A survey of kernel and spectral methods for clustering

M. Filippone[1], F. Camastra[2], F. Masulli[1], S. Rovetta[1]
1 - Dipartimento di Informatica e Scienze dell'Informazione, Università di Genova, and CNISM,
Via Dodecaneso 35, I-16146 Genova, Italy
email - {filippone, masulli, rovetta}@disi.unige.it
2 - Dipartimento di Scienze Applicate, Università di Napoli Parthenope,
Via A. De Gasperi 5, I-80133 Napoli, Italy
email - francesco.camastra@uniparthenope.it

18th October 2006

Technical report DISI-TR-06-19

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 extension of kernel K-means clustering algorithm.

1 Introduction

Unsupervised data analysis using clustering algorithms provide a useful tool to explore data structures. Clustering methods [40], [82] 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 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. For instance, we can imagine to cluster documents by using the occurrences of a set of keywords in order to group together documents with the same content. Using the occurrences of a set of articles would lead to worse results in comparison with the previous choice. Whether an appropriate set of features can be chosen or not it 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 [26].

Clustering techniques can be roughly divided into two categories:

- *hierarchical*;
- *partitioning*.

Hierarchical clustering techniques [40], [70], [78] 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 [9], 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 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 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 separation between clusters in the input space.

Spectral clustering methods arise from concepts in spectral graph theory. The basic idea is to construct, from the initial data set, a weighted graph. Each node represent 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 spectral

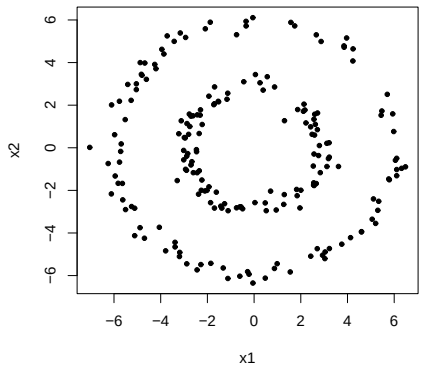


Figure 1: A data set composed of two rings of points.

graph theory. The core of this theory is the singular values decomposition of the Laplacian matrix of the weighted graph obtained from data, which is related to its cut.

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 given. In particular Kernel K-Means and spectral clustering with the ratio association as the objective function are perfectly equivalent, allowing to use the less expensive approach in terms of complexity for a particular application.

The core of both approaches lies in their ability of construct an adjacency structure between data avoiding to deal with a prefixed shape of clusters. These approach 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 [75], while there are some theoretical works on the capabilities and convergence properties of spectral methods for clustering [41], [76], [77], [85]. Recently kernel methods have been applied even to Fuzzy c-Varieties [51] with the aim of finding varieties in feature space. There are some interesting methods that use kernels such as [34] and [35].

Since the choice of the kernel and of the similarity measure is crucial in these methods, many techniques have been implemented in order to learn automatically the shape of kernels from data as in [5], [20], [28], [56].

Regarding the applications, most of these algorithms (e.g., [14], [20], [51]) have been applied to standard benchmarks such as Ionosphere [69], Breast Cancer [80] and Iris [29][1]. Kernel Fuzzy c-Means proposed in [16], [88], [89] has been applied in image segmentation problems while in [33] it has been applied in handwritten digits recognition. The kernel SOM has been applied in face recognition in [72]. There are applications of kernel clustering methods in speech recognition in [66] and in prediction of crop yield from climate and plantation data [4]. Spectral methods have been applied in clustering

[1]They can be found at: ftp://ftp.ics.uci.edu/pub/machine-learning-databases/

of artificial data [59], [63], in image segmentation [6], [56], [68], [71], in bioinformatics [21], and in co-clustering problems of words and documents [23] and genes and conditions [43]. A semi-supervised spectral approach to bioinformatics and handwritten character recognition have been proposed in [49]. The protein sequence clustering problem has been faced using spectral techniques in [60] and kernel methods in [79].

In the next section we briefly introduce the concepts of linear partitioning methods 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 show explicitly 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 methods for partitioning a space 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 $V = \{\mathbf{v}_1, \ldots, \mathbf{v}_c\}$, 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 $\mathbf{v}_i$ 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 [52] 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:

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

The partitioning of $\mathbb{R}^d$ formed by all Voronoi regions is called *Voronoi tessellation* or *Dirichlet tessellation*.

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], [54] and it is known as *batch K-Means*. Starting from a finite data set 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*:

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

[2]There are many terms to denote such objects since there are applications of clustering in many disciplines. Here we will use codevector as in vector quantization theory.

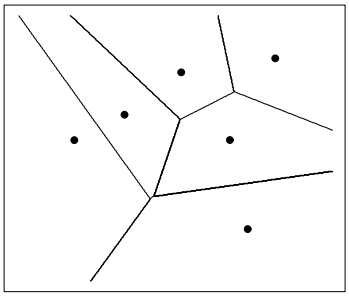


Figure 2: An example of Voronoi tessellation where each black point is a codevector.

is that each codevector $\mathbf{v}_i$ fulfills the *centroid condition* [30]. In the case of finite data set X and 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 chosen randomly 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* (*EM*) [10] algorithm, ensuring the convergence after a finite number of step.

This approach presents many disadvantages [26]. First of all the presence of local minima of $E(X)$ and *outliers* (points whose position is far from the others) can strongly influence the results. Then the number of clusters to find must be provided and this can be done only using some a priori information. 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 invariances as in the case of Mahalanobis distances which produces invariance on ellipsoids [26].

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., ten thousand) the batch procedure is computationally expensive. For this reason an on-line procedure of update has been introduced leading to the *on-line K-Means* algorithm. At each step, this method simply chooses randomly an input patterns and updates its nearest codevector.

2.2 Self Organizing Maps - SOM

Self Organizing Maps (*SOM*) [44] is used as an online clustering algorithm although it was designed as a nonlinear embedding (dimensionality reduction) technique. There are several models of SOM but in this paper we consider the one in which the centroids topology is constrained to be a two dimensional grid [45], [46]. The distance on the grid is used to determine how strongly a unit is adapted when the unit a_{ij} is the winner. The metric used on the grid is the Manhattan distance in the case of a rectangular one. The distance between two elements $\mathbf{r} = (r_1, r_2)$ and $\mathbf{s} = (s_1, s_2)$ on the grid is:

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

The SOM algorithm is the following:

1. Initialize the codebook V choosing randomly 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. Choose randomly 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 function decreasing with d and can be:
$$h(d_{rs}) = \exp\left(-\frac{d_{rs}^2}{2\sigma^2(t)}\right) \tag{8}$$
7. Increment t
8. if $t > t_{\max}$ stop else go to step 4

The functions $\epsilon(t)$ and $\sigma(t)$ are decreasing as t increases, for example as in:

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

$$\epsilon = \epsilon_i \left(\frac{\epsilon_\mathrm{f}}{\epsilon_\mathrm{i}}\right)^{t/t_{\max}} \tag{10}$$

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. The difference with the SOM is that codevectors are not constrained to be on a grid and that the adaptation of the codevectors near the winner are update on the basis of a rank criteria on distances. Each time a pattern $\mathbf{x}$ is presented, the decreasing ranking of the distances between $\mathbf{x}$ and all the codevectors $\mathbf{v}_j$ is computed. Denoting with ρ_j the rank of the distance between $\mathbf{x}$ and the codevector $\mathbf{v}_j$, the update rule is:

$$\Delta \mathbf{v}_j = \epsilon(t) h_\lambda(\rho_j)(\mathbf{x} - \mathbf{v}_j) \tag{11}$$

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 to contain codevectors $\mathbf{v}_j$ with vectors chosen *randomly* from the training set X
2. Initialize the time parameter $t = 0$
3. Choose randomly 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. 11
6. Increase the time parameter $t = t + 1$
7. if $t < t_{\max}$ go to step 3.

2.4 Fuzzy clustering methods

Bezdek [9] introduced the concept of hard and fuzzy partition in order to extend the notion of membership of a point to a cluster. The motivation of this extension is related to the fact that a data point often cannot be thought as belonging to a single cluster only. In many cases, sharing the membership of a pattern among many cluster 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:*

$$M_{\mathrm{fc}} = \left\{ U \in A_{cn} \;\middle|\; u_{ik} \in [0,1]\ \forall i,k;\ \sum_{i=1}^{c} u_{ik} = 1\ \forall k\ ;\ 0 < \sum_{k=1}^{n} u_{ik} < n\ \forall i \right\} \tag{12}$$

The matrix U is the so called *membership matrix* since the element u_{ik} is the fuzzy membership of the k-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 [9].

The mathematical tool used in all these method 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 point 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. At each iteration a subset of parameters is fixed while the other is optimized until variables change less than a fixed threshold (Picard iterations method).

2.4.1 Fuzzy c-Means

Fuzzy c-Means algorithm [9] identifies clusters as fuzzy sets. It minimizes the functional:

$$J(U,V) = \sum_{k=1}^{n} \sum_{i=1}^{c} (u_{ik})^m \left\| \mathbf{x}_k - \mathbf{v}_i \right\|^2 \tag{13}$$

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

$$\sum_{i=1}^{c} u_{ik} = 1 \qquad \forall i = 1, \ldots, n \tag{14}$$

The parameter m controls the softness 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. Minimization of eq. 13 is done introducing a Lagrangian function for each pattern in which the constraint is eq. 14.

$$L_k = \sum_{i=1}^{c} (u_{ik})^m \left\| \mathbf{x}_k - \mathbf{v}_i \right\|^2 + \alpha_k \left(\sum_{i=1}^{c} u_{ik} - 1 \right) \tag{15}$$

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

$$u_{ik}^{-1} = \sum_{j=1}^{c} \left(\frac{\left\| \mathbf{x}_k - \mathbf{v}_i \right\|}{\left\| \mathbf{x}_k - \mathbf{v}_j \right\|} \right)^{\frac{2}{m-1}} \tag{16}$$

$$\mathbf{v}_i = \frac{\sum_{k=1}^{n} (u_{ik})^m \mathbf{x}_k}{\sum_{k=1}^{n} (u_{ik})^m} \tag{17}$$

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, the possibilistic approach [47], [48] relaxes the probabilistic constraint on the membership of a pattern to a 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 memberships represents of a degree of typicality not depending on the membership values of the same point in 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 [47] and PCM-II [48]. The first one aims to minimize the following functional:

$$J(U,V) = \sum_{k=1}^{n}\sum_{i=1}^{c}(u_{ik})^m \|\mathbf{x}_k - \mathbf{v}_i\|^2 + \sum_{i=1}^{c}\eta_i \sum_{k=1}^{n}(1-u_{ik})^m \tag{18}$$

while the second one addresses the functional:

$$J(U,V) = \sum_{k=1}^{n}\sum_{i=1}^{c}u_{ik}\|\mathbf{x}_k - \mathbf{v}_i\|^2 + \sum_{i=1}^{c}\eta_i \sum_{k=1}^{n}(u_{ik}\ln(u_{ik}) - u_{ik}) \tag{19}$$

with respect to the membership matrix U and the codebook $V = \{\mathbf{v}_1, \ldots, \mathbf{v}_c\}$. The minimization of eq. 18 and eq. 19 with respect to the u_{ik} leads respectively to the iteration of these equations:

$$u_{ik}^{-1} = 1 + \left(\frac{\|\mathbf{x}_k - \mathbf{v}_i\|^2}{\eta_i}\right)^{\frac{1}{m-1}} \tag{20}$$

$$u_{ik} = \exp\left(-\frac{\|\mathbf{x}_k - \mathbf{v}_i\|^2}{\eta_i}\right) \tag{21}$$

The constraint on the memberships $u_{ik} \in [0,1]$ is automatically satisfied given the form assumed by eq. 20 and eq. 21. The updates of the centroids for PCM-I and PCM-II are respectively:

$$\mathbf{v}_i = \frac{\sum_{k=1}^{n}(u_{ik})^m \mathbf{x}_k}{\sum_{k=1}^{n}(u_{ik})^m} \tag{22}$$

$$\mathbf{v}_i = \frac{\sum_{k=1}^{n}u_{ik}\mathbf{x}_k}{\sum_{k=1}^{n}u_{ik}} \tag{23}$$

The parameter η_i regulates the tradeoff between the two terms in eq. 18 and eq. 19 and it is related to the width of the clusters. The authors suggest to estimate η_i for PCM-I using this formula:

$$\eta_i = \gamma\frac{\sum_{k=1}^{n}(u_{ik})^m \|\mathbf{x}_k - \mathbf{v}_i\|^2}{\sum_{k=1}^{n}(u_{ik})^m} \tag{24}$$

which is a weighted mean of the intracluster distance of the i-th cluster and the constant γ is typically chosen to be one[3]. The parameter 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 results have been obtained considering η_i fixed. In the latter one a good estimation of η_i can be done only when an approximate solution is achieved. For this reason often the Possibilistic c-Means is run as a refining step of a Fuzzy c-Means.

[3]Selection of η_i can also be approached with other scale estimation techniques as developed in the robust clustering literature for M-estimators [36], [58].

3 Kernel Clustering Methods

In machine learning, the use of the kernels functions has been introduced by Aizerman et al. [1] in 1964. In 1995 Cortes and Vapnik introduced *Support Vector Machines* (SVMs) [19] which perform better than other machine learning algorithms in several problems. The success of SVM has brought to extend the use of kernels to other learning algorithm (e.g. *Kernel PCA* [67]). The choice of the kernel is crucial to incorporate a priori knowledge on the application, for which sometimes it is needed to design *ad hoc* kernels.

3.1 Mercer kernels

We recall the definition of Mercer kernels [3] [65], 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:*

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j K(\mathbf{x}_i, \mathbf{x}_j) \geq 0 \quad \forall n \geq 2 \tag{25}$$

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

Each Mercer kernel can be expressed in terms of:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \tag{26}$$

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 aspect in applications is that it is possible to compute Euclidean distances in $\mathcal{F}$ without knowing explicitly Φ.

$$\begin{aligned} \|\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)\|^2 &= (\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)) \cdot (\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)) = \\ &= \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_i) + \Phi(\mathbf{x}_j) \cdot \Phi(\mathbf{x}_j) - 2\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) = \\ &= K(\mathbf{x}_i, \mathbf{x}_i) + K(\mathbf{x}_j, \mathbf{x}_j) - 2K(\mathbf{x}_i, \mathbf{x}_j) \end{aligned} \tag{27}$$

which is a function of the input vectors. This is known as *distance kernel trick* [67]. 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. 27 will be rewritten as:

$$\|\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)\|^2 = k_{ii} + k_{jj} - 2k_{ij} \tag{28}$$

Examples of Mercer kernels are the following [74]:

- linear:

$$K^{(\mathrm{l})}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j \tag{29}$$

- polynomial with degree p:

$$K^{(\mathrm{p})}(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i \cdot \mathbf{x}_j)^p \qquad p \in \mathbb{N} \tag{30}$$

- Gaussian:

$$K^{(\mathrm{g})}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right) \qquad \sigma \in \mathbb{R} \tag{31}$$

It is important to stress that the use of the linear kernel in eq. 27 simply leads to the computation of the Euclidean norm in the input space. In fact:

$$\begin{aligned} \|\mathbf{x}_i - \mathbf{x}_j\|^2 &= \mathbf{x}_i \cdot \mathbf{x}_i + \mathbf{x}_j \cdot \mathbf{x}_j - 2\mathbf{x}_i \cdot \mathbf{x}_j = \\ &= K^{(\mathrm{l})}(\mathbf{x}_i, \mathbf{x}_i) + K^{(\mathrm{l})}(\mathbf{x}_j, \mathbf{x}_j) - 2K^{(\mathrm{l})}(\mathbf{x}_i, \mathbf{x}_j) = \\ &= \|\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)\|^2 \end{aligned} \tag{32}$$

that shows that choosing the kernel $K^{(\mathrm{l})}$ implies $\Phi = I$ (where I is the identity function). Following this consideration we can think of the representation via kernels as a more general way to represent the elements of a set X.

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

- *kernelization of the metric* [81], [87], [88];
- *clustering in feature space* [33], [39], [53], [62], [86];
- *description via support vectors* [14], [38].

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

$$\|\Phi(\mathbf{x}_k) - \Phi(\mathbf{v}_i)\|^2 = K(\mathbf{x}_k, \mathbf{x}_k) + K(\mathbf{v}_i, \mathbf{v}_i) - 2K(\mathbf{x}_k, \mathbf{v}_i) \tag{33}$$

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 distance $\left\|\Phi(\mathbf{x}_k) - \mathbf{v}_i^\Phi\right\|^2$ 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 Support Vector Clustering algorithm allows to label the enclosed patterns in feature space. 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 $i = 1, \ldots, k$) [31], [67]. 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

$$R_i^\Phi = \left\{ \mathbf{x}^\Phi \in \mathcal{F} \;\middle|\; i = \arg\min_j \left\| \mathbf{x}^\Phi - \mathbf{v}_j^\Phi \right\| \right\} \tag{34}$$

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* for their images $\Phi(\mathbf{x})$ in the feature space:

$$\pi_i^\Phi = \left\{ \mathbf{x} \in X \;\middle|\; i = \arg\min_j \left\| \Phi(\mathbf{x}) - \mathbf{v}_j^\Phi \right\| \right\} \tag{35}$$

These definitions, as in the case of the codevectors in the input space, induce 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 $\mathcal{F}$
$$\mathbf{v}_i^\Phi = \frac{1}{\left|\pi_i^\Phi\right|} \sum_{\mathbf{x} \in \pi_i^\Phi} \Phi(\mathbf{x}) \tag{36}$$
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. 36. 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 the centroid in feature space as a combination of data vectors in feature space we have:

$$\mathbf{v}_j^\Phi = \sum_{k=1}^{n} \gamma_{jk} \Phi(\mathbf{x}_k) \tag{37}$$

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

$$\left\| \Phi(\mathbf{x}_i) - \mathbf{v}_j^\Phi \right\|^2 = \left\| \Phi(\mathbf{x}_i) - \sum_{k=1}^{n} \gamma_{jk} \Phi(\mathbf{x}_k) \right\|^2 \tag{38}$$

can be expanded by using the scalar product and the kernel trick:

$$\left\| \Phi(\mathbf{x}_i) - \sum_{k=1}^{n} \gamma_{jk} \Phi(\mathbf{x}_k) \right\|^2 = k_{ii} - 2 \sum_k \gamma_{jk} k_{ik} + \sum_r \sum_s \gamma_{jr} \gamma_{js} k_{rs} \tag{39}$$

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

An online version of the K-Means algorithm can be found in [67]. A further version of K-Means in feature space have been proposed by Girolami [31]. In his formulation the number of clusters is denoted by c and a fuzzy membership matrix U is introduced. Each element u_{ik} denotes the fuzzy membership of the point $\mathbf{x}_k$ to the Voronoi set π_i^Φ. This algorithm tries to minimize the following functional with respect to U:

$$J^\Phi(U, V^\Phi) = \sum_{k=1}^{n} \sum_{i=1}^{c} u_{ik} \left\| \Phi(\mathbf{x}_k) - \mathbf{v}_i^\Phi \right\|^2 \tag{40}$$

The minimization technique used by Girolami is *Deterministic Annealing* [64] which is a stochastic method for optimization. A parameter controls the softness 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 and 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 [39], [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:

$$\mathbf{v}_j^\Phi = \sum_{k=1}^{n} \gamma_{jk} \Phi(\mathbf{x}_k) \tag{41}$$

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

$$\mathbf{s}(\Phi(\mathbf{x}_i)) = \arg \min_{\mathbf{v}_j^\Phi \in V} \|\Phi(\mathbf{x}_i) - \mathbf{v}_j^\Phi\| \tag{42}$$

that can be written, using the kernel trick:

$$\mathbf{s}(\Phi(\mathbf{x}_i)) = \arg \min_{\mathbf{v}_j^\Phi \in V} k_{ii} - 2 \sum_k \gamma_{jk} k_{ik} \sum_r \sum_s \gamma_{jr} \gamma_{js} k_{rs} \tag{43}$$

To update the codevectors we rewrite eq. 7:

$$\mathbf{v}_j^{\Phi\prime} = \mathbf{v}_j^\Phi + \epsilon(t) h(d_{rs}) \left(\Phi(\mathbf{x}) - \mathbf{v}_j^\Phi \right) \tag{44}$$

Using eq. 41:

$$\sum_{k=1}^{n} \gamma'_{jk} \Phi(\mathbf{x}_k) = \sum_{k=1}^{n} \gamma_{jk} \Phi(\mathbf{x}_k) + \epsilon(t) h(d_{rs}) \left(\Phi(\mathbf{x}) - \sum_{k=1}^{n} \gamma_{jk} \Phi(\mathbf{x}_k) \right) \tag{45}$$

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

$$\gamma'_{jk} = \begin{cases} (1 - \epsilon(t) h(d_{rs})) \gamma_{jk} & \text{if } i \neq j \\ (1 - \epsilon(t) h(d_{rs})) \gamma_{jk} + \epsilon(t) h(d_{rs}) & \text{otherwise} \end{cases} \tag{46}$$

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. 11 in feature space for the update of the codevectors we have:

$$\Delta \mathbf{v}_j^{\Phi} = \epsilon h_\lambda(\rho_j) \left(\Phi(\mathbf{x}) - \mathbf{v}_j^{\Phi} \right) \tag{47}$$

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. 41, 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 [37], [38], [73]. 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:

$$\|\Phi(\mathbf{x}_j) - \mathbf{v}\|^2 \leq R^2 \quad \forall j \tag{48}$$

This constraint can be relaxed adding slack variables ξ_j:

$$\|\Phi(\mathbf{x}_j) - \mathbf{v}\|^2 \leq R^2 + \xi_j \quad \forall j \tag{49}$$

The Lagrangian for this problem is defined [13]:

$$L = R^2 - \sum_j (R^2 + \xi_j - \|\Phi(\mathbf{x}_j) - \mathbf{v}\|^2)\beta_j - \sum_j \xi_j \mu_j + C \sum_j \xi_j \tag{50}$$

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 equating them to zero leads to the following equations:

$$\sum_j \beta_j = 1 \tag{51}$$

$$\mathbf{v} = \sum_j \beta_j \Phi(\mathbf{x}_j) \tag{52}$$

$$\beta_j = C - \mu_j \tag{53}$$

The Karush-Kuhn-Tucker (KKT) complementary conditions [13] result in:

$$\xi_j \mu_j = 0 \tag{54}$$

$$(R^2 + \xi_j - \|\Phi(\mathbf{x}_j) - \mathbf{v}\|^2)\beta_j = 0 \tag{55}$$

Since $\beta_j \geq 0$, if the image of a point $\mathbf{x}_j$ lies outside the enclosing sphere $\xi_j > 0$. This follows from eq. 55. Using the second KKT complementary condition, for such points, it follows that μ_j must

be zero. These points are called bounded support vectors (BSV) and for them $\beta_j = C$. When $0 < \beta_j < C$, from eq. 55 it follows that $\mathbf{x}_j$ is mapped on the surface of the sphere. These points are called support vector (SV). If $\xi_j = 0$, it is easy to see that the point $\mathbf{x}_j$ is mapped inside the sphere. Values of C greater than one are not allowed since eq. 51 must hold. From these considerations we can write the Wolfe dual form:

$$J_W = \sum_j \Phi^2(\mathbf{x}_j)\beta_j - \sum_i \sum_j \Phi(\mathbf{x}_i)\Phi(\mathbf{x}_j)\beta_i\beta_j \tag{56}$$

Using the definition of kernel we can rewrite the last equation in this form:

$$J_W = \sum_j k_{jj}\beta_j - \sum_i \sum_j k_{ij}\beta_i\beta_j \tag{57}$$

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:

$$d_j = \|\Phi(\mathbf{x}_j) - \mathbf{v}\|^2 = k_{jj} - 2\sum_r \beta_r k_{jr} + \sum_r \sum_s \beta_r\beta_s k_{rs} \tag{58}$$

The radius of the sphere is simply the distance between support vectors and the center $\mathbf{v}$. In fig. 3 it is possible to see the ability of this algorithm to find the smallest enclosing sphere without outliers.

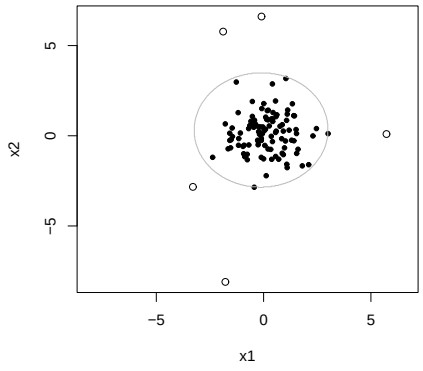


Figure 3: One class SVM applied to a data set with outliers. The grey line shows the smallest enclosing sphere of data without outliers.

3.5.1 Support Vector Clustering

Once boundaries in input space are found, a labelling procedure is necessary in order to complete clustering. In [38] the cluster assignment procedure follows a simple geometric idea. Any path connecting a pair of points belonging to different clusters must exit from the sphere in feature space.

Thus such paths will include a segment of points $\mathbf{y}$ such that $R(\mathbf{y}) > R$. Letting Y be the path connecting two points in feature space, it is possible to define an adjacency structure in this form:

$$\begin{cases} 1 & \text{if } R(\mathbf{y}) < R \quad \forall \mathbf{y} \in Y \\ 0 & \text{otherwise} \end{cases} \tag{59}$$

Clusters are simply the connected components of the graph with the adjacency matrix just defined. In the implementation in [37] the check is made sampling the line segment Y in 20 equidistant points. There are some modifications on this labelling algorithm (e.g., [50], [83]) that improve performances. An improved version of SVC algorithm with application in handwritten digits recognition can be found in [17].

3.5.2 Camastra and Verri algorithm

Another technique that combines K-Means and One Class SVM can be found in [14]. 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

$$\pi_i^\Phi(\rho) = \{\mathbf{x}_j \in \pi_i^\Phi \text{ and } \|\Phi(\mathbf{x}_j) - \mathbf{v}_i^\Phi\| < \rho\} \tag{60}$$

$\pi_i^\Phi(\rho)$ 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 [11] (e.g., crossvalidation). Resuming, 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 $V^\Phi = (\mathbf{v}_1^\Phi, \ldots, \mathbf{v}_k^\Phi)$ 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.

In fig. 4 it is possible to see the Voronoi regions in input space as obtained by this algorithm on a nonlinear separable data set. The same data set is not separable using two codevectors by the classical clustering algorithms, e.g., K-Means, SOM and Neural Gas.

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.

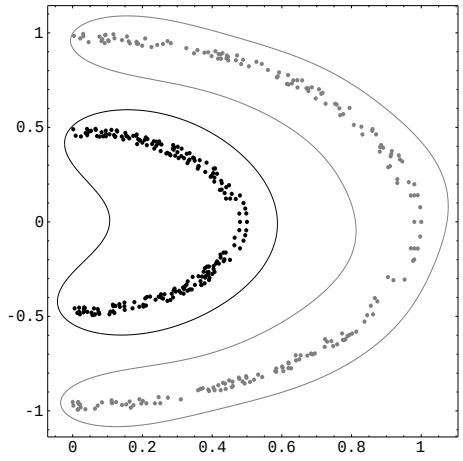


Figure 4: The Camastra and Verri algorithm applied to a nonlinear separable data set. The black and grey curves delimitate the two Voronoi sets produced by the algorithm.

3.6.1 Kernel Fuzzy c-Means with kernelization of the metric

The basic idea is to minimize the functional [81], [87], [88]:

$$J^{\Phi}(U,V) = \sum_{k=1}^{n}\sum_{i=1}^{c}(u_{ik})^m \left\|\Phi(\mathbf{x}_k) - \Phi(\mathbf{v}_i)\right\|^2 \tag{61}$$

with the probabilistic constraint over the memberships (eq. 14). The optimization procedure for the optimization of $J^{\Phi}(U,V)$ is again the Picard iteration technique. The minimization of the functional in eq. 61 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:

$$\frac{\partial K(\mathbf{x}_k,\mathbf{v}_i)}{\partial \mathbf{v}_i} = \frac{(\mathbf{x}_k - \mathbf{v}_i)}{\sigma^2}K(\mathbf{x}_k,\mathbf{v}_i) \tag{62}$$

We obtain for the memberships:

$$u_{ik}^{-1} = \sum_{j=1}^{c}\left(\frac{1-K(\mathbf{x}_k,\mathbf{v}_i)}{1-K(\mathbf{x}_k,\mathbf{v}_j)}\right)^{\frac{1}{m-1}} \tag{63}$$

and for the codevectors:

$$\mathbf{v}_i = \frac{\sum_{k=1}^{n}(u_{ik})^m K(x_k,v_i)\mathbf{x}_k}{\sum_{k=1}^{n}(u_{ik})^m K(x_k,v_i)} \tag{64}$$

3.6.2 Kernel Fuzzy c-Means in feature space

This method is the Fuzzy c-Means in feature space allowing to find a soft linear partitioning. This partitioning, back to the input space, results in a soft non linear partitioning of data. The functional

to optimize [33], [86] with the probabilistic constraint in eq. 14 is:

$$J^{\Phi}(U, V^{\Phi}) = \sum_{k=1}^{n} \sum_{i=1}^{c} (u_{ik})^m \left\| \Phi(\mathbf{x}_k) - \mathbf{v}_i^{\Phi} \right\|^2 \tag{65}$$

It is possible to rewrite the norm in eq. 65 explicitly by using:

$$\mathbf{v}_i^{\Phi} = \frac{\sum_{k=1}^{n} (u_{ik})^m \Phi(\mathbf{x}_k)}{\sum_{k=1}^{n} (u_{ik})^m} = a_i \sum_{k=1}^{n} (u_{ik})^m \Phi(\mathbf{x}_k) \tag{66}$$

which is the kernel version of eq. 17. For simplicity of notation we used:

$$a_i^{-1} = \sum_{r=1}^{n} (u_{ir})^m \tag{67}$$

Now it is possible to write the kernel version of eq. 16:

$$u_{ik}^{-1} = \sum_{j=1}^{c} \left[\frac{k_{kk} - 2a_i \sum_{r=1}^{n} (u_{ir})^m k_{kr} + a_i^2 \sum_{r=1}^{n} \sum_{s=1}^{n} (u_{ir})^m (u_{is})^m k_{rs}}{k_{kk} - 2a_j \sum_{r=1}^{n} (u_{jr})^m k_{kr} + a_j^2 \sum_{r=1}^{n} \sum_{s=1}^{n} (u_{jr})^m (u_{js})^m k_{rs}} \right]^{\frac{1}{m-1}} \tag{68}$$

Eq. 68 gives the rule for the update of the membership u_{ik}.

3.6.3 Possibilistic c-Means with the kernelization of the metric

The formulation of the Possibilistic c-Means with the kernelization of the metric used in [87] involves the minimization of the following functional:

$$J^{\Phi}(U, V) = \sum_{k=1}^{n} \sum_{i=1}^{c} (u_{ik})^m \left\| \Phi(\mathbf{x}_k) - \Phi(\mathbf{v}_i) \right\|^2 + \sum_{i=1}^{c} \eta_i \sum_{k=1}^{n} (1 - u_{ik})^m \tag{69}$$

Minimization leads to:

$$u_{ik}^{-1} = 1 + \left(\frac{\left\| \Phi(\mathbf{x}_k) - \Phi(\mathbf{v}_i) \right\|^2}{\eta_i} \right)^{\frac{1}{m-1}} \tag{70}$$

Considering again a Gaussian kernel, it can be rewritten as:

$$u_{ik}^{-1} = 1 + 2 \left(\frac{1 - K(\mathbf{x}_k, \mathbf{v}_i)}{\eta_i} \right)^{\frac{1}{m-1}} \tag{71}$$

and:

$$\mathbf{v}_i = \frac{\sum_{k=1}^{n} (u_{ik})^m K(x_k, v_i) \mathbf{x}_k}{\sum_{k=1}^{n} (u_{ik})^m K(x_k, v_i)} \tag{72}$$

Using the alternative formulation of the Possibilistic c-Means we can introduce this functional:

$$J^{\Phi}(U,V) = \sum_{k=1}^{n}\sum_{i=1}^{c} u_{ik}\|\Phi(\mathbf{x}_k) - \Phi(\mathbf{v}_i)\|^2 + \sum_{i=1}^{c}\eta_i \sum_{k=1}^{n}(u_{ik}\ln(u_{ik}) - u_{ik}) \tag{73}$$

Thus we have:

$$u_{ik} = \exp\left(\frac{\|\Phi(\mathbf{x}_k) - \Phi(\mathbf{v}_i)\|^2}{\eta_i}\right) \tag{74}$$

that can be rewritten as:

$$u_{ik} = \exp\left(\frac{2\left(1 - K(\mathbf{x}_k, \mathbf{v}_i)\right)}{\eta_i}\right) \tag{75}$$

For the update of the $\mathbf{v}_i$ we obtain:

$$\mathbf{v}_i = \frac{\sum_{k=1}^{n} u_{ik}K(x_k, v_i)\mathbf{x}_k}{\sum_{k=1}^{n} u_{ik}K(x_k, v_i)} \tag{76}$$

The computation of the η_i is straightforward.

4 Spectral Clustering

Spectral clustering methods [21] have a strong connection with graph theory [18], [25]. A comparison of some spectral clustering methods have been recently proposed in [75]. Let $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be the set of patterns to cluster. We can construct a *weighted undirected graph*[4] [18] G starting from X, where the adjacency[5] between two nodes is defined as:

$$a_{ij} = \begin{cases} h(\mathbf{x}_i, \mathbf{x}_j) & \text{if } i \neq j \\ 0 & \text{otherwise} \end{cases} \tag{77}$$

The function h measures the similarity between patterns. Typically a Gaussian function is used:

$$h(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{d(\mathbf{x}_i, \mathbf{x}_j)}{2\sigma^2}\right) \tag{78}$$

where d measures the dissimilarity between patterns and σ controls the rapidity of decay of h.

For a weighted graph G we set the weight matrix $W = A$ as the adjacency matrix of G. Then the degree matrix D as the matrix whose diagonal elements are the degrees of the nodes of G.

$$d_{ii} = \sum_{j=1}^{n} a_{ij} \tag{79}$$

In this framework the clustering problem can be seen as a graph cut problem 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

[4] A weighted undirected graph $G(V, E, W)$ has a set of nodes $V = \{v_1, \ldots, v_n\}$, a set of edges $E \subset V \times V$ connecting pairs of nodes and a $n \times n$ matrix W of weights in which each element w_{ij} gives the weight of the connection between nodes i and j. A undirected weighted graph has the property that $w_{ij} = w_{ji}$.

[5] Sometimes the term adjacency is used instead of *weight* or *affinity*.

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 [18]:

$$cut(S, \bar{S}) = \sum_{v_i \in S, v_j \in \bar{S}} a_{ij} \tag{80}$$

It is easy to verify that the minimization of this objective function favors the cut of isolated nodes. To achieve a better balance in the cardinality of S and $\bar{S}$ it is suggested to optimize the normalized cut[68]:

$$Ncut(S, \bar{S}) = cut(S, \bar{S}) \left(\frac{1}{assoc(S, V)} + \frac{1}{assoc(\bar{S}, V)} \right) \tag{81}$$

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

$$assoc(S, V) = \sum_{v_i \in S, v_j j \in V} a_{ij} \equiv vol(S) = \sum_{v_i \in S} d_{ii} \tag{82}$$

There are other definitions of functions to optimize (e.g., the conductance [41], the normalized association [68], ratio cut [22]).

The complexity in optimizing these kind of objective is very high 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 [18]:

$$L = D - A \tag{83}$$

which can be seen as a linear operator on G (see appendix A). 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 and the corresponding eigenvector can cluster together similar patterns (see appendix A).

4.1 Shi and Malik algorithm

This algorithm has been proposed by Shi et al. [68]. Their aim was to apply 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:

$$a_{ij} = \exp\left(-\frac{\|\mathbf{f}_i - \mathbf{f}_j\|^2}{2\sigma_1^2} \right) \cdot \begin{cases} \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_2^2} \right) & \text{if } \|\mathbf{x}_i - \mathbf{x}_j\| < R \\ 0 & \text{otherwise} \end{cases} \tag{84}$$

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}. Resuming, 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. 84
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 $\mathbf{e}_2$ to segment G

In the ideal case of two non connected subgraphs, $\mathbf{e}_2$ assumes just two values; this allows to cluster together the the components of $\mathbf{e}_2$ with the same value. In a real case the splitting point must be chosen to cluster the components of $\mathbf{e}_2$ 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 [84].

4.2 Ng, Jordan and Weiss algorithm

The algorithm that has been proposed by Ng et al. [59] and uses the adjacency matrix A as Laplacian. This new definition allows to consider the eigenvector associated with the largest eigenvalues as the "good" one for clustering. The idea is the same of the 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 $A \in \mathbb{R}^{n \times n}$:

$$w_{ij} = \begin{cases} \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right) & \text{if } i \neq j \\ 0 & \text{otherwise} \end{cases} \tag{85}$$

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

$$L = D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \tag{86}$$

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:

$$y_{ij} = \frac{z_{ij}}{\sum_{r=1}^{k} z_{ir}^2} \tag{87}$$

   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 criteria 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 as in fig. 5. Here we show the steps

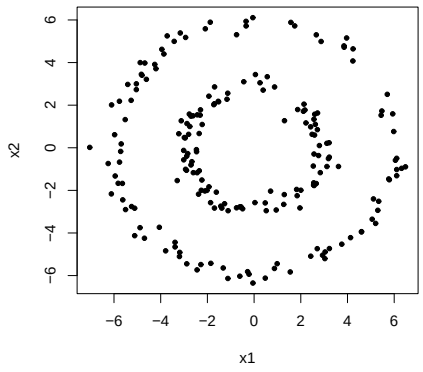


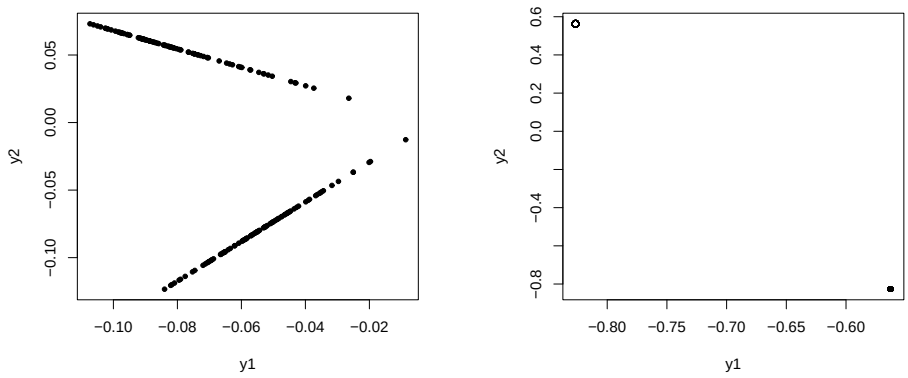
Figure 5: A data set composed of two rings of points.

of the algorithm when applied to the data set in fig. 5. Once the singular value decomposition of L is computed[6], we can see the matrices Z and Y in fig. 6. Once Y is computed, it is easy to cluster the two groups of points. The result is shown in fig. 7.

4.3 Other Methods

An interesting view of spectral clustering is provided by Meilă et al. [56] which 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 $P = D^{-1}A$, the 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 the a function of the features producing correct segmentation given a segmented image.

[6]For the adjacency matrix $\sigma = 0.4$ has been used.



(a) (b)

Figure 6: (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.

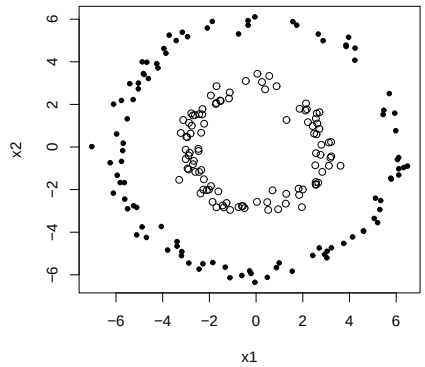


Figure 7: The result of the Ng and Jordan algorithm on the ring data set.

An interesting study on spectral clustering is by Kannan et al. [41] where they exploit the objective function with respect to some artificial data sets. They show that there is no objective function able to properly cluster every data set, i.e., there always exists some data set for which a particular objective function perform badly. For this reason they propose a bi-criteria objective function. These two objectives are respectively a conductance greater than a threshold α and the association between a subset of nodes S and itself is at most a fraction ϵ of the volume of S. 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 [7], [8]. The hint for this unifying theory is in 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 and more recently a unifying view of kernel K-means and spectral clustering methods [22], [24]. In this section we show explicitly the equivalence between them highlighting that these two approaches have the same foundation. In particular both can be viewed as a trace maximization problem.

5.1 Kernel Clustering Methods as a trace maximization problem

To show the direct equivalence between kernel and spectral clustering methods we introduce the weighted version of the kernel K-means [24]. 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:

$$J^{\Phi}(W, V^{\Phi}) = \sum_{i=1}^{c} \sum_{\mathbf{x}_k \in \pi_i} w_k \left\| \Phi(\mathbf{x}_k) - \mathbf{v}_i^{\Phi} \right\|^2 \tag{88}$$

where:

$$\mathbf{v}_i^{\Phi} = \frac{\sum_{\mathbf{x}_k \in \pi_i} w_k \Phi(\mathbf{x}_k)}{\sum_{\mathbf{x}_k \in \pi_i} w_k} = \frac{\sum_{\mathbf{x}_k \in \pi_i} w_k \Phi(\mathbf{x}_k)}{s_i} \tag{89}$$

where we have introduced:

$$s_i = \sum_{\mathbf{x}_k \in \pi_i} w_k \tag{90}$$

Now let's define the matrix Z having:

$$z_{ki} = \begin{cases} s_i^{-1/2} & \text{if } \mathbf{x}_k \in \pi_i \\ 0 & \text{otherwise} \end{cases} \tag{91}$$

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

$$s_i^{-1} = (Z^T Z)_{ii} \tag{92}$$

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. 89 in eq. 88 we obtain the following matrix expression for $J^\Phi(W, V^\Phi)$:

$$J^\Phi(W, V^\Phi) = \sum_{k=1}^{n} w_k \left\| F_{\cdot k} - (FWZZ^T)_{\cdot k} \right\|^2 \tag{93}$$

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:

$$\begin{aligned} J^\Phi(W, V^\Phi) &= \sum_{k=1}^{n} w_k \left\| F_{\cdot k} - (FW^{1/2}YY^TW^{-1/2})_{\cdot k} \right\|^2 = \\ &= \sum_{k=1}^{n} \left\| F_{\cdot k} w_k^{1/2} - (FW^{1/2}YY^T)_{\cdot k} \right\|^2 = \\ &= \left\| FW^{1/2} - FW^{1/2}YY^T \right\|_{\mathrm{F}}^2 \end{aligned} \tag{94}$$

The norm $\|\|_{\mathrm{F}}$ is the Frobenius norm [32]. Since $\|A\|_{\mathrm{F}} = \mathrm{tr}(AA^T)$ and $(AB)^T = B^TA^T$ we rewrite the last equation:

$$\begin{aligned} J^\Phi(W, V^\Phi) &= \mathrm{tr}(W^{1/2}F^TFW^{1/2} - W^{1/2}F^TFW^{1/2}YY^T + \\ &- YY^TW^{1/2}F^TFW^{1/2} + YY^TW^{1/2}F^TFW^{1/2}YY^T) \end{aligned} \tag{95}$$

Now we can further manipulate the last expression using some properties of the trace. In particular we use the linearity of the trace operation and the fact that $\mathrm{tr}(AB) = \mathrm{tr}(BA)$. Applying the linearity we can decompose the trace of a sum in a sum of traces. Then applying the second property to the last three terms and the orthonormality of Y we obtain:

$$\mathrm{tr}(W^{1/2}F^TFW^{1/2}YY^T) = \mathrm{tr}(Y^TW^{1/2}F^TFW^{1/2}Y) \tag{96}$$

$$\mathrm{tr}(YY^TW^{1/2}F^TFW^{1/2}) = \mathrm{tr}(Y^TW^{1/2}F^TFW^{1/2}Y) \tag{97}$$

$$\begin{aligned} \mathrm{tr}(YY^TW^{1/2}F^TFW^{1/2}YY^T) &= \mathrm{tr}(Y^TW^{1/2}F^TFW^{1/2}YY^TY) = \\ &= \mathrm{tr}(Y^TW^{1/2}F^TFW^{1/2}Y) \end{aligned} \tag{98}$$

Thus all these terms can be summed and the objective function become:

$$J^\Phi(W, V^\Phi) = \mathrm{tr}(W^{1/2}F^TFW^{1/2}) - \mathrm{tr}(Y^TW^{1/2}F^TFW^{1/2}Y) \tag{99}$$

Since the first term of eq. 99 is constant we can formulate the weighted kernel K-means as the problem of maximizing over Y the following functional:

$$J^\Phi(W, V^\Phi) = \mathrm{tr}(Y^TW^{1/2}F^TFW^{1/2}Y) \tag{100}$$

5.2 Spectral Clustering Methods as a trace maximization problem

The objective function to optimize in graph cut problems could be several. Recalling that the definition of association between two sets of edges S and T of a weighted graph is the following:

$$assoc(S,T) = \sum_{i \in S, j \in T} a_{ij} \tag{101}$$

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

$$J(S_1, \ldots, S_c) = \sum_{i=1}^{c} \frac{assoc(S_i, S_i)}{|S_i|} \tag{102}$$

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:

$$J(S_1, \ldots, S_c) = \sum_{i=1}^{c} \frac{\mathbf{z}_i^T A \mathbf{z}_i}{\mathbf{z}_i^T \mathbf{z}_i} \tag{103}$$

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

$$\mathbf{y}_i = \frac{\mathbf{z}_i}{(\mathbf{z}_i^T \mathbf{z}_i)^{1/2}} \tag{104}$$

we obtain:

$$J(S_1, \ldots, S_c) = \sum_{i=1}^{c} \mathbf{y}_i^T A \mathbf{y}_i \tag{105}$$

The last equation can be rewritten in matrix form:

$$J(S_1, \ldots, S_c) = \mathrm{tr}(Y^T A Y) \tag{106}$$

5.3 A unified view of the two approaches

Comparing eq. 106 and eq. 100 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 [15], the normalized cut and the Kernighan-Lin [42] 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:

$$J(S_1, \ldots, S_c) = \mathrm{tr}(Y^T D^{-1/2} A D^{-1/2} Y) \tag{107}$$

Thus the correspondence with the objective in the kernel K-means would impose to choose $Y = D^{1/2}Z$, $W = D$ and $K = D^{-1}AD^{-1}$. Since the mathematical foundation of these methods is the same, it is possible to choose what algorithm to use for clustering choosing the approach which has 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 spectral and kernel approach. Unlike classical 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 spectral and kernel clustering methods.

A The graph cut

In this appendix we motivate the use of the spectral decomposition of the Laplacian for the clustering problem. We describe the connection between the $cut(S, \bar{S})$ of a graph and the second eigenvalue of L [18].

The Laplacian can be seen as a discrete version of the Laplacian operator $\triangle$ which plays a key role in mathematical physics. The Laplacian L can be seen as an operator on the space of the functions $g : V \to \mathbb{R}$ which satisfies[7] [18]:

$$L(g(v_i)) = \sum_{v_j \sim v_i} (g(v_i) - g(v_j))a_{ij} \tag{108}$$

Since L is a linear operator it is possible to use the concepts of the linear algebra to infer important characteristics of G.

There are some basic properties of the eigenvalues and eigenvectors of L [57]. The eigenvalues $\{\lambda_1, \ldots, \lambda_n\}$ of L are real and non negative since L is symmetric. Recalling that the Rayleight quotient [32] is defined as:

$$\mathcal{R} = \frac{(g, L(g))}{(g, g)} \tag{109}$$

it is easy to see that L is semidefinite positive since the Rayleight quotient is greater than zero.

$$\frac{(g, L(g))}{(g, g)} = \frac{\sum_{v_j \sim v_i} (g(v_i) - g(v_j))^2 a_{ij}}{\sum_x g^2(x)} \geq 0 \tag{110}$$

The smallest eigenvalue of L is $\lambda_1 = 0$ with an associate eigenvector proportional to $(1, 1, \ldots, 1)$ (it follows from the definition of L). Another interesting property associated with Laplacian is that the multiplicity of the eigenvalue equal to zero is exactly the number of connected components of the graph. This can be seen easily applying simple algebraic concepts [18], [2] noting that L assumes a block diagonal structure.

[7] The sum where $v_j \sim v_i$ means that we have to sum where v_j and v_i are connected.

Recalling that the cut of G in S and $\bar{S}$ is defined as:

$$cut(S, \bar{S}) = \sum_{v_i \in S, v_j \in \bar{S}} a_{ij} \tag{111}$$

The cut measures the loss in removing a set of edges when disconnecting a set of nodes S from its complement. A small value of cut indicates that S and its complement are weakly connected revealing a strong biclustered structure.

The Cheeger constant is defined as:

$$h_g = \min_S \frac{cut(S, \bar{S})}{\min(vol(S), vol(\bar{S}))} \tag{112}$$

The second smallest eigenvalue λ_2 of L is related to the Cheeger constant by the following relation:

$$2h_g \geq \lambda_2 \geq h_g^2/2 \tag{113}$$

Thus λ_2 of L [27] gives useful informations on how the graph is clustered and on what is the best that we can achieve in cutting a graph.

Applying simple concepts of algebra it is possible to show that the second smallest eigenvalue λ_2 and its corresponding eigenvector $\mathbf{e}_2$ are related to the Rayleight quotient [32] by the following:

$$\lambda_2 = \inf_{g \perp (1,1,\ldots,1)} \frac{(g, L(g))}{(g, g)} \tag{114}$$

$$\mathbf{e}_2 = \arg \min_{g \perp (1,1,\ldots,1)} \frac{(g, L(g))}{(g, g)} \tag{115}$$

This approach can be useful when only λ_2 is needed. In fact, solving the complete eigenproblem could be absolutely time consuming, especially for large databases (as in image segmentation problems). In cases where the eigenvectors and eigenvalues of a sparse matrix have to be computed, the Lanczos method [61] can be used speeding up the computing process.

In this kind of applications, where the number of nodes (pixels) can be huge, the computing of the second smallest eigenvalue and its eigenvector can be done using eq. 114 and eq. 115. This approach can take the advantage of the sparsity of the Laplacian matrix. One efficient algorithm in this situation is the Lanczos method [32] which works quite well even for thousands of nodes.

To motivate the fact that the spectral decomposition of the Laplacian allows to cluster the data set, let's consider the minimization of the normalized cut:

$$Ncut(S, \bar{S}) = cut(S, \bar{S}) \cdot \left(\frac{1}{vol(S)} + \frac{1}{vol(\bar{S})} \right) \tag{116}$$

By manipulating this equation it can be proven that this optimization problem is equivalent to the minimization of the Rayleight quotient $\mathcal{R}$. Since the lowest eigenvalue is zero with a constant eigenvector, the last is exactly the computation of λ_2.

In another work it has been shown that when few eigenvectors are used to represent the patterns, the similarities and dissimilarities between them are highlighted [12]. This result shows that the cosines between similar patterns increases while decreases for dissimilar patterns allowing to cluster the resulting representation quite comfortably.

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] T. M. Apostol. *Calculus, 2 vols.* Wiley, 2 edition, 1967.

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

[4] Majid A. and Awan. An intelligent system based on kernel methods for crop yield prediction. In *PAKDD*, pages 841–846, 2006.

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

[6] Marco Barreno. Spectral methods for image clustering, 2004.

[7] 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.

[8] Yoshua Bengio, Pascal Vincent, and Jean F. Paiement. Spectral clustering and kernel pca are learning eigenfunctions. CIRANO Working Papers 2003s-19, CIRANO, May 2003.

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

[10] C. M. Bishop. *Neural networks for pattern recognition.* Oxford University Press, Oxford, UK, 1996.

[11] Christopher M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, November 1995.

[12] 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.

[13] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.*, 2(2):121–167, 1998.

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

[15] Pak K. Chan, Martine Schlag, and Jason 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.

[16] Songcan Chen and Daoqiang 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.

[17] Jung-Hsien Chiang and Pei-Yi Hao. A new kernel-based fuzzy clustering approach: support vector clustering with cell growing. *IEEE Transactions on Fuzzy Systems*, 11(4):518–527, 2003.

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

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

[20] Nello Cristianini, John S. Taylor, André Elisseeff, and Jaz S. Kandola. On kernel-target alignment. In *NIPS*, pages 367–373, 2001.

[21] Nello Cristianini, John S. Taylor, and Jaz S. Kandola. Spectral kernel methods for clustering. In *NIPS*, pages 649–655, 2001.

[22] Inderjit Dhillon, Yuqiang Guan, and Brian Kulis. A unified view of kernel k-means, spectral clustering and graph partitioning. Technical Report Technical Report TR-04-25, UTCS, 2005.

[23] Inderjit 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.

[24] Inderjit S. Dhillon, Yuqiang Guan, and Brian 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.

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

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

[27] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(98), 1973.

[28] Igor Fischer and Ian Poland. New methods for spectral clustering. Technical Report IDSIA-12-04, IDSIA, 2004.

[29] Ronald A. Fisher. The use of multiple measurements in taxonomic problems. *Annals Eugen.*, 7:179–188, 1936.

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

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

[32] Gene H. Golub and Charles F. Van Loan. *Matrix Computations (Johns Hopkins Studies in Mathematical Sciences)*. The Johns Hopkins University Press, October 1996.

[33] 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.

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

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

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

[37] Asa B. Hur, David Horn, Hava T. Siegelmann, and Vladimir Vapnik. A support vector method for clustering. In *NIPS*, pages 367–373, 2000.

[38] Asa B. Hur, David Horn, Hava T. Siegelmann, and Vladimir Vapnik. Support vector clustering. *Journal of Machine Learning Research*, 2:125–137, 2001.

[39] R. Inokuchi and S. Miyamoto. LVQ clustering and som using a kernel function. volume 3, pages 1497–1500 vol.3, 2004.

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

[41] Ravi Kannan, Santosh Vempala, and Adrian 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.

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

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

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

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

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

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

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

[49] Brian Kulis, Sugato Basu, Inderjit Dhillon, and Raymond 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.

[50] Daewon Lee. An improved cluster labeling method for support vector clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(3):461–464, 2005.

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

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

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

[57] Bojan Mohar. Laplace eigenvalues of graphs: a survey. *Discrete Math.*, 109(1-3):171–183, 1992.

[58] O. Nasraoui and R. Krishnapuram. An improved possibilistic c-means algorithm with finite rejection and robust scale estimation. Berkeley, California, June 1996.

[59] 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.

[60] 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.

[61] Alex Pothen, Horst D. Simon, and Kan-Pu Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM J. Matrix Anal. Appl.*, 11(3):430–452, July 1990.

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

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

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

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

[66] 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.

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

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

[69] 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.

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

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

[72] Xiaoyang Tan, Songcan Chen, Zhi H. Zhou, and Fuyan Zhang. Robust face recognition from a single training image per person with kernel-based som-face. In *ISNN (1)*, pages 858–863, 2004.

[73] David M. J. Tax and Robert P. W. Duin. Support vector domain description. *Pattern Recogn. Lett.*, 20(11-13):1191–1199, 1999.

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

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

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

[77] 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.

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

[79] 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.

[80] 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.

[81] Zhong D. Wu, Wei X. Xie, and Jian P. Yu. Fuzzy c-means clustering algorithm based on kernel method. *ICCIMA*, 00, 2003.

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

[83] Jianhua Yang, , and S. K. Chalup. Support vector clustering through proximity graph modelling. volume 2, pages 898–903 vol.2, 2002.

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

[85] Hongyuan Zha, Xiaofeng He, Chris H. Q. Ding, Ming Gu, and Horst D. Simon. Spectral relaxation for k-means clustering. In *NIPS*, pages 1057–1064, 2001.

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

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

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

[89] Dao-Qiang Zhang, Song-Can Chen, Zhi-Song Pan, and Ke-Ren Tan. Kernel-based fuzzy clustering incorporating spatial constraints for image segmentation. volume 4, pages 2189–2192 Vol.4, 2003.