Kernel and graph: Two approaches for nonlinear competitive learning clustering

Abstract

Competitive learning has attracted a significant amount of attention in the past decades. In this paper, we will present two works done by our group which address the nonlinearly separable problem suffered by the classical competitive learning clustering algorithms. They are kernel competitive learning (KCL) and graph-based multi-prototype competitive learning (GMPCL), respectively. In KCL, data points are first mapped from the input data space into a high-dimensional kernel space where the nonlinearly separable pattern becomes linear one. Then the classical competitive learning is performed in this kernel space to generate a cluster structure. To realize on-line learning in the kernel space without knowing the explicit kernel mapping, we propose a prototype descriptor, each row of which represents a prototype by the inner products between the prototype and data points as well as the squared length of the prototype. In GMPCL, a graph-based method is employed to produce an initial, coarse clustering. After that, a multi-prototype competitive learning is introduced to refine the coarse clustering and discover clusters of an arbitrary shape. In the multi-prototype competitive learning, to generate cluster boundaries of arbitrary shapes, each cluster is represented by multiple prototypes, whose subregions of the Voronoi diagram together approximately characterize one cluster of an arbitrary shape. Moreover, we introduce some extensions of these two approaches with experiments demonstrating their effectiveness.

Keywords

competitive learning, clustering, nonlinearly separable, kernel, graph

1 Introduction

Competitive learning has received a significant amount of attention in the past decades. Due to its adaptive on-line learning, it has been widely applied in the fields of data clustering, vector quantization, RBF net learning, shape detection, discrete-valued source separation, Markov model identification, component analysis, scheduling, etc. Among them, clustering analysis is still one of the most active fields. Despite significant success of competitive learning in data clustering, most competitive learning clustering algorithms assume that data are linearly separable, which is unfortunately not always held in real-world applications. The square distance based winner selection only generates a list of convex regions, each for one cluster. These convex regions can separate linear clusters (i.e., clusters with convex boundaries) as shown in Fig. 1(a); however, they fail to identify nonlinear clusters (i.e., clusters with concave boundaries) as shown in Fig. 1(b). Note that real-world data do not always have convex boundaries. This motivates the development of nonlinear competitive learning to solve this problem. In this paper, two approaches are presented to realize nonlinear competitive learning clustering, one of which is from the viewpoint of kernel clustering and the other is based on the graph method.

Kernel method is one of the most popular methodologies for nonlinear clustering. By mapping data points from the input data space into a linearly separable kernel space via a kernel mapping, the nonlinear clustering can be easily realized. Recently, a few kernel versions of competitive learning have been developed. However, these works are confined to some special cases. For instance, two methods in Refs. [14,15] mapped the data into normalization-invariant kernel space (i.e., on a unit hypersphere \( \| \phi(x) \|_2^2 = 1 \)) before performing clustering. The constraint of normalization-invariant kernel helps avoid the computational challenge...
of on-line learning without explicit mapping $\phi$. Unfortunately, most kernels do not satisfy this constraint such as polynomial kernel $\kappa(x, z) = ((x, z) + \alpha)^\beta$, all-subsets kernel $\kappa(x, z) = \prod_{i=1}^d (1 + x_i z_i)$ and ANOVA kernel $\kappa(x, z) = \sum_{1 \leq i_1 < ... < i_p \leq d} \prod_{j=1}^p x_{i_j} z_{i_j}$, while normalization would greatly re-distribute the data [13]. Thus, they are only allowed in a few kernels (e.g., Gaussian kernel). The other method [16] suggested updating the distance between prototypes and data points. However, some of the crucial components of competitive learning can not be realized. In particular, the convergence criterion that is critical for measuring the convergence cannot be computed. This leads to some loss of the advantages of competitive learning, e.g., convergence guarantee.

We provide a complete solution for nonlinear competitive learning and propose kernel competitive learning (KCL). We first obtain a kernel model of competitive learning by mapping the data into kernel space before performing on-line learning, which is able to identify nonlinear clusters. However, it is practically infeasible to realize such a kernel model when the mapping function is unknown, since the winning prototype is updated in on-line mode rather than in batch like that in kernel $k$-means [17]. To tackle this problem, we develop an efficient and elegant computational method by proposing a new prototype representation termed prototype descriptor, through which the computational challenge of the four crucial components including clustering initialization, winner selection, winner updating and convergence criterion is dexterously handled. Therefore, a complete nonlinear competitive learning clustering method termed kernel competitive learning (KCL) is proposed.

Graph-based method is another popular approach for nonlinear clustering [18]. In the graph-based method, a graph is first constructed, in which data points are represented by nodes, and the proximity between two data points is represented by the weight of the edge between the corresponding nodes. Then the notion of useful links between data points or the eigenstructure of the proximity matrix is used to generate clusters of an arbitrary shape. For instance, in the shared nearest neighbor (SNN) clustering method [19], the weight between two data points (nodes) is computed as the number of shared neighbors between nodes given that the nodes are connected (i.e., having useful link). The core-points are defined according to some prespecified parameters $MinPts$ and $Eps$ and used to obtain clusters of arbitrary shapes. In spectral clustering [20], the clustering problem is transferred to a graph partitioning problem where the goal is to obtain the prespecified number of components of the graph with minimizing some objective function such as normalized cut. To realize this, the eigenstructure of the proximity matrix is obtained, where the eigenvectors with the smallest eigenvalues are used to construct a partitioning. Each component obtained corresponds to a major structure of the graph, such that these components (i.e., clusters) can be of arbitrary shapes revealing the real structures. However, this type of method needs to prespecify either some parameters (e.g., $MinPts$ and $Eps$ in SNN) or the number of real clusters (e.g., in spectral clustering).

Based on the graph method, we propose a novel nonlinear clustering algorithm named graph-based multi-prototype competitive learning (GMPCL). This algorithm employs a graph-based method to generate an initial, coarse clustering. Multi-prototype competitive learning is then performed to refine the clustering and identify clusters of an arbitrary shape. Therefore, the proposed approach consists of two phases, namely, graph-based initial clustering and multi-prototype competitive learning. In the multi-prototype competitive learning, to generate cluster boundaries of arbitrary shapes, each cluster is represented by multiple prototypes, whose subregions of the Voronoi diagram together
approximately characterize one cluster of an arbitrary shape.

The remainder of the paper is organized as follows. Section 2 introduces the classical competitive learning clustering algorithm with describing its linear clustering limitation. The KCL algorithm, as well as its extension, is described in Section 3. In Section 4, we describe the GMPCL algorithm. We conclude our paper in Section 5.

## 2 Classical competitive learning

Given a dataset $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ of $n$ data points in a space $\mathbb{R}^d$ and the number of clusters $c$, the goal of classical competitive learning (CCL) is to find $c$ optimal prototypes $\{\mu_k^{\text{opt}} : k = 1, 2, \ldots, c\}$ by minimizing the objective function

$$J(\mu_1, \mu_2, \ldots, \mu_c) = \sum_{i=1}^{n} \| x_i - \mu_{\nu_i} \|^2$$

with $\nu_i$ denoting the cluster label of $x_i$ as computed by Eq. (2), such that an optimal set of clusters $\{C_1, C_2, \ldots, C_c : x_i \in C_{\nu_i}, i = 1, 2, \ldots, n\}$ can be generated. To this end, the first step is to randomly initialize prototypes $\{\mu_k : k = 1, 2, \ldots, c\}$, which are iteratively updated as follows. In the $t$th iteration, for each randomly taken data point $x_i \in \mathcal{X}$, select a winning prototype $\mu_{\nu_i}$ via

$$\nu_i = \arg \min_{k=1,2,\ldots,c} \| x_i - \mu_k \|^2,$$

and update the winner with learning rate $\eta_t$ in $(0, 1)$

$$\mu_{\nu_i} \leftarrow \mu_{\nu_i} + \eta_t (x_i - \mu_{\nu_i}).$$

The procedure continues until either all prototypes converge or the number of iterations reaches a prespecified value $t_{\text{max}}$. Algorithm 1 summaries the CCL method.

To find the globally optimal clustering, the crucial components of CCL include the following four aspects.

1) Random initialization of prototypes. A rational choice is to initialize prototypes in the way of randomly weighted linear combination of data points. That is,

$$\mu_k = \sum_{i=1}^{n} A_{k,i} x_i, \forall k = 1, 2, \ldots, c,$$

where $A$ is a randomly generated weight matrix,

$$A = \begin{pmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,n} \\ A_{2,1} & A_{2,2} & \cdots & A_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{c,1} & A_{c,2} & \cdots & A_{c,n} \end{pmatrix}$$

with each row summing to 1, i.e., $\sum_{i=1}^{n} A_{k,i} = 1, \forall k = 1, 2, \ldots, c$.

### Algorithm 1 Classical Competitive Learning (CCL)

**Input:** dataset $\mathcal{X} = \{x_i : i = 1, 2, \ldots, n\}$, $c$, $\{\eta_t\}$, $t_{\text{max}}$.

**Output:** clusters $\{C_k : k = 1, 2, \ldots, c\}$.

Randomly initialize $\{\mu_k : k = 1, 2, \ldots, c\}$, set $t = 0$.

**repeat**

1. Save the current prototypes $\{\mu_k\}$ before this iteration, i.e., $\{\hat{\mu}_k\} = \{\mu_k\}$.
2. set $t = t + 1$, randomly re-sort $\{x_i : i = 1, 2, \ldots, n\}$.
3. **for** $i = 1, 2, \ldots, n$ **do**
   - Select the winning prototype $\mu_{\nu_i}$ of $x_i$ by Eq. (2).
   - Update $\mu_{\nu_i}$ with learning rate $\eta_t$ by Eq. (3).
4. **end for**

5. **until** $\sum_{k=1}^{c} \| \mu_k - \hat{\mu}_k \|^2 \leq \epsilon$ or $t \geq t_{\text{max}}$

6. Assign $x_i$ to the cluster $C_{\nu_i}$, where $\nu_i$ is computed by Eq. (2), $\forall i = 1, 2, \ldots, n$.

**2) Competition to select the winning prototype corresponding to the input object by Eq. (2).**

**3) Learning to update the winner by Eq. (3) with learning rate $\{\eta_t\}$ satisfying conditions [21]:**

$$\lim_{t \to \infty} \eta_t = 0, \quad \sum_{t=1}^{\infty} \eta_t = \infty, \quad \sum_{t=1}^{\infty} \eta_t^2 < \infty.$$  \hspace{1cm} (6)

A practical choice of $\eta$ is $\eta = \text{const}/t$, where “const” is some small constant, i.e., 1.

**4) Iteration stopping criteria, i.e., $\sum_{k=1}^{c} \| \mu_k - \hat{\mu}_k \|^2 \leq \epsilon$ or $t \geq t_{\text{max}}$.**

It can be shown [21] that competitive learning given by the four components does indeed converge to the global optimum with probability one.

**Proposition 1** Given appropriate initialization, CCL converges to the global optimum with probability one, i.e., $P\left(\{\mu_1, \mu_2, \ldots, \mu_c\} = \{\mu_1^{\text{opt}}, \mu_2^{\text{opt}}, \ldots, \mu_c^{\text{opt}}\}\right) = 1$, where $\{\mu_1^{\text{opt}}, \mu_2^{\text{opt}}, \ldots, \mu_c^{\text{opt}}\} = \arg \min J(\mu_1, \mu_2, \ldots, \mu_c)$.

The initialization problems such as estimating the number of clusters and allocating their initial prototypes can be effectively handled by various mechanisms. For instance, adding conscience to CCL could avoid the neuron underutilization [22], and employing rival penalization can help get rid of redundant prototypes [3]. Thus, the initialization condition for the convergence of CCL is easily satisfied.

However, from the rule for selecting a winning prototype, i.e., Eq. (2), a list of convex regions have been generated, each for one cluster, as shown in Fig. 1. Thus, it is well-known that CCL has a limitation to only linearly separable datasets. In our work, two approaches are presented to realize nonlinear competitive learning clustering: one from the viewpoint of kernel clustering and the other based on the graph method.
3 Kernel competitive learning

3.1 Problem formulation

By mapping the unlabeled dataset \( \mathcal{X} = \{x_1, x_2, \ldots, x_n\} \) of \( n \) data points from \( \mathbb{R}^d \) into a kernel space \( \mathcal{Y} \) via a mapping \( \phi \), we wish to find an assignment of each data point to one of \( c \) clusters, such that in the kernel space \( \mathcal{Y} \), the within-cluster similarity is high and the between-cluster one is low. That is, we seek a map \( \nu : \mathcal{X} \rightarrow \{1, 2, \ldots, c\} \) to optimize [13]

\[
\nu = \arg \min_{\nu} \left\{ \sum_{i,j; \nu_i = \nu_j} \| \phi(x_i) - \phi(x_j) \|^2 - \lambda \sum_{i,j; \nu_i \neq \nu_j} \| \phi(x_i) - \phi(x_j) \|^2 \right\}, \tag{8}
\]

where \( \lambda > 0 \) is some parameter, and we use the short notation \( \nu_i = \nu(x_i) \). In this paper, \( \nu^{-1}(k) \) denotes the indices of all the data points that are assigned to the \( k \)th cluster.

**Theorem 1** The optimization criterion Eq. (8) is equivalent to the criterion

\[
\nu = \arg \min_{\nu} \sum_{i=1}^{n} \| \phi(x_i) - \mu_{\nu_i} \|^2, \tag{9}
\]

where \( \mu_k \) is the mean of data points assigned to cluster \( k \),

\[
\mu_k = \frac{1}{|\nu^{-1}(k)|} \sum_{i \in \nu^{-1}(k)} \phi(x_i), \quad \forall k = 1, 2, \ldots, c, \tag{10}
\]

and \( \nu_i \) satisfies

\[
\nu_i = \arg \min_{k=1,2,\ldots,c} \| \phi(x_i) - \mu_k \|^2, \quad \forall i = 1, 2, \ldots, n. \tag{11}
\]

**proof** See Ref. [13]. \( \square \)

Thus, the goal of kernel clustering is to solve the optimization problem in Eq. (9). The objective term

\[
\sum_{i=1}^{n} \| \phi(x_i) - \mu_{\nu_i} \|^2 \tag{12}
\]

is known as the distortion error [21]. 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 dataset, since the number of all possible partitions of a dataset grows exponentially with the number of data points. Hence, efficient algorithms are required.

In practice, the mapping function \( \phi \) is often not known or hard to obtain, and the dimensionality of \( \mathcal{Y} \) is quite high. The feature space \( \mathcal{Y} \) is characterized by the kernel function \( \kappa \) and corresponding kernel matrix \( K \) [13].

**Definition 1** A kernel is a function \( \kappa \), such that \( \kappa(x, z) = \langle \phi(x), \phi(z) \rangle \) for all \( x, z \in \mathcal{X} \), where \( \phi \) is a mapping from \( \mathcal{X} \) to an (inner product) feature space \( \mathcal{Y} \). A kernel matrix is a square matrix \( K \in \mathbb{R}^{n \times n} \) such that \( K_{i,j} = \kappa(x_i, x_j) \) for some \( x_1, x_2, \ldots, x_n \in \mathcal{X} \) and some kernel function \( \kappa \).

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

3.2 Kernel competitive learning

For efficiently performing on-line learning in kernel space with only kernel matrix \( K \), we proposed an On-Line Learning (OLL) framework [23,24], which is based on a new prototype representation termed prototype descriptor \( W^\phi \). The rows of \( W^\phi \) represent prototypes as the inner products between a prototype and the feature space images of data points, as well as the squared length of the prototype [23,24].

**Definition 2 (Prototype descriptor)** A prototype descriptor is a real-valued matrix \( W^\phi \), such that the \( k \)th row represents prototype \( \mu_k \) by

\[
W^\phi_{k,i} = \langle \mu_k, \phi(x_i) \rangle, \quad \forall i = 1, 2, \ldots, n, \quad W^\phi_{k,n+1} = \langle \mu_k, \mu_k \rangle, \tag{13}
\]

i.e.,

\[
W^\phi = \begin{pmatrix}
\langle \mu_1, \phi(x_1) \rangle & \cdots & \langle \mu_1, \phi(x_n) \rangle & \langle \mu_1, \mu_1 \rangle \\
\langle \mu_2, \phi(x_1) \rangle & \cdots & \langle \mu_2, \phi(x_n) \rangle & \langle \mu_2, \mu_2 \rangle \\
\vdots & \vdots & \vdots & \vdots \\
\langle \mu_c, \phi(x_1) \rangle & \cdots & \langle \mu_c, \phi(x_n) \rangle & \langle \mu_c, \mu_c \rangle
\end{pmatrix}. \tag{14}
\]

With this definition, the four components of on-line learning in kernel space can be realized as follows [23,24].

**Theorem 2 (Initialization)** The random initialization can be realized by

\[
W^\phi_{1:1:n} = AK, \quad W^\phi_{1:n+1} = \text{diag}(AKA^T), \tag{15}
\]

where \( \text{diag}(M) \) denotes the main diagonal of a matrix \( M \) and the positive matrix \( A = [A_{k,i}]_{n \times n} \) has the form

\[
A_{k,i} = \begin{cases}
1 & \text{if } i \in \nu^{-1}(k), \\
0 & \text{otherwise}.
\end{cases} \tag{16}
\]

That is, the matrix \( A \) reflects the initial assignment \( \nu \).

**Proof** Assume the assignment is randomly initialized as \( \nu \). Substitute the computation of the prototypes Eq.
Thus, we obtain the initialization of $W^\phi$ as Eq. (15). The proof is finished. \hfill \blacksquare

**Theorem 3 (Winner selection rule)** The winner selection rule can be realized by

$$
\nu_i = \arg\min_{k=1,2,...,c} \{ K_{i,i} + W^\phi_{k,n+1} - 2W^\phi_{k,i} \}.
$$

**Proof** Consider the winner selection rule, i.e., Eq. (2), one can get

$$
\nu_i = \arg\min_{k=1,2,...,c} \{ \| \phi(x_i) - \mu_k \|^2 \}
= \arg\min_{k=1,2,...,c} \{ (K_{i,i} + W^\phi_{k,n+1} - 2W^\phi_{k,i}) \}.
$$

Thus, we get the formula required. \hfill \blacksquare

**Theorem 4 (On-line winner updating rule)** The on-line winner updating rule can be realized by

$$
W^\phi_{\nu_i,j} \leftarrow \begin{cases} 
(1 - \eta_i)W^\phi_{\nu_i,j} + \eta_i K_{i,j}, & \text{if } j = 1, 2, \ldots, n, \\
(1 - \eta_i)^2 W^\phi_{\nu_i,j} + \eta_i^2 K_{i,i} + 2(1 - \eta_i)\eta_i W^\phi_{\nu_i,i}, & \text{if } j = n + 1. 
\end{cases}
$$

**Proof** Although we do not know exactly the expression of $\mu_{\nu_i}$, however, we can simply take $\mu_{\nu_i}$ as a symbol of this prototype and denote its updated one as $\hat{\mu}_{\nu_i}$. Substitute the on-line winner updating rule Eq. (3) to the winning prototype $W^\phi_{\nu_i,j}$, we have

$$
W^\phi_{\nu_i,j} \leftarrow \begin{cases} 
(\hat{\mu}_{\nu_i} + \eta_i(\phi(x_i) - \mu_{\nu_i}), \phi(x_j)) & \forall j = 1, 2, \ldots, n \\
(1 - \eta_i)\hat{\mu}_{\nu_i} + \eta_i \phi(x_i), \phi(x_j)) & \forall j = n + 1. 
\end{cases}
$$

Then we get the on-line winner updating rule as Eq. (21).

It is a bit complicated to compute the convergence criterion without explicit expression of $\{\mu_1, \mu_2, \ldots, \mu_c\}$. Notice that, in one iteration, each point $\phi(x_i)$ is assigned to one and only one winning prototype. Let array $\pi^k = [\pi^k_1, \pi^k_2, \ldots, \pi^k_{m_k}]$ store the indices of $m_k$ ordered points assigned to the $k$th prototype in one iteration. For instance, if $\phi(x_1), \phi(x_3), \phi(x_8), \phi(x_20)$, $\phi(x_{15})$ are $5$ ordered points assigned to the 2nd prototype in the 7th iteration, then the index array of the 2nd prototype is $\pi^2 = [\pi^2_1, \pi^2_2, \ldots, \pi^2_{m_2}] = [1, 32, 8, 20, 15]$ with $\pi^2_1 = 1, \pi^2_2 = 32, \pi^2_3 = 8, \pi^2_4 = 20, \pi^2_5 = 15$ and $m_2 = 5$. The following lemma formulates the cumulative update of the $k$th prototype based on the array $\pi^k = [\pi^k_1, \pi^k_2, \ldots, \pi^k_{m_k}]$.

**Lemma 1** In the $t$th iteration, the relationship between the updated prototype $\mu_k$ and the old $\mu_k$ is

$$
\mu_k = (1 - \eta_t)^m k \hat{\mu}_k + \sum_{i=1}^{m_k} (1 - \eta_t)^{m_k - l} \eta_t \phi(x_{\pi^k_i}),
$$

where array $\pi^k = [\pi^k_1, \pi^k_2, \ldots, \pi^k_{m_k}]$ 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 Eq. (24) is true for $m_k = 1$ directly from Eq. (3),

$$
\mu_k = \hat{\mu}_k + \eta_t (\phi(x_{\pi^k_1}) - \mu_k)
= (1 - \eta_t)^1 \hat{\mu}_k + \sum_{i=1}^{1} (1 - \eta_t)^0 \eta_t \phi(x_{\pi^k_1}).
$$

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

$$
\mu_k = (1 - \eta_t)^m \hat{\mu}_k + \sum_{i=1}^{m} (1 - \eta_t)^{m - l} \eta_t \phi(x_{\pi^k_i}).
$$

Then for $m_k = m + 1$, i.e., the $(m + 1)$th point, from Eq. (3) we have

$$
\mu_k = \hat{\mu}_k + \eta_t (\phi(x_{\pi^k_{m_k}}) - \mu_k)
= (1 - \eta_t)^m \hat{\mu}_k + \sum_{i=1}^{m} (1 - \eta_t)^{m - l} \eta_t \phi(x_{\pi^k_i}) + \eta_t \phi(x_{\pi^k_{m_k}})
= (1 - \eta_t)^{m + 1} \hat{\mu}_k + \sum_{i=1}^{m+1} (1 - \eta_t)^{m+1 - l} \eta_t \phi(x_{\pi^k_i}).
$$
This expression shows that Eq. (24) 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

\[
e^\phi = \sum_{k=1}^{c} \left( \frac{1}{1-\eta_k} \right)^2 W_{k,n+1}^\phi - \frac{\eta_k}{1-\eta_k} \sum_{i=1}^{m_k} \frac{\phi(x^+_{\pi_k^i})}{1-\eta_k}, \]

\[
+ \eta_k \sum_{k=1}^{c} \sum_{h=1}^{m_k} \sum_{l=1}^{m_k} \frac{K_{\pi_k^h,\pi_k^l}}{(1-\eta_k)^{h+l}} W_{h,l}^\phi, \]

\[
+ 2\eta_k \sum_{k=1}^{c} \left( 1 - \frac{1}{(1-\eta_k)^{m_k}} \right) \sum_{l=1}^{m_k} \frac{W_{k,l}^\phi}{(1-\eta_k)} . \tag{28}
\]

**Proof** According to Lemma 1, the old \( \hat{\mu}_k \) can be retained from the updated \( \mu_k \) as

\[
\hat{\mu}_k = \frac{\mu_k}{(1-\eta_k)^{m_k}} - \eta_k \sum_{i=1}^{m_k} \frac{\phi(x^+_{\pi_k^i})}{1-\eta_k}, \tag{29}
\]

Substitute it to \( e^\phi = \sum_{k=1}^{c} \left\| \mu_k - \hat{\mu}_k \right\|^2 \), we have

\[
e^\phi = \sum_{k=1}^{c} \left\| \mu_k - \left( \frac{\mu_k}{(1-\eta_k)^{m_k}} - \eta_k \sum_{i=1}^{m_k} \frac{\phi(x^+_{\pi_k^i})}{1-\eta_k} \right) \right\|^2
\]

\[
= \sum_{k=1}^{c} \left( 1 - \frac{1}{(1-\eta_k)^{m_k}} \right)^2 \langle \mu_k, \mu_k \rangle
\]

\[
+ \eta_k \sum_{k=1}^{c} \sum_{h=1}^{m_k} \sum_{l=1}^{m_k} \frac{\langle \phi(x^+_{\pi_k^h}), \phi(x^+_{\pi_k^l}) \rangle}{(1-\eta_k)^{h+l}}
\]

\[
+ 2\eta_k \sum_{k=1}^{c} \left( 1 - \frac{1}{(1-\eta_k)^{m_k}} \right) \sum_{l=1}^{m_k} \langle \mu_k, \phi(x^+_{\pi_k^l}) \rangle . \tag{30}
\]

Thus, \( e^\phi \) can be computed by Eq. (28). This ends the proof.

Algorithm 2 summarizes the schedule of KCL, while Fig. 2 illustrates its key procedure. Figure 3 demonstrates the nonlinear separation ability of KCL in clustering the nonlinearly separable two moons dataset. A satisfactory clustering result is generated.

**Algorithm 2** Kernel Competitive Learning (KCL)

**Input:** kernel matrix \( K \in \mathbb{R}^{n \times n} \), \( c \), \( \{\eta_i\} \), \( \epsilon, t_{\text{max}} \).

**Output:** clusters \( \{C_k : k = 1, 2, \ldots, c\} \).

Randomly initialize prototype descriptor \( W^\phi \in \mathbb{R}^{c \times (n+1)} \) via Eq. (15), set \( t = 0 \).

repeat

Set \( t = t + 1 \), get random permutation \( \{I_1, I_2, \ldots, I_n\} \), \( I_i \in \{1, 2, \ldots, n\}, I_i \neq I_j, i \neq j \), and initialize \( c \) empty index arrays.

for \( l = 1, 2, \ldots, n \) do

Select the winning prototype \( \nu_i \) of the \( i \)th sample \( (i = I_l) \) by Eq. (19), and append \( l \) to the \( \nu_i \)th index array.

Update the winner, i.e., the \( \nu_i \)th row of \( W^\phi \) with learning rate \( \eta_k \) by Eq. (21).

end for

Compute \( \epsilon^\phi \) via Eq. (28).

until \( \epsilon^\phi \leq \epsilon \) or \( t \geq t_{\text{max}} \)

Assign \( x_i \) to the cluster \( C_{\nu_i} \), where \( \nu_i \) is computed by Eq. (19), \( \forall i = 1, 2, \ldots, n \).

3.3 Extension of KCL and experiment evaluation

Apart from the capability of partitioning nonlinearily separable datasets, another advantage of KCL lies in its potential of utilizing various mechanisms to eliminate the sensitivity in ill-initialization and to automatically estimate the cluster number. For instance, by adopting the conscience mechanism, which controls the winning prototype selection via the winning frequency, the under-utilized problem [22] can be easily tackled. By utilizing rival penalization, which eliminates redundant prototypes via not only moving the winning prototype towards the input but also delearning the rival at a smaller delearning rate, it is easy to realize nonlinear clustering and automatic cluster number estimate simultaneously.

In our work [23,24], the conscience mechanism is adopted to tackle the under-utilized problem suffered from ill-initialization. The key idea is to introduce the

---

**Fig. 2** Illustration of KCL. The function \( \phi \) embeds the data into a feature space where the original nonlinear pattern becomes linear. Then competitive learning is performed in this linear feature space. The new inputs are denoted in blue \( \mathbf{x'} \) and \( \phi(\mathbf{x'}) \) while the red arrow plots the update of the winning prototype, i.e., \( \mathbf{W}_c, \phi(\mathbf{x'}) \) and \( \mathbf{W}_c, \phi(\mathbf{x'}) \). Note that, the linear separator in feature space is actually nonlinear in input space, so is the update of the winner.
winning frequency of each prototype, and the winning prototype \( \nu_i \) is selected by taking the winning frequency into consideration for each input data point \( \phi(x) \). That is,

\[
\nu_i = \arg\min_{k=1,2,\ldots,c} \{ f_k(K_{i,i} + W_{k,n+1}^\phi - 2W_{k,i}^\phi) \}. 
\]

The winning frequencies \( \{f_1, f_2, \ldots, f_c\} \) are updated as follows:

\[
n_{\nu_i} \leftarrow n_{\nu_i} + 1, \quad f_k = \frac{n_k}{\sum_{i=1}^c n_{\nu_i}}, \quad \forall k = 1, 2, \ldots, c, 
\]

where \( n_k \) denotes the cumulative winning number of the \( k \)th prototype. This approach is called conscience on-line learning (COLL) in our previous work [23,24].

To demonstrate the effectiveness of COLL in kernel-based clustering, we select four widely tested digit datasets, including Pen-based recognition of handwritten digit dataset (Pendigits), Multi-feature digit dataset (Mfeat), USPS [25] and MNIST [26]. The first two datasets are from UCI repository [27]. Table 1 summarizes the properties of the four datasets, as well as the \( \sigma \) used in constructing the Gaussian kernel matrix. Note that appropriate kernel selection is out of the scope of this paper, and we only try the Gaussian kernel with some appropriate \( \sigma \) value according to the distance matrix \( X = [\| x_i - x_j \|]_{n \times n} \).

Table 1: Summary of digit datasets. \( n \) is the number of data points; \( c \) is the number of classes; \( d \) is the dimensionality; “balanced” means whether all classes are of the same size. The \( \sigma \) is fixed for all compared kernel based methods.

<table>
<thead>
<tr>
<th>dataset</th>
<th>( n )</th>
<th>( c )</th>
<th>( d )</th>
<th>balanced</th>
<th>( \sigma )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pendigits</td>
<td>10992</td>
<td>10</td>
<td>16</td>
<td>( \times )</td>
<td>60.53</td>
</tr>
<tr>
<td>Mfeat</td>
<td>2000</td>
<td>10</td>
<td>649</td>
<td>( \checkmark )</td>
<td>809.34</td>
</tr>
<tr>
<td>USPS</td>
<td>11000</td>
<td>10</td>
<td>256</td>
<td>( \checkmark )</td>
<td>1286.70</td>
</tr>
<tr>
<td>MNIST</td>
<td>5000</td>
<td>10</td>
<td>784</td>
<td>( \checkmark )</td>
<td>2018.30</td>
</tr>
</tbody>
</table>

Figure 4 plots the distortion error as a function of the preselected number of clusters obtained by COLL, kernel \( k \)-means [17] and global kernel \( k \)-means [28]. It is obvious that, COLL achieves the smallest distortion errors among the compared methods. Especially, it outperforms the latest development of kernel \( k \)-means, i.e., global kernel \( k \)-means on all the four datasets. It should be pointed out that, ideally, the globally minimal distortion error, e.g., the one obtained by testing all the possible assignments \( \nu \) with the best one selected, should have monotonically decreased as the number of clusters increases. However, since all the compared methods are just heuristic methods that can achieve local optima, it is not necessary to be monotonically decreasing.

4 Graph-based multi-prototype competitive learning

The proposed GMPCL approach consists of two phases, namely, graph-based initial clustering (Subsection 4.1) and multi-prototype competitive learning (Subsection 4.2) [29].

4.1 Graph-based initial clustering

Given a dataset \( \mathcal{X} = \{ x_1, x_2, \ldots, x_n \} \) of \( n \) points in \( \mathbb{R}^d \), the first step of the graph-based algorithm is to construct a graph \( G^c = (\mathcal{V}, A, e) \). The vertex set \( \mathcal{V} \) contains one node for each sample in \( \mathcal{X} \). The affinity matrix \( A = [A_{ij}]_{n \times n} \) is defined as

\[
A_{ij} \triangleq \begin{cases} 
\exp(-\| x_i - x_j \|^2), & \text{if } x_i \in \mathcal{N}_k(x_j) \land x_j \in \mathcal{N}_k(x_i), \\
0, & \text{otherwise}, 
\end{cases} 
\]

where \( \mathcal{N}_k(x_i) \) denotes the set consisting of \( k \) nearest neighbors of \( x_i \). The vertex energy vector \( e = [e_1, e_2, \ldots, e_n]^T \) is defined as

\[
e_i \triangleq \log_2 \left( 1 + \frac{\sum_j A_{ij}}{\max_{l=1,2,\ldots,n} \sum_j A_{lj}} \right), \quad i = 1, 2, \ldots, n.
\]
The component $e_i \in [0,1]$ is the vertex energy of $x_i$, which measures how “important” $x_i$ is.

Figure 5(a) shows a two moons dataset, and Fig. 5(b) plots its vertex energy. In Ref. [30], the density is measured simply by the number of points within the $\epsilon$-neighborhood of one point. However, the proposed vertex energy defined in Eq. (34) 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 proposed vertex energy is more suitable for datasets containing clusters of differing densities. A possible limitation is that in an extremely unbalanced dataset, the presence of a cluster that is dense and contains a large number of points will hinder the detection of smaller ones due to the global estimate of the vertex energy. This is a problem to be addressed in our future research.

A subset $S$ consisting of the vertices of higher energy is obtained (e.g., Fig. 5(c)), which is termed core point set.

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

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

**Definition 4 (Core-point-connectivity)** Two core points $p$ and $q$ in $S$ are core-point-connected w.r.t. $k$ (denoted as $p \preceq_k q$) if there exists a chain of core points $p_1, p_2, \ldots, p_m$, \( p_1 = p, p_m = q \) such that $p_{i+1} \in N_k(p_i) \cap S$ and $p_i \in N_k(p_{i+1}) \cap S$.

From the viewpoint of density-based clustering [19,30], the core-point-connectivity separates $S$ into some natural subgroups, as shown in Fig. 5(d), which are defined as connected components as follows.

**Definition 5** A set of $c$ connected components $\{I_1, I_2, \ldots, I_c\}$ is obtained by separating the core point set $S$ w.r.t. $k$, such that $\forall i \neq j$, $I_i \cap I_j = \emptyset$, $S = \bigcup_{i=1}^c I_i$, and $\forall p, q \in I_i$, $p \preceq_k q$, while $\forall p \in I_i$, $\forall q \in I_j$, $i \neq j$, $p \preceq_k q$ does not hold.

The connected components $\{I_1, I_2, \ldots, I_c\}$ are taken as initial clusters which will be further refined via multi-protoype competitive learning.
4.2 Multi-prototype competitive learning

The initial clusters \{I_1, I_2, \ldots, 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 Ref. [19], this section employs classical competitive learning to refine the initial clustering and assign cluster labels to all data points. Experimental results show that the proposed approach can obtain at least 9.8% improvement over the direct assignment.

Since the dataset is nonlinearly separable, a nonlinear cluster with concave boundaries would always exist, which cannot be characterized by a single prototype that produces convex boundaries [31]. 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 \(I_j\) can be taken as one of the initial prototypes representing the \(j\)th cluster \(C_j\). 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 [32] can generate suitable prototypes to represent an input dataset without preselecting the number of prototypes. In our work, the representative points are obtained by applying affinity propagation to each \(I_j\). The similarity \(s(x_i, x_i')\) between \(x_i, x_i' \in I_j\) is set to \(-\|x_i - x_i'\|^2\), and the preferences are set to the median of the similarities, which outputs \(p_j\) suitable multi-prototypes \(\mu_1^j, \ldots, \mu_p^j\). In this way, we obtain an initial multi-prototype set

\[
W = \{\mu_1^1, \ldots, \mu_{p_1}^1, \mu_2^1, \ldots, \mu_{p_2}^1, \ldots, \mu_1^c, \ldots, \mu_{p_c}^c\}. \tag{35}
\]

Throughout the paper, we use the index notation \(\omega^\nu_j\) to denote the multi-prototype \(\mu^\nu_j\). That is, referring to the \(\omega^\nu_j\)th multi-prototype is equivalent to mentioning \(\mu^\nu_j\), and \(\omega = \{\omega_1^1, \omega_1^2, \omega_2^1, \omega_2^2, \ldots, \omega_1^c, \omega_2^c, \ldots, \omega_p^c\}\).

After the initial multi-prototype set \(W\) is obtained, classical competitive learning is performed to iteratively update the multi-prototypes such that the multi-prototype objective function is minimized:

\[
J(W) = \sum_{i=1}^n \|x_i - \mu_{v_i}^{\omega_i}\|^2, \tag{36}
\]
where $\mu^{vi}_i$ satisfies $\omega^{vi}_i = \arg \min_{\omega^j \in \omega} \| x_i - \mu^j \|^2$, i.e., the winning multi-prototype of $x_i$ that is nearest to $x_i$. For each randomly taken $x_i$, the winning multi-prototype $\omega^{vi}_i$ is selected via the winner selection rule:

$$\omega^{vi}_i = \arg \min_{\omega^j \in \omega} \| x_i - \mu^j \|^2,$$

(37)

and is updated by the winner update rule:

$$\mu^{vi}_i \leftarrow \mu^{vi}_i + \eta_i (x_i - \mu^{vi})$$

(38)

with learning rates $\{\eta_i\}$ satisfying [21]: $\lim_{i \to \infty} \eta_i = 0$, $\sum_{i=1}^{\infty} \eta_i = \infty$, $\sum_{i=1}^{\infty} \eta_i^2 < \infty$. In practice, $\eta_i = \text{const}/t$, where “const” is a small constant, e.g., 0.5.

Figures 6(a) and 6(b) illustrate the procedure of updating the winning multi-prototype. The converged multi-prototype set $\mathcal{W}$ and the corresponding Voronoi diagram are shown in Fig. 6(c). 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. 6(d).

Algorithm 3 summarizes the proposed graph-based multi-prototype competitive learning (GMPCL) method.

Algorithm 3: Graph-Based Multi-Prototype Competitive Learning (GMPCL)

Input: $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}, \rho, t_{\text{max}}, \{\eta_i\}, \epsilon$.

1. Construct a graph $G^e = (V, A, e)$ and initialize a clustering $\{I_1, I_2, \ldots, I_c\}$.
2. Initialize a multi-prototype set $\mathcal{W}$, set $t = 0$.
3. Repeat
   - Randomly permute $\{x_1, x_2, \ldots, x_n\}$, $\mathcal{W} = \mathcal{W}$, $t \leftarrow t + 1$.
   - For $i = 1, 2, \ldots, n$ do
     - Select a winning multi-prototype $\omega^{vi}_i$ by Eq. (37).
     - Update $\mu^{vi}_i$ with learning rate $\eta_i$ by Eq. (38).
   - End for
4. Until $L \leq \epsilon$ or $t \geq t_{\text{max}}$

Output: Clusters $\{C_1, C_2, \ldots, C_c : x_i \in C_{v_i}, s.t. \omega^{vi}_i = \arg \min_{\omega^j \in \omega} \| x_i - \mu^j \|^2, \forall i = 1, 2, \ldots, n\}.$

4.3 Extension of GMPCL and experiment evaluation

High-dimensional clustering applications, such as video clustering, are characterized by a high computational load, which is mainly due to the redundant calculation.

Fig. 6 GMPCL: the winner update and the clustering result of two moons. (a) and (b) are the procedure of updating the winning multi-prototype. By comparing (a) and (b), both the winning multi-prototype and the corresponding lines in Voronoi diagram move slightly. (c) is the converged multi-prototypes and the corresponding Voronoi diagram, the multi-prototypes representing different clusters are plotted in different markers. (d) is the final clusters.
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 [17] is considered. First, an inner product matrix 
\( M = [M_{i,j}]_{n \times n} \) of the dataset \( \mathcal{X} \) is computed such that 
\( M_{i,j} = \langle x_i, x_j \rangle \). Then the computation of \( \| x_i - x_j \|^2 \) is efficiently accomplished by 
\( \| x_i - x_j \|^2 = M_{i,i} + M_{j,j} - 2M_{i,j} \). 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 \( w_{vi}^{\nu} \) is updated by 
\( w_{vi}^{\nu} \rightarrow w_{vi}^{\nu} + \eta_i (x_i - w_{vi}^{\nu}) \), it is unlikely that the updated \( w_{vi}^{\nu} \) satisfies \( w_{vi}^{\nu} \in \mathcal{X} \). No pre-computed distance \( \| x_i - w_{vi}^{\nu} \|^2 \) is available for calculating Eq. (37).

In KCL, a prototype descriptor \( W^\phi \) is designed to represent \( c \) prototypes \( \{\mu_1, \mu_2, \ldots, \mu_c\} \) in the kernel space induced by a mapping \( \phi \). The prototype descriptor \( W^\phi \) 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. Similarly, we can develop a multi-prototype descriptor, which is a row-block matrix independent of the dimensionality, and extend GMPCL to deal with high-dimensional clustering.

According to the initialization of multi-prototypes, the initial \( W \) satisfies \( W \subset \mathcal{X} \). 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 \( |W| \times (n + 1) \),

\[
W = \begin{pmatrix}
W_1 \\
W_2 \\
\vdots \\
W_c
\end{pmatrix}
\]

such that the \( j \)th block \( W_j \) represents \( C_j \), and the \( q \)th row of \( W_j \), i.e., \( W_j^q \), represents \( w_j^q \) by

\[
W_j^q = \langle w_j^q, x_i \rangle, \quad i = 1, 2, \ldots, n, \quad W_j^{q+1} = \langle w_j^q, w_j^q \rangle,
\]

where \( W_j^q \) denotes the \( i \)th column of \( W_j^q \). That is,

\[
\begin{pmatrix}
\langle w_1^1, x_1 \rangle & \langle w_1^1, x_2 \rangle & \cdots & \langle w_1^1, x_n \rangle & \langle w_1^1, w_1^1 \rangle \\
\langle w_2^1, x_1 \rangle & \langle w_2^1, x_2 \rangle & \cdots & \langle w_2^1, x_n \rangle & \langle w_2^1, w_2^1 \rangle \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
\langle w_c^1, x_1 \rangle & \langle w_c^1, x_2 \rangle & \cdots & \langle w_c^1, x_n \rangle & \langle w_c^1, w_c^1 \rangle \\
\langle w_1^q, x_1 \rangle & \langle w_1^q, x_2 \rangle & \cdots & \langle w_1^q, x_n \rangle & \langle w_1^q, w_1^q \rangle \\
\langle w_2^q, x_1 \rangle & \langle w_2^q, x_2 \rangle & \cdots & \langle w_2^q, x_n \rangle & \langle w_2^q, w_2^q \rangle \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
\langle w_c^q, x_1 \rangle & \langle w_c^q, x_2 \rangle & \cdots & \langle w_c^q, x_n \rangle & \langle w_c^q, w_c^q \rangle 
\end{pmatrix}
\]

Using the \( \omega \)-notation, the \( \omega_j^q \)th row, i.e., \( W_j^q \), represents the \( \omega_j^q \)th multi-prototype, i.e., \( w_j^q \).

The initial multi-prototype descriptor \( W \) is obtained as a sub-matrix of \( M \). In Algorithm 3, 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, similar to kernel competitive learning, we implement these procedures whose computational complexity is independent of the dimensionality. This fast approach is termed fast graph-based multi-prototype competitive learning (FGMPCL) in our previous work [29].

To demonstrate the effectiveness of FGMPCL, we report the experimental results for the video clustering task. Video clustering aims at clustering video frames according to different scenes. It plays an important role in automatic video summarization/abstraction as a pre-processing step [33]. Since our intention here is to show that FGMPCL provides an effective and efficient tool for video clustering, it is beyond the scope of this paper to use the domain specific cues [33]. The gray-scale values of raw pixels were used as the feature vector for each frame.

One video sequence was used, which we termed “qiangqiangsrx20100603”, and was downloaded from vi.ifeng.com1. This video has a duration of 101 seconds, consisting of 5402 frames of size 576 × 432. During this video, different types of scenes are present. Figure 7

![Figure 7](http://v.ifeng.com/society/201006/999a2cc5-d6b0-43f4-8bf9-b829fced1654.shtml)
plots the video clustering result on this video by the proposed FGMPCL algorithm with the parameter $k$ in $k$-NN graph set to 100. From the figure, we can see that, a good segmentation of this video has been obtained. It can correctly cluster the 5402 frames into seven different scenes presented in the video, as listed in Fig. 7, with each scene plotted by two presentative frames.

5 Conclusions

Competitive learning clustering is a hot research topic in the field of data clustering. This paper presents two works done by our group which address the nonlinearly separable problem suffered by the classical competitive learning clustering algorithms. They are kernel competitive learning (KCL) and graph-based multi-prototype competitive learning (GMPC), respectively. The former is derived from the viewpoint of kernel clustering, and the latter is based on the graph method. In KCL, a novel prototype representation termed prototype descriptor is proposed, which represents prototypes without explicit kernel mapping. In GMPC, multi-prototype representation is introduced to characterize clusters of irregular shapes. Some extensions of these two approaches have been presented. And experimental results have been reported to demonstrate their effectiveness.

Acknowledgements This project was supported by the National Natural Science Foundation of China (NSFC) (Grant No. 61173084) and the NSFC-GuangDong (U0835005).

References


Jianhuang LAI received his M.Sc. degree in applied mathematics in 1989 and his Ph.D. degree in mathematics in 1999 from Sun Yat-sen University, Guangzhou, China. He joined Sun Yat-sen University in 1989 as an Assistant Professor, where currently, he is a Professor with the Department of Automation of School of Information Science and Technology and Vice Dean of School of Information Science and Technology. His current research interests are in the areas of digital image processing, pattern recognition, multimedia communication, wavelet and its applications. He has published over 80 scientific papers in the international journals and conferences on image processing and pattern recognition, e.g., IEEE TNN, IEEE TIP, IEEE TSMC (Part B), Pattern Recognition, ICCV, CVPR, and ICDM. Prof. Lai is chair of the Image and Graphics Association of Guangdong Province and also serves as a standing member of the Image and Graphics Association of China.

Changdong WANG received the B.S. degree in applied mathematics in 2008 and M.Sc. degree in computer science in 2010 from Sun Yat-sen University, Guangzhou, China. He started the pursuit of the Ph.D. degree with Sun Yat-sen University in September 2010. His current research interests include machine learning and data mining, especially focusing on data clustering and its applications. He has published over 10 scientific papers in several international journals and conferences such as Neurocomputing, Knowledge and Information System, IEEE TSMC-C, IEEE TKDE, and IEEE ICDM. His ICDM 2010 paper won the Honorable Mention for Best Research Paper Awards. He won the Student Travel Award from ICDM 2010 and ICDM 2011, respectively.