Nonlinear Clustering: Methods and Applications

Chang-Dong Wang and Jian-Huang Lai

Abstract As a fundamental classification method for pattern recognition, data clustering plays an important role in various fields such as computer science, medical science, social science, and economics. According to the data distribution of clusters, data clustering problem can be categorized into linearly separable clustering and nonlinearly separable clustering. Due to the complex manifold of the real-world data, nonlinearly separable clustering is one of most popular and widely studied clustering problems. This chapter reviews nonlinear clustering algorithms from four viewpoints, namely kernel-based clustering, multi-exemplar model, graph-based method, and support vector clustering (SVC). Accordingly, this chapter reviews four nonlinear clustering methods, namely conscience on-line learning (COLL) for kernel-based clustering, multi-exemplar affinity propagation (MEAP), graph-based multi-prototype competitive learning (GMPCL), and position regularized support vector clustering (PSVC), and demonstrates their applications in computer vision such as digital image clustering, video segmentation, and color image segmentation.

1 Introduction

Data clustering plays an indispensable role in various fields such as computer science, medical science, social science, and economics [47]. According to the data distribution of clusters, data clustering problem can be categorized into linearly separable clustering and nonlinearly separable clustering. Nonlinearly separable clustering means that the data set to be partitioned contains at least one cluster with concave boundaries or even of arbitrary shapes. Figure 1 illustrates two typical

C.-D. Wang
School of Mobile Information Engineering, Sun Yat-sen University, Zhuhai, People's Republic of China
e-mail: changdongwang@hotmail.com

J.-H. Lai (✉)
School of Information Science and Technology, Sun Yat-sen University, Guangzhou, People's Republic of China
e-mail: stsljh@mail.sysu.edu.cn

M.E. Celebi, K. Aydin (eds.), *Unsupervised Learning Algorithms*,
DOI 10.1007/978-3-319-24211-8_11

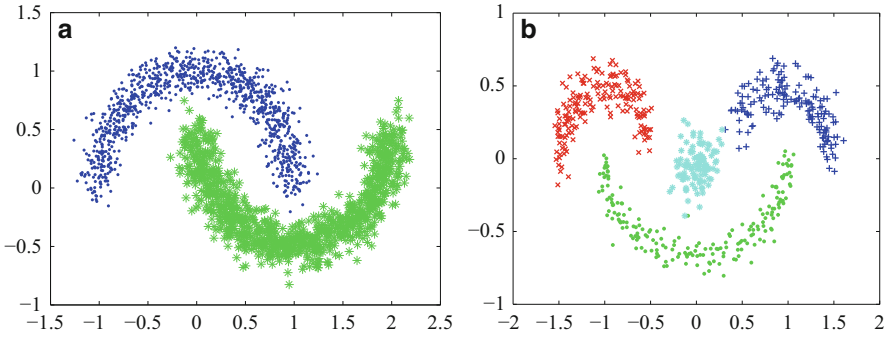


Fig. 1 Two typical data sets consisting of nonlinear separable clusters. (**a**) Two moons data set; (**b**) smile face data set

data sets consisting of nonlinear separable clusters. Due to the complex manifold of the real-world data, it is important to identify the nonlinear clusters. However, classical linear separable clustering methods such as *k*-means cannot meet the need of identifying nonlinear clusters. In this chapter, we will review several nonlinear clustering works from four viewpoints, namely kernel-based clustering, multi-exemplar model, graph-based method and support vector clustering (SVC), and analyze their applications in computer vision such as digital image clustering, video segmentation, and color image segmentation. They are summarized as follows.

1. To address the ill-initialization problem in kernel-based clustering, Sect. 2 first of all reviews a conscience on-line learning (COLL) for kernel-based clustering [40, 44]. Kernel-based clustering is one of the most widely used nonlinear clustering methods. However, classical kernel-based clustering methods, such as kernel *k*-means [31], suffer from one shortcoming that their clustering results seriously degenerate due to ill-initialization [6, 7]. The advantage of the conscience mechanism is that, by decreasing the winning frequency of the frequent winners, all the prototypes would converge to the optimal solution quickly, such that the degeneration problem due to ill-initialization can be successfully avoided. By conducting experiments in video segmentation, we validate the effectiveness of the COLL algorithm in the applications of computer vision.
2. From the viewpoint of multi-exemplar model, Sect. 3 reviews a novel multi-exemplar clustering method, termed multi-exemplar affinity propagation (MEAP) [43]. In the multi-exemplar model, each cluster is represented by an automatically determined number of exemplars and a super-exemplar. Each data point is assigned to the most suitable exemplar and each exemplar is assigned to the most suitable super-exemplar. The model aims to maximize the sum of all similarities between data points and the corresponding exemplars *plus* the sum of all linkages between exemplars and the corresponding super-exemplars. To solve this NP-hard problem, we use the max-sum belief propagation,

producing clusters insensitive to initialization and converged to the neighborhood maximum. Experimental results in natural image categorization validate the effectiveness of MEAP in computer vision application.

3. Based on the advantages of graph method, multi-prototype representation and competitive learning, Sect. 4 reviews a novel nonlinear clustering method, namely graph-based multi-prototype competitive learning (GMPCL) [45]. Based on the graph method, the GMPCL method firstly generates an initial coarse clustering. That is, we first construct a graph with vertex energy vector from the data set; a core-point set is computed according to the vertex energy, from which we can obtain an initial clustering by constructing connected components via neighbor connectivity. Then, a multi-prototype competitive learning is designed to refine this initial clustering so as to generate accurate clustering results. Experimental results in automatic color image segmentation validate the effectiveness of GMPCL and show that GMPCL provides a useful tool for computer vision.
4. To eliminate the influence of the trade-off parameter C to SVC [3], Sect. 5 reviews a position regularized support vector clustering (PSVC) [39]. SVC is a nonlinear clustering method developed from support vector domain description (SVDD). Compared with other clustering methods, the advantages of SVC include the abilities to discover clusters of arbitrary shapes and to deal with outliers. However, the value of the trade-off parameter directly determines the size of sphere and therefore influences the data distribution of sphere surface, i.e., the clustering results of SVC. By assigning each data point with a position based weighting, rather than using the same trade-off parameter for all data points, the PSVC algorithm can adaptively generate a suitable sphere, leading to accurate clustering results.

In the discussion and conclusion section, we will discuss the issues associated with these methods, as well as the relation between these methods and other nonlinear clustering methods.

2 COLL for Kernel-Based Clustering

In this section, we review a COLL for kernel-based clustering [40, 44], which is designed to address the ill-initialization problem in kernel-based clustering. The advantage of the conscience mechanism is that, by decreasing the winning frequency of the frequent winners, all the prototypes would converge to the optimal solution quickly, such that the degeneration problem due to ill-initialization can be successfully avoided. Experimental results in video segmentation have validated the effectiveness of the COLL algorithm in the applications of computer vision.

2.1 Problem Formulation

Given an unlabelled data set $\mathscr{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ of n data points in $\mathbb{R}^d$ which is projected into a kernel space $\mathscr{Y}$ by a mapping ϕ, and the number of clusters c, we wish to find an assignment of each data point to one of c clusters, such that in the kernel space $\mathscr{Y}$, the within-cluster similarity is high and the between-cluster one is low. That is, we seek a map

$$\nu : \mathscr{X} \to \{1, \ldots, c\} \tag{1}$$

to optimize [32]

$$\nu = \arg\min_{\nu} \left\{ \sum_{i,j:\nu_i=\nu_j} \| \phi(\mathbf{x}_i) - \phi(\mathbf{x}_j) \|^2 - \lambda \sum_{i,j:\nu_i\neq\nu_j} \| \phi(\mathbf{x}_i) - \phi(\mathbf{x}_j) \|^2 \right\}, \tag{2}$$

where $\lambda > 0$ is some parameter and we use the short notation $\nu_i = \nu(\mathbf{x}_i)$.

Theorem 1. *The optimization criterion (2) is equivalent to the criterion*

$$\nu = \arg\min_{\nu} \sum_{i=1}^{n} \| \phi(\mathbf{x}_i) - \boldsymbol{\mu}_{\nu_i} \|^2, \tag{3}$$

where $\boldsymbol{\mu}_k$ is the mean of data points assigned to cluster k

$$\boldsymbol{\mu}_k = \frac{1}{|\nu^{-1}(k)|} \sum_{i\in\nu^{-1}(k)} \phi(\mathbf{x}_i), \ \forall k = 1, \ldots, c \tag{4}$$

and ν_i satisfies

$$\nu_i = \arg\min_{k=1,\ldots,c} \| \phi(\mathbf{x}_i) - \boldsymbol{\mu}_k \|^2, \ \forall i = 1, \ldots, n. \tag{5}$$

Proof. See [32]. □

Thus the goal of kernel clustering is to solve the optimization problem in (3). The objective term

$$\sum_{i=1}^{n} \| \phi(\mathbf{x}_i) - \boldsymbol{\mu}_{\nu_i} \|^2 \tag{6}$$

is known as the *distortion error* [4]. Ideally, all possible assignments of the data into clusters should be tested and the best one with smallest *distortion error* selected. This procedure is unfortunately computationally infeasible in even a very small data set, since the number of all possible partitions of a data set grows exponentially with the number of data points. Hence, efficient algorithms are required.

In practice, the mapping function ϕ is often not known or hard to obtain and the dimensionality of $\mathscr{Y}$ is quite high. The feature space $\mathscr{Y}$ is characterized by the kernel function κ and corresponding kernel matrix K [32].

Definition 1. A kernel is a function κ, such that $\kappa(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle$ for all $\mathbf{x}, \mathbf{z} \in \mathscr{X}$, where ϕ is a mapping from $\mathscr{X}$ to an (inner product) feature space $\mathscr{Y}$. A kernel matrix is a square matrix $K \in \mathbb{R}^{n \times n}$ such that $K_{i,j} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ for some $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathscr{X}$ and some kernel function κ.

Thus for an efficient approach, the computation procedure using only the kernel matrix is also required.

2.2 Batch Kernel k-Means and Issues

The k-means [26] algorithm is one of most popular iterative methods for solving the optimization problem (3). It begins by initializing a random assignment ν and seeks to minimize the *distortion error* by iteratively updating the assignment ν

$$\nu_i \leftarrow \arg \min_{k=1,\ldots,c} \| \phi(\mathbf{x}_i) - \boldsymbol{\mu}_k \|^2, \ \forall i = 1, \ldots, n \tag{7}$$

and the prototypes $\boldsymbol{\mu}$

$$\boldsymbol{\mu}_k \leftarrow \frac{1}{|\nu^{-1}(k)|} \sum_{i \in \nu^{-1}(k)} \phi(\mathbf{x}_i), \ \forall k = 1, \ldots, c, \tag{8}$$

until all prototypes converge or the number of iterations reaches a prespecified value $t_{\max}$. Let $\hat{\boldsymbol{\mu}}$ denote the old prototypes before the t-th iteration, the convergence of all prototypes is characterized by the *convergence criterion*

$$\mathfrak{e}^{\phi} = \sum_{k=1}^{c} \| \boldsymbol{\mu}_k - \hat{\boldsymbol{\mu}}_k \|^2 \leq \epsilon, \tag{9}$$

where ϵ is some very small positive value, e.g., 10^{-4}.

Since in practice, only the kernel matrix K is available, the updating of assignment (7) is computed based on the kernel trick [30]

$$\begin{aligned} \nu_i &\leftarrow \arg \min_{k=1,\ldots,c} \| \phi(\mathbf{x}_i) - \boldsymbol{\mu}_k \|^2 \\ &= \arg \min_{k=1,\ldots,c} \left\{ K_{i,i} + \frac{\sum_{h \in \nu^{-1}(k)} \sum_{l \in \nu^{-1}(k)} K_{h,l}}{|\nu^{-1}(k)|^2} - \frac{2 \sum_{j \in \nu^{-1}(k)} K_{i,j}}{|\nu^{-1}(k)|} \right\}. \end{aligned} \tag{10}$$

Then the updated prototypes $\boldsymbol{\mu}$ are implicitly expressed by the assignment ν, which is further used in the next iteration. Let $\hat{\nu}$ denote the old assignment before the t-th iteration, the *convergence criterion* (9) is computed as

$$\begin{aligned}\mathfrak{e}^{\phi} &= \sum_{k=1}^{c} \| \boldsymbol{\mu}_k - \hat{\boldsymbol{\mu}}_k \|^2 \\ &= \sum_{k=1}^{c} \left\{ \frac{\sum_{h\in\nu^{-1}(k)} \sum_{l\in\nu^{-1}(k)} K_{h,l}}{|\nu^{-1}(k)|^2} \right. \\ &\quad \left. + \frac{\sum_{h\in\hat{\nu}^{-1}(k)} \sum_{l\in\hat{\nu}^{-1}(k)} K_{h,l}}{|\hat{\nu}^{-1}(k)|^2} - \frac{2\sum_{h\in\nu^{-1}(k)} \sum_{l\in\hat{\nu}^{-1}(k)} K_{h,l}}{|\nu^{-1}(k)||\hat{\nu}^{-1}(k)|} \right\}. \end{aligned} \tag{11}$$

The above procedures lead to the well-known kernel k-means [31]. Given the appropriate initialization, it can find the sub-optimal solution (local minima).

Despite great success in practice, it is quite sensitive to the initial positions of cluster prototypes, leading to degenerate local minima.

Example 1. In Fig. 2b, $\boldsymbol{\mu}_2$ is ill-initialized relatively far away from any points such that it will get no chance to be assigned with points and updated in the iterations of k-means. As a result, the clustering result by k-means is trapped in degenerate local minima as shown in Fig. 2c, where $\boldsymbol{\mu}_2$ is assigned with no point.

To overcome this problem, current k-means algorithm and its variants usually run many times with different initial prototypes and select the result with the smallest *distortion error* (6) [36], which are, however, computationally too expensive. The COLL method is an efficient and effective approach to the optimization problem (3), which can output smaller *distortion error* that is insensitive to the initialization.

2.3 Conscience On-Line Learning

In this section, we will introduce COLL to overcome the above issue.

2.3.1 The COLL Model

Let n_k denote the cumulative winning number of the k-th prototype, and $f_k = n_k / \sum_{l=1}^{c} n_l$ the corresponding winning frequency. In the beginning, they are initialized as

$$n_k = |\nu^{-1}(k)|, f_k = n_k/n, \forall k = 1, \ldots, c. \tag{12}$$

The *COLL* is performed as follows: initialize the same random assignment ν as k-means, and iteratively update the assignment ν and the prototypes $\boldsymbol{\mu}$ based on the *frequency sensitive (conscience) on-line learning* rule. That is, in the t-th iteration, for one randomly taken data point $\phi(\mathbf{x}_i)$, *select* the winning prototype $\boldsymbol{\mu}_{\nu_i}$ guided by the winning frequency f_k, i.e.,

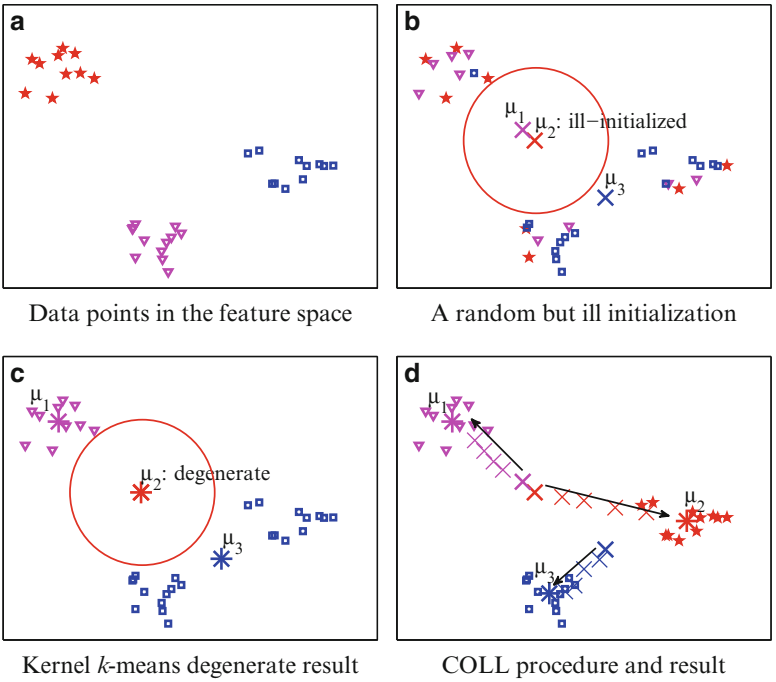


Fig. 2 Illustration of ill-initialization and comparison of kernel k-means and COLL. Different clusters are plotted in *different colors* and *marks*, and the initial prototypes are plotted in "×" while the final in "*". (**a**) Data points in the feature space. (**b**) A random but ill initialization. Each prototype is computed as the mean of points randomly assigned to the corresponding cluster. $\boldsymbol{\mu}_2$ is ill-initialized, i.e., it is relatively far away from any points. (**c**) The degenerate result by kernel k-means. The ill-initialization makes the final $\boldsymbol{\mu}_2$ assigned with no point. (**d**) The procedure and result of COLL. The updating procedure of prototypes is plotted in thinner "×". The *conscience* mechanism successfully makes $\boldsymbol{\mu}_2$ win in the iterations, leading to a satisfying result

conscience based winner selection rule:

$$v_i = \arg \min_{k=1,\ldots,c} \{f_k \parallel \phi(\mathbf{x}_i) - \boldsymbol{\mu}_k \parallel^2\}, \tag{13}$$

and *update* the winner $\boldsymbol{\mu}_{v_i}$ with learning rate η_t, i.e.,
on-line winner updating rule:

$$\boldsymbol{\mu}_{v_i} \leftarrow \boldsymbol{\mu}_{v_i} + \eta_t \left(\phi(\mathbf{x}_i) - \boldsymbol{\mu}_{v_i}\right), \tag{14}$$

as well as update the **winning frequency**

$$n_{v_i} \leftarrow n_{v_i} + 1, f_k = n_k \Big/ \sum_{l=1}^{c} n_l, \forall k = 1, \ldots, c. \tag{15}$$

The iteration procedure continues until all prototypes converge or the number of iterations reaches a prespecified value $t_{\max}$. The *convergence criterion* identical to that of k-means (9) is used. The learning rates $\{\eta_t\}$ satisfy conditions [4]:

$$\lim_{t\to\infty} \eta_t = 0, \qquad \sum_{t=1}^{\infty} \eta_t = \infty, \qquad \sum_{t=1}^{\infty} \eta_t^2 < \infty. \tag{16}$$

In practice, $\eta_t = const/t$, where *const* is some small constant, e.g., 1.

By reducing the winning rate of the frequent winners according to (13), the goal of the *conscience* mechanism is to bring all the prototypes available into the solution quickly and to bias the competitive process so that each prototype can win the competition with almost the same probability. In this way, it is insensitive to the ill-initialization, and thus prevents the result from being trapped into degenerate local minima; meanwhile converges much faster than other iterative methods [8].

Example 2. The same ill-initialization as kernel k-means in Example 1 is used by COLL. However, since the winning frequency of the ill-initialized $\boldsymbol{\mu}_2$ will become smaller in the later iterations, it gets the chance to win according to (13). Finally, an almost perfect clustering with small distortion error is obtained as shown in Fig. 2d.

2.3.2 The Computation of COLL

As discussed before, any effective algorithm in the kernel space must compute with only the kernel matrix K. To this end, we devise an efficient framework for computation of COLL based on a novel representation of prototypes termed *prototype descriptor*.

Let $\bar{\mathbb{R}}^{c\times(n+1)}$ denote the $c \times (n+1)$ matrix space satisfying $\forall A \in \bar{\mathbb{R}}^{c\times(n+1)}$, $A_{:,n+1} \geq 0$, i.e., the last column of matrix A is nonnegative. We define the *prototype descriptor* based on *kernel trick* as follows.

Definition 2 (Prototype Descriptor). A prototype descriptor is a matrix $W^{\phi} \in \bar{\mathbb{R}}^{c\times(n+1)}$, such that the k-th row represents prototype $\boldsymbol{\mu}_k$ by

$$W^{\phi}_{k,i} = \langle \boldsymbol{\mu}_k, \phi(\mathbf{x}_i)\rangle, \forall i = 1,\ldots,n,\ W^{\phi}_{k,n+1} = \langle \boldsymbol{\mu}_k, \boldsymbol{\mu}_k\rangle, \tag{17}$$

i.e.,

$$W^{\phi} = \left(\begin{array}{ccc|c} \langle \boldsymbol{\mu}_1, \phi(\mathbf{x}_1)\rangle & \ldots & \langle \boldsymbol{\mu}_1, \phi(\mathbf{x}_n)\rangle & \langle \boldsymbol{\mu}_1, \boldsymbol{\mu}_1\rangle \\ \langle \boldsymbol{\mu}_2, \phi(\mathbf{x}_1)\rangle & \ldots & \langle \boldsymbol{\mu}_2, \phi(\mathbf{x}_n)\rangle & \langle \boldsymbol{\mu}_2, \boldsymbol{\mu}_2\rangle \\ \vdots & \vdots & \vdots & \vdots \\ \langle \boldsymbol{\mu}_c, \phi(\mathbf{x}_1)\rangle & \ldots & \langle \boldsymbol{\mu}_c, \phi(\mathbf{x}_n)\rangle & \langle \boldsymbol{\mu}_c, \boldsymbol{\mu}_c\rangle \end{array}\right). \tag{18}$$

With this definition, the computation of the *distortion error* (6) now becomes:

$$\sum_{i=1}^{n} \| \phi(\mathbf{x}_i) - \boldsymbol{\mu}_{\nu_i} \|^2 = \sum_{i=1}^{n} \left(K_{i,i} + W^{\phi}_{\nu_i,n+1} - 2W^{\phi}_{\nu_i,i} \right). \tag{19}$$

Let's consider the computation of four ingredients of the COLL model.

Theorem 2 (Initialization). *The random initialization can be realized in the way of*

$$W^{\phi}_{:,1:n} = AK, \qquad W^{\phi}_{:,n+1} = diag(AKA^{\top}) \tag{20}$$

where diag(M) *denotes the main diagonal of a matrix* M *and the positive matrix* $A = [A_{k,i}]^{c\times n} \in R_{+}^{c\times n}$ *has the form*

$$A_{k,i} = \begin{cases} \frac{1}{|\nu^{-1}(k)|} & \text{if } i \in \nu^{-1}(k) \\ 0 & \text{otherwise.} \end{cases} \tag{21}$$

That is, the matrix A *reflects the initial assignment* ν.

Proof. Assume the assignment is randomly initialized as ν. Substitute the computation of the prototypes (4) to the definition of $W^{\phi}_{k,:}$ in (17), we get

$$\begin{aligned} W^{\phi}_{k,i} &= \langle \boldsymbol{\mu}_k, \phi(\mathbf{x}_i) \rangle, \forall i = 1, \ldots, n \\ &= \left\langle \frac{\sum_{j\in\nu^{-1}(k)} \phi(\mathbf{x}_j)}{|\nu^{-1}(k)|}, \phi(\mathbf{x}_i) \right\rangle \\ &= \sum_{j\in\nu^{-1}(k)} \frac{1}{|\nu^{-1}(k)|} K_{j,i} \\ &= \sum_{j=1}^{n} A_{k,j} K_{j,i} \\ &= A_{k,:} K_{:,i} \\ W^{\phi}_{k,n+1} &= \langle \boldsymbol{\mu}_k, \boldsymbol{\mu}_k \rangle \\ &= \left\langle \frac{\sum_{h\in\nu^{-1}(k)} \phi(\mathbf{x}_h)}{|\nu^{-1}(k)|}, \frac{\sum_{l\in\nu^{-1}(k)} \phi(\mathbf{x}_l)}{|\nu^{-1}(k)|} \right\rangle \\ &= \sum_{h\in\nu^{-1}(k)} \sum_{l\in\nu^{-1}(k)} \frac{1}{|\nu^{-1}(k)|^2} K_{h,l} \end{aligned} \tag{22}$$

$$= \sum_{h=1}^{n}\sum_{l=1}^{n} A_{k,h}A_{k,l}K_{h,l}$$
$$= A_{k,:}KA_{k,:}^{\top}. \tag{23}$$

Thus we obtain the initialization of W^{ϕ} as (20). The proof is finished. □

Theorem 3 (Conscience Based Winner Selection Rule). *The conscience based winner selection rule (13) can be realized in the way of*

$$v_i = \arg\min_{k=1,\ldots,c}\{f_k \cdot (K_{i,i} + W^{\phi}_{k,n+1} - 2W^{\phi}_{k,i})\}. \tag{24}$$

Proof. Consider the winner *selection* rule, i.e., (13), one can get

$$v_i = \arg\min_{k=1,\ldots,c}\{f_k \parallel \phi(\mathbf{x}_i) - \boldsymbol{\mu}_k \parallel^2\}$$
$$= \arg\min_{k=1,\ldots,c}\{f_k \cdot (K_{i,i} + W^{\phi}_{k,n+1} - 2W^{\phi}_{k,i})\}. \tag{25}$$

Thus we get the formula required. □

Theorem 4 (On-Line Winner Updating Rule). *The on-line winner updating rule (14) can be realized in the way of*

$$W^{\phi}_{v_i,j} \leftarrow \begin{cases} (1-\eta_t)W^{\phi}_{v_i,j} + \eta_t K_{i,j} & j = 1,\ldots,n, \\ (1-\eta_t)^2 W^{\phi}_{v_i,j} + \eta_t^2 K_{i,i} + 2(1-\eta_t)\eta_t W^{\phi}_{v_i,i} & j = n+1. \end{cases} \tag{26}$$

Proof. Although we do not know exactly the expression of $\boldsymbol{\mu}_{v_i}$, however we can simply take $\boldsymbol{\mu}_{v_i}$ as a symbol of this prototype and denote its updated one as $\acute{\boldsymbol{\mu}}_{v_i}$. Substitute the on-line winner *updating* rule (14) to the winning prototype $W^{\phi}_{v_i,:}$, we have

$$W^{\phi}_{v_i,j} \leftarrow \langle \acute{\boldsymbol{\mu}}_{v_i}, \phi(\mathbf{x}_j)\rangle \quad \forall j = 1,\ldots,n$$
$$= \langle \boldsymbol{\mu}_{v_i} + \eta_t(\phi(\mathbf{x}_i) - \boldsymbol{\mu}_{v_i}), \phi(\mathbf{x}_j)\rangle$$
$$= (1-\eta_t)\langle \boldsymbol{\mu}_{v_i}, \phi(\mathbf{x}_j)\rangle + \eta_t\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle, \tag{27}$$
$$W^{\phi}_{v_i,j} \leftarrow \langle \acute{\boldsymbol{\mu}}_{v_i}, \acute{\boldsymbol{\mu}}_{v_i}\rangle \quad j = n+1$$
$$= \langle \boldsymbol{\mu}_{v_i} + \eta_t(\phi(\mathbf{x}_i) - \boldsymbol{\mu}_{v_i}), \boldsymbol{\mu}_{v_i} + \eta_t(\phi(\mathbf{x}_i) - \boldsymbol{\mu}_{v_i})\rangle$$
$$= (1-\eta_t)^2\langle \boldsymbol{\mu}_{v_i}, \boldsymbol{\mu}_{v_i}\rangle + \eta_t^2\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_i)\rangle + 2(1-\eta_t)\eta_t\langle \boldsymbol{\mu}_{v_i}, \phi(\mathbf{x}_i)\rangle. \tag{28}$$

Then we get the on-line winner *updating* rule as (26). □

It is a bit complicated to compute the *convergence criterion* without explicit expression of $\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_c\}$. Notice that, in **one** iteration, each point $\phi(\mathbf{x}_i)$ is assigned to **one and only one** winning prototype. Let array $\pi^k = [\pi_1^k, \pi_2^k, \ldots, \pi_{m_k}^k]$ store the indices of m_k *ordered* points assigned to the k-th prototype in **one** iteration. For instance, if $\phi(\mathbf{x}_1), \phi(\mathbf{x}_{32}), \phi(\mathbf{x}_8), \phi(\mathbf{x}_{20}), \phi(\mathbf{x}_{15})$ are 5 *ordered* points assigned to the 2-nd prototype in the t-th iteration, then the index array of the 2-nd prototype is $\pi^2 = [\pi_1^2, \pi_2^2, \ldots, \pi_{m_2}^2] = [1, 32, 8, 20, 15]$ with $\pi_1^2 = 1, \pi_2^2 = 32, \pi_3^2 = 8, \pi_4^2 = 20, \pi_5^2 = 15$ and $m_2 = 5$. The following lemma formulates the cumulative update of the k-th prototype based on the array $\pi^k = [\pi_1^k, \pi_2^k, \ldots, \pi_{m_k}^k]$.

Lemma 1. *In the t-th iteration, the relationship between the updated prototype* $\boldsymbol{\mu}_k$ *and the old* $\hat{\boldsymbol{\mu}}_k$ *is:*

$$\boldsymbol{\mu}_k = (1-\eta_t)^{m_k}\hat{\boldsymbol{\mu}}_k + \sum_{l=1}^{m_k}(1-\eta_t)^{m_k-l}\eta_t\phi(\mathbf{x}_{\pi_l^k}), \tag{29}$$

where array $\pi^k = [\pi_1^k, \pi_2^k, \ldots, \pi_{m_k}^k]$ *stores the indices of* m_k *ordered points assigned to the k-th prototype in this iteration.*

Proof. To prove this relationship, we use the Principle of Mathematical Induction. One can easily verify that (29) is true for $m_k = 1$ directly from (14),

$$\begin{aligned}\boldsymbol{\mu}_k &= \hat{\boldsymbol{\mu}}_k + \eta_t\left(\phi(\mathbf{x}_{\pi_1^k}) - \hat{\boldsymbol{\mu}}_k\right)\\ &= (1-\eta_t)^1\hat{\boldsymbol{\mu}}_k + \sum_{l=1}^{1}(1-\eta_t)^0\eta_t\phi(\mathbf{x}_{\pi_l^k}).\end{aligned} \tag{30}$$

Assume that it is true for $m_k = m$, that is, for the first m *ordered* points,

$$\boldsymbol{\mu}_k = (1-\eta_t)^m\hat{\boldsymbol{\mu}}_k + \sum_{l=1}^{m}(1-\eta_t)^{m-l}\eta_t\phi(\mathbf{x}_{\pi_l^k}). \tag{31}$$

Then for $m_k = m+1$, i.e., the $(m+1)$-th point, from (14) we have

$$\begin{aligned}\boldsymbol{\mu}_k &= \boldsymbol{\mu}_k + \eta_t\left(\phi(\mathbf{x}_{\pi_{m_k}^k}) - \boldsymbol{\mu}_k\right)\\ &= (1-\eta_t)\boldsymbol{\mu}_k + \eta_t\phi(\mathbf{x}_{\pi_{m_k}^k})\\ &= (1-\eta_t)\left((1-\eta_t)^m\hat{\boldsymbol{\mu}}_k + \sum_{l=1}^{m}(1-\eta_t)^{m-l}\eta_t\phi(\mathbf{x}_{\pi_l^k})\right) + \eta_t\phi(\mathbf{x}_{\pi_{m_k}^k})\\ &= (1-\eta_t)^{m+1}\hat{\boldsymbol{\mu}}_k + \sum_{l=1}^{m+1}(1-\eta_t)^{m+1-l}\eta_t\phi(\mathbf{x}_{\pi_l^k}).\end{aligned} \tag{32}$$

This expression shows that (29) is true for $m_k = m+1$. Therefore, by mathematical induction, it is true for all positive integers m_k. □

Theorem 5 (Convergence Criterion). *The convergence criterion can be computed by*

$$\mathfrak{e}^{\phi} = \sum_{k=1}^{c}\left(\left(1-\frac{1}{(1-\eta_t)^{m_k}}\right)^2 W^{\phi}_{k,n+1}\right) + \eta_t^2 \sum_{k=1}^{c}\sum_{h=1}^{m_k}\sum_{l=1}^{m_k}\frac{K_{\pi_h^k,\pi_l^k}}{(1-\eta_t)^{h+l}}$$
$$+2\eta_t \sum_{k=1}^{c}\left(\left(1-\frac{1}{(1-\eta_t)^{m_k}}\right)\sum_{l=1}^{m_k}\frac{W^{\phi}_{k,\pi_l^k}}{(1-\eta_t)^{l}}\right). \tag{33}$$

Proof. According to Lemma 1, the old $\hat{\boldsymbol{\mu}}_k$ can be retained from the updated $\boldsymbol{\mu}_k$ as

$$\hat{\boldsymbol{\mu}}_k = \frac{\boldsymbol{\mu}_k}{(1-\eta_t)^{m_k}} - \eta_t \sum_{l=1}^{m_k}\frac{\phi(\mathbf{x}_{\pi_l^k})}{(1-\eta_t)^{l}}. \tag{34}$$

Substitute it to $\mathfrak{e}^{\phi} = \sum_{k=1}^{c} \| \boldsymbol{\mu}_k - \hat{\boldsymbol{\mu}}_k \|^2$, we have

$$\begin{aligned}\mathfrak{e}^{\phi} &= \sum_{k=1}^{c} \| \boldsymbol{\mu}_k - \left(\frac{\boldsymbol{\mu}_k}{(1-\eta_t)^{m_k}} - \eta_t \sum_{l=1}^{m_k}\frac{\phi(\mathbf{x}_{\pi_l^k})}{(1-\eta_t)^{l}}\right) \|^2 \\ &= \sum_{k=1}^{c}\left(\left(1-\frac{1}{(1-\eta_t)^{m_k}}\right)^2 \langle \boldsymbol{\mu}_k, \boldsymbol{\mu}_k \rangle\right) + \eta_t^2 \sum_{k=1}^{c}\sum_{h=1}^{m_k}\sum_{l=1}^{m_k}\frac{\langle \phi(\mathbf{x}_{\pi_h^k}), \phi(\mathbf{x}_{\pi_l^k}) \rangle}{(1-\eta_t)^{h+l}} \\ &\quad +2\eta_t \sum_{k=1}^{c}\left(\left(1-\frac{1}{(1-\eta_t)^{m_k}}\right)\sum_{l=1}^{m_k}\frac{\langle \boldsymbol{\mu}_k, \phi(\mathbf{x}_{\pi_l^k}) \rangle}{(1-\eta_t)^{l}}\right).\end{aligned} \tag{35}$$

Thus $\mathfrak{e}^{\phi}$ can be computed by (33). This ends the proof. □

For clarification, Algorithm 1 summaries the *COLL* method. Figure 3 compares the clustering results generated by kernel k-means and COLL on one nonlinearly separable data set.

2.3.3 Computational Complexity

The computation of the COLL method consists of two parts: initialization of W^{ϕ} and iterations to update W^{ϕ}. From (20), the initialization of W^{ϕ} takes $O\left(cn^2\right)$ operations. For each iteration, the computational complexity is $O\left(n(c+(n+1))+(c+n^2+n)\right)$. Since $O\left(n(c+(n+1))\right)$ operations are needed to perform the iteration (for each point, $O(c)$ to select a winner and $O(n+1)$ to update the winner, there being n points) and $O\left(c+n^2+n\right)$ operations are needed to compute the *convergence criterion* $\mathfrak{e}^{\phi}$ (the first term of (33) taking $O(c)$ operations, the second term at most $O(n^2)$ operations, and the third term $O(n)$ operations). Assume the iteration number is $t_{\max}$, since

Algorithm 1: Conscience on-line learning (COLL)

Input: kernel matrix $K \in \mathbb{R}^{n\times n}$, c, $\{\eta_t\}$, ϵ, t_{max}.
Output: cluster assignment ν subject to (3).
Randomly initialize assignment ν and set $t = 0$;
Initialize the winning frequency $\{f_k\}$ by (12);
Initialize *prototype descriptor* $W^\phi \in \mathbb{R}^{c\times(n+1)}$ by (20).
repeat
  Get c empty index arrays $\{\pi^k = \emptyset : k = 1, \ldots, c\}$ and a random permutation $\{I_1, \ldots, I_n : I_i \in \{1, \ldots, n\},$ s.t. $I_i \neq I_j, \forall i \neq j\}$, set $t = t + 1$.
  for $l = 1, \ldots, n$ **do**
    Select the winning prototype $W^\phi_{\nu_i,:}$ of the i-th point ($i = I_l$) by (24), and append i to the ν_i-th index array π^{ν_i}.
    Update the winning prototype $W^\phi_{\nu_i,:}$ with learning rate η_t by (26) and the winning frequency by (15).
  end for
  Compute $\mathfrak{e}^\phi$ via (33).
until $\mathfrak{e}^\phi \leq \epsilon$ or $t \geq t_{max}$
Obtain the cluster assignment ν_i by (24), $\forall i = 1, \ldots, n$.

in general $1 < c < n$, the computational complexity for iteration procedure is $O\left(t_{\max}\left(n(c + (n + 1)) + (c + n^2 + n)\right)\right) = O\left(t_{\max}n^2\right)$. Consequently the total computational complexity of the COLL method is $O\left(cn^2 + t_{\max}n^2\right) = O\left(\max(c, t_{\max})n^2\right)$, which is the same as that of kernel k-means if the same number of iterations is obtained. However, due to the *conscience* mechanism [8] and on-line learning rule [4], the COLL achieves faster convergence rate than its counterpart. Thus fewer iterations are needed for COLL to converge. This is especially beneficial in large-scale data clustering.

2.4 Experiments and Applications

In this section, we report the experimental results in the application of video clustering (automatic segmentation). Video clustering plays an important role in automatic video summarization/abstraction as a preprocessing step [35]. Consider a video sequence in which the camera is fading/switching/cutting among a number of scenes, the goal of automatic video clustering is to cluster the video frames according to the different scenes. The gray-scale values of the raw pixels were used as the feature vector for each frame. For one video sequence, the frames $\{\mathbf{f}_i \in \mathbb{R}^d\}_{i=1}^n$ are taken as the data set, where $d = \text{width} \times \text{height}$ and n is the length of the video sequence. We selected the 11 video sequences from the open-video website [16], which are 11 segments of the whole "NASA 25th Anniversary Show" with $d = 320 \times 240 = 76800$ and n (i.e., the duration of the sequence) varying from one sequence to another.

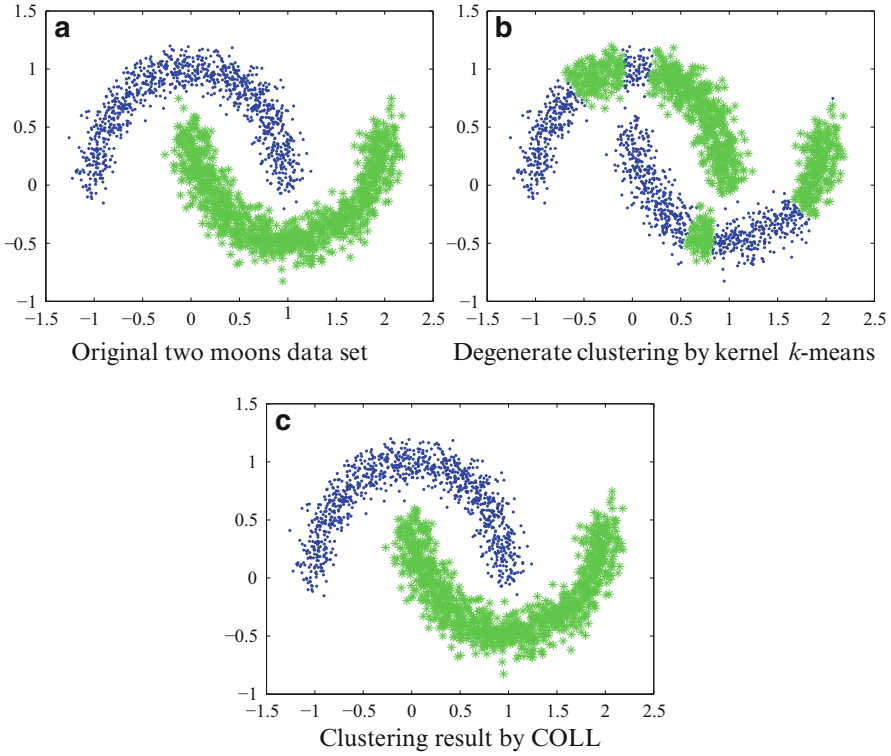


Fig. 3 Original two moons data set and clustering results by kernel k-means and the COLL method. (**a**) Original two moons data set. (**b**) Kernel k-means result. It is obvious that degenerate local optimum is obtained due to the sensitivity to ill-initialization. (**c**) Clustering result by COLL with the same initialization. An almost perfect clustering is obtained with only a few points being erroneously assigned

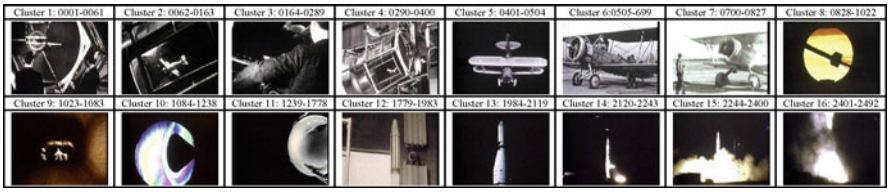


Fig. 4 Clustering frames of the video sequence ANNI002 into 16 scenes using COLL. A satisfying segmentation has been achieved

Figure 4 illustrates the clustering result of one video sequence "NASA 25th Anniversary Show, Segment 2" (ANNI002) by COLL. 2492 frames have been clustered into 16 scenes. Except for the frames from 400 to 405 and 694 to 701 as well as the last two clusters, where the separation boundaries are not so clear, satisfactory segmentation has been obtained. For comparison, "ground truth"

Table 1 The means (running 10 times) of NMI and computational time in seconds on the 11 video sequences, The bold values denote the best results

	Kernel *k*-means [31]		Global kernel *k*-means [36]		COLL	
Video sequence (#frames)	NMI	Time	NMI	Time	NMI	Time
ANNI001 (914)	0.781	72.2	0.801	94.0	**0.851**	**70.4**
ANNI002 (2492)	0.705	94.7	0.721	126.4	**0.741**	**89.0**
ANNI003 (4265)	0.712	102.2	0.739	139.2	**0.762**	**99.5**
ANNI004 (3897)	0.731	98.3	0.750	121.6	**0.759**	**93.6**
ANNI005 (11361)	0.645	152.2	0.656	173.3	**0.680**	**141.2**
ANNI006 (16588)	0.622	193.0	0.638	255.5	**0.642**	**182.3**
ANNI007 (1588)	0.727	81.1	0.740	136.7	**0.770**	**79.1**
ANNI008 (2773)	0.749	95.9	0.771	119.0	**0.794**	**81.5**
ANNI009 (12304)	0.727	167.0	0.763	184.4	**0.781**	**160.4**
ANNI010 (30363)	0.661	257.2	0.709	426.4	**0.734**	**249.0**
ANNI011 (1987)	0.738	85.4	0.749	142.7	**0.785**	**83.7**

#frames denote the number of frames of each video sequence

segmentation of each video sequence has been manually obtained, through which NMI values are computed to compare COLL with kernel *k*-means and global kernel *k*-means. Table 1 lists the average values (running 10 times) of NMI and computational time in seconds on the 11 video sequences. Additionally, the length of each video sequence (i.e., number of the frames) is also listed. The results in terms of average values of NMI and computational time reveal that COLL generates the best segmentation among the compared methods with the least computational time, which is a significant improvement.

3 Multi-exemplar Affinity Propagation

Multi-exemplar is another method for solving nonlinear clustering problem. From the viewpoint of multi-exemplar model, this section reviews a novel multi-exemplar clustering method, termed MEAP [43]. In the multi-exemplar model, each cluster is represented by an automatically determined number of exemplars and a super-exemplar. Each data point is assigned to the most suitable exemplar and each exemplar is assigned to the most suitable super-exemplar. The model aims to maximize the sum of all similarities between data points and the corresponding exemplars *plus* the sum of all linkages between exemplars and the corresponding super-exemplars. To solve this NP-hard problem, we use the max-sum belief propagation, producing clusters insensitive to initialization and converged to the neighborhood maximum. Experimental results in natural image categorization validate the effectiveness of MEAP in computer vision application.

3.1 Affinity Propagation

Affinity propagation is a single-exemplar clustering algorithm using max-sum (max-product) belief propagation to obtain good exemplars. Given a user-defined similarity matrix $[s_{ij}]_{N\times N}$ of N points, it aims at searching for a valid configuration of labels $\mathbf{c} = [c_1, \ldots, c_N]$ to maximize the following objective function [10],

$$\mathscr{S}(\mathbf{c}) = \sum_{i=1}^{N} s_{ic_i} + \sum_{k=1}^{N} \delta_k(\mathbf{c}), \tag{36}$$

where $\delta_k(\mathbf{c})$ is an *exemplar-consistency* constraint such that if some data point i has selected k as its exemplar, i.e., $c_i = k$, then data point k must select itself as an exemplar, i.e., $c_k = k$,

$$\delta_k(\mathbf{c}) = \begin{cases} -\infty & \text{if } c_k \neq k \text{ but } \exists i : c_i = k \\ 0 & \text{otherwise.} \end{cases} \tag{37}$$

AP is an optimized max-sum belief propagation algorithm over the factor graph in Fig. 5. It begins by simultaneously considering all data points as potential exemplars, and recursively transmits real-valued messages between data points, until high-quality exemplars emerge. There are two kinds of messages, which are $r(i,k)$, sent from point i to the candidate exemplar k, reflecting the accumulated evidence for how well-suited point k is to serve as the exemplar for point i, and $a(i,k)$, sent from the candidate exemplar k to point i, reflecting the accumulated evidence for how appropriate it would be for point i to choose point k as its exemplar (Fig. 5). They are initialized as zero, and updated, respectively, as follows:

$$r(i,k) \leftarrow s_{ik} - \max_{j\neq k}[s_{ij} + a(i,j)] \tag{38}$$

$$a(i,k) \leftarrow \min\Big[0, r(k,k) + \sum_{i'\notin\{i,k\}} \max[0, r(i',k)]\Big], k \neq i \tag{39}$$

$$a(k,k) \leftarrow \sum_{i'\neq k} \max[0, r(i',k)]. \tag{40}$$

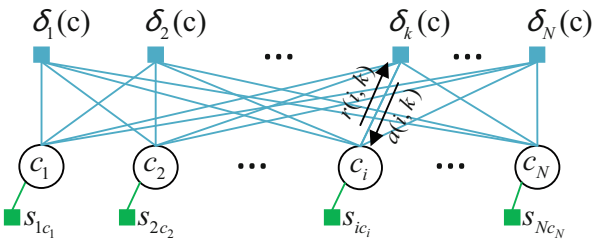


Fig. 5 Factor graph of the AP method and its messages

The assignment vector $\mathbf{c} = [c_1, \ldots, c_N]$ is computed as $c_i = \arg\max_j [a(i,j) + r(i,j)]$ after convergence.

3.2 *Multi-exemplar Affinity Propagation*

Let $[s_{ij}]_{N\times N}$ be a user-defined similarity matrix with s_{ij} measuring the similarity between point i and the potential exemplar j, and $[l_{ij}]_{N\times N}$ a linkage matrix with l_{ij} measuring the linkage between exemplar i and its potential super-exemplar j. We develop a multi-exemplar model that seeks two mappings, $\psi_1 : \{1, \ldots, N\} \to \{1, \ldots, N\}$ assigning data point i to exemplar $\psi_1(i)$, and $\psi_2 : \{\psi_1(1), \ldots, \psi_1(N)\} \to \{\psi_1(1), \ldots, \psi_1(N)\}$ assigning exemplar $\psi_1(i)$ to super-exemplar $\psi_2(\psi_1(i)) = (\psi_2 \circ \psi_1)(i)$, where $\circ$ denotes the function composition. The goal is to maximize the sum $\mathscr{S}_1$ of all similarities between data points and the corresponding exemplars *plus* the sum $\mathscr{S}_2$ of all linkages between exemplars and the corresponding super-exemplars.

3.2.1 The Model

Let $C = [c_{ij}]_{N\times N}$ be an assignment matrix, where the non-diagonal elements $c_{ij} \in \{0, 1\} (j \neq i)$ denote that point j is the exemplar of point i if $c_{ij} = 1$, i.e. $\psi_1(i) = j$, and the diagonal elements $c_{ii} \in \{0, \ldots, N\}$ denote that exemplar c_{ii} is the super-exemplar of exemplar i if $c_{ii} \in \{1, \ldots, N\}$, i.e. $\psi_2(i) = c_{ii}$,

$$c_{ij} = \begin{cases} 1 & \text{if } j \text{ is an exemplar of } i \\ 0 & \text{otherwise} \end{cases} \quad \forall i \neq j, \tag{41}$$

$$c_{ii} = \begin{cases} k \in \{1, \ldots, N\} & \text{if } k \text{ is a super} - \text{exemplar of } i \\ 0 & \text{if } i \text{ is not an exemplar.} \end{cases} \tag{42}$$

The sum of all similarities between data points and the corresponding exemplars, i.e., $\mathscr{S}_1$, and the sum of all linkages between exemplars and the corresponding super-exemplars, i.e., $\mathscr{S}_2$, can be respectively expressed as

$$\mathscr{S}_1 = \sum_{i=1}^{N} \sum_{j=1}^{N} s_{ij} \cdot [c_{ij} \neq 0], \quad \mathscr{S}_2 = \sum_{i=1}^{N} l_{ic_{ii}} \cdot [c_{ii} \neq 0], \tag{43}$$

where $[\cdot]$ is the Iverson notation with [true] $= 1$ and [false] $= 0$. We define a function matrix $[S_{ij}(c_{ij})]_{N\times N}$ with non-diagonal elements incorporating the similarities s_{ij} between data point i and the potential exemplar j, and diagonal elements incorporating the exemplar preference s_{ii} *plus* the linkage $l_{ic_{ii}}$ between exemplar i and its super-exemplar c_{ii}. That is,

$$S_{ij}(c_{ij}) = \begin{cases} s_{ij} & \text{if } i \neq j \& c_{ij} \neq 0 \\ s_{ii} + l_{ic_{ii}} & \text{if } i = j \& c_{ii} \neq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{44}$$

We have $\mathscr{S}_1 + \mathscr{S}_2 = \sum_{i=1}^{N} \sum_{j=1}^{N} S_{ij}(c_{ij})$. The valid assignment matrix C must satisfy the following three constraints:

1. *Exemplar's "1-of-N"* constraint [15]: Each data point i must be assigned to *exactly one* exemplar,

$$I_i(c_{i1}, \ldots, c_{iN}) = \begin{cases} -\infty & \text{if } \sum_{j=1}^{N} [c_{ij} \neq 0] \neq 1 \\ 0 & \text{otherwise.} \end{cases} \tag{45}$$

2. *Exemplar consistency* constraint [15]: If there exists a data point i selecting data point j as its exemplar, then data point j must be an exemplar itself,

$$E_j(c_{1j}, \ldots, c_{Nj}) = \begin{cases} -\infty & \text{if } c_{jj} = 0 \text{ but } \exists i : c_{ij} = 1 \\ 0 & \text{otherwise.} \end{cases} \tag{46}$$

3. *Super-exemplar consistency* constraint: If some exemplar i has chosen exemplar k as its super-exemplar, i.e., $c_{ii} = k$, then k must be a super-exemplar itself,

$$F_k(c_{11}, \ldots, c_{NN}) = \begin{cases} -\infty & \text{if } c_{kk} \neq k \text{ but } \exists i : c_{ii} = k \\ 0 & \text{otherwise.} \end{cases} \tag{47}$$

The goal of the multi-exemplar model is to maximize the following objective function

$$\begin{aligned} \mathscr{S}(C) &= \mathscr{S}_1 + \mathscr{S}_2 + \text{three constraints} \\ &= \sum_{i=1}^{N} \sum_{j=1}^{N} S_{ij}(c_{ij}) + \sum_{i=1}^{N} I_i(c_{i1}, \ldots, c_{iN}) \\ &\quad + \sum_{j=1}^{N} E_j(c_{1j}, \ldots, c_{Nj}) + \sum_{k=1}^{N} F_k(c_{11}, \ldots, c_{NN}). \end{aligned} \tag{48}$$

Figure 6 illustrates the multi-exemplar model. The data points, exemplars and super-exemplars form a two-layer structure by the two mappings ψ_1 and ψ_2. The lower-layer is modeled by the mapping ψ_1. The sum of all similarities between data points and the corresponding exemplars, i.e. $\mathscr{S}_1$, is used to measure the *within-subcluster* compactness. The higher-layer is modeled by the mapping ψ_2. The sum of all linkages between exemplars and the corresponding super-exemplars, i.e. $\mathscr{S}_2$,

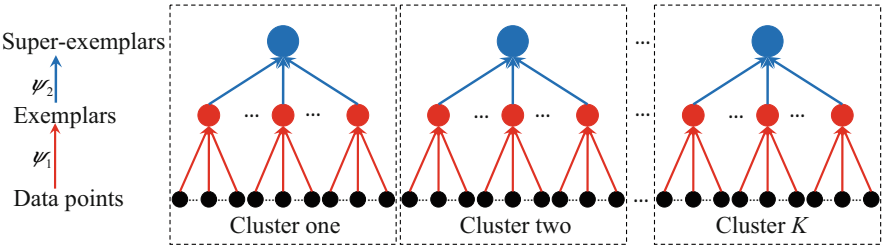


Fig. 6 Illustration of the multi-exemplar model. The mapping ψ_1 assigns each data point to the most appropriate exemplar and the mapping ψ_2 assigns each exemplar to the most appropriate super-exemplar. This model can effectively characterize clusters consisting of multiple subclusters

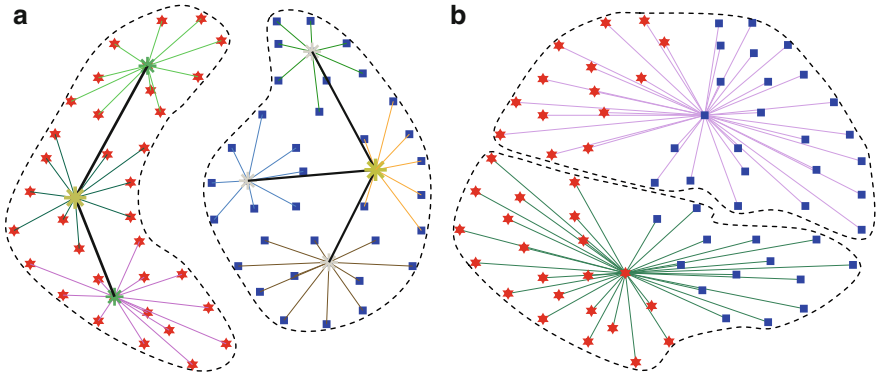


Fig. 7 MEAP vs. AP. The two ground-truth categories are plotted by *red hexagrams* and *blue squares*, respectively. The decision boundaries are plotted by *dash curves*. Each point is assigned to the most appropriate exemplar by *thin lines*. In MEAP, each exemplar (small "*") is assigned to its most appropriate super-exemplar (large "*") by *thick lines*. (**a**) MEAP; (**b**) AP

is used to measure the *within-cluster* compactness. From the single-exemplar theory, maximizing the within-cluster similarity automatically maximizes the between-cluster separation [32]. Therefore, the appropriate multi-exemplar model should be that both the *within-subcluster* compactness and the *within-cluster* compactness are maximized. Maximizing $\mathscr{S}_1 + \mathscr{S}_2$ under the constraints of producing valid clusters (i.e., I, E, F) makes the model effectively characterize clusters consisting of multiple subclusters. Figure 7 compares MEAP with AP in one synthetic data set. From the viewpoint of the maximum margin clustering [49], MEAP finds better decision boundaries than AP, i.e., larger margins are obtained.

3.2.2 Optimization

Exactly searching for an optimal assignment matrix that maximizes the objective function (48) is NP-hard, since the multi-exemplar model is a generalization of

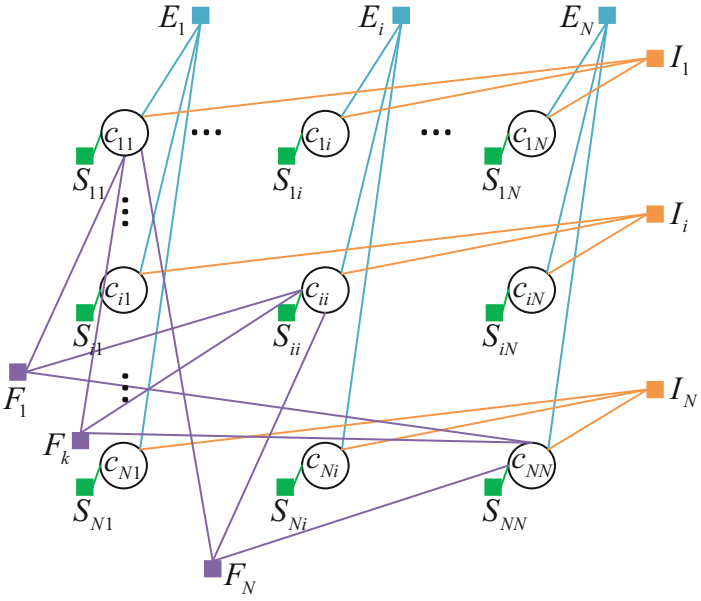


Fig. 8 Factor graph of MEAP

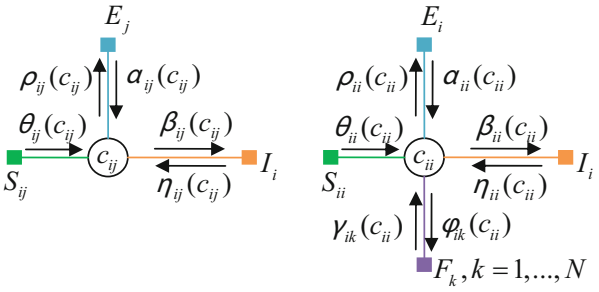
the single-exemplar one, the optimization of which is proved to be NP-hard [14]. To this end, the max-sum belief propagation is utilized, which is a local-message-passing algorithm guaranteed to converge to the neighborhood maximum [46]. The factor graph is shown in Fig. 8. There are seven types of messages passing between variable nodes and function nodes as shown in Fig. 9. In the max-sum algorithm, the message updating involves either a message from a variable to each adjacent function or that from a function to each adjacent variable. The message from a variable to a function sums together the messages from all adjacent functions except the one receiving the message [19],

$$\mu_{x\to f}(x) \leftarrow \sum_{h\in \mathrm{ne}(x)\setminus\{f\}} \mu_{h\to x}(x), \tag{49}$$

where $\mathrm{ne}(x)$ denotes the set of adjacent functions of variable x. The message from a function to a variable involves a maximization over all arguments of the function except the variable receiving the message [19],

$$\mu_{f\to x}(x) \leftarrow \max_{X\setminus\{x\}} \Big[f(X) + \sum_{y\in X\setminus\{x\}} \mu_{y\to f}(y) \Big], \tag{50}$$

Fig. 9 Messages of MEAP



where $X = \mathrm{ne}(f)$ is the set of arguments of function f. Since the messages associated with the non-diagonal variables (i.e., $c_{ij}, i \neq j$) and the diagonal variables (i.e., c_{ii}) are quite different, we will discuss them separately.

1. **Messages of Non-diagonal Elements**

   As shown on the left of Fig. 9, there are five types of messages associated with $c_{ij}, i \neq j$ as follows ($m = 0, 1$):

$$\theta_{ij}(m) = \mu_{S_{ij} \to c_{ij}}(m) = S_{ij}(m) \tag{51}$$

$$\rho_{ij}(m) = \mu_{c_{ij} \to E_j}(m) = \theta_{ij}(m) + \eta_{ij}(m) \tag{52}$$

$$\begin{aligned} \alpha_{ij}(m) &= \mu_{E_j \to c_{ij}}(m) \\ &= \max_{c_{i'j}: i' \neq i} \left[E_j(c_{1j}, \ldots, c_{Nj}) + \sum_{i'' \neq i} \rho_{i''j}(c_{i''j}) \right] \end{aligned} \tag{53}$$

$$\beta_{ij}(m) = \mu_{c_{ij} \to I_i}(m) = \theta_{ij}(m) + \alpha_{ij}(m) \tag{54}$$

$$\begin{aligned} \eta_{ij}(m) &= \mu_{I_i \to c_{ij}}(m) \\ &= \max_{c_{ij'}: j' \neq j} \left[I_i(c_{i1}, \ldots, c_{iN}) + \sum_{j'' \neq j} \beta_{ij''}(c_{ij''}) \right]. \end{aligned} \tag{55}$$

   According to (44) and (51), these messages take as input the similarity $s_{ij}, i \neq j$. Consequently, these messages reflect the accumulated evidence for deciding the partial mapping ψ_1 from data point i to the potential exemplar j ($j \neq i$). That is, $m = c_{ij} = 1$ implies that $\psi_1(i) = j$, and $m = c_{ij} = 0$ implies that $\psi_1(i) \neq j$.

2. **Messages of Diagonal Elements**

There are seven types of messages associated with $c_{ij}, i = j$ (the right of Fig. 9) as follows ($m = 0, \ldots, N$):

$$\theta_{ii}(m) = \mu_{S_{ii} \to c_{ii}}(m) = S_{ii}(m) \tag{56}$$

$$\begin{aligned} \rho_{ii}(m) &= \mu_{c_{ii} \to E_i}(m) \\ &= \theta_{ii}(m) + \eta_{ii}(m) + \sum_{k=1}^{N} \gamma_{ik}(m) \end{aligned} \tag{57}$$

$$\begin{aligned} \alpha_{ii}(m) &= \mu_{E_i \to c_{ii}}(m) \\ &= \max_{c_{i'i}: i' \neq i} \left[E_i(c_{1i}, \ldots, c_{Ni}) + \sum_{i'' \neq i} \rho_{i''i}(c_{i''i}) \right] \end{aligned} \tag{58}$$

$$\begin{aligned} \beta_{ii}(m) &= \mu_{c_{ii} \to I_i}(m) \\ &= \theta_{ii}(m) + \alpha_{ii}(m) + \sum_{k=1}^{N} \gamma_{ik}(m) \end{aligned} \tag{59}$$

$$\begin{aligned} \eta_{ii}(m) &= \mu_{I_i \to c_{ii}}(m) \\ &= \max_{c_{ii'}: i' \neq i} \left[I_i(c_{i1}, \ldots, c_{iN}) + \sum_{i'' \neq i} \beta_{ii''}(c_{ii''}) \right] \end{aligned} \tag{60}$$

$$\begin{aligned} \phi_{ik}(m) &= \mu_{c_{ii} \to F_k}(m) \\ &= \theta_{ii}(m) + \alpha_{ii}(m) + \eta_{ii}(m) + \sum_{k' \neq k} \gamma_{ik'}(m) \end{aligned} \tag{61}$$

$$\begin{aligned} \gamma_{ik}(m) &= \mu_{F_k \to c_{ii}}(m) \\ &= \max_{c_{i'i'}: i' \neq i} \left[F_k(c_{11}, \ldots, c_{NN}) + \sum_{j \neq i} \phi_{jk}(c_{jj}) \right]. \end{aligned} \tag{62}$$

According to (44) and (56), these messages take as input both of the exemplar preference s_{ii} and the linkage $l_{ic_{ii}}$ between exemplars and super-exemplars. Consequently, these messages reflect the accumulated evidence for simultaneously deciding the partial mapping ψ_1 from point i to itself and the mapping ψ_2 from exemplar i to the potential super-exemplar k (k can be either $k = i$ or $k \neq i$). That is, $m = c_{ii} \in \{1, \ldots, N\}$ implies that $\psi_1(i) = i, \psi_2(i) = c_{ii}$, and $m = c_{ii} = 0$ implies that $\psi_1(i) \neq i$ and i is not in the domain of ψ_2.

3. **Simplified Messages**

By applying some mathematical tricks used in [14, 15], we can obtain the simplified messages as follows:

$$\begin{aligned} &i \neq j \\ &\tilde{\rho}_{ij} \leftarrow s_{ij} - \max \left[\max_{j' \notin \{j,i\}} [s_{ij'} + \tilde{\alpha}_{ij'}], \max_{m \in \{1,\ldots,N\}} [l_{im} + \tilde{\gamma}_{im}] + s_{ii} + \tilde{\alpha}_{ii} \right] \end{aligned} \tag{63}$$

$$\tilde{\alpha}_{ij} \leftarrow \min\left[0, \max_{m\in\{1,\ldots,N\}} \tilde{\rho}_j^m + \sum_{i'\notin\{i,j\}} \max[0, \tilde{\rho}_{i'j}]\right] \tag{64}$$

$$\forall i = 1,\ldots,N,\ k = 1,\ldots,N$$

$$\tilde{\rho}_i^k \leftarrow s_{ii} + l_{ik} - \max_{i'\neq i}[s_{ii'} + \tilde{\alpha}_{ii'}] + \tilde{\gamma}_{ik} \tag{65}$$

$$\tilde{\alpha}_{ii} \leftarrow \sum_{i'\neq i} \max\left[0, \tilde{\rho}_{i'i}\right] \tag{66}$$

$$\tilde{\phi}_{ik} \leftarrow \min\left[l_{ik} - \max_{m\neq k}[l_{im} + \tilde{\gamma}_{im}], \tilde{\alpha}_{ii} + \tilde{\rho}_i^k - \tilde{\gamma}_{ik}\right] \tag{67}$$

$$\tilde{\gamma}_{kk} \leftarrow \sum_{i'\neq i} \max\left[0, \tilde{\phi}_{i'k}\right] \tag{68}$$

$$\tilde{\gamma}_{ik} \leftarrow \min\left[0, \tilde{\phi}_{kk} + \sum_{i'\notin\{i,k\}} \max[0, \tilde{\phi}_{i'k}]\right], k \neq i, \tag{69}$$

where $\tilde{\rho}_{ij} = \rho_{ij}(1) - \rho_{ij}(0)$, $\tilde{\alpha}_{ij} = \alpha_{ij}(1) - \alpha_{ij}(0)$, $\tilde{\rho}_i^k = \rho_{ii}(k) - \rho_{ii}(0)$, $\tilde{\alpha}_{ii} = \alpha_{ii}(k) - \alpha_{ii}(0)$, $\tilde{\phi}_{ik} = \phi_{ik}(k) - \max_{m\neq k} \phi_{ik}(m)$, $\tilde{\gamma}_{ik} = \gamma_{ik}(k) - \gamma_{ik}(m : m \neq k)$, and β, η are eliminated. The values of all messages $\tilde{\rho}$, $\tilde{\alpha}$, $\tilde{\phi}$, and $\tilde{\gamma}$ are initialized as zero and updated via Eqs. (63)–(69). The message-updating procedure may be terminated after the local decisions stay constant for some number of iterations t_{conv}, or after a fixed number of iterations $t_{\max}$.

4. **Computing Assignment Matrix**

To estimate the value of an element c_{ij}, we sum together all incoming messages to c_{ij} and take the value $\hat{c}_{ij}$ that maximizes the sum. That is,

$$\begin{aligned}\hat{c}_{ij} &= \arg\max_{c_{ij}} \left[\theta_{ij}(c_{ij}) + \alpha_{ij}(c_{ij}) + \eta_{ij}(c_{ij})\right] \\ &= \begin{cases} 1 & \text{if } \tilde{\alpha}_{ij} + \tilde{\rho}_{ij} \geq 0 \\ 0 & \text{otherwise} \end{cases} \forall i \neq j,\end{aligned} \tag{70}$$

$$\begin{aligned}\hat{c}_{ii} &= \arg\max_{c_{ii}} \left[\theta_{ii}(c_{ii}) + \alpha_{ii}(c_{ii}) + \eta_{ii}(c_{ii}) + \sum_{k=1}^{N} \gamma_{ik}(c_{ii})\right] \\ &= \begin{cases} \arg\max_k \tilde{\rho}_i^k & \text{if } \tilde{\alpha}_{ii} + \max_k \tilde{\rho}_i^k \geq 0 \\ 0 & \text{otherwise.} \end{cases}\end{aligned} \tag{71}$$

The derivation can be found in the supplementary material. From the assignment matrix, we obtain the two mappings ψ_1 and ψ_2, and thus generate the clustering labels $\{(\psi_2 \circ \psi_1)(1), \ldots, (\psi_2 \circ \psi_1)(N)\}$ of N data points.

3.3 Experiments and Applications

In this section, we report the applications in image categorization on the JAFFE data set.

The JAFFE database [25] contains 213 images of 7 facial expressions posed by 10 Japanese females. The 7 facial expressions include 6 basic facial expressions, i.e., happiness, sadness, surprise, anger, disgust, fear, *plus* 1 neutral expression. There are 3 or 4 examples for each expression per person. Images of the same person in different facial expressions should be taken as in distinct subclasses [52, 53]. The goal is to cluster the 213 images into 10 groups according to identity.

We plot in Fig. 10 the complete results on the JAFFE data set by MEAP when setting the preferences at the median of the similarities. The exemplars and super-exemplars are bounded by thin and thick red lines, respectively. Each exemplar is

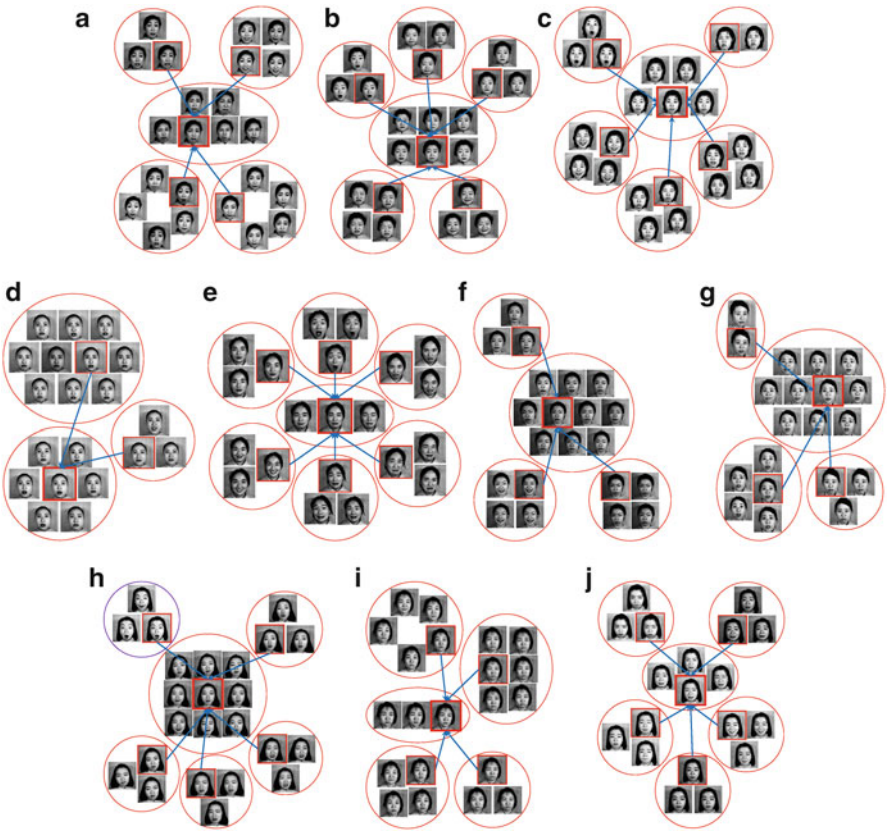


Fig. 10 Clusters learned by MEAP on the JAFFE data set. (**a**) KA cluster; (**b**) KL cluster; (**c**) KM cluster; (**d**) KR cluster; (**e**) MK cluster; (**f**) NA cluster; (**g**) NM cluster; (**h**) TM cluster; (**i**) UY cluster; (**j**) YM cluster

connected to the corresponding super-exemplar by the blue arrow. In this setting, although the AP method results in a moderate number of clusters [14], yet it over-segments the images of ten subjects into more clusters than the actual, with each cluster corresponding to a facial expression. When setting the preferences at the minimum of similarities, leading to a small number of clusters in AP, some images are misclassified by AP due to the presence of facial expression. However, MEAP can overcome the problem and almost perfectly group images of the same subject into one cluster by assigning exemplars of the same subject to the corresponding super-exemplar. Among 213 images, only 3 images belonging to the YM category are misclassified to the TM category, leading to a 98.6 % classification rate. The reason is that the exemplar of these 3 images is more similar to the super-exemplar of the TM category than YM. On the other hand, the AP algorithm misclassifies 36 images, resulting in a 83.1 % classification rate.

4 Graph-Based Multi-prototype Competitive Learning

Based on the advantages of graph method, multi-prototype representation, and competitive learning, this section reviews a novel nonlinear clustering method, namely GMPCL [45]. Based on the graph method, the GMPCL method firstly generates an initial coarse clustering. That is, we first construct a graph with vertex energy vector from the data set; a core-point set is computed according to the vertex energy, from which we can obtain an initial clustering by constructing connected components via neighbor connectivity. Then, a multi-prototype competitive learning is designed to refine this initial clustering so as to generate accurate clustering results. Experimental results in automatic color image segmentation validate the effectiveness of GMPCL and show that GMPCL provides a useful tool for computer vision.

4.1 Graph-Based Initial Clustering

Given a data set $\mathscr{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ of n points in $\mathbb{R}^d$, the first step of the graph-based algorithm is to construct a graph $G^{\mathbf{e}} = (\mathscr{V}, A, \mathbf{e})$. The vertex set $\mathscr{V}$ contains one node for each sample in $\mathscr{D}$. The affinity matrix $A = [A_{ij}]_{n\times n}$ is defined as

$$A_{ij} \triangleq \begin{cases} \exp(-\| \mathbf{x}_i - \mathbf{x}_j \|^2) & \text{if } \mathbf{x}_i \in \mathscr{N}_k(\mathbf{x}_j) \wedge \mathbf{x}_j \in \mathscr{N}_k(\mathbf{x}_i) \\ 0 & \text{otherwise,} \end{cases} \tag{72}$$

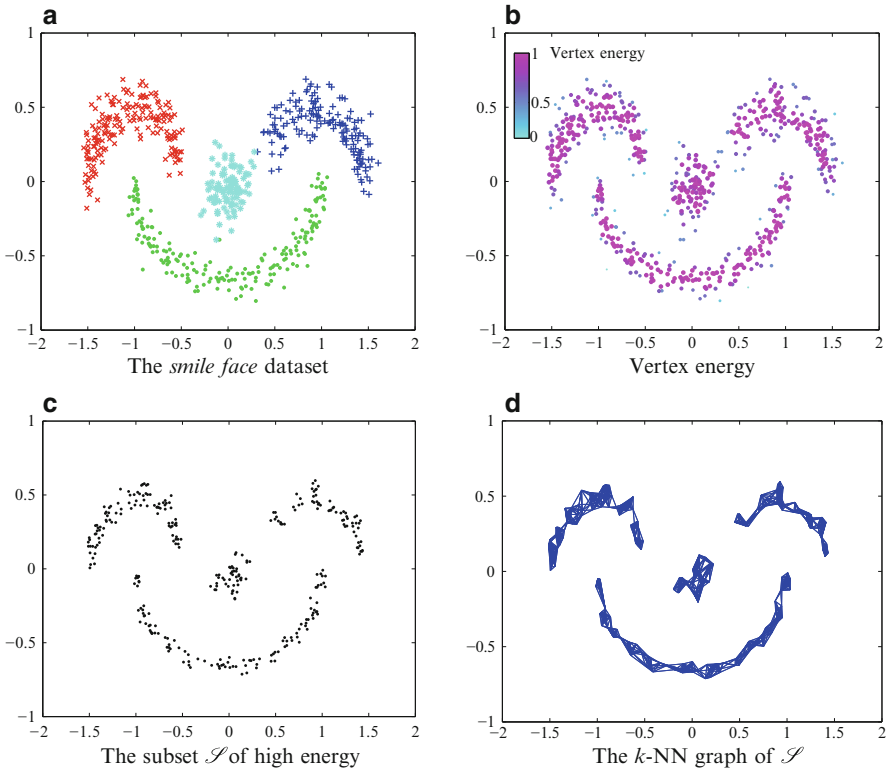


Fig. 11 *Smile face* example. (**a**) The *smile face* data set, different classes are plotted in *different markers*. (**b**) The vertex energy of *smile face*, the color and size of each point is marked according to its vertex energy. (**c**) The subset $\mathscr{S}$ comprising vertices of high energy (*core points*). (**d**) The k-NN graph of $\mathscr{S}$ with $k = 21$

where $\mathscr{N}_k(\mathbf{x}_i)$ denotes the set consisting of k nearest neighbors of $\mathbf{x}_i$. The vertex energy vector $\mathbf{e} = [e_1, \ldots, e_n]^\top$ is defined as

$$e_i \triangleq \log_2 \left(1 + \frac{\sum_j A_{ij}}{\max_{l=1,\ldots,n} \sum_j A_{lj}} \right), i = 1, \ldots, n. \tag{73}$$

The component $e_i \in [0, 1]$ is the vertex energy of $\mathbf{x}_i$, which measures how "important" $\mathbf{x}_i$ is.

Figure 11a shows a *smile face* data set, and Fig. 11b plots its vertex energy. In [12], the density is measured simply by the number of points within the ϵ-neighborhood of one point. However, the vertex energy defined in (73) takes into account the correlations between all data points, which results in a global estimate of the vertex energy. Although both of them can be used to discover arbitrarily shaped clusters, the vertex energy is more suitable for datasets containing clusters of differing densities.

A subset $\mathscr{S}$ consisting of the vertices of higher energy is obtained (e.g., Fig. 11c), which is termed *core-point* set.

Definition 3. Given a graph $G^{\mathbf{e}} = (\mathscr{V}, A, \mathbf{e})$ and a percentage ρ, the core-point set $\mathscr{S}$ is defined as $\mathscr{S} \triangleq \{\mathbf{x}_i | e_i \geq \zeta\}$ with $\zeta \in [0, 1]$ such that $|\mathscr{S}|/|\mathscr{V}| = \rho$.

The core-point-connectivity of any two core points $\mathbf{p}$ and $\mathbf{q}$ in $\mathscr{S}$ is defined as follows.

Definition 4 (Core-Point-Connectivity). Two core points $\mathbf{p}$ and $\mathbf{q}$ in $\mathscr{S}$ are core-point-connected w.r.t. k (denoted as $\mathbf{p} \bowtie_k^{\mathscr{S}} \mathbf{q}$) if there exists a chain of core points $\mathbf{p}_1, \ldots, \mathbf{p}_m$, $\mathbf{p}_1 = \mathbf{p}$, $\mathbf{p}_m = \mathbf{q}$ such that $\mathbf{p}_{i+1} \in \mathscr{N}_k(\mathbf{p}_i) \bigcap \mathscr{S}$ and $\mathbf{p}_i \in \mathscr{N}_k(\mathbf{p}_{i+1}) \bigcap \mathscr{S}$.

From the viewpoint of density-based clustering [11, 12], the core-point-connectivity separates $\mathscr{S}$ into some natural subgroups, as shown in Fig. 11d, which are defined as connected components as follows.

Definition 5. A set of c connected components $\{\mathscr{I}_1, \ldots, \mathscr{I}_c\}$ is obtained by separating the core-point set $\mathscr{S}$ w.r.t. k, such that $\forall i \neq j, \mathscr{I}_i \bigcap \mathscr{I}_j = \emptyset$, $\mathscr{S} = \bigcup_{i=1}^{c} \mathscr{I}_i$, and $\forall \mathbf{p}, \mathbf{q} \in \mathscr{I}_i$, $\mathbf{p} \bowtie_k^{\mathscr{S}} \mathbf{q}$, while $\forall \mathbf{p} \in \mathscr{I}_i, \forall \mathbf{q} \in \mathscr{I}_j, i \neq j, \mathbf{p} \bowtie_k^{\mathscr{S}} \mathbf{q}$ does not hold.

The connected components $\{\mathscr{I}_1, \ldots, \mathscr{I}_c\}$ are taken as initial clusters which will be further refined via *multi-prototype* competitive learning.

4.2 Multi-prototype Competitive Learning

The initial clusters $\{\mathscr{I}_1, \ldots, \mathscr{I}_c\}$ obtained in the first phase take into account only data points of higher energy, and the remaining data points are not assigned with cluster labels. Therefore, the output of the first phase is only a coarse clustering that requires further refinement. Rather than directly assigning the unlabeled data points to the *core points* as in [11], this section employs classical competitive learning to refine the initial clustering and assign cluster labels to all data points. Experimental results show that the approach can obtain at least 9.8 % improvement over the direct assignment.

Since the data set is nonlinearly separable, a nonlinear cluster with concave boundaries would always exist, which cannot be characterized by a *single* prototype that produces *convex* boundaries [24]. However, *multiple* prototypes produce subregions of the Voronoi diagram which can approximately characterize one cluster of an arbitrary shape. Therefore, we represent each cluster by *multiple* prototypes.

Every point in $\mathscr{I}_j$ can be taken as one of the initial prototypes representing the j-th cluster $\mathscr{C}_j$. But there is no need of using so many prototypes to represent one cluster and some of them are more appropriate and more effective than others. These points should be as few as possible to lower the computational complexity of *multi-prototype* competitive learning, meanwhile be scattered in the whole space of the

initial cluster in order to suitably characterize the corresponding cluster. Affinity propagation [14] can generate suitable prototypes to represent an input data set without preselecting the number of prototypes. In experiments, the representative points are obtained by applying affinity propagation to each $\mathscr{I}_j$. The similarity $s(\mathbf{x}_i, \mathbf{x}_{i'})$ between $\mathbf{x}_i, \mathbf{x}_{i'} \in \mathscr{I}_j$ is set to $- \parallel \mathbf{x}_i - \mathbf{x}_{i'} \parallel^2$ and the preferences are set to the median of the similarities, which outputs p_j suitable *multi-prototypes* $\mathbf{w}_j^1, \ldots, \mathbf{w}_j^{p_j}$. In this way, we obtain an initial *multi-prototype* set

$$\mathscr{W} = \{\underbrace{\mathbf{w}_1^1, \ldots, \mathbf{w}_1^{p_1}}_{\text{represent}\mathscr{C}_1}, \underbrace{\mathbf{w}_2^1, \ldots, \mathbf{w}_2^{p_2}}_{\text{represent}\mathscr{C}_2}, \ldots, \underbrace{\mathbf{w}_c^1, \ldots, \mathbf{w}_c^{p_c}}_{\text{represent}\mathscr{C}_c}\}. \tag{74}$$

We use the index notation ω_j^q to denote the *multi-prototype* $\mathbf{w}_j^q$. That is, referring to the ω_j^q-th *multi-prototype* is equivalent to mentioning $\mathbf{w}_j^q$, and $\omega = \{\omega_1^1, \ldots, \omega_1^{p_1}, \omega_2^1, \ldots, \omega_2^{p_2}, \ldots, \omega_c^1, \ldots, \omega_c^{p_c}\}$.

After the initial *multi-prototype* set $\mathscr{W}$ is obtained, classical competitive learning is performed to iteratively update the *multi-prototypes* such that the *multi-prototype objective* function is minimized:

$$J(\mathscr{W}) = \sum_{i=1}^{n} \parallel \mathbf{x}_i - \mathbf{w}_{\upsilon_i}^{\upsilon_i} \parallel^2, \tag{75}$$

where $\mathbf{w}_{\upsilon_i}^{\upsilon_i}$ satisfies $\omega_{\upsilon_i}^{\upsilon_i} = \arg\min_{\omega_j^q \in \omega} \parallel \mathbf{x}_i - \mathbf{w}_j^q \parallel^2$, i.e., the winning *multi-prototype* of $\mathbf{x}_i$ that is nearest to $\mathbf{x}_i$. For each randomly taken $\mathbf{x}_i$, the winning *multi-prototype* $\omega_{\upsilon_i}^{\upsilon_i}$ is selected via the **winner selection rule**:

$$\omega_{\upsilon_i}^{\upsilon_i} = \arg\min_{\omega_j^q \in \omega} \parallel \mathbf{x}_i - \mathbf{w}_j^q \parallel^2, \tag{76}$$

and is updated by the **winner update rule**:

$$\mathbf{w}_{\upsilon_i}^{\upsilon_i} \leftarrow \mathbf{w}_{\upsilon_i}^{\upsilon_i} + \eta_t(\mathbf{x}_i - \mathbf{w}_{\upsilon_i}^{\upsilon_i}) \tag{77}$$

with learning rates $\{\eta_t\}$ satisfying [4]: $\lim_{t\to\infty} \eta_t = 0$, $\sum_{t=1}^{\infty} \eta_t = \infty$, $\sum_{t=1}^{\infty} \eta_t^2 < \infty$. In practice, $\eta_t = const/t$, where *const* is a small constant, e.g., 0.5.

Figure 12a illustrates the procedure of updating the winning *multi-prototype*. The converged *multi-prototype* set $\mathscr{W}$ and the corresponding Voronoi diagram are shown in Fig. 12b. The *multi-prototypes* representing different clusters are plotted in different markers. The *piecewise linear separator* consists of hyperplanes shared by two subregions which are induced by the *multi-prototypes* representing different clusters. This *piecewise linear separator* is used to identify nonlinearly separable clusters, as shown in Fig. 12c.

Algorithm 2 summarizes the *GMPCL* method.

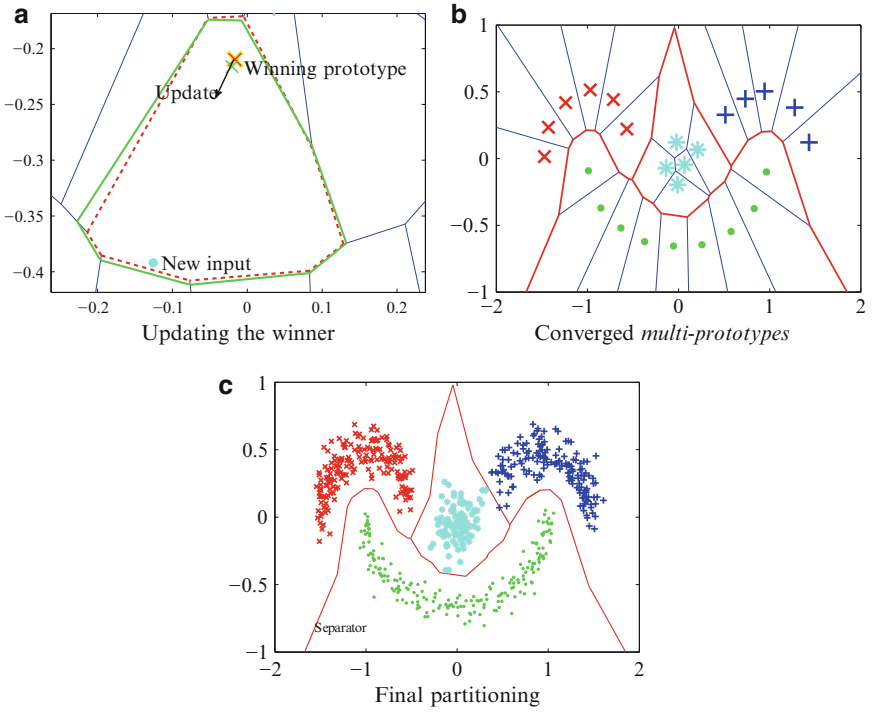


Fig. 12 The winner update and the clustering result of *smile face*. (**a**) The procedure of updating the winning *multi-prototype*, during which both the winning *multi-prototype* and the corresponding lines in Voronoi diagram move slightly: *red* "×" marker and *dashed lines* before update, while *green* "+" marker and *lines* after update. (**b**) The converged *multi-prototypes* and the corresponding Voronoi diagram, the *multi-prototypes* representing different clusters are plotted in *different markers*, and the *piecewise linear separator* is plotted in *red*. (**c**) The final clusters separated by the *piecewise linear separator*

4.3 Fast GMPCL

High-dimensional clustering applications, such as video clustering, are characterized by a high computational load, which is mainly due to the *redundant* calculation of the distances between high-dimensional points in the update procedure of competitive learning. To overcome this problem, an approach similar to the kernel trick [31] is considered. First, an *inner product* matrix $M = [M_{i,j}]_{n\times n}$ of the data set $\mathscr{D}$ is computed such that $M_{i,j} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$. Then the computation of $\| \mathbf{x}_i - \mathbf{x}_j \|^2$ is efficiently accomplished by $\| \mathbf{x}_i - \mathbf{x}_j \|^2 = M_{i,i} + M_{j,j} - 2M_{i,j}$. Thus, the *redundant* high-dimensional computation is avoided. Unfortunately, it cannot be directly applied in competitive learning due to the incremental update rule. Since the winning *multi-prototype* $\mathbf{w}_{v_i}^{v_i}$ is updated by $\mathbf{w}_{v_i}^{v_i} \leftarrow \mathbf{w}_{v_i}^{v_i} + \eta_t(\mathbf{x}_i - \mathbf{w}_{v_i}^{v_i})$, it is unlikely

Algorithm 2: GMPCL

Input: $\mathscr{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, ρ, t_{max}, $\{\eta_t\}$, ϵ.
Construct a graph $G^{\mathbf{e}} = (\mathscr{V}, A, \mathbf{e})$ and initialize a clustering $\{\mathscr{I}_1, \ldots, \mathscr{I}_c\}$.
Initialize a *multi-prototype* set $\mathscr{W}$, set $t = 0$.
repeat
  Randomly permute $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, $\hat{\mathscr{W}} = \mathscr{W}$, $t \leftarrow t + 1$.
  for $i = 1, \ldots, n$ **do**
    Select a winning *multi-prototype* $\omega_{v_i}^{v_i}$ by (76).
    Update $\mathbf{w}_{v_i}^{v_i}$ with learning rate η_t by (77).
  end for
  Compute $L = \| \mathscr{W} - \hat{\mathscr{W}} \|^2 = \sum_{\omega_j^q \in \omega} \| \mathbf{w}_j^q - \hat{\mathbf{w}}_j^q \|^2$.
until $L \leq \epsilon$ or $t \geq t_{max}$
Output: clusters
$\{\mathscr{C}_1, \ldots, \mathscr{C}_c : \mathbf{x}_i \in \mathscr{C}_{v_i}, \text{ s.t. } \omega_{v_i}^{v_i} = \arg\min_{\omega_j^q \in \omega} \| \mathbf{x}_i - \mathbf{w}_j^q \|^2, \forall i = 1, \ldots, n\}$.

that the updated $\mathbf{w}_{v_i}^{v_i}$ satisfies $\mathbf{w}_{v_i}^{v_i} \in \mathscr{D}$. No pre-computed distance $\| \mathbf{x}_i - \mathbf{w}_{v_i}^{v_i} \|^2$ is available for calculating (76).

In [40], a *prototype descriptor* W^{ψ} is designed to represent c prototypes $\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_c\}$ in the kernel space induced by a mapping ψ. The *prototype descriptor* W^{ψ} is a $c \times (n+1)$ matrix, whose rows represent prototypes as the inner products between a prototype and the data points, as well as the squared length of the prototype. That is,

$$W^{\psi} = \begin{pmatrix} \langle \boldsymbol{\mu}_1, \psi(\mathbf{x}_1) \rangle & \ldots & \langle \boldsymbol{\mu}_1, \psi(\mathbf{x}_n) \rangle & \langle \boldsymbol{\mu}_1, \boldsymbol{\mu}_1 \rangle \\ \langle \boldsymbol{\mu}_2, \psi(\mathbf{x}_1) \rangle & \ldots & \langle \boldsymbol{\mu}_2, \psi(\mathbf{x}_n) \rangle & \langle \boldsymbol{\mu}_2, \boldsymbol{\mu}_2 \rangle \\ \vdots & \vdots & \vdots & \vdots \\ \langle \boldsymbol{\mu}_c, \psi(\mathbf{x}_1) \rangle & \ldots & \langle \boldsymbol{\mu}_c, \psi(\mathbf{x}_n) \rangle & \langle \boldsymbol{\mu}_c, \boldsymbol{\mu}_c \rangle \end{pmatrix}. \tag{78}$$

Competitive learning in the kernel space becomes a process of updating W^{ψ}. *Multi-prototype descriptor* is developed, which is a row-block matrix independent of the dimensionality, and extend GMPCL to deal with high-dimensional clustering.

4.3.1 Inner Product Based Computation

According to the initialization of *multi-prototypes*, the initial $\mathscr{W}$ satisfies $\mathscr{W} \subset \mathscr{D}$. The *multi-prototype descriptor* is defined as follows.

Definition 6 (Multi-prototype Descriptor). A multi-prototype descriptor is a row-block matrix W of size $|\mathscr{W}| \times (n+1)$,

$$W = \begin{pmatrix} W_1 \\ \vdots \\ W_c \end{pmatrix} \tag{79}$$

such that the j-th block W_j represents $\mathscr{C}_j$, and the q-th row of W_j, i.e. $W^q_{j,:}$, represents $\mathbf{w}^q_j$ by

$$W^q_{j,i} = \langle \mathbf{w}^q_j, \mathbf{x}_i \rangle, i = 1, \ldots, n,\ W^q_{j,n+1} = \langle \mathbf{w}^q_j, \mathbf{w}^q_j \rangle, \tag{80}$$

where $W^q_{j,i}$ denotes the i-th column of $W^q_{j,:}$. That is,

$$\left(\begin{array}{ccccc} \langle \mathbf{w}^1_1, \mathbf{x}_1 \rangle & \langle \mathbf{w}^1_1, \mathbf{x}_2 \rangle & \ldots & \langle \mathbf{w}^1_1, \mathbf{x}_n \rangle & \langle \mathbf{w}^1_1, \mathbf{w}^1_1 \rangle \\ \langle \mathbf{w}^2_1, \mathbf{x}_1 \rangle & \langle \mathbf{w}^2_1, \mathbf{x}_2 \rangle & \ldots & \langle \mathbf{w}^2_1, \mathbf{x}_n \rangle & \langle \mathbf{w}^2_1, \mathbf{w}^2_1 \rangle \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \langle \mathbf{w}^{p_1}_1, \mathbf{x}_1 \rangle & \langle \mathbf{w}^{p_1}_1, \mathbf{x}_2 \rangle & \ldots & \langle \mathbf{w}^{p_1}_1, \mathbf{x}_n \rangle & \langle \mathbf{w}^{p_1}_1, \mathbf{w}^{p_1}_1 \rangle \\ \hline \vdots & \vdots & \vdots & \vdots & \vdots \\ \hline \langle \mathbf{w}^1_c, \mathbf{x}_1 \rangle & \langle \mathbf{w}^1_c, \mathbf{x}_2 \rangle & \ldots & \langle \mathbf{w}^1_c, \mathbf{x}_n \rangle & \langle \mathbf{w}^1_c, \mathbf{w}^1_c \rangle \\ \langle \mathbf{w}^2_c, \mathbf{x}_1 \rangle & \langle \mathbf{w}^2_c, \mathbf{x}_2 \rangle & \ldots & \langle \mathbf{w}^2_c, \mathbf{x}_n \rangle & \langle \mathbf{w}^2_c, \mathbf{w}^2_c \rangle \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \langle \mathbf{w}^{p_c}_c, \mathbf{x}_1 \rangle & \langle \mathbf{w}^{p_c}_c, \mathbf{x}_2 \rangle & \ldots & \langle \mathbf{w}^{p_c}_c, \mathbf{x}_n \rangle & \langle \mathbf{w}^{p_c}_c, \mathbf{w}^{p_c}_c \rangle \end{array}\right).$$

Using the ω-notation, the ω^q_j-th row, i.e. $W^q_{j,:}$, represents the ω^q_j-th multi-prototype, i.e. $\mathbf{w}^q_j$.

The initial *multi-prototype descriptor* W is obtained as a sub-matrix of M. In Algorithm 2, three key procedures of *multi-prototype* competitive learning are involved with the *redundant* computation of distances, which are the winning *multi-prototype* selection, the winner update, and the computation of the sum of prototype update, i.e. L. Based on the *multi-prototype descriptor*, we implement these procedures whose computational complexity is independent of the dimensionality. For detailed proofs of these theorems and lemmas, readers are encouraged to refer to the [40].

Theorem 6 (Winner Selection Rule). *The selection of the winning multi-prototype* $\omega^{v_i}_{v_i}$ *of* $\mathbf{x}_i$ *can be realized by*

$$\omega^{v_i}_{v_i} = \arg\min_{\omega^q_j \in \omega} \left(W^q_{j,n+1} - 2W^q_{j,i} \right). \tag{81}$$

Theorem 7 (Winner Update Rule). *The update of the winning multi-prototype* $\omega^{v_i}_{v_i}$ *of* $\mathbf{x}_i$ *can be realized by*

$$W_{v_i,j}^{v_i} \leftarrow \begin{cases} (1-\eta_t)W_{v_i,j}^{v_i} + \eta_t M_{i,j} & \text{if } j = 1,\ldots,n, \\ (1-\eta_t)^2 W_{v_i,j}^{v_i} + \eta_t^2 M_{i,i} + 2(1-\eta_t)\eta_t W_{v_i,i}^{v_i} & \text{if } j = n+1. \end{cases} \tag{82}$$

Similar to [40], in **one** iteration of competitive learning, each data point $\mathbf{x}_i$ is assigned to **exactly one** *multi-prototype*. Let the index array $\pi_j^q = [\pi_j^q(1), \pi_j^q(2), \ldots, \pi_j^q(m_j^q)]$ store the indices of m_j^q *ordered* data points assigned to the ω_j^q-th *multi-prototype* in **one** iteration. For instance, if $\mathbf{x}_1$, $\mathbf{x}_{32}$, $\mathbf{x}_8$, $\mathbf{x}_{20}$, $\mathbf{x}_{15}$ are 5 *ordered* data points assigned to the ω_3^2-th *multi-prototype* in the t-th iteration, then the index array $\pi_3^2 = [\pi_3^2(1), \pi_3^2(2), \ldots, \pi_3^2(m_3^2)] = [1, 32, 8, 20, 15]$ with $\pi_3^2(1) = 1, \pi_3^2(2) = 32, \pi_3^2(3) = 8, \pi_3^2(4) = 20, \pi_3^2(5) = 15$ and $m_3^2 = 5$. The following lemma formulates the cumulative update of the ω_j^q-th *multi-prototype* based on the index array π_j^q.

Lemma 2. *In the t-th iteration, the relation between the updated multi-prototype* $\mathbf{w}_j^q$ *and the old* $\hat{\mathbf{w}}_j^q$ *is:*

$$\mathbf{w}_j^q = (1-\eta_t)^{m_j^q} \hat{\mathbf{w}}_j^q + \eta_t \sum_{l=1}^{m_j^q} (1-\eta_t)^{m_j^q - l} \mathbf{x}_{\pi_j^q(l)}. \tag{83}$$

Theorem 8 (Iteration Stopping Criteria). *The iteration stopping criteria of multi-prototype competitive learning can be realized by* $L \leq \epsilon$ *or* $t \geq t_{max}$*, where* L *is computed as*

$$\begin{aligned} L = &\sum_{\omega_j^q \in \omega} \left(\left(1 - \frac{1}{(1-\eta_t)^{m_j^q}} \right)^2 W_{j,n+1}^q \right) + \eta_t^2 \sum_{\omega_j^q \in \omega} \sum_{h=1}^{m_j^q} \sum_{l=1}^{m_j^q} \frac{M_{\pi_j^q(h),\pi_j^q(l)}}{(1-\eta_t)^{h+l}} \\ &+ 2\eta_t \sum_{\omega_j^q \in \omega} \left(\left(1 - \frac{1}{(1-\eta_t)^{m_j^q}} \right) \sum_{l=1}^{m_j^q} \frac{W_{j,\pi_j^q(l)}^q}{(1-\eta_t)^l} \right). \end{aligned} \tag{84}$$

4.3.2 FGMPCL in High Dimension

Based on the *multi-prototype descriptor* W and the above theorems, a *fast* GMPCL (FGMPCL) is proposed for high-dimensional clustering, which is summarized in Algorithm 3. According to the three theorems, it is easy to prove that FGMPCL generates the same clustering as GMPCL. The analysis of the asymptotic computational complexity reveals that, in high-dimensional clustering, i.e. when $d \gg n$, FGMPCL can save $O(t_{\max} n(d|\mathscr{W}| - n))$ computations.

Algorithm 3: FGMPCL

Input: $\mathscr{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, ρ, t_{max}, $\{\eta_t\}$, ϵ.
Compute an *inner product* matrix $M = [\langle \mathbf{x}_i, \mathbf{x}_j \rangle]_{n \times n}$.
The same graph-based initial clustering step as GMPCL is performed except directly taking values from M.
Initialize a *multi-prototype descriptor* W, set $t = 0$.
repeat
  Randomly permute $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, initialize $|\mathscr{W}|$ empty index arrays $\{\pi_j^q = \emptyset : \omega_j^q \in \omega\}$, $t \leftarrow t + 1$.
  for $i = 1, \ldots, n$ **do**
    Select a winning *multi-prototype* $\omega_{v_i}^{v_i}$ by (81), and append i to the index array $\pi_{v_i}^{v_i}$.
    Update $W_{v_i,:}^{v_i}$ (i.e., the $\omega_{v_i}^{v_i}$-th *multi-prototype*) with learning rate η_t by (82).
  end for
  Compute L by (84).
until $L \leq \epsilon$ or $t \geq t_{max}$
Output: clusters
$\{\mathscr{C}_1, \ldots, \mathscr{C}_c : \mathbf{x}_i \in \mathscr{C}_{v_i}, \text{ s.t. } \omega_{v_i}^{v_i} = \arg\min_{\omega_j^q \in \omega} \left(W_{j,n+1}^q - 2W_{j,i}^q \right), \forall i = 1, \ldots, n\}$.

First, we should notice that one computation of the distances or the inner products between all data points is unavoidable, which takes $O(n^2 d)$ operations. Our goal here is to eliminate the *redundant* computation occurring in the procedure of the *multi-prototype* update.

The initialization of the *multi-prototype descriptor* W only takes $O(|\mathscr{W}|(n+1))$ operations (taking $|\mathscr{W}|(n+1)$ entries from the matrix M). For the t-th iteration, the initialization takes $O(|\mathscr{W}| + 1 + n)$ operations ($O(|\mathscr{W}|)$ for initializing $|\mathscr{W}|$ empty index arrays, $O(1)$ for increasing $t = t + 1$ and $O(n)$ for randomly permuting the data set). There are n data points, and each takes $O(|\mathscr{W}|)$ operations to select a winner by (81) and $O(n+1)$ to update the winner by (82). Thus $O(n(n+1+|\mathscr{W}|))$ operations are needed for the update procedure in one iteration. The computation of (84) takes $O(|\mathscr{W}|+n^2+n)$ operations (the first term is $O(|\mathscr{W}|)$, the second term is $O(n^2)$ and the third term is $O(n)$). Therefore, in one iteration, the total computational complexity is $O(|\mathscr{W}|+1+n)+O(n(n+1+|\mathscr{W}|))+O(|\mathscr{W}|+n^2+n) = O(n^2)$, since $|\mathscr{W}| < n$. Assume that the iteration number reaches the maximum iteration number $t_{\max}$, the computational complexity of iteration procedure is $O(t_{\max} n^2)$. Therefore, in FGMPCL, the computational complexity of multi-prototype competitive learning is $O(|\mathscr{W}|(n+1)) + O(t_{\max} n^2) = O(t_{\max} n^2)$.

However, in GMPCL, the computational complexity of multi-prototype competitive learning is $O(t_{\max} nd|\mathscr{W}|)$. In high-dimensional clustering where $d \gg n$, it is easy to see that $O(t_{\max} nd|\mathscr{W}|) \gg O(t_{\max} n^2|\mathscr{W}|) > O(t_{\max} n^2)$. The fast version can save $O(t_{\max} nd|\mathscr{W}|) - O(t_{\max} n^2) = O\left(t_{\max} n(d|\mathscr{W}| - n)\right)$ computations when $d \gg n$.

In practice, we suggest making a choice between GMPCL and FGMPCL before performing clustering based on the relation between n and d. If $d \gg n$, FGMPCL is preferable; otherwise, use GMPCL to perform clustering.

4.4 Experiments and Applications

In this section, we applied GMPCL to automatic color image segmentation. Our intention here is to demonstrate that GMPCL has the capability of finding visually appealing structures in real color images. The experiment was performed on images from the Berkeley Segmentation data set (BSDS)[1] [28]. BSDS contains 300 images of a wide variety of natural scenes, as well as "ground truth" segmentations produced by humans [13], aiming at providing an empirical basis for research on image segmentation and boundary detection. The size of each image is either 480×320 or 320×480, which is too large for directly computing 153,600 pixel feature vectors. So we resized the images by 0.4 into either 192×129 or 129×192. We used the 3-D vectors of color features for each pixel as the feature vectors to segment a color image. Since the L*a*b color is designed to approximate human vision and suitable for interpreting the real world [17], the coordinates in the L*a*b* color space were used as the features. Thus for each image we obtained a data set $\mathscr{D} = \{\mathbf{x}_i \in \mathbb{R}^3 : i = 1, \ldots, 24768\}$. Before applying clustering, smoothing was performed using the averaging filter of size 3×3 to avoid the over-segmentation caused by local color variants.

Figure 13 displays some of the segmentation results obtained by GMPCL, without any further postprocessing. Table 2 lists the means and variances of PRI, NPRI and NMI, and the average computational time in seconds on 300 images from BSDS. The compared methods include Rival penalized competitive learning (RPCL) [48], kernel k-means (k$k$means) [31], Ncut [33], Graclus [9], and gPb-owt-ucm [1]. The results show that although GMPCL is not the best of all compared methods, it is ranked in the second place and is comparable to the best algorithm. Please note that, gPb-owt-ucm is coupled to a high-performance contour detector proposed in [27], meanwhile the other algorithms only rely on the color information to perform image segmentation.

5 Position Regularized Support Vector Clustering

SVC is a nonlinear clustering method developed from SVDD. Compared with other clustering methods, the advantages of SVC include the abilities to discover clusters of arbitrary shapes and to deal with outliers. However, the value of the trade-off parameter directly determines the size of sphere and therefore influences the data distribution of sphere surface, i.e., the clustering results of SVC. To eliminate the influence of the trade-off parameter C to SVC [3], this section reviews a PSVC [39].

[1] www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/.

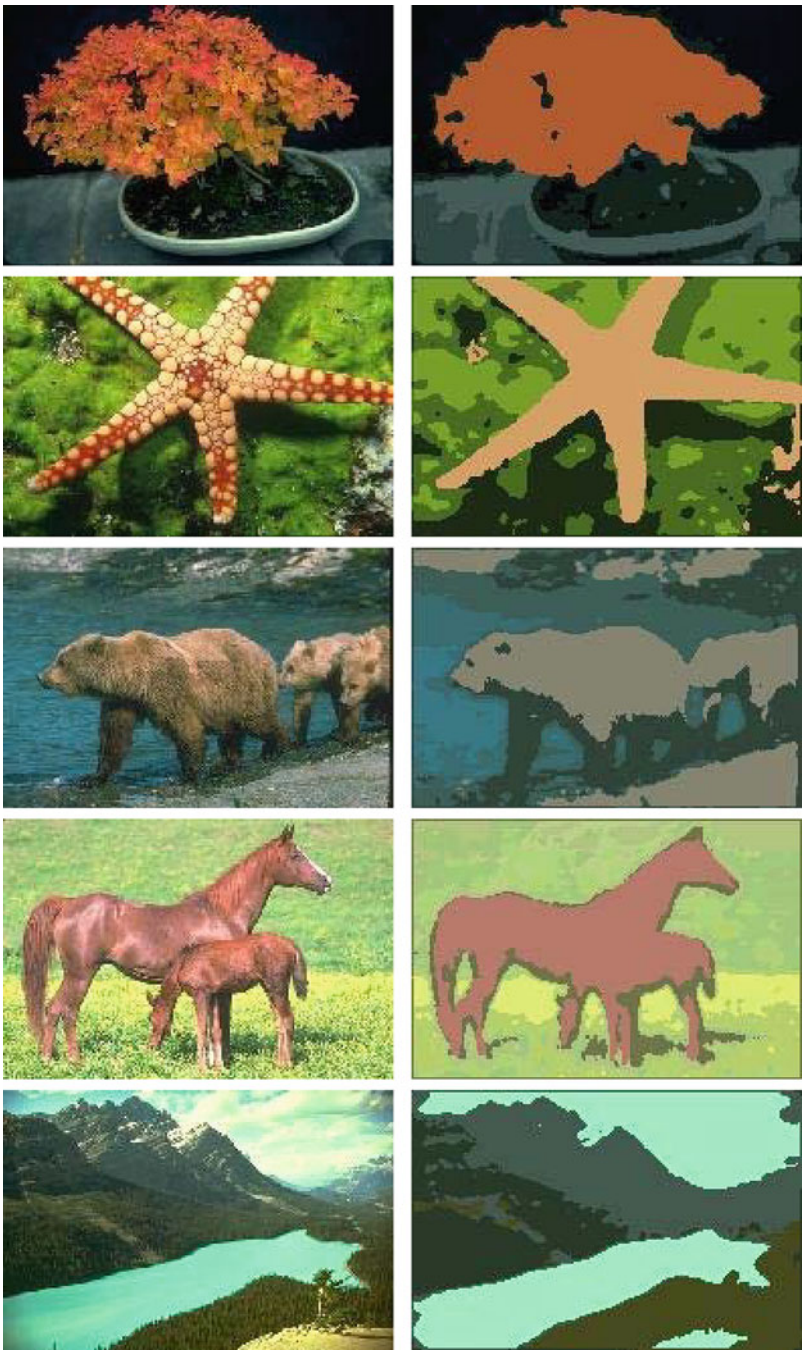


Fig. 13 Some of the image segmentation results by GMPCL. The first column displays the original images and the second column displays the segmentation results. Each segment (cluster) is painted with its mean color

Table 2 The means and variances of PRI, NPRI and NMI, and the average computational time in seconds on images from BSDS, The bold values denote the best results

Methods	PRI	NPRI	NMI	Time
RPCL [48]	0.739 ± 0.020	0.701 ± 0.022	0.607 ± 0.025	89.17
k*k*means [31]	0.743 ± 0.014	0.723 ± 0.016	0.626 ± 0.020	74.05
Ncut [33]	0.741 ± 0.015	0.722 ± 0.017	0.623 ± 0.022	28.32
Graclus [9]	0.751 ± 0.011	0.742 ± 0.014	0.625 ± 0.015	19.24
GMPCL	$\mathbf{0.758 \pm 0.009}$	$\mathbf{0.753 \pm 0.010}$	$\mathbf{0.642 \pm 0.012}$	**18.53**
gPb-owt-ucm [1]	$\mathbf{0.760 \pm 0.007}$	$\mathbf{0.756 \pm 0.007}$	$\mathbf{0.651 \pm 0.008}$	21.09

5.1 Background

This section briefly reviews SVC [3] as background.

5.1.1 Support Vector Domain Description

In domain description, the task is to give a description of a set of objects, which should cover the positive objects and reject the negative ones in the object space [34]. SVDD is a sphere shaped data description. By using a nonlinear transformation to map the data from the input data space into a high-dimensional kernel space, SVDD can obtain a very flexible and accurate data description relying on only a small number of SVs.

Given a data set $\mathscr{X} = \{\mathbf{x}_i \in \mathbb{R}^d | i = 1, \ldots, N\}$ consisting of N data points and a nonlinear transformation ϕ from the input space to a Gaussian kernel feature space,[2] we look for the smallest sphere that encloses most of the mapped data points in the feature space. Using the center $\boldsymbol{\mu}$ and radius R to describe the sphere, we minimize

$$F(R, \boldsymbol{\mu}, \xi_i) = R^2 + C \sum_{i=1}^{N} \xi_i, \tag{85}$$

under the constraints

$$\|\phi(\mathbf{x}_i) - \boldsymbol{\mu}\|^2 \leq R^2 + \xi_i, \forall i = 1, \ldots, N, \tag{86}$$

where the trade-off parameter C gives the trade-off between the volume of the sphere and the accuracy of data description, and $\xi_i \geq 0$ are some slack variables allowing

[2]Although any kernel space works here, as discussed in [3, 34], Gaussian kernels provide more tight contour representations. Therefore, Gaussian kernels with appropriate kernel parameters are used in most SVDD-based approaches, e.g., [3, 5, 21, 22, 34, 41, 42, 50].

for soft boundaries. By introducing the Lagrangian, we have

$$\mathbb{L}(R,\ \boldsymbol{\mu}, \xi_i, \beta_i, \alpha_i) = R^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\beta_i(R^2 + \xi_i - \|\phi(\mathbf{x}_i) - \boldsymbol{\mu}\|^2) - \sum_{i=1}^{N}\alpha_i\xi_i, \tag{87}$$

where $\beta_i \geq 0$, $\alpha_i \geq 0$ are Lagrange multipliers. By eliminating the variables R, $\boldsymbol{\mu}$, ξ_i and α_i, the Lagrangian can be turned into the Wolfe dual form

$$\max_{\beta_i} \mathbb{W} = \sum_{i=1}^{N}\beta_i K(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i,j=1}^{N}\beta_i\beta_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{subject to } \sum_{i=1}^{N}\beta_i = 1,\ 0 \leq \beta_i \leq C, \forall i = 1, \ldots, N, \tag{88}$$

where the dot products $\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$ are replaced by the corresponding Gaussian kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-q\|\mathbf{x}_i - \mathbf{x}_j\|^2)$ with the width parameter q.

According to the values of Lagrange multipliers $\beta_i, i = 1, \ldots, N$, the data points are classified into three types:

- Inner point (IP): $\beta_i = 0$, which lies *inside* the sphere surface.
- Support vector (SV): $0 < \beta_i < C$, which lies *on* the sphere surface.
- Bounded support vector (BSV): $\beta_i = C$, which lies *outside* the sphere surface.

It is obvious that setting $C \geq 1$ will result in no BSV.

The kernel radius function, defined by the Euclidian distance of $\phi(\mathbf{x})$ from $\boldsymbol{\mu}$, is given by

$$\begin{aligned} R(\mathbf{x}) &= \|\phi(\mathbf{x}) - \boldsymbol{\mu}\| \\ &= \sqrt{1 - 2\sum_{i=1}^{N}\beta_i K(\mathbf{x}_i, \mathbf{x}) + \sum_{i,j=1}^{N}\beta_i\beta_j K(\mathbf{x}_i, \mathbf{x}_j)}. \end{aligned} \tag{89}$$

The radius of the sphere is defined as

$$R = \max\{R(\mathbf{x}_i) | \mathbf{x}_i \text{ is a SV, i.e.} 0 < \beta_i < C\}. \tag{90}$$

Ideally, all the SVs should have the same $R(\mathbf{x}_i)$. However, due to the numerical problem, they may be slightly different. A practical strategy is to use their maximum value as the radius. The contours enclosing most of the data points in the data space are defined by the set $\{\mathbf{x}|R(\mathbf{x}) = R\}$. In the original SVDD algorithm [34], data points lying outside the sphere surface, namely BSVs, are taken as outliers, from which outlier detection can be derived.

5.1.2 Support Vector Clustering

The SVC algorithm is directly based on the previously described SVDD by adding a cluster labeling step after obtaining the domain description of the data set [3]. By mapping the data back to the input data space, the sphere surface corresponds to contours in the data space, which separate the data into several components, each enclosing a cluster of data points. Constructing connected components is accomplished by computing an adjacency matrix of the data set according to the sphere.

To this end, an adjacency matrix $A = [A_{ij}]^{N\times N}$ is computed as

$$A_{ij} = \begin{cases} 1 & \text{if } \forall \mathbf{y} \text{ on the line segment connecting } \mathbf{x}_i \text{ and } \mathbf{x}_j, R(\mathbf{y}) \leq R, \\ 0 & \text{otherwise.} \end{cases} \tag{91}$$

Checking the line segment is implemented by sampling a number of points (usually ten points are used) [3]. Clusters are defined as the connected components of the graph induced by the adjacency matrix A. BSVs are unclassified by this procedure since their feature space images always lie outside the sphere. Each of them is directly assigned to the closest cluster [3].

5.2 Position Regularized Support Vector Clustering

One inherent drawback associated with the SVC algorithm is that the constructed sphere is very sensitive to the selection of the trade-off parameter C. Especially, by directly controlling the penalty term $C\sum_{i=1}^{N}\xi_i$ in the objective function (85), the volume of the sphere and the accuracy of data description strongly rely on the value of C.

Figure 14 illustrates the sensitivity of SVC to the value of C on a data set consisting of 200 data points with the best kernel parameter q. The results show that when different values of C are used, different spheres (including SVs, BSVs, and contours) are generated but none of them can accurately discover the four classes of this data distribution.

1. When using a small C, too many BSVs are generated. Some non-outliers are erroneously taken as BSVs, which divide the class containing these data points into at least two clusters, as shown in Fig. 14b.
2. When using a large C, too few BSVs are generated. Some outliers are incorrectly taken as IPs, which connect two classes (that should be separated by these outliers) into one cluster, as shown in Fig. 14e, f.
3. Even using some "median" C, say, $C = 0.2$ (Fig. 14c) and $C = 0.3$ (Fig. 14d), which produces an appropriate number of SVs and BSVs, there are still some erroneously detected SVs and BSVs, leading to the incorrect identification of classes.

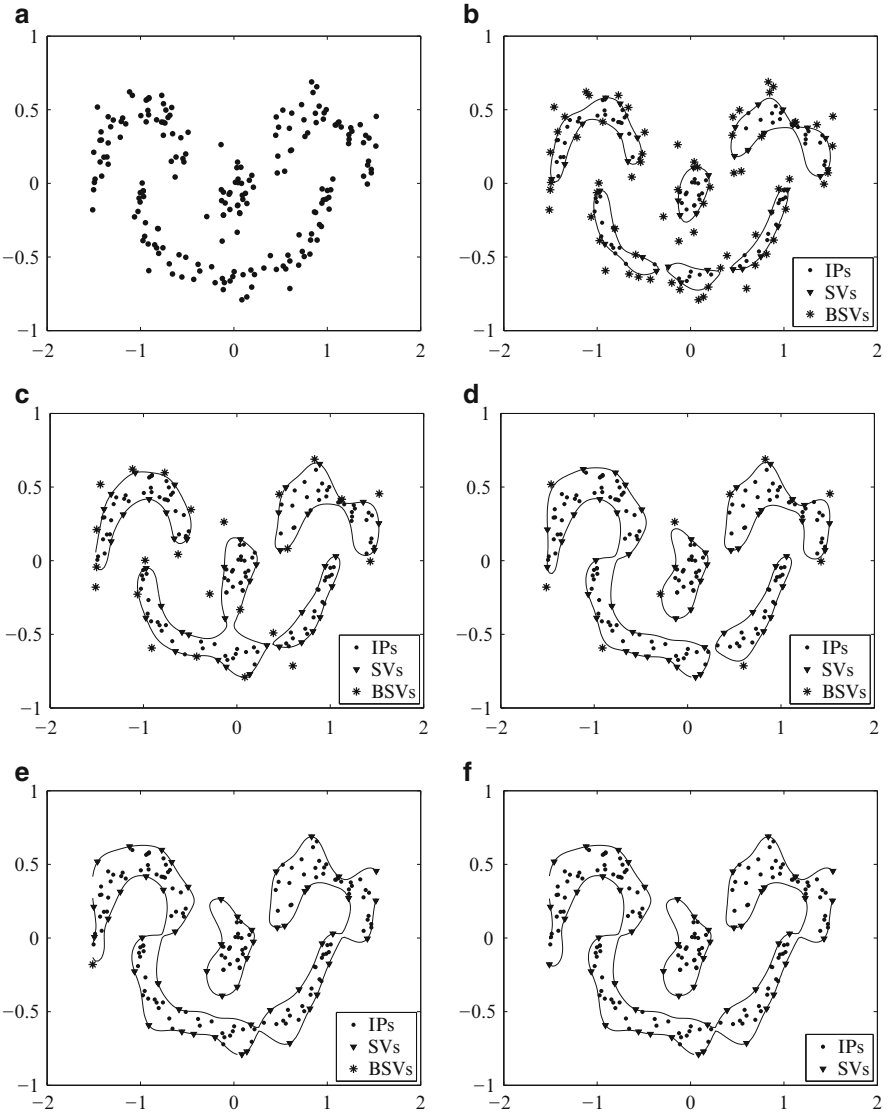


Fig. 14 The spheres generated by SVC on a synthetic data set with the best kernel parameter $q = 13$ using different C. (**a**) The data set of 200 data points. (**b**) The sphere by SVC with $C = 0.1$. (**c**) The sphere by SVC with $C = 0.2$. (**d**) The sphere by SVC with $C = 0.3$. (**e**) The sphere by SVC with $C = 0.4$. (**f**) The sphere by SVC with $C \geq 0.5$

The main reason is that, by using the same global trade-off parameter C for all data points, the likelihood of each data point to be an outlier is taken the same, which is actually not true due to the change of the density distribution or the relation between data points.

According to the above analysis, a distinct trade-off parameter should be used for each data point, reflecting the possibility of each data point to be outside the sphere surface. To address the above issue, one can assign a weighting to each data point so as to represent the confidence of the corresponding data point to be an outlier. In the PSVC algorithm, the weighting W_i is defined to be inversely proportional to the distance between the feature space image of $\mathbf{x}_i$ (i.e., $\phi(\mathbf{x}_i)$) and the mean of feature space images (i.e. $\frac{1}{N}\sum_{j=1}^{N}\phi(\mathbf{x}_j)$).

The underlying rationale is that, in kernel space, if the feature space image of a data point is too distant from the mean of feature space images, which implies that the corresponding data point is relatively isolated from the rest data points in the input data space, this data point should be more likely considered as an outlier. Additionally, since the larger the slack variable ξ_i, the more likely the data point corresponds to an outlier, the corresponding slack variables ξ_i of the isolated data points should be larger than those of the remaining data points. Consequently, if we replace C with the corresponding W_i in the objective function (85), the relatively isolated data points should have smaller weightings W_i. This weighting is called position-based weighting and the corresponding SVC with such weighting is called PSVC.

To compute the position-based weighting, we first compute a kernel distance vector $D^\phi = [D_l^\phi | l = 1, \ldots, N]$ from the kernel matrix K,

$$\begin{aligned} D_l^\phi &= \|\phi(\mathbf{x}_l) - \frac{1}{N}\sum_{j=1}^{N}\phi(\mathbf{x}_j)\|^2 \\ &= K(\mathbf{x}_l, \mathbf{x}_l) + \frac{1}{N^2}\sum_{i,j=1}^{N}K(\mathbf{x}_i, \mathbf{x}_j) - \frac{2}{N}\sum_{j=1}^{N}K(\mathbf{x}_l, \mathbf{x}_j), \quad \forall l = 1, \ldots, N. \end{aligned} \tag{92}$$

The weighting $W_i, \forall i = 1, \ldots, N$ is computed as

$$W_i = \max_{l=1,\ldots,N}\{D_l^\phi\} - D_i^\phi, \quad W_i = \frac{W_i}{\max_{l=1,\ldots,N} W_l}, \tag{93}$$

which is inversely proportional to the distance D_i^ϕ and normalized to be no larger than 1.

Figure 15 illustrates the concept of weighting. The farther away one feature space image $\phi(\mathbf{x}_i)$ from the mean $\frac{1}{N}\sum_{j=1}^{N}\phi(\mathbf{x}_j)$ is, the smaller the corresponding weighting W_i is. The weightings $W_i, i = 1, \ldots, N$ are used to regularize the sphere volume in place of the trade-off parameter C shared by all data points. That is, we minimize the radius

$$F(R, \boldsymbol{\mu}, \xi_i) = R^2 + \sum_{i=1}^{N} W_i \xi_i, \tag{94}$$

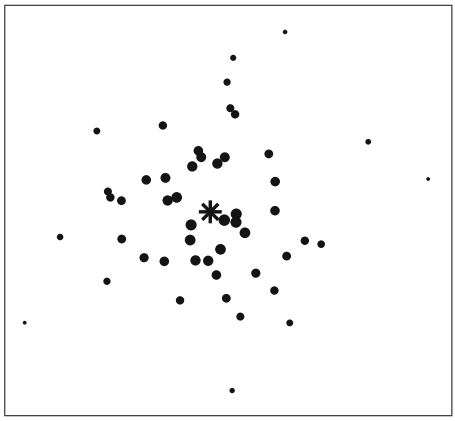


Fig. 15 Illustration of the position-based weighting in kernel space. The mean vector of the kernel space images of data points is shown as a bold start. The size of each data point is marked according to its weighting

under the constraints

$$\|\phi(\mathbf{x}_i) - \boldsymbol{\mu}\|^2 \leq R^2 + \xi_i, \forall i = 1, \ldots, N. \tag{95}$$

Unlike in the original SVC, in the objective function (94), each weighting W_i, respectively, regularizes the likelihood for the corresponding data point $\mathbf{x}_i$ to be an outlier. The smaller W_i is, the larger ξ_i is. The slack variable ξ_i is used to generate soft boundaries and BSV. We will show later that when $\xi_i > 0$, $\mathbf{x}_i$ is taken as a BSV. Therefore, the weighting mechanism adaptively regularizes each data point to be an outlier or not.

The main difference between the weighting mechanism and the one proposed in [51] is that the weighting of [51] is computed according to the Mahalanobis distance of the corresponding factor analyzer; meanwhile the weighting is computed according to the position of the feature space image in the kernel space. The weighting of [51] strongly relies on the results of the mixture of factor analyzers (MFA). Different number of analyzers leads to different weightings, and estimating the number of analyzers is itself a difficult task. In addition, the PSVC algorithm does not require performing MFA to learn the weighting and is therefore more efficient than its counterpart.

By introducing the Lagrangian for (94), we have

$$\mathbb{L}(R, \boldsymbol{\mu}, \xi_i, \beta_i, \alpha_i) = R^2 + \sum_{i=1}^{N} W_i \xi_i - \sum_{i=1}^{N} \beta_i (R^2 + \xi_i - \|\phi(\mathbf{x}_i) - \boldsymbol{\mu}\|^2) - \sum_{i=1}^{N} \alpha_i \xi_i, \tag{96}$$

where $\beta_i \geq 0, \alpha_i \geq 0$ are Lagrange multipliers. Setting to 0 the derivative of $\mathbb{L}$ w.r.t. R, $\boldsymbol{\mu}$ and ξ_i, respectively, leads to

$$\sum_{i=1}^{N} \beta_i = 1, \quad \boldsymbol{\mu} = \sum_{i=1}^{N} \beta_i \phi(\mathbf{x}_i), \quad \beta_i = W_i - \alpha_i. \tag{97}$$

The KKT complementarity conditions result in

$$\alpha_i \xi_i = 0, \quad \beta_i (R^2 + \xi_i - \|\phi(\mathbf{x}_i) - \boldsymbol{\mu}\|^2) = 0. \tag{98}$$

By eliminating the variables R, $\boldsymbol{\mu}$, ξ_i, and α_i, the Lagrangian (96) can be turned into the Wolfe dual form

$$\max_{\beta_i} \mathbb{W} = \sum_{i=1}^{N} \beta_i K(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i,j=1}^{N} \beta_i \beta_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{subject to } \sum_{i=1}^{N} \beta_i = 1,\ 0 \le \beta_i \le W_i, \forall i = 1, \ldots, N. \tag{99}$$

Please notice that, the upper bounds for Lagrange multipliers $\beta_i, i = 1, \ldots, N$ are no longer the same. Instead, each of them is, respectively, controlled by the corresponding weighting. According to (93) and (99), one side effect of the position-based weighting is the increase in the influence of the kernel parameter. This is because not only the construction of kernel space but also the computation of weighting, as well as the upper bounds for Lagrange multipliers, rely on the kernel parameter. However, as we will show later, the most suitable kernel parameter can be estimated based on stability arguments.

According to the values of Lagrange multipliers β_i and the corresponding weightings $W_i, i = 1, \ldots, N$, the data points are classified into three types:

- Inner point (IP): $\beta_i = 0$, which lies *inside* the sphere surface. That is, $\beta_i = 0$ implies that $\xi_i = 0$ according to $\beta_i = W_i - \alpha_i$ and $\alpha_i \xi_i = 0$. Therefore, $\|\phi(\mathbf{x}_i) - \boldsymbol{\mu}\|^2 \le R^2 + \xi_i = R^2$.
- Support vector (SV): $0 < \beta_i < W_i$, which lies *on* the sphere surface. That is, $\beta_i < W_i$ implies that $\xi_i = 0$ according to $\beta_i = W_i - \alpha_i$ and $\alpha_i \xi_i = 0$; and $\beta_i > 0$ implies that $\|\phi(\mathbf{x}_i) - \boldsymbol{\mu}\|^2 = R^2 + \xi_i$ according to $\beta_i (R^2 + \xi_i - \|\phi(\mathbf{x}_i) - \boldsymbol{\mu}\|^2) = 0$. Therefore, $\|\phi(\mathbf{x}_i) - \boldsymbol{\mu}\|^2 = R^2 + \xi_i = R^2$.
- BSV: $\beta_i = W_i$, which lies *outside* the sphere surface. That is, $\beta_i = W_i$ implies that $\xi_i \ge 0$ according to $\beta_i = W_i - \alpha_i$ and $\alpha_i \xi_i = 0$, and that $\|\phi(\mathbf{x}_i) - \boldsymbol{\mu}\|^2 = R^2 + \xi_i$ according to $\beta_i (R^2 + \xi_i - \|\phi(\mathbf{x}_i) - \boldsymbol{\mu}\|^2) = 0$. Therefore, $\|\phi(\mathbf{x}_i) - \boldsymbol{\mu}\|^2 = R^2 + \xi_i \ge R^2$.

The kernel radius function $R(\mathbf{x}_i)$, the radius of the sphere and the contours enclosing most of the data points in the data space are defined the same as the original SVDD. This sphere shaped domain description is termed PSVDD.

After obtaining PSVDD, we follow the same post-processing steps of SVC, i.e., constructing the adjacency matrix A via (91) and defining clusters as the connected components of the graph induced by the adjacency matrix A, leading to the new PSVC algorithm.

Figure 16 illustrates some key concepts of PSVC. By comparing the weightings W_i (Fig. 16a) and the resulting multipliers $\beta_i, i = 1, \ldots, N$ (Fig. 16b), the sphere structure is obtained, which accurately discovers the clusters in the data distribution (Fig. 16c) and therefore generates a correct clustering (Fig. 16d).

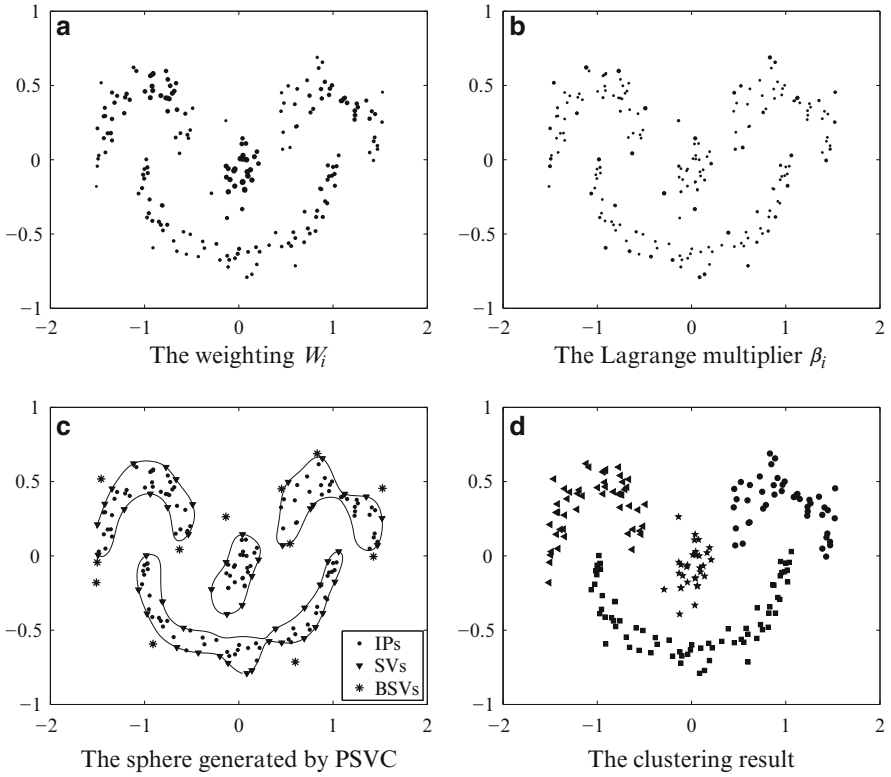


Fig. 16 PSVC demonstration. The same kernel space as Fig. 14 is used, i.e., the same data set with the same kernel parameter q. In (**a**) and (**b**), the size of each data point is marked according to its weighting and Lagrange multiplier, respectively. In (**d**), different clusters are plotted in *different markers*

5.3 Experiments and Applications

In this section, we report the performance improvement achieved by PSVC over its original counterpart SVC.

Seven data sets from the UCI repository [2] are used in comparative experiments, which are summarized in Table 3. In sccts (the synthetic control chart time series data set), four statistical features of each series are used as the feature vector, i.e., the mean, the standard deviation, the skewness, and the kurtosis. The original Wisconsin's breast cancer database consists of 699 samples and after removing 16 samples with missing values, the database consists of 683 samples belonging to 2 classes, benign and malignant tumors. In pendigits (the pen-based recognition of handwritten digit data set), we use a subset consisting of 3000 samples belonging to 3 classes, digit `1`, digit `2`, and digit `3`. The mfeat (the multiple features data set)

Table 3 Summary of the data sets used in comparative experiments

Data set	Number of samples	Dimension	Number of classes
Wine	178	13	3
Glass	214	9	6
Sccts	600	4	6
Wisconsin	683	10	2
Pendigits	3000	16	3
Mfeat	2000	649	10
Isolet	900	617	3

consists of 2000 samples belonging to 10 digit classes. In isolet (the isolated letter speech recognition data set), a subset consisting of 900 samples belonging to classes A, B and C, is used.

In this comparison, the best kernel parameters q for SVC and PSVC, respectively are used. Figure 17 compares the performances of SVC and PSVC when the best kernel parameters q (for SVC and PSVC, respectively) are used on each data set. The clustering results obtained by SVC are plotted as a function of the trade-off parameter C, meanwhile the clustering result obtained by PSVC is plotted as a baseline independent of C since there is no such trade-off parameter in PSVC. The comparative results reveal that, in most cases, no matter how the trade-off parameter is adjusted, SVC is not comparable with PSVC. The exception is that, on the wine data set, when q is set to 0.2 (the best for SVC), SVC with C set to 0.4 can generate higher NMI than PSVC. On data sets such as wine, mfeat, and isolet, when too small values of C are used, e.g., $C = 0.1$ and $C = 0.2$, SVC will generate 0 NMI value, which indicates that the entire data set is partitioned into only one cluster according to the definition of NMI. Additionally, on all data sets, the fact that SVC generates the dramatically changing NMI with different C also reveals that SVC is quite sensitive to the selection of the trade-off parameter.

We also compare the performances of PSVC with three recently improved variants of SVC, including LSVC [51], VM-SVC [38], and DDM-SVC [20]. All the three variants are proposed for improving the clustering performance of SVC. For each algorithm, the best clustering result (i.e., the highest NMI) is reported on each data set by adjusting parameters under the constraint of generating the actual number of clusters.

The comparative results are listed in Table 4, which show that the PSVC algorithm significantly outperforms the compared methods in terms of clustering accuracy (i.e., NMI) and computational time. Especially, on the isolet data set, PSVC has obtained an NMI higher than 0.9, achieving a significantly 37 % improvement over its counterparts. On the Wisconsin data set, although all the compared algorithms have got NMI values as high as 0.94, the PSVC algorithm is still better than the other three algorithms by achieving an NMI value higher than 0.96. These comparative results demonstrate the effectiveness of the PSVC algorithm.

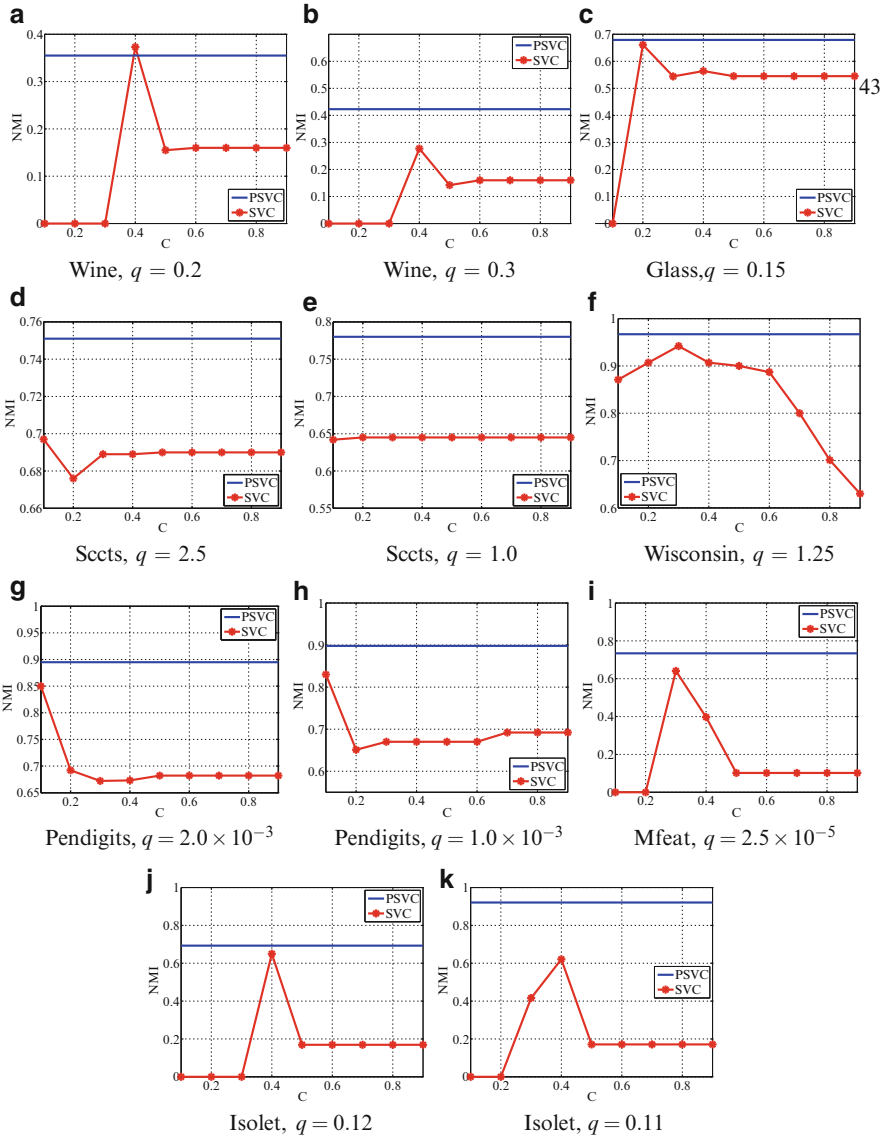


Fig. 17 Comparing the performances of SVC and PSVC using the best kernel parameter q for SVC and PSVC, respectively. For each data set, the clustering results generated by SVC are plotted as a function of C. (**a**) Wine, $q = 0.2$ best for SVC; (**b**) Wine, $q = 0.3$ best for PSVC; (**c**) Glass, $q = 0.15$ best for both; (**d**) Sccts, $q = 2.5$ best for SVC; (**e**) Sccts, $q = 1.0$ best for PSVC; (**f**) Wisconsin, $q = 1.25$ for both; (**g**) Pendigits, $q = 2.0 \times 10^{-3}$ best for SVC; (**h**) Pendigits, $q = 1.0 \times 10^{-3}$ best for PSVC; (**i**) Mfeat, $q = 2.5 \times 10^{-5}$ best for both; (**j**) Isolet, $q = 0.12$ best for SVC; (**k**) Isolet, $q = 0.11$ best for PSVC

Table 4 Comparative results

	SVC [3]		LSVC [51]		VM-SVC [38]		DDM-SVC [20]		PSVC	
Data sets	NMI	Time	NMI	Time	NMI	Time	NMI	Time	NMI	Time
Wine	0.373	9.74×10^0	0.418	8.31×10^0	0.413	4.27×10^0	0.410	3.43×10^0	**0.423**	$\mathbf{0.99 \times 10^0}$
Glass	0.661	3.31×10^1	0.665	2.79×10^1	0.666	1.33×10^1	0.668	1.23×10^1	**0.679**	$\mathbf{3.10 \times 10^0}$
Sccts	0.697	8.11×10^2	0.738	6.43×10^2	0.751	3.34×10^2	0.745	3.23×10^2	**0.780**	$\mathbf{8.65 \times 10^1}$
Wisconsin	0.942	9.07×10^2	0.957	6.65×10^2	0.953	3.92×10^2	0.949	3.74×10^2	**0.967**	$\mathbf{9.18 \times 10^1}$
Pendigits	0.850	7.59×10^4	0.863	4.87×10^4	0.864	4.24×10^4	0.869	4.12×10^4	**0.898**	$\mathbf{3.37 \times 10^4}$
Mfeat	0.641	3.01×10^4	0.671	1.94×10^4	0.672	1.88×10^4	0.667	1.73×10^4	**0.734**	$\mathbf{1.44 \times 10^4}$
Isolet	0.650	2.12×10^3	0.671	1.77×10^3	0.670	1.61×10^3	0.670	1.54×10^3	**0.921**	$\mathbf{1.49 \times 10^3}$

On each data set, we report the highest NMI value by each algorithm, and the computational time in seconds used to achieve such good results

6 Conclusion and Discussion

Nonlinear clustering is one of the most important research problems in data clustering. In this chapter, we have reviewed several nonlinear clustering algorithms from the viewpoints of kernel-based clustering, multi-exemplar model, graph-based method, and SVC. In particular, we have reviewed four nonlinear clustering methods, namely COLL for kernel-based clustering, MEAP, GMPCL, and PSVC. Experimental results in computer visions have shown the effectiveness of these methods.

Despite the great attraction of these methods, there still exist some issues needed to be further discussed and addressed. For instance, for the kernel based methods, namely COLL and PSVC, one challenging issue is to select a suitable kernel function to map the data set from the original data space into a high-dimensional kernel space. Obviously, different kernel transformation would lead to different clustering results no matter what clustering mechanism is employed. Usually but most naively, this issue is addressed by trial and error, which is, however, not applicable due to the lack of prior information. Another strategy in the unsupervised scenario is to select the suitable kernel function by using stability arguments—one chooses the kernel parameter such that the resulting domain descriptions are "most stable" [37, 39].

For the multi-exemplar/multi-prototype methods, they have the same observation, i.e. applicable to only a special case of nonlinearly separable clusters with multiple sub-clusters. This assumption implies that they only focus on a very small portion of a nonlinear problem, where the data points are relatively homogenous. In many applications, such as image categorization, face categorization, multi-font optical character recognition, and handwritten digit classification, each cluster may contain several sub-clusters. For instance, in the natural scene categorization experiments, a scene category often contains multiple "themes," e.g., the street scene may contain themes like "road," "car," "pedestrian," "building," etc. Similarly, in the face categorization experiments, images of the same person in different facial expressions should be taken as in distinct sub-clusters. In the applications of optical character recognition and handwritten digit classification, the cluster representing a letter or a digit could be composed of several sub-clusters, each corresponding to a different style or font. The data containing multiple sub-clusters obviously cannot be represented by a single exemplar/prototype, which however is suitable for the multi-exemplar/multi-prototype model. Therefore, apart from the above reviewed multi-exemplar/multi-prototype methods, some other strategies have been developed for sub-cluster cases. For instance, in [23], Lin et al. developed a cohesion-based self-merging clustering method which combines partitional and hierarchical clustering methods by first partitioning the input data into several small sub-clusters and then merging the sub-clusters based on cohesion in a hierarchical way.

Apart from the above nonlinear clustering methods, there are also some other widely used approaches such as density based clustering (e.g., the well-known

DBSCAN [12]), ensemble clustering (e.g., Link-based ensemble clustering [18]), and clustering by fast search and find of density peaks [29]. The basic idea of density based clustering methods, say, DBSCAN, is to group together data points that are close to each other with high density, and mark as outliers data points that lie alone in low-density regions. The ensemble clustering aggregates several input data clusterings to generate a single output clustering, which can be taken as a space transformation from nonlinear space to linear space. The clustering by fast search and find of density peaks works by using the assumption that cluster centers are characterized by a higher density than their neighbors and by a relatively large distance from points with higher densities, which can be viewed as a combination of both density and distance. From the above discussion, it is clear that nonlinear clustering has been and will continue to be a hot research topic.

References

1. Arbeláez, P., Maire, M., Fowlkes, C., Malik, J.: From contours to regions: an empirical evaluation. In: Proceedings of the 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 2294–2301 (2009)
2. Asuncion, A., Newman, D.: UCI machine learning repository. http://www.ics.uci.edu/~mlearn/MLRepository.html (2007)
3. Ben-Hur, A., Horn, D., Siegelmann, H.T., Vapnik, V.: Support vector clustering. J. Mach. Learn. Res. **2**, 125–137 (2001)
4. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, Berlin (2006)
5. Camastra, F., Verri, A.: A novel kernel method for clustering. IEEE Trans. Pattern Anal. Mach. Intell. **27**(5), 801–805 (2005)
6. Celebi, M.E., Kingravi, H.A.: Deterministic initialization of the k-means algorithm using hierarchical clustering. Int. J. Pattern Recogn. Artif. Intell. **26**(7) 1–23 (2012)
7. Celebi, M.E., Kingravi, H.A., Vela, P.A.: A comparative study of efficient initialization methods for the k-means clustering algorithm. Expert Syst. Appl. **40**, 200–210 (2013)
8. DeSieno, D.: Adding a conscience to competitive learning. In: IEEE International Conference on Neural Networks (1988)
9. Dhillon, I.S., Guan, Y., Kulis, B.: Weighted graph cuts without eigenvectors: a multilevel approach. IEEE Trans. Pattern Anal. Mach. Intell. **29**(11), 1944–1957 (2007). http://www.cs.utexas.edu/users/dml/Software/graclus.html
10. Dueck, D.: Affinity propagation: clustering data by passing messages. Ph.D. thesis, University of Toronto (2009)
11. Ertöz, L., Steinbach, M., Kumar, V.: Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. In: Proceedings of the 3rd SIAM International Conference on Data Mining, pp. 47–58 (2003)
12. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, pp. 226–231 (1996)
13. Fowlkes, C., Martin, D., Malik, J.: Local figure-ground cues are valid for natural images. J. Vis. **7**(8), 2, 1–9 (2007)
14. Frey, B.J., Dueck, D.: Clustering by passing messages between data points. Science **315**, 972–976 (2007)
15. Givoni, I.E., Frey, B.J.: A binary variable model for affinity propagation. Neural Comput. **21**(6), 1589–1600 (2009)

16. http://www.open-video.org. The Open Video Project is managed at the Interaction Design Laboratory, at the School of Information and Library Science, University of North Carolina at Chapel Hill (2015)
17. Hunter, R.S.: Photoelectric color difference meter. J. Opt. Soc. Am. **48**(12), 985–993 (1958)
18. Iam-On, N., Boongoen, T., Garrett, S., Price, C.: A link-based approach to the cluster ensemble problem. IEEE Trans. Pattern Anal. Mach. Intell. **33**(12), 2396–2409 (2011)
19. Kschischang, F.R., Frey, B.J., Loeliger, H.A.: Factor graphs and the sum-product algorithm. IEEE Trans. Inf. Theory **47**(2), 498–519 (2001)
20. Lee, D., Lee, J.: Dynamic dissimilarity measure for support-based clustering. IEEE Trans. Knowl. Data Eng. **22**(6), 900–905 (2010)
21. Lee, J., Lee, D.: An improved cluster labeling method for support vector clustering. IEEE Trans. Pattern Anal. Mach. Intell. **27**(3), 461–464 (2005)
22. Lee, J., Lee, D.: Dynamic characterization of cluster structures for robust and inductive support vector clustering. IEEE Trans. Pattern Anal. Mach. Intell. **28**(11), 1869–1874 (2006)
23. Lin, C.R., Chen, M.S.: Combining partitional and hierarchical algorithms for robust and efficient data clustering with cohesion self-merging. IEEE Trans. Knowl. Data Eng. **17**(2), 145–159 (2005)
24. Liu, M., Jiang, X., Kot, A.C.: A multi-prototype clustering algorithm. Pattern Recogn. **42**, 689–698 (2009)
25. Lyons, M.J., Akamatsu, S., Kamachi, M., Gyoba, J.: Coding facial expressions with gabor wavelets. In: Proceedings of the 3rd IEEE International Conference on Automatic Face and Gesture Recognition, pp. 200–205 (1998)
26. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the 15th Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, pp. 281–297. University of California Press, Berkeley (1967)
27. Maire, M., Arbeláez, P., Fowlkes, C., Malik, J.: Using contours to detect and localize junctions in natural images. In: Proceedings of the 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 1–8 (2008)
28. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: Proceedings of the 8th International Conference on Computer Vision, vol. 2, pp. 416–423 (2001). doi:10.1109/ICCV.2001.937655
29. Rodriguez, A., Laio, A.: Clustering by fast search and find of density peaks. Science **344**(6191), 1492–1496 (2014)
30. Schölkopf, B.: The kernel trick for distances. In: Advances in Neural Information Processing Systems 301–307 (2000)
31. Schölkopf, B., Smola, A., Müller, K.R.: Nonlinear component analysis as a kernel eigenvalue problem. Neural Comput. **10**, 1299–1319 (1998)
32. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press, Cambridge (2004)
33. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **22**(8), 888–905 (2000). http://www.cis.upenn.edu/~jshi/software/
34. Tax, D.M., Duin, R.P.: Support vector domain description. Pattern Recogn. Lett. **20**, 1191–1199 (1999)
35. Truong, B.T., Venkatesh, S.: Video abstraction: a systematic review and classification. ACM Trans. Multimed. Comput. Commun. Appl. **3**(1), 1–37 (2007)
36. Tzortzis, G.F., Likas, A.C.: The global kernel *k*-means algorithms for clustering in feature space. IEEE Trans. Neural Netw. **20**(7), 1181–1194 (2009)
37. von Luxburg, U.: Clustering stability: an overview. Found. Trends Mach. Learn. **2**, 235–274 (2009)
38. Wang, J.S., Chiang, J.C.: A cluster validity measure with outlier detection for support vector clustering. IEEE Trans. Syst. Man Cybern. Part B Cybern. **38**(1), 78–89 (2008)
39. Wang, C.D., Lai, J.H.: Position regularized support vector domain description. Pattern Recogn. **46**, 875–884 (2013)

40. Wang, C.D., Lai, J.H., Zhu, J.Y.: A conscience on-line learning approach for kernel-based clustering. In: Proceedings of the 10th International Conference on Data Mining, pp. 531–540 (2010)
41. Wang, C.D., Lai, J.H., Huang, D.: Incremental support vector clustering. In: Proceedings of the ICDM 2011 Workshop on Large Scale Visual Analytics, pp. 839–846 (2011)
42. Wang, C.D., Lai, J.H., Huang, D., Zheng, W.S.: SVStream: a support vector based algorithm for clustering data streams. IEEE Trans. Knowl. Data Eng. **25**(6), 1410–1424 (2013)
43. Wang, C.D., Lai, J.H., Suen, C.Y., Zhu, J.Y.: Multi-exemplar affinity propagation. IEEE Trans. Pattern Anal. Mach. Intell. **35**(9), 2223–2237 (2013)
44. Wang, C.D., Lai, J.H., Zhu, J.Y.: Conscience online learning: an efficient approach for robust kernel-based clustering. Knowl. Inf. Syst. **31**(1), 79–104 (2012)
45. Wang, C.D., Lai, J.H., Zhu, J.Y.: Graph-based multiprototype competitive learning and its applications. IEEE Trans. Syst. Man Cybern. C Appl. Rev. **42**(6), 934–946 (2012)
46. Weiss, Y., Freeman, W.T.: On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs. IEEE Trans. Inf. Theory **47**(2), 736–744 (2001)
47. Xu, R., Wunsch II, D.: Survey of clustering algorithms. IEEE Trans. Neural Netw. **16**(3), 645–678 (2005). doi:10.1109/TNN.2005.845141
48. Xu, L., Krzyżak, A., Oja, E.: Rival penalized competitive learning for clustering analysis, RBF net, and curve detection. IEEE Trans. Neural Netw. **4**(4), 636–649 (1993)
49. Xu, L., Neufeld, J., Larson, B., Schuurmans, D.: Maximum margin clustering. In: NIPS 17 (2004)
50. Yang, J., Estivill-Castro, V., Chalup, S.K.: Support vector clustering through proximity graph modelling. In: Proceedings of the 9th International Conference on Neural Information Processing (2002)
51. Yankov, D., Keogh, E., Kan, K.F.: Locally constrained support vector clustering. In: Proceedings of the 7th International Conference on Data Mining, pp. 715–720 (2007)
52. Zhou, S.K., Chellappa, R.: Multiple-exemplar discriminant analysis for face recognition. In: ICPR, pp. 191–194 (2004)
53. Zhu, M., Martinez, A.M.: Subclass discriminant analysis. IEEE Trans. Pattern Anal. Mach. Intell. **28**(8), 1274–1286 (2006)