Self-Adaptive Multiprototype-Based Competitive Learning Approach: A *k*-Means-Type Algorithm for Imbalanced Data Clustering

Yang Lu, *Student Member, IEEE*, Yiu-Ming Cheung, *Fellow, IEEE*, and Yuan Yan Tang, *Life Fellow, IEEE*

***Abstract*—Class imbalance problem has been extensively studied in the recent years, but imbalanced data clustering in unsupervised environment, that is, the number of samples among clusters is imbalanced, has yet to be well studied. This paper, therefore, studies the imbalanced data clustering problem within the framework of *k*-means-type competitive learning. We introduce a new method called self-adaptive multiprototype-based competitive learning (SMCL) for imbalanced clusters. It uses multiple subclusters to represent each cluster with an automatic adjustment of the number of subclusters. Then, the subclusters are merged into the final clusters based on a novel separation measure. We also propose a new internal clustering validation measure to determine the number of final clusters during the merging process for imbalanced clusters. The advantages of SMCL are threefold: 1) it inherits the advantages of competitive learning and meanwhile is applicable to the imbalanced data clustering; 2) the self-adaptive multiprototype mechanism uses a proper number of subclusters to represent each cluster with any arbitrary shape; and 3) it automatically determines the number of clusters for imbalanced clusters. SMCL is compared with the existing counterparts for imbalanced clustering on the synthetic and real datasets. The experimental results show the efficacy of SMCL for imbalanced clusters.**

***Index Terms*—Class imbalance learning, competitive learning, data clustering, internal validation measure, *k*-means-type algorithm, multiprototype clustering.**

I. Introduction

THE IMBALANCED data means that the number of data points in one class is significantly more than the number of data points in another class. Thus far, the majority of studies on imbalanced data learning in the literature are conducted in the manner of supervised learning, for example, binary classification [1], [2]. However, the label information is usually unavailable in many real-world applications, for example, bioinformatics [3], medical imaging [4], marketing research [5], crime analysis [6], and so on. For example, in social network analysis [7], user grouping can be utilized for quick searching of special groups with abnormal behavioral intention. This task is a clustering problem on imbalanced data because the labels of abnormal behaviors are difficult to be previously defined and the size of the special groups are generally much smaller than the one of the normal groups. For the imbalanced clustering problem, majority clusters refer to the clusters that contain much more data points than the minority clusters. For the problem of classification on imbalanced data, the label information is known so that the sampling-based and cost-sensitive methods [1] can be adopted to alleviate the performance deterioration caused by imbalance. Clustering on imbalanced data is a totally different and more challenging problem. Its difficulty is basically on two aspects: 1) the imbalance status, that is, the number of minority clusters and the imbalance ratio, is unknown and 2) it is more difficult to determine the number of clusters for imbalanced data. A common clustering result on imbalanced clusters is that the minority cluster is either merged into the majority cluster as one cluster or treated as noises and outliers. As far as we know, imbalanced data clustering has yet to be well studied in the literature.

In terms of clustering, *k*-means is one of the most well-known methods in the framework of competitive learning [8]. Thus far, it, as well as its variants, has been widely used in both academia and industry [9]–[12]. The adaptive version of *k*-means is to update the seed points, that is, those data points learnable toward the center of clusters in the input space, after each data point comes. That is, at each time step, the winner, that is, the winning seed point, is selected by choosing the seed point with the minimum distance to the coming data point and is updated by moving toward it. However, despite the success of the adaptive *k*-means in its application domain, it suffers from the problem that a seed point has no chance to win if it is initialized far away from the region of the input data. It is called the dead-unit problem [13]. Frequency sensitive competitive learning (FSCL), as a *k*-means-type algorithm, improves adaptive *k*-means by adding a frequency weight to the distance so that the seed point that wins less frequently in the past will have more chance to win in the future [13]. Rival penalized competitive learning (RPCL) further improves FSCL by

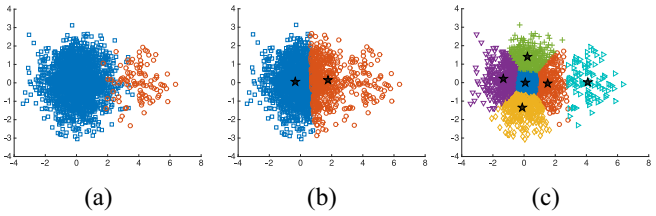
Manuscript received January 21, 2019; revised March 28, 2019; accepted May 7, 2019. Date of publication May 29, 2019; date of current version February 17, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 61672444 and Grant 61272366, in part by the Faculty Research Grant of Hong Kong Baptist University (HKBU) under Project FRG2/17-18/082, in part by KTO Grant of HKBU under Project MPCF-004-2017/18, and in part by SZSTI under Grant JCYJ20160531194006833. This paper was recommended by Associate Editor N. Zhang. *(Corresponding author: Yiu-Ming Cheung.)*

Y. Lu and Y.-M. Cheung are with the Department of Computer Science, Hong Kong Baptist University, Hong Kong (e-mail: yanglu@comp.hkbu.edu.hk; ymc@comp.hkbu.edu.hk).

Y. Y. Tang is with the Department of Computer and Information Science, Faculty of Science and Technology, University of Macau, Macau 999078, China (e-mail: yytang@umac.mo).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2019.2916196



(a) (b) (c)

Fig. 1. Example of two imbalanced clusters with size clustered [2000, 100]. (a) Ground truth, (b) clustering result of adaptive *k*-means with two seed points, and (c) clustering result of adaptive *k*-means with six seed points.

introducing the heuristic rival penalization mechanism to make the rival seed point move to the opposite direction along the data point [14]. Consequently, the seed points sharing the same cluster will be pushed away until that there is only one seed point left for each cluster. Thus, the number of clusters can be determined by eliminating the seed points that are pushed away. Due to the simplicity and efficiency of RPCL, a number of improvements and variations of RPCL have been further proposed [15]–[19]. In particular, paper [20] has presented a maximum weighted likelihood (MWL) learning framework to describe such rival penalization mechanism from a theoretical viewpoint.

Besides these efforts, recent studies have also shown that *k*-means tends to produce balanced clustering result [21]. This preference seriously deteriorates the performance of *k*-means when the clusters are imbalanced. The consequence is that the seed point of the minority cluster will gradually move to territory of the majority cluster. Finally, nearly half of the data points in the majority cluster will be classified into the minority cluster. This is called uniform effect [21]. As an online version of original *k*-means, adaptive *k*-means also suffers from uniform effect as shown in Fig. 1(b). Nevertheless, uniform effect does not always happen to adaptive *k*-means when the data is imbalanced. At each iteration, the only information that adaptive *k*-means keeps is the current position of the seed points and the cluster assignment is not related to the imbalance status of the past data points. Therefore, if the clusters are well separated, data imbalance will not affect the clustering performance. By contrast, RPCL, as well as FSCL, records the past winning frequency which reflects the imbalance status of the clusters, in addition to the positions of the seed point. Subsequently, even the clusters are well separated, the seed point of the minority cluster has more chance to win the data points in the majority cluster due to its low winning frequency in the past. Therefore, uniform effect becomes more serious if frequency weights are used. Besides, when the data is imbalanced, RPCL and its variants fail to automatically select the number of clusters by rival penalization. Instead, either the majority cluster is shared by several seed points or the minority cluster is treated as a part of the majority cluster by pushing away the seed point of the minority cluster.

To deal with class imbalance, the most common approaches are sampling on the data or assigning different weights to different classes. However, they are designed for supervised learning problems only and not applicable to clustering on imbalanced clusters. In fact, no matter which kinds of methods are adopted for class imbalance problem, acquiring the label is prerequisite. In the literature, focusing on the imbalanced data clustering problem, Liang *et al.* [22] has proposed a multiprototype clustering algorithm for imbalanced data clusters. It is based on the fuzzy *k*-means with a multiprototype mechanism to deal with uniform effect. Given the number of prototypes, the algorithm can locate the position of prototypes properly to the imbalanced clusters, such that the majority clusters receive more prototypes and the minority cluster receives less. After that the algorithm merges the subclusters represented by the prototypes to form the final clusters. However, one of the major drawbacks of this algorithm is that the number of prototypes is a user-defined parameter. This algorithm fails if the number of clusters of the imbalanced data is more than the predefined number of prototypes. Besides, the number of clusters in the merging process is selected manually.

To solve the aforementioned problems, we, therefore, propose self-adaptive multiprototype-based competitive learning (SMCL) to keep the advantages of the competitive learning methods and avoid problems caused by imbalanced data. To tackle the uniform effect, SMCL adopts the multiprototype clustering mechanism. Each cluster is represented by one or more subclusters. Thus, the majority cluster contains more subclusters and the minority cluster contains less subclusters. By applying competitive learning among subclusters, the number of samples in each subclusters is relatively balanced, and uniform effect does not happen to the subclusters, as shown in Fig. 1(c). Since a proper number of subclusters depends on the data structure, in order to produce a proper number of subclusters, SMCL adopts a self-adaptive way that incrementally adds new seed points until one seed point is driven away by the rival penalization mechanism. At that moment, the clusters are well represented by enough number of seed points. Then, SMCL utilizes 1-d binary Gaussian mixture probability density function to measure the separability between each subcluster and gradually merge the subclusters with low separation measure. During the merging stage, a new internal clustering validation measurement is used to evaluate the clustering quality and determine the number of clusters. Experiments on synthetic and real benchmark datasets have shown that SMCL produces better clustering results for imbalanced data. The contribution of this paper is summarized as follows.

1) This paper systematically studies the performance of *k*-means-type competitive learning methods on imbalanced data clustering.
2) A novel *k*-means-type method SMCL is proposed for imbalanced data clustering.
3) A new separation measure based on 1-d binary Gaussian probability density function is proposed for subcluster merging.
4) An internal clustering validation measure to select the proper number of clusters is proposed for imbalanced data.

The remainder of this paper is organized as follows. Section II presents an overview of the related work and discusses the problems when clustering on imbalanced data. The proposed SMCL is described in detail in Section III.

Section IV shows the experimental results with some discussions. Finally, a conclusion is drawn in Section V.

II. Overview of Related Work

This section briefly reviews the k-means-type competitive learning methods, the imbalance classification methods, the nonlinear clustering methods, and the imbalance clustering methods, respectively.

A. k-Means-Type Competitive Learning

Adaptive k-means is the simplest competitive learning method [8], [23]. Suppose the data point coming at time t is $\mathbf{x}_t$ and there are K seed points: $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_K$ to represent the centroids of K clusters. Accordingly, the values of $\mathbf{m}_j$'s at time t are denoted as: $\mathbf{m}_1(t), \ldots, \mathbf{m}_K(t)$. For simplicity, we will hereinafter utilize $\mathbf{m}_j$'s and $\mathbf{m}_j(t)$'s interchangeably without further distinction. When $\mathbf{x}_t$ arrives, the winner seed point is selected by the following indicator function:

$$I_{j,\mathbf{x}_t} = \begin{cases} 1 & \text{if } j = c = \arg\min_{1\le i\le K}\left(\|\mathbf{m}_i - \mathbf{x}_t\|^2\right) \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where the cth seed point, that is, the winner, has the minimum distance to $\mathbf{x}_t$. Then, the winner seed point is updated by moving toward $\mathbf{x}_t$ controlled by a small learning rate α_c

$$\mathbf{m}_j(t+1) = \begin{cases} \mathbf{m}_j(t) + \alpha_c\left(\mathbf{x}_t - \mathbf{m}_j(t)\right) & \text{if } I_{j,\mathbf{x}_t} = 1 \\ \mathbf{m}_j(t) & \text{otherwise} \end{cases} \tag{2}$$

for $j = 1, \ldots, K$. To overcome the dead-unit problem that some seed points may never win, FSCL [13] uses the frequency weighted distance to determine the winner indicator [13]

$$I_{j,\mathbf{x}_t} = \begin{cases} 1 & \text{if } j = c = \arg\min_{1\le i\le K}\left(\gamma_i\|\mathbf{m}_i - \mathbf{x}_t\|^2\right) \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where $\gamma_i = n_i / \sum_{l=1}^{K} n_l$ is the frequency weight and n_i is the cumulative number of winning times of $\mathbf{m}_i$. The seed point updating remains the same as (2). By adopting this frequency weight, the seed point that barely wins in the past gains more chance to win in the future. Furthermore, RPCL [14] improves FSCL to make the number of clusters can be determined automatically by utilizing the rival penalization mechanism. In addition to updating the winner seed point, RPCL updates the rival seed point toward the opposite direction. The rival seed point is selected as the second closest one to $\mathbf{x}_t$

$$I_{j,\mathbf{x}_t} = \begin{cases} 1 & \text{if } j = c = \arg\min_{1\le l\le K}\left(\gamma_l\|\mathbf{m}_l - \mathbf{x}_t\|^2\right) \\ -1 & \text{if } j = r = \arg\min_{1\le l\le K, l\ne c}\left(\gamma_l\|\mathbf{m}_l - \mathbf{x}_t\|^2\right) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

and the seed points are updated by

$$\mathbf{m}_j(t+1) = \begin{cases} \mathbf{m}_j(t) + \alpha_c\left(\mathbf{x}_t - \mathbf{m}_j(t)\right) & \text{if } I_{j,\mathbf{x}_t} = 1 \\ \mathbf{m}_j(t) - \alpha_r\left(\mathbf{x}_t - \mathbf{m}_j(t)\right) & \text{if } I_{j,\mathbf{x}_t} = -1 \\ \mathbf{m}_j(t) & \text{otherwise} \end{cases} \tag{5}$$

for $j = 1, \ldots, K$. α_r is the delearning rate for the rivals, which is generally smaller than the learning rate α_c. By driving the rival seed point away, each cluster will not be shared by two or more seed points. Therefore, the number of clusters can be determined automatically by counting the remaining seed points [14]. RPCCL [16] further improves RPCL by making the rival penalization self-adaptable. According to the positions of the incoming data point, winner seed point, and rival seed point, RPCCL determines

$$\alpha_r = \alpha_c \frac{\min(\|\mathbf{m}_r - \mathbf{m}_c\|, \|\mathbf{x}_t - \mathbf{m}_c\|)}{\|\mathbf{m}_r - \mathbf{m}_c\|}. \tag{6}$$

Thus, RPCCL only needs one parameter, that is, the learning rate α_c. The value of rival penalization is reduced if $\mathbf{x}_t$ is closer to $\mathbf{m}_c$ than $\mathbf{m}_r$. It overcomes the drawback of RPCL that an appropriate value of α_r is hard to be chosen. Further, rival penalized expectation–maximization (RPEM) makes the clustering components in a density mixture compete with each other, and the rivals intrinsically penalized with a dynamic control during the learning [20]. Thus, the number of clustering components, that is, the number of clusters, can be determined with the redundant densities gradually faded out automatically from the density mixture. Ma and Wang [17] used a cost-function approach to solve the convergence problem of RPCL. Based on the theoretical analysis, they propose distance-sensitive RPCL (DSRPCL). It aims to minimize a specifically designed cost function for RPCL to make it theoretically sound. Competitive repetition-suppression (CoRe) [18] is inspired by biological phenomenon. It improves RPCL by allowing multiple winners existing in each clustering iteration. It uses a gradient descent strategy to update the positions of the seed point and the spread in terms of a Gaussian function. For the datasets that are not linearly separable, stochastic competitive learning (SCL) [24] and graph-based multiprototype competitive learning (GMPCL) [25] utilize kNN to construct the neighborhood graph before carrying on competitive learning. SCL is a stochastic competitive learning model. The seed points try to occupy the nodes in the network by random walking and defending their territory from rival seed points at the same time. Finally, the dominance of each node is determined by the visiting frequency of the seed points. GMPCL first selects a portion of data points as the core points, according to their connectivity in the graph, to produce coarse clusters. Then, it applies affinity propagation and competitive learning to refine the coarse clusters on all data points. Moreover, the competitive learning is integrated with cooperative learning [26], [27]. The winner seed point is assigned a confidence coefficient based on its past winning frequency. The winner seed point with high confidence coefficient will cooperate more and penalize less the nearby seed points. Finally, the seed points in the same cluster will merge together so that the number of clusters is selected. Its kernel version can also handle nonlinearly separable data. However, none of the above-mentioned methods consider the situation of imbalanced data clustering.

B. Nonlinear Clustering

Some advanced k-means-type clustering methods [22], [25], [28] can produce nonspherical clusters by merging subclusters. In contrast, the nonlinear clustering methods can directly generate clusters with arbitrary

shapes. Kernel-based clustering methods map the data into a high-dimensional space, that is, the kernel space, where the projected points can be linearly clustered [29]. Support vector clustering (SVC) brings the framework of SVM [30]. In the mapped high-dimensional space, SVC looks for the smallest sphere that encloses the data points for each cluster. As an improvement to tackle the parameter selection problem of SVC, position regularized SVC (PSVC) [31] weighs each data point by its position as a regularizer, based on the distance between each data point and the mean of all points in the kernel space. Another family of nonlinear clustering is the spectral clustering methods [32], which construct a weighted graph between all data points as their similarity and then use eigendecomposition to obtain the clustering result. The graph can also be used for subspace clustering with low-rank representation [33]–[35]. In addition to these two categories, density peak clustering (DPC) [36] assumes that cluster centers with density peaks are surrounded by neighbors with lower local density and that they are at a relatively large distance from any points with a higher local density. DPC is able to efficiently produce nonspherical clusters and automatically exclude the outliers. Recently, CHKNN [37] utilizes hybrid *k*-nearest neighbor graph to represent the nonlinear data with a new internal validity index. In CCMS [38], a new metric called attraction degree is introduced to describe the data density difference. Then, a rule is designed upon the density difference to move the data points and construct the clusters. Although nonlinear clustering methods are well designed to deal with clusters with an arbitrary shape, they are unaware of the imbalanced clusters, thus easily grouping the minority cluster into other clusters or treating it as an outlier.

C. Imbalance Classification

Most of the existing work of imbalanced data learning is within the framework of supervised learning. They can be basically categorized into three groups [39]. The first group is on data level. The methods in this group aim to balance the data before training. The most well-known methods in this group are synthetic minority over-sampling technique (SMOTE) [40] and its variances. SMOTE synthesizes new samples to the minority class by interpolating the minority class samples with their neighbors. In addition to oversampling, undersampling techniques have also been used in data preprocessing. For example, Batista *et al.* [41] adopted Tomek links to clean the overlapping area between classes so that the classification boundary becomes clear after introducing synthetic samples. Sampling methods are usually integrated with ensemble methods to increase the diversity and avoid information loss [42]. The second group is on algorithm level. They modify the existing learning methods by adapting them to the imbalanced data. For example, Hong *et al.* [43] modified the kernel classifiers by orthogonal forward selection to optimize the model generalization for imbalanced datasets. Raskutti and Kowalczyk [44] have proposed to use one-class SVM that is only trained with the minority class samples. Therefore, the classification hyperplane is not influenced by minimizing the margin error of the majority class. The last group is related to the framework of cost-sensitive learning [45]. They assign different costs to the samples in difference classes. The idea of cost-sensitive can also be applied to many existing algorithms to adapt them to deal with class imbalance, such as decision tree [46] and SVM [47].

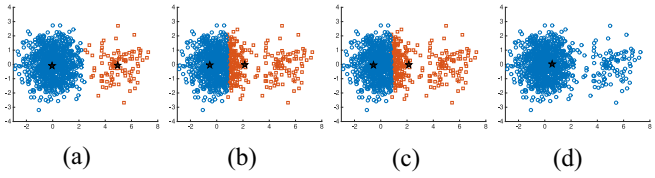


Fig. 2. Example of two well-separated imbalanced clusters conducted by (a) adaptive *k*-means with $\alpha_c = 0.01$, (b) FSCL with $\alpha_c = 0.01$, (c) RPCL with $\alpha_c = 0.01$ and $\alpha_r = 0.0001$, and (d) RPCCL with $\alpha_c = 0.0001$. The initial positions of the seed points are set at the true cluster centers.

D. Imbalance Clustering

Few papers have studied the clustering problem on imbalanced data. In the literature, Xiong *et al.* [21] have defined the uniform effect by investigating the performance of *k*-means on imbalanced data. They proposed to use the evaluation metric coefficient of variance (CV) to evaluate the performance of clustering methods on imbalanced dataset. To make *k*-means resist from uniform effect, Liang *et al.* have proposed an algorithm with three stages to cluster imbalanced data [22] based on fuzzy *k*-means clustering [48]. However, the number of prototypes is predefined that cannot adapt to the data, and the selection of the number of clusters needs to inspect the knee point manually from the plot. An extension of [22] is [49] which adopts ensemble clustering to tackle the problem that fuzzy *k*-means cannot recognize arbitrarily shaped clusters. However, the imbalanced clusters are not considered. Multiexemplar merging clustering (MEMC) [50] improves multiexemplar affinity propagation [28] for imbalanced data. MEMC merges the exemplars in the same cluster by measuring the overlapping degree between each cluster. However, the number of clusters must be preassigned to stop the merging process. In [51], the probability models are selected to suit the imbalanced clusters and the parameters of the models are estimated by evolutionary computation. Yang and Jiang [52] transplanted the ensemble algorithms from supervised class imbalanced problem to imbalanced data clustering. They combined bagging and boosting to produce consolidated clustering result by a novel consensus function. Furthermore, Aksoylar *et al.* [53] adopted spectral clustering on imbalanced data. They proposed to deal with imbalanced data by minimizing cut partitions which yield a set of parameter-dependent cuts according to the different levels of imbalance degree. In summary, clustering on imbalanced data is a challenging problem that has been receiving increasing interests in recent years, but has yet to be well studied, as far as we know.

III. Proposed Method

When the data is imbalanced, the competitive learning methods encounter two major problems. The first is that the

frequency weight makes the uniform effect more serious. Fig. 2 shows an example when the two clusters are imbalanced but well separated. Under this situation, adaptive k-means can avoid uniform effect and produces good clustering result if the initial seed points are set at the positions of the true cluster centers, as shown in Fig. 2(a). However, the frequency weight of FSCL and RPCL makes the seed point in the minority cluster have a larger chance to win the data point with nearly the same distance to both seed points, as shown in Fig. 2(b) and (c). As the frequency weight is multiplied to the distance as shown in (3) and (4), it makes the seed point in the minority cluster move toward the majority cluster, whose phenomenon is, namely the uniform effect. The second problem is that the selection of number of clusters by RPCL fails when the data clusters are imbalanced. In spite of RPCL or RPCCL, the original design of their model selection mechanism is based on the assumption that the clusters are balanced. Under this assumption, the rival penalization between clusters are balanced to keep the correct number of seed points left. For example, the rival penalization of RPCCL is usually larger than the one of RPCL because it equals to the learning rate if $\|\mathbf{m}_r - \mathbf{m}_c\| < \|\mathbf{x}_t - \mathbf{m}_c\|$. This strong rival penalization works well when the clusters are balanced because the rival penalization counteracts between clusters. However for imbalanced clusters, the penalization from the majority cluster is too strong to keep the seed point staying in the region of the minority cluster. The result of RPCCL shown in Fig. 2(d) is even worse. All of the samples are grouped into one cluster due to the class imbalanced number of samples in the clusters.

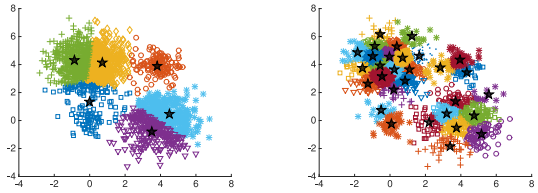
To solve the aforementioned problems, the proposed method SMCL consists of two stages.

1) In the first stage, the subalgorithm prototype number selection (PNS) automatically selects a proper number of seed points to represent the data by the multiprototype clustering mechanism. Each cluster is represented by one or more subclusters by competitive learning among subclusters. PNS starts from a few seed points and gradually increases their number in a self-adaptive manner.
2) In the second stage, the subalgorithm subcluster grouping with model selection (SGMS) gradually merges the subclusters produced by PNS and the final number of clusters is determined during this process. We propose a new 1-d binary Gaussian mixture probability density function to measure the global separability between subclusters.

The details of these two stages are described in the following sections.

A. Selection of Number of Prototypes

To solve uniform effect, one solution is to use the idea of multiprototype [22], [54], [55]. Each cluster can be represented by more than one seed point (also called prototype interchangeably) and made up by multiple subclusters. For imbalanced data, the majority cluster tends to have more seed points and the minority cluster has less. Therefore, the number of data points in each subcluster is relatively balanced and the uniform effect does not happen among the subclusters as shown in Fig. 1(c). However, a key problem related to multiprototype is that how many prototypes are needed to well represent the entire dataset. The proper number of subclusters should depend on the data structure. The existing multiprototype methods (see [22], [54], [55]) simply use a predefined number of prototypes. The problem of this strategy is that if the given number of prototypes is too small as shown in Fig. 3(a), uniform effect still happens between the minority subcluster and the majority subcluster. Conversely, if the given number of prototypes is too large as shown in Fig. 3(b), the subclusters may be located in the overlapping area between clusters, and noises and outliers would be treated as subclusters as well. Moreover, if the number of prototypes is less than the number of clusters, the clustering result will never be correct. Therefore, we propose PNS to determine the number of seed points in a self-adaptive manner. The seed points are gradually added by the algorithm until one seed point is driven away by the rival penalization mechanism.



(a) (b)

Fig. 3. Example of four imbalanced clusters with size [61, 1212, 606, 121]. The clustering result is generated by adaptive k-means with (a) 6 seed points and (b) 30 seed points.

Specifically, given the initial number of seed points $K = K_0$, a modified RPCCL is carried on. That is, the winner seed point is selected by

$$I_{j,\mathbf{x}_t} = \begin{cases} 1 & \text{if } j = \arg\min_{1\le i\le K}\left(\|\mathbf{m}_i - \mathbf{x}_t\|^2\right) \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

The frequency term γ_j is removed here to reduce the uniform effect. In terms of the dead unit problem, it can be alternatively solved by randomly selecting the data points as the initial seed points [17]. PNS stops when one seed point is driven away, which is supposed to be one of the seed points in the majority cluster. The reason is that if the seed point in the minority cluster is driven away, another seed point from the majority cluster will move toward the minority cluster. That seed point may stay at the position between the majority and minority cluster, that is, uniform effect happens again. The rival penalization term (6) in RPCCL is not able to distinguish these cases. Therefore, we propose to utilize the rival penalization coefficient β_j to differentiate the rival penalization of the within-cluster subclusters and between-cluster subclusters. First, all seed points except the winner are marked as the rivals. Different rival penalization values are then assigned based on the position of the winner, the rivals, and $\mathbf{x}_t$

$$\mathbf{m}_j(t+1) = \begin{cases} \mathbf{m}_j(t) + \alpha_c\left(\mathbf{x}_t - \mathbf{m}_j(t)\right) & \text{if } I_{j,\mathbf{x}_t} = 1 \\ \mathbf{m}_j(t) - \eta\beta_j\alpha_c\left(\mathbf{x}_t - \mathbf{m}_j(t)\right) & \text{otherwise} \end{cases} \tag{8}$$

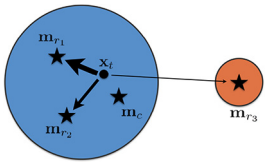


Fig. 4. Example of two imbalanced classes with four seed points. $\mathbf{m}_c$ is the winner of $\mathbf{x}_t$. $\mathbf{m}_{r_1}$, $\mathbf{m}_{r_2}$, and $\mathbf{m}_{r_3}$ are three rivals. The majority cluster (in blue) is shared by three seed points and the minority cluster (in red) is represented by one seed point. The thickness of the arrow indicates the value of β_j.

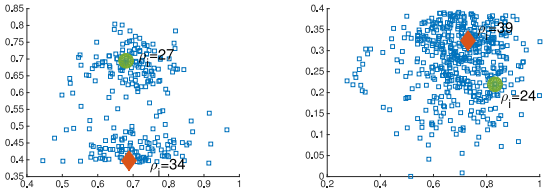


Fig. 5. Two subclusters with (a) density gap and (b) no obvious density gap. The red diamond point is the density peak with the highest local density and the green circle point is the density peak that it has the largest distance to the point with higher local density. ρ_i is the local density.

where

$$\beta_j = \exp\left(-\frac{\|\mathbf{m}_j - \mathbf{x}_t\|^2 - \|\mathbf{m}_c - \mathbf{x}_t\|^2}{\|\mathbf{m}_c - \mathbf{m}_j\|^2}\right) \tag{9}$$

and $\mathbf{m}_c$ is the winner seed point. β_j makes the rival penalization large if $\mathbf{x}_t$ is close to the middle line between $\mathbf{m}_j$ and $\mathbf{m}_c$ because the points are more dense if they are between two seed points in the same cluster. The points close to the gap between two different clusters are less frequently appeared so that the rival penalization between clusters is smaller. Thus, the within-cluster rival penalization will be larger than the between-cluster rival penalization. When there are enough number of seed points added, the first priority is to drive one of the seed points in the majority cluster away. As shown in Fig. 4, $\mathbf{m}_{r_1}$ gets larger β_j than $\mathbf{m}_{r_2}$ and $\mathbf{m}_{r_3}$ because $\mathbf{x}_t$ is close to the middle line between $\mathbf{m}_{r_1}$ and $\mathbf{m}_c$. The additional term η is used to decrease the magnitude of rival penalization. There are two reasons behind. The first is that the rival penalization is applied on all seed points except the winner. Each seed point gets more times of penalization than RPCCL, in which only the seed point with the second minimum distance to $\mathbf{x}_t$ get penalized. Another reason is to make PNS not stop quickly such that each cluster has yet to be well represented.

If the modified RPCCL converges and there is no seed point driven away, one new seed point is added. The position of the new seed point is selected by duplicating one of the existing seed points. The selection criterion is based on two factors. The first is the number of winning times n_j. The larger subcluster that wins most frequently is split first. The second is the maximal density gap δ_j. If a subcluster consists of two or more density peaks, it is probably made up of data points from two or more clusters. To calculate δ_j, we first calculate the local density ρ_i^j for each cluster member $\mathbf{x}_i$ in the jth subcluster $\mathcal{C}_j$ by ϵ-ball graph [36]

$$\rho_i^j = \sum_{i' \in \mathcal{C}_j} [\![d(\mathbf{x}_i, \mathbf{x}_{i'}) < \epsilon]\!] \tag{10}$$

where $[\![\cdot]\!]$ is the indicator function which returns 1 if the statement is true and 0 otherwise, and $d(\cdot, \cdot)$ is the Euclidean distance between two points. The suggested value of ϵ is by making the averaged number of neighbors equal to 2% of the total number of data points [36]. The density gap of $\mathcal{C}_j$ is calculated by

$$\delta_j = \max_{i \in \mathcal{C}_j} \min_{i' \in \mathcal{C}_j : \rho_{i'} > \rho_i} \frac{d(\mathbf{x}_i, \mathbf{x}_{i'})}{\bar{d}_j}. \tag{11}$$

The minimization taken in (11) calculates the minimum distance between the member $\mathbf{x}_i$ and any other member with higher local density. Thus, if there are more than two density peaks far from each other in the same subcluster, the peak with the second highest local density gains large value from the minimization. δ_j is selected as the maximum of such minimum distances among all members. $\bar{d}_j$ is the averaged pairwise distance of the all cluster members in the $\mathcal{C}_j$. The pairwise distance in $\mathcal{C}_j$ is divided by its $\bar{d}_j$ to make it comparable with other subclusters. Fig. 5 shows the positions of the density peaks. The maximal density gap δ_j is 1.41 for Fig. 5(a) and $= 0.72$ for Fig. 5(b). Thus, the case in Fig. 5(a) has higher priority to be selected as the subcluster to add new seed point. In summary, the new subcluster to be split by the new seed point is selected by

$$j^* = \arg\max_{j=1,\ldots,K} n_j \delta_j. \tag{12}$$

As the number of seed points K is increasing, the averaged number of winning times N/K for each seed point is decreasing. Thus, the moving magnitude of the seed points is decreasing as more seed points are added. To avoid the algorithm terminating due to this reason, the learning rate α_c is amplified by the number of seed points. Equation (8) is updated by

$$\mathbf{m}_j(t+1) = \begin{cases} \mathbf{m}_j(t) + K\alpha_c\big(\mathbf{x}_t - \mathbf{m}_j(t)\big) & \text{if } I_{j,\mathbf{x}_t} = 1 \\ \mathbf{m}_j(t) - K\eta\beta_j\alpha_c\big(\mathbf{x}_t - \mathbf{m}_j(t)\big) & \text{otherwise.} \end{cases} \tag{13}$$

Thus, the learning rate is proportional to the number of subclusters K to counteract the influence from the decreasing number of winning times.

The algorithm PNS is summarized in Algorithm 1. The winner seed point selection and updating is shown in line 7. The rival seed point updating is shown in line 8. After a single epoch that iterates all N data points, lines 11–27 show how to add new seed points. The convergence condition is shown in line 11. Traditionally, the sum of the distance between the new and old position of all seed points are calculated. However, if the number of seed points is increasing, this sum will be larger as K increases. Therefore, we take the maximal to determine if the seed points are converging. In lines 13–17, if the winner frequency of a seed point is less than θ, it is treated as the one that is driven away by rival penalization. Those seed points are deleted and the algorithm terminates as shown in lines 19–21. If no seed point is driven away, a new seed point

Algorithm 1 PNS

Input: $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, α_c, η, K_0, E, θ, ξ.

1: Initialize the seed point set $\mathcal{M}$ by randomly selecting K_0 data points from $\mathcal{D}$;
2: Initialize termination flag: $flag \leftarrow 0$;
3: $\mathbf{m}_j^{old} \leftarrow \mathbf{0}$, $j = 1, \ldots, K_0$;
4: $K = K_0$;
5: **for** $e \leftarrow 1$ to E **do**
6: $n_j \leftarrow 0$, $j = 1, \ldots, K$;
7: **for** $t \leftarrow 1$ to N **do**
8: Update the winner seed point: $c \leftarrow \arg\min_j(\|\mathbf{m}_j - \mathbf{x}_t\|^2)$; $\mathbf{m}_c \leftarrow \mathbf{m}_c + K\alpha_c(\mathbf{x}_t - \mathbf{m}_c)$; $n_c \leftarrow n_c + 1$;
9: Update each rival seed point: $\mathbf{m}_j \leftarrow \mathbf{m}_j - K\eta\beta_j\alpha_c(\mathbf{x}_t - \mathbf{m}_j)$, $\forall j \neq c$;
10: **end for**
11: **if** $\max_j(\|\mathbf{m}_j - \mathbf{m}_j^{old}\|^2) < \xi$ **then**
12: **for** $j \leftarrow 1$ to K **do**
13: **if** $n_j < \theta$ **then**
14: Delete $\mathbf{m}_j$, n_j;
15: $K \leftarrow K - 1$;
16: $flag \leftarrow 1$;
17: **end if**
18: **end for**
19: **if** $flag = 1$ **then**
20: Return $\mathcal{M}$;
21: **else**
22: Select the subcluster to add new seed point: $j^* \leftarrow \arg\max_{j=1,\ldots,K} n_j\delta_j$;
23: $K \leftarrow K + 1$;
24: $\mathbf{m}_K \leftarrow \mathbf{m}_{j^*}$;
25: **end if**
26: $\mathbf{m}_j^{old} \leftarrow \mathbf{m}_j$, $j = 1, \ldots, K$;
27: **end if**
28: **end for**
29: $\hat{K} = K$;

Output: $\hat{K}$, seed point set $\mathcal{M} = \{\mathbf{m}_1, \ldots, \mathbf{m}_{\hat{K}}\}$

is added to the seed point set $\mathcal{M}$ as shown in lines 22–24. Finally, the old position of seed points are recorded by $\mathbf{m}_j^{\text{old}}$. When the algorithm finishes, the final number of seed points $\hat{K}$ and the seed point set $\mathcal{M}$ are produced. The computational cost of PNS algorithm is $O(E^*\hat{K}N + N^2/\hat{K})$, where E^* is the number of epochs used before termination.

B. Subcluster Grouping With Model Selection

Once PNS terminates when at least one seed points are driven away, the number of remaining seed points $\hat{K}$ is used to represent the data, namely, $\hat{K}$ subclusters. We propose the second subalgorithm SGMS to merge the subclusters and select the number of clusters at the same time. The objective of SGMS is to merge the subclusters in the same cluster first. At each iteration, SGMS merge one subcluster with another subcluster to form a new subcluster. The merging process produces $\hat{K}-1$ intermediate clustering results. The best clustering result is then selected from them by a new internal validation clustering measure.

The subclusters with low separation measure should be merged first because they are probably in the same cluster. There are some existing methods to measure the separation between subclusters. In [54], the data points are projected and represented by histogram and bins. The measure is then calculated by checking the number of samples in the bins. One of the drawbacks of using this method is that the number of used bins influences the measure result. In [22], the separation measure is calculated by the overlapping degree between two subclusters. However, it fails if the low density region between two subclusters is not close to the middle line of two centers. Therefore, to overcome the problems, we propose to use 1-d binary Gaussian mixture probability density function to calculate the separation measure. Denote s_{ij} as the separation measure between $\mathcal{C}_i$ and $\mathcal{C}_j$. Their cluster members $\mathbf{x}$'s are first projected into the line between $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_j$ [54]

$$x' = \frac{(\mathbf{x} - \boldsymbol{\mu}_0)^T(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)}{\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|^2} \tag{14}$$

where $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_j$ are the centers of $\mathcal{C}_i$ and $\mathcal{C}_j$, and $\boldsymbol{\mu}_0 = (\boldsymbol{\mu}_i + \boldsymbol{\mu}_j)/2$ is the middle point between two centers. The projected members are 1-D points. The centers $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_i$ are projected to the position -0.5 and 0.5, respectively. Thus, we can obtain two sets of 1-D points by projecting the cluster members from $\mathcal{C}_i$ and $\mathcal{C}_j$. The mean and variance of the projected points, μ_i, μ_j, σ_i^2, and σ_j^2 can then be calculated. Therefore, the binary Gaussian mixture probability density function is written as

$$f(u) = \frac{|\mathcal{C}_i|}{|\mathcal{C}_i| + |\mathcal{C}_j|} p\left(u_i|0.5, \sigma_i^2\right) + \frac{|\mathcal{C}_i|}{|\mathcal{C}_i| + |\mathcal{C}_j|} p\left(u_j| - 0.5, \sigma_j^2\right) \tag{15}$$

where $|\mathcal{C}_i|$ is the size of $\mathcal{C}_i$. Then, we use an interval with step 0.01 from -0.5 to 0.5 $\mathcal{A} = \{-0.5, -0.49, \ldots, 0.49, 0.5\}$ to calculate the discrete probability densities $f(\mathcal{A})$. The separation s_{ij} is then calculated by

$$s_{ij} = \frac{1}{\min f(\mathcal{A})}. \tag{16}$$

If two subclusters are well-separated, $\min f(\mathcal{A})$ is close to 0 and s_{ij} is large as shown in Fig. 6(a). If two subclusters are close to each other, the probability density of the overlapping area is high and s_{ij} is low as shown in Fig. 6(b). Thus, higher value of s_{ij} indicates higher separability of $\mathcal{C}_i$ and $\mathcal{C}_j$. As shown in Fig. 6, s_{ij} for Fig. 6(a) is 13.37 and s_{ij} for Fig. 6(b) is 2.86. By using the 1-d binary Gaussian mixture probability density function, the problem of bin size in [54] is avoided. In addition, the low density region between two subclusters can be any position between the subcluster centers because SGMS locates the region by the minimum position in the mixture probability density function.

After obtaining the separation measure of each pair of subclusters, we merge them according to the value of s_{ij}. We adopt

Algorithm 2 SGMS

Input: $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, $\mathcal{M}$, $\hat{K}$, κ.

1: Assign cluster membership of data points in $\mathcal{D}$ to subclusters $\{\mathcal{C}_1, \ldots, \mathcal{C}_{\hat{K}}\}$ by minimum distance with seed points in $\mathcal{M}$.
2: **for** $i \leftarrow 1$ to $\hat{K}$ **do**
3: **for** $j \leftarrow i+1$ to $\hat{K}$ **do**
4: Calculate pairwise separation s_{ij} by (16);
5: **end for**
6: **end for**
7: **for** $K \leftarrow 1$ to $\hat{K}$ **do**
8: $G_K \leftarrow \{\mathcal{C}_K\}$
9: **end for**
10: $\mathcal{G}^{\hat{K}} \leftarrow \{G_1, \ldots, G_{\hat{K}}\}$;
11: **for** $K \leftarrow \hat{K}$ to 2 **do**
12: Select G_i, G_j and calculate com_{K-1} by (17);
13: $\mathcal{G}^{K-1} \leftarrow \mathcal{G}^K \backslash \{G_i, G_j\} \cup \{G_i \cup G_j\}$.
14: Calculate sep_{K-1} by (19);
15: **end for**
16: Select the best number of clusters:
$K^* \leftarrow \arg\min_{1 \le K \le \hat{K}-1}(sep_K + com_K)$;

Output: Group set $\mathcal{G}^{K^*}$.

the grouping algorithm used in [22]. We denote the subclusters $\{\mathcal{C}_1, \ldots, \mathcal{C}_{\hat{K}}\}$ and the initial group set $\mathcal{G}^{\hat{K}} = \{G_1, \ldots, G_{\hat{K}}\}$. Initially, each group contains only one subcluster $G_K = \{\mathcal{C}_K\}$ for $K = 1, \ldots, \hat{K}$. Then, the groups G_i and G_j are selected by

$$com_{K-1} = \min_{1 \le i < j \le \hat{K}} \min_{\mathcal{C}_{i'} \in G_i, \mathcal{C}_{j'} \in G_j} s_{i'j'}. \tag{17}$$

The groups G_i and G_j are selected as the closest groups because the minimum separation measure is from their group members. The group set $\mathcal{G}^{K-1}$ for $K = 2, \ldots, \hat{K}$ is sequentially obtained by

$$\mathcal{G}^{K-1} = \mathcal{G}^K \backslash \{G_i, G_j\} \cup \{G_i \cup G_j\}. \tag{18}$$

The original groups G_i and G_j are removed from G and the new group $G_i \cup G_j$ is added to G. K is decreased by 1 and the grouping algorithm stops at $K = 1$. The minimum value of *com* in (17) is recorded as the global compactness of $\mathcal{G}^K$. It is the maximum within-group separation measure in the current group set. The lower value of *com* indicates better clustering result because each cluster is compact. Therefore, com_K is increasing from $K = \hat{K} - 1$ to 1. When two groups in different clusters are merged, the separation measure between them is supposed to be much higher than the one when merging the groups in the same cluster. That results in large value of *com* after merging. In addition to the global compactness, we modify global separability defined in [56]

$$\text{sep}_K = \max_{G \in \mathcal{G}^K} \sum_{i \in G} (q_i/\kappa) \tag{19}$$

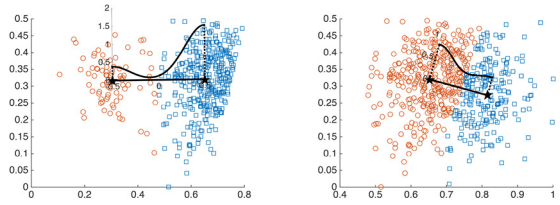
where κ is the predefined number of nearest neighbors and q_i is the number of points that are not in G from κ nearest neighbors. The original separability in [56] is divided by the number of members $|G_i|$ in each group. However, for imbalanced clusters, if one group only contains a minority cluster, it will dominate in the maximization because q_i/κ is multiplied with $1/|G_i|$. Finally, we select the minimum value from the sum of global separability and global compactness

$$K^* = \arg\min_{1 \le K \le \hat{K}-1} \left(\frac{\text{sep}_K}{\max_{K'}\{\text{sep}_{K'}\}} + \frac{com_K}{\max_{K'}\{com_{K'}\}} \right). \tag{20}$$

The global compactness and global separability are normalized by dividing their maximal value to scale into the same range.

The summary of SGMS is shown in Algorithm 2. The data points are first assigned to subclusters according to the distance to the seed points shown in line 1. The pairwise separation measure between subclusters is then calculated as shown in lines 2–6. The grouping and the clustering validation measure to determine the number of clusters are shown in lines 7–16. The computational cost of SGMS algorithm is $O(\hat{K}N + \hat{K}^2 \log \hat{K} + N^2/\hat{K})$.

In summary, the overall computational cost of SMCL, including PNS and SGMS, i s $O(E^*\hat{K}N + \hat{K}^2 \log \hat{K} + N^2/\hat{K})$. The first term $E^*\hat{K}N$ refers to the updating of winner and rival seed points in PNS, and the second term $\hat{K}^2 \log \hat{K}$ refers to the calculation of global compactness and separation. The last term $N^2/\hat{K}$ refers to the calculations of density gap (11) in PNS and global separability (19) in SGMS. Both of them use the pairwise distance of the original data points, which can be computed in advance to save the computational cost.



(a) (b)

Fig. 6. Example of calculating separation measure by using 1-d Gaussian mixture probability density function. (a) Two subclusters belong to different clusters. $s_{ij} = 13.37$. (b) Two subclusters belong to the same cluster. $s_{ij} = 2.86$.

IV. Experimental Results

A. Datasets and Compared Methods

To evaluate the performance of the proposed method, we compare SMCL with nine clustering algorithms on four synthetic and eight real datasets. For synthetic datasets, two of them are generated from a mixture of bivariate Gaussian density functions $\sum_i \alpha_i f(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$. Dataset *gaussian* has 2000 data points with two majority clusters and two minority clusters, whose centers are located in the corner of a square with edge length 4. The proportion of these four classes are 1:2:10:20. The two majority clusters have identity covariance and the minority clusters have 0.25 times the identity covariance. Dataset *ids2* has 3200 data points with one majority cluster and four minority clusters. The parameters are set as same as in [22]. Datasets *banana* and *lithuanian* are generated by PRTools [57] with the thickness parameter 0.4. There

TABLE I
INFORMATION OF FOUR SYNTHETIC DATASETS AND EIGHT UCI REAL DATASETS

Dataset	#Feature	#Data	Cluster size
gaussian	2	2000	61, 1212, 606, 121
ids2	2	3200	2000, 200, 200, 400, 400
banana	2	2400	2000, 400
lithuanian	2	2400	2000, 400
breast cancer	9	683	444, 239
ecoli	5	327	143, 77, 52, 35, 20
haberman	3	306	225, 81
car	6	1728	1210, 384, 69, 65
pageblock	10	5357	4913, 329, 115
wpbc	32	198	47,151
avila	10	20867	8572, 10, 206, 705, 2190, 3923, 893, 1039, 1663, 89, 1044, 533
shuttle	9	43500	34108, 37, 132, 6748, 2458, 6, 11

are one minority cluster with 400 data points and one majority cluster with 2000 data points. The clusters in *banana* and *lithuanian* are nonspherical. In addition, eight real datasets from UCI data repository [58] are used in the experiment. The information of the datasets is listed in Table I. The features of all datasets are normalized to [0, 1].

The compared methods are adaptive k-means [8], RPCL [14], RPCCL [16], DSRPCL [17], CPCL [27], GMPCL [25], MC [22], RMD [53], and CHKNN [37]. Among these methods, adaptive k-means RPCL, RPCCL, CPCL, DSRPCL, and GMPCL are competitive learning-based methods. MC and RMD are especially designed for imbalanced clustering problem. CHKNN is the state-of-the-art nonlinear clustering method. Except adaptive k-means and RMD, all methods are able to automatically determine the number of clusters. Therefore, we use the real number of clusters as the number of seed points for adaptive k-means and RMD. For other competitive learning-based methods with automatically determination of the number of clusters, the initial number of seed points for RPCL, RPCCL, DSRPCL, and CPCL are set at double of the real number of clusters. Given a relative large number of seed points, the rival penalization mechanism is capable of driving the redundant seed points away and keeping a few seed points only to represent the clusters. For MC, the number of prototypes $k_{\max}$ is also set at double of the real number of clusters. For learning rate and delearning rate, we set $\alpha_c = 0.01$ for adaptive k-means; $\alpha_c = 0.01$ and $\alpha_r = 0.001$ for RPCL; $\alpha_c = 0.001$ for RPCCL; $\alpha_c = 0.01$ for DSRPCL; and $\alpha_c = 0.0001$ for CPCL. For other parameters of DSRPCL, GMPCL, and MC, we use the suggested value in the papers. For GMPCL and MC, their own methods are used to select the number of clusters. The epoch number of all compared competitive learning methods are set at 100.

For the parameters of SMCL, we set $K_0 = 2$ as the initial number of seed points because the number of seed points can be adaptively increased by SMCL until a proper size is achieved. The learning rate α_c is set at 0.01 and η is set at 0.01, in order to make sure that PNS does not terminate too soon. The frequency threshold is set at $\theta = 0.01N$, which means if the winning times of a seed points is less than 1/100 of the total number of data points in an epoch, it will be driven off.

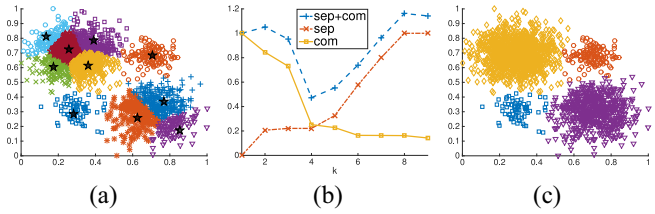


Fig. 7. Dataset *gaussian*. (a) Ten subclusters generated by PNS. (b) Global separability, global compactness, and their sum. (c) Clustering result by SMCL.

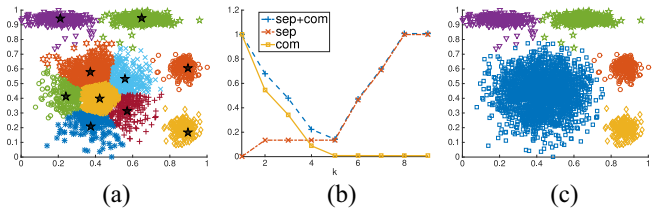


Fig. 8. Dataset *ids2*. (a) Ten subclusters generated by PNS. (b) Global separability, global compactness, and their sum. (c) Clustering result by SMCL.

The convergence termination threshold is set at $\xi = 0.0001$. When we calculate the global separability, we use $\kappa = 5$ for kNN search.

B. Evaluation Metrics

For numerical comparison, we adopt four measurements to evaluate the clustering results: 1) accuracy; 2) F-measure; 3) normalized mutual information (NMI); and 4) difference of CV (DCV) [21]. Among these four measurements, accuracy, F-measure, and NMI are commonly used to evaluate the clustering result. DCV is specifically designed for clustering on imbalanced datasets [21]. It measures the difference of ratios of the standard deviation to the mean between the number of data points in ground truth clusters and predicted clusters

$$\mathrm{DCV} = |\mathrm{CV}_g - \mathrm{CV}_p| \tag{21}$$

where $\mathrm{CV}_g = ([\sqrt{\sum_{i=1}^{k}(n_i - \bar{n})}]/[(k-1)\bar{n}])$, $\mathrm{CV}_p = ([\sqrt{\sum_{j=1}^{k'}(n'_j - \bar{n}')}]/[(k'-1)\bar{n}'])$, $\bar{n} = (1/k)\sum_{i=1}^{k} n_i$, and $\bar{n}' = (1/k')\sum_{j=1}^{k'} n_j$ are the mean number of data points in each cluster for ground truth clusters and predicted clusters, respectively. If the data is imbalanced, CV_g will be high because the standard deviation of the number of data points in each cluster is high. If the cluster size of predicted clustering result tends to be balanced, CV_p will be low, so that the difference DCV is high. DCV is used as a necessary criterion to evaluate the clustering results [22]. Small DCV does not necessarily mean good clustering result, but large DCV indicates poor clustering result. The estimated number of clusters K is also shown to evaluate if the methods can correctly determine the number of clusters for real imbalanced datasets. The results are averaged by ten runs with different random initialization of the seed points.

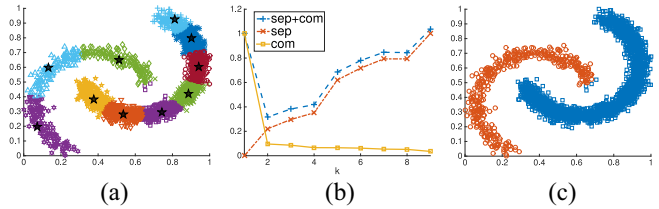


Fig. 9. Dataset *banana*. (a) Ten subclusters generated by PNS. (b) Global separability, global compactness, and their sum. (c) Clustering result by SMCL.

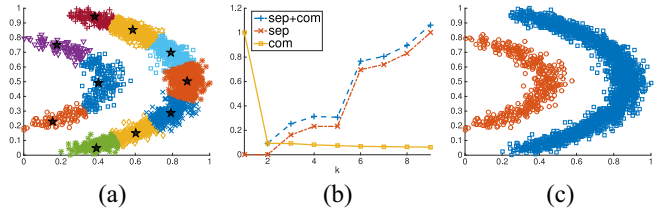


Fig. 10. Dataset *lithuanian*. (a) Ten subclusters generated by PNS. (b) Global separability, global compactness, and their sum. (c) Clustering result by SMCL.

C. Results and Discussion

The visual clustering results of SMCL on synthetic datasets are shown in Figs. 7–10. The black stars show the final positions of the seed points after PNS terminates. For dataset *gaussian*, ten subclusters are generated when PNS terminates. It can be observed that two majority clusters are well represented by five and three seed points, respectively. The seed points in the minority clusters are not influenced by the uniform effect and they are located close to the centers of the clusters. In Fig. 7(b), the global compactness drops significantly from $k = 3$ to $k = 4$. The reason is that, at $k = 3$, one minority cluster and one majority cluster are grouped together and the gap between them makes global compactness large. The global separation increases from $k = 4$ to $k = 5$ while it keeps stable from $k = 2$ to $k = 4$. A plausible reason is that at $k = 5$, one majority cluster is split and the number of nearest neighbors in different groups increases by (19). Thus, the best number of clusters obtained at $k = 4$ shows the correct number of clusters. Due to space limitation, we only show the moving trajectory of the seed points by SMCL on dataset *gaussian* in Fig. 11. The squares are the starting positions and the stars are the final positions. The number beside the square indicates the order of appearance of the seed point. It can be observed that all new seed points are generated in the majority cluster. Finally, the seed point marked as "4" is driven away from one of the majority clusters when the number of the seed points in that cluster increases to 6.

For dataset *ids2*, ten subclusters are generated when PNS terminates. The majority cluster is represented by six seed points and each minority cluster is represented by one. As shown in Fig. 8(b), the global separation increases significantly from $k = 5$ to $k = 6$ when the majority cluster is split. The global compactness decreases to nearly 0 at $k = 5$. Therefore, the number of clusters is chosen to be 5. For dataset *banana*, ten subclusters are generated when PNS terminates. The majority cluster and the minority cluster have seven and

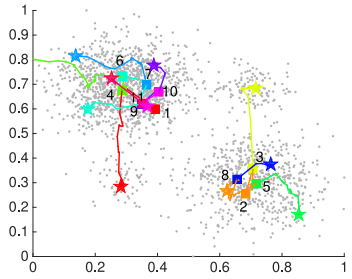


Fig. 11. Moving trajectory of seed points by SMCL on dataset *gaussian*. The square is the starting position and the star is the final position. The number indicates the appearing order of seed points.

three seed points, respectively. As shown in Fig. 9(b), the global compactness drops significantly from $k = 1$ to $k = 2$ and, therefore, the minimum value of the internal validation measure is at $k = 2$. There are only four data points in the majority clusters are misclassified into the minority cluster due to the shape of the subclusters. Nevertheless, each cluster is well represented by multiple seed points. For dataset *lithuanian*, ten subclusters are generated when PNS terminates. As shown in Fig. 10(b), the global compactness drops significantly from $k = 1$ to $k = 2$ and the correct number of clusters are determined. The global separation is close to 0 at $k = 1$ and $k = 2$ because none of the neighbors are in different groups. All the data points are correctly clustered by SMCL.

The numerical results of synthetic datasets are shown in Table II. The best and second best results are shown in bold face and underline, respectively. For dataset *gaussian*, GMPCL achieves best results on accuracy, F-measure, and NMI; while SMCL, RMD, MC, and DSRPCL achieve comparable results on these measurements. However, DCV of DSRPCL is larger, which indicates that DSRPCL tends to produce more balanced clustering results on this dataset than other four methods. RPCL and CPCL fail to drive the rival seed points away and, therefore, all eight initial seed points are left. It can be observed that all multiprototype methods, that is, GMPCL, MC, and SMCL, correctly estimate the number of clusters. However, GMPCL achieves the best performance on accuracy, F-measure, NMI, and DCV among them. It indicates that the different number and position of the seed points may lead to different clustering results. The same phenomenon also appears on dataset *ids2*, while both SMCL and GMPCL correctly estimate K but SMCL performs better. For dataset *ids2*, SMCL achieves the best results on accuracy and F-measure, and the second best results on NMI and DCV. RMD also achieves good performance on all measurements probably because of the input of real number of clusters. GMPCL, MC, and SMCL correctly determine the number of clusters, although their performance on other measurements is slightly lower than SMCL. Besides, DSRPCL also achieves good performance. However, RPCL, CPCL, and RPCCL remain a relative large number of seed points and thus achieve poor clustering performance. For dataset *banana* and *lithuanian*, the shape of the clusters are nonspherical. No single seed point can represent the cluster. Therefore, the single prototype methods, that is, Adaptive k-means, RPCL,

TABLE II
RESULTS OF THE SYNTHETIC DATASETS. THE BEST AND SECOND BEST RESULTS ARE SHOWN IN BOLD FACE AND UNDERLINE, RESPECTIVELY

Dataset	Measurement	Adaptive k-means	RPCL	RPCCL	DSRPCL	CPCL	GMPCL	MC	RMD	CHKNN	SMCL
gaussian	Accuracy	0.8369	0.4292	0.7805	0.9753	0.5137	**0.9841**	0.9679	0.9810	0.8560	0.9780
	F-measure	0.8733	0.5666	0.8288	0.9579	0.6239	**0.9844**	0.9681	0.9815	0.9010	0.9791
	NMI	0.7013	0.6199	0.7383	0.7264	0.5218	**0.9135**	0.9005	0.9002	0.8261	0.9102
	DCV	0.4242	0.6532	0.2155	0.1032	0.4143	0.0266	0.0448	**0.0166**	0.0829	0.0400
	Estimated K	-	5.0000	5.3000	3.8000	8.0000	**4.0000**	**4.0000**	-	8.0000	**4.0000**
ids2	Accuracy	0.7424	0.4960	0.5951	0.9890	0.5506	0.9888	0.9891	0.9909	0.9631	**0.9930**
	F-measure	0.7742	0.5782	0.6909	0.9808	0.6416	0.9889	0.9891	0.9909	0.9771	**0.9930**
	NMI	0.7337	0.7044	0.7562	**0.9646**	0.6620	0.7564	0.9506	0.9568	0.9498	0.9620
	DCV	0.6383	0.9202	0.7432	0.0656	0.6235	0.0298	0.0298	**0.0171**	0.1539	0.0177
	Estimated K	-	9.8000	8.0000	5.1000	10.0000	**5.0000**	**5.0000**	-	6.0000	**5.0000**
banana	Accuracy	0.7329	0.3863	0.5509	0.4855	0.4282	0.6150	0.7517	0.5946	0.6183	**0.9890**
	F-measure	0.7623	0.5016	0.5913	0.6028	0.5419	0.7515	0.6756	0.6184	0.7641	**0.9932**
	NMI	0.1021	0.1686	0.1689	0.1808	0.1150	0.7959	0.7052	0.1774	0.5547	**0.9722**
	DCV	0.4608	0.8553	0.6420	0.4458	0.5903	0.4946	0.2310	0.6753	0.0824	**0.0733**
	Estimated K	-	3.9000	2.8000	3.6000	3.9000	7.0000	**2.0000**	-	6.0000	2.2000
lithuanian	Accuracy	0.5307	0.4019	0.5101	0.5513	0.4236	**1.0000**	0.6558	0.7825	0.5317	**1.0000**
	F-measure	0.5894	0.5302	0.6242	0.6698	0.5530	**1.0000**	0.6987	0.6937	0.6942	**1.0000**
	NMI	0.0008	0.1930	0.1358	0.1607	0.1940	**1.0000**	0.2077	0.0933	0.5655	**1.0000**
	DCV	0.8580	0.7658	0.7031	0.5924	0.5894	**0.0000**	0.9263	0.1438	0.1986	**0.0000**
	Estimated K	-	3.9000	3.0000	3.2000	3.9000	**2.0000**	**2.0000**	-	5.0000	**2.0000**

RPCCL, DSRPCL, and CPCL, generate poor results on them. For dataset *banana*, SMCL achieves the best performance on accuracy, F-measure, NMI, and DCV. It outperforms the second best performance 0.2373, 0.2291, and 0.1763 on accuracy, NMI, and F-measure, respectively. MC estimates the correct number of clusters but the performance is poor. The reason is that the number of prototypes of MC, which is predefined, does not well represent the clusters. For dataset *lithuanian*, both GMPCL and SMCL correctly cluster all the data points. MC suffers the same problem as in dataset *banana*.

The numerical results of real datasets are shown in Table III. The best and second best results are shown in bold face and underline, respectively. For dataset *breast cancer*, adaptive k-means and RMD achieve the best performance on accuracy and F-measure. However, the real number of clusters are given to adaptive k-means and RMD. SMCL, RPCCL, and GMPCL all correctly estimate the number of clusters, meanwhile SMCL achieves the best accuracy, F-measure, and DCV among them. The performance of SMCL is very close to RMD, while the number of clusters is automatically determined by SMCL. RPCL, DSRPCL, CPCL, and CHKNN produce inaccurate clustering results due to wrong estimation of K. For dataset *ecoli*, MC achieves the best accuracy. However, its F-measure, NMI, and DCV are poor due to wrong estimation of K. SMCL achieves the best F-measure and the second best NMI. Although no method correctly estimates the number of clusters is 5, the estimated K of SMCL is the closest to the real number of clusters. Besides, DCV of SMCL is the lowest, which indicates that the variance of the sizes of the predicted clusters is close to the variance of the ground truth. GMPCL, MC, and CHKNN underestimate the number of clusters and, therefore, receive low F-measure and NMI, although they get high accuracy. Other competitive learning methods generally overestimate the number of clusters on this dataset. For dataset *haberman*, CHKNN achieves the best accuracy and F-measure followed by SMCL. NMI for all methods on this dataset is low and the best 0.0538 is produced by MC. A possible reason of the low NMI is that the clusters of dataset *haberman* are highly overlapped. DCV of MC is the best but its accuracy and F-measure is poor. Except CHKNN, which correctly estimates the number of clusters, the method with estimated K closest to the real number of cluster is SMCL. For dataset *car*, SMCL achieves the best accuracy, F-measure, and DCV. However, it only estimates two clusters and the real number of clusters of dataset *car* is 5. GMPCL and CHKNN overestimate the number of clusters and their F-measure is comparable with SMCL. But their performance on accuracy and NMI is not good. DCV of SMCL is the lowest. A possible explanation is that SMCL merges two majority cluster and two minority cluster in dataset *car* and, therefore, it has high accuracy and F-measure, and low DCV. For dataset *pageblock*, SMCL achieves the best NMI and DCV, and estimates the correct K. CHKNN achieves the best F-measure and the second best NMI. RPCCL and RMD also achieve high accuracy and F-measure. However, their NMI is low. GMPCL and MC correctly estimate K with comparable results on other measurements. For dataset *wpbc*, SMCL achieves the best accuracy and correctly estimates K. GMPCL has the best F-measure and second best accuracy. However, its DCV is relatively large which means that the clustering results are relatively balanced. Among three methods that correctly estimates K, MC and SMCL have generally low DCV, and high accuracy and F-measure. For dataset *avila*, RMD achieves the best accuracy followed by SMCL. However, the NMI and DCV of RMD is worse than SMCL. DSRPCL and CHKNN achieve comparable F-measure with SMCL, but their accuracy and NMI are lower than the ones of SMCL. SMCL has the estimated K that is closest to the real number of clusters. For comparison, RPCCL overestimates K although it has the lowest DCV. For dataset *shuttle*, RMD and SMCL generally achieves the best result, while RMD has the best results on accuracy and NMI, and SMCL has the best results on F-measure and DCV. MC is the only method that correctly estimates K, while the second closest method is SMCL whose estimated K is 5.8. To sum up, MC, RMD, and SMCL all achieve the comparable results on dataset *shuttle*.

D. Parameter Sensitivity

The key parameters of SMCL are α_c and η, where α_c is the learning rate and η is to control the magnitude of rival

TABLE III
RESULTS OF THE REAL DATASETS. THE BEST AND SECOND BEST RESULTS ARE SHOWN IN BOLD FACE AND UNDERLINE, RESPECTIVELY

	Measures	Adaptive k-means	RPCL	RPCCL	DSRPCL	CPCL	GMPCL	MC	RMD	CHKNN	SMCL
	Accuracy	0.9602	0.6843	0.9482	0.7753	0.6551	0.9266	0.8594	**0.9663**	0.5124	0.9590
	F-measure	0.9600	0.7823	0.9475	0.8298	0.7724	0.9249	0.9074	**0.9664**	0.6013	0.9588
breast cancer	NMI	0.7412	0.5859	0.7058	0.6303	0.5713	0.5209	**0.8259**	0.7783	0.4898	0.7622
	DCV	**0.0414**	0.3454	0.0944	0.6306	0.1824	0.1636	0.3910	0.0490	0.3152	0.0431
	Estimated K	-	3.8000	**2.0000**	3.9000	4.0000	**2.0000**	3.0000	-	7.0000	**2.0000**
	Accuracy	0.7346	0.4878	0.6052	0.8917	0.5223	0.9606	**0.9908**	0.8318	**0.9633**	0.9407
	F-measure	0.7070	0.5904	0.6458	0.7876	0.5550	0.6373	0.4281	0.7870	0.6396	**0.8460**
ecoli	NMI	0.6318	0.5634	0.5678	**0.7243**	0.5077	0.5128	0.5527	0.6815	0.5327	0.7239
	DCV	0.3206	0.3090	0.1578	0.5788	0.1001	0.2313	0.6501	0.1198	0.2235	**0.0318**
	Estimated K	-	10.0000	8.7000	7.9000	9.8000	2.0000	2.0000	-	2.0000	**4.0000**
	Accuracy	0.5261	0.3268	0.4944	0.5042	0.4141	0.5131	0.5033	0.5556	**0.8824**	0.8085
	F-measure	0.5527	0.4212	0.5260	0.5329	0.4758	0.5264	0.5459	0.5587	**0.6924**	0.6667
haberman	NMI	0.0009	0.0148	0.0121	0.0120	0.0229	0.0429	**0.0538**	0.0002	0.0078	0.0013
	DCV	0.6082	0.4684	0.1632	0.2866	0.2514	0.5245	**0.0467**	0.5084	0.4159	0.4046
	Estimated K	-	4.0000	3.5000	3.7000	4.0000	7.0000	3.0000	-	**2.0000**	2.5000
	Accuracy	0.3885	0.2363	0.2821	0.2377	0.3713	0.8052	0.8750	0.3912	0.8519	**0.9344**
	F-measure	0.4639	0.3243	0.3626	0.3298	0.4377	0.6289	0.6304	0.4603	0.6398	**0.6455**
car	NMI	0.1101	**0.1522**	0.1250	0.1266	0.1251	0.0522	0.0304	0.1441	0.0948	0.1092
	DCV	1.1242	1.1378	0.8451	1.0954	0.6783	0.4424	0.1888	0.9982	0.5725	**0.0393**
	Estimated K	-	8.0000	7.9000	8.0000	5.9000	**5.0000**	2.0000	-	**5.0000**	2.0000
	Accuracy	0.6599	0.3342	**0.9974**	0.9014	0.4857	0.9699	0.9110	0.9953	0.9055	0.9366
	F-measure	0.7310	0.4668	0.8858	0.8915	0.5911	0.8846	0.8868	0.8837	**0.9034**	0.8944
pageblock	NMI	0.0647	0.1133	0.0143	0.1233	0.0730	0.1717	0.1678	0.0081	0.2117	**0.2261**
	DCV	0.7242	1.0043	0.1110	0.3781	0.9001	0.1360	0.1136	0.2020	0.2012	**0.0323**
	Estimated K	-	6.0000	2.0000	3.9000	4.5000	**3.0000**	**3.0000**	-	4.0000	**3.0000**
	Accuracy	0.5929	0.3278	0.5727	0.3581	0.5273	0.9061	0.7727	0.8131	0.8673	**0.9850**
	F-measure	0.6231	0.4263	0.5814	0.4322	0.5522	**0.7334**	0.6593	0.6906	0.5767	0.6604
wpbc	NMI	0.0233	0.0240	0.0172	0.0231	0.0189	0.0052	0.0178	0.0013	0.0189	**0.0472**
	DCV	0.6228	0.5303	0.1907	0.3424	0.2559	0.4057	**0.0286**	0.1428	0.3425	0.1041
	Estimated K	-	4.0000	3.3000	4.0000	3.7000	**2.0000**	**2.0000**	-	3.0000	**2.0000**
	Accuracy	0.3241	0.1490	0.3312	0.4625	0.3542	0.9367	0.9289	**0.9565**	0.7826	0.9514
	F-measure	0.3057	0.2033	0.3103	0.3450	0.3374	0.2685	0.3078	**0.3768**	0.3471	0.3425
avila	NMI	0.1478	0.1692	0.1598	0.0982	0.0816	0.0471	0.1289	0.0425	0.1078	**0.1874**
	DCV	0.4775	1.0218	**0.1006**	1.3865	0.9275	1.2378	0.6253	0.5966	1.3865	0.3966
	Estimated K	-	24.0000	18.5000	4.8000	5.5000	8.6000	9.0000	-	4.0000	**10.4000**
	Accuracy	0.4238	0.2404	0.3739	0.2573	0.3781	0.0379	0.9041	**0.9127**	0.7829	0.8944
	F-measure	0.5808	0.3449	0.5247	0.4827	0.5165	0.0715	0.8236	0.8425	0.6898	**0.8932**
shuttle	NMI	0.4855	0.4259	0.4798	0.3827	0.4381	0.3094	0.5728	**0.7018**	0.3627	0.6141
	DCV	1.4500	1.6825	1.2336	2.0240	1.5364	1.1674	0.5627	0.3827	1.2834	**0.1712**
	Estimated K	-	14.0000	10.7000	4.2000	5.0000	5.5000	**7.0000**	-	4.0000	5.8000

penalization. We first give a guidance of how to select these two parameters. The learning rate α_c is related to convergence. If the seed points cannot converge to satisfy the condition in line 11 of Algorithm 1, α_c should be decreased. On the other hand, if α_c is set too small, the degree of updating the seed points will be too tiny such that the convergence is detected but the seed points are not in their destinations. η is related to the termination of PNS. If η is set too large, PNS will terminate very soon with few seed points remaining, which is inefficient to represent the clusters as shown in Fig. 3(a). On the other hand, if η is too small, PNS will run until the last epoch with no seed points driven away. That will lead to the result as shown in Fig. 3(b). As a rule of thumb, α_c and η should be adjusted to make sure that PNS successfully converges for every few epochs after adding a new seed point and finally stops when one seed point is driven away. To show the sensitivity of these two parameters, we test their performance with the values in $[0.0001, 0.001, 0.01, 0.1]$ on the eight real datasets with three metrics. We fix one at 0.01 as used in the above experiments and test another. Fig. 12 shows the results on accuracy. It can be observed that α_c is stable for datasets *ecoli*, *pageblock*, and *shuttle*. For datasets *car* and *wpbc*, the accuracy drops at α_c is 0.1. Except datasets *haberman* and *shuttle*, all other datasets show relative stable performance with parameter η. Fig. 13 shows the results on F-measure. Both α_c and η are not very sensitive in terms of F-measure, except a significant drop of dataset *breast cancer* at $eta = 0.0001$. Fig. 14 shows the results on NMI. The datasets *car*, *wpbc*, and *haberman* are not sensitive to the parameter selection because they have very low NMI values. For dataset *breast cancer*, *ecoli*, and *shuttle*, it can be observed that α_c and η generally perform well between the value of 0.001 and 0.01. Fig. 15 shows the results on DCV. It can be observed that when α_c and η are set at 0.001, the DCVs for most of the cases are less than 0.4 and when they are set at 0.01, the DCVs for most of the cases are less than 0.2. Overall speaking, the parameters α_c and η are not very sensitive in terms of the performance for most of the cases when their values range from 0.001 to 0.01.

In addition, there is another parameter, that is, the number of nearest neighbors κ, to calculate the global separability in Algorithm 2. Our empirical studies have found that the selection of κ does not influence on determining the number of clusters of SGMS too much. In general, a value between 5 and 10 is a good choice for κ and the clustering result is not very sensitive to this choice. A plausible reason is that the value of κ will not affect the ratio q_i/κ too much in (19) because q_i increases over κ. Besides, the calculated global separability is further normalized by dividing its maximum value as shown in (20) so that if a large κ is used, sep_K for all K will be increased and the normalized sep_K will not be influenced too much. However, if κ is set too large and even

TABLE IV
RUNNING TIME (IN SECOND) OF SMCL AND THE COMPARED METHODS

Dataset	Adaptive k-means	RPCL	RPCCL	DSRPCL	CPCL	GMPCL	MC	RMD	CHKNN	SMCL
gaussian	31.37	31.66	48.15	2.38	2.55	3.05	53.24	80.49	0.77	32.49
ids2	62.99	71.50	99.00	2.41	0.81	5.30	149.39	387.91	1.06	66.92
banana	33.58	39.21	57.34	2.13	0.06	4.72	32.82	130.72	0.42	36.25
lithuanian	33.95	32.27	54.58	2.02	0.46	4.65	32.09	141.65	0.48	32.50
breast cancer	7.91	6.80	13.53	0.61	0.36	1.91	3.60	8.10	0.17	7.57
ecoli	4.14	3.51	7.04	0.36	0.02	0.44	5.72	4.01	0.20	3.65
haberman	3.71	2.79	5.98	0.31	0.02	0.42	1.11	3.03	0.02	3.62
car	27.50	28.33	47.10	1.61	1.99	6.44	39.98	57.47	0.27	26.00
pageblock	148.11	184.15	222.20	4.89	2.65	46.00	227.86	865.35	6.49	180.01
wpbc	2.52	2.05	4.11	0.22	0.01	0.30	0.74	1.17	0.01	2.44
avila	2432.36	3146.75	3453.32	21.33	42.35	504.32	16 189.49	5521.84	50.20	2657.73
shuttle	13 220.84	16 399.48	18 885.28	40.87	98.20	2785.44	57 225.48	34 808.60	167.22	15 807.23

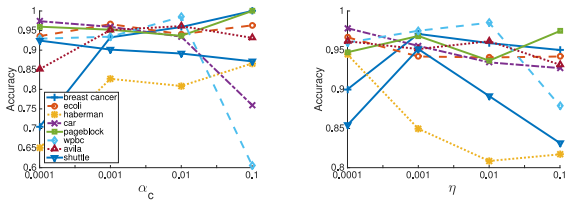


Fig. 12. Parameter sensitivity of SMCL on accuracy.

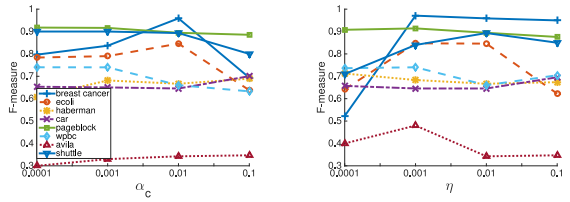


Fig. 13. Parameter sensitivity of SMCL on F-measure.

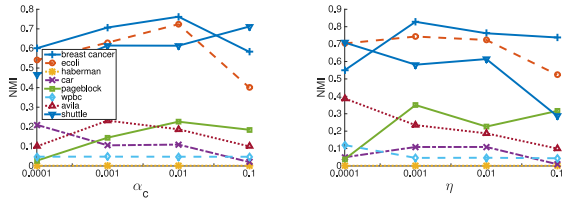


Fig. 14. Parameter sensitivity of SMCL on NMI.

more than the number of points in the subcluster, the calculated global separability will not provide informative leap to show the change of separability when subclusters are merging. On the other hand, if κ is set too small, the calculation of global separability will not be accurate as expected.

E. Running Time Analysis

The codes are written by MATLAB 2017b and all experiments run on a Window 10 PC with i7 3.60 GHz and 64-GB RAM. The running time in second is shown in Table IV. Among the competitive learning-based methods, DSRPCL and CPCL have the lowest computational cost due to their high convergence speed. However, they do not perform well as shown in Tables II and III. GMPCL also has low cost on low-dimensional datasets. The running time of SMCL is comparable with adaptive k-means and lower than RPCL and RPCCL. The reason is that the number of seed points of SMCL is initialized as 2 and increases incrementally. Thus, less distance calculation between the data points and the seed points are needed at the early learning stages of the algorithm. Besides, many components based on pairwise distance between the original data points can be precomputed in Algorithms 1 and 2. As a conclusion, the computational cost of SMCL is in the same level of k-means, which is consistent with the computational cost analysis in Section III.

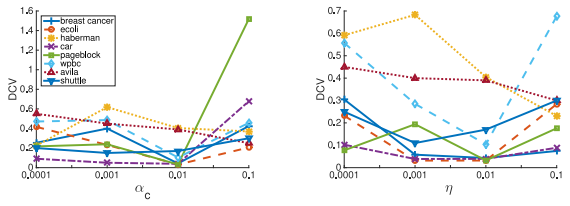


Fig. 15. Parameter sensitivity of SMCL on DCV.

V. CONCLUSION

In this paper, we have studied the competitive learning methods on imbalanced datasets and proposed a novel method SMCL, which uses the multiprototype strategy with automatically choosing the number of seed points in a self-adaptive way. It incrementally generates new seed points to the data so that each cluster can be well represented by a proper number of seed points. Then, we have also presented a new separation measure to merge the subclusters and find the best number of clusters during the process by a new internal clustering validation measure. Experiments on synthetic and real imbalanced datasets show that SMCL outperforms the other competitive learning methods and multiprototype clustering methods in terms of accuracy, F-measure, and DCV. In particular, it correctly determines an appropriate number of clusters in most of the cases we have tried so far.

Along this paper, future work has at least three directions. One direction is how to extend the proposed SMCL or related clustering methods to multiview clustering, where multiple sets of features and information for one individual subject are integrated. Multiview features might enhance and improve the clustering results of the minority clusters. The second direction is to explore a robust clustering method against the imbalanced data and outliers. The key challenge of this direction is to distinguish the minority clusters and the outliers. The last one is to apply index technologies to improve the efficiency of SMCL or propose a new efficient clustering algorithm so that it could be scalable to large-scale datasets with imbalanced clusters.

REFERENCES

[1] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.

[2] P. Branco, L. Torgo, and R. P. Ribeiro, "A survey of predictive modeling on imbalanced domains," *ACM Comput. Surveys*, vol. 49, no. 2, p. 31, Nov. 2016.

[3] W. Li, L. Fu, B. Niu, S. Wu, and J. Wooley, "Ultrafast clustering algorithms for metagenomic sequence analysis," *Briefings Bioinformatics*, vol. 13, no. 6, pp. 656–668, Jul. 2012.

[4] K.-S. Chuang, H.-L. Tzeng, S. Chen, J. Wu, and T.-J. Chen, "Fuzzy c-means clustering with spatial information for image segmentation," *Comput. Med. Imag. Graph.*, vol. 30, no. 1, pp. 9–15, Jan. 2006.

[5] R. J. Kuo, L. M. Ho, and C. M. Hu, "Integration of self-organizing feature map and *K*-means algorithm for market segmentation," *Comput. Oper. Res.*, vol. 29, no. 11, pp. 1475–1493, Sep. 2002.

[6] H. Chen, W. Chung, J. J. Xu, G. Wang, Y. Qin, and M. Chau, "Crime data mining: A general framework and some examples," *Computer*, vol. 37, no. 4, pp. 50–56, Apr. 2004.

[7] D. Centola, "The spread of behavior in an online social network experiment," *Science*, vol. 329, no. 5996, pp. 1194–1197, Sep. 2010.

[8] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Stat. Probability*, Oakland, CA, USA, 1967, pp. 281–297.

[9] G. Tzortzis and A. Likas, "The MinMax *k*-means clustering algorithm," *Pattern Recognit.*, vol. 47, no. 7, pp. 2505–2516, 2014.

[10] J. Qi, Y. Yu, L. Wang, and J. Liu, "K*-means: An effective and efficient K-means clustering algorithm," in *Proc. IEEE Int. Conf. Big Data Cloud Comput. Social Comput. Netw., Sustain. Comput. Commun.*, 2016, pp. 242–249.

[11] H. Jia, Y.-M. Cheung, and J. Liu, "A new distance metric for unsupervised learning of categorical data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 5, pp. 1065–1079, May 2016.

[12] J. Qi, Y. Yu, L. Wang, J. Liu, and Y. Wang, "An effective and efficient hierarchical *K*-means clustering algorithm," *Int. J. Distrib. Sensor Netw.*, vol. 13, no. 8, pp. 1–17, 2017.

[13] S. C. Ahalt, A. K. Krishnamurthy, P. Chen, and D. E. Melton, "Competitive learning algorithms for vector quantization," *Neural Netw.*, vol. 3, no. 3, pp. 277–290, 1990.

[14] L. Xu, A. Krzyzak, and E. Oja, "Rival penalized competitive learning for clustering analysis, RBF net, and curve detection," *IEEE Trans. Neural Netw.*, vol. 4, no. 4, pp. 636–649, Jul. 1993.

[15] L.-T. Law and Y.-M. Cheung, "Color image segmentation using rival penalized controlled competitive learning," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, vol. 1, 2003, pp. 108–112.

[16] Y.-M. Cheung, "On rival penalization controlled competitive learning for clustering with automatic cluster number selection," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 11, pp. 1583–1588, Nov. 2005.

[17] J. Ma and T. Wang, "A cost-function approach to rival penalized competitive learning (RPCL)," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 36, no. 4, pp. 722–737, Aug. 2006.

[18] D. Bacciu and A. Starita, "Competitive repetition suppression (CoRe) clustering: A biologically inspired learning model with application to robust clustering," *IEEE Trans. Neural Netw.*, vol. 19, no. 11, pp. 1922–1941, Nov. 2008.

[19] C.-D. Wang and J.-H. Lai, "Energy based competitive learning," *Neurocomputing*, vol. 74, nos. 12–13, pp. 2265–2275, Jun. 2011.

[20] Y.-M. Cheung, "Maximum weighted likelihood via rival penalized EM for density mixture clustering with automatic model selection," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 6, pp. 750–761, Jun. 2005.

[21] H. Xiong, J. Wu, and J. Chen, "K-means clustering versus validation measures: A data-distribution perspective," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 2, pp. 318–331, Apr. 2009.

[22] J. Liang, L. Bai, C. Dang, and F. Cao, "The *K*-means-type algorithms versus imbalanced data distributions," *IEEE Trans. Fuzzy Syst.*, vol. 20, no. 4, pp. 728–745, Aug. 2012.

[23] Y.-M. Cheung, "K-means: A new generalized *k*-means clustering algorithm," *Pattern Recognit. Lett.*, vol. 24, no. 15, pp. 2883–2893, 2003.

[24] T. C. Silva and L. Zhao, "Stochastic competitive learning in complex networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 3, pp. 385–398, Mar. 2012.

[25] C.-D. Wang, J.-H. Lai, and J.-Y. Zhu, "Graph-based multiprototype competitive learning and its applications," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 6, pp. 934–946, Nov. 2012.

[26] Y.-M. Cheung, "A competitive and cooperative learning approach to robust data clustering," in *Proc. Int. Conf. Neural Netw. Comput. Intell.*, Grindelwald, Switzerland, 2004, pp. 131–136.

[27] H. Jia, Y.-M. Cheung, and J. Liu, "Cooperative and penalized competitive learning with application to kernel-based clustering," *Pattern Recognit.*, vol. 47, no. 9, pp. 3060–3069, Sep. 2014.

[28] C.-D. Wang, J.-H. Lai, C. Y. Suen, and J.-Y. Zhu, "Multi-exemplar affinity propagation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 9, pp. 2223–2237, Sep. 2013.

[29] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Comput.*, vol. 10, no. 5, pp. 1299–1319, Jul. 1998.

[30] A. Ben-Hur, D. Horn, H. T. Siegelmann, and V. Vapnik, "Support vector clustering," *J. Mach. Learn. Res.*, vol. 2, pp. 125–137, Dec. 2001.

[31] C.-D. Wang and J.-H. Lai, "Position regularized support vector domain description," *Pattern Recognit.*, vol. 46, no. 3, pp. 875–884, 2013.

[32] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.

[33] X. Fang, Y. Xu, X. Li, Z. Lai, and W. K. Wong, "Robust semi-supervised subspace clustering via non-negative low-rank representation," *IEEE Trans. Cybern.*, vol. 46, no. 8, pp. 1828–1838, Aug. 2016.

[34] X. Fang, Y. Xu, X. Li, Z. Lai, and W. K. Wong, "Learning a nonnegative sparse graph for linear regression," *IEEE Trans. Image Process.*, vol. 24, no. 9, pp. 2760–2771, Sep. 2015.

[35] X. Fang *et al.*, "Flexible affinity matrix learning for unsupervised and semisupervised classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 4, pp. 1133–1149, Apr. 2019.

[36] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, Jun. 2014.

[37] Y. Qin, Z. L. Yu, C.-D. Wang, Z. Gu, and Y. Li, "A novel clustering method based on hybrid K-nearest-neighbor graph," *Pattern Recognit.*, vol. 74, pp. 1–14, Feb. 2018.

[38] C. Chen, K.-Y. Lin, C.-D. Wang, J.-B. Liu, and D. Huang, "CCMS: A nonlinear clustering method based on crowd movement and selection," *Neurocomputing*, vol. 269, pp. 120–131, Dec. 2017.

[39] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 4, pp. 463–484, Jul. 2012.

[40] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, no. 1, pp. 321–357, Jun. 2002.

[41] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD Explor. Newslett.*, vol. 6, no. 1, pp. 20–29, 2004.

[42] X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory undersampling for class-imbalance learning," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 2, pp. 539–550, Apr. 2009.

[43] X. Hong, S. Chen, and C. J. Harris, "A kernel-based two-class classifier for imbalanced data sets," *IEEE Trans. Neural Netw.*, vol. 18, no. 1, pp. 28–41, Jan. 2007.

[44] B. Raskutti and A. Kowalczyk, "Extreme re-balancing for SVMs: A case study," *ACM SIGKDD Explor. Newslett.*, vol. 6, no. 1, pp. 60–69, Jun. 2004.

[45] C. Elkan, "The foundations of cost-sensitive learning," in *Proc. 17th Int. Joint Conf. Artif. Intell.*, Seattle, WA, USA, 2001, pp. 973–978.

[46] C. X. Ling, V. S. Sheng, and Q. Yang, "Test strategies for cost-sensitive decision trees," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 8, pp. 1055–1067, Aug. 2006.

[47] M. A. Davenport, R. G. Baraniuk, and C. D. Scott, "Tuning support vector machines for minimax and Neyman–Pearson classification," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 32, no. 10, pp. 1888–1898, Oct. 2010.

[48] M. J. Li, M. K. Ng, Y.-M. Cheung, and J. Z. Huang, "Agglomerative fuzzy K-means clustering algorithm with selection of number of clusters," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 11, pp. 1519–1534, Nov. 2008.

[49] L. Bai, J. Liang, and Y. Guo, "An ensemble clusterer of multiple fuzzy *k*-means clusterings to recognize arbitrarily shaped clusters," *IEEE Trans. Fuzzy Syst.*, vol. 26, no. 6, pp. 3524–3533, Dec. 2018.

[50] Y. Wang and L. Chen, "Multi-exemplar based clustering for imbalanced data," in *Proc. 13th Int. Conf. Control Autom. Robot. Vis.*, Singapore, 2014, pp. 1068–1073.

[51] J. Fan, Z. Niu, Y. Liang, and Z. Zhao, "Probability model selection and parameter evolutionary estimation for clustering imbalanced data without sampling," *Neurocomputing*, vol. 211, pp. 172–181, Oct. 2016.

[52] Y. Yang and J. Jiang, "Hybrid sampling-based clustering ensemble with global and local constitutions," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 5, pp. 952–965, May 2016.

[53] C. Aksoylar, J. Qian, and V. Saligrama, "Clustering and community detection with imbalanced clusters," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 3, no. 1, pp. 61–76, Mar. 2017.
[54] M. Liu, X. Jiang, and A. C. Kot, "A multi-prototype clustering algorithm," *Pattern Recognit.*, vol. 42, no. 5, pp. 689–698, May 2009.
[55] T. Luo, C. Zhong, H. Li, and X. Sun, "A multi-prototype clustering algorithm based on minimum spanning tree," in *Proc. 7th Int. Conf. Fuzzy Syst. Knowl. Disc.*, Yantai, China, 2010, pp. 1602–1607.
[56] Y. Liu, Z. Li, H. Xiong, X. Gao, J. Wu, and S. Wu, "Understanding and enhancement of internal clustering validation measures," *IEEE Trans. Cybern.*, vol. 43, no. 3, pp. 982–994, Jun. 2013.
[57] R. Duin. (2007). *PR-Tools4.1, A MATLAB Toolbox for Pattern Recognition*. [Online]. Available: http://prtools.org
[58] A. Asuncion and D. Newman. (2007). *UCI Repository of Machine Learning Databases*. [Online]. Available: http://archive.ics.uci.edu/ml/

Yang Lu (S'13) received the B.S. and M.S. degrees in software engineering from the University of Macau, Macau, China, in 2012 and 2014, respectively. He is currently pursuing the Ph.D. degree in computer science with Hong Kong Baptist University, Hong Kong.

His current research interests include class imbalance learning, imbalance clustering, ensemble learning, and online learning.

Yiu-Ming Cheung (SM'06–F'18) received the Ph.D. degree from the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong.

He is currently a Full Professor with the Department of Computer Science, Hong Kong Baptist University, Hong Kong. His current research interests include machine learning, pattern recognition, visual computing, and optimization.

Dr. Cheung is the Founding Chair of the Computational Intelligence Chapter of the IEEE Hong Kong Section, and the Chair of the Technical Committee on Intelligent Informatics of the IEEE Computer Society. He serves as an Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, the IEEE TRANSACTIONS ON CYBERNETICS, *Pattern Recognition*, *Knowledge and Information Systems*, and *Neurocomputing*. He is a fellow of IET, BCS, and RSA. He is a Distinguished Fellow of IETI.

Yuan Yan Tang (S'88–M'88–SM'96–F'04–LF'16) received the B.Sc. degree in electrical and computer engineering from Chongqing University, Chongqing, China, the M.Eng. degree in electrical engineering from the Beijing Institute of Posts and Telecommunications, Beijing, China, and the Ph.D. degree in computer science from Concordia University, Montreal, QC, Canada.

He is currently a Chair Professor with the Faculty of Science and Technology, University of Macau, Macau, China. He is also a Professor with Chongqing University, an Adjunct Professor with Concordia University, and an Honorary Professor with Hong Kong Baptist University, Hong Kong. He has authored or coauthored over 400 academic papers, over 25 monographs, books, and book chapters. His current research interests include wavelets, pattern recognition, and image processing.

Prof. Tang is the Founder and the Editor-in-Chief of the *International Journal on Wavelets, Multiresolution, and Information Processing*, and an associate editor of several international journals. He is the Founder and the Chair of the Pattern Recognition Committee in the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS. He is the Founder and the General Chair of the series International Conferences on Wavelets Analysis and Pattern Recognition. He is the Founder and the Chair of the Macau Branch of International Association of Pattern Recognition. He was the general chair, the program chair, and a committee member for many international conferences. He is a fellow of IAPR.