking, SDSN has outperformance over the original CG approach, which illustrates the improvement of the proposed clustering idea over the classical unsupervised learning idea. And SDSN has the competitive performance with the popular SVDD variant, which verifies SDSN's fine clustering ability. Figure 7 illustrates the time consumption of these methods in one running, which shows SDSN's time efficiency as competitive with CCL.

7.2. SDSN versus Clustering Methods. Now, compare SDSN with some clustering methods: K-means [17], Girolami [18], NJW [4], and NI [19]. Girolami is a Kernel-based method that depends on an expectation-maximum process. NJW is a representative of spectrum clustering methods. NI is an agglomerative clustering method. It creates the distance definition according to information entropy and takes this information-based distance as clustering criteria to accumulate subclusters. Clustering accuracies of these methods are listed in Table 4.

Among five methods, it can be found that NJW is the best clustering tool. It probes spectra of all data, to obtain data inherent distribution directions, which assists much in

Table 4: Comparison of clustering accuracies of methods (%).

	K-means	Girolami	NI	NJW	SDSN
(1)	79.4	83.9	82.8	86.7	86.6
(2)	81.8	83.7	80.2	85	84.3
(3)	75	78	73	81.1	79.5
(4)	66.7	68	67.5	71.6	68.5
(5)	65.3	66.2	66.8	68.9	68.6
Wine	94	95.7	96	97.5	95.8
Liver	71.1	72.6	70.3	73.1	71.3
Monk3	96.2	97	96.4	97.3	97.3
Letter	86.9	87.2	87.5	90.5	88.2

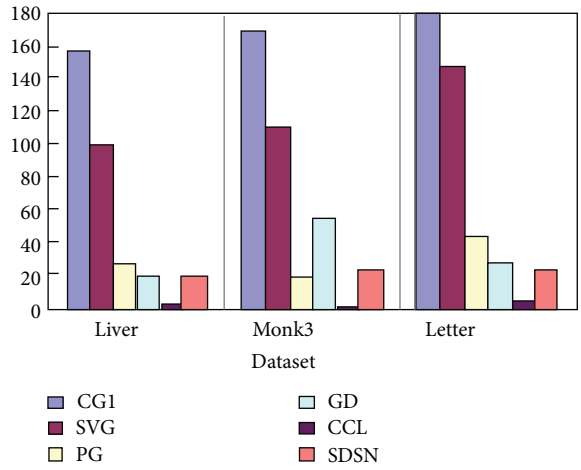


Figure 7: One-run time consumptions of methods (second).

detecting correct clusters. But NJW consumes huge cost, with $O(N^3)$ time complexity. For Girolami, NI, and K-means, their performance decreases in turn. NI shows instability in experiments. It is because, as an agglomerative approach, NI is fairly easy to be affected by data input order. Although the information-based metric helps a lot to provide accurate distance information, NI still exhibits unstable behaviors. K-means is a hard partition process completely dependent on Euclidean metric. When the Euclidean metric does not match data distribution, more errors will happen. So, K-means does not behave well. SDSN does the best job in Monk3 and follows closely NJW in (1), (2), and (5), which demonstrates SDSN's fine clustering performance. In the remainder cases, the difference between SDSN and NJW is acceptable. These experiments verify the correctness and validation of the proposed clustering idea. And if we take both the performance and the efficiency into consideration, we will find that SDSN is of higher popularity than its peers.

8. Conclusion

This paper presents a general clustering framework that extracts and clusters DRs firstly and then classifies non-DRs. The idea is implemented by an algorithm SDSN. SDSN consists of a tuning-scaled SVDD optimization that extracts DRs, a spectrum-analysis-based method to cluster DRs, the nearest neighbor classifier that works with an informed metric to classify non-DRs, and the fast training approach that facilitates the computation ease. To illuminate the validation of the spectrum-analysis-based method, the geometric property of Gaussian Kernel feature space is explored and proofed. Experiments demonstrate the performance of the tuning-scaled SVDD, of the new metric, and of the fast training approach. The improved popularity of SDSN over the pure clustering approach is verified too.

Since the supervised information can be utilized in the unsupervised learning process, it is well expected to obtain help from clustering information for the classification mining task. The future work direction is focused on solving multi-classification issue based on the geometric properties of multihyperspheres of the feature space, as mentioned in Section 4.2.

Acknowledgments

This research is partially supported by the National Natural Science Foundation of China under Grants nos. 61105129, 61304174, and 11226146 and the Natural Science Foundation of Jiangsu Province of China under Grant no. BK2011581.

References

[1] I. Jonyer, L. Holder, and D. Cook, "Graph-based hierarchical conceptual clustering," *International Journal on Artificial Intelligence Tools*, vol. 10, pp. 107–135, 2001.

[2] K. J. Anil, "Data clustering: 50 years beyond K-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

[3] A. Ben-Hur, D. Horn, and H. T. Siegelmann, "Support vector clustering," *Journal of Machine Learning Research*, vol. 2, pp. 125–137, 2001.

[4] A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: analysis and an algorithm," *Advances in Neural Information Processing Systems*, vol. 14, pp. 849–856, 2001.

[5] X. Liu and J. Cao, "Robust state estimation for neural networks with discontinuous activations," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 40, no. 6, pp. 1425–1437, 2010.

[6] X. Liu, J. Cao, and W. Yu, "Filippov systems and quasi-synchronization control for switched networks," *Chaos*, vol. 22, Article ID 033110, 2012.

[7] X. Liu, T. Chen, J. Cao, and W. Lu, "Dissipativity and quasi-synchronization for neural networks with discontinuous activations and parameter mismatches," *Neural Networks*, vol. 24, no. 10, pp. 1013–1021, 2011.

[8] X. Liu, N. Jiang, J. Cao, S. Wang, and Z. Wang, "Finite-time stochastic stabilization for BAM neural networks with uncertainties," *Journal of the Franklin Institute*, vol. 350, no. 8, pp. 2109–2123, 2013.

[9] D. Horn and A. Gottlieb, "Algorithm for data clustering in pattern recognition problems based on quantum mechanics," *Physical Review Letters*, vol. 88, no. 1, Article ID 018702, 2002.

[10] T. Zheng, L. Xiaobin, and J. Yanwei, "Disturbing analysis on spectrum clustering," *Science in China*, vol. 37, no. 4, pp. 527–543, 2007.

[11] S.-H. Lee and K. M. Daniels, "Cone cluster labeling for support vector clustering," in *Proceedings of the 6th SIAM International Conference on Data Mining*, pp. 484–488, April 2006.

[12] P. Ling, C. G. Zhou, and Z. Xu, "Improved support vector clustering," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 4, pp. 552–559, 2010.

[13] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[14] J. Yang, V. Estivill-Castro, and S. Chalup, "Support vector clustering through proximity graph modeling," in *Proceedings of 9th International Conference on Neural Information Processing*, pp. 898–903, 2002.

[15] D. T. Lee and B. J. Schachter, "Two algorithms for constructing a Delaunay triangulation," *International Journal of Computer & Information Sciences*, vol. 9, no. 3, pp. 219–242, 1980.

[16] J. Lee and D. Lee, "An improved cluster labeling method for support vector clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 3, pp. 461–464, 2005.

[17] P. S. Bradley and U. M. Fayyad, "Refining initial points for k-means clustering," in *Proceedings of 15th International Conferenc on Machine Learning*, pp. 91–99, 1998.

[18] M. Girolami, "Mercer kernel-based clustering in feature space," *IEEE Transactions on Neural Networks*, vol. 13, no. 3, pp. 780–784, 2002.

[19] S. F. Ding, Z. Z. Shi, F. X. Jin, and S. Xia, "Direct clustering algorithm based on generalized information distance," *Computer Research and Development*, vol. 44, no. 4, pp. 674–679, 2007.