A survey of kernel and spectral methods for clustering

Maurizio Filippone[a] Francesco Camastra[b] Francesco Masulli[a] Stefano Rovetta[a]

[a]*Department of Computer and Information Science, University of Genova, and CNISM, Via Dodecaneso 35, I-16146 Genova, Italy*

[b]*Department of Applied Science, University of Naples Parthenope, Via A. De Gasperi 5 I-80133 Napoli, Italy*

Abstract

Clustering algorithms are a useful tool to explore data structures and have been employed in many disciplines. The focus of this paper is the partitioning clustering problem with a special interest in two recent approaches: kernel and spectral methods. The aim of this paper is to present a survey of kernel and spectral clustering methods, two approaches able to produce nonlinear separating hypersurfaces between clusters. The presented kernel clustering methods are the kernel version of many classical clustering algorithms, e.g., K-means, SOM and Neural Gas. Spectral clustering arise from concepts in spectral graph theory and the clustering problem is configured as a graph cut problem where an appropriate objective function has to be optimized. An explicit proof of the fact that these two paradigms have the same objective is reported since it has been proven that these two seemingly different approaches have the same mathematical foundation. Besides, fuzzy kernel clustering methods are presented as extensions of kernel K-means clustering algorithm.

Key words: partitional clustering, Mercer kernels, kernel clustering, kernel fuzzy clustering, spectral clustering

Email addresses: `filippone@disi.unige.it` (Maurizio Filippone), `francesco.camastra@uniparthenope.it` (Francesco Camastra), `masulli@disi.unige.it` (Francesco Masulli), `rovetta@disi.unige.it` (Stefano Rovetta).

1 Introduction

Unsupervised data analysis using clustering algorithms provides a useful tool to explore data structures. Clustering methods [39,87] have been addressed in many contexts and disciplines such as data mining, document retrieval, image segmentation and pattern classification. The aim of clustering methods is to *group* patterns on the basis of a *similarity* (or *dissimilarity*) criteria where groups (or *clusters*) are set of similar patterns. Crucial aspects in clustering are *pattern representation* and the *similarity measure*. Each pattern is usually represented by a set of *features* of the system under study. It is very important to notice that a good choice of representation of patterns can lead to improvements in clustering performance. Whether it is possible to choose an appropriate set of features depends on the system under study. Once a representation is fixed it is possible to choose an appropriate similarity measure among patterns. The most popular dissimilarity measure for metric representations is the *distance*, for instance the Euclidean one [25].

Clustering techniques can be roughly divided into two categories:

- *hierarchical*;
- *partitioning*.

Hierarchical clustering techniques [39,74,83] are able to find structures which can be further divided in substructures and so on recursively. The result is a hierarchical structure of groups known as *dendrogram*.

Partitioning clustering methods try to obtain a single partition of data without any other sub-partition like hierarchical algorithms do and are often based on the optimization of an appropriate objective function. The result is the creation of separations hypersurfaces among clusters. For instance we can consider two nonlinear clusters as in figure 1. Standard partitioning methods (e.g., K-Means, Fuzzy c-Means, SOM and Neural Gas) using two centroids are not able to separate in the desired way the two rings. The use of many centroids could solve this problem providing a complex description of a simple data set. For this reason several modifications and new approaches have been introduced to cope with this problem.

Among the large amount of modifications we can mention the Fuzzy c-Varieties [8], but the main drawback is that some a priori information on the shape of clusters must be included. Recently, some clustering methods that produce nonlinear separating hypersurfaces among clusters have been proposed. These algorithms can be divided in two big families: kernel and spectral clustering methods.

Regarding kernel clustering methods, several clustering methods have been

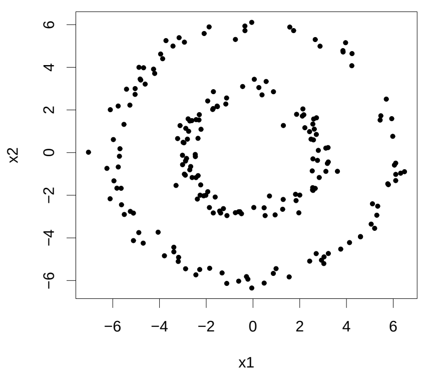


Figure 1. A data set composed of two rings of points.

modified incorporating kernels (e.g., K-Means, Fuzzy c-Means, SOM and Neural Gas). The use of kernels allows to map implicitly data into an high dimensional space called feature space; computing a linear partitioning in this feature space results in a nonlinear partitioning in the input space.

Spectral clustering methods arise from concepts in spectral graph theory. The basic idea is to construct a weighted graph from the initial data set where each node represents a pattern and each weighted edge simply takes into account the similarity between two patterns. In this framework the clustering problem can be seen as a graph cut problem, which can be tackled by means of the spectral graph theory. The core of this theory is the eigenvalue decomposition of the Laplacian matrix of the weighted graph obtained from data. In fact, there is a close relationship between the second smallest eigenvalue of the Laplacian and the graph cut [16,26].

The aim of this paper is to present a survey of kernel and spectral clustering methods. Moreover, an explicit proof of the fact that these two approaches have the same mathematical foundation is reported. In particular it has been shown by Dhillon et al. that Kernel K-Means and spectral clustering with the ratio association as the objective function are perfectly equivalent [20,21,23]. The core of both approaches lies in their ability to construct an adjacency structure between data avoiding to deal with a prefixed shape of clusters. These approaches have a slight similarity with hierarchical methods in the use of an adjacency structure with the main difference in the philosophy of the grouping procedure.

A comparison of some spectral clustering methods has been recently proposed in [79], while there are some theoretical results on the capabilities and convergence properties of spectral methods for clustering [40,80,81,90]. Recently kernel methods have been applied to Fuzzy c-Varieties also [50] with the aim of finding varieties in feature space and there are some interesting clustering methods using kernels such as [33] and [34].

Since the choice of the kernel and of the similarity measure is crucial in these methods, many techniques have been proposed in order to learn automatically the shape of kernels from data as in [4,18,27,56].

Regarding the applications, most of these algorithms (e.g., [12,18,50]) have been applied to standard benchmarks such as Ionosphere [73], Breast Cancer [85] and Iris [28] [1]. Kernel Fuzzy c-Means proposed in [14,93,94] has been applied in image segmentation problems while in [32] it has been applied in handwritten digits recognition. There are applications of kernel clustering methods in face recognition using kernel SOM [76], in speech recognition [69] and in prediction of crop yield from climate and plantation data [3]. Spectral methods have been applied in clustering of artificial data [60,63], in image segmentation [56,72,75], in bioinformatics [19], and in co-clustering problems of words and documents [22] and genes and conditions [42]. A semi-supervised spectral approach to bioinformatics and handwritten character recognition have been proposed in [48]. The protein sequence clustering problem has been faced using spectral techniques in [61] and kernel methods in [84].

In the next section we briefly introduce the concepts of linear partitioning methods by recalling some basic crisp and fuzzy algorithms. Then the paper is organized as follows: section 3 shows the kernelized version of the algorithms presented in section 2, in section 4 we discuss spectral clustering, while in section 5 we report the equivalence between spectral and kernel clustering methods. In the last section conclusions are drawn.

2 Partitioning Methods

In this section we briefly recall some basic facts about partitioning clustering methods and we will report the clustering methods for which a kernel version has been proposed. Let $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be a data set composed by n *patterns* for which every $\mathbf{x}_i \in \mathbb{R}^d$. The *codebook* (or *set of centroids*) V is defined as the set $V = \{\mathbf{v}_1, \ldots, \mathbf{v}_c\}$, typically with $c \ll n$. Each element $\mathbf{v}_i \in \mathbb{R}^d$ is called *codevector* (or *centroid* or *prototype*) [2].

The *Voronoi region* R_i of the codevector $\mathbf{v}_i$ is the set of vectors in $\mathbb{R}^d$ for which

[1] These data sets can be found at: ftp://ftp.ics.uci.edu/pub/machine-learning-databases/
[2] Among many terms to denote such objects, we will use codevectors as in vector quantization theory.

$\mathbf{v}_i$ is the nearest vector:

$$R_i = \left\{ \mathbf{z} \in \mathbb{R}^d \,\middle|\, i = \arg\min_j \|\mathbf{z} - \mathbf{v}_j\|^2 \right\} . \tag{1}$$

It is possible to prove that each Voronoi region is convex [51] and the boundaries of the regions are linear segments.

The definition of the *Voronoi set* π_i of the codevector $\mathbf{v}_i$ is straightforward. It is the subset of X for which the codevector $\mathbf{v}_i$ is the nearest vector:

$$\pi_i = \left\{ \mathbf{x} \in X \,\middle|\, i = \arg\min_j \|\mathbf{x} - \mathbf{v}_j\|^2 \right\} , \tag{2}$$

that is, the set of vectors belonging to R_i. A partition on $\mathbb{R}^d$ induced by all Voronoi regions is called *Voronoi tessellation* or *Dirichlet tessellation.*

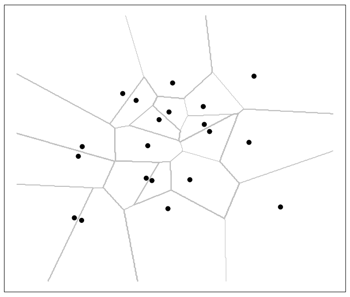


Figure 2. An example of Voronoi tessellation where each black point is a codevector.

2.1 Batch K-Means

A simple algorithm able to construct a Voronoi tessellation of the input space was proposed in 1957 by Lloyd [52] and it is known as *batch K-Means.* Starting from the finite data set X this algorithm moves iteratively the k codevectors to the arithmetic mean of their Voronoi sets $\{\pi_i\}_{i=1,\ldots,k}$. Theoretically speaking, a necessary condition for a codebook V to minimize the *Empirical Quantization Error*:

$$E(X) = \frac{1}{2n} \sum_{i=1}^{k} \sum_{\mathbf{x} \in \pi_i} \|\mathbf{x} - \mathbf{v}_i\|^2 \tag{3}$$

is that each codevector $\mathbf{v}_i$ fulfills the *centroid condition* [29]. In the case of a finite data set X and with Euclidean distance, the centroid condition reduces to:

$$\mathbf{v}_i = \frac{1}{|\pi_i|} \sum_{\mathbf{x} \in \pi_i} \mathbf{x} \,. \tag{4}$$

Batch K-Means is formed by the following steps:

(1) choose the number k of clusters;
(2) initialize the codebook V with vectors randomly picked from X;
(3) compute the Voronoi set π_i associated to the codevector $\mathbf{v}_i$;
(4) move each codevector to the mean of its Voronoi set using Eq. 4;
(5) return to step 3 if any codevector has changed otherwise return the codebook.

At the end of the algorithm a codebook is found and a Voronoi tessellation of the input space is provided. It is guaranteed that after each iteration the quantization error does not increase. Batch K-Means can be viewed as an *Expectation-Maximization* [9] algorithm, ensuring the convergence after a finite number of steps.

This approach presents many disadvantages [25]. Local minima of $E(X)$ make the method dependent on initialization, and the average is sensitive to outliers. Moreover, the number of clusters to find must be provided, and this can be done only using some a priori information or additional validity criterion. Finally, K-Means can deal only with clusters with spherically symmetrical point distribution, since Euclidean distances of patterns from centroids are computed leading to a spherical invariance. Different distances lead to different invariance properties as in the case of Mahalanobis distance which produces invariance on ellipsoids [25].

The term *batch* means that at each step the algorithm takes into account the whole data set to update the codevectors. When the cardinality n of the data set X is very high (e.g., several hundreds of thousands) the batch procedure is computationally expensive. For this reason an on-line update has been introduced leading to the *on-line K-Means* algorithm [51,54]. At each step, this method simply randomly picks an input pattern and updates its nearest codevector, ensuring that the scheduling of the updating coefficient is adequate to allow convergence and consistency.

A *Self Organizing Map* (*SOM*) [43] also known as *Self Organizing Feature Map* (*SOFM*) represents data by means of codevectors organized on a grid with fixed topology. Codevectors move to adapt to the input distribution, but adaptation is propagated along the grid also to neighboring codevectors, according to a given propagation or neighborhood function. This effectively constrains the evolution of codevectors. Grid topologies may differ, but in this paper we consider a two-dimensional, square-mesh topology [44,45]. The distance on the grid is used to determine how strongly a codevector is adapted when the unit a_{ij} is the winner. The metric used on a rectangular grid is the Manhattan distance, for which the distance between two elements $\mathbf{r} = (r_1, r_2)$ and $\mathbf{s} = (s_1, s_2)$ is:

$$d_{rs} = |r_1 - s_1| + |r_2 - s_2| \; . \tag{5}$$

The SOM algorithm is the following:

(1) Initialize the codebook V randomly picking from X
(2) Initialize the set C of connections to form the rectangular grid of dimension $n_1 \times n_2$
(3) Initialize $t = 0$
(4) Randomly pick an input $\mathbf{x}$ from X
(5) Determine the winner

$$\mathbf{s}(\mathbf{x}) = \arg \min_{\mathbf{v}_j \in V} \|\mathbf{x} - \mathbf{v}_j\| \tag{6}$$

(6) Adapt each codevector:

$$\Delta \mathbf{v}_j = \epsilon(t) h(d_{rs})(\mathbf{x} - \mathbf{v}_j) \tag{7}$$

where h is a decreasing function of d as for instance:

$$h(d_{rs}) = \exp\left(-\frac{d_{rs}^2}{2\sigma^2(t)}\right) \tag{8}$$

(7) Increment t
(8) if $t < t_{\max}$ go to step 4

$\sigma(t)$ and $\epsilon(t)$ are decreasing functions of t, for example [64]:

$$\sigma(t) = \sigma_{\mathrm{i}} \left(\frac{\sigma_{\mathrm{f}}}{\sigma_{\mathrm{i}}}\right)^{t/t_{\max}} , \qquad \epsilon(t) = \epsilon_{\mathrm{i}} \left(\frac{\epsilon_{\mathrm{f}}}{\epsilon_{\mathrm{i}}}\right)^{t/t_{\max}} , \tag{9}$$

where σ_{i}, σ_{f} and ϵ_{i}, ϵ_{f} are the initial and final values for the functions $\sigma(t)$ and $\epsilon(t)$.

A final note on the use of SOM for clustering. The method was originally devised as a tool for embedding multidimensional data into typically two dimensional spaces, for data visualization. Since then, it has also been frequently used as a clustering method, which was originally not considered appropriate because of the constraints imposed by the topology. However, the topology itself serves an important purpose, namely, that of limiting the flexibility of the mapping in the first training cycles, and gradually increasing it (while decreasing the magnitude of updates to ensure convergence) as more cycles were performed. The strategy is similar to that of other algorithms, including these described in the following, in the *capacity control* of the method which has the effect of avoiding local minima. This accounts for the fast convergence often reported in experimental works.

2.3 Neural Gas

Another technique that tries to minimize the distortion error is the neural gas algorithm [55], based on a *soft* adaptation rule. This technique resembles the SOM in the sense that not only the winner codevector is adapted. It is different in that codevectors are not constrained to be on a grid, and the adaptation of the codevectors near the winner is controlled by a criterion based on distance ranks. Each time a pattern $\mathbf{x}$ is presented, all the codevectors $\mathbf{v}_j$ are ranked according to their distance to $\mathbf{x}$ (the closest obtains the lowest rank). Denoting with ρ_j the rank of the distance between $\mathbf{x}$ and the codevector $\mathbf{v}_j$, the update rule is:

$$\Delta\mathbf{v}_j = \epsilon(t) h_\lambda(\rho_j)(\mathbf{x} - \mathbf{v}_j) \tag{10}$$

with $\epsilon(t) \in [0, 1]$ gradually lowered as t increases and $h_\lambda(\rho_j)$ a function decreasing with ρ_j with a characteristic decay λ; usually $h_\lambda(\rho_j) = \exp(-\rho_j/\lambda)$. The Neural Gas algorithm is the following:

(1) Initialize the codebook V randomly picking from X
(2) Initialize the time parameter $t = 0$
(3) Randomly pick an input $\mathbf{x}$ from X
(4) Order all elements $\mathbf{v}_j$ of V according to their distance to $\mathbf{x}$, obtaining the ρ_j
(5) Adapt the codevectors according to Eq. 10
(6) Increase the time parameter $t = t + 1$
(7) if $t < t_{\max}$ go to step 3.

Bezdek [8] introduced the concept of hard and fuzzy partition in order to extend the notion of membership of pattern to clusters. The motivation of this extension is related to the fact that a pattern often cannot be thought of as belonging to a single cluster only. In many cases, a description in which the membership of a pattern is shared among clusters is necessary.

Definition 2.1 *Let A_{cn} denote the vector space of $c \times n$ real matrices over $\mathbb{R}$. Considering X, A_{cn} and $c \in \mathbb{N}$ such that $2 \leq c < n$, the* Fuzzy c-partition space for X *is the set:*

$$M_{\text{fc}} = \left\{ U \in A_{cn} \;\middle|\; u_{ih} \in [0,1] \, \forall i,h; \; \sum_{i=1}^{c} u_{ih} = 1 \, \forall h \; ; \; 0 < \sum_{h=1}^{n} u_{ih} < n \, \forall i \right\} . \quad (11)$$

The matrix U is the so called *membership matrix* since each element u_{ih} is the fuzzy membership of the h-th pattern to the i-th cluster. The definition of M_{fc} simply tells that the sum of the memberships of a pattern to all clusters is one (*probabilistic constraint*) and that a cluster cannot be empty or contain all patterns. This definition generalizes the notion of hard c-partitions in [8].

The mathematical tool used in all these methods for working out the solution procedure is the Lagrange multipliers technique. In particular a minimization of the intraclusters distance functional with a probabilistic constraint on the memberships of a pattern to all clusters has to be achieved. Since all the functionals involved in these methods depend on both memberships and codevectors, the optimization is iterative and follows the so called *Picard iterations method* [8] where each iteration is composed of two steps. In the first step a subset of variables (memberships) is kept fixed and the optimization is performed with respect to the remaining variables (codevectors) while in the second one the role of the fixed and moving variables is swapped. The optimization algorithm stops when variables change less than a fixed threshold.

2.4.1 Fuzzy c-Means

The Fuzzy c-Means algorithm [8] identifies clusters as fuzzy sets. It minimizes the functional:

$$J(U,V) = \sum_{h=1}^{n} \sum_{i=1}^{c} (u_{ih})^m \left\| \mathbf{x}_h - \mathbf{v}_i \right\|^2 \quad (12)$$

with respect to the membership matrix U and the codebook V with the probabilistic constraints:

$$\sum_{i=1}^{c} u_{ih} = 1 \ , \qquad \forall i = 1, \ldots, n \ . \tag{13}$$

The parameter m controls the fuzziness of the memberships and usually it is set to two; for high values of m the algorithm tends to set all the memberships equals while for m tending to one we obtain the K-Means algorithm where the memberships are crisp. The minimization of Eq. 12 is done introducing a Lagrangian function for each pattern for which the constraint is in Eq. 13.

$$L_h = \sum_{i=1}^{c} (u_{ih})^m \|\mathbf{x}_h - \mathbf{v}_i\|^2 + \alpha_h \left(\sum_{i=1}^{c} u_{ih} - 1 \right) \ . \tag{14}$$

Then the derivatives of the sum of the Lagrangian are computed with respect to the u_{ih} and $\mathbf{v}_i$ and are set to zero. This yields the iteration scheme of these equations:

$$u_{ih}^{-1} = \sum_{j=1}^{c} \left(\frac{\|\mathbf{x}_h - \mathbf{v}_i\|}{\|\mathbf{x}_h - \mathbf{v}_j\|} \right)^{\frac{2}{m-1}} \ , \tag{15}$$

$$\mathbf{v}_i = \frac{\sum_{h=1}^{n} (u_{ih})^m \mathbf{x}_h}{\sum_{h=1}^{n} (u_{ih})^m} \ . \tag{16}$$

At each iteration it is possible to evaluate the amount of change of the memberships and codevectors and the algorithm can be stopped when these quantities reach a predefined threshold. At the end a soft partitioning of the input space is obtained.

2.5 Possibilistic clustering methods

As a further modification of the K-Means algorithm, the possibilistic approach [46,47] relaxes the probabilistic constraint on the membership of a pattern to all clusters. In this way a pattern can have a low membership to all clusters in the case of outliers, whereas for instance, in the situation of overlapped clusters, it can have high membership to more than one cluster. In this framework the membership represents a degree of typicality not depending on the membership values of the same pattern to other clusters. Again the optimization procedure is the Picard iteration method, since the functional depends both on memberships and codevectors.

2.5.1 Possibilistic c-Means

There are two formulations of the Possibilistic c-Means, that we will call PCM-I [46] and PCM-II [47]. The first one aims to minimize the following functional with respect to the membership matrix U and the codebook $V = \{\mathbf{v}_1, \ldots, \mathbf{v}_c\}$:

$$J(U,V) = \sum_{h=1}^{n} \sum_{i=1}^{c} (u_{ih})^m \|\mathbf{x}_h - \mathbf{v}_i\|^2 + \sum_{i=1}^{c} \eta_i \sum_{h=1}^{n} (1 - u_{ih})^m \ , \tag{17}$$

while the second one addresses the functional:

$$J(U,V) = \sum_{h=1}^{n} \sum_{i=1}^{c} u_{ih} \|\mathbf{x}_h - \mathbf{v}_i\|^2 + \sum_{i=1}^{c} \eta_i \sum_{h=1}^{n} (u_{ih} \ln(u_{ih}) - u_{ih}) \ . \tag{18}$$

The minimization of Eq. 17 and Eq. 18 with respect to the u_{ih} leads respectively to the following equations:

$$u_{ih} = \left[1 + \left(\frac{\|\mathbf{x}_h - \mathbf{v}_i\|^2}{\eta_i} \right)^{\frac{1}{m-1}} \right]^{-1} \ , \tag{19}$$

$$u_{ih} = \exp\left(- \frac{\|\mathbf{x}_h - \mathbf{v}_i\|^2}{\eta_i} \right) \ . \tag{20}$$

The constraint on the memberships $u_{ih} \in [0,1]$ is automatically satisfied given the form assumed by Eq. 19 and Eq. 20. The updates of the centroids for PCM-I and PCM-II are respectively:

$$\mathbf{v}_i = \frac{\sum_{h=1}^{n} (u_{ih})^m \mathbf{x}_h}{\sum_{h=1}^{n} (u_{ih})^m} \ , \tag{21}$$

$$\mathbf{v}_i = \frac{\sum_{h=1}^{n} u_{ih} \mathbf{x}_h}{\sum_{h=1}^{n} u_{ih}} \ . \tag{22}$$

The parameter η_i regulates the trade-off between the two terms in Eq. 17 and Eq. 18 and it is related to the width of the clusters. The authors suggest to estimate η_i for PCM-I using this formula:

$$\eta_i = \gamma \frac{\sum_{h=1}^{n} (u_{ih})^m \|\mathbf{x}_h - \mathbf{v}_i\|^2}{\sum_{h=1}^{n} (u_{ih})^m} \tag{23}$$

which is a weighted mean of the intracluster distance of the i-th cluster and the constant γ is typically set at one. The parameter η_i can be estimated with scale estimation techniques as developed in the robust clustering literature for M-estimators [35,59]. The value of η_i can be updated at each step of the algorithm or can be fixed for all iterations. The former approach can lead to instabilities since the derivation of the equations has been obtained considering η_i fixed. In the latter case a good estimation of η_i can be done only starting from an approximate solution. For this reason often the Possibilistic c-Means is run as a refining step of a Fuzzy c-Means.

3 Kernel Clustering Methods

In machine learning, the use of the kernel functions [57] has been introduced by Aizerman et al. [1] in 1964. In 1995 Cortes and Vapnik introduced *Support Vector Machines* (SVMs) [17] which perform better than other classification algorithms in several problems. The success of SVM has brought to extend the use of kernels to other learning algorithms (e.g., *Kernel PCA* [70]). The choice of the kernel is crucial to incorporate a priori knowledge on the application, for which it is possible to design *ad hoc* kernels.

3.1 Mercer kernels

We recall the definition of Mercer kernels [2,68], considering, for the sake of simplicity, vectors in $\mathbb{R}^d$ instead of $\mathbb{C}^d$.

Definition 3.1 *Let* $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ *be a nonempty set where* $\mathbf{x}_i \in \mathbb{R}^d$*. A function* $K : X \times X \to \mathbb{R}$ *is called a* positive definite kernel *(or* Mercer kernel*) if and only if* K *is symmetric (i.e.* $K(\mathbf{x}_i, \mathbf{x}_j) = K(\mathbf{x}_j, \mathbf{x}_i)$*) and the following equation holds:*

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j K(\mathbf{x}_i, \mathbf{x}_j) \geq 0 \quad \forall n \geq 2 \ , \tag{24}$$

where $c_r \in \mathbb{R} \ \forall r = 1, \ldots, n$

Each Mercer kernel can be expressed as follows:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \ , \tag{25}$$

where $\Phi : X \to \mathcal{F}$ performs a mapping from the input space X to a high dimensional *feature space* $\mathcal{F}$. One of the most relevant aspects in applications is that it is possible to compute Euclidean distances in $\mathcal{F}$ without knowing explicitly Φ. This can be done using the so called *distance kernel trick* [58,70]:

$$\begin{aligned}\|\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)\|^2 &= (\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)) \cdot (\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)) \\ &= \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_i) + \Phi(\mathbf{x}_j) \cdot \Phi(\mathbf{x}_j) - 2\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \\ &= K(\mathbf{x}_i, \mathbf{x}_i) + K(\mathbf{x}_j, \mathbf{x}_j) - 2K(\mathbf{x}_i, \mathbf{x}_j) \end{aligned} \qquad (26)$$

in which the computation of distances of vectors in feature space is just a function of the input vectors. In fact, every algorithm in which input vectors appear only in dot products with other input vectors can be kernelized [71]. In order to simplify the notation we introduce the so called *Gram matrix* K where each element k_{ij} is the scalar product $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_i)$. Thus, Eq. 26 can be rewritten as:

$$\|\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)\|^2 = k_{ii} + k_{jj} - 2k_{ij} \; . \qquad (27)$$

Examples of Mercer kernels are the following [78]:

- linear:

$$K^{(\mathrm{l})}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j \qquad (28)$$

- polynomial of degree p:

$$K^{(\mathrm{p})}(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i \cdot \mathbf{x}_j)^p \qquad p \in \mathbb{N} \qquad (29)$$

- Gaussian:

$$K^{(\mathrm{g})}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right) \qquad \sigma \in \mathbb{R} \qquad (30)$$

It is important to stress that the use of the linear kernel in Eq. 26 simply leads to the computation of the Euclidean norm in the input space. Indeed:

$$\begin{aligned}\|\mathbf{x}_i - \mathbf{x}_j\|^2 &= \mathbf{x}_i \cdot \mathbf{x}_i + \mathbf{x}_j \cdot \mathbf{x}_j - 2\mathbf{x}_i \cdot \mathbf{x}_j \\ &= K^{(\mathrm{l})}(\mathbf{x}_i, \mathbf{x}_i) + K^{(\mathrm{l})}(\mathbf{x}_j, \mathbf{x}_j) - 2K^{(\mathrm{l})}(\mathbf{x}_i, \mathbf{x}_j) \\ &= \|\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)\|^2 \; , \end{aligned} \qquad (31)$$

shows that choosing the kernel $K^{(\mathrm{l})}$ implies $\Phi = I$ (where I is the identity function). Following this consideration we can think that kernels can offer a more general way to represent the elements of a set X and possibly, for some of these representations, the clusters can be easily identified.

In literature there are some applications of kernels in clustering. These methods can be broadly divided in three categories, which are based respectively on:

- *kernelization of the metric* [86,92,93];

- *clustering in feature space* [32,38,53,62,91];
- *description via support vectors* [12,37].

Methods based on kernelization of the metric look for centroids in input space and the distances between patterns and centroids is computed by means of kernels:

$$\|\Phi(\mathbf{x}_h) - \Phi(\mathbf{v}_i)\|^2 = K(\mathbf{x}_h, \mathbf{x}_h) + K(\mathbf{v}_i, \mathbf{v}_i) - 2K(\mathbf{x}_h, \mathbf{v}_i) \; . \tag{32}$$

Clustering in feature space is made by mapping each pattern using the function Φ and then computing centroids in feature space. Calling $\mathbf{v}_i^\Phi$ the centroids in feature space, we will see in the next sections that it is possible to compute the distances $\left\|\Phi(\mathbf{x}_h) - \mathbf{v}_i^\Phi\right\|^2$ by means of the kernel trick.

The description via support vectors makes use of One Class SVM to find a minimum enclosing sphere in feature space able to enclose almost all data in feature space excluding outliers. The computed hypersphere corresponds to nonlinear surfaces in input space enclosing groups of patterns. The Support Vector Clustering algorithm allows to assign labels to patterns in input space enclosed by the same surface. In the next subsections we will outline these three approaches.

3.2 Kernel K-Means

Given the data set X, we map our data in some feature space $\mathcal{F}$, by means of a nonlinear map Φ and we consider k centers in feature space ($\mathbf{v}_i^\Phi \in \mathcal{F}$ with $i = 1, \ldots, k$) [30,70]. We call the set $V^\Phi = (\mathbf{v}_1^\Phi, \ldots, \mathbf{v}_k^\Phi)$ *Feature Space Codebook* since in our representation the centers in the feature space play the same role of the codevectors in the input space. In analogy with the codevectors in the input space, we define for each center $\mathbf{v}_i^\Phi$ its *Voronoi Region* and *Voronoi Set* in feature space. The *Voronoi Region in feature space (R_i^Φ)* of the center $\mathbf{v}_i^\Phi$ is the set of all vectors in $\mathcal{F}$ for which $\mathbf{v}_i^\Phi$ is the closest vector

$$R_i^\Phi = \left\{ \mathbf{x}^\Phi \in \mathcal{F} \;\middle|\; i = \arg\min_j \left\|\mathbf{x}^\Phi - \mathbf{v}_j^\Phi\right\| \right\} \; . \tag{33}$$

The *Voronoi Set in Feature Space* π_i^Φ of the center $\mathbf{v}_i^\Phi$ is the set of all vectors $\mathbf{x}$ in X such that $\mathbf{v}_i^\Phi$ is the *closest vector* to their images $\Phi(\mathbf{x})$ in the feature space:

$$\pi_i^\Phi = \left\{ \mathbf{x} \in X \;\middle|\; i = \arg\min_j \left\|\Phi(\mathbf{x}) - \mathbf{v}_j^\Phi\right\| \right\} \; . \tag{34}$$

The set of the Voronoi Regions in feature space define a *Voronoi Tessellation of the Feature Space*. The Kernel K-Means algorithm has the following steps:

(1) Project the data set X into a feature space $\mathcal{F}$, by means of a nonlinear mapping Φ.
(2) Initialize the codebook $V^\Phi = (\mathbf{v}_1^\Phi, \ldots, \mathbf{v}_k^\Phi)$ with $\mathbf{v}_i^\Phi \in \mathcal{F}$
(3) Compute for each center $\mathbf{v}_i^\Phi$ the set π_i^Φ
(4) Update the codevectors $\mathbf{v}_i^\Phi$ in $\mathcal{F}$

$$\mathbf{v}_i^\Phi = \frac{1}{|\pi_i^\Phi|} \sum_{\mathbf{x} \in \pi_i^\Phi} \Phi(\mathbf{x}) \tag{35}$$

(5) Go to step 3 until any $\mathbf{v}_i^\Phi$ changes
(6) Return the feature space codebook.

This algorithm minimizes the quantization error in feature space.

Since we do not know explicitly Φ it is not possible to compute directly Eq. 35. Nevertheless, it is always possible to compute distances between patterns and codevectors by using the kernel trick, allowing to obtain the Voronoi sets in feature space π_i^Φ. Indeed, writing each centroid in feature space as a combination of data vectors in feature space we have:

$$\mathbf{v}_j^\Phi = \sum_{h=1}^{n} \gamma_{jh} \Phi(\mathbf{x}_h) \ , \tag{36}$$

where γ_{jh} is one if $\mathbf{x}_h \in \pi_j^\Phi$ and zero otherwise. Now the quantity:

$$\left\| \Phi(\mathbf{x}_i) - \mathbf{v}_j^\Phi \right\|^2 = \left\| \Phi(\mathbf{x}_i) - \sum_{h=1}^{n} \gamma_{jh} \Phi(\mathbf{x}_h) \right\|^2 \tag{37}$$

can be expanded by using the scalar product and the kernel trick in Eq. 26:

$$\left\| \Phi(\mathbf{x}_i) - \sum_{h=1}^{n} \gamma_{jh} \Phi(\mathbf{x}_h) \right\|^2 = k_{ii} - 2 \sum_h \gamma_{jh} k_{ih} + \sum_r \sum_s \gamma_{jr} \gamma_{js} k_{rs} \ . \tag{38}$$

This allows to compute the closest feature space codevector for each pattern and to update the coefficients γ_{jh}. It is possible to repeat these two operations until any γ_{jh} changes to obtain a Voronoi tessellation of the feature space.

An on-line version of the kernel K-Means algorithm can be found in [70]. A further version of K-Means in feature space has been proposed by Girolami [30]. In his formulation the number of clusters is denoted by c and a fuzzy membership matrix U is introduced. Each element u_{ih} denotes the fuzzy membership of the point $\mathbf{x}_h$ to the Voronoi set π_i^Φ. This algorithm tries to minimize the following functional with respect to U:

$$J^\Phi(U, V^\Phi) = \sum_{h=1}^{n} \sum_{i=1}^{c} u_{ih} \left\| \Phi(\mathbf{x}_h) - \mathbf{v}_i^\Phi \right\|^2 . \tag{39}$$

The minimization technique used by Girolami is *Deterministic Annealing* [65] which is a stochastic method for optimization. A parameter controls the fuzziness of the membership during the optimization and can be thought proportional to the temperature of a physical system. This parameter is gradually lowered during the annealing and at the end of the procedure the memberships have become crisp; therefore a tessellation of the feature space is found. This linear partitioning in $\mathcal{F}$, back to the input space, forms a nonlinear partitioning of the input space.

3.3 Kernel SOM

The kernel version of the SOM algorithm [38,53] is based on the distance kernel trick. The method tries to adapt the grid of codevectors $\mathbf{v}_j^\Phi$ in feature space. In kernel SOM we start writing each codevector as a combination of points in feature space:

$$\mathbf{v}_j^\Phi = \sum_{h=1}^{n} \gamma_{jh} \Phi(\mathbf{x}_h) , \tag{40}$$

where the coefficients γ_{ih} are initialized once the grid is created. Computing the winner by writing Eq. 6 in feature space leads to:

$$\mathbf{s}(\Phi(\mathbf{x}_i)) = \arg \min_{\mathbf{v}_j^\Phi \in V} \| \Phi(\mathbf{x}_i) - \mathbf{v}_j^\Phi \| , \tag{41}$$

that can be written, using the kernel trick:

$$\mathbf{s}(\Phi(\mathbf{x}_i)) = \arg \min_{\mathbf{v}_j^\Phi \in V} \left(k_{ii} - 2 \sum_h \gamma_{jh} k_{ih} \sum_r \sum_s \gamma_{jr} \gamma_{js} k_{rs} \right) . \tag{42}$$

To update the codevectors we rewrite Eq. 7:

$$\mathbf{v}_j^{\Phi\prime} = \mathbf{v}_j^\Phi + \epsilon(t) h(d_{rs}) \left(\Phi(\mathbf{x}) - \mathbf{v}_j^\Phi \right) . \tag{43}$$

Using Eq. 40:

$$\sum_{h=1}^{n} \gamma'_{jh} \Phi(\mathbf{x}_h) = \sum_{h=1}^{n} \gamma_{jh} \Phi(\mathbf{x}_h) + \epsilon(t) h(d_{rs}) \left(\Phi(\mathbf{x}) - \sum_{h=1}^{n} \gamma_{jh} \Phi(\mathbf{x}_h) \right) . \tag{44}$$

Thus the rule for the update of γ_{jh} is:

$$\gamma'_{jh} = \begin{cases} (1 - \epsilon(t) h(d_{rs})) \gamma_{jh} & \text{if } i \neq j \\ (1 - \epsilon(t) h(d_{rs})) \gamma_{jh} + \epsilon(t) h(d_{rs}) & \text{otherwise.} \end{cases} \tag{45}$$

3.4 Kernel Neural Gas

The Neural Gas algorithm provides a soft update rule for the codevectors in input space. The kernel version of neural gas [62] applies the soft rule for the update to the codevectors in feature space. Rewriting Eq. 10 in feature space for the update of the codevectors we have:

$$\Delta \mathbf{v}_j^{\Phi} = \epsilon h_{\lambda}(\rho_j) \left(\Phi(\mathbf{x}) - \mathbf{v}_j^{\Phi} \right) . \tag{46}$$

Here ρ_j is the rank of the distance $\|\Phi(\mathbf{x}) - \mathbf{v}_j^{\Phi}\|$. Again it is possible to write $\mathbf{v}_j^{\Phi}$ as a linear combination of $\Phi(\mathbf{x}_i)$ as in Eq. 40, allowing to compute such distances by means of the kernel trick. As in the kernel SOM technique, the updating rule for the centroids becomes an updating rule for the coefficients of such combination.

3.5 One Class SVM

This approach provides a support vector description in feature space [36,37,77]. The idea is to use kernels to project data into a feature space and then to find the sphere enclosing almost all data, namely not including outliers. Formally a radius R and the center $\mathbf{v}$ of the smallest enclosing sphere in feature space are defined. The constraint is thus:

$$\|\Phi(\mathbf{x}_j) - \mathbf{v}\|^2 \leq R^2 + \xi_j \quad \forall j , \tag{47}$$

where the non negative slack variables ξ_j have been added. The Lagrangian for this problem is defined [11]:

$$L = R^2 - \sum_j (R^2 + \xi_j - \|\Phi(\mathbf{x}_j) - \mathbf{v}\|^2) \beta_j - \sum_j \xi_j \mu_j + C \sum_j \xi_j \tag{48}$$

where $\beta_j \geq 0$ and $\mu_j \geq 0$ are Lagrange multipliers, C is a constant and $C\sum_j \xi_j$ is a penalty term. Computing the partial derivative of L with respect to R, $\mathbf{v}$ and ξ_j and setting them to zero leads to the following equations:

$$\sum_j \beta_j = 1, \quad \mathbf{v} = \sum_j \beta_j \Phi(\mathbf{x}_j), \quad \beta_j = C - \mu_j. \tag{49}$$

The Karush-Kuhn-Tucker (KKT) complementary conditions [11] result in:

$$\xi_j \mu_j = 0, \quad (R^2 + \xi_j - \|\Phi(\mathbf{x}_j) - \mathbf{v}\|^2)\beta_j = 0. \tag{50}$$

Following simple considerations regarding all these conditions it is possible to see that:

- when $\xi_j > 0$, the image of $\mathbf{x}_j$ lies outside the hypersphere. These points are called *bounded support vectors*;
- when $\xi_j = 0$ and $0 < \beta_j < C$, the image of $\mathbf{x}_j$ lies on the surface of the hypersphere. These points are called *support vectors*.

Moreover, it is possible to write the Wolfe dual form [36], whose optimization leads to this quadratic programming problem with respect to the β_j:

$$J_W = \sum_j k_{jj}\beta_j - \sum_i \sum_j k_{ij}\beta_i\beta_j \tag{51}$$

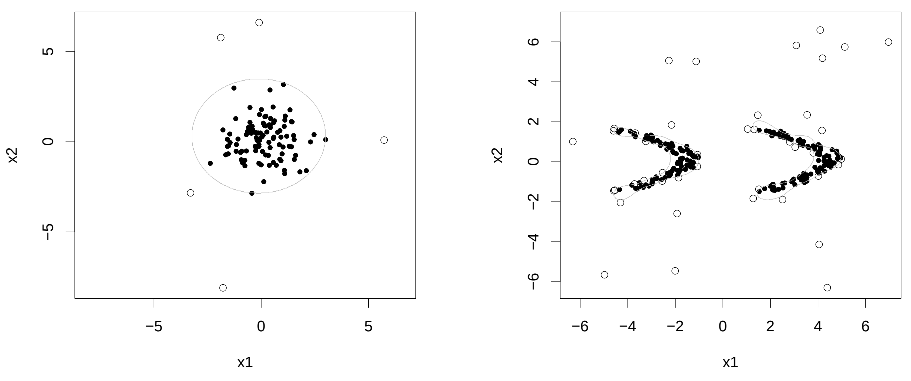
The distance from the image of a point $\mathbf{x}_j$ and the center $\mathbf{v}$ of the enclosing sphere can be computed as follows:

$$d_j = \|\Phi(\mathbf{x}_j) - \mathbf{v}\|^2 = k_{jj} - 2\sum_r \beta_r k_{jr} + \sum_r \sum_s \beta_r \beta_s k_{rs} \tag{52}$$

In Fig. 3 it is possible to see the ability of this algorithm to find the smallest enclosing sphere without outliers.

3.5.1 Support Vector Clustering

Once boundaries in input space are found, a labeling procedure is necessary in order to complete clustering. In [37] the cluster assignment procedure follows a simple geometric idea. Any path connecting a pair of points belonging to different clusters must exit from the enclosing sphere in feature space. Denoting with Y the image in feature space of one of such paths and with $\mathbf{y}$ the elements of Y, it will result that $R(\mathbf{y}) > R$ for some $\mathbf{y}$. Thus it is possible to



(a) (b)

Figure 3. One class SVM applied to two data sets with outliers. The gray line shows the projection in input space of the smallest enclosing sphere in feature space. In (a) a linear kernel and in (b) a Gaussian kernel have been used.

define an adjacency structure in this form:

$$\begin{cases} 1 & \text{if } R(\mathbf{y}) < R \quad \forall \mathbf{y} \in Y \\ 0 & \text{otherwise.} \end{cases} \tag{53}$$

Clusters are simply the connected components of the graph with the adjacency matrix just defined. In the implementation in [36] the check is made sampling the line segment Y in 20 equidistant points. There are some modifications on this labeling algorithm (e.g., [49,88]) that improve performances. An improved version of SVC algorithm with application in handwritten digits recognition can be found in [15].

3.5.2 Camastra and Verri algorithm

A technique combining K-Means and One Class SVM can be found in [12]. The algorithm uses a K-Means-like strategy, i.e., moves repeatedly all centers $\mathbf{v}_i^\Phi$ in the feature space, computing One Class SVM on their Voronoi sets π_i^Φ, until no center changes anymore. Moreover, in order to introduce robustness against outliers, the authors have proposed to compute One Class SVM on $\pi_i^\Phi(\rho)$ of each center $\mathbf{v}_i^\Phi$. The set $\pi_i^\Phi(\rho)$ is defined as

$$\pi_i^\Phi(\rho) = \{\mathbf{x}_j \in \pi_i^\Phi \text{ and } \|\Phi(\mathbf{x}_j) - \mathbf{v}_i^\Phi\| < \rho\} \; . \tag{54}$$

$\pi_i^\Phi(\rho)$ is the Voronoi set in the feature space of the center $\mathbf{v}_i^\Phi$ without outliers, that is the images of data points whose distance from the center is larger than

ρ. The parameter ρ can be set up using model selection techniques [9] (e.g., cross-validation). In summary, the algorithm has the following steps:

(1) Project the data Set X into a feature space $\mathcal{F}$, by means of a nonlinear mapping Φ.
(2) Initialize the codebook $V^\Phi = (\mathbf{v}_1^\Phi, \dots, \mathbf{v}_k^\Phi)$ with $\mathbf{v}_i^\Phi \in \mathcal{F}$
(3) Compute $\pi_i^\Phi(\rho)$ for each center $\mathbf{v}_i^\Phi$
(4) Apply One Class SVM to each $\pi_i^\Phi(\rho)$ and assign the center obtained to $\mathbf{v}_i^\Phi$
(5) Go to step 2 until any $\mathbf{v}_i^\Phi$ changes
(6) Return the feature space codebook.

3.6 Kernel fuzzy clustering methods

Here we show some kernelized versions of Fuzzy c-Means algorithms, showing in particular Fuzzy and Possibilistic c-Means. In the first subsection we show the method of the kernelization of the metric while in the second one the Fuzzy c-Means in feature space is shown. The third subsection is devoted to the kernelized version of the Possibilistic c-Means.

3.6.1 Kernel Fuzzy c-Means with kernelization of the metric

The basic idea is to minimize the functional [86,92,93]:

$$J^\Phi(U,V) = \sum_{h=1}^{n}\sum_{i=1}^{c}(u_{ih})^m \left\|\Phi(\mathbf{x}_h) - \Phi(\mathbf{v}_i)\right\|^2 , \tag{55}$$

with the probabilistic constraint over the memberships (Eq. 13). The procedure for the optimization of $J^\Phi(U,V)$ is again the Picard iteration technique. Minimization of the functional in Eq. 55 has been proposed only in the case of a Gaussian kernel $K^{(\mathrm{g})}$. The reason is that the derivative of $J^\Phi(U,V)$ with respect to the $\mathbf{v}_i$ using a Gaussian kernel is particularly simple since it allows to use the kernel trick:

$$\frac{\partial K(\mathbf{x}_h,\mathbf{v}_i)}{\partial \mathbf{v}_i} = \frac{(\mathbf{x}_h - \mathbf{v}_i)}{\sigma^2} K(\mathbf{x}_h,\mathbf{v}_i) . \tag{56}$$

We obtain for the memberships:

$$u_{ih}^{-1} = \sum_{j=1}^{c}\left(\frac{1-K(\mathbf{x}_h,\mathbf{v}_i)}{1-K(\mathbf{x}_h,\mathbf{v}_j)}\right)^{\frac{1}{m-1}} , \tag{57}$$

and for the codevectors:

$$\mathbf{v}_i = \frac{\sum_{h=1}^{n} (u_{ih})^m K(x_h, v_i)\mathbf{x}_h}{\sum_{h=1}^{n} (u_{ih})^m K(x_h, v_i)} \ . \tag{58}$$

3.6.2 Kernel Fuzzy c-Means in feature space

Here we derive the Fuzzy c-Means in feature space, which is a clustering method which allows to find a soft linear partitioning of the feature space. This partitioning, back to the input space, results in a soft nonlinear partitioning of data. The functional to optimize [32,91] with the probabilistic constraint in Eq. 13 is:

$$J^{\Phi}(U, V^{\Phi}) = \sum_{h=1}^{n} \sum_{i=1}^{c} (u_{ih})^m \left\| \Phi(\mathbf{x}_h) - \mathbf{v}_i^{\Phi} \right\|^2 \ . \tag{59}$$

It is possible to rewrite the norm in Eq. 59 explicitly by using:

$$\mathbf{v}_i^{\Phi} = \frac{\sum_{h=1}^{n} (u_{ih})^m \Phi(\mathbf{x}_h)}{\sum_{h=1}^{n} (u_{ih})^m} = a_i \sum_{h=1}^{n} (u_{ih})^m \Phi(\mathbf{x}_h) \ , \tag{60}$$

which is the kernel version of Eq. 16. For simplicity of notation we used:

$$a_i^{-1} = \sum_{r=1}^{n} (u_{ir})^m \ . \tag{61}$$

Now it is possible to write the kernel version of Eq. 15:

$$u_{ih}^{-1} = \sum_{j=1}^{c} \left[\frac{k_{hh} - 2a_i \sum_{r=1}^{n} (u_{ir})^m k_{hr} + a_i^2 \sum_{r=1}^{n} \sum_{s=1}^{n} (u_{ir})^m (u_{is})^m k_{rs}}{k_{hh} - 2a_j \sum_{r=1}^{n} (u_{jr})^m k_{hr} + a_j^2 \sum_{r=1}^{n} \sum_{s=1}^{n} (u_{jr})^m (u_{js})^m k_{rs}} \right]^{\frac{1}{m-1}} . \tag{62}$$

Eq. 62 gives the rule for the update of the membership u_{ih}.

3.6.3 Possibilistic c-Means with the kernelization of the metric

The formulation of the Possibilistic c-Means PCM-I with the kernelization of the metric used in [92] involves the minimization of the following functional:

$$J^{\Phi}(U, V) = \sum_{h=1}^{n} \sum_{i=1}^{c} (u_{ih})^m \left\| \Phi(\mathbf{x}_h) - \Phi(\mathbf{v}_i) \right\|^2 + \sum_{i=1}^{c} \eta_i \sum_{h=1}^{n} (1 - u_{ih})^m \tag{63}$$

Minimization leads to:

$$u_{ih}^{-1} = 1 + \left(\frac{\|\Phi(\mathbf{x}_h) - \Phi(\mathbf{v}_i)\|^2}{\eta_i} \right)^{\frac{1}{m-1}} , \tag{64}$$

that can be rewritten, considering a Gaussian kernel, as:

$$u_{ih}^{-1} = 1 + 2 \left(\frac{1 - K(\mathbf{x}_h, \mathbf{v}_i)}{\eta_i} \right)^{\frac{1}{m-1}} . \tag{65}$$

The update of the codevectors follows:

$$\mathbf{v}_i = \frac{\sum_{h=1}^{n} (u_{ih})^m K(x_h, v_i) \mathbf{x}_h}{\sum_{h=1}^{n} (u_{ih})^m K(x_h, v_i)} . \tag{66}$$

The computation of the η_i is straightforward.

4 Spectral Clustering

Spectral clustering methods [19] have a strong connection with graph theory [16,24]. A comparison of some spectral clustering methods has been recently proposed in [79]. Let $X = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ be the set of patterns to cluster. Starting from X, we can build a *complete, weighted undirected graph* $G(V, A)$ having a set of nodes $V = \{v_1, \dots, v_n\}$ corresponding to the n patterns and edges defined through the $n \times n$ adjacency (also *affinity*) matrix A. The adjacency matrix for a weighted graph is given by the matrix whose element a_{ij} represents the weight of the edge connecting nodes i and j. Being an undirected graph, the property $a_{ij} = a_{ji}$ holds. Adjacency between two patterns can be defined as follows:

$$a_{ij} = \begin{cases} h(\mathbf{x}_i, \mathbf{x}_j) & \text{if } i \neq j \\ 0 & \text{otherwise.} \end{cases} \tag{67}$$

The function h measures the similarity between patterns and typically a Gaussian function is used:

$$h(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(- \frac{d(\mathbf{x}_i, \mathbf{x}_j)}{2\sigma^2} \right) , \tag{68}$$

where d measures the dissimilarity between patterns and σ controls the rapidity of decay of h. This particular choice has the property that A has only some terms significantly different from 0, i.e., it is sparse.

The degree matrix D is the diagonal matrix whose elements are the degrees of the nodes of G.

$$d_{ii} = \sum_{j=1}^{n} a_{ij} \ . \tag{69}$$

In this framework the clustering problem can be seen as a graph cut problem [16] where one wants to separate a set of nodes $S \subset V$ from the complementary set $\bar{S} = V \setminus S$. The graph cut problem can be formulated in several ways depending on the choice of the function to optimize. One of the most popular functions to optimize is the cut [16]:

$$cut(S, \bar{S}) = \sum_{v_i \in S, v_j \in \bar{S}} a_{ij} \ . \tag{70}$$

It is easy to verify that the minimization of this objective function favors partitions containing isolated nodes. To achieve a better balance in the cardinality of S and $\bar{S}$ it is suggested to optimize the normalized cut [72]:

$$Ncut(S, \bar{S}) = cut(S, \bar{S}) \left(\frac{1}{assoc(S, V)} + \frac{1}{assoc(\bar{S}, V)} \right) \ , \tag{71}$$

where the association $assoc(S, V)$ is also known as the volume of S:

$$assoc(S, V) = \sum_{v_i \in S, v_j \in V} a_{ij} \equiv vol(S) = \sum_{v_i \in S} d_{ii} \ . \tag{72}$$

There are other definitions of functions to optimize (e.g., the conductance [40], the normalized association [72], ratio cut [21]).

The complexity in optimizing these objective functions is very high (e.g., the optimization of the normalized cut is a NP-hard problem [72,82]) and for this reason it has been proposed to relax it by using spectral concepts of graph analysis. This relaxation can be formulated by introducing the *Laplacian* matrix [16]:

$$L = D - A \ , \tag{73}$$

which can be seen as a linear operator on G. In addition to this definition of Laplacian there are alternative definitions:

- Normalized Laplacian $L_{\mathrm{N}} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$
- Generalized Laplacian $L_{\mathrm{G}} = D^{-1} L$
- Relaxed Laplacian $L_\rho = L - \rho D$

Each definition is justified by special properties desirable in a given context. The spectral decomposition of the Laplacian matrix can give useful information about the properties of the graph. In particular it can be seen that the second smallest eigenvalue of L is related to the graph cut [26] and the corresponding eigenvector can cluster together similar patterns [10,16,72].

Spectral approach to clustering has a strong connection with *Laplacian Eigenmaps* [5]. The dimensionality reduction problem aims to find a proper low dimensional representation of a data set in a high dimensional space. In [5], each node in the graph, which represents a pattern, is connected just with nodes corresponding to neighboring patterns and the spectral decomposition of the Laplacian of the obtained graph permits to find a low dimensional representation of X. The authors point out the close connection with spectral clustering and *Local Linear Embedding* [67] providing theoretical and experimental validations.

4.1 Shi and Malik algorithm

The algorithm proposed by Shi and Malik. [72] applies the concepts of spectral clustering to image segmentation problems. In this framework each node is a pixel and the definition of adjacency between them is suitable for image segmentation purposes. In particular, if $\mathbf{x}_i$ is the position of the i-th pixel and $\mathbf{f}_i$ a feature vector which takes into account several of its attributes (e.g., intensity, color and texture information), they define the adjacency as:

$$a_{ij} = \exp\left(-\frac{\|\mathbf{f}_i - \mathbf{f}_j\|^2}{2\sigma_1^2}\right) \cdot \begin{cases} \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_2^2}\right) & \text{if } \|\mathbf{x}_i - \mathbf{x}_j\| < R \\ 0 & \text{otherwise.} \end{cases} \tag{74}$$

Here R has an influence on how many neighboring pixels can be connected with a pixel, controlling the sparsity of the adjacency and Laplacian matrices. They provide a proof that the minimization of $Ncut(S, \bar{S})$ can be done solving the eigenvalue problem for the normalized Laplacian L_{N}. In summary, the algorithm is composed of these steps:

(1) Construct the graph G starting from the data set X calculating the adjacency between patterns using Eq. 74
(2) Compute the degree matrix D
(3) Construct the matrix $L_{\mathrm{N}} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$

(4) Compute the eigenvector $\mathbf{e}_2$ associated to the second smallest eigenvalue λ_2
(5) Use $D^{-\frac{1}{2}}\mathbf{e}_2$ to segment G

In the ideal case of two non connected subgraphs, $D^{-\frac{1}{2}}\mathbf{e}_2$ assumes just two values; this allows to cluster together the components of $D^{-\frac{1}{2}}\mathbf{e}_2$ with the same value. In a real case the splitting point must be chosen to cluster the components of $D^{-\frac{1}{2}}\mathbf{e}_2$ and the authors suggest to use the median value, zero or the value for which the clustering gives the minimum $Ncut$. The successive partitioning can be made recursively on the obtained sub-graphs or it is possible to use more than one eigenvector. An interesting approach for clustering simultaneously the data set in more than two clusters can be found in [89].

4.2 Ng, Jordan and Weiss algorithm

The algorithm that has been proposed by Ng et al. [60] uses the adjacency matrix A as Laplacian. This definition allows to consider the eigenvector associated with the largest eigenvalues as the "good" one for clustering. This has a computational advantage since the principal eigenvectors can be computed for sparse matrices efficiently using the power iteration technique. The idea is the same as in other spectral clustering methods, i.e., one finds a new representation of patterns on the first k eigenvectors of the Laplacian of the graph.

The algorithm is composed of these steps:

(1) Compute the affinity matrix $A \in \mathbb{R}^{n \times n}$:

$$a_{ij} = \begin{cases} \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right) & \text{if } i \neq j \\ 0 & \text{otherwise} \end{cases} \tag{75}$$

(2) Construct the matrix D
(3) Compute a normalized version of A, defining this Laplacian:

$$L = D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \tag{76}$$

(4) Find the k eigenvectors $\{\mathbf{e}_1, \ldots, \mathbf{e}_k\}$ of L associated to the largest eigenvalues $\{\lambda_1, \ldots, \lambda_k\}$.
(5) Form the matrix Z by stacking the k eigenvectors in columns.
(6) Compute the matrix Y by normalizing each of the Z's rows to have unit length:

$$y_{ij} = \frac{z_{ij}}{\sum_{r=1}^{k} z_{ir}^2} \tag{77}$$

In this way all the original points are mapped into a unit hypersphere.

(7) In this new representation of the original n points apply a clustering algorithm that attempts to minimize distortion such as K-means.

As a criterion to choose σ they suggest to use the value that guarantees the minimum distortion when the clustering stage is performed on Y. They tested this algorithm on artificial data sets showing the capability of the algorithm to separate nonlinear structures.Here we show the steps of the algorithm when applied to the data set in Fig. 1. Once the singular value decomposition of L is computed, we can see the matrices Z and Y in Fig. 4 (here obtained with $\sigma = 0.4$). Once Y is computed, it is easy to cluster the two groups of points

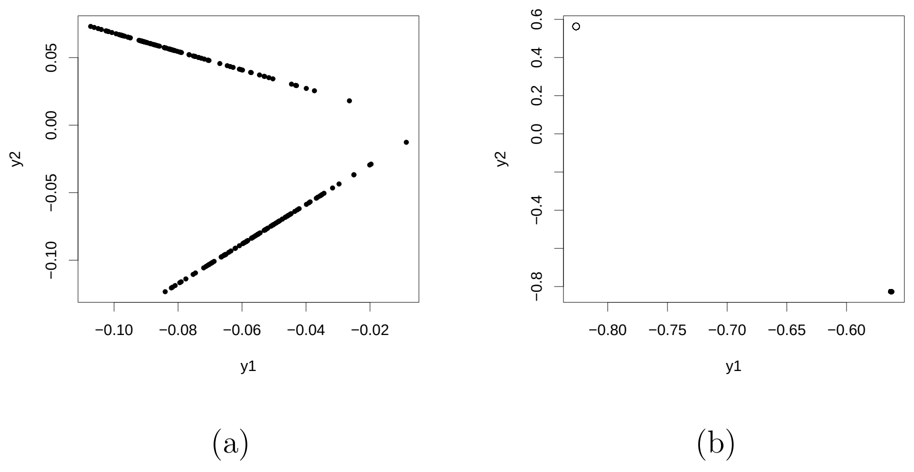


Figure 4. (a) The matrix Z obtained with the first two eigenvectors of the matrix L. (b) The matrix Y obtained by normalizing the rows of Z clustered by K-means algorithm with two centroids.

obtaining the result shown in Fig. 5.

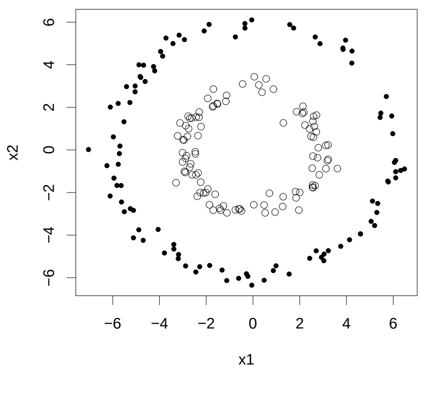


Figure 5. The result of the Ng and Jordan algorithm on the ring data set.

An interesting view of spectral clustering is provided by Meilă et al. [56] who describe it in the framework of Markov random walks [56] leading to a different interpretation of the graph cut problem. It is known, from the theory of Markov random walks, that if we construct the stochastic matrix $P = D^{-1}A$, each element p_{ij} represents the probability of moving from node i to node j. In their work they provide an explicit connection between the spectral decomposition of L and P showing that both have the same solution with eigenvalues of P equal to $1 - \lambda_i$ where λ_i are the eigenvalues of L. Moreover they propose a method to learn a function of the features able to produce a correct segmentation starting from a segmented image.

An interesting study on spectral clustering has been conducted by Kannan et al. [40]. The authors exploit the objective function with respect to some artificial data sets showing that there is no objective function able to properly cluster every data set. In other words there always exists some data set for which the optimization of a particular objective function has some drawback. For this reason they propose a bi-criteria objective function. These two objectives are respectively based on the conductance and the ratio between the auto-association of a subset of nodes S and its volume. Again the relaxation of this problem is achieved by the decomposition of the Laplacian of the graph associated to the data set.

5 A Unified View of Spectral and Kernel Clustering Methods

Recently a possible connection between unsupervised kernel algorithms and spectral methods has been studied to find whether these two seemingly different approaches can be described under a more general framework. The hint for this unifying theory lies the adjacency structure constructed by both these approaches. In the spectral approach there is an adjacency between patterns which is the analogous of the kernel functions in kernel methods.

A direct connection between Kernel PCA and spectral methods has been shown [6,7]. More recently a unifying view of kernel K-means and spectral clustering methods [20,21,23] has been pointed out. In this section we show explicitly the equivalence between them highlighting that these two approaches have the same foundation and in particular that both can be viewed as a matrix trace maximization problem.

To show the direct equivalence between kernel and spectral clustering methods we introduce the weighted version of the kernel K-means [23]. We introduce a weight matrix W having weights w_k on the diagonal. Recalling that we denote with π_i the i-th cluster we have that the functional to minimize is the following:

$$J^{\Phi}(W, V^{\Phi}) = \sum_{i=1}^{c} \sum_{\mathbf{x}_k \in \pi_i} w_k \left\| \Phi(\mathbf{x}_k) - \mathbf{v}_i^{\Phi} \right\|^2 , \tag{78}$$

where:

$$\mathbf{v}_i^{\Phi} = \frac{\sum_{\mathbf{x}_k \in \pi_i} w_k \Phi(\mathbf{x}_k)}{\sum_{\mathbf{x}_k \in \pi_i} w_k} = \frac{\sum_{\mathbf{x}_k \in \pi_i} w_k \Phi(\mathbf{x}_k)}{s_i} , \tag{79}$$

where we have introduced:

$$s_i = \sum_{\mathbf{x}_k \in \pi_i} w_k . \tag{80}$$

Now let's define the matrix Z having:

$$z_{ki} = \begin{cases} s_i^{-1/2} & \text{if } \mathbf{x}_k \in \pi_i \\ 0 & \text{otherwise.} \end{cases} \tag{81}$$

Since the columns of Z are mutually orthogonal it is easy to verify that:

$$s_i^{-1} = (Z^T Z)_{ii} , \tag{82}$$

and that only the diagonal elements are not null.

Now we denote with F the matrix whose columns are the $\Phi(\mathbf{x}_k)$. It is easy to verify that the matrix FW yields a matrix whose columns are the $w_k \Phi(\mathbf{x}_k)$. Moreover the expression $FWZZ^T$ gives a matrix having n columns which are the nearest centroids in feature space of the $\Phi(\mathbf{x}_k)$.

Thus, substituting Eq. 79 in Eq. 78 we obtain the following matrix expression for $J^{\Phi}(W, V^{\Phi})$:

$$J^{\Phi}(W, V^{\Phi}) = \sum_{k=1}^{n} w_k \left\| F_{\cdot k} - (FWZZ^T)_{\cdot k} \right\|^2 \tag{83}$$

Here the dot has to be considered as a selection of the k-th column of the matrices. Introducing the matrix $Y = W^{1/2}Z$, which is orthonormal ($Y^TY = I$), the objective function can be rewritten as:

$$\begin{aligned} J^{\Phi}(W, V^{\Phi}) &= \sum_{k=1}^{n} w_k \left\| F_{\cdot k} - (FW^{1/2}YY^TW^{-1/2})_{\cdot k} \right\|^2 \\ &= \left\| FW^{1/2} - FW^{1/2}YY^T \right\|_{\mathrm{F}}^2 \end{aligned} \tag{84}$$

where the norm $\|\|_{\mathrm{F}}$ is the Frobenius norm [31]. Using the fact that $\|A\|_{\mathrm{F}} = \mathrm{tr}(AA^T)$ and the properties of the trace, it is possible to see that the minimization of the last equation is equivalent to the maximization of the following [20,21]:

$$J^{\Phi}(W, V^{\Phi}) = \mathrm{tr}(Y^TW^{1/2}F^TFW^{1/2}Y) \tag{85}$$

5.2 Spectral clustering methods objective

Recalling that the definition of association between two sets of edges S and T of a weighted graph is the following:

$$assoc(S, T) = \sum_{i \in S, j \in T} a_{ij} \tag{86}$$

it is possible to define many objective functions to optimize in order to perform clustering. Here, for the sake of simplicity, we consider just the ratio association problem, where one has to maximize:

$$J(S_1, \dots, S_c) = \sum_{i=1}^{c} \frac{assoc(S_i, S_i)}{|S_i|} \tag{87}$$

where $|S_i|$ is the size of the i-th partition. Now we introduce the indicator vector $\mathbf{z}_i$ whose k-th value is zero if $\mathbf{x}_k \notin \pi_i$ and one otherwise. Rewriting the last equation in a matrix form we obtain the following:

$$J(S_1, \dots, S_c) = \sum_{i=1}^{c} \frac{\mathbf{z}_i^T A \mathbf{z}_i}{\mathbf{z}_i^T \mathbf{z}_i} \tag{88}$$

Normalizing the $\mathbf{z}_i$ letting:

$$\mathbf{y}_i = \frac{\mathbf{z}_i}{(\mathbf{z}_i^T \mathbf{z}_i)^{1/2}} \tag{89}$$

we obtain:

$$J(S_1, \ldots, S_c) = \sum_{i=1}^{c} \mathbf{y}_i^T A \mathbf{y}_i = \mathrm{tr}(Y^T A Y) \tag{90}$$

5.3 A unified view of the two approaches

Comparing Eq. 90 and Eq. 85 it is possible to see the perfect equivalence between kernel K-means and the spectral approach to clustering when one wants to maximize the ratio association. To this end, indeed, it is enough to set the weights in the weighted kernel K-means equal to one obtaining the classical kernel K-means. It is possible to obtain more general results when one wants to optimize other objective functions in the spectral approach, such as the ratio cut [13], the normalized cut and the Kernighan-Lin [41] objective. For instance, in the case of the minimization of the normalized cut which is one of the most used objective functions, the functional to minimize is:

$$J(S_1, \ldots, S_c) = \mathrm{tr}(Y^T D^{-1/2} A D^{-1/2} Y) \tag{91}$$

Thus the correspondence with the objective in the kernel K-means imposes to choose $Y = D^{1/2} Z$, $W = D$ and $K = D^{-1} A D^{-1}$. It is worth noting that for an arbitrary A it is not guaranteed that $D^{-1} A D^{-1}$ is definite positive. In this case the kernel K-means will not necessarily converge. To cope with this problem in [20] the authors propose to enforce positive definiteness by means of a diagonal shift [66]:

$$K = \sigma D^{-1} + D^{-1} A D^{-1} \tag{92}$$

where σ is a positive coefficient large enough to guarantee the positive definiteness of K. Since the mathematical foundation of these methods is the same, it is possible to choose which algorithm to use for clustering choosing, for instance, the approach with the less computational complexity for the particular application.

6 Conclusions

Clustering is a classical problem in pattern recognition. Recently spectral and kernel methods for clustering have provided new ideas and interpretations to the solution of this problem. In this paper spectral and kernel methods for clustering have been reviewed paying attention to fuzzy kernel methods for clustering and to the connection between kernel and spectral approaches. Unlike classical partitioning clustering algorithms they are able to produce nonlinear separating hypersurfaces among data since they construct an adjacency structure from data. These methods have been successfully tested on several benchmarks, but we can find few applications to real world problem due to the high computational cost. Therefore an extensive validation on real world applications remains a big challenge for kernel and spectral clustering methods.

Acknowledgments

Work partially supported by the Italian Ministry of Education, University, and Research (PRIN 2004 Project 2004062740) and by the University of Genova.

References

[1] M. Aizerman, E. Braverman, and L. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

[2] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404, 1950.

[3] Majid A. Awan and Mohd. An intelligent system based on kernel methods for crop yield prediction. In Wee K. Ng, Masaru Kitsuregawa, Jianzhong Li, and Kuiyu Chang, editors, *PAKDD*, volume 3918 of *Lecture Notes in Computer Science*, pages 841–846, 2006.

[4] F. R. Bach and M. I. Jordan. Learning spectral clustering. Technical Report UCB/CSD-03-1249, EECS Department, University of California, Berkeley, 2003.

[5] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, June 2003.

[6] Y. Bengio, O. Delalleau, N. Le Roux, J. F. Paiement, P. Vincent, and M. Ouimet. Learning eigenfunctions links spectral embedding and kernel PCA. *Neural Computation*, 16(10):2197–2219, 2004.

[7] Y. Bengio, P. Vincent, and J. F. Paiement. Spectral clustering and kernel PCA are learning eigenfunctions. Technical Report 2003s-19, CIRANO, 2003.

[8] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms.* Kluwer Academic Publishers, Norwell, MA, USA, 1981.

[9] C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, Oxford, UK, 1996.

[10] M. Brand and K. Huang. A unifying theorem for spectral embedding and clustering. In Christopher M. Bishop and Brendan J. Frey, editors, *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, 2003.

[11] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[12] F. Camastra and A. Verri. A novel kernel method for clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(5):801–804, 2005.

[13] P. K. Chan, M. Schlag, and J. Y. Zien. Spectral k-way ratio-cut partitioning and clustering. In *Proceeding of the 1993 symposium on Research on integrated systems*, pages 123–142, Cambridge, MA, USA, 1993. MIT Press.

[14] S.-C. Chen and D.-Q. Zhang. Robust image segmentation using FCM with spatial constraints based on new kernel-induced distance measure. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 34(4):1907–1916, 2004.

[15] J.-H. Chiang and P.-Y. Hao. A new kernel-based fuzzy clustering approach: support vector clustering with cell growing. *IEEE Transactions on Fuzzy Systems*, 11(4):518–527, 2003.

[16] F. R. K. Chung. *Spectral Graph Theory (CBMS Regional Conference Series in Mathematics, No. 92).* American Mathematical Society, February 1997.

[17] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

[18] N. Cristianini, J. S. Taylor, A. Elisseeff, and J. S. Kandola. On kernel-target alignment. In *NIPS*, pages 367–373, 2001.

[19] N. Cristianini, J. S. Taylor, and J. S. Kandola. Spectral kernel methods for clustering. In *NIPS*, pages 649–655, 2001.

[20] I. S. Dhillon, Y. Guan, and B. Kulis. Weighted graph cuts without eigenvectors: A multilevel approach. to appear in IEEE Transactions on Pattern Analysis and Machine Intelligence, 2007.

[21] I. S. Dhillon, Y. Guan, and B. Kulis. A unified view of kernel k-means, spectral clustering and graph partitioning. Technical Report Technical Report TR-04-25, UTCS, 2005.

[22] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–274, New York, NY, USA, 2001. ACM Press.

[23] I. S. Dhillon, Y. Guan, and B. Kulis. Kernel k-means: spectral clustering and normalized cuts. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 551–556, New York, NY, USA, 2004. ACM Press.

[24] W. E. Donath and A. J. Hoffman. Lower bounds for the partitioning of graphs. *IBM Journal of Research and Development*, 17:420–425, 1973.

[25] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* Wiley, 1973.

[26] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(98):298–305, 1973.

[27] I. Fischer and I. Poland. New methods for spectral clustering. Technical Report IDSIA-12-04, IDSIA, 2004.

[28] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals Eugenics*, 7:179–188, 1936.

[29] A. Gersho and R. M. Gray. *Vector quantization and signal compression.* Kluwer, Boston, 1992.

[30] M. Girolami. Mercer kernel based clustering in feature space. *IEEE Transactions on Neural Networks*, 13(3):780–784, 2002.

[31] G. H. Golub and C. F. V. Loan. *Matrix Computations (Johns Hopkins Studies in Mathematical Sciences).* The Johns Hopkins University Press, October 1996.

[32] T. Graepel and K. Obermayer. Fuzzy topographic kernel clustering. In W. Brauer, editor, *Proceedings of the 5th GI Workshop Fuzzy Neuro Systems '98*, pages 90–97, 1998.

[33] A. S. Have, M. A. Girolami, and J. Larsen. Clustering via kernel decomposition. *IEEE Transactions on Neural Networks*, 2006.

[34] D. Horn. Clustering via Hilbert space. *Physica A Statistical Mechanics and its Applications*, 302:70–79, December 2001.

[35] P. J. Huber. *Robust Statistics.* John Wiley and Sons, New York, 1981.

[36] A. B. Hur, D. Horn, H. T. Siegelmann, and V. Vapnik. A support vector method for clustering. In Todd, editor, *NIPS*, pages 367–373, 2000.

[37] A. B. Hur, D. Horn, H. T. Siegelmann, and V. Vapnik. Support vector clustering. *Journal of Machine Learning Research*, 2:125–137, 2001.

[38] R. Inokuchi and S. Miyamoto. LVQ clustering and SOM using a kernel function. In *Proceedings of IEEE International Conference on Fuzzy Systems*, volume 3, pages 1497–1500, 2004.

[39] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[40] R. Kannan, S. Vempala, and A. Vetta. On clusterings: Good, bad, and spectral. In *Proceedings of the 41st Annual Symposium on the Foundation of Computer Science*, pages 367–380. IEEE Computer Society, November 2000.

[41] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell system technical journal*, 49(1):291–307, 1970.

[42] Y. Kluger, R. Basri, J. T. Chang, and M. Gerstein. Spectral biclustering of microarray data: coclustering genes and conditions. *Genome Research*, 13(4):703–716, April 2003.

[43] T. Kohonen. The self-organizing map. In *Proceedings of the IEEE*, volume 78, pages 1464–1480, 1990.

[44] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69, 1982.

[45] T. Kohonen. *Self-Organizing Maps.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001.

[46] R. Krishnapuram and J. M. Keller. A possibilistic approach to clustering. *IEEE Transactions on Fuzzy Systems*, 1(2):98–110, 1993.

[47] R. Krishnapuram and J. M. Keller. The possibilistic c-means algorithm: insights and recommendations. *IEEE Transactions on Fuzzy Systems*, 4(3):385–393, 1996.

[48] B. Kulis, S. Basu, I. S. Dhillon, and R. Mooney. Semi-supervised graph clustering: a kernel approach. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 457–464, New York, NY, USA, 2005. ACM Press.

[49] D. Lee. An improved cluster labeling method for support vector clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):461–464, 2005.

[50] J. Leski. Fuzzy c-varieties/elliptotypes clustering in reproducing kernel hilbert space. *Fuzzy Sets and Systems*, 141(2):259–280, 2004.

[51] Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 1:84–95, 1980.

[52] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28:129–137, 1982.

[53] D. Macdonald and C. Fyfe. The kernel self-organising map. In *Fourth International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies, 2000*, volume 1, pages 317–320, 2000.

[54] J. B. Macqueen. Some methods of classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathemtical Statistics and Probability*, pages 281–297, 1967.

[55] T. M. Martinetz, S. G. Berkovich, and K. J. Schulten. 'Neural gas' network for vector quantization and its application to time-series prediction. *IEEE Transactions on Neural Networks*, 4(4):558–569, 1993.

[56] M. Meila and J. Shi. Learning segmentation by random walks. In *NIPS*, pages 873–879, 2000.

[57] J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Proceedings of the Royal Society of London*, 209:415–446, 1909.

[58] K. R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–202, 2001.

[59] O. Nasraoui and R. Krishnapuram. An improved possibilistic c-means algorithm with finite rejection and robust scale estimation. In *North American Fuzzy Information Processing Society Conference*, Berkeley, California, June 1996.

[60] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

[61] A. Paccanaro, C. Chennubhotla, J. A. Casbon, and M. A. S. Saqi. Spectral clustering of protein sequences. In *International Joint Conference on Neural Networks*, volume 4, pages 3083–3088, 2003.

[62] A. K. Qinand and P. N. Suganthan. Kernel neural gas algorithms with application to cluster analysis. *ICPR*, 04:617–620, 2004.

[63] A. Rahimi and B. Recht. Clustering with normalized cuts is clustering with a hyperplane. *Statistical Learning in Computer Vision*, 2004.

[64] H. J. Ritter, T. M. Martinetz, and K. J. Schulten. *Neuronale Netze*. Addison-Wesley, München, Germany, 1991.

[65] K. Rose. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *Proceedings of IEEE*, 86(11):2210–2239, November 1998.

[66] V. Roth, J. Laub, M. Kawanabe, and J. M. Buhmann. Optimal cluster preserving embedding of nonmetric proximity data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(12):1540–1551, 2003.

[67] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, December 2000.

[68] S. Saitoh. *Theory of Reproducing Kernels and its Applications*. Longman Scientific & Technical, Harlow, England, 1988.

[69] D. S. Satish and C. C. Sekhar. Kernel based clustering and vector quantization for speech recognition. In *Proceedings of the 2004 14th IEEE Signal Processing Society Workshop*, pages 315–324, 2004.

[70] B. Schölkopf, A. J. Smola, and K. R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

[71] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, Cambridge, MA, USA, 2001.

[72] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2000.

[73] V. G. Sigillito, S. P. Wing, L. V. Hutton, and K. B. Baker. Classification of radar returns from the ionosphere using neural networks. *Johns Hopkins APL Technical Digest*, 10:262–266, 1989.

[74] P. H. A. Sneath and R. R. Sokal. *Numerical Taxonomy: The Principles and Practice of Numerical Classification.* W.H. Freeman, San Francisco, 1973.

[75] A. N. Srivastava. Mixture density Mercer kernels: A method to learn kernels directly from data. In *SDM*, 2004.

[76] X. Tan, S. Chen, Z. H. Zhou, and F. Zhang. Robust face recognition from a single training image per person with kernel-based som-face. In *ISNN (1)*, pages 858–863, 2004.

[77] D. M. J. Tax and R. P. W. Duin. Support vector domain description. *Pattern Recognition Letters*, 20(11-13):1191–1199, 1999.

[78] V. N. Vapnik. *The nature of statistical learning theory.* Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[79] D. Verma and M. Meila. A comparison of spectral clustering algorithms. Technical report, Department of CSE University of Washington Seattle, WA 98195-2350, 2005.

[80] U. von Luxburg, M. Belkin, and O. Bousquet. Consistency of spectral clustering. Technical Report 134, Max Planck Institute for Biological Cybernetics, 2004.

[81] U. von Luxburg, O. Bousquet, and M. Belkin. Limits of spectral clustering. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems (NIPS) 17.* MIT Press, Cambridge, MA, 2005.

[82] D. Wagner and F. Wagner. Between min cut and graph bisection. In *Mathematical Foundations of Computer Science*, pages 744–750, 1993.

[83] J. H. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58:236–244, 1963.

[84] J. Weston, C. Leslie, E. Ie, D. Zhou, A. Elisseeff, and W. S. Noble. Semi-supervised protein classification using cluster kernels. *Bioinformatics*, 21(15):3241–3247, August 2005.

[85] W. H. Wolberg and O. L. Mangasarian. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. *Proceedings of the National Academy of Sciences,U.S.A.*, 87:9193–9196, 1990.

[86] Z. D. Wu, W. X. Xie, and J. P. Yu. Fuzzy c-means clustering algorithm based on kernel method. *Computational Intelligence and Multimedia Applications*, 2003.

[87] R. Xu and D. I. I. Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.

[88] J. Yang, Estivill, and S. K. Chalup. Support vector clustering through proximity graph modelling. In *Proceedings of the 9th International Conference on Neural Information Processing*, volume 2, pages 898–903, 2002.

[89] S. X. Yu and J. Shi. Multiclass spectral clustering. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, Washington DC, USA, 2003. IEEE Computer Society.

[90] H. Zha, X. He, C. H. Q. Ding, M. Gu, and H. D. Simon. Spectral relaxation for k-means clustering. In *NIPS*, pages 1057–1064, 2001.

[91] D.-Q. Zhang and S.-C. Chen. Fuzzy clustering using kernel method. In *The 2002 International Conference on Control and Automation, 2002. ICCA*, pages 162–163, 2002.

[92] D.-Q. Zhang and S.-C. Chen. Kernel based fuzzy and possibilistic c-means clustering. In *Proceedings of the International Conference Artificial Neural Network*, pages 122–125. Turkey, 2003.

[93] D.-Q. Zhang and S.-C. Chen. A novel kernelized fuzzy c-means algorithm with application in medical image segmentation. *Artificial Intelligence in Medicine*, 32(1):37–50, 2004.

[94] D.-Q. Zhang, S.-C. Chen, Z.-S. Pan, and K.-R. Tan. Kernel-based fuzzy clustering incorporating spatial constraints for image segmentation. In *International Conference on Machine Learning and Cybernetics*, volume 4, pages 2189–2192, 2003.