Novel Three-Phase Clustering based on Support Vector Technique

Ping Ling
College of Computer Science and Technology, Xuzhou Normal University, Xuzhou, China
Email: lingicehan@yahoo.cn

Xiangsheng Rong, Xiangyang You
Training Department, Xuzhou Air Force College of PLA, Xuzhou, China
Email: rxs12@126.com, xyyou@126.com

Ming Xu
Department of Logistic Command, Xuzhou Air Force College of PLA, Xuzhou, China
Email: mingxu@sina.com

***Abstract*—As an important issue of machine learning, clustering receives much care in recent years. Among all clustering approaches, most of them conduct clustering operations on overall data. That is, they learn label information from all data. That comes across critical challenge in times of high-sized datasets. This paper proposes a novel Three-phase Labeling algorithm (TPL) based on SVC to overcome this problem. TPL consists of selecting data representatives (Data representatives), clustering (Data representatives) and then classifying non-Data representatives respectively. Support vector clustering process is modified to select qualified Data representatives in first phase. Spectrum technique governs the second-phase clustering task. Therein, the geometric properties of feature space, a new metric, and a tuning strategy of Kernel scale are used. In experiments on real datasets, TPL achieves clear improvement in accuracy and efficiency over its counterparts, and demonstrates highly competitive clustering performance in comparison with some state of the arts.**

***Index Terms*—Three-phase clustering, support vector clustering, data representatives, new metric, tuning strategy**

I. INTRODUCTION

Clustering methods focus on grouping data into clusters that yield maximum intra-similarity and minimum inter-similarity of clusters. Conventionally clustering methods are categorized into several branches: partition clustering, hierarchical clustering, density-based clustering, grid-based clustering, model-based clustering, boundary-detecting clustering, and some other approaches. As an appealing boundary detecting method, Support Vector Clustering (SVC) [1] finds Support Vectors (SVs) to describe cluster contours and fulfills clustering according to contour information. SVC can address diverse-shaped datasets and outliers, and by employing Kernel function, SVC is able to address highly structured data because of Kernel function's ability to map data from the input space to a feature space. In spite of much popularity in bioinformatics, marketing, fault detection etc., SVC is adversely affected by its expensive and poor-qualified labeling piece. Classical SVC's labeling approach constructs a complete graph and takes connected components as clusters, so named as CG (Complete Graph). The complete graph is represented by an adjacent matrix, whose development involves random sampling. That causes considerable randomness and degrades clustering accuracy. And the number of sampling points creates a tradeoff between clustering quality and time cost.

There have been literatures covering variants of CG to overcome these problems. Support Vector Graph (SVG) [1] is a natural modification. It computes the adjacent matrix and connected components with respect to SVs, so does a lot of time reduction, but it simultaneously experiences a drop in clustering quality. Proximity Graph (PG) [2] also computes adjacent matrix among SVs, but it takes some simulation approach to learn connected components. In existing implementations, Delaunay Triangulation (DT) [3, 4, 5, 6, 7], K-means etc. are alternatives of simulation. PG consumes less time than SVG, and produces worse result compared with SVG. Another method, Gradient Descendent (GD) [8] builds adjacent matrix and connected components based on Stable Equilibrium Points (SEPs). These SEPs are generated based on rich geometric and computing information. And each SEP represents some data or SVs within its neighborhood. Data is labeled the same membership as its SEP. Neighborhood specification is also in need of much computing operations.

These methods share the similar idea as CG, that is, to construct adjacent matrix and connected components that are encoded with randomness, and they obtain the improvement of time cost at the price of the decrease of clustering quality.

Recently, Sei-Hyung Lee proposed Cone Cluster Labeling (CCL) that removes randomness for SVC [9]. CCL is different from above variants. It also creates

doi:10.4304/jsw.8.4.955-962

adjacent matrix and connected components for SVs like SVG, but the formulation of adjacent matrix comes from geometric checking. It defines cone-shaped region for each SV in feature space. Whether two SVs belong the same cluster is determined by whether their cones are overlapping. Non-SVs are labeled according to the cone which they are located in. Through mapping cones back to input space, CCL carries out all computations in input space. CCL shows better performance than other methods empirically. However CCL still has heavy burden of cost, since it runs on a list of Kernel scale values; the final result is selected manually. Usually the scale is required to be fairly large to produce a great number of SVs, the smaller-sized cones, and consequently the desired result. This means a long scale list, that is, large cost. Another unpleasant fact is that when the scale is large, clusters begin to split unreasonably, which decrease algorithm's quality.

Focused on these problems, this paper presents a novel SVC-based algorithm that is equipped with three phases, TPL. TPL finds data representatives (Data representatives) firstly, and then clusters Data representatives, thirdly classifies non-Data representatives. Data representatives are produced by the support vector clustering process. The second phase is accomplished by spectrum technique. In third step, with help of the tuning strategy of Kernel scale parameter, non-Data representatives are labeled through the nearest neighbor classifier. Experiments on real datasets demonstrate the improvement of TPL over its peers in time consumption and clustering accuracy, and its competitive performance versus some state of the arts. The underlying idea of TPL can be generalized into an open three-phase clustering framework: find Data representatives, cluster Data representatives, and then classify other data. Some discussion about this idea is presented in the last section.

II. Overview Related Work

A. SVC

Given n-dimension dataset X, $X=\{x_1 \ldots x_N\}$. SVC aims to look for a minimum hyper sphere containing all data, which is expressed by below optimization problem:

$$\min_{R,\xi} \quad R^2 + C\Sigma_i \xi_i \tag{1}$$

$$\text{s.t.} \ \| \Phi(x_i) - a \|^2 \le R^2 + \xi_i , \ \xi_i \ge 0$$

There Φ is the non-linear map from the input space to the feature space, ξ_i is slack variables, a is sphere center of, R is the radius of sphere, and C is penalty parameter to tradeoff radius and slack variables. Transfer it to the Lagrange function, then the Wolfe dual, leading to:

$$\min_{\beta} \quad \Sigma_{i,j} \beta_i \beta_j K(x_i, x_j) - \Sigma_i \beta_i K(x_i, x_i) \tag{2}$$

Points with $0 < \beta_i <= C$ are SVs. K is the Gaussian Kernel: $K(x, y) = \exp(-q\|x-y\|^2)$. Cluster assignment is done based on an observation that given a pair of data points that belong to different components, any path that connects them must exit from the sphere. Therefore, that path contains a segment of points y such that $R(y) > R$, where $R(y)$ is the distance from y to a. This leads to an adjacency matrix A:

$$A_{ij} = \begin{cases} 1 & \forall y, y \in path(x_i, x_j), \ R(y) \le R \\ 0 & otherwise \end{cases} \tag{3}$$

Clusters are defined as the connected components of the graph induced by A. Clearly, both the expensive computation for this matrix and the sampling manner to choose point y deteriorates strictness and veracity of the algorithm.

B Spectrum Analysis

This paper employs Spectrum Analysis (SA) [10] to cluster Data representatives, so it is introduced in brief. SA spans the spectrum space through eigen-decomposing the pair-wise matrix of data, and clusters data there. The pair-wise matrix is usually the affinity matrix. Its main steps are: 1) Compute pair-wise matrix H; 2) Normalize H to form H' in some version; 3) Eigen-decompose H'; 4) Select top p eigenvectors and form spectrum matrix by stacking p eigenvectors in columns. 5) Cluster rows of spectrum matrix with a simple method and label x_i as the i^{th} row of spectrum matrix. p is specified by the number of eigen values that are larger than 1 [11].

III. Geometry Properties of Feature Space

A. Boundary Function

Through non-linear map implied by Kernel, data are embedded into the hyper sphere of feature space. Because $\|\Phi(x)\|^2 = <\Phi(x), \Phi(x)> = K(x, x) = 1$, all data are located on the surface of the unit ball. Name the sphere and the unit ball as S and B respectively. Then data are located on the intersection of S and B's surface. That intersection is shaped like a cap, named as *Cap*, as shown in Fig1. Assume a' as the center of *Cap*, and a' is the intersection point of Oa vector and B's surface. Since SVs are on the S's surface simultaneously, SVs are actually located on the rim of *Cap*. Obviously that rim is a hyper circle, which is expressed as:

$$\begin{cases} \|\Phi(x)\|^2 = 1 \\ \|\Phi(x) - a\|^2 = R^2 \end{cases} \tag{4}$$

According to byproducts of SVC optimization, there are:

$$\begin{cases} a = \Sigma_j \beta_j \Phi(x_j) \\ R^2 = 1 - 2\Sigma_i \beta_i \Phi(x_i)\Phi(x^*) - \Sigma_{i,j} \beta_i \beta_j \Phi(x_i)\Phi(x_j) \end{cases} \tag{5}$$

where x^* is a SV. Adopt a and R^2 into (4), we have:

$$\begin{aligned} & 1 - 2\Sigma_i \beta_i \Phi(x_i)\Phi(x) - \Sigma_{i,j} \beta_i \beta_j \Phi(x_i)\Phi(x_j) \\ & = 1 - 2\Sigma_i \beta_i \Phi(x_i)\Phi(x^*) - \Sigma_{i,j} \beta_i \beta_j \Phi(x_i)\Phi(x_j) \end{aligned} \tag{6}$$

That is:

$$\Sigma_i \beta_i \Phi(x_i) \cdot (\Phi(x) - \Phi(x^*)) = 0 \tag{7}$$

(7) is the curve of SVs, and is indeed the boundary function. Theoretically the intersection of surfaces of S and B is the hyper circle, while (7) is a linear version. That can be explained if thinking SVC as the one-class SVM [11].

One-class SVM's classification idea is to create a hyper plane that data points can be determined whether within dataset according to which side of the plane it is projected to. That hyper plane is reduced to a closed hyper sphere with minimal area. The corresponding target function is:

$$\min \quad \tfrac{1}{2} \| w \|^2 + C\Sigma_i \xi_i - b \tag{8}$$

$$\text{s.t. } w \cdot \Phi(x_i) \geq b - \xi_i \,,\ \xi_i \geq 0$$

Transfer (8) into Lagrange function, set to zero its derivatives with respect to w, b, and ξ_i, and it results to the same optimization function of (1), with $w = \Sigma\beta_i\Phi(x_i)$ and $b = w\cdot\Phi(x^*) = \Sigma\beta_i\Phi(x_i)\,\Phi(x^*)$. The hyper plane, $w\cdot\Phi(x^*) = b$, is rewritten as:

$$\Sigma_i \beta_i \Phi(x_i)\Phi(x) = \Sigma_i \beta_i \Phi(x_i)\Phi(x^*) \tag{9}$$

That is the same as (7). That means the minimum sphere of constructed by SVC accounts to a decision plane that separates dataset's occupying landscape from its complement space. According to this opinion, it is natural that the intersection circle corresponds to a linear version of feature space.

B. Geometry Properties of SVs

Given $V=\{v_i\}$ as the set of SVs. This paper proposes that SVs appear in terms of clusters on the rim. This conclusion is based on some previous statements about feature space geometry [10]. In [12], we verify below lemma and corollary.

[Lemma] For any $x\in X$, $v\in V$, let $\theta = \angle(\Phi(v)\,Oa')$, there is:

$$\angle(\Phi(v)O\Phi(x)) < \theta$$

$$\Leftrightarrow \quad \| v - x \| < \| v - \Phi^{-1}(a') \|$$

$\Leftrightarrow$ $\Phi(x)$ is within $\Phi(v)$'s cone

$\Leftrightarrow$ x is within v's small sphere

$\Leftrightarrow$ x and v have same cluster label.

[Corollary] In feature space of Gaussian Kernel, SVs are collected in terms of clusters on the intersection hyper line of S and B.

Above lemma and corollary guarantee the validation of second phase of proposed algorithm.

IV. TPL ALGORITHM

Bearing basic information of SVC and geometric properties of feature space in mind, TPL find data representatives firstly, and then clusters data representatives, and finally it classifies non-data-representatives with pre-specified methods. More details are discussed in below sections.

A. Cluster Data representatives in Feature Space

In TPL, support vectors generated by SVC are used as data representatives. According to geometric property mentioned in above section, data representatives are grouped in terms of clusters on the rim, so their distribution has apparent and regular geometric directions. In this situation angle information is a good criterion for clustering. Angle distance between two Data representatives is expressed by *Cosine* value, that is, the inner product: $Cos(x_i, x_i) = \langle\Phi(x_i), \Phi(x_i)\rangle$. Here the spirit of SA is borrowed to group Data representatives, since SA exactly clusters data according to data distributing direction. Let the affinity measurement of SA take the inner product, which well avoids the explicit computation of $\Phi(x)$. Steps of first-phase clustering approach are: 1) Compute the pairwise matrix $H_{ij} = \langle\Phi(x_i), \Phi(x_j)\rangle = K(x_i, x_j)$. 2) Normalize H into H': $H'=\Lambda^{-1/2}H\Lambda^{-1/2}$, where $\Lambda = diag(\Lambda_i) = diag(\Sigma_j H_{ij})$. 3) Take top p eigen vectors as columns to form spectrum matrix; 4) Perform K-means on rows of spectrum matrix, with the cluster number being p. 5) Label x_i as the i^{th} row's cluster membership.

B. Classify Non-Data representatives

Non-Data representatives are classified by nearest neighbor classifier that is encoded with a new metric. The new metric integrates the information of feature space and input space, to look for the true neighbor. The new metric is:

$$\| x - y \|_*^2 = \lambda \| x - y \|^2 + (1-\lambda)\cdot(1 - K(x,y))/2 \tag{10}$$

Generally $\lambda = 0.5$, or it can be specified by background knowledge. The confidence of exploiting nearest neighbor rule is provided by the self-tuning strategy of Kernel scale parameter, which is described in the next section.

C. Self-tuning Strategy of q

The precondition to label data as its nearest DR is that Data representatives can serve as data representatives, say, they form a sketch of dataset. But Data representatives produced by conventional SVC optimization only describe cluster contours, without giving the information about inner-cluster structure. So these Data representatives are somewhat weak to act as data representatives. A solution to this problem is to increase Kernel scale q, so that contours become sharper and more Data representatives are yielded. However that leads to an unpleasant fact that clusters are split into non-instinctive and unreasonable sub-clusters and decreases clustering accuracy.

This paper proposes a self-tuning strategy for q, with intention to generate Data representatives that can serve as data representatives. In more details, for x, define its scale factor as $\sigma_x = \|x - x_r\|$. x_r is the r^{th} furthest point to x. σ_x reflects the local density information of x's neighborhood. If $\|x - x_r\| < \|y - y_r\|$, it means the neighborhood of x is denser than that of y. To measure Kernel affinity between x and y, their scale factors are combined into $q_{xy} = 1/\sigma_x\sigma_y$, which results to the modified Gaussian Kernel:

$$k(x,y) = \exp(-\frac{\|x-y\|^2}{\sigma_x \cdot \sigma_y}) = \exp(-\frac{\|x-y\|^2}{\|x-x_r\| \cdot \|y-y_r\|}) \quad (11)$$

r is specified as: $r = \max_i \{ (\|x-x_i\| - \|x-x_{i-1}\|) / \|x-x_i\| \}$. This setting employs the max gap in the list of distances from x to other points as the desired choice. In above setting, $\|x-x_i\|$ is the distance list of x to other points and it is sorted in the ascending order.

To observe the effect of tuning strategy, Figure 1 and Figure 2 show the results produced by classical SVC with original q and the new SVC with the tuning-flexibly q_{xv}. Clearly, in Figure 1, Data representatives generated by the traditional SVC are only located on cluster boundaries. But in Figure 2, Data representatives generated by tuning SVC are located on both cluster boundaries and the important positions within clusters. It is noticeable that on these positions sharp changes of density happen. It is consequent to arrive to the conclusion that these data representatives well describe the whole dataset, and then form a qualified sketch of dataset. That suggests nearest neighbor rule can work well after the Kernel scale is tuned adaptively.

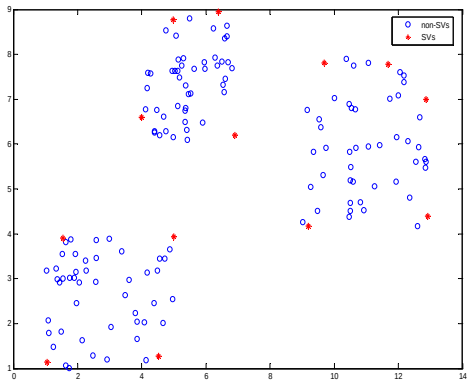

Figure 1. Data representatives produced by SVC

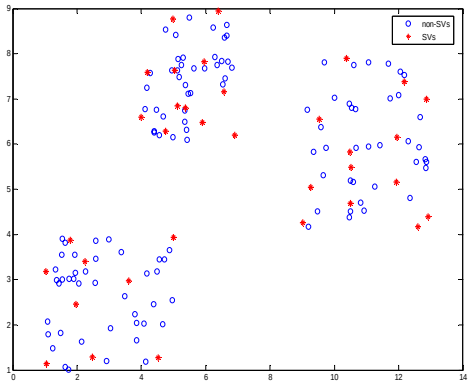

Figure 2. Data representatives produced by new SVC

V. Empirical Analysis

In experimental section, through comparing with TPL's peer algorithms, and other popular clustering algorithms, the goal is to observe the time performance, clustering performance.

A. Time Cost Analysis

Since the novel point of TPL is that it learns clusters information from Data representatives, and consequently groups other data, it is natural to ask whether this could bring the improvement in time cost. This section compares time efficiency of TPL with mentioned labeling methods, to find whether the cost consumed by selecting Data representatives is deserved. Before labeling, SVC optimization process is conducted firstly to find Data representatives. This process of TPL uses tuned scale. In other methods, if they employ scale parameters, they set their scale parameters through 15-fold cross-validation.

Datasets are taken from UCI [14]. Their some basic information is listed in Table I. Therein D is the dimensionality, N_{all} is the size of entire dataset, N is the size of experiment set, and N_{cl} is cluster number. For Letter dataset, we randomly sample 100 data points from 'A' to 'J', the top 10 categories, to form 1000-sized experimental set. Table II records one-run time consumption of each method, where the sampling number is 15.

TABLE I.
BASIC INFORMATION OF DATASETS

	D	N_{all}	N	N_{cl}
1) Iris	4	150	150	3
2) Wine	13	178	178	3
3) Vote	17	435	435	2
4) Liver	7	345	345	2
5) Monk3	8	432	432	2
6) Letter	17	20000	1000	26

TABLE II.
TIME COST COMPARISON (SECONDS)

	1)	2)	3)	4)	5)	6)
CG	83	90	205	197	181	213
SVG	69	71	150	112	127	168
PG	9	9	18	20	22	28
GD	3	19	11	29	55	30
CCL	0	0	0	0.3	0.5	1
TPL	0	0	5.2	3	3	6

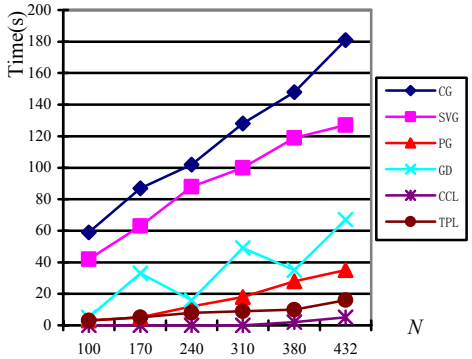

Figure 3. Time stability comparison

Besides, to observe the stability of various methods, on Monk3 dataset, different-sized subsets are randomly found, and the time cost of each method is recorded. The concrete time cost is illustrated in Figure 3.

According to seconds consumed by each method, it is obvious to find that CG consumes highest time consumption, followed by SVG and then PG. CCL takes the least time on average. TPL achieves the second least place. GD has somewhat similar level of efficiency with PG, but its behaviors are unstable. These empirical behaviors are analyzed as below theory discussion.

In the theory discussion, here, time expense of labeling process can be divided into three parts: T_M, the time used to construct the adjacent matrix; T_C, the time used to induce connected components; and T_L, the time used to label non-data-representatives. Time cost of three parts of methods is listed in Table III in details. In Table III, m represents the number of sampling points; k represents the iteration numbers for GD to converge to a SEP; N_{sv} and N_{sep} represent numbers of Data representatives and SEPs. In below experiments, PG uses DT as simulation technique. And in DT process, E is used to represent the number of edges of DT. Of course, value of E decides the efficiency of PG algorithm.

TABLE III.
TIME COST OF THREE PARTS OF LABELING METHODS

	T_M	T_C	T_L	One-run Time	Total Time
CG	$O(N^3m)$	$O(N^2)$	-	$O(N^3m)$	$L_1 \cdot O(N^3m)$
SVG	$O(N(N_{sv})^2 m)$	$O(N_{sv}^2)$	$O((N-N_{sv})N_{sv})$	$O(N(N_{sv})^2 m)$	$L_1 \cdot O(N(N_{sv})^2 m)$
PG	$O(N \log N + Em)$	$O(N)$	$O((N-N_{sv})N_{sv})$	$O(N \log N + Em)$	$L_1 \cdot O(N \log N + Em)$
GD	$O(N^2mk) + O(N_{sep}Nm)$	$O(N_{sep}^{2)})$	$O(N_{sep})$	$O(N^2mk)$	$L_1 \cdot O(N^2mk)$
CCL	$O(N_{sv}^2)$	$O(N_{sv}^2)$	$O((N-N_{sv})N_{sv})$	$O(NN_{sv})$	$L_2 \cdot O(NN_{sv})$
TPL	-	$O(N_{sv}^3)$	$O((N-N_{sv})N_{sv})$	$O(N_{sv}^3)$	$O(N_{sv}^3)$

L_1 is the fold number of cross-validation and L_2 is the length of q-list. Then, theoretically, time consumption of each method can be computed as the results of Table III.

According to Table III, for T_M, GD's cost is the highest, because GD builds the adjacent matrix among SEPs and it has to pay extra effort to look for SEPs. TPL does not take time, for it uses the sub-matrix of the complete affinity matrix directly. Except TPL, CCL spends the least time. SVG constructs the same-sized adjacent matrix as CCL, but it requires sampling operation, which leads to more time cost than CCL.

For T_C, CG is the top consumer. TPL is the second one due to its eigen decomposition operation. SVG, CCL follows TPL. CG, SVG, CCL all compute the transitive closure of adjacent matrix to obtain connected components, so their time complexity is proportion to their matrix sizes. Often N_{sep} is lower than N_{sv}, so GD achieves best efficiency among methods. PG uses DT to approximate graph, and consumes a little time.

As to T_L, CG has finished clustering task, so with zero cost. SVG, PG, CCL and TPL perform the same operation to computing pair-wise information between Data representatives and non-Data representatives, thus yielding same cost. GD takes least time by labeling data directly according to its representative SEP.

Now look at the empirical evidence of one-run time case. Clearly CG is the leading consumer, followed by SVG, GD and then PG. Their cost is controlled by m. CCL and TPL reduce time cost dramatically and CCL is more effective than TPL. But if the total time is concerned, TPL is the better tool since it is equipped with the adaptive parameterization. CCL's running is based on a q-list, from which a good setting is selected. Usually this list is of a big length, so that a desired result can be obtained. Other methods have to perform some trials for cross-validation, also incurring huge cost. For GD, its complexity is controlled by two factors, m and k, therefore its cost is more changeful than other methods.

To investigate time stability, we input Monk3 data in a batch-incremental fashion, to observe behaviors of methods in this ever-increasing dataset. The trend curves of methods' one-run time are shown in Table III. It can be seen that with N increasing, all methods see their increase of cost. CG's curve sees the fast speeding rate as N increasing. SVG, PG, TPL and CCL have relatively stable varying tendency. As mentioned before, GD does experience sharp fluctuations.

This analysis coincides with empirical results.

Then take a look at the time stability behaviors. From Figure 3, it is safe to come to the conclusion that CCL presents the highest stability among all methods. Its success depends on its complex process to find a large amount of geometric information. This rich information guarantees the fine clustering performance, and consequently guarantees the increase of time cost not to be so dramatic. TPL's behaviors follow CCL, and the difference between TPL and CCL is not so clear. Take the cluster labeling cost into account, the behaviors of TPL can be said fine, and it should be a desired choice in practice. As to other methods, their time stability drops in turn. Reasons can also be found in above analysis.

B. Clustering Performance Analysis

In this section, another dataset is involved to observe the clustering performance of some algorithms: News Group [15]. News Groups contains about 20, 000 articles divided into 20 news groups.

This paper names news groups as below.

NG1: alt.atheism;
NG2: comp.graphics;
NG3: comp.os.ms.windows.misc;
NG4: comp.sys.ibm.pc.hardware;
NG5: comp.sys.mac.hardware;
NG6: comp.windows.x;
NG7: misc.forsale;
NG8: rec.autos;
NG9: rec.motorcycles;
NG10: rec.sport.baseball;
NG11: rec.sport.hockey;
NG12: sci.crypt;
NG13: sci.electronics;
NG14: sci.med;

NG15: sci.space;
NG16: soc.religion.christian;
NG17: talk.politics.guns;
NG18: talk.politics.mideast;
NG19: talk.politics.misc;
NG20: talk.religion.misc.

Before clustering task, the usual *tf.idf* weighting schema is exploited to express documents, and words that appear too few times are deleted. Normalize each document vector to have unit Euclidean length. For empirical ease, some classes are sampled randomly to form 6 experimental sets:

a) {NG1, NG2, NG7} (200) ;
b) {NG3, NG7, NG8, NG12} (310) ;
c) {NG2, NG3, NG4, NG5} (350) ;
d) {NG8, NG9, NG10, NG11} (210) ;
e) {NG17, NG18, NG19, NG20} (170) ;
f) {NG12, NG13, NG14, NG15} (260).

In above datasets, the number in the bracket is the number of samples of each class. Clustering accuracies of methods are recorded in Table IV. Therein the sampling number is set as 15. Note that PG has no result in News Group dataset, because DT used by PG is infeasible in high-dimensional data space.

TABLE IV.
CLUSTERING ACCURACIES COMPARISON (%)

	CG	SVG	PG	GD	CCL	TPL
1)	96.6	95.7	96	96.4	97	97
2)	95.8	92.2	92.1	94.3	96.3	95.0
3)	95.1	94	93.5	95	95.8	95.3
4)	70.6	69.1	68.1	71	71.8	71.3
5)	96.7	95.2	95	96	97.0	97.3
6)	87.3	85.7	85.5	85	88.5	88.1
a)	83.9	81.7	-	84.7	85	84.7
b)	83.0	81.8	-	82	84	83.7
c)	78.8	78	-	77.7	80.3	79.5
d)	67	66.1	-	67.2	68.5	67.3
e)	68.9	65.9	-	65	70.8	70.1
f)	67.0	66	-	67.0	68	67.3

From Table IV, CCL and TPL give higher accuracy than other methods, which is due to their removing randomness. Generally speaking, CCL outperforms TPL subtly. Like Vote, Liver, Letter and some News Group subsets, CCL presents the best results and TPL follows it. CCL's advantage is attributed to its *q*-list. Through searching parameter space CCL can find a desired scale than TPL's auto-parameterized scale. Of course CCL's good performance is at the price of high cost. In some scenarios, like Iris and Monk3, TPL can achieve the optimal results, which is attributed to two reasons.

The first one lies in the first-phase clustering. The cooperation between Data representatives' geometric properties and SA technique renders SA plays all its potential and produces the clustering results as good as possible. While CCL clusters Data representatives according to the observation of overlapping status among cone-shaped regions, which lacks theoretical support and consequently leads to moderate results.

The second reason comes from the process of second phase, that is, non-Data-representative classification. TPL fulfills it based on information of both feature space and input space, but CCL only depends on information of input space, and its decision is only based on the geometric observation. Note that the auto parameterization does not always work well. In Wine, TPL does a poor job. That is caused by the fact that Wine data has 178 points but 13 dimensions makes the neighborhood information be weak, and then the q_{xv} be imprecise, and consequently, the final result is affected. In experiments, TPL can reach or get close the optimal clustering results with less cost, so it is a more welcome choice among its counterparts under the consideration of both efficiency and clustering accuracy.

CG, SVG and PG are all based on the derivation of adjacent matrix and connected components, CG presents good and stable performance thanks to its employment of complete graph. SVG is a rough version of CG, so its performance follows CG. PG is the approximated version of SVG, so its performance follows SVG. Actually CG's performance is limited by sampling number *m*. If *m* could go up to a fairly large value, CG would present the perfect result. GD produces good results in Iris, Liver and dataset a), but behaves poorly in Wine and Letter. As mentioned before, GD is an unstable approach. It is susceptive to data distribution and data dimensionality. GD only produces good results in datasets whose distribution it adapts to.

C. Compare with Popular Clustering Methods

The last section compares TPL with some popular clustering methods: K-means [16], Girolami [17], NJW [18], and NI [19].

TABLE V.
COMPARISON OF CLUSTERING ACCURACIES OF METHODS (%)

	K-means	Girolami	NJW	NI	TPL	qTPL
1)	95.8	97	97	97	97	97
2)	94	95.7	97.5	96	95	97.2
3)	94.2	95.8	97	95.3	95.8	96.3
4)	71.1	73	73.7	70.3	72.5	73
5)	96.2	97	97.3	96.4	97.1	97.5
6)	86.9	89	90.5	88.1	89	89.9
a)	79.4	82.7	85.1	82.8	84.8	85
b)	81.8	82.7	85	80.2	83	84.1
c)	75	78	81.1	73	79.5	80.3
d)	65.1	66.1	68.8	67.2	67.3	68
e)	66.7	68	71.6	67.5	70.0	70.8
f)	65.3	66.2	68.8	66.8	67	68

Girolami is a Kernel-based method. It shares the spirit of expectation-maximum process, and uses this process as its core idea. NJW is a popular version of spectrum clustering method family. It plays much importance in spectral grouping tasks. Different from above two algorithms, NI is an agglomerative clustering method. It defines the metric according to information entropy, and takes this information-based distance as clustering criteria to accumulate sub-clusters gradually. To test the effect of tuning strategy proposed in this paper, here another version of TPL is conducted: *q*TPL. *q*TPL looks for the optimal *q* through searching parameter space. Clustering accuracies of these methods are listed in Table V.

From Table V, it can be found that NJW is the best clustering tool. NJW's best performance is due to the fact that it probes spectrums of all data, thus obtains the inherent distributing directions of datasets. That assists much in detecting correct clusters. Of course NJW's best behaviors are at the price of large cost; its time complexity is $O(N^3)$. That affects its performance in many applications. As to *q*TPL, it provides the best job in Iris and Monk3. And *q*TPL follows NJW with a small gap in other datasets. That illustrates validation and performance of TPL idea. The behavior difference between *q*TPL and TPL is not so distinct. So it verifies the effect of tuning strategy that in most cases, this strategy can find proper scale parameter. For NI, Girolami and K-means, their clustering accuracy drops gradually in turn. Between Girolami and NI, the former works better than the latter. It is because, as an agglomerative approach, NI is fairly easy to be influenced by data input order. Although the information-based metric helps a lot to measure accurate distance among data, NI still cannot give stable results. K-means is a hard partition process based on Euclidean metric. It lacks adaptation to overlapping clusters and diverse distributions. So K-means presents moderate behaviors.

Generally speaking, TPL gives good results with less computation, so it is a more feasible choice in applications.

VI. A General Clustering Idea

Take a further look at the process of TPL, it is easy to summarize a general clustering framework that has following steps: 1) Extract Data representatives; 2) Cluster Data representatives; 3) Construct a classifier based on labeled Data representatives; 4) Classify other data.

Actually, this framework divides clustering task into clustering and classifying two sub-tasks. And Data representatives generated in the first step bridge the two sub-tasks.

The performance of this framework heavily depends on the quality of Data representatives. If qualified representatives that can act as the sketch of the whole dataset can be found, the framework is expected to produce good clusters. Some approaches of data reduction can do this job if they are modified specially. This paper recommends the tuning-scaled SVC, or other support-vector-based algorithms, since support vectors produced by these algorithms just reveal some information of data distribution. Like tuning-scaled SVC, its support vectors tell data contours, and it adapts to any data distribution shape. That brings much advantage in clustering scenario.

As to the classifier of the second phase, it has many choices. Those well-known classifiers are all fine options, like, SVM [20, 21], kNN [22], C4.5 [23, 24, 25], etc.

It is believed that clustering machines based on this idea will produce higher accuracy while less cost.

VII. Conclusion

This paper presents a novel Three-Phase Labeling Algorithm (TPL). TPL extracts Data representatives firstly, and then clusters them in feature space according to their geometric properties; finally the algorithm classifies non-Data representatives based on the information provided by feature space and input space. Therein support vector clustering technique is used to find Data representatives. The second phase is governed by spectrum clustering, whose theoretical support is guaranteed by Data representatives' geometry corollary. The third phase is implemented through nearest neighbor classifier that is based on a new metric; the confidence of this operation is guaranteed by the designed self-tuning strategy of Kernel scale.

Empirical evidence on real datasets demonstrate that in practical applications, TPL has the higher efficiency and competitive, even better, performance over its counterparts. Starting from TPL, a new general clustering framework can be discussed. That is, select Data representatives, and then cluster Data representatives, finally classify other data. In future work, more promising clustering approaches can be developed by implementing this idea into diverse versions.

Acknowledgment

This work is supported by the Youth National Natural Science Foundation of China under Grant No. 61105129.

References

[1] A. Ben-Hur, D.Horn, H. T. Siegelmann, "Support Vector Clustering," *Journal of Machine Learning Research*, vol. 2, pp. 125-137, 2001.

[2] J. Yang, V. Estivill-Castro, S. Chalup, "Support Vector Clustering Through Proximity Graph Modeling," *Proceedings of 9th International Conference on Neural Information Processing*, pp. 898-903, 2002.

[3] D. T. Lee, B. J. Schachter, "Two algorithms for constructing a Delaunay triangulation," *International. Journal of Parallel Programming*, vol. 9(3), pp. 219-242, 1980.

[4] M. Qi, T. T. Cao, T. S. Tan, "Computing 2D Constrained Delaunay Triangulation Using Graphics Hardware," *Technical Report, #TRB3/11, of National University of Singapore*, 2011.

[5] B. C. Wu, A. P. Tang, L. F. Wang, "A Constrained Delaunay Triangulation Algorithm Based on Incremental Points," *Applied Mechanics and Materials*, vol. 90, pp. 3277-3282, 2011.

[6] S. W. Yang, Y. Choi, C. K. Jung, "A divide-and conquer Delaunay triangulation algorithm with a vertex array and flip operations in two-dimensional space," *International Journal of Precision Engineering and Manufacturing*, vol. 12(3), pp. 435-442, 2011.

[7] O. Devillers, "Vertex removal in two-dimensional Delaunay triangulation: Speed-up by low degrees optimization," *Computational Geometry*, vol. 44(3), pp. 169–177, 2011.

[8] J. Lee, D. Lee, "An Improved Cluster Labeling Method for Support Vector Clustering," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 27(3), pp. 461-464, 2005.

[9] S.H. Lee, K.M. Daniels, "Cone Cluster Labeling for Support Vector Clustering," *Proceedings of the Sixth SIAM*

International Conference on Data Mining, pp. 484-488, 2006.
[10] A. Ng, M. Jordan, Y. Weiss, "On spectral clustering: Analysis and an algorithm," *Proceedings of the 14th Advances in Neural Information Processing Systems*, pp. 849-856, 2001.
[11] T. Zheng, X. B. Li, Y. W. Ju, "Disturbing Analysis on Spectrum Clustering," *Science in China (Series E)*, vol. 37(4), pp. 527-543, 2007.
[12] P. Ling, C. G. Zhou, "A new learning schema based on support vector for multi-classification," *NEURAL COMPUTING & APPLICATIONS*, vol. 17(2), pp. 119-127, 2008.
[13] A. Kowalczyk, B. Raskutti, "One class SVM for yeast regulation prediction," *ACM SIGKDD Explorations Newsletter*, vol. 4(2), pp. 99-100, 2002.
[14] http://www.uncc.edu/knowledgediscovery
[15] http://www.cs.cmu.edu/afs/cs/project/theo-11/www/naive-bayes.html
[16] P. S. Bradley, U. M. Fayyad. "Refining Initial Points for K-Means Clustering," *Proceeding of 15th International Conference on Machine Learning*, pp. 91-99, 1998.
[17] M. Girolami, "Mercer Kernel-Based Clustering in Feature Space," *IEEE Trans. on Neural Networks*, vol. 13(3), pp. 780-784, 2002.
[18] S. F. Ding, Z. Z. Shi, F. X. Jin, et al., "A Direct Clustering Algorithm Based on Generalized Information Distance," *Journal of Computer Research and Development*, vol. 4, pp. 674-679, 2007.
[19] L. You, S. L. Zhou, G. Gao, M. Leng, "Scalable Spectral Clustering Combined with Adjacencies Merging for Image Segmentation," *Lecture Notes in Electrical Engineering*, Vol. 121, pp.709-717, 2012.
[20] B. Schölkopf, A. J. Smola, "Learning with Kernels: Support Vector Machines," *Regularization, Optimization,* and Beyond, Cambridge, MIT Press, 2002.
[21] Y. Aflalo, A. M. Bronstein, M. M. Bronstein, R. Kimmel, "Deformable Shape Retrieval by Learning Diffusion Kernels," *Lecture Notes in Computer Science*, vol. 6667, pp. 689-700, 2012.
[22] K.N.N. Unni, R. D. Bettignies, S.-D. Seignon and J.-M. Nunzi, "Application Physics Letters," vol. 18, pp. 1823-1838, 2004.
[23] J. R. Quinlan, "C4.5: Programs for Machine Learning," Morgan-Kaufmann Publishers, 1993.
[24] P. K. Douglas, S. Harris, A. Yuille, M. S. Cohen, "Performance comparison of machine learning algorithms and number of independent components used in fMRI decoding of belief vs. disbelief," *NeuroImage*, vol. 56(2), pp. 544–553, 2011.
[25] H. W. Peng, P. J. Chiang, "Control of mechatronics systems: Ball bearing fault diagnosis using machine learning techniques," *Proceedings of Control Conference*, pp. 175-180, 2011

Ping Ling was born in Xuzhou, Jiangsu Province, China, Feb. 1979. She received her Bachelor's degree in 2000, from College of Computer Science and Technology, Xuzhou Normal University. And then she received her Master's degree and PHD from College of Computer Science and Technology, Jilin University in 2006 and 2010 respectively. She research field focuses on data mining, intelligence computing, support vector machine and support vector clustering, etc.

Xiangsheng Rong was born in Yanggu, Shandong Province, China, 1975. He received his Bachelor's degree in 1997, from Department of Logistic Command, Xuzhou Air Force College of P. L. A. And then he received his Master's degree in 2003 from Xuzhou Air Force College of P. L. A. His major research directions include the application of information technology and dynamic programming technique in military logistic command, intelligence command in combined operations of a sham battle, etc.

Xiangyang You was born in Xuzhou, Jiangsu Province, China, 1972. He received his Bachelor's degree in 1994, from College of Computer Science and Technology, Harbin Institute of Technology. Now his research directions are operational research in military logistic command, intelligent computing application in logistic command, etc.

Ming Xu was in Suqian, Jiangsu Province, China, 1968. He received his Bachelor's degree in 1994, from Department of Logistic Command, Xuzhou Air Force College of P. L. A. And then he received his Master's degree in 2005 from Xuzhou Air Force College of P. L. A. His research fields are centered in data integration, data mining, semantic network, etc.