KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering

Clustering evolving data using kernel-based methods

Rocco Langone

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Engineering

July 2014

Clustering evolving data using kernel-based methods

Rocco LANGONE

Supervisory Committee:
Em. prof. dr. ir. Paul Van Houtte, chair
Prof. dr. ir. Johan A. K. Suykens, promotor
Em. prof. dr. ir. Joos Vandewalle
Prof. dr. ir. Marc Van Barel
Dr. ir. Bart De Ketelaere
Prof. dr. Renaud Lambiotte
(Université de Namur)

Dissertation presented in partial
fulfillment of the requirements for
the degree of Doctor
in Engineering

July 2014

© 2013 KU Leuven – Faculty of Engineering
Uitgegeven in eigen beheer, Rocco Langone, Kasteelpark Arenberg 10, B-3001 Heverlee (Belgium)

Alle rechten voorbehouden. Niets uit deze uitgave mag worden vermenigvuldigd en/of openbaar gemaakt worden door middel van druk, fotokopie, microfilm, elektronisch of op welke andere wijze ook zonder voorafgaande schriftelijke toestemming van de uitgever.

All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm, electronic or any other means without written permission from the publisher.

ISBN 978-94-6018-844-2

D/2014/7515/68

A mio papá, che mi ha insegnato l'importanza della memoria storica per capire il presente ed immaginare il futuro. A mia mamma, la cui disarmante semplicitá mi ricorda che i modelli piú semplici ed eleganti vanno preferiti a quelli contorti e complicati. Alle mie sorelle Luisa e Laura, studiose dell' "intelligenza naturale". A mio fratello Rosario, "ossessionato" dai rankings.

Preface

The work presented in this thesis is related to the research carried out during my doctoral studies at the STADIUS (ex SISTA) research group, inspired by the magnificent view of the Arenberg castle. It has been a priceless time full of enriching experiences both at a professional and a personal level.
First of all, I wish to thank my promotor Johan Suykens, who about 4 years ago believed in my abilities and gave me the opportunity to start constructing my future. During the Phd, he gave me continuous support and was always prodigal of suggestions. Many thanks also to the jury members of this thesis, who accepted to review the dissertation and provided valuable comments. I would like to acknowledge my actual and former colleagues and friends at ESAT for their helpfulness regarding work issues, the free time spent together in Leuven, the spare time enjoyed after national and international conferences. Many thanks to the Erasmus students, visiting doctoral students and all the friends I met during these years. I will always remember the nice moments shared at the occasion of lunches, dinners, trips, parties etc. Many acknowledgements to my family for cheering me up in difficult times and sharing my joy in happy moments. Finally, my most special thanks go to my girlfriend Bruna, who around 2 years ago swept away the shadow surrounding my heart with love, sweetness, and empathy...

Abstract

Thanks to recent developments of Information Technologies, there is a profusion of available data in a wide range of application domains ranging from science and engineering to biology and business. For this reason, the demand for real-time data processing, mining and analysis is experiencing an explosive growth in recent years. Since labels are usually not available and in general a full understanding of the data is missing, clustering plays a major role in shedding an initial light. In this context, elements such as generalization to out-of-sample data, model selection criteria, consistency of the clustering results over time and scalability to large data become key issues. A successful modelling framework is offered by Least Squares Support Vector Machine (LS-SVM), which is designed in a primal-dual optimization setting. The latter allows extensions of the core models by adding additional constraints to the primal problem, by changing the objective function or by introducing new model selection criteria. In this thesis, we propose several modelling strategies to tackle evolving data in different contexts. In the framework of static clustering, we start by introducing a soft kernel spectral clustering (SKSC) algorithm, which can better deal with overlapping clusters with respect to kernel spectral clustering (KSC) and provides more interpretable outcomes. Afterwards, a whole strategy based upon KSC for community detection of static networks is proposed, where the extraction of a high quality training sub-graph, the choice of the kernel function, the model selection and the applicability to large-scale data are key aspects. This paves the way for the development of a novel clustering algorithm for the analysis of evolving networks called kernel spectral clustering with memory effect (MKSC), where the temporal smoothness between clustering results in successive time steps is incorporated at the level of the primal optimization problem, by properly modifying the KSC formulation. Later on, an application of KSC to fault detection of an industrial machine is presented. Here, a smart pre-processing of the data by means of a proper windowing operation is necessary to catch the ongoing degradation process affecting the machine. In this way, in a genuinely unsupervised manner, it is possible to raise an early warning when necessary, in an online fashion. Finally, we propose a new algorithm called incremental kernel spectral clustering (IKSC) for online learning

of non-stationary data. This ambitious challenge is faced by taking advantage of the out-of-sample property of kernel spectral clustering (KSC) to adapt the initial model, in order to tackle merging, splitting or drifting of clusters across time. Real-world applications considered in this thesis include image segmentation, time-series clustering, community detection of static and evolving networks.

Abbreviations

AMS	Average Membership Strength
ARI	Adjusted Rand Index
BLF	Balanced Line Fit
BLF_{Mem}	Smoothed Balanced Line Fit
Cond	Conductance
$Cond_{Mem}$	Smoothed Conductance
EF	Expansion Factor
ESC	Evolutionary Spectral Clustering
IKSC	Incremental Kernel Spectral Clustering
IKM	Incremental K-means
KKT	Karush-Kuhn-Tucker
KM	K-means
KSC	Kernel Spectral Clustering
LFR	Benchmark graphs for testing community detection algorithms
LOUV	Louvain Method
LS-SVM	Least Squares Support Vector Machine
MKSC	Kernel Spectral Clustering with Memory effect
Mod	Modularity
Mod_{Mem}	Smoothed Modularity
NMI	Normalized Mutual Information
PCA	Principal Component Analysis
RBF	Radial Basis Function
SKSC	Soft Kernel Spectral Clustering
SC	Spectral Clustering
SVM	Support Vector Machine

Notation

x^T	Transpose of a vector x
Ω^T	Transpose of a matrix Ω
Ω_{ij}	ij-th entry of the matrix Ω
I_N	$N \times N$ Identity matrix
1_N	$N \times 1$ Vector of ones
$\mathcal{D}_{\text{Tr}} = \{x_i\}_{i=1}^{N_{\text{Tr}}}$	Training sample of N_{Tr} data points
$\varphi(\cdot)$	Feature map
$\mathcal{F}$	Feature space of dimension d_h
$K(x_i, x_j)$	Kernel function evaluated on data points x_i, x_j
$\{\mathcal{A}_p\}_{p=1}^{k}$	Partition composed of k clusters
$\alpha_i^{(l)} \in \mathbb{R}$	i-th entry of the dual solution vector $\alpha^{(l)} \in \mathbb{R}^{N_{\text{Tr}}}$
D	$N \times N$ graph degree matrix
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	Set of N vertices $\mathcal{V} = \{v_i\}_{i=1}^{N}$ and m edges $\mathcal{E}$ of a graph
$\mathcal{S} = \{(\mathcal{V}_t, \mathcal{E}_t)\}_{t=1}^{T}$	Sequence of networks over time T
$\lvert \cdot \rvert$	Cardinality of a set

Contents

Abstract v

Abbreviations vii

Notation ix

Contents xi

1 Introduction 1

1.1 Background 1

1.2 Challenges 2

1.3 Objectives 3

1.4 Chapter by Chapter Overview 4

1.5 Main Contributions 6

2 Spectral clustering 9

2.1 Classical Spectral Clustering 10

2.1.1 Introduction 10

2.1.2 The Graph Partitioning Problem 10

2.1.3 Link with Markov Chains 12

2.1.4 Basic Algorithm 12

2.2 Kernel Spectral Clustering 13
2.2.1 Generalities 13
2.2.2 Least Squares Support Vector Machine 13
2.2.3 Primal-Dual Formulation 15
2.2.4 Model Selection 16
2.2.5 Generalization 18
2.3 Soft Kernel Spectral Clustering 18
2.3.1 Overview 18
2.3.2 Algorithm 20
2.3.3 Model Selection 21
2.3.4 Toy Examples 23
2.3.5 Application: Image Segmentation 29
2.4 Conclusions 30

3 Community Detection in Complex Networks 33

3.1 Related work 34
3.2 Methods 36
3.2.1 Representative Sub-graph Extraction 36
3.2.2 Model Selection Criteria 38
3.2.3 Choice of the Kernel Function 39
3.2.4 Computational Complexity 40
3.3 Simulations on Synthetic Networks 41
3.4 Real-World Applications 48
3.5 Conclusions 58

4 Clustering Evolving Networks 61

4.1 Literature Review 62
4.2 The MKSC Model 63

4.2.1 Cluster Quality Measures in a Dynamic Scenario 66
4.2.2 Computational Complexity 67
4.3 Framework 1 67
4.3.1 Artificial Examples 67
4.3.2 Real-Life Application 76
4.4 Framework 2 82
4.4.1 Object appearing and leaving over time 84
4.4.2 Tracking the clusters 84
4.4.3 Description of the data sets 87
4.4.4 Experiments 87
4.4.5 Visualizing the clusters evolution 94
4.5 Flexibility of MKSC 97
4.6 Conclusions 98

5 Predicting Maintenance of Industrial Machines 101
5.1 Problem Description 101
5.2 Materials and Methods 102
5.3 Results 103
5.3.1 Hard Clustering 104
5.3.2 Probabilistic Output 106
5.4 Comparison with K-means 108
5.5 Conclusions 113

6 Clustering Non-Stationary Data 115
6.1 General Overview 115
6.2 Incremental Kernel Spectral Clustering 116
6.2.1 Algorithm 117
6.2.2 Computational Complexity 119

6.3 Synthetic Experiments 120
6.3.1 Description of the data 120
6.3.2 Simulation results 122
6.3.3 Analysis of the eigenvectors 129
6.4 Real-Life Example 130
6.4.1 The PM_{10} data-set 130
6.4.2 Results of the simulations 132
6.5 Incremental K-means Clustering 134
6.6 Conclusions 134

7 Conclusions and Future Challenges **139**
7.1 General Conclusions 139
7.2 Perspectives 141

A Appendix **143**
A.1 Cluster Quality Evaluation 143
A.1.1 Data Clustering 143
A.1.2 Community Detection 144
A.1.3 Comparing Partitions 146

Bibliography **147**

Chapter 1

Introduction

1.1 Background

We live in the Information Age. The recent development of Information Technologies (computers, internet, smart phones, sensors etc.) has a big impact on science and society. In principle, the large amount of available data can help to grasp the complexity of many phenomena of interest, in order to make new scientific discoveries, designing optimal business strategies, optimizing industrial processes etc.

Recognition of complex patterns in the data is of crucial importance to extract useful knowledge. In this context, clustering is a fundamental mode of understanding and learning [60, 59]. It refers to the task of organizing the data into meaningful groupings based only on the similarity between the data elements, and therefore is exploratory in its essence. Since no target or desired patterns are known a priori, it belongs to the family of unsupervised learning techniques [18].

Unveiling the underlying structure of the data through the cluster analysis is just one side of the coin. Other important elements are related to the dynamic version of the problem, i.e. monitoring the evolution of the clusters. Understanding how the behaviour of the system under study changes in time represents a key issue in many domains [103, 50]. From this point of view dynamic clustering would be a useful tool to investigate how clusters form, evolve and disappear.

The topic of this thesis is related to the design and application of kernel-based methods to perform dynamic clustering. Kernel methods are a class of machine learning techniques where two main modelling phases are present. First a mapping of the data into a high dimensional feature space is performed. Then, the design of learning

algorithms in that space allows to discover complex and non-linear relations in the original input space [122]. A major role in this work is played by Least Squares Support Vector Machine (LS-SVM) [128], which is a class of Support Vector Machine (SVM) [32] based on a constrained optimization framework with the presence of the L_2 loss function in the objective and equality instead of inequality constraints. By modifying and extending the objective and/or the constraints of the core formulation, it is possible to develop models tailored for a given application, with a systematic model selection procedure and high generalization abilities.

1.2 Challenges

The main issues tackled in this thesis can be summarized as follows:

- **Community detection via kernel methods**: A network is a collection of nodes or vertices joined by edges and represents the patterns of connections between the components of complex systems [106]. Usually real-life networks display a high level of order and organization. For example, the distribution of edges is characterized by high concentrations of edges within special groups of vertices, and low concentrations between these groups. The problem of identifying such clusters of nodes is called community detection [42]. The main challenges posed by the usage of kernel methods for community detection are related to the choice of the kernel function, the model selection, the out-of-sample extension, the scalability to large datasets.
- **Analysis of dynamic communities**: Community detection of evolving networks aims to understand how the community structure of a complex network changes over time [20, 103]. A desirable feature of a clustering model which has to capture the evolution of communities is the temporal smoothness between clusters in successive time-steps. Providing a consistent clustering at each time results in a smooth view of the changes and a greater robustness against noise [27, 25].
- **Fault detection**: With the development of information and sensor technology many process variables in a power plant like temperature, pressure etc. can be monitored. These measurements give an information on the current status of a machine and can be used to predict the faults and plan an optimal maintenance strategy [28, 65]. A useful model in this case must be able to catch, in an online fashion, the degradation process affecting the machine, to avoid future failures of the components and unplanned downtimes.
- **Clustering in a non-stationary environment**: In many real-life applications non-stationary data are generated according to some distribution models which

change over time. Therefore, a proper cluster analysis can be useful to detect important change points and in general to better understand the dynamics of the system under investigation. In this case a clustering algorithm is required to continuously adapt in response to new data and to be computationally efficient for real-time applications [21].

1.3 Objectives

In this thesis the following objectives can be outlined:

- **to envisage a whole kernel-based framework for community detection**. In this context many issues arise. First of all it is important to choose a proper kernel function to describe the similarity between the nodes of the network under investigation. Then a key point is represented by the model selection, i.e. finding the natural number of communities which are present in the network and eventually tuning the kernel hyper-parameters. The kernel-based model must also be able to accurately predict the membership of new nodes joining the network, without performing the clustering from scratch. Moreover, since many real-world networks contain millions of nodes and edges, the network data have to be processed in a reasonable time. Finally, the research carried out for solving the static community detection problem paves the way for the development of models for the analysis of evolving networks.

- **to design a model for community detection in a changing scenario**. An evolving network can be described as a sequence of snapshot graphs, where each snapshot represents the configuration of the network at a particular time instant. When community detection is performed at time t, the clustering should be similar to the clustering at the previous time-step $t-1$, and should accurately incorporate the actual data. In this way, if the data at time t does not deviate from historical expectations, the clustering should be similar to that from time $t-1$, while if the structure of the data changes significantly, the clustering must be modified to account for the new structure. Thus, a good clustering algorithm must trade-off the benefit of maintaining a consistent clustering over time with the cost of deviating from an accurate representation of the current data.

- **to conceive modelling strategies for clustering stationary and non-stationary data streams**. Data streams are a sequence of data records stamped and ordered by time [50]. Clustering data streams in real time is an ambitious problem with ample applications. If the data distribution is stationary, emphasis should be given to the off-line construction of the model. Once properly designed, such a model can be used to cluster the data stream in an online fashion by means of the

out-of-sample extension. However, if the data distribution is non-stationary, the initial model soon becomes obsolete and must be quickly updated. Therefore the development of a fast, adaptive and accurate model is an important objective.

1.4 Chapter by Chapter Overview

The general structure of this thesis is sketched in figure 1.1 and can be described as follows:

- **Chapter** 2 contains four sections. First of all a general introduction to spectral clustering, one of the most successful clustering algorithms, is given. Then kernel spectral clustering (KSC) is reviewed. KSC is a spectral clustering algorithm formulated in the LS-SVM optimization framework, with the possibility to extend the clustering model to out-of-sample data for predictive purposes. Moreover a model selection criterion called Balanced Line Fit (BLF) is also present. For these reasons it represents the starting point to face the challenges described in section 1.2. Later on the soft kernel spectral clustering (SKSC) algorithm is introduced. Instead of using the hard assignment rule present in KSC, a fuzzy assignment based on the cosine distance from the cluster prototypes in the projections space is suggested. We also introduce a related model selection technique, called Average Membership Strength criterion (AMS), which solves the major drawbacks of BLF. Finally, we show that SKSC can improve the interpretability of the results and the clustering performance with respect to KSC, mainly in cases of large overlap between the clusters.
- **Chapter** 3 is dedicated to the community detection problem. After reviewing the important literature in the field we illustrate our methodology, which is composed by four main cornerstones. First, it is crucial to extract from the given network a small sub-graph representative of its community structure, which is a challenging problem. This sub-graph can then be used to train a KSC model in a computationally efficient way and forms the basis for a good generalization. Second, the correct tuning of the kernel hyper-parameters (if any) and the number of communities is another important issue, which is solved by proposing a new model selection criterion based on the Modularity statistics. Third, the kernel functions used to properly describe the similarity between the nodes are presented. Finally, the out-of-sample extension allows not only to accurately predict the community affiliation of new nodes, but also can make the algorithm cluster millions of data in a short time on a desktop computer.
- **Chapter** 4 introduces a novel model called kernel spectral clustering with memory effect (MKSC). This method is designed to cluster evolving networks

(described as a sequence of snapshot graphs), aiming to track the long-term drift of the communities by ignoring meanwhile the short-term fluctuations. In this new formulation the desired temporal smoothness is incorporated in the objective function of the primal problem through the maximization of the correlation between the actual and the previous models. Moreover, new measures are presented in order to judge the quality of a partitioning produced at a given time. The new measures are the weighted sum of the snapshot quality and the temporal quality. The former only measures the quality of the current clustering with respect to the current data, while the latter measures the temporal smoothness in terms of the ability of the actual model to cluster the historic data. These new measures can be also used to perform the model selection.

- **Chapter** 5 discusses the application of KSC to an industrial case. Here we assume stationarity, i.e. the regimes experienced by the system under analysis do not change over time. Vibration data are collected from a packing machine to monitor its conditions. In order to describe the ongoing degradation process due to the dirt accumulation in the sealing jaws we first apply a windowing operation on the data, accounting for historical values of the sealing quality. The size of the window, together with the bandwidth of the Radial Basis Function (RBF) kernel and the number of clusters are tuned using the BLF criterion. Then an optimal kernel spectral clustering model is trained offline to identify two main regimes, that we can interpret as normal behaviour and critical conditions (need of maintenance). Thanks to the out-of-sample extension property, this model is used online to predict in advance when the machine needs maintenance. In principle, this implies the maximization of the production capacity and the minimization of downtimes.

- **Chapter** 6 deals with clustering non-stationary data. A new adaptive method named incremental kernel spectral clustering (IKSC) is devised. In a first phase a KSC model is constructed, then it is updated online according to the new points belonging to the data stream. The central idea behind the proposed technique concerns expressing the clustering model in terms of prototypes in the eigenspace, which are continuously adapted through the out-of-sample eigenvectors calculation. Moreover, the training set is formed only by the cluster centers in the input space, which are also updated in response to new data. This compact representation of the model and the training set in terms of cluster centroids makes the method computationally efficient and allows to properly track the evolution of complex patterns over time.

- **Chapter** 7 concludes the thesis and proposes future research directions.

In general, in all the experiments discussed in each chapter the values reported to assess the cluster quality are average measures over 10 runs of the algorithm under

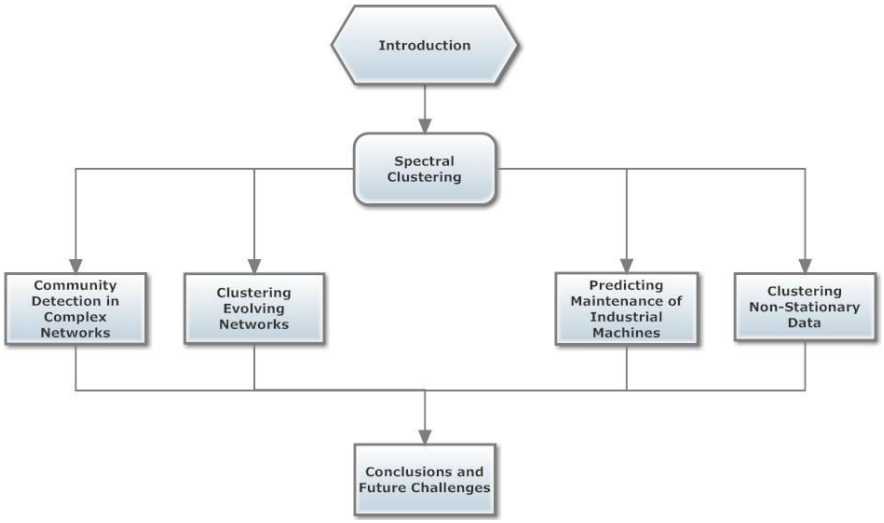


Figure 1.1: **Thesis overview**.

investigation. Although a fully statistical significance analysis has not always been performed, the mean values give good indications about the performances of the different methods.

1.5 Main Contributions

In what follows the main contributions of this thesis are resumed:

- **Community detection via KSC**. We conceived a complete methodology to cluster network data. The whole procedure can be summarized in three main stages: extract from the network a small sub-graph which retains the community structure of the entire graph, train an optimal KSC model using the selected sub-graph as training set, use the out-of-sample extension to assign the memberships to the remaining nodes. The precise tuning of the hyper-parameters and the choice of an appropriate kernel function are also essential. Furthermore, an algorithm to provide moderated outputs named SKSC and a related model selection criterion called AMS are proposed. Finally we mention how the technique can be used for large scale applications. The relevant papers are [76, 79, 81, 96, 97, 94, 93, 92].

- **KSC with memory effect**. A new model is designed to handle evolving networks. At the level of the primal optimization problem typical of LS-SVM, we introduce a memory effect in the objective function to account for the temporal smoothness of the clustering results over time. Also new model selection criteria specific for the given application are introduced. The related publications are [80, 82, 114, 77].

- **KSC for predictive maintenance**. We have successfully applied KSC to a complex industrial case. We developed a clustering model able to infer the degradation process affecting a packing machine from the vibration signals registered by accelerometers placed on the sealing jaws. A critical modelling phase concerned the windowing of the data in order to describe the history of the sealing quality. Moreover the model selection stage was also crucial. Finally, to improve the interpretability of the results a probabilistic output has been provided. This contribution is reported in [78].

- **Incremental KSC (IKSC)**. We presented a new algorithm to perform online clustering in a non-stationary environment. IKSC exploits the out-of-sample extension property of KSC to continuously adapt the initial model. In this way it is able to catch the dynamics of the clusters evolving over time. The IKSC method can model merging, splitting, appearance, death, expansion and shrinking of clusters, in a fast and accurate way [75].

Chapter 2

Spectral clustering

Spectral clustering methods have been reported to often outperform the traditional approaches such as K-means and hierarchical clustering in many real-life problems. We start this chapter with a description of the basic concepts behind spectral partitioning. We discuss advantages, like the ability of detecting complex clustering boundaries, and disadvantages, mainly related to the absence of a model selection scheme and the out-of-sample extension to unseen data.

Then we summarize the kernel spectral clustering (KSC) model, which is formulated as a weighted kernel PCA problem in the primal-dual optimization framework typical of Least Squares Support Vector Machines (LS-SVMs). Thanks to this representation, KSC solves the above mentioned drawbacks of spectral clustering, since it can be trained, validated by means of a tuning criterion called Balanced Line Fit (BLF), and tested in an unsupervised learning procedure.

Finally we propose an algorithm for soft (or fuzzy) clustering named soft kernel spectral clustering (SKSC). Basically, instead of using the hard assignment method present in KSC, we suggest a fuzzy assignment based on the cosine distance from the cluster prototypes in the space of the projections. We also introduce a related model selection technique, called average membership strength (AMS) criterion, which solves the main difficulties of BLF. Roughly speaking, SKSC is observed to improve the cluster performance upon KSC mainly when the clusters overlap to a large extent.

2.1 Classical Spectral Clustering

2.1.1 Introduction

Spectral clustering (SC) represents an elegant and effective solution to the graph partitioning problem. It makes use of the spectral properties of a matrix representation of the graph called Laplacian to divide it into weakly connected sub-graphs. It can be applied directly to network data to divide the vertices into several non-overlapping groups, or it can be used to cluster any kind of data. In this case the matrix of pairwise similarities between the data points serves as the network to partition. In [29] an explanation of SC by the point of view of graph theory is given. The authors of [108] present a SC algorithm which successfully deals with a number of challenging clustering problems. Moreover, an analysis of the algorithm by means of matrix perturbation theory gives conditions under which a good performance is expected. In [11] a new cost function for SC based on a measure of error between a given partition and a solution of the spectral relaxation of a minimum normalized cut problem is derived. The authors of [63] analyse the SC technique by means of a bi-criteria measure to assess the quality of a clustering result. An exhaustive tutorial on SC has been presented in [136]. In what follows we only depict the basic idea behind spectral partitioning, which originated by studying the graph partitioning problem in graph theory. The interested reader can refer to the aforementioned works for a deeper discussion.

2.1.2 The Graph Partitioning Problem

A graph (or network) $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a mathematical structure used to model pairwise relations between certain objects. It refers to a set of N vertices or nodes $\mathcal{V} = \{v_i\}_{i=1}^{N}$ and a collection of edges $\mathcal{E}$ that connect pairs of vertices. If the edges are provided with weights the corresponding graph is weighted, otherwise it is referred as an unweighted graph. The topology of a graph is described by the similarity (or affinity) matrix, which is an $N \times N$ matrix S, where S_{ij} indicates the link between the vertices i and j. Associated to the similarity matrix there is the degree matrix $D = \text{diag}(d)$, with $d = [d_1, \ldots, d_N]^T = S1_N$ and 1_N indicating the $N \times 1$ vector of ones. Basically the degree d_i of node i is the sum of all the edges (or weights) connecting node i with the other vertices: $d_i = \sum_{j=1}^{N} S_{ij}$.

In the most basic formulation of the graph partitioning problem one is given an unweighted graph and asked to split it into k non-overlapping groups $\mathcal{A}_1, \ldots, \mathcal{A}_k$ in order to minimize the cut size, which is the number of edges running between the

groups[1]. In order to favour balanced clusters, we can consider the normalized cut NC, defined as:

$$\text{NC}(\mathcal{A}_1, \ldots, \mathcal{A}_k) = k - \text{tr}(G^T L_n G) \tag{2.1}$$

where:

- $L_n = I - D^{-\frac{1}{2}} S D^{-\frac{1}{2}}$ is called the normalized Laplacian
- $G = [g_1, \ldots, g_k]$ is the matrix containing the normalized cluster indicator vectors $g_l = \frac{D^{\frac{1}{2}} f_l}{||D^{\frac{1}{2}} f_l||_2}$
- f_l, with $l = 1, \ldots, k$, is the cluster indicator vector for the l-th cluster. It has a 1 in the entries corresponding to the nodes in the l-th cluster and 0 otherwise. Moreover, the cluster indicator matrix can be defined as $F = [f_1, \ldots, f_k] \in \{0, 1\}^{N \times k}$.

The NC optimization problem is stated as follows:

$$\begin{aligned} \min_{G} \quad & k - \text{tr}(G^T L_n G) \\ \text{subject to} \quad & G^T G = I \end{aligned} \tag{2.2}$$

with I denoting the identity matrix. Unfortunately this is a NP-hard problem. However we can find good approximate solutions in polynomial time by using a relaxation method, i.e. allowing G to take continuous values:

$$\begin{aligned} \min_{\hat{G}} \quad & k - \text{tr}(\hat{G}^T L_n \hat{G}) \\ \text{subject to} \quad & \hat{G}^T \hat{G} = I. \end{aligned} \tag{2.3}$$

with $\hat{G} \in \mathbb{R}^{N \times k}$. In this case it can be shown that solving problem (2.3) is equivalent to finding the solution to the following eigenvalue problem:

$$L_n g = \lambda g. \tag{2.4}$$

Basically, the relaxed clustering information is contained in the eigenvectors corresponding to the k smallest eigenvalues of the normalized Laplacian L_n. In addition to the normalized Laplacian, other Laplacians can be defined, like the unnormalized Laplacian $L = D - S$ and the random walk Laplacian $L_{rw} = D^{-1} S$. The latter is appealing for its suggestive interpretation in terms of a Markov random walk.

[1] If the graph is weighted, the objective is to find a partition of the graph such that the edges between different groups have very low weights.

Algorithm 1: SC algorithm [54]

Data: positive (semi-) definite similarity matrix $S \in \mathbb{R}^{N \times N}$, number k of clusters to construct.
Result: clusters $\mathcal{A}_1, \ldots, \mathcal{A}_k$.

1 compute the graph Laplacian (L, L_n or L_{rw})
2 compute the eigenvectors $\hat{g_1}, \ldots, \hat{g_k}$ corresponding to the smallest k eigenvalues
3 let $\hat{G} \in \mathbb{R}^{N \times k}$ be the matrix containing the vectors $\hat{g_1}, \ldots, \hat{g_k}$ as columns
4 for $i = 1, \ldots, N$ let $u_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of G
5 cluster the points u_i into clusters $\mathcal{C}_1, \ldots, \mathcal{C}_k$
6 compute the final partitioning $\mathcal{A}_1, \ldots, \mathcal{A}_k$, with $A_i = \{j | u_j \in C_i\}$.

2.1.3 Link with Markov Chains

A well known relationship between graphs and Markov chains exists: any graph has an associated random walk in which the probability of leaving a vertex is distributed among the outgoing edges according to their weight. For a given graph with N nodes and m edges the probability vector can be defined as $p_{t+1} = Pp_t$, where $P = D^{-1}S$ indicates the transition matrix with the ij-th entry representing the probability of moving from node i to node j in one step. Under these assumptions we have an ergodic and reversible Markov chain with stationary distribution vector π with components $\pi_i = \frac{d_i}{2m}$. It can be shown that this distribution describes the situation in which the random walker remains most of the time in the same cluster with rare jumps to the other clusters [100]. Moreover $L_{rw} = I - P$ and the eigenvectors corresponding to the smallest eigenvalues of L_{rw} are the same as the eigenvectors related to the largest eigenvalues of P:

$$L_n g = \lambda g \Rightarrow (I - P)g = \lambda g \Rightarrow g - Pg = \lambda g \Rightarrow Pg = (I - \lambda I)g. \tag{2.5}$$

For the reader interested in having a deeper insight into this topic, we advice to explore [100, 99, 36].

2.1.4 Basic Algorithm

As mentioned before, the classical spectral clustering algorithm can be applied to partition any kind of data, not only networks. Indeed, if for the networks we are immediately provided with the affinity matrix S, in case of data points we have to construct S starting from some similarity function. The basic steps are described in algorithm 1. Thanks to the mapping of the original input data in the eigenspace, SC is able to unfold the manifold the data are embedded in and to detect complex clustering boundaries. On the other hand, it has some clear disadvantages:

- it is not clear how to properly construct the similarity matrix S and the number of clusters must be provided beforehand. In [143] the authors proposed a solution to this model selection issue by introducing a parameter-free SC.
- there is no clear way as how test data points should be assigned to the initial clusters, since the embedding eigenvectors are only defined for the full dataset. In [138] and [44] the authors employed the Nyström method to find approximate eigenvectors for out-of-sample data and reduce the computational load for large scale applications. In [45] the authors proposed a sparse spectral clustering method based on the incomplete Cholesky decomposition (ICD), which constructs an approximation of the Laplacian in terms of capturing the structure of the matrix. By using only the information related to the pivots selected by the ICD, a method to compute cluster memberships for out-of-sample points is also introduced.

2.2 Kernel Spectral Clustering

2.2.1 Generalities

Kernel Spectral Clustering (KSC) represents a spectral clustering formulation in the LS-SVM optimization framework with primal and dual representations. The dual problem is an eigenvalue problem, related to spectral clustering. KSC has two main advantages with respect to classical spectral clustering:

- a precise model selection scheme to tune the hyper-parameters
- the out-of-sample extension to test points by means of an underlying model.

After finding the optimal hyper-parameters, the clustering model can be trained on a subset of the full data and readily applied to unseen test points in a learning framework.

2.2.2 Least Squares Support Vector Machine

The Support Vector Machine (SVM) is a state-of-the-art classification method. It performs linear classification in a high-dimensional kernel-induced feature space, which corresponds to a non-linear decision boundary in the original input space. LS-SVM differs from SVM because it uses a L_2 loss function in the primal problem and equality instead of inequality constraints. This typically leads to eigenvalue problems or linear systems at the dual level, in the context of principal component analysis [129] and classification or regression [130], respectively.

Given a training data set $\mathcal{D}_{\text{Tr}} = \{(x_i, y_i)\}_{i=1}^{N_{\text{Tr}}}$, where $x_i \in \mathbb{R}^d$ are the training points and $y_i \in \{-1, 1\}$ are the related labels, the primal problem of the LS-SVM binary classifier[2] can be stated as [130]:

$$\begin{aligned} \min_{w,e_i,b} \quad & \frac{1}{2}w^T w + \gamma \frac{1}{2}\sum_{i=1}^{N_{\text{Tr}}} e_i^2 \\ \text{subject to} \quad & y_i(w^T\varphi(x_i) + b) = 1 - e_i, i = 1, \ldots, N_{\text{Tr}}. \end{aligned} \tag{2.6}$$

The expression $\hat{y} = w^T\varphi(x) + b$ indicates the model in the primal space. It is linear with respect to the parameter vector w but the relationship between x and y can be non-linear if the feature map $\varphi(\cdot)$ is a non-linear function. With γ we indicate the regularization parameter which controls the trade-off between the model complexity and the minimization of the training error. If we construct the Lagrangian we have:

$$\mathcal{L}(w, e_i, b, \alpha_i) = \frac{1}{2}w^T w + \gamma \frac{1}{2}\sum_{i=1}^{N_{\text{Tr}}} e_i^2 - \sum_{i=1}^{N_{\text{Tr}}} \alpha_i(y_i(w^T\varphi(x_i) + b) - 1 + e_i) \tag{2.7}$$

where α_i are the Lagrange multipliers. The KKT optimality conditions are:

$\frac{\partial \mathcal{L}}{\partial w} = 0 \rightarrow w = \sum_{i=1}^{N_{\text{Tr}}} \alpha_i y_i \varphi(x_i)$,
$\frac{\partial \mathcal{L}}{\partial e_i} = 0 \rightarrow \alpha_i = \gamma e_i$,
$\frac{\partial \mathcal{L}}{\partial b} = 0 \rightarrow \sum_{i=1}^{N_{\text{Tr}}} \alpha_i y_i = 0$,
$\frac{\partial \mathcal{L}}{\partial \alpha_i} = 0 \rightarrow y_i(w^T\varphi(x_i) + b) - 1 + e_i = 0$.

Eliminating the primal variables e_i and w leads to the following linear system in the dual problem:

$$\left[\begin{array}{c|c} \Omega + I_{N_{\text{Tr}}}/\gamma & y \\ \hline y^T & 0 \end{array}\right] \left[\begin{array}{c} \alpha \\ \hline b \end{array}\right] = \left[\begin{array}{c} 1_{N_{\text{Tr}}} \\ \hline 0 \end{array}\right] \tag{2.8}$$

where $y = [y_1; \ldots; y_{N_{\text{Tr}}}]$, $1_{N_{\text{Tr}}} = [1; \ldots; 1]$, $\alpha = [\alpha_1; \ldots; \alpha_{N_{\text{Tr}}}]$. The term Ω means the kernel matrix with entries $\Omega_{ij} = \varphi(x_i)^T\varphi(x_j) = K(x_i, x_j)$. With $K : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ we denote the kernel function which maps the input points into the high dimensional feature space $\varphi(\cdot)$. For example, by using a Radial Basis Function (RBF) kernel expressed by $K(x_i, x_j) = \exp(-||x_i - x_j||_2^2/\sigma^2)$, one is able to construct a model of arbitrary complexity. Finally, after solving the previous linear system, the LS-SVM classification model in the dual representation becomes:

$$y(x) = \text{sign}(\sum_{i=1}^{N_{\text{Tr}}} \alpha_i y_i K(x, x_i) + b). \tag{2.9}$$

[2]Multi-class classification problems are decomposed into multiple binary classification tasks with the possibility to use several coding-decoding schemes [131].

The constrained optimization framework with explicit use of regularization explained above represents the core model not only for classification, but also regression and unsupervised learning, as we will see in the remainder of this dissertation.

2.2.3 Primal-Dual Formulation

Given a training data set $\mathcal{D}_{\text{Tr}} = \{x_i\}_{i=1}^{N_{\text{Tr}}}$, the multi-cluster KSC model [6] in the LS-SVM framework is formulated as a weighted kernel PCA problem [101] decomposed in $l = k-1$ binary problems, where k is the number of clusters to find:

$$\begin{aligned} \min_{w^{(l)}, e^{(l)}, b_l} \quad & \frac{1}{2}\sum_{l=1}^{k-1} w^{(l)^T} w^{(l)} - \frac{1}{2N_{\text{Tr}}}\sum_{l=1}^{k-1} \gamma_l e^{(l)^T} V e^{(l)} \\ \text{subject to} \quad & e^{(l)} = \Phi w^{(l)} + b_l 1_{N_{\text{Tr}}}. \end{aligned} \tag{2.10}$$

The $e^{(l)} = [e_1^{(l)}, \ldots, e_{N_{\text{Tr}}}^{(l)}]^T$ are the projections of the data points $\{x_i\}_{i=1}^{N_{\text{Tr}}}$ mapped in the feature space along the direction $w^{(l)}$, also called score variables. The optimization problem (2.10) can then be interpreted as the maximization of the weighted variances $C_l = e^{(l)^T} V e^{(l)}$ and the contextual minimization of the squared norm of the vector $w^{(l)}$, $\forall l$. Through the regularization constants $\gamma_l \in \mathbb{R}^+$ we trade-off the model complexity expressed by $w^{(l)}$ with the correct representation of the training data. $V \in \mathbb{R}^{N_{\text{Tr}} \times N_{\text{Tr}}}$ is the weighting matrix and Φ is the $N_{\text{Tr}} \times d_h$ feature matrix $\Phi = [\varphi(x_1)^T; \ldots; \varphi(x_{N_{\text{Tr}}})^T]$. The clustering model is expressed by:

$$e_i^{(l)} = w^{(l)^T} \varphi(x_i) + b_l, i = 1, \ldots, N_{\text{Tr}} \tag{2.11}$$

where as usual $\varphi : \mathbb{R}^d \to \mathbb{R}^{d_h}$ indicates the mapping to a high-dimensional feature space, b_l are bias terms, with $l = 1, \ldots, k-1$. The projections $e_i^{(l)}$ represent also the latent variables of the $k-1$ binary clustering indicators given by $\text{sign}(e_i^{(l)})$. The set of binary indicators form a code-book $\mathcal{CB} = \{c_p\}_{p=1}^k$, where each code-word is a binary word of length $k-1$ representing a cluster. The Lagrangian associated with the primal problem is:

$$\begin{aligned} \mathcal{L}(w^{(l)}, e^{(l)}, b_l, \alpha^{(l)}) = & \frac{1}{2}\sum_{l=1}^{k-1} w^{(l)^T} w^{(l)} - \frac{1}{2N_{\text{Tr}}}\sum_{l=1}^{k-1} \gamma_l e^{(l)^T} V e^{(l)} - \\ & \sum_{l=1}^{k-1} \alpha^{(l)^T} (e^{(l)} - \Phi w^{(l)} - b_l 1_{N_{\text{Tr}}}) \end{aligned} \tag{2.12}$$

where $\alpha^{(l)}$ are the Lagrange multipliers. The KKT optimality conditions are:

$\frac{\partial \mathcal{L}}{\partial w^{(l)}} = 0 \rightarrow w^{(l)} = \Phi^T \alpha^{(l)}$,
$\frac{\partial \mathcal{L}}{\partial e^{(l)}} = 0 \rightarrow \alpha^{(l)} = \frac{\gamma_l}{N_{\text{Tr}}} V e^{(l)}$,
$\frac{\partial \mathcal{L}}{\partial b_l} = 0 \rightarrow 1_{N_{\text{Tr}}}^T \alpha^{(l)} = 0$,
$\frac{\partial \mathcal{L}}{\partial \alpha^{(l)}} = 0 \rightarrow e^{(l)} - \Phi w^{(l)} - b_l 1_N = 0$.

Once we have solved the KKT conditions for optimality, if we set $V = D^{-1}$, we can derive the following dual problem:

$$D^{-1} M_D \Omega \alpha^{(l)} = \lambda_l \alpha^{(l)} \tag{2.13}$$

where Ω is the kernel matrix with ij-th entry $\Omega_{ij} = K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$. D is the graph degree matrix which is diagonal with positive elements $D_{ii} = \sum_j \Omega_{ij}$, M_D is a centering matrix defined as $M_D = I_{N_{\text{Tr}}} - \frac{1}{1_{N_{\text{Tr}}}^T D^{-1} 1_{N_{\text{Tr}}}} 1_{N_{\text{Tr}}} 1_{N_{\text{Tr}}}^T D^{-1}$, the $\alpha^{(l)}$ are dual variables, $\lambda_l = \frac{N_{\text{Tr}}}{\gamma_l}$, $K : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ is the kernel function. The dual representation of the model becomes:

$$e_i^{(l)} = \sum_{j=1}^{N_{\text{Tr}}} K(x_j, x_i) \alpha_j^{(l)} + b_l, j = 1, \ldots, N_{\text{Tr}}. \tag{2.14}$$

Moreover, by observing problem (2.13) one can realize that, apart from the presence of the centering matrix M_D, it is similar to problem (2.5). In fact the kernel matrix plays the role of the similarity matrix associated to the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $v_i \in \mathcal{V}$ equals to x_i. This is also the reason behind the choice of the weighting matrix in the primal problem as the inverse of the degree matrix D related to the kernel matrix Ω. Finally, the KSC method is sketched in algorithm 2. A visual representation of the KSC technique is also illustrated in figure 2.1.

2.2.4 Model Selection

In general the kernel hyper-parameters should be chosen carefully in order to ensure good generalization. This is particularly important for very flexible kernels such as the RBF kernel where too small values of the bandwidth σ will result in overfitting and too high values in a poor model.

To deal with this crucial issue, the KSC algorithm is provided with a model selection procedure based on the Balanced Line Fit (BLF) criterion [6]. It can be shown that in the ideal situation of compact and well separated clusters, the points $[e_i^{(1)}, \ldots, e_i^{(k-1)}]$, $i = 1, \ldots, N_{\text{Tr}}$, form lines, one per each cluster. Then, by exploiting this shape of the points in the projections space, BLF can be used to select optimal clustering parameters such as the number of clusters and eventually the kernel hyper-parameters. In particular the BLF is defined in the following way:

$$\text{BLF}(\mathcal{D}^V, k) = \eta \text{linefit}(\mathcal{D}^V, k) + (1 - \eta) \text{balance}(\mathcal{D}^V, k) \tag{2.15}$$

Algorithm 2: KSC algorithm [6]

Data: Training set $\mathcal{D}_{\text{Tr}} = \{x_i\}_{i=1}^{N_{\text{Tr}}}$, test set $\mathcal{D}_{\text{test}} = \{x_m^{\text{test}}\}_{m=1}^{N_{\text{test}}}$ kernel function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ positive definite and localized ($K(x_i, x_j) \to 0$ if x_i and x_j belong to different clusters), kernel parameters (if any), number of clusters k.

Result: Clusters $\{\mathcal{A}_1, \ldots, \mathcal{A}_k\}$, codebook $\mathcal{CB} = \{c_p\}_{p=1}^k$ with $\{c_p\} \in \{-1, 1\}^{k-1}$.

1 compute the training eigenvectors $\alpha^{(l)}$, $l = 1, \ldots, k-1$, corresponding to the $k-1$ largest eigenvalues of problem (2.13)

2 let $A \in \mathbb{R}^{N_{\text{Tr}} \times k-1}$ be the matrix containing the vectors $\alpha^{(1)}, \ldots, \alpha^{(k-1)}$ as columns

3 binarize A and let the code-book $\mathcal{CB} = \{c_p\}_{p=1}^k$ be composed by the k encodings of $Q = \text{sign}(A)$ with the most occurrences

4 $\forall i$, $i = 1, \ldots, N_{\text{Tr}}$, assign x_i to A_{p^*} where $p^* = \text{argmin}_p d_H(\text{sign}(\alpha_i), c_p)$ and $d_H(.,.)$ is the Hamming distance

5 binarize the test data projections $\text{sign}(e_m^{(l)})$, $m = 1, \ldots, N_{\text{test}}$, and let $\text{sign}(e_m) \in \{-1, 1\}^{k-1}$ be the encoding vector of x_m^{test}

6 $\forall m$, assign x_m^{test} to A_{p^*}, where $p^* = \text{argmin}_p d_H(\text{sign}(e_m), c_p)$.

where $\mathcal{D}^V$ represents the validation set and k as usual indicates the number of clusters. The linefit index equals 0 when the score variables are distributed spherically and equals 1 when the score variables are collinear, representing points in the same cluster. The balance index equals 1 when the clusters have the same number of elements and tends to 0 in extremely unbalanced cases. The parameter η controls the importance given to the linefit with respect to the balance index and takes values in the range $[0, 1]$. Thus, for instance BLF reaches its maximum value 1 in case of well distinct clusters of the same size if $\eta = 0.5$.

Extensive experiments have shown the usefulness of BLF for model selection. However, some drawbacks have been observed:

- often the criterion is biased toward $k = 2$ clusters
- it is not clear how to choose η
- it fails in case of large overlap between the clusters
- it is more suited for data points than network data.

In sections 2.3 and 3.2.2 we will show how two new model selection criteria introduced in [81] and [76] can solve these difficulties and then be used as a valid alternative to BLF.

2.2.5 Generalization

Spectral clustering methods provide a clustering only for the given training data without a clear extension to test points, as discussed in [44]. Moreover, the out-of-sample technique proposed therein consists of applying the Nyström method [12] in order to give an embedding for the test points by approximating the underlying eigenfunction. In [45] the information related to the pivots selected by the Incomplete Cholesky Decomposition (ICD) allows to compute cluster memberships for out-of-sample points. Other similar numerical techniques are used in [110] and [37] as a solution to using spectral clustering for large scale applications. On the other hand, the extension proposed in KSC is model-based, in the sense that the out-of-sample points are mapped onto the eigenvectors found in the training phase:

$$e^{(l)}_{\text{test}} = \Omega_{\text{test}}\alpha^{(l)} + b_l 1_{\text{Ntest}} \tag{2.16}$$

where Ω_{test} is the $N_{\text{test}} \times N$ kernel matrix evaluated using the test points with entries $\Omega_{\text{test,ri}} = K(x^{\text{test}}_r, x_i)$, $r = 1, \ldots, N_{\text{test}}$, $i = 1, \ldots, N_{\text{Tr}}$. The cluster indicators can be obtained by binarizing the score variables. As for the training nodes, the memberships are assigned by comparing these indicators with the code-book and selecting the nearest prototype based on Hamming distance. This scheme corresponds to an ECOC (Error Correcting Output Codes) decoding procedure.

To conclude, the LS-SVM framework in which KSC has been designed allows to train, validate and test the clustering model in an unsupervised learning scheme.

2.3 Soft Kernel Spectral Clustering

2.3.1 Overview

Most of the clustering methods performs only hard clustering, where each item is assigned to only one group. However, this works fine when the clusters are compact and well separated, while the performance can decrease dramatically when they overlap. Since this is the case in many real-world scenarios, soft or fuzzy clustering is becoming popular in many fields [57, 141]. In soft clustering each object belongs to several groups at the same time, with a different degree of membership. This is desirable not only to cope in a more effective way with overlapping clusters, but the uncertainties on data-to-clusters assignments help also to improve the overall interpretability of the results.

In what follows we describe a novel algorithm for fuzzy clustering named soft kernel spectral clustering (SKSC) [81]. SKSC is characterized by the same core model

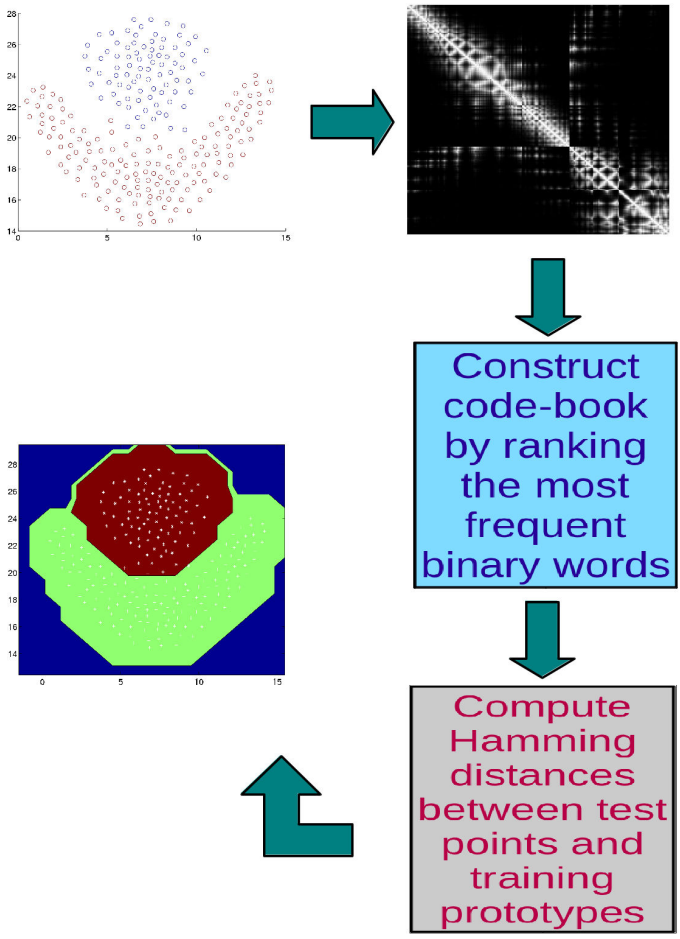


Figure 2.1: **KSC algorithm**. The dataset consists of a set $\mathcal{D} = \{x_i\}_{i=1}^{N}$ where $x_i \in \mathbb{R}^2$, and relates to a binary clustering problem with nonlinear boundary. After binarizing the matrix containing the eigenvectors of the Laplacian as columns, a code-book with the most frequent binary words representing the training cluster prototypes is formed. The test points are mapped into the training eigenspace through the out-of-sample extension. These projections are then binarized and the points are assigned to the closest prototype in terms of Hamming distance, by means of an ECOC decoding procedure.

as KSC, but it is provided with a different assignment rule allowing soft cluster memberships. A first attempt to provide a sort of probabilistic output in KSC was already done in [5]. However, in the cited work the underlying assumption is that there is few overlap between the clusters. On the other hand SKSC can handle cases where a large amount of overlap between clusters is present. Moreover, SKSC uses a new method to tune the number of clusters and the kernel hyper-parameters based on the soft assignment. This model selection technique is called Average Membership Strength (AMS) criterion. The latter can solve the issues arising with BLF mentioned in section 2.2.4. In fact, unlike BLF, AMS is not biased toward two clusters, does not have any parameter to tune and can be used in an effective way also with overlapping clusters.

2.3.2 Algorithm

The main idea behind soft kernel spectral clustering is to use KSC as an initialization step in order to find a first division of the training data into clusters. Then this grouping is refined by re-calculating the prototypes in the score variables space, and the cluster assignments are performed by means of the cosine distance between each point and the prototypes. This allows also to obtain highly sparse models as explained in [91], where a possible alternative to reduced set methods (see [95]) is proposed.

As already pointed out in section 2.2.4, in the projections/score variables space the points belonging to the same cluster appear aligned in the absence of overlap (see center of Figure 2.2). In this ideal situation of clear and well distinct groupings, any soft assignment should reduce to a hard assignment, where every point must belong to one cluster with membership 1 and to the other clusters with membership 0. In fact, the membership reflects the certainty with which we can assign a point to a cluster and it can be thought as a kind of subjective probability. In order to cope with this situation, the cosine distance from every point to the prototypes can be used as the basis for the soft assignment. In this way, in the perfect above-mentioned scenario, every point positioned along one line will be assigned to that cluster with membership or probability equal to 1, since the cosine distance from the corresponding prototype is 0, being the two vectors parallel (see bottom of Figure 2.2). Given the projections for the training points $e_i = [e_i^{(1)}, \ldots, e_i^{(k-1)}]$, $i = 1, \ldots, N_{\text{Tr}}$ and the corresponding hard assignments q_i^p we can calculate for each cluster the new prototypes $s_1, \ldots, s_p, \ldots, s_k$, $s_p \in \mathbb{R}^{k-1}$ as:

$$s_p = \frac{1}{n_p} \sum_{i=1}^{n_p} e_i \tag{2.17}$$

where n_p is the number of points assigned to cluster p during the initialization step by KSC. Then we can calculate the cosine distance between the i-th point in the score

Algorithm 3: SKSC algorithm [81]

Data: Training set $\mathcal{D}_{\text{Tr}} = \{x_i\}_{i=1}^{N_{\text{Tr}}}$ and test set $\mathcal{D}_{\text{test}} = \{x_m^{\text{test}}\}_{m=1}^{N_{\text{test}}}$, kernel function $K : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ positive definite and localized ($K(x_i, x_j) \rightarrow 0$ if x_i and x_j belong to different clusters), kernel parameters (if any), number of clusters k.

Result: Clusters $\{\mathcal{A}_1, \ldots, \mathcal{A}_p, \ldots, \mathcal{A}_k\}$, soft cluster memberships $cm^{(p)}, p = 1, \ldots, k$, cluster prototypes $\mathcal{SP} = \{s_p\}_{p=1}^k$, $s_p \in \mathbb{R}^{k-1}$.

1 Initialization by solving eq. (2.14).

2 Compute the new prototypes $s_1, \ldots, s_k$ (eq. (2.17)).

3 Calculate the test data projections $e_m^{(l)}, m = 1, \ldots, N_{\text{test}}, l = 1, \ldots, k-1$.

4 Find the cosine distance between each projection and all the prototypes (eq. (2.18)) $\forall m$, assign x_m^{test} to cluster A_p with membership $cm^{(p)}$ according to eq. (2.19).

variables space and a prototype s_p using the following formula:

$$d_{ip}^{\cos} = 1 - e_i^T s_p / (||e_i||_2 ||s_p||_2). \tag{2.18}$$

The membership of point i to cluster q can be expressed as:

$$cm_i^{(q)} = \frac{\prod_{j \neq q} d_{ij}^{\cos}}{\sum_{p=1}^k \prod_{j \neq p} d_{ij}^{\cos}} \tag{2.19}$$

with $\sum_{p=1}^k cm_i^{(p)} = 1$. As discussed in [14], this membership is given as a subjective probability and it indicates the strength of belief in the clustering assignment.

The out-of-sample extension on unseen data consists of two steps:

1. project the test points onto the eigenspace spanned by $[\alpha^{(1)}, \ldots, \alpha^{(k-1)}]$ using eq. (2.16)
2. calculate the cosine distance between these projections and the training cluster prototypes, and then the corresponding soft assignment by means of eq. (2.19).

The SKSC method is summarized in algorithm 3. The main steps of this technique are depicted in figure 2.3.

2.3.3 Model Selection

From the soft assignment technique explained in the previous section a new model selection method can be derived. In fact, we can calculate a kind of mean membership per cluster indicating the average degree of belonging of the points to that cluster. If

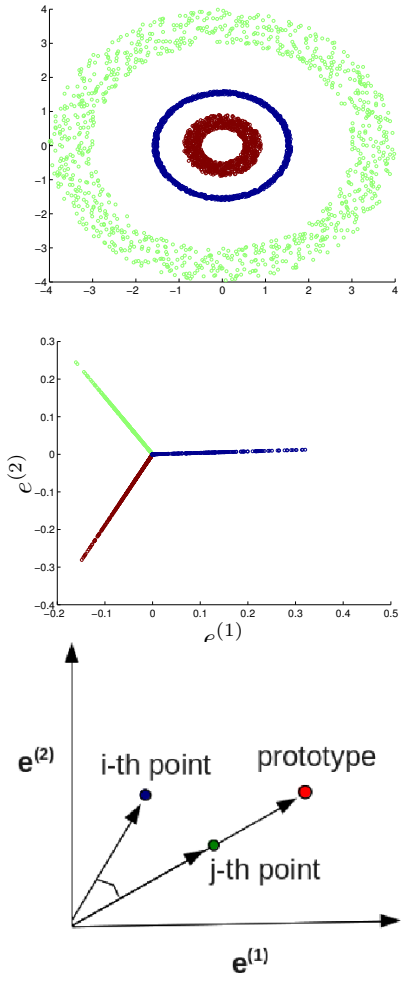


Figure 2.2: **Clusters as lines in the $e^{(l)}$ space**: example of how three distinct clusters (3 concentric rings on the top in a 2D space) appear in the projection space in the KSC formulation (center). On the bottom an illustration of the cosine distance in the score variable space is given: the j-th point is parallel to the prototype so its membership is 1. This is not the case for the i-th point.

Algorithm 4: AMS model selection algorithm for SKSC [81]

Data: training and validation sets, kernel function $K : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ positive definite and localized.

Result: selected number of clusters k, kernel parameters (if any).

1 Define a grid of values for the parameters to select
2 Train the related SKSC model using the training set
3 Compute the soft cluster assignments for the points of the validation set
4 For every partition calculate the related AMS score using eq. (2.20)
5 Choose the model with the highest score.

we do the same for every cluster and we take the mean, we have what we call the Average Membership Strength (AMS) criterion[3]:

$$\text{AMS} = \frac{1}{k}\sum_{p=1}^{k}\frac{1}{n_p}\sum_{i=1}^{n_p} cm_i^{(p)}. \tag{2.20}$$

For $k = 2$ the Euclidean distance between the points and the prototypes is used[4], while for $k > 2$ we consider the cosine distance. Unlike BLF, AMS does not have any parameter to tune, it is not biased toward $k = 2$ and as we will show in the experimental section it can be used in an effective way also in case of overlapping clusters. The model selection procedure is described in algorithm 4.

2.3.4 Toy Examples

In this section some experimental examples with artificial data are presented to illustrate the proposed soft clustering technique. We compare SKSC with KSC in terms of Adjusted Rand Index (ARI) [58], mean Silhouette value (MSV) [119] and Davies-Bouldin index (DBI) [34] (see appendix[5]), and AMS with BLF. In general we observe that SKSC achieves better results than KSC and the AMS criterion appears to be more useful than BLF for model selection, mostly when a big amount of overlap between clusters is present.

We have designed an experiment with the goal of understanding the role of the overlap rather than the nonlinearities to affect the clustering performances. Thus, synthetic data consisting of 1000 data points have been generated from a mixture of three 2D Gaussians, with 500 samples drawn from each component of the mixture. We then

[3] Future work can be related to experiment with a weighted version of the AMS criterion. The weights could be chosen, for instance, to emphasize the contribution of points with highest membership, in order to reduce the influence of the noise.

[4] In this case the cosine distance is not discriminative because the points lie on a line.

[5] A summary of the main cluster quality measures used in this thesis will be given in the appendix.

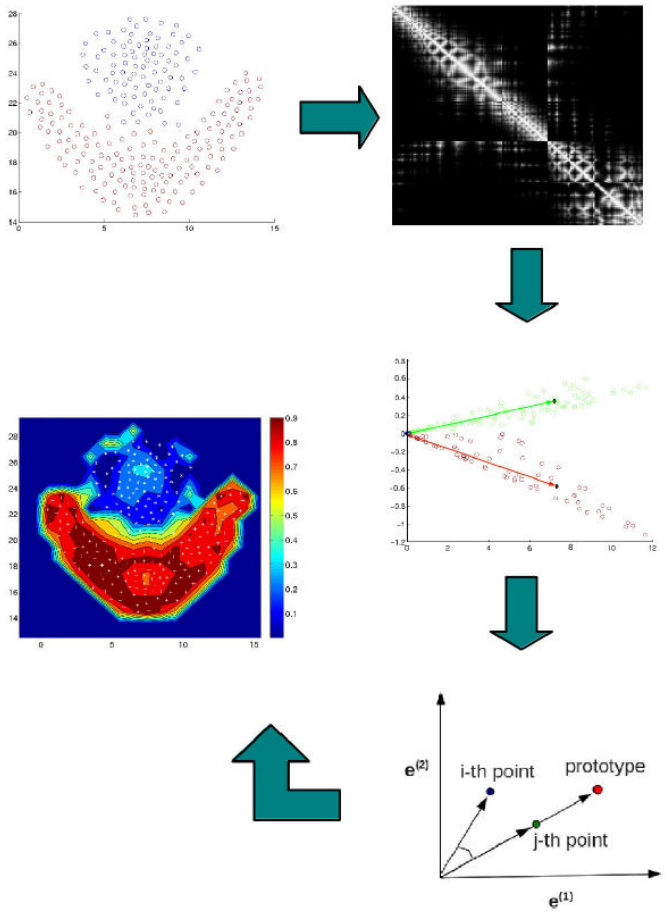


Figure 2.3: **SKSC algorithm**. The dataset consists of a set $\mathcal{D} = \{x_i\}_{i=1}^{N}$ where $x_i \in \mathbb{R}^2$, and relates to a binary clustering problem with nonlinear boundary. From the projections of the training points in the eigenspace of the Laplacian some cluster prototypes are computed. The projection of the test points are then assigned to the different clusters with a probability depending on the cosine distance from each of them.

consider different amounts of overlap between these clouds of points, as depicted in figure 2.4 from top to bottom.

In figure 2.5 we show the model selection plots for these three cases. We can see that AMS, even in case of large overlap, is able to correctly identify the presence of the three distinct Gaussian clouds. This is not the case for the BLF criterion. In tables 2.1, 2.2, 2.3 the clustering results of KSC and SKSC (when fed with correct parameters) are evaluated. We can notice that SKSC outperforms KSC when the overlap between the clusters increases. Finally, in figure 2.6 the soft clustering results produced by SKSC for the large overlap case are depicted. As we would expect, the probability to belong to each of the 3 clusters decreases in the overlapping regions.

	KSC	**SKSC**
MSV	0.96	0.96
DBI	0.24	0.24
ARI	1	1

Table 2.1: **Three Gaussians, no overlap**: cluster quality evaluation of KSC and SKSC based on Mean Silhouette Value (MSV), Davies-Bouldin Index (DBI), and Adjusted Rand Index (ARI). A summary of these and others cluster quality criteria is given in the Appendix.

	KSC	**SKSC**
MSV	0.90	**0.91**
DBI	0.34	**0.33**
ARI	0.96	**0.99**

Table 2.2: **Three Gaussians, few overlap**: clustering results of KSC and SKSC according to Mean Silhouette Value, Davies-Bouldin Index, and Adjusted Rand Index (see Appendix). The best performances are in bold.

	KSC	**SKSC**
MSV	0.59	**0.64**
DBI	0.79	**0.76**
ARI	0.65	**0.74**

Table 2.3: **Three Gaussians, large overlap**: cluster quality measures (see Appendix) for KSC and SKSC, with the best performance in bold.

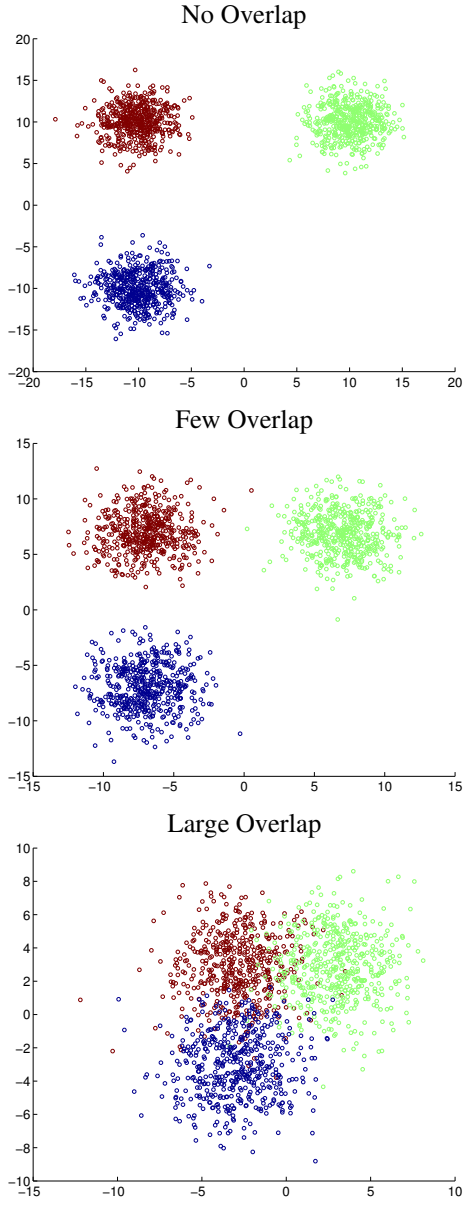


Figure 2.4: **Three Gaussians dataset**. The dataset consists of three sets of N points in a 2D space ($x_i \in \mathbb{R}^2, i = 1, \ldots, N$) representing Gaussian mixtures with different amount of overlap.

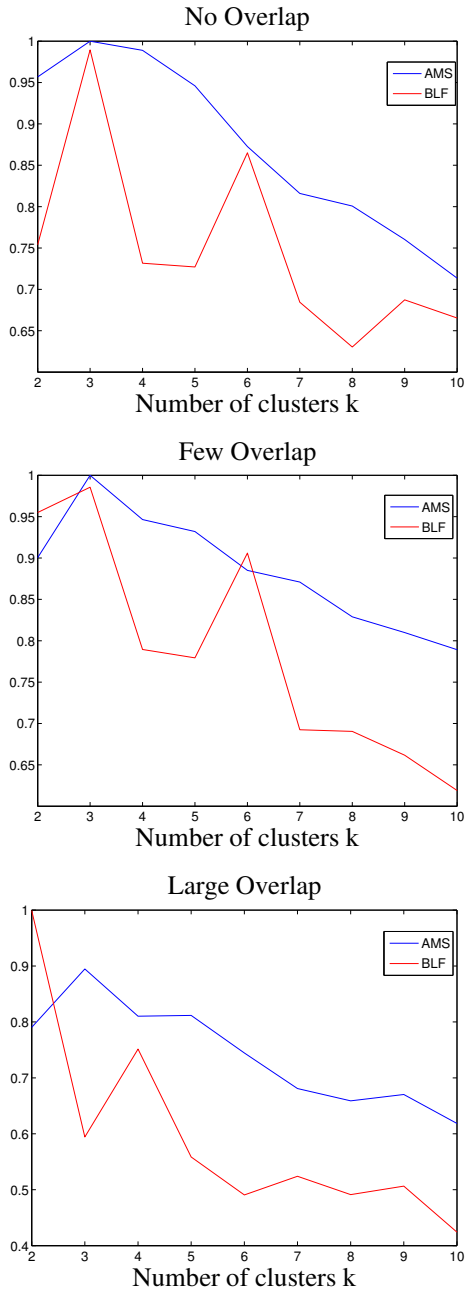


Figure 2.5: **Model selection for the three Gaussians dataset**. The true number of clusters is $k = 3$. Two model selection criteria, namely AMS and BLF, are contrasted. We can notice how the former can better deal with overlapping clusters.

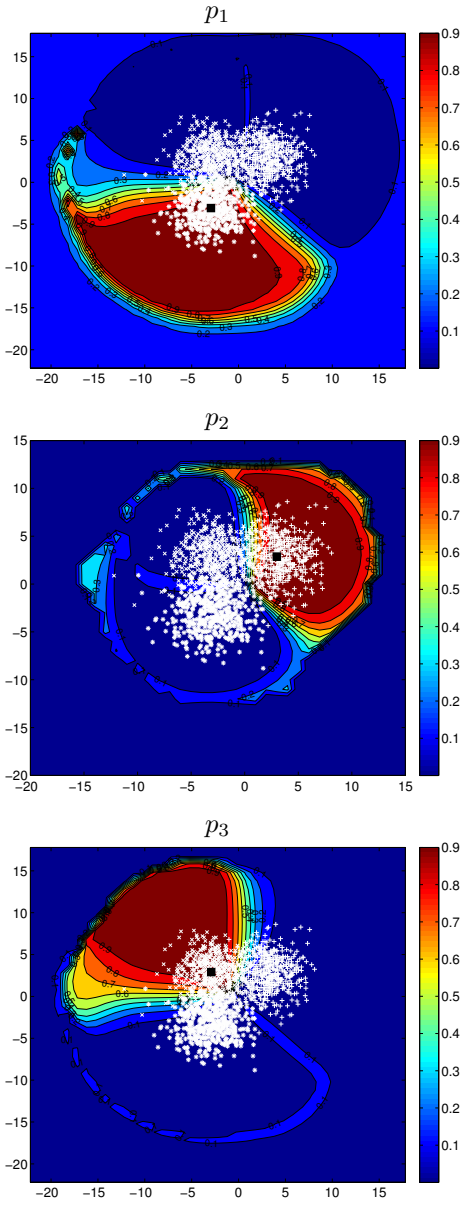


Figure 2.6: **Soft clustering results for three Gaussians dataset with large overlap**: probability to belong to cluster 1 (p_1, top), to cluster 2 (p_2, center) and to cluster 3 (p_3, bottom) detected by SKSC for the large overlap case. We can notice that all the probabilities reach their maximum value around the centroids and then decrease in the overlapping region.

2.3.5 Application: Image Segmentation

Here we show the usage of SKSC for the image segmentation task. In computer vision, image segmentation is the process of partitioning a digital image into multiple sets of pixels, such that pixels in the same group share certain visual characteristics.

We randomly selected 10 color images from the Berkeley image dataset [98]. Before clustering the images, a pre-processing of the data was necessary. In particular, we computed a local color histogram with a 5×5 pixels window around each pixel using minimum variance color quantization of 8 levels. Then, in order to compare the similarity between two histograms $h^{(i)}$ and $h^{(j)}$, the positive definite χ^2 kernel $K(h^{(i)}, h^{(j)}) = \exp(-\frac{\chi^2_{ij}}{\sigma^2_\chi})$ has been adopted [44]. The symbol χ^2_{ij} denotes the χ^2_{ij} statistical test used to compare two probability distributions [115], σ_χ as usual indicates the bandwidth of the kernel.

A randomly selected subset of 1000 pixels forms the training set and the whole image is the test set, in order to obtain the final segmentation. After properly selecting the number of clusters and the kernel bandwidth for each image, we compared the segmentation obtained by KSC and SKSC by using three evaluation measures [8]:

- F-measure ($\frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$) with respect to human ground-truth boundaries
- Variation of information (VI): it measures the distance between two segmentations in terms of their average conditional entropy. Low values indicate good match between the segmentations.
- Probabilistic Rand Index (PRI): it operates by comparing the congruity of assignments between pairs of elements in the clusters found by an algorithm and multiple ground-truth segmentations.

The segmentation results are shown in figure 2.7 (F-measure) and tables 2.4 (VI) and 2.5 (PRI). For comparison purposes, the soft partitioning obtained by SKSC is transformed in hard clustering results by assigning the pixels to the closest prototype. In this case the performances of SKSC and KSC are overall similar, most probably due to the fact that the images are characterized by clear segments (i.e. small overlap). Finally, in figure 2.8 the segmentations accomplished by SKSC on three selected images are depicted for illustration purpose.

2.4 Conclusions

In this chapter, after introducing the basics of spectral partitioning, we have discussed the LS-SVM modelling framework and we summarized the KSC algorithm. The latter is a spectral clustering model formulated as weighted kernel PCA in a primal-dual optimization setting. This formulation allows a simple extension of the model to out-of-sample points and a systematic model selection procedure based on the BLF criterion. Then, we presented the soft kernel spectral clustering (SKSC) technique, which is based on KSC but it uses a fuzzy assignment rule. The latter depends on the cosine distance of every point from the cluster prototypes in the projections space. A new model selection technique derived from this soft assignment, namely the average membership strength (AMS) criterion, is also proposed. We showed how AMS solves the main drawbacks of BLF and can be used more effectively in presence of overlap between clusters. We illustrated on toy data and an image segmentation problem, that SKSC outperforms KSC mainly in the most difficult tasks (i. e. when clusters overlap to a large extent). Moreover, SKSC produces more interpretable results due to its fuzzy nature.

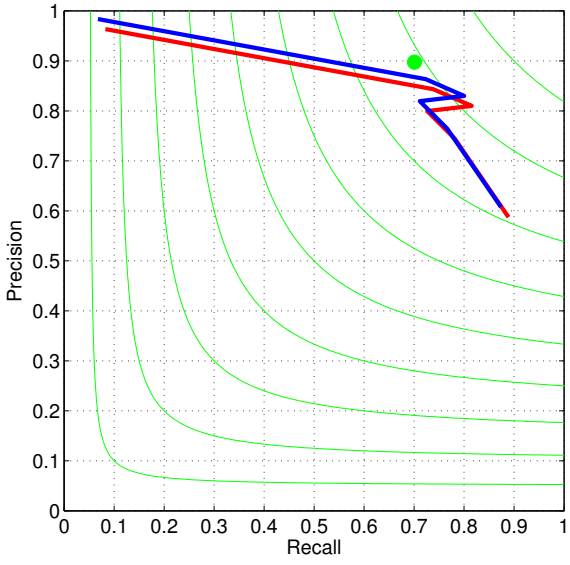


Figure 2.7: **F-measure plot**: evaluation of KSC (red) and SKSC (blue) algorithms with respect to human boundaries in the image segmentation task. Iso-F curves are shown in green. The green dot indicates the average agreement between the human subjects who performed the segmentations used as ground-truth.

Image ID	KSC	SKSC
145086	3.11	**3.10**
42049	3.42	**3.41**
167062	1.84	**1.69**
147091	1.10	1.10
196073	0.18	0.18
62096	0.35	**0.31**
101085	2.77	**2.69**
69015	2.36	2.36
119082	0.90	**0.89**
3096	2.97	2.97

Table 2.4: **VI criterion**: comparison of KSC and SKSC segmentations in terms of variation of information (the lower the better). Best performance in bold. The algorithms have a similar performance.

Image ID	KSC	SKSC
145086	0.59	0.59
42049	0.70	**0.71**
167062	**0.83**	0.77
147091	0.83	0.83
196073	0.98	0.98
62096	**0.95**	0.94
101085	0.37	**0.38**
69015	0.58	0.58
119082	0.87	0.87
3096	0.19	0.19

Table 2.5: **PRI criterion**: comparison of KSC and SKSC segmentations in terms of probabilistic rand index (the higher the better). Best performance in bold. In general, the methods produce a similar outcome.

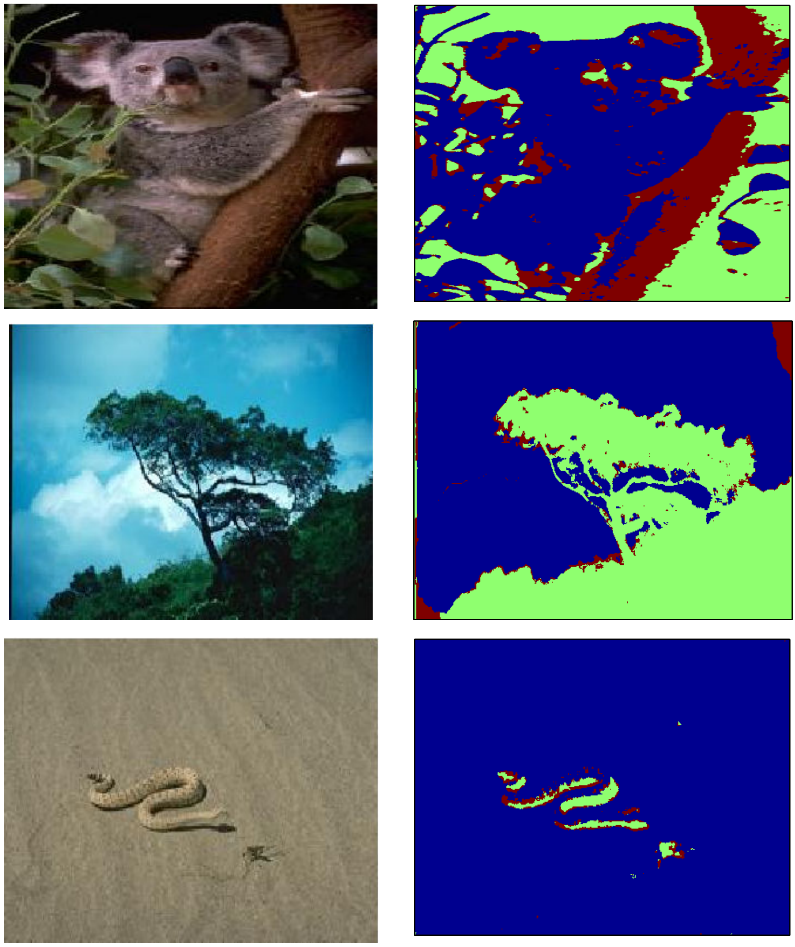


Figure 2.8: **SKSC segmentations**: original images (left) and SKSC segmentations (right) for image IDs 69015, 147091, 196073.

Chapter 3

Community Detection in Complex Networks

Community detection refers to the problem of partitioning a complex network into clusters of nodes with high density of edges, in order to understand its structure and function. A profusion of algorithms has been proposed in the recent past since many real-world applications arise in different fields, from biology to computational social sciences. In this chapter we propose a whole kernel-based framework for community detection. The main steps of our approach are the extraction of a subgraph representative of the community structure of the entire network for training the clustering model, the validation stage by means of proper model selection criteria, and the membership assignments for the remainder of the nodes via out-of-sample extension. In contrast to most of the methods present in the literature which are rather specific, our technique is more flexible because in the model selection phase the user can provide the desired criterion, in order to obtain a final partitioning with certain characteristics. Moreover, the out-of-sample extension allows to readily assign the membership to new nodes joining the network without using heuristics. Also we can deal with weighted and unweighted networks by choosing the suitable kernel function, and graphs with overlapping communities can be analysed. Finally, by exploiting the high sparsity of the majority of the real graphs, the method can scale to large network data even on a desktop computer and can be easily parallelized.

3.1 Related work

In nature, science and technology complex systems are usually organized in networks of interactions between many components. Although every system has its peculiar properties, a common architecture can be recognized [123]. It seems that complex networks are organized in modules or communities, and this feature can be explained by considering two important aspects: resilience and adaptation. Many studies suggested that if a part of a complex system stops working properly, the modular organization allows to circumvent the problem, making it more resilient to failures. Moreover, the community structure helps such a system to adapt quickly to changes in the environment [31].

A network (or graph) with N nodes is characterized by a community structure when groups of nodes forming tightly connected units are only weakly linked to each other. Uncovering the natural communities present in a complex network has recently become a major topic in science and engineering with an interdisciplinary effort [42, 106, 30, 9, 116]. This interest is due to the fact that revealing the modular structure of a network can be very useful for a number of reasons like visualizing large graphs composed of millions of nodes and edges, investigating the function of each module, discovering the role of the different nodes inside a community.

Many algorithms to handle the community detection problem have been proposed, with different characteristics. The most popular algorithm is that introduced in [47]. The method is historically important because it popularized community detection among physicists and engineers and favoured the search of new algorithmic solutions. It is based on the removal of edges with high betweenness, which expresses the participation of an edge to the information flow on a graph. The connected components of the remaining network are the communities detected by the method. Even if quite appealing the algorithm is slow (it scales as $O(N^3)$) and cannot deal with overlapping communities, as each vertex is assigned to a single cluster. A large number of successful techniques is based on Modularity maximization, which describes the strength of the community structure by comparing the actual graph with a random graph sharing the same degree distribution [105]. A well-known algorithm (with linear run-time $O(N)$) is the Louvain method [19], based on a greedy optimization of the Modularity. The Modularity maxima found by the method are also better than those found with another well-known greedy technique used in [30]. The most popular algorithm for discovering overlapping communities is the Clique Percolation Method (CPM) [112], which is based on the idea that nodes inside a community are likely to form cliques, in contrast to inter-community edges. Another recent and very fast technique for overlapping community detection is [140], built on a novel and counter-current observation that in many networks overlaps between communities are densely connected and can reveal the modules. In [41] the authors use a partition of the links of a network (instead of clustering the nodes) to uncover its

community structure. In this way their method can better detect nodes belonging to more than one community. A different class of methods are those based on information-theoretic ideas, such as the minimum-description-length method called Infomap introduced in [118]. All the above-mentioned algorithms can discover clusters at a particular scale, and even if they produce hierarchical results there is no guarantee that the relevant partitions are not artefacts. On the other hand multi-resolution techniques able to identify the different levels of organization of vertices in clusters have been proposed. They are inspired by different principles. For example in [117] and [9] the authors use different definitions of Modularity and the Laplacian matrix respectively, the algorithm introduced in [36] is developed on the stability of clustering under random walk, in [72] a fitness function which estimates the strength of a cluster and entails a resolution parameter is optimized.

In this context, also spectral algorithms play an important role. Although most spectral methods have been focused on data clustering, many applications to network data for community detection exist [38, 61, 24]. As for data clustering some issues arise when using spectral clustering for community detection:

- the construction and storage of the $N \times N$ adjacency matrix describing the graph becomes infeasible for large networks
- the computation of the eigenvectors of the Laplacian is computationally expensive $O(N^3)$, although some approximation techniques like the Lanczos method [74], the Nyström algorithm [44] and the power iteration method [87] can speed-up the process
- it is not clear how to choose the number of communities into which to split the network at hand
- it does not incorporate a clear extension to out-of-sample nodes. However, numerical techniques like the Nyström method [44], or incremental algorithms like [110, 37] provide a possible way to tackle this issue. Recently, also a sparse spectral clustering method based on the Incomplete Cholesky Decomposition (ICD) gives the possibility to perform out-of-sample extension by exploiting the pivots selected by the ICD [45]. Besides spectral clustering, most of the algorithms for community detection work only at the training level. They produce a partition of the network into modules but cannot assign the membership to new nodes joining the network without running from the beginning or using heuristics. In the community detection field, to the best of our knowledge the only kind of algorithm providing a systematic out-of-sample extension to test nodes are those one based on statistical inference by means of block models [125, 111, 3].

In what follows we describe a methodology that casts the community detection problem in a machine learning algorithm. Our core model is KSC [6], which has been summarized in the previous chapter. In this way we are able to handle the aforementioned issues in a unique framework, and to propose a competitive technique for the analysis of complex networks. Moreover, this work paves the way to the development of the MKSC model for the analysis of evolving networks discussed in chapter 4.

3.2 Methods

The usage of KSC for community detection can be summarized in three main stages:

- subset extraction: in order to solve an affordable eigenvalue problem for training the model, we need to select a subset of the entire network that captures the inherent community structure[1]
- model selection: the proper number of communities and the kernel hyper-parameters must be tuned carefully to ensure good results
- out-of-sample extension: once constructed an optimal model, the memberships for the remainder of the network can be assigned via eq. (2.16).

Furthermore, the choice of an appropriate kernel function which depicts the similarity between the nodes is very important.

3.2.1 Representative Sub-graph Extraction

Sampling a sub-graph representative of the community structure of the whole network under study is a crucial task in our framework, since it allows a meaningful out-of-sample extension to nodes not present in the training set. Simply taking a random sample of nodes can lead to very different results in several runs, since the quality of the selected sub-graph can have a huge variability. Also selecting a subset in such a way that it follows the same degree distribution or betweenness centrality distribution of the whole graph can produce samples that are not representative of the community structure of the larger network. Recently, a new quality function describing the representativeness of the sample with respect to the community structure of the whole

[1] How to choose such a subgraph is an open problem. In general it is selected by experimental evaluation, where one wants to find a trade-off between a small size (to reduce the computational burden) and a good quality of the clustering results on the entire network (which depends on how well the subset represents the community structure of the whole graph).

graph has been introduced in [90]. This quality function is called Expansion Factor (EF) and is defined as $\frac{|N(\mathcal{A}_G)|}{|\mathcal{A}_G|}$, where $\mathcal{A}_G$ indicates a subset of the graph $\mathcal{G}$, $N(\mathcal{A}_G)$ its neighbourhood, i.e. the remaining part of the network to which $\mathcal{A}_G$ is connected, and $|\cdot|$ denotes the cardinality of a set. The idea is that by selecting a sub-graph for which the expansion factor is maximum, one samples a representative subset. Roughly speaking, by including in $\mathcal{A}_G$ nodes that best contribute to the expansion factor, we are taking nodes that are more connected to the rest of the network than to $\mathcal{A}_G$. These nodes are then very likely to belong to clusters not yet represented in the sub-graph, allowing us to produce a sample which is a condensed representation of the community structure of the whole network.

In [79] we proposed a greedy strategy for the optimization of EF, which can be summarized in algorithm 5. The selection of the active subset can take from a few to several minutes or hours depending on the size N of the entire network and its sparsity, the chosen size N_A for the active subset and the threshold ϵ. Moreover, although the variability of the method is less compared to the random sampling, the technique is stochastic by nature. In [94] the Fast and Unique Representative Subset (FURS) selection technique has been proposed to cope with these issues. The method has been shown to be computationally less expensive and deterministic: it greedily selects nodes with high-degree centrality from most or all the modules present in the network, which are usually located at the center rather than the periphery and can better capture the community structure.

Algorithm 5: EF-based subset selection [79]

Data: network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ composed of N nodes $\mathcal{V} = \{n_i\}_{i=1}^{N}$ (represented as the adjacency matrix $A \in \mathbb{R}^{N \times N}$), N_A (size of sub-graph $\mathcal{A}_G$).
Result: active set of N_A selected nodes belonging to the subset $\mathcal{A}_G = (\mathcal{V}_A, \mathcal{E}_A)$.

```
select randomly an initial sub-graph A_G = {n_j}_{j=1}^{N_A} ⊂ V
while δ(EF) > ε do
    compute EF(A_G)
    randomly pick two nodes as n_* ∈ V_A and n_+ ∈ V \ V_A
    let {W = V_A \ {n_*}} ∪ {n_+}
    if EF(W) > EF(A_G) then
        swap({n_+},{n_*})
    else
        do not swap({n_+},{n_*})
    end
end
```

3.2.2 Model Selection Criteria

Often people use heuristics to select the tuning parameters present in their models. Since model selection is a crucial point, here we describe a systematic way to do it properly. Our method is based on a validation procedure. We train the KSC model with different number of communities and (where needed) several values of the kernel hyper-parameters. In the validation step the obtained groupings are judged depending on some quality functions like BLF [6], AMS [81] (described in the previous chapter), Modularity [76] (see Appendix): the one (or more) partition with the highest value of the chosen criterion is selected.

For the model selection algorithm using AMS the reader can refer to section 2.3.3, while the tuning scheme performed by means of Modularity can briefly be expressed by algorithm 6. The training set, validation set and the two stages of the validation

Algorithm 6: Modularity-based model selection for KSC [76]

Data: training set, validation set stage I, validation set stage II, positive (semi-) definite kernel function $K(x_i, x_j)$

Result: selected number of clusters k and (if any) kernel hyper-parameters.

1 compute cluster indicator matrix X from the cluster results of the different models, obtained using the training set and the validation set I stage in the learning process

2 compute the Modularity matrix $M = S - \frac{dd^T}{2m}$, where all the quantities refer to the validation set used in the II stage of the validation process

3 compute the Modularity Mod $= \frac{1}{2m}\text{tr}(X^T M X)$

4 select the model (i.e. k and the eventually the kernel hyper-parameters) corresponding to the partition(s) which gives the highest Modularity value.

process have the following meaning. The training set is the matrix given as input to the kernel spectral clustering model during the training phase. The validation process can be divided into two stages:

1. stage I: the cluster memberships for the validation set (data not belonging to the training set) are predicted by the model based on eq. (2.16)
2. stage II: the quality of the predicted memberships are judged by means of a certain criterion.

In these two stages the validation sets involve the same data (the nodes of the graph under study) but represented in different ways. In stage I some rows of the adjacency matrix are considered: this is called an adjacency list. In stage II, if we consider the Modularity criterion, the validation set is an adjacency matrix, because this is needed in order to calculate the related Modularity. For BLF and AMS this is not the case (see figure 3.1 for a further clarification).

Finally we can point out that the definition of the Modularity function is general because it does not make any assumption on the kind of the community structure of the network to detect. This, together with the fact that it is a quality function particularly suited for graphs, is one of the main advantages with respect to the BLF criterion, which as we already mentioned in section 2.2.4 is optimized to detect clusters that are well separated. Moreover, the Modularity-based model selection algorithm is feasible for large scale applications because a sparse representation of the validation adjacency matrix can be used.

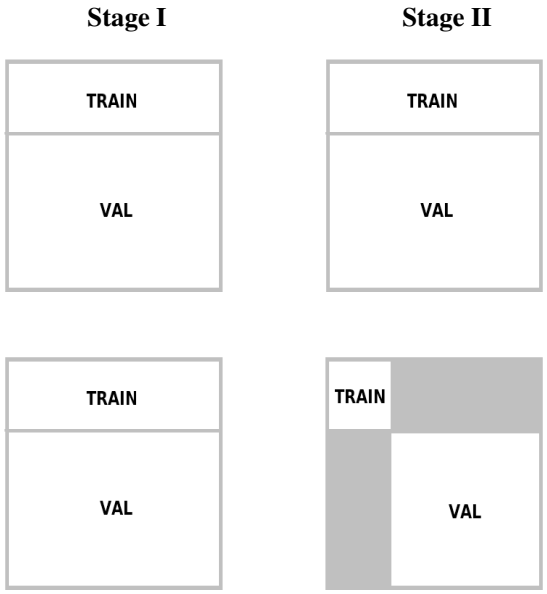


Figure 3.1: **Training and validation sets**. Example showing the way the datasets for the study of a network via KSC are built up. In this specific case the first 25% of the total nodes form the training set and the remaining 75% the validation set. The first row refers to BLF or AMS, the second to Modularity. The first column represents the first stage of the validation process (prediction of memberships), the second column depicts the second stage. For Modularity, in the second stage of the validation process we change representation from adjacency list to adjacency matrix.

3.2.3 Choice of the Kernel Function

Unlike classical spectral clustering, when using KSC or SKSC for community detection we need to build a graph over the initial network to represent its topology in

the feature space. In this context, the choice of a proper kernel function is fundamental to be able to unveil the modular organization of the network.

In the analysis of the weighted networks we use the RBF kernel to capture the similarity within the nodes. We treat the i-th row of the adjacency list of the entire graph representing node v_i as a point in an Euclidean space of dimension N, say x_i. Then the similarity between two nodes is given by $K(x_i, x_j) = \exp(-||x_i - x_j||_2^2/\sigma^2)$, where the hyper-parameter σ denotes the kernel bandwidth.

In dealing with unweighted networks a recently proposed kernel function named community kernel [62] is used to build up the similarity matrix of the graph. This kernel function does not have any parameter to tune and the similarity Ω_{ij} between two nodes v_i and v_j is defined as the number of edges connecting the common neighbours of these two nodes: $\Omega_{ij} = \sum_{k,l \in \mathcal{N}_{ij}} A_{kl}$. Here $\mathcal{N}_{ij}$ is the set of the common neighbours of nodes v_i and v_j, A indicates the adjacency matrix of the graph, Ω is the kernel matrix. As a consequence, even if two nodes are not directly connected to each other, if they share many common neighbours their similarity Ω_{ij} will be set to a large value. Moreover, in [62] it is empirically observed that this kernel matrix is positive definite, a fundamental requirement in order to use the community kernel in our kernel-based framework.

The above-mentioned kernel functions are able to correctly capture the similarity between the nodes of weighted and unweighted networks. Unfortunately, they are not computationally efficient. In order to be able to perform in a reasonable time the computation of the kernel matrix for large graphs, it is advisable to use the cosine kernel, since it works with a sparse representation of the variables [96].

3.2.4 Computational Complexity

Since we can use the Lanczos method [74] or the Incomplete Cholesky Decomposition (ICD) [4] to compute the top k eigenvectors of eq. (2.13) in quadratic run-time[2], the computational complexity for the training phase is dominated by the time required to construct the kernel matrix Ω. Since the networks are usually very sparse, the latter is given by $O(N_{\text{Tr}}^2)$. If we consider the whole network as test set, the run-time of the test phase concerns the computation of eq. (2.16) and equals $O(N_{\text{Tr}}N)$. Moreover, the out-of-sample extension can be easily scaled down of a big factor in a distributed environment. In fact, by chunking the test set into blocks with size proportional to the training set and parallelizing the computations, the overall complexity is reduced to $O(2N_{\text{Tr}}^2)$. Finally, in figure 3.2 the computational burden required by KSC is evaluated on some of the synthetic networks described in the next section. Here the complexity seems quadratic and not linear according to a least squares curve fitting: this can be

[2]Quadratic with respect to the number of training points or the pivots of the ICD respectively.

explained by the fact that a non-sparse representation of the variables and an inefficient implementation of the kernel function have been used in the simulations.

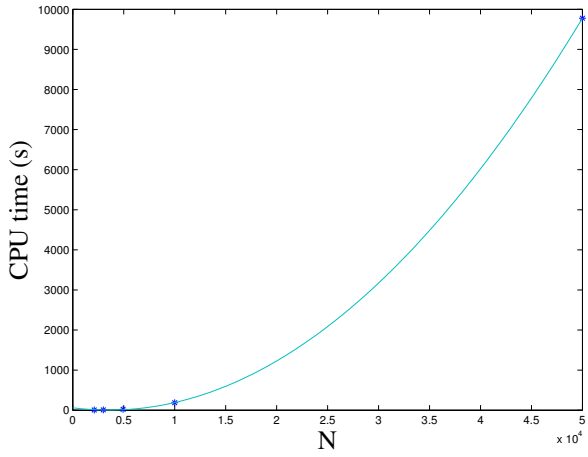


Figure 3.2: **Computational complexity KSC** Run-time required to obtain the partitioning of some computer generated networks with different size. The curve represents a second degree polynomial yielding the best fit to the simulation data.

3.3 Simulations on Synthetic Networks

Methods to detect communities in graphs are required to be deeply tested. To do that, one needs benchmark graphs with a known community structure that the algorithms should identify. In [69, 73] the authors proposed new classes of benchmark graphs referred as LFR where the distributions of node degree and community size are both power laws, which mimics an important feature of real networks. By means of the LFR benchmark it is possible to generate undirected, unweighted, directed, weighted and hierarchical graphs with overlapping nodes and different degree of mixing between communities. Thanks to its generality, it quickly became a standard tool for testing community detection methods, and for this reason we used it in all the experiments described in this dissertation.

In table 3.1 a description of all the artificial graphs used to test KSC and SKSC is given. For instance *Net_unw3C1000ov* indicates an unweighted network formed by 3 communities with 1000 overlapping nodes. For illustration purposes, in figure 3.3 the synthetic networks with $1\,000$ nodes and 4 communities with mixing parameters equal to 0.05 (small overlap), 0.2 (medium overlap), 0.35 (large overlap) are illustrated

by means of the corresponding adjacency matrix. We can recognize the block configuration, indicating the presence of an underlying community structure.

In most of the experiments the training set is usually composed by the 15% of the nodes of the starting graph extracted by using algorithm 5, and it is randomized 10 times. How to chose in practice the size of the maximal subgraph is an important aspect. On one hand one would like to select a very small subset in order to save computing time. On the other hand the subgraph should be large enough to accurately represent the community structure of the whole graph. By experimental evaluations (like the one depicted in figure 3.6) we set the size of the training subgraph to 15% of the size of the entire network. Surprisingly, the same percentage is used in [85], where the optimal subgraph size is determined based on its representativeness in terms of graph parameters like degree distribution, clustering coefficient distribution etc.

We consider as test set the entire network. The model selection results are shown by means of boxplots or only the average values, and usually the validation set is formed by the 35% of the starting graph. Moreover, for big network data, the training and validation sets size are constrained by the maximum size of the kernel matrix that can be stored in the memory of our PC, which is 5000×5000.

When evaluating the quality of the final partitions, for the synthetic networks we compare KSC or SKSC with spectral clustering using the Nyström method for out-of-sample extension. In case of the real graphs, the results obtained with KSC or SKSC depending on the data-sets are reported.

Figure 3.4 depicts a typical example of the greedy optimization of the EF performed by the algorithm explained in section 3.2.1, where the artificial network *Net_unw9C* is considered. Moreover, in figure 3.6 it is shown how the active sampling technique produces a better sample than random sampling. To see this, we compare the ARI index [58] (see appendix) between the partitions predicted by the kernel spectral clustering model and the true memberships, by using the training set selected randomly or actively. In figure 3.5 we can notice that the degree and betweenness centrality distribution [46] of the active set is quite different from those one of the whole graph. All these empirical observations are in agreement with what has been discussed in [90]. In this work the authors observed that selecting a sub-graph which captures the community structure of the entire graph often does not implies that the sub-graph is also representative of the same degree or betweenness centrality distribution. On the other hand, if we compare the degree distribution associated to the full kernel matrix Ω and the out-of-sample degree distribution related to the test kernel matrix Ω_{test}, they are quite similar (see figure 3.7). This indicates that the active selected subset is meaningful and allows our model to correctly generalize to unseen nodes.

Regarding the model selection issue, we compare the BLF, Modularity and AMS

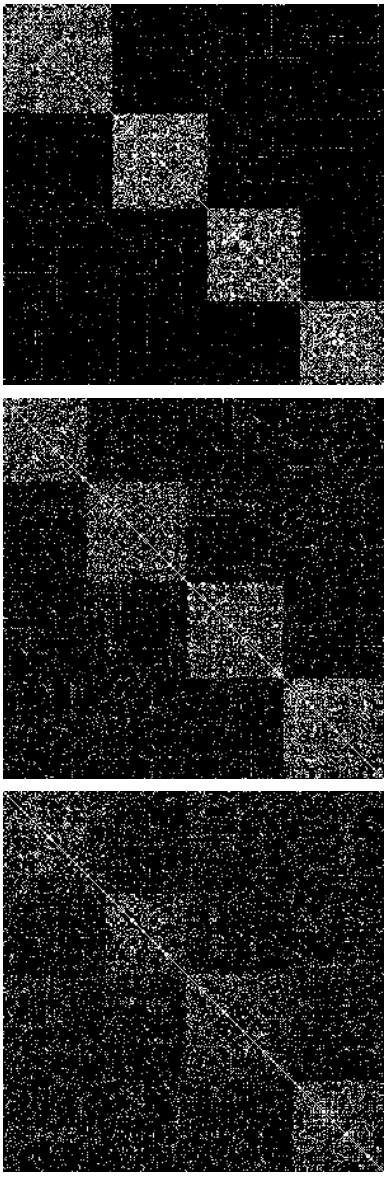


Figure 3.3: **Artificial networks Net_unw4C**. Adjacency matrix of the unweighted synthetic networks composed by 4 communities with small overlap (top), medium overlap (center) and large overlap (bottom).

Network	Nodes	Edges	Sparsity (%)	Overlap (μ)
Net_unw9C	3 000	22 904	99.49	0.1
Net_unw13C	10 000	76 789	99.85	0.1
Net_unw22C	50 000	383 220	99.97	0.1
Net_unw3C1000ov	3 000	149 535	98.34	0.1
Net_unw4CmuS	1 000	11 867	98.81	0.05
Net_unw4CmuM	1 000	13 451	98.65	0.2
Net_unw4CmuL	1 000	14 223	98.57	0.35
Net_w6C	3 000	148 928	99.35	0.1
Net_w4C1000ov	3 000	149 033	98.34	0.1

Table 3.1: **Synthetic network data**. Summary of some properties of the computer generated graphs that have been analysed. In the name of the data-sets, the acronym *unw* means unweighted, *w* stands for weighted, *ov* indicates the presence of overlapping nodes, *C* means communities. Sparsity refers to the adjacency matrix associated to each graph and indicates the percentage of zero entries with respect the total number of elements of the matrix. The mixing parameter μ is related to the number of inter-community edges and describes the degree of overlap between the communities. The acronym *muS* means small overlap, *muM* medium overlap, *muL* large overlap.

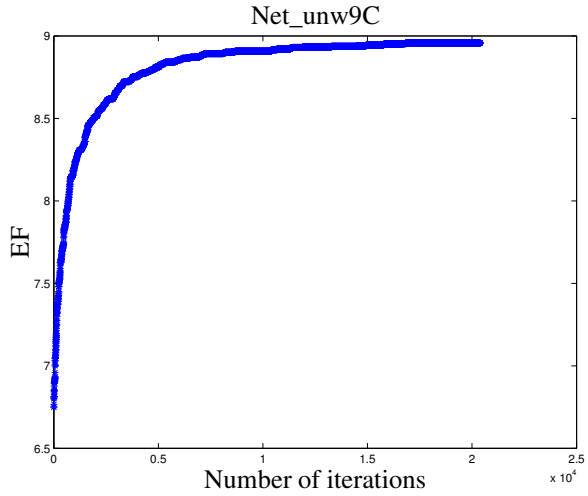


Figure 3.4: **Expansion factor optimization**. Example of the greedy optimization of the expansion factor (EF) for extracting a sub-graph from the artificial network *Net_unw9C*.

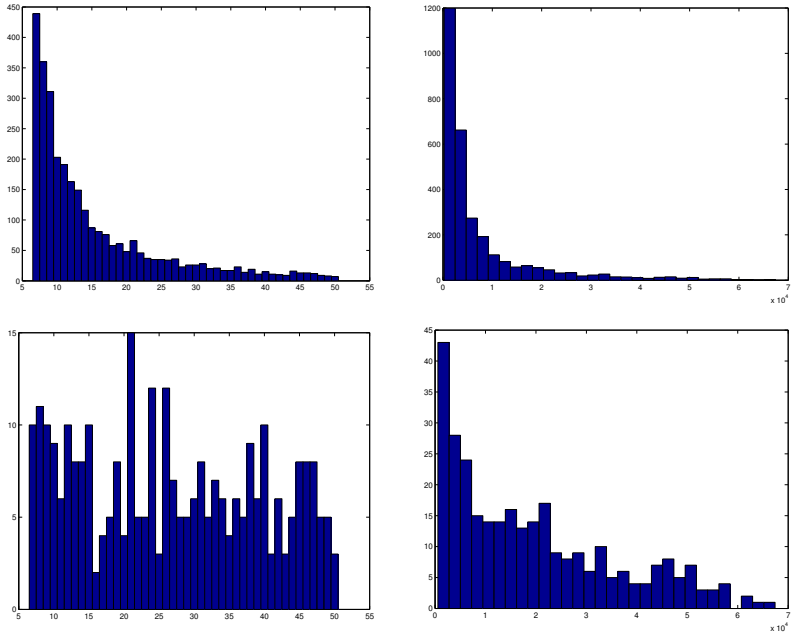


Figure 3.5: **Properties EF sub-graph**. Degree and betweenness centrality distribution (number of nodes) of the synthetic network *Net_unw9C* (top left and top right) and of a typical active set selected from the EF algorithm (bottom left and bottom right). We can notice that the representativeness of the set in terms of community structure can not be related to its representativeness in terms of degree and betweenness centrality distribution.

criteria for some of the synthetic networks reported in table 3.1. Roughly speaking, the main observation we can draw is that in the most difficult and more realistic cases where a certain overlap between the communities is present, Modularity and AMS outperform BLF. For instance in the analysis of *Net_unw3C1000ov* depicted in figure 3.9, from the boxplot it could seem that the BLF is slightly more likely to detect 3 communities rather than 4 (less variability). However, by viewing the results from another perspective, we can be convinced that this is not true. In fact in the space of the latent variables, as we already mentioned in section 2.2.4, in the ideal case of perfectly separated clusters every line represents a different community. In figure 3.10 we compare the line structure in this space for 3 and 4 clusters, showing as in the latter case the line structure is more clear. This is an indication that the BLF criterion is probably detecting 4 clusters rather than 3. Also in figure 3.11 we can observe

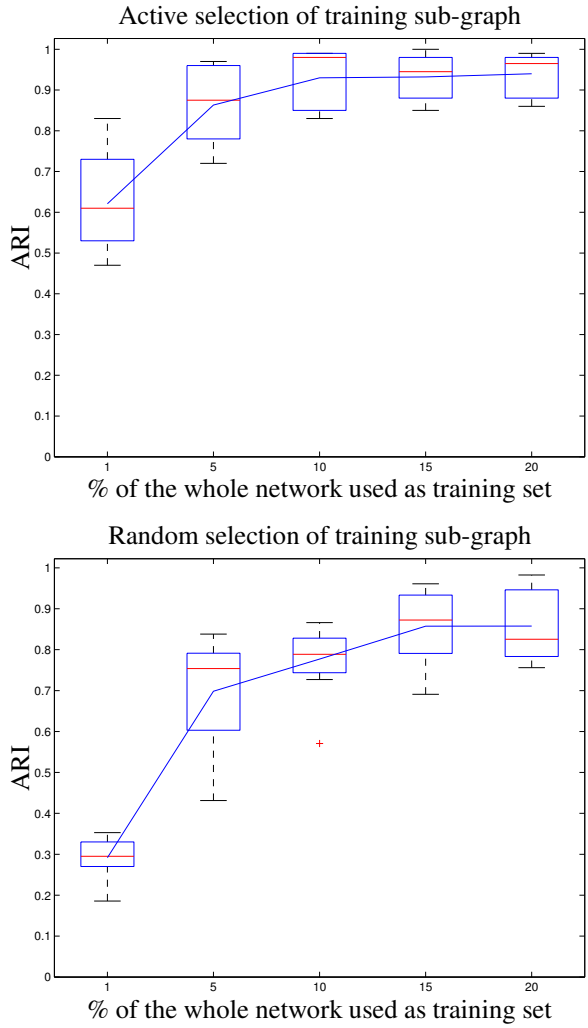


Figure 3.6: **Quality EF sub-graph**. The quality of the sub-graph extracted from the artificial graph *Net_unw9C* using the EF algorithm is assessed (top). The ARI between the predicted partition where the EF sub-graph is used as training set and the true partition is computed. A comparison with random sampling is performed (bottom). We can notice that, mostly when few nodes are selected, the EF sub-graph allows a more accurate out-of-sample extension. Moreover the variability is reduced when more nodes are considered.

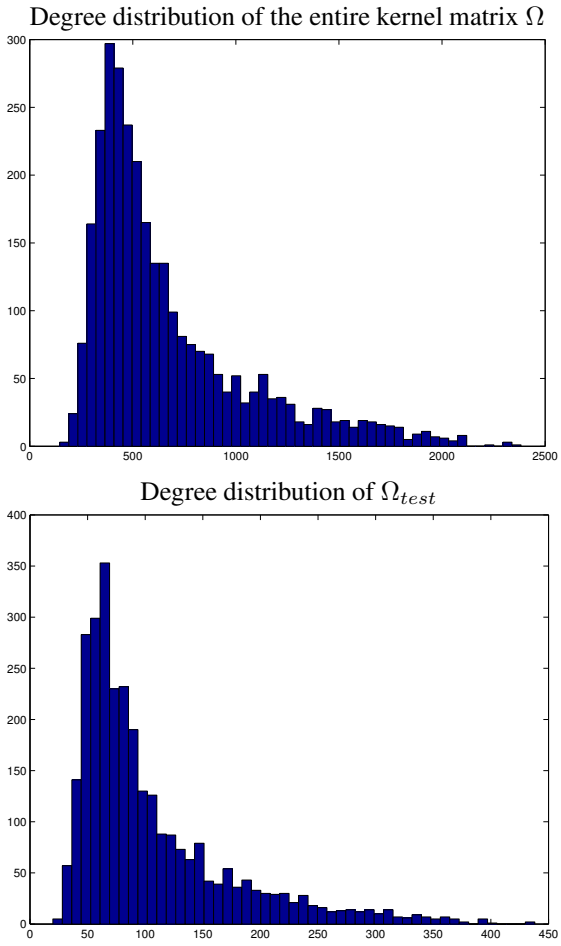


Figure 3.7: **Generalization with the EF sub-graph**. The degree distribution related to the full kernel matrix (of size $3\,000 \times 3\,000$) describing the similarity between the nodes of the network *Net_unw9C* is pictured in the first row. In the second row the degree distribution associated to the test kernel matrix (of size $3\,000 \times 300$) is shown. The two distributions look like quite similar, meaning that the KSC model trained on the active selected subset can generalize well in the test phase.

that concerning the networks *Net_unw4CmuS*, *Net_unw4CmuM* and *Net_unw4CmuL* the BLF criterion works only in case of small overlap, while Modularity and AMS criteria correctly identify 4 communities in the medium overlap case. Moreover, BLF and Modularity fail in case of large overlap, while AMS detects 3 communities, giving then an useful indication of the community structure to the user. Finally, the model selection results related to the weighted networks are depicted in figures 3.12 and 3.13. In both cases AMS suggests a number of clusters closer to the true community structure.

In table 3.2 we evaluate the ability of KSC and SKSC of correctly partitioning the computer generated networks described in table 3.1. Average values of Modularity, Conductance and ARI have been used as cluster quality measures. A comparison with the spectral clustering using Nyström approximation for the out-of-sample extension named SC-Nyström is performed. The Nyström method is a technique for finding numerical approximations of eigenfunctions. It has been proposed in [44] to reduce the computational burden in spectral clustering eigenvalue problems. In fact, it allows one to extrapolate the complete grouping solution using only a small subset extracted randomly from the whole dataset to partition. From table 3.2 we can notice that the kernel spectral clustering approach in general performs better than SC-Nyström for most of the graphs and is faster for larger networks.

3.4 Real-World Applications

In order to test KSC and SKSC algorithms on real-life data the following graphs have been used:

- **Yeast_pro**: interaction network data for yeast formed by $2\,114$ nodes and $4\,480$ edges. As explained in [13] proteins can have direct or indirect interactions with one another. Indirect interaction refers to being a member of the same functional module but without directly binding to one another. In contrast, direct interaction, refers to two amino acid chains that bind to each other. Obviously, many of these interactions reflect the dynamic state of the cell and are present or absent depending on the particular environment or developmental status of the cell.
- **Power grid**: the network of Western USA power grid [137] formed by $4\,941$ nodes and $6\,594$ edges. The vertices represent generators, transformers and substations, and edges represent high voltage transmission lines between them.
- **Karate**: the Zachary's karate club network [142] consists of 34 member nodes, and splits in two smaller clubs after a dispute arose during the course of Zachary's study between the administrator and the instructor.

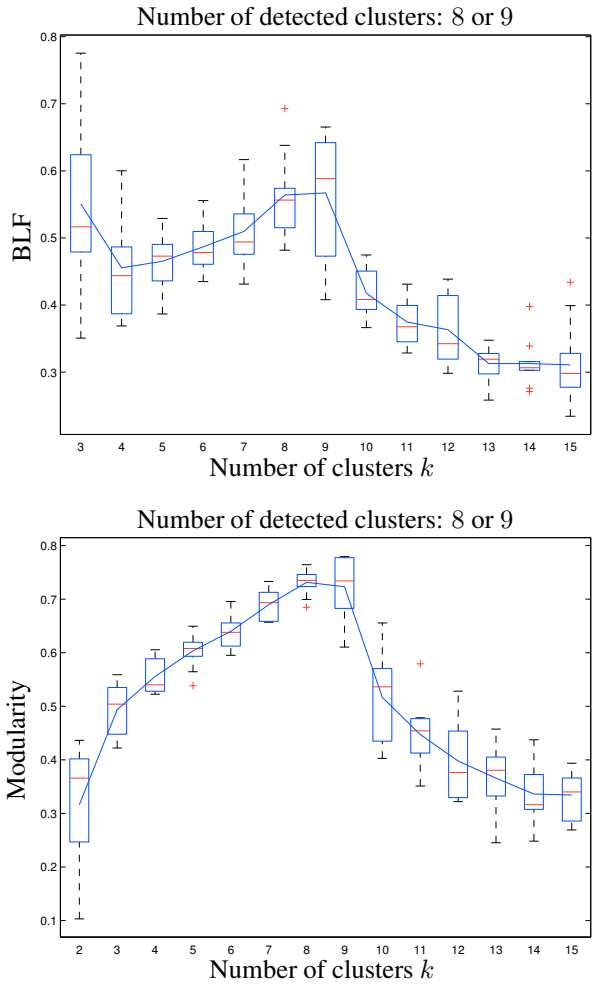


Figure 3.8: **Model selection Net_unw9C**. Modularity and BLF criteria are compared in terms of their ability of detecting the right number of communities, which is 9 in this case.

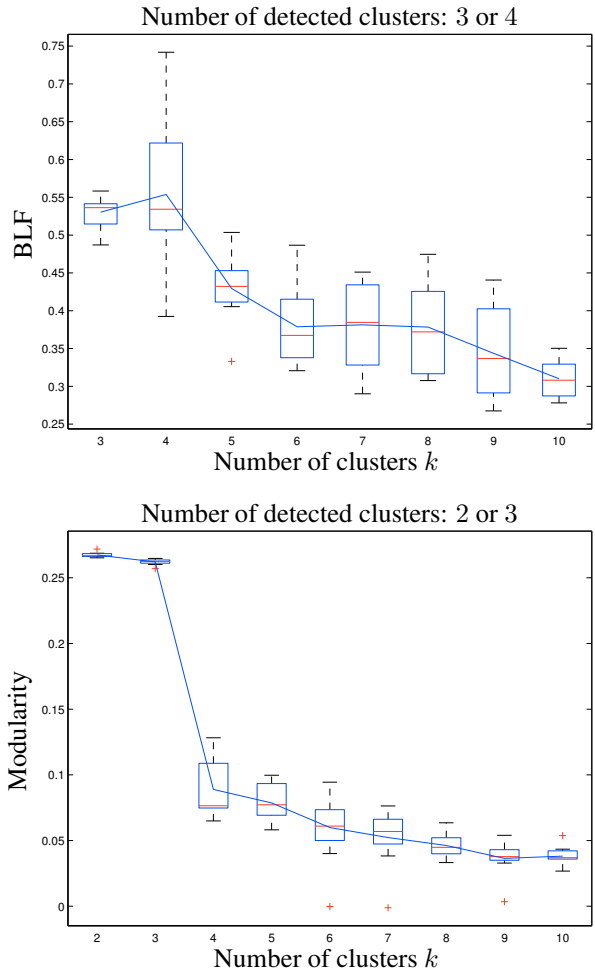
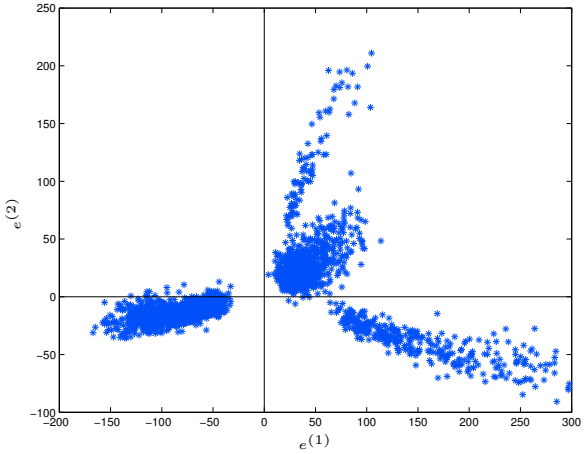
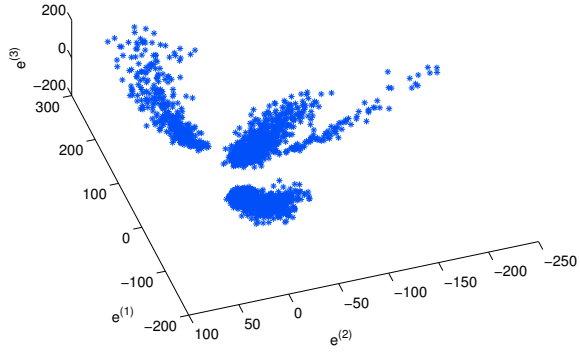


Figure 3.9: **Model selection *Net_unw3C1000ov***. It is interesting to notice that in this case BLF also suggests to select 4 clusters. On the other hand Modularity criterion correctly identifies the possible presence of 3 clusters without falling in this confusion.



(a) 3 clusters



(b) 4 clusters

Figure 3.10: **Projections space *Net_unw3C1000ov***. Nodes represented in the space of the latent variables for 3 and 4 clusters. We can notice that in the case of 3 communities there are not 3 clear lines, while in the case of 4 communities the line structure is more evident. Probably the BLF criterion is considering the overlapping nodes as belonging to a separate community.

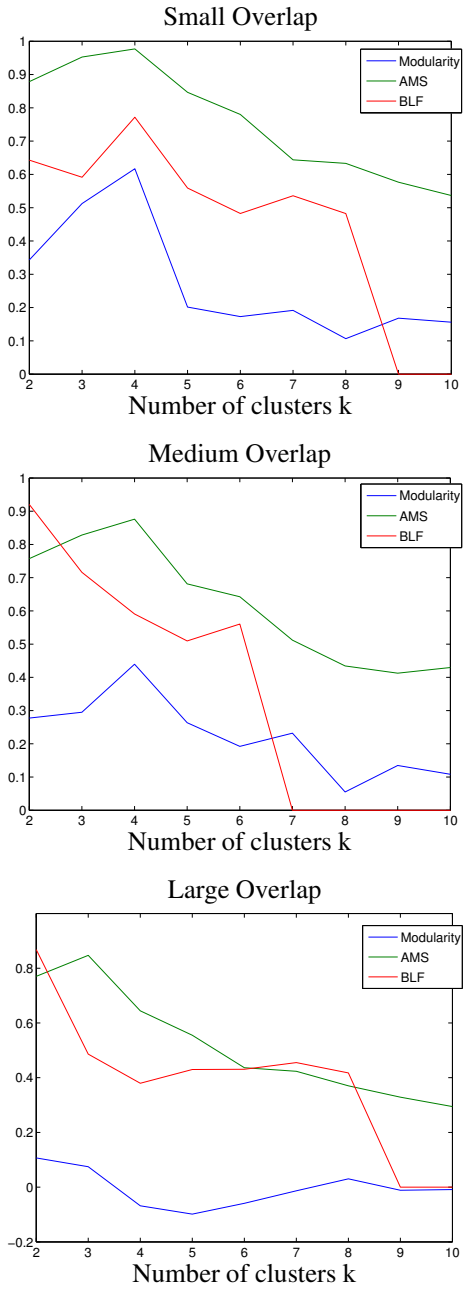


Figure 3.11: **Model selection for the synthetic network *Net_unw4C***. BLF (red), Modularity (blue) and AMS (green) are compared. The true number of communities is $k = 4$.

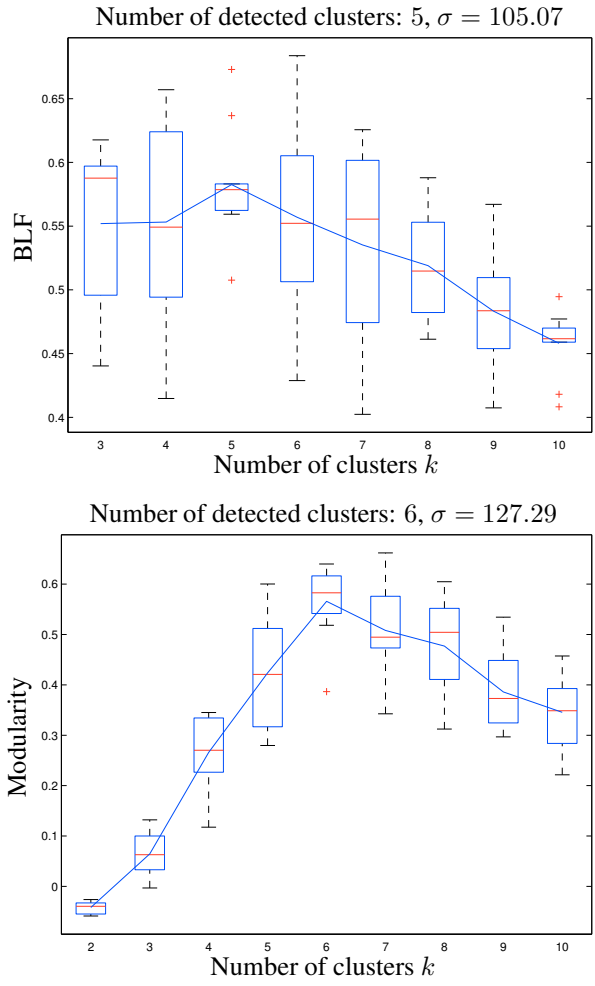


Figure 3.12: **Model selection network *Net_w6C***. BLF and Modularity are contrasted in relation to their ability of selecting the proper number of communities, which is 6. Moreover since we deal with a weighted network, we use the RBF kernel to describe the similarity between the nodes and its bandwidth σ must also be carefully tuned.

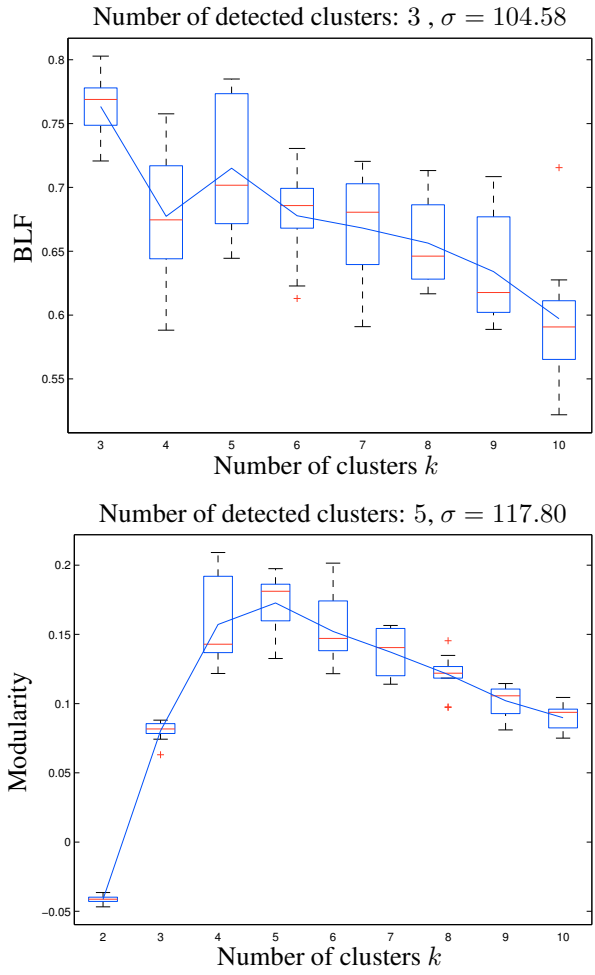


Figure 3.13: **Model selection network *Net_w4C1000ov***. BLF and Modularity are compared. Both criteria do not seem to be able to detect the 4 overlapping clusters. This is likely due to the not strong community structure testified by quite low values of Modularity corresponding to the various partitions.

Network	Algorithm	ARI	Mod	Cond	CPU time (s)
Net_unw9C	**KSC**	**0.92**	**0.75**	**0.0021**	8.28
	SC-Nyström	0.85	0.69	0.0025	**7.82**
Net_unw13C	**KSC**	**0.90**	**0.62**	**0.0015**	**187.03**
	SC-Nyström	0.87	0.56	0.0019	457.29
Net_unw22C	**KSC**	0.69	0.53	0.0008	**9778.57**
	SC-Nyström	**0.79**	**0.58**	0.0008	14 889.68
Net_unw3C1000ov	**KSC**	**0.55**	**0.27**	**0.0061**	8.49
	SC-Nyström	0.46	0.25	0.0064	**7.89**
Net_unw4CmuS	**SKSC**	**0.95**	**0.66**	**0.0039**	6.28
	SC-Nyström	0.89	0.58	0.0040	**5.71**
Net_unw4CmuM	**SKSC**	**0.91**	**0.50**	0.0042	6.27
	SC-Nyström	0.84	0.45	0.0042	**5.73**
Net_unw4CmuL	**SKSC**	**0.62**	**0.31**	0.0043	6.29
	SC-Nyström	0.58	0.26	0.0043	**5.74**
Net_w6C	**KSC**	**0.61**	**0.58**	**0.0019**	**4.33**
	SC-Nyström	0.56	0.55	0.0020	4.51
Net_w4C1000ov	**KSC**	**0.42**	**0.23**	0.0029	4.41
	SC-Nyström	0.39	0.21	0.0029	**4.34**

Table 3.2: **Synthetic networks results**. The performance of KSC, SKSC and spectral clustering with the Nyström method used for out-of-sample extension are reported. The results are evaluated according to several measures like ARI, Modularity, Conductance (see Appendix and references therein). Moreover also the time required for execution of the algorithms is illustrated. In the majority of the cases the kernel spectral clustering is more accurate and faster for bigger network data.

- **Football**: this network [47] describes American college football games and is formed by 115 nodes (the teams) and 616 edges (the games). It can be divided into 12 communities according to athletic conferences.
- **Dolphins**: the dataset, described in [89], depicts the associations between 62 dolphins with ties between dolphin pairs representing statistically significant frequent associations. As pointed out in [89], this network can be naturally split in two main groups (female-female and male-male associations), even if another cluster of mixed sex associations could also be considered.

These networks have been widely studied and represent standard real-world benchmarks for testing community detection algorithms, since for most of them the underlying community structure is known.

The model selection outcomes are depicted in figure 3.14. For the first two networks Modularity and BLF have been compared. Since BLF did not give any good indication we only show the results related to Modularity. The latter suggests the presence of 7 and 16 communities for the *Yeast_pro* and *Power grid* graphs respectively. Regarding *Karate*, *Football* and *Dolphins* networks AMS, BLF and Modularity model selection criteria have been contrasted. In the *Karate* network case all methods detect 2 communities. For what concerns the *Football* dataset BLF fails, while Modularity and AMS select 10 communities (AMS suggests also 3). Finally in the *Dolphins* network AMS selects 3 communities, Modularity 6 and BLF 2. In all these cases, given the limited size of the graphs, the training and validation sub-graphs have been chosen to be 70% and 30%.

In table 3.3 the partitions found by KSC or SKSC are evaluated according to Conductance, Modularity and ARI. We can notice how for all the networks the proposed kernel-based approach for community detection is able to discover a meaningful modular structure. Regarding the *Yeast_pro* graph, unlike the analysis performed in [13] we consider both direct and indirect interactions, for a total of $2\,114$ nodes instead of $1\,870$. We found 7 clusters, with the largest component containing about the 60% of the linked proteins (and all the proteins with indirect interactions). This outcome is shown in the bottom of figure 3.15 (only the directly interacting proteins are considered). The community structure found by our algorithm seems quite attractive for its simplicity, but needs further investigation in order to assess a meaningful biological interpretation of the discovered modules. The same considerations can be drawn for the Power grid network, which has been partitioned into 16 distinct communities as illustrated in the bottom of figure 3.16. Regarding the analysis of the smaller networks, more observations can be done by considering the moderated outputs produced by SKSC:

- Karate network: node 31 has the most certain membership, being assigned with probability 1 to its community. This is reasonable since it is linked to the highest degree nodes present in the same module
- Football network: node 25 and 59 are assigned to every cluster with nearly the same probability (i.e. they are overlapping nodes)
- Dolphins network: the membership of node 39 is equally shared between the 2 communities of the network.

However, these insights need to be further investigated in order to better interpret the results.

Finally, an exhaustive comparison between the proposed KSC-based method for community detection explained in this chapter and other state-of-the-art algorithms has been performed in [96]. In the cited work we contrasted our technique with the

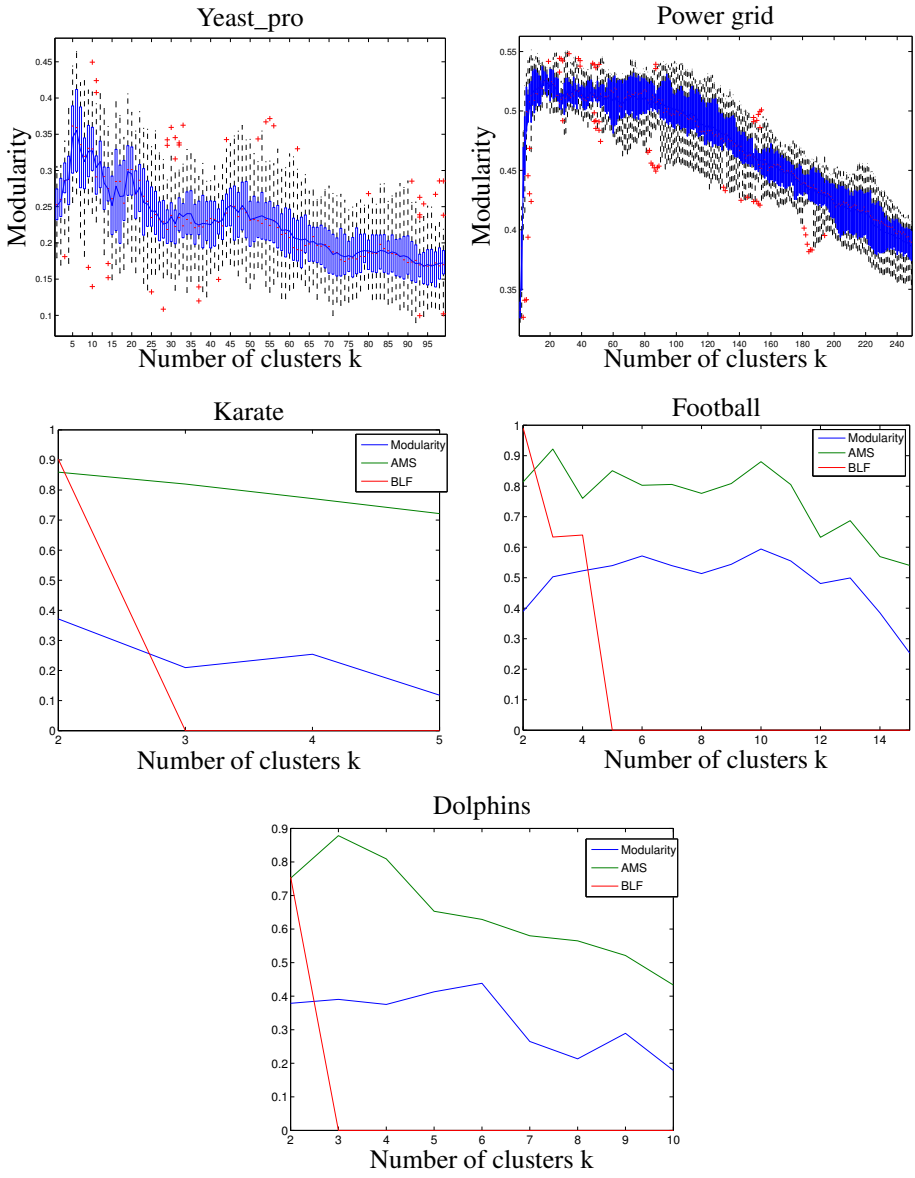


Figure 3.14: **Validation procedure for the real-world networks**. BLF, Modularity and AMS model selection criteria are compared.

Louvain method [19], Infomap [118] and CNM [30] on several synthetic and real-world graphs. In general, we observed that KSC produces a smaller number of clusters with better (i.e. lower) Conductance, while the Louvain method is the best w.r.t. the Modularity quality metric.

Network	Algorithm	Mod	Cond	ARI
Yeast_pro	**KSC**	0.35	0.029	–
Power grid	**KSC**	0.52	0.033	–
Karate	**SKSC**	0.37	0.083	1
Football	**SKSC**	0.58	0.76	0.77
Dolphins	**KSC**	0.35	0.13	0.91

Table 3.3: **Evaluation partitions of the real-world graphs**. The community structure detected by the proposed kernel- based approach (SKSC or KSC) is evaluated in terms of Modularity, Conductance and ARI (when a sort of ground-truth describing the real splitting is available).

3.5 Conclusions

This chapter focused on the community detection problem. The latter has been the subject of an extensive research in recent years, due to the large availability of network data coming from many sources. After discussing the literature in the field, we have introduced our approach to deal with community detection, which is based on the KSC and SKSC methods presented in the previous chapter. In particular, we have presented a model selection criterion based on the Modularity quality function, which is more suited for network data. Moreover, a sampling method based on the greedy maximization of the expansion factor is used to extract a small sub-graph representative of the community structure of the entire network. This sub-graph is used to train the KSC model, and in a second phase the out-of-sample extension property allows to predict the memberships of the remaining nodes. We showed how our technique produces a high-quality partitioning both in terms of Modularity and Conductance, and it outperforms spectral clustering using the Nyström method for out-of-sample extension. Finally, we discussed the computational complexity of our approach.

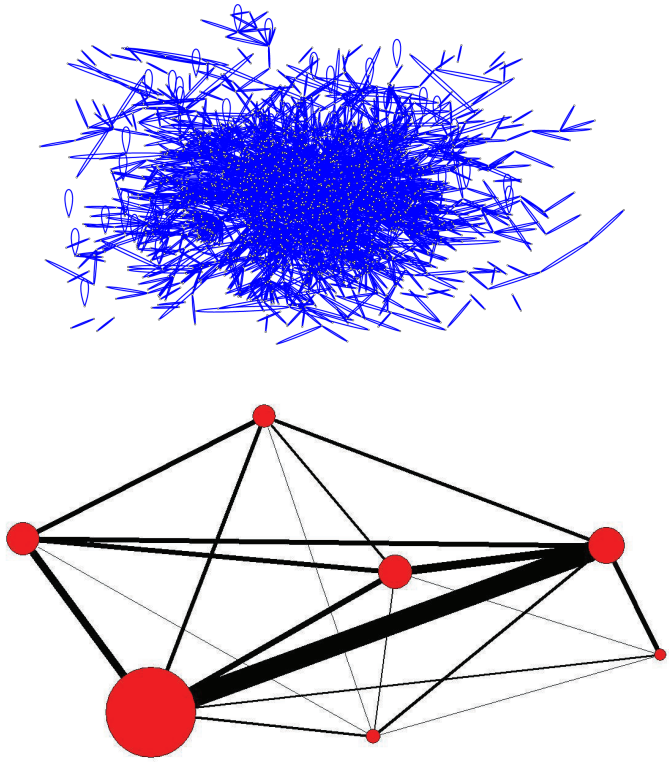


Figure 3.15: ***Yeast_pro* network visualization**. Original network (top) and partition discovered by KSC for the network of interacting proteins of yeast (bottom). Every red circle represents a cluster with size related to the number of nodes belonging to it. The position of the circles is not relevant. The edges are the links between nodes belonging to different communities, with thickness proportional to the number of these inter-community edges. The nodes and edges in each detected community, for simplicity, are not shown. The figure has been made by using the software for large network analysis `Pajek` (see *http://pajek.imfm.si/doku.php*).

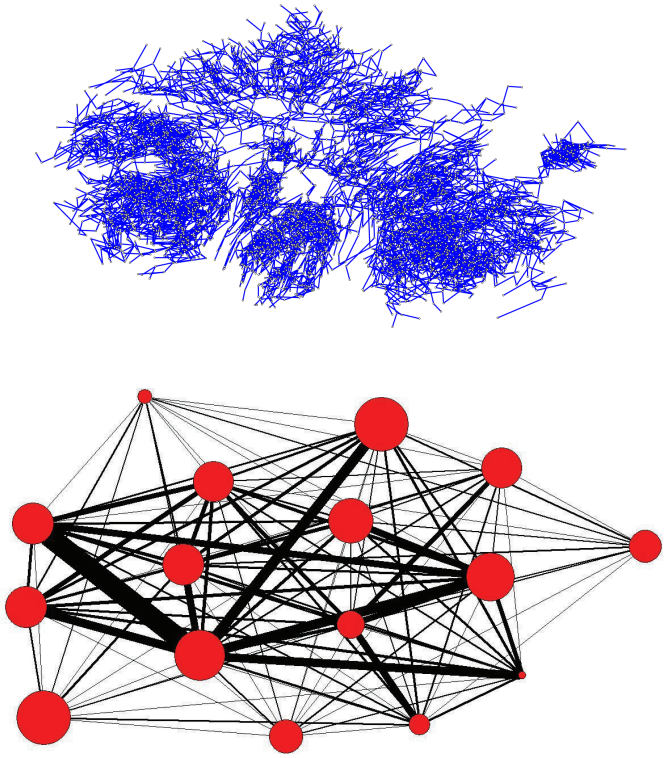


Figure 3.16: ***Power grid* network visualization**. Original network (top) and community structure of the Western USA power grid discovered by the kernel spectral clustering model (bottom). The comments made for Figure 3.15 are still valid here.

Chapter 4

Clustering Evolving Networks

Evolving graphs describe many natural phenomena changing over time, such as social relationships, trade markets, metabolic networks etc. In this chapter we describe a new model for clustering evolving networks, where the smoothness of the clustering results over time can be considered as a valid prior knowledge. This algorithm, called kernel spectral clustering with memory effect (MKSC), is based on a constrained optimization formulation typical of LS-SVM, where the objective function is designed to explicitly incorporate temporal smoothness. The latter allows the model to cluster the current data well and to be consistent with the recent past. We also introduce new cluster quality measures that can be used for model selection or better assess the performance of a clustering algorithm in a dynamic scenario. Moreover, we consider two frameworks:

- framework 1: we assume that at every time step the partitioning has to be smoothed, and we focus on the case where the number of nodes and communities is not changing across time.
- framework 2: the smoothness parameter ν is tuned, and the number of nodes and communities is varying over time.

Also a compact visualization of the cluster dynamics in a 3D embedding is proposed. Finally, the MKSC model is successfully applied to a number of toy problems and a real-world network. We also compare the novel algorithm with some state-of-the-art methods like Evolutionary Spectral Clustering (ESC [27]), the Louvain method applied separately in each snapshot, and Adaptive Evolutionary Clustering (AFFECT [139]).

4.1 Literature Review

In many practical applications we deal with community detection in dynamic scenarios, which recently has been the subject of a common research endeavour in the science community. The first article that formalized the problem is [25], where dynamic community detection is named evolutionary clustering. This work is based on the intuition that if the new data does not deviate from the recent history the clustering should be similar to that performed for the previous data. However, if the data changes significantly, the clustering must be modified to reflect the new structure. This temporal smoothness between clusters in successive time-steps is also the main principle behind the methods introduced in [27], [88] and [139]. In particular, in [27] the evolutionary spectral clustering algorithm (ESC) has been proposed, which aims to optimize the cost function $J_{tot} = \eta J_{\text{temp}} + (1-\eta) J_{\text{snap}}$. J_{snap} describes the classical spectral clustering objective related to each snapshot of an evolving graph. J_{temp} measures the cost of applying the partitioning found at time t to the snapshot at time $t-1$, penalizing then clustering results that disagree with the recent past. In [139] an evolutionary clustering framework that adaptively estimates the optimal smoothing parameter using shrinkage estimation is presented. The method, called AFFECT, allows to extend a number of static clustering algorithms into evolutionary clustering techniques. In [103] the authors developed a methodology that generalizes the determination of community structure to multi-slice networks that are defined by coupling multiple adjacency matrices. By encoding in the inter-slice connections variations across time, it is possible to analyse evolving networks. The work presented in [113] discovered an interesting relationship between the size of a community, its lifetime and stationarity. A typical small and stationary community is observed to undergo minor changes, while living for a long time. In contrast, a small community with high turnover of its members tends to have a small lifetime. The opposite is observed for large communities: a big stationary community disintegrates early, while a large non-stationary community whose members change dynamically has a longer lifetime. In [10] the authors perform clustering separately for each snapshot of an evolving network and develop a framework for capturing and identifying interesting events from them. They use then these events to characterize complex behavioural patterns of nodes and communities over time. Also [49] introduces a method for efficiently identifying and tracking dynamic communities, which involves matching communities found at consecutive time steps in the individual snapshot graphs.

In this chapter, we describe a new method for clustering dynamic networks called kernel spectral clustering with memory effect (MKSC). Our technique is developed in the LS-SVM learning framework, where we incorporate the temporal smoothness between clusters in successive time-steps at the primal level. In this way the model is able to track the long-term trend and at the same time it reduces the short-term variation due to noise, similarly to what happens with moving averages in time-series

analysis. Moreover, a precise model selection scheme and the out-of-sample extension to new nodes is presented. Up to our knowledge, all these features make MKSC unique in its kind. Finally, we consider two main frameworks:

- **framework 1**: we assume that at every time step the partitioning has to be smoothed. Thus, we fix the regularization constant ν (which acts as a smoothness parameter) to $\nu = 1$. Then the amount of smoothness is chosen by the user through the memory M. Moreover, we focus on the case where the number of nodes and communities is not changing across time.
- **framework 2**: the smoothness parameter ν is tuned, meaning that the memory activates automatically only when is needed, i.e. when the network undergoes major changes in its community structure. Moreover, we allow MKSC to deal with nodes entering and leaving over time and to recognize a rich variety of events (splitting, merging, dissolving etc.) by introducing a simple tracking mechanism.

4.2 The MKSC Model

A dynamic network is a sequence of networks $\mathcal{S} = \{\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)\}_{t=1}^{T}$ over time T, where t indicates the time index. The symbol $\mathcal{V}_t$ indicates the set of nodes in the graph $\mathcal{G}_t$ and $\mathcal{E}_t$ the related set of edges. In what follows we assume that $|\mathcal{V}_t|$ is constant, that is all the graphs in the sequence have the same number of nodes (the symbol $|\cdot|$ indicates the cardinality of a set). The MKSC model implements a trade-off between the current clustering and the previous partitioning by incorporating the temporal smoothness as prior knowledge.

Given an evolving network with N nodes, for each snapshot the primal problem of the MKSC model, where N_{Tr} nodes are used for training, can be stated as follows in matrix notation [80]:

$$\begin{aligned} \min_{w^{(l)}, e^{(l)}, b_l} \quad & \frac{1}{2}\sum_{l=1}^{k-1} {w^{(l)}}^T w^{(l)} - \frac{\gamma}{2N_{\text{Tr}}}\sum_{l=1}^{k-1} {e^{(l)}}^T D_{\text{Mem}}^{-1} e^{(l)} - \nu \sum_{l=1}^{k-1} {w^{(l)}}^T w_{\text{old}}^{(l)} \\ \text{subject to} \quad & e^{(l)} = \Phi w^{(l)} + b_l 1_{N_{\text{Tr}}}. \end{aligned} \tag{4.1}$$

The first term in the objective (4.1) indicates the minimization of the model complexity, while the second term casts the clustering problem in a weighted kernel PCA formulation as in [6]. The third term, i.e. $\sum_{l=1}^{k-1} {w^{(l)}}^T w_{\text{old}}^{(l)}$, describes the correlation between the actual and the previous models, which we want to maximize. In this way it is possible to introduce temporal smoothness in our formulation, such that the

current partition does not deviate too dramatically from the recent past. The subscripts *old* and *Mem* refer to time steps $t-1,\ldots,t-M$, where M indicates the memory, that is the past information we want to consider when performing the clustering at the actual time step t. The symbols have the following meaning:

- $e^{(l)}$ represent the l-th binary clustering model for the N points and are referred interchangeably as projections, latent variables or score variables.
- the index $l=1,\ldots,k-1$ indicates the score variables needed to encode the k clusters to find via an Error Correcting Output Codes (ECOC) encoding-decoding procedure. In other words, $e_i^{(l)} = w^{(l)^T}\varphi(x_i) + b_l$ are the latent variables of a set of $k-1$ binary clustering indicators given by sign$(e_i^{(l)})$. The binary indicators are combined to form a codebook $\mathcal{CB} = \{c_p\}_{p=1}^k$, where each codeword is a binary string of length $k-1$ representing a cluster.
- $w^{(l)} \in \mathbb{R}^{d_h}$ and b_l are the parameters of the model at time t, and $w_{\text{old}}^{(l)} = \sum_{i=1}^M w_{\text{prev,i}}^{(l)} = \sum_{i=1}^M \Phi_{\text{prev,i}}^{(l)} \alpha_{\text{prev,i}}^{(l)}$ are the model parameters related to the M previous snapshots $\mathcal{G}_{t-1},\ldots,\mathcal{G}_{t-M}$. The subscript *prev,i* means time step $t-i$. By considering M time stamps in the past, we say that our model has a memory of M snapshots.
- $D^{-1} \in \mathbb{R}^{N_{\text{Tr}}\times N_{\text{Tr}}}$ is the inverse of the degree matrix D related to the actual kernel matrix Ω, i.e. $D_{ii} = \sum_j \Omega_{ij}$, while $D_{\text{Mem}}^{-1} \in \mathbb{R}^{N_{\text{Tr}}\times N_{\text{Tr}}}$ is the inverse of the degree matrix $D_{\text{Mem}} = D + \sum_{r=1}^M D_{\text{new-prev,r}}$, which is the sum of the actual degree matrix D and the M previous degree matrices, each with entries $D_{\text{new-prev},ii} = \sum_j \Omega_{\text{new-prev},ij}$.
- Φ is the $N_{\text{Tr}} \times d_h$ feature matrix $\Phi = [\varphi(x_1)^T;\ldots;\varphi(x_{N_{\text{Tr}}})^T]$ which expresses the relationship between each pair of data objects in a high dimensional feature space $\varphi : \mathbb{R}^d \to \mathbb{R}^{d_h}$.
- $\gamma \in \mathbb{R}^+$ and $\nu \in \mathbb{R}^+$ are regularization constants. In particular ν can be thought as a kind of smoothness parameter, since it enforces the current model to resemble the old models, that is the ones developed for the previous M snapshots.

The dual solution to the constrained optimization problem (4.1) is formalized in the following Lemma.

Lemma [80] *Given a positive definite kernel function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, with $K(x_i,x_j) = \varphi(x_i)^T\varphi(x_j)$, non-negative regularization constants γ and ν, the inverse of the degree matrix D_{Mem} which is diagonal with positive entries, the Karush-Kuhn-Tucker (KKT) optimality conditions of the Lagrangian of (4.1) are satisfied by the*

following set of linear systems:

$$\begin{aligned}(D_{\text{Mem}}^{-1}M_{D_{\text{Mem}}}\Omega - \frac{I}{\gamma})\alpha^{(l)} &= -\nu D_{\text{Mem}}^{-1}M_{D_{\text{Mem}}}\Omega_{\text{new-old}}\alpha_{\text{old}}^{(l)} \\ &= -\nu D_{\text{Mem}}^{-1}M_{D_{\text{Mem}}}\sum_{r=1}^{M}\Omega_{\text{new-prev,r}}\alpha_{\text{prev,r}}^{(l)}\end{aligned} \tag{4.2}$$

where

- Ω indicates the current kernel matrix with ij-th entry $\Omega_{ij} = K(x_i, x_j) = \varphi(x_i)^T\varphi(x_j)$. $\Omega_{\text{new-old}}$ captures the similarity between the objects of the current snapshot and the ones of the previous M snapshots, and has the ij-th entry $\Omega_{\text{new-old},ij} = \sum_{r=1}^{M} K(x_i^{\text{new}}, x_j^{\text{prev,r}})$.
- $M_{D_{\text{Mem}}}$ is the centering matrix equal to $M_{D_{\text{Mem}}} = I_{N_{\text{Tr}}} - \frac{1}{1_{N_{\text{Tr}}}^T D_{\text{Mem}}^{-1} 1_{N_{\text{Tr}}}} 1_{N_{\text{Tr}}} 1_{N_{\text{Tr}}}^T D_{\text{Mem}}^{-1}$.

Proof: The Lagrangian of the problem (4.1) is:

$$\begin{aligned}\mathcal{L}(w^{(l)}, e^{(l)}, b_l; \alpha^{(l)}) =& \frac{1}{2}\sum_{l=1}^{k-1} {w^{(l)}}^T w^{(l)} - \frac{\gamma}{2N_{\text{Tr}}}\sum_{l=1}^{k-1} {e^{(l)}}^T D_{\text{Mem}}^{-1} e^{(l)} \\ & - \nu\sum_{l=1}^{k-1} {w^{(l)}}^T w_{\text{old}}^{(l)} + {\alpha^{(l)}}^T(e^{(l)} - \Phi w^{(l)} - b_l 1_{N_{\text{Tr}}}).\end{aligned}$$

The KKT optimality conditions are:

$\frac{\partial \mathcal{L}}{\partial w^{(l)}} = 0 \rightarrow w^{(l)} = \Phi^T\alpha^{(l)} + \nu w_{\text{old}}^{(l)}$,
$\frac{\partial \mathcal{L}}{\partial e^{(l)}} = 0 \rightarrow \alpha^{(l)} = \gamma D_{\text{Mem}}^{-1} e^{(l)}$,
$\frac{\partial \mathcal{L}}{\partial b_l} = 0 \rightarrow 1_{N_{\text{Tr}}}^T\alpha^{(l)} = 0$,
$\frac{\partial \mathcal{L}}{\partial \alpha^{(l)}} = 0 \rightarrow e = \Phi w^{(l)} + b_l 1_{N_{\text{Tr}}}$.

From $w_{\text{old}}^{(l)} = \Phi_{\text{old}}^T\alpha_{\text{old}}^{(l)}$, the bias term becomes

$b_l = -\frac{1}{1_{N_{\text{Tr}}}^T D_{\text{Mem}}^{-1} 1_{N_{\text{Tr}}}} 1_{N_{\text{Tr}}}^T D_{\text{Mem}}^{-1}(\Omega\alpha^{(l)} + \nu\Omega_{\text{new-old}}\alpha_{\text{old}}^{(l)})$. Eliminating the primal variables $e^{(l)}$, $w^{(l)}$, b_l leads to the dual problem (4.2) □

The cluster indicators for the training data are:

$$\text{sign}(e^{(l)}) = \text{sign}(\Omega\alpha^{(l)} + \nu\Omega_{\text{new-old}}\alpha_{\text{old}}^{(l)} + b1_{N_{\text{Tr}}}). \tag{4.3}$$

The score variables for test points are defined as follows:

$$e_{\text{test}}^{(l)} = \Omega^{\text{test}}\alpha^{(l)} + \nu\Omega_{\text{new-old}}^{\text{test}}\alpha_{\text{old}}^{(l)} + b_l 1_{N_{\text{test}}}. \tag{4.4}$$

Thus, once we have properly trained our model, the cluster memberships of new points can be predicted by projecting the test data into the solution vectors $\alpha^{(l)}$ and $\alpha_{\text{old}}^{(l)}$ via eq. (4.4).

4.2.1 Cluster Quality Measures in a Dynamic Scenario

In order to assess the quality of a partition related to an evolving network, new measures are introduced. The new measures are the weighted sum of the snapshot quality and the temporal quality. The former only measures the quality of the current clustering with respect to the current data, while the latter measures the temporal smoothness in terms of the ability of the actual model to cluster the historic data. A partition related to a particular snapshot will then receive a score that is higher as it is more consistent with the past, for a particular value of the parameter η. For a given cluster quality criterion CQ, we can define its smoothed version related to the actual snapshot $\mathcal{G}_t$ in this way:

$$\text{CQ}_{\text{Mem}}(X_\alpha, \mathcal{G}_t) = \eta\text{CQ}(X_\alpha, \mathcal{G}_t) + (1-\eta)\text{CQ}(X_\alpha, \mathcal{G}_{t-1}), \tag{4.5}$$

where X_α indicates the cluster indicator matrix calculated by using the current solution vectors $\alpha^{(l)}$. With η we denote a user-defined parameter which takes values in the range $[0, 1]$ and reflects the emphasis given to the snapshot quality and the temporal smoothness, respectively.

The smoothed counterparts of the model selection criteria introduced for KSC and SKSC, that is BLF_{Mem}, Mod_{Mem} and AMS_{Mem} can be used also to select optimal parameters for the MKSC model. Nevertheless, in the experiments described later, we did not notice big differences in using the static or smoothed measures when performing the model selection for MKSC.

Model selection

The model selection scheme for MKSC is summarized in algorithm 7. This procedure is standard but often we do not need to tune all the parameters. For instance in case of network data we can use the cosine kernel and the community kernel which are parameter-free. Moreover we have observed by extensive simulations that we can fix one of the two regularization constants and tune only the other one. In particular, in *framework 1* we tune γ and we fix ν and vice-versa in *framework 2*.

Algorithm 7: Model selection algorithm for MKSC [80]

Data: Training sets and validation sets (actual and previous snapshots), kernel function $K : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ positive definite and localized.

Result: Selected number of clusters k, kernel parameters (if any), γ, ν.

1 define a grid of values for the parameters to select
2 select training and validation sets
3 train the related MKSC model
4 compute the cluster indicator matrix corresponding to the validation data
5 for every partition calculate the related score by using the chosen criterion
6 select the model with the highest score.

4.2.2 Computational Complexity

The time required to solve the set of linear systems (4.2) scales as $O(N_{\text{Tr}}^3)$. However, the runtime can be reduced by exploiting the matrix inversion Lemma [55]. We can solve the dual problem of MKSC for a smaller training set and then calculate iteratively the $N_{\text{Tr}} \times 1$ solution vector $\alpha^{(l)}$, $\forall l$, by using the Woodbury formula. Considering the whole network as test set, the time needed to compute the membership for all the N nodes in each snapshot scales as $O(N_{\text{Tr}}N)$ if we use a sparse representation. Moreover, in case the matrices Ω^{test} and $\Omega^{\text{test}}_{\text{new-old}}$ cannot fit the memory we can divide the test set into blocks and perform the testing operations iteratively on a single computer or in parallel in a distributed environment, as shown in [96, 97] for kernel spectral clustering (KSC). However, in the worst case scenario (non-sparse representation of the variables and inefficient implementation of the kernel function), the complexity can rise to $O(N^2)$ or $O(N^3)$. Finally, in figure 4.1 the runtime of MKSC and ESC needed for clustering some artificial evolving networks of different size are compared.

4.3 Framework 1

In this section we describe the experimental results when considering networks where the number of nodes and communities is not varying across time. Moreover we require that the clustering results are smoothed at each time stamp by fixing $\nu = 1$. The method that we use can be summarized in algorithm 8.

4.3.1 Artificial Examples

The synthetic datasets used in the simulations can be described as follows:

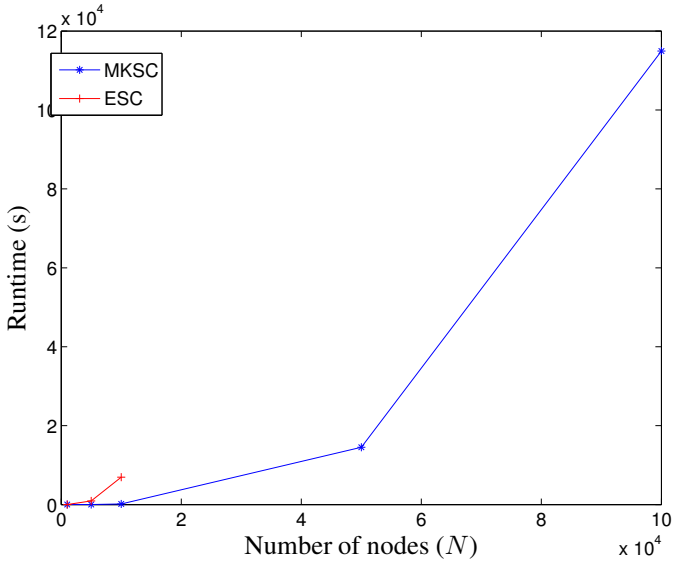


Figure 4.1: **MKSC computational complexity.** Evolution of the speed with the size of a benchmark network for ESC (until $N = 10^3$ because of memory problems) and MKSC. Although MKSC is faster than ESC, the runtime of the former seems to scale as $O(N^3)$. The is due to the usage of a non-sparse representation of the variables and an inefficient implementation of the community kernel function.

- **MG2**: the setup for this experiment is shown in figure 4.2. We generate 1000 samples from a mixture of two 2D Gaussian clouds, with 500 samples drawn from each component of the mixture. From time-steps 1 to 10 we move the means of the two Gaussian clouds towards each other until they overlap almost completely. This phenomenon can be also described by considering the points of the two Gaussians as nodes of a weighted network, where the weights of the edges change across time.

- **MG3**: three merging Gaussian clouds, shown in figure 4.3.

- **SwitchingNet**: we build up 9 snapshots of a network of 1000 nodes formed by 2 communities. At each time-step some nodes switch their membership between the two clusters. We used the software related to [49] to generate this benchmark.

- **ECNet**: a network with 5 communities experiences over time 24 expansion events and 16 contractions of its communities.

Algorithm 8: Clustering evolving networks: framework 1 [80]

Data: Training sets $\mathcal{D} = \{x_i\}_{i=1}^{N_{\text{Tr}}}$ and $\mathcal{D}_{\text{old}} = \{x_i^{\text{old}}\}_{i=1}^{N_{\text{Tr}}}$, test sets $\mathcal{D}^{\text{test}} = \{x_m^{\text{test}}\}_{m=1}^{N_{\text{test}}}$ and $\mathcal{D}_{\text{old}}^{\text{test}} = \{x_m^{\text{test,old}}\}_{m=1}^{N_{\text{test}}}$, $\alpha_{\text{old}}^{(l)}$ (the $\alpha^{(l)}$ calculated for the previous M snapshots), positive definite kernel function $K : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ such that $K(x_i, x_j) \rightarrow 0$ if x_i and x_j belong to different clusters, kernel parameters (if any), number of clusters k, regularization constants γ and ν found using algorithm 7.

Result: Clusters $\{\mathcal{C}_1^t, \ldots, \mathcal{C}_p^t\}$, cluster codeset $\mathcal{CB} = \{c_p\}_{p=1}^k$, $c_p \in \{-1, 1\}^{k-1}$.

if *t==1* **then**

  Initialization by using kernel spectral clustering (KSC [6]).

else

  Compute the solution vectors $\alpha^{(l)}$, $l = 1, \ldots, k-1$, related to the linear systems described by eq. (4.2):

$$(D_{\text{Mem}}^{-1} M_{D_{\text{Mem}}} \Omega - \frac{I}{\gamma})\alpha^{(l)} = -\nu D_{\text{Mem}}^{-1} M_{D_{\text{Mem}}} \Omega_{\text{new-old}} \alpha_{\text{old}}^{(l)}$$

  .

  Binarize the solution vectors: $\text{sign}(\alpha_i^{(l)})$, $i = 1, \ldots, N_{\text{Tr}}$, $l = 1, \ldots, k-1$, and let $\text{sign}(\alpha_i) \in \{-1, 1\}^{k-1}$ be the encoding vector for the training data point x_i.

  Count the occurrences of the different encodings and find the k encodings with most occurrences. Let the codeset be formed by these k encodings: $\mathcal{CB} = \{c_p\}_{p=1}^k$, with $c_p \in \{-1, 1\}^{k-1}$.

  $\forall i$, assign x_i to C_{p^*} where $p^* = \text{argmin}_p d_H(\text{sign}(\alpha_i), c_p)$ and $d_H(\cdot, \cdot)$ is the Hamming distance.

  Binarize the test data projections $\text{sign}(e_m^{(l)})$, $m = 1, \ldots, N_{\text{test}}$, $l = 1, \ldots, k-1$ and let $\text{sign}(e_m) \in \{-1, 1\}^{k-1}$ be the encoding vector of x_m^{test}, $m = 1, \ldots, N_{\text{test}}$.

  $\forall m$, assign x_m^{test} to $C_{p^*}^t$ using an ECOC decoding scheme, i.e. $p^* = \text{argmin}_p d_H(\text{sign}(e_m), c_p)$.

end

In the next sections we describe the model selection issue and we assess the quality of the partitions produced by MKSC. We compare MKSC with KSC and ESC. In general, as expected, we observe that the dynamic clustering algorithms MKSC and ESC produce better clustering results over time than KSC.

Tuning the hyper-parameters

Regarding KSC we use the BLF criterion described in section 2.2.4 to tune the number of clusters k and the hyper-parameter σ of the RBF kernel in the two and three moving Gaussians experiments. The results are shown at the top of figures 4.4 and 4.5 and refer to the first snapshot (for the other snapshots the plots are similar). For the SwitchingNet graph in the first row of figure 4.6 we illustrate how the Modularity-

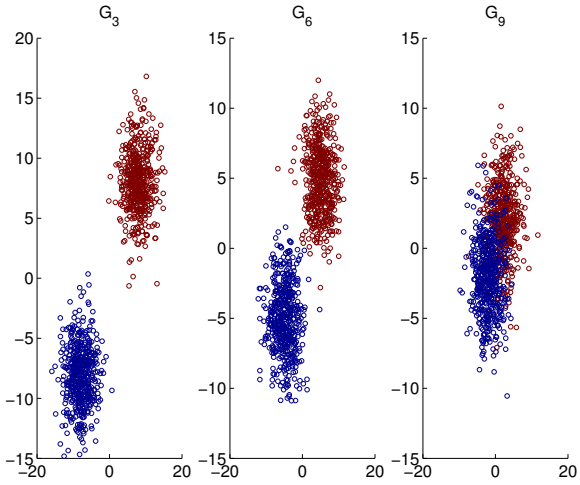


Figure 4.2: **MG2 dataset**. The snapshots number 3, 6 and 9 of the two moving Gaussians synthetic example are shown. The axes represent the two dimensions of the N input vectors $x_i \in \mathbb{R}^2, i = 1, \ldots, N$.

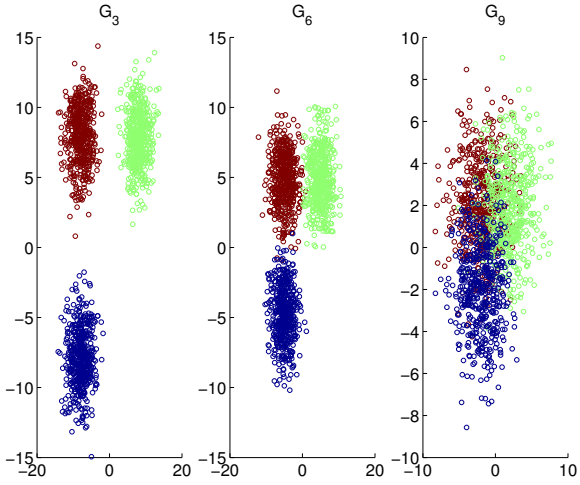


Figure 4.3: **MG3 dataset**. The snapshots number 3, 6 and 9 of the three moving Gaussians artificial example are depicted. The axes represent the two dimensions of the N input vectors $x_i \in \mathbb{R}^2, i = 1, \ldots, N$.

based model selection scheme introduced in section 3.2.2 correctly identifies the possible presence of two communities. In this case k is the only parameter to tune since the community kernel used in this case is parameter-free as explained in section 3.2.3. Also in the case of the ECNet artificial network the 5 communities are correctly detected, as illustrated at the top of figure 4.7.

Concerning the MKSC model, for all the datasets we use the same values of σ and k found for KSC and we need to tune ν and γ. For simplicity we fix the value of ν to 1 and we tune only γ. The optimal γ over time for the two and three moving Gaussian experiments are shown respectively in the second row of figures 4.4 and 4.5, where the BLF_{Mem} criterion is used for tuning. At the bottom of figure 4.6 the optimal value of γ across time for SwitchingNet suggested by the Smoothed Modularity-based model selection criterion sketched in algorithm 7 is shown. The bottom side of figure 4.7 depicts the optimal γ for the ECNet synthetic network.

Evaluation of the results

Here we present the simulation results. For what concerns the models with temporal smoothness MKSC and ESC the first partition is found by applying the corresponding static model KSC and SC to the first snapshot since we do not have any information from the past. Then we move along the next snapshots, and we apply the ESC method and MKSC algorithm with memory of $M_1 = 1$, $M_2 = 2$ and $M_3 = 3$ snapshots. In figures 4.8, 4.9, 4.10 and 4.11 we present the performance of KSC, MKSC with memory $M = M_1$ and ESC in analysing the four artificial datasets under study, in terms of the smoothed cluster quality measures introduced before. Moreover, for what concerns the two and three Gaussians experiments, we also show the out-of-sample clustering results evaluated on grid points surrounding the Gaussian clouds. By looking at the figures, we can draw the following observations:

- *MG2* experiment: the models with temporal smoothness (ESC and MKSC) can better distinguish between the two Gaussians while they are very overlapping with respect to the static model KSC, and obtain comparable results.
- *MG3* dataset: the same consideration valid for the two moving Gaussians can be drawn (here the ESC algorithm obtains the best results). In this case, however, we can notice also from the out-of-sample plot that MKSC, thanks to the memory effect introduced in the formulation of the primal problem, remembers the old clustering boundaries compared to KSC (see in particular the results related to the 9-th snapshot).
- *SwitchingNet*: MKSC performs slightly better than KSC and much better than ESC. The bad results obtained by ESC are quite unexpected and need further investigation. Probably they can be explained by considering that the

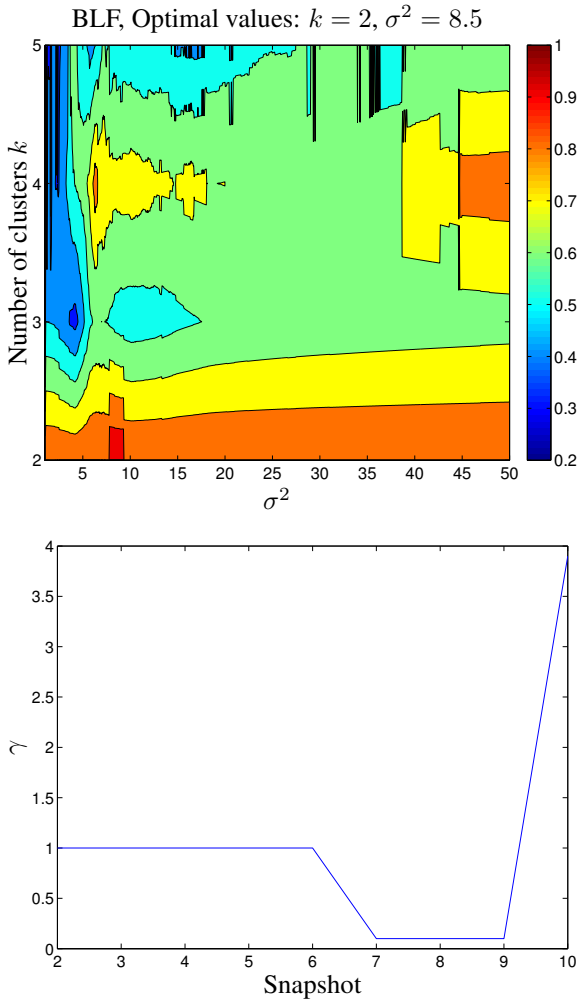


Figure 4.4: **Model selection MG2**. Top: tuning of the number of clusters k and the RBF kernel parameter σ^2 related to the first snapshot of the two moving Gaussians experiment, for KSC. The optimal σ^2 does not change over time (the model selection procedure gives similar results also for the other snapshots). Moreover, this value is also used for the MKSC model. Bottom: optimal value of γ over time for MKSC, tuned using the BLF_{Mem} method.

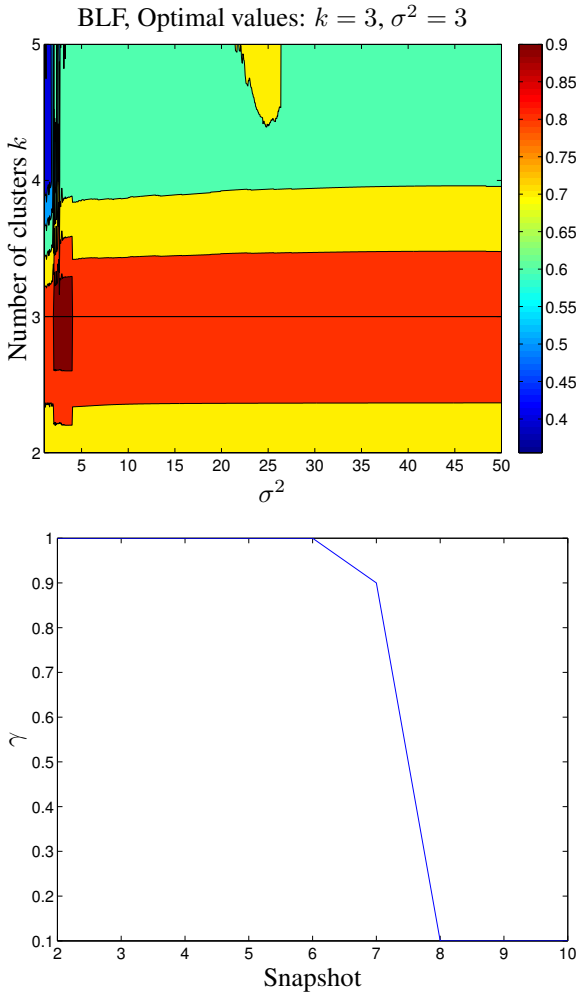


Figure 4.5: **Model selection MG3**. Top: tuning of the RBF kernel parameter σ^2 and the number of clusters k related to the first snapshot of the three moving Gaussians experiment, for KSC. Bottom: tuning of γ for MKSC. The same comments made for Fig. 4.4 are still valid here.

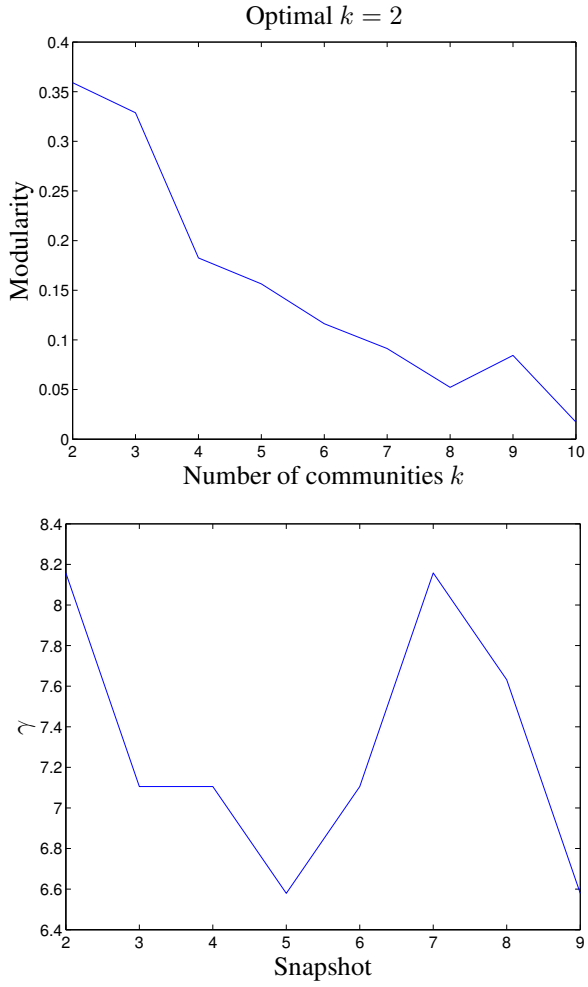


Figure 4.6: **Model selection SwitchingNet**. Top: tuning of the number of clusters k for the switching network, related to the first snapshot and the KSC model. The results are similar for the other snapshots and this k is also used as input to the MKSC algorithm. Bottom: optimal value of γ over time for the MKSC model, selected by using the Mod_{Mem} criterion.

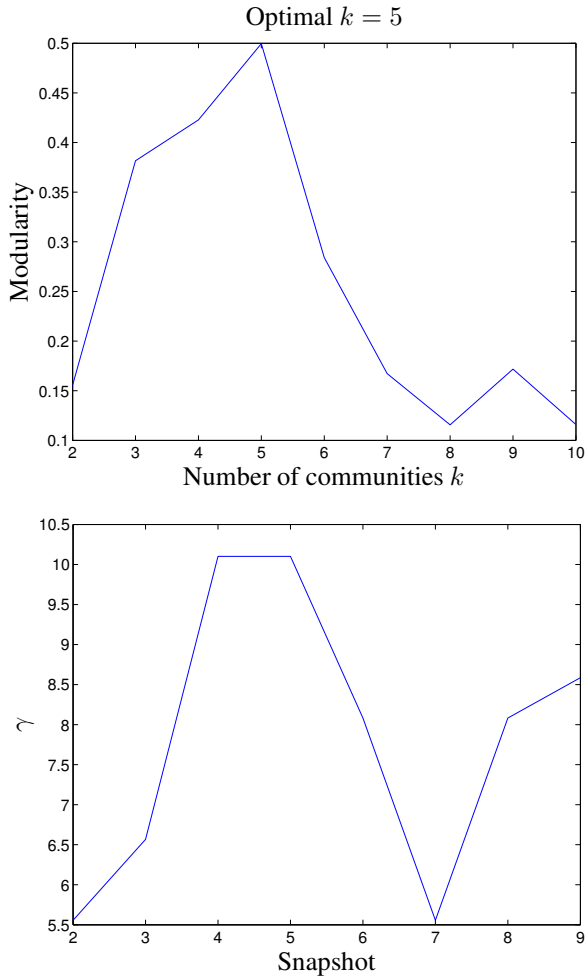


Figure 4.7: **Model selection ECNet**. Top: tuning of the number of clusters k for the expanding/contracting network, related to the first snapshot and the KSC algorithm. Bottom: tuning of γ for MKSC. The comments made for Fig. 4.6 hold also in this case.

community structure is quite different from snapshot to snapshot and while MKSC is flexible in adapting to this situation, ESC is not.

- *ECNet* graph: as expected the models with temporal smoothness (MKSC and ESC) obtain better results than the static KSC model. MKSC produces the best performances.

In table 4.1 the results regarding MKSC with memory M_1, M_2 and M_3 are summarized, together with the memory requirement and the computation time. For the two and three moving Gaussians datasets we can notice how increasing the memory of the MKSC model can give smoother and better clustering results over time, measured in terms of Normalized Mutual Information (NMI) between consecutive partitioning and the smoothed ARI. For what concerns *SwitchingNet* and *ECNet* networks, the more memory we add the more similar are the clustering results over time. Moreover it seems that one snapshot of memory is enough to have good performances. Finally it has to be mentioned that ESC provides unstable results, since sometimes the performances can decrease in quality (see for example the NMI plot in figure 4.9). This is possibly due to the use of K-means to produce the final clustering. Indeed it is well known that the K-means algorithm depends on a random initialization which can lead sometimes to suboptimal results. On the other hand, MKSC does not suffer from this drawback.

4.3.2 Real-Life Application

The **CellphoneNet** has been investigated. This dataset records the cellphone activity for students and staff from two different labs in MIT [39]. It is constructed on users whose cellphones periodically scan for nearby phones over Bluetooth at five minute intervals. The similarity between two users is related to the number of intervals where they were in physical proximity. Each graph snapshot is a weighted network corresponding to 1 week activity. In particular we consider 42 nodes, representing students always present during the fall term of the academic year 2004-2005, for a total of 12 snapshots.

Regarding the model selection, we have a partial ground truth, namely the affiliations of each participant. In particular, as observed in [39] and in [127], 2 dominant clusters could be identified from the Bluetooth proximity data, corresponding to new students at the Sloan business school and co-workers who work in the same building. Then for this experiment we perform clustering with number of clusters $k = 2$, while the optimal σ over time is estimated by using the Modularity-based model selection algorithm on each snapshot (see figure 4.12). So $k = 2$ and $\gamma = 1$ are optimal hyper-parameters for each of the 12 weeks, while the values of σ^2 over time are reported in figure 4.12.

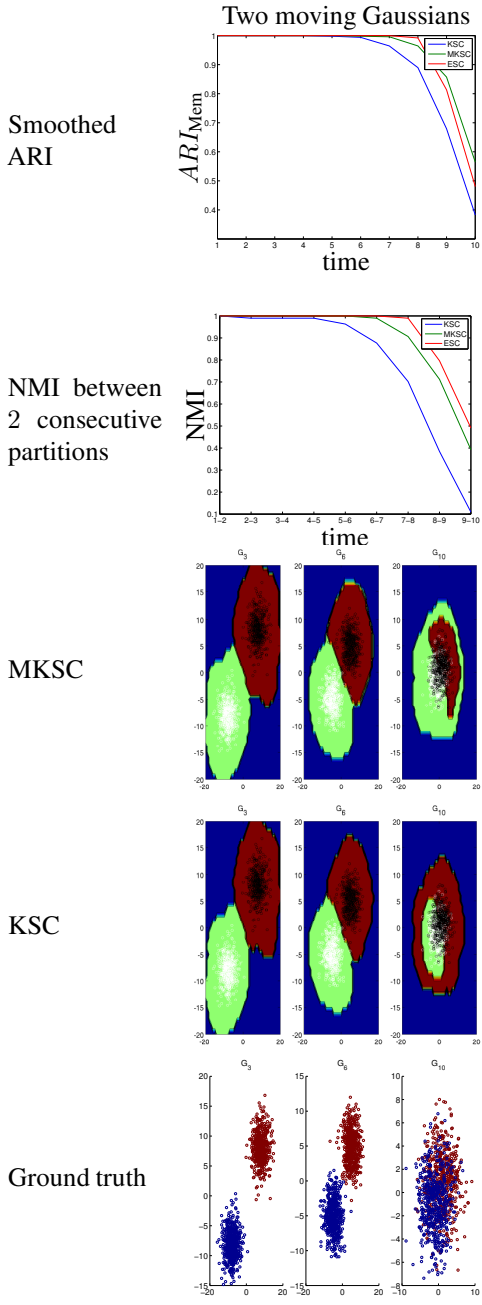


Figure 4.8: **Clustering results MG2**: performance of MKSC, KSC and ESC in the two moving Gaussians experiment (first row) and out-of-sample plot for MKSC and KSC (only the results related to snapshots 3, 6 and 10 are shown). The true partitioning is depicted in the fifth row. The smoothed ARI plot and the NMI trend tell us that, as expected, the models with temporal smoothness are more able than KSC to produce clustering results which are more similar to the ground truth and also more consistent and smooth over time. However, in the out-of-sample plot we cannot visually appreciate the better performance of MKSC with respect to KSC.

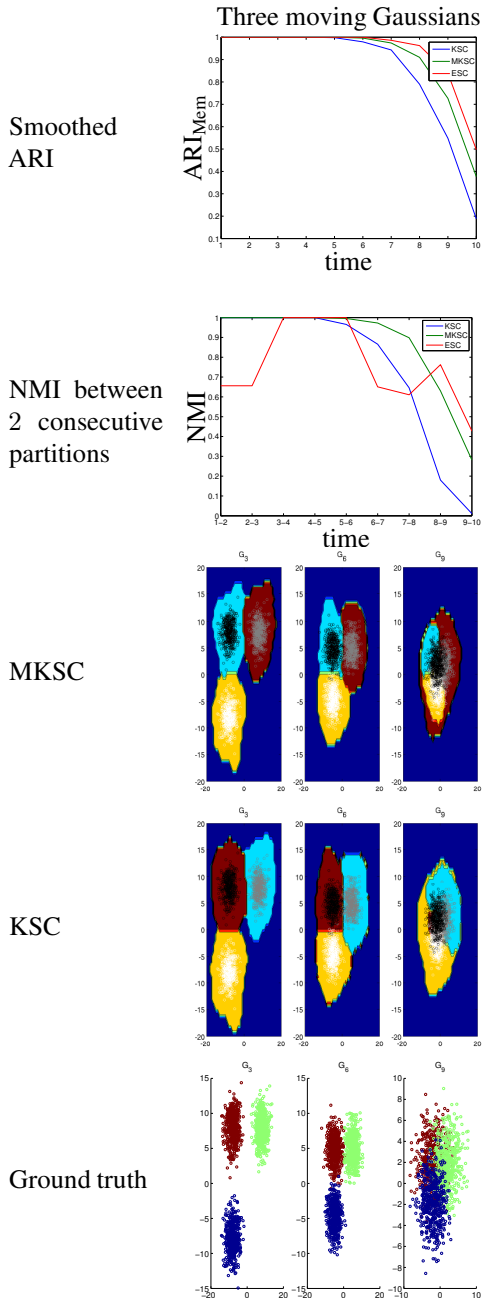


Figure 4.9: **Clustering results MG3**: Performance of MKSC, KSC and ESC in the three moving Gaussians experiment in terms of smoothed ARI and NMI between two consecutive partitions, and out-of-sample plot for MKSC and KSC (only the results related to snapshots 3, 6 and 9 are shown). The true partitioning is depicted in the last row. The same observations made for Fig 4.8 are still valid here. In this case, however, in the out-of-sample plots we can better recognize that MKSC, thanks to the memory effect introduced in the formulation of the primal problem, is more able than KSC to remember the old clustering boundaries and produces then smoother results over time (consider in particular the 9-th snapshot). Finally, from the NMI plot we can notice that sometimes the ESC algorithm produces unstable results.

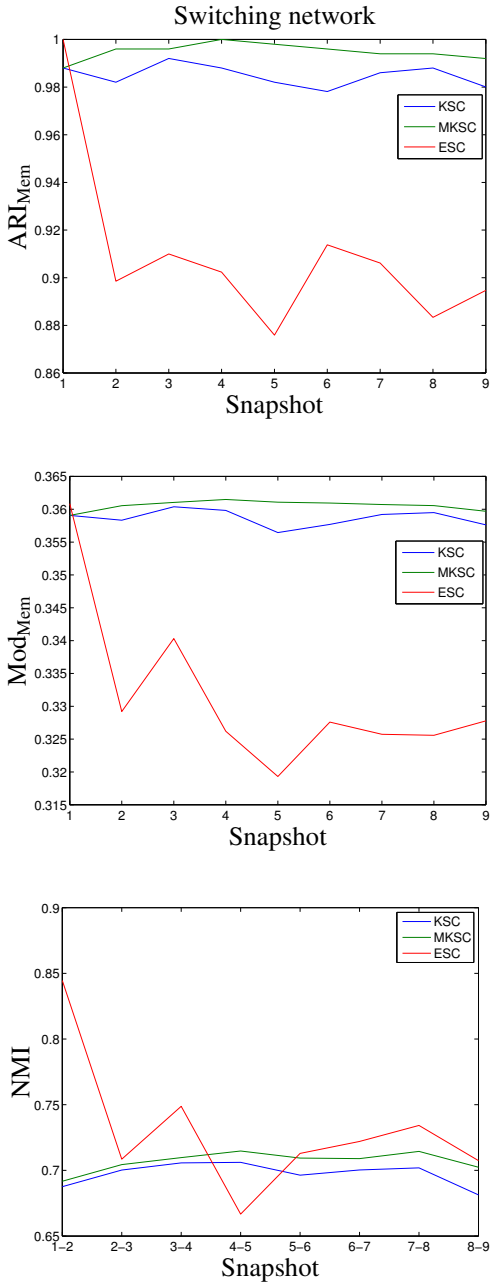


Figure 4.10: **Clustering results SwitchingNet**: performance of MKSC, KSC and ESC on the artificial evolving network with 2 communities in terms of the new smoothed cluster measures explained in Section 4.2.1. Here, surprisingly, KSC produces better results than the ESC model according to the smoothed ARI and Modularity. However MKSC performs better than KSC.

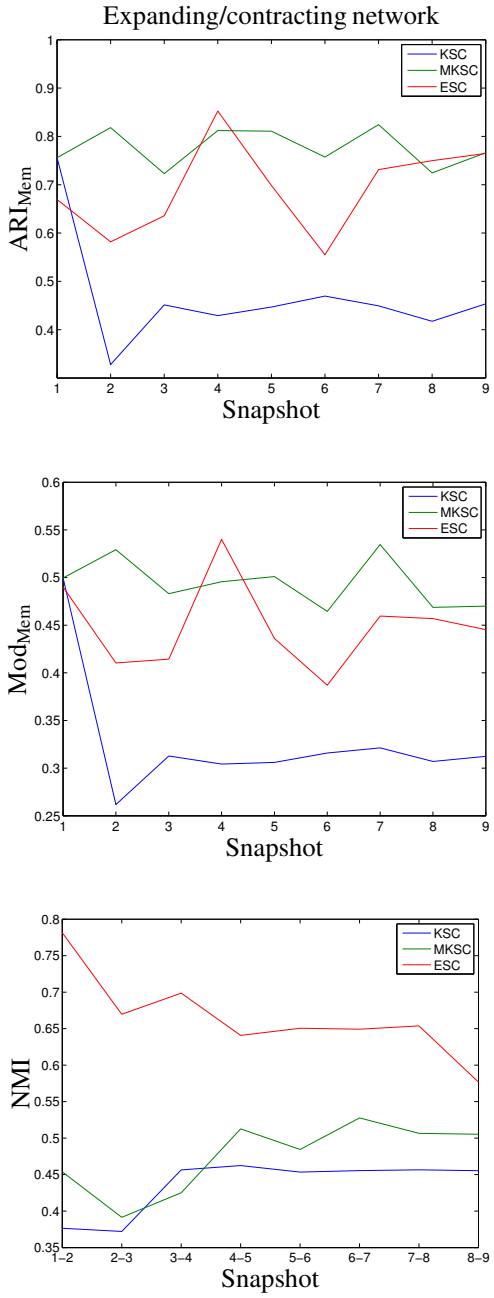


Figure 4.11: **Clustering results ECNet**: performance of MKSC, KSC and ESC on the artificial evolving network with 5 communities. The models with temporal smoothness produce partitions of higher quality than KSC (according to the smoothed measures introduced in Section 4.2.1), encouraging more consistent clustering over time. If we consider the NMI plot, ESC is the best method while MKSC outperforms all the others in terms of the smoothed measures.

MG2		
MEASURE	**MKSC**	**ESC**
$\mathbf{ARI}_{\text{Mem}}$	$\mathbf{0.88}$ (M_3)	0.87
NMI	$\mathbf{0.86}$ (M_3)	0.85
CPU time (s)	$\mathbf{0.53}$ (M_1)	5.36
Memory requirement (s)	$\mathbf{0.29}$ (M_1)	2.92
MG3		
MEASURE	**MKSC**	**ESC**
$\mathbf{ARI}_{\text{Mem}}$	$\mathbf{0.92}$ (M_3)	0.81
NMI	$\mathbf{0.91}$ (M_3)	0.75
CPU time (s)	$\mathbf{1.30}$ (M_1)	18.95
Memory requirement (s)	$\mathbf{0.73}$ (M_1)	10.77
SwitchingNet		
MEASURE	**MKSC**	**ESC**
$\mathbf{ARI}_{\text{Mem}}$	$\mathbf{0.99}$ (M_1)	0.91
$\mathbf{Mod}_{\text{Mem}}$	$\mathbf{0.36}$ (M_1)	0.31
NMI	0.73 (M_3)	0.73
CPU time (s)	52.23 (M_1)	$\mathbf{5.56}$
Memory requirement (s)	4.31 (M_1)	$\mathbf{3.06}$
ECNet		
MEASURE	**MKSC**	**ESC**
$\mathbf{ARI}_{\text{Mem}}$	0.76 (M_1)	$\mathbf{0.78}$
$\mathbf{Mod}_{\text{Mem}}$	$\mathbf{0.51}$ (M_1)	0.50
NMI	0.45 (M_1)	$\mathbf{0.61}$
CPU time (s)	51.83 (M_1)	$\mathbf{5.08}$
Memory requirement (s)	$\mathbf{2.57}$ (M_1)	2.79

Table 4.1: **Clustering results synthetic datasets**. For each evaluation measure the values represent an average over time (i.e. the mean value per snapshot). The NMI is calculated between consecutive partitions, while the other quality functions are evaluated in each snapshot. Regarding the MKSC model, between parenthesis we indicate with which amount of memory we obtained the best results (1, 2 or 3 snapshots of memory, that is M_1, M_2, or M_3). Concerning the datasets MG2 and MG3, MKSC is the best performer in terms of all the evaluation measures. Since an efficient implementation of the RBF kernel and a small training set are used, the runtime required by MKSC is rather small. On the other hand, we can notice how the CPU time required by MKSC to analyse the synthetic network data is much higher than the time needed by ESC. This can be explained by considering that in the ESC method it is not necessary to construct a kernel matrix, which in this case represents a computational burden for MKSC, where the community kernel has been used.

For what concerns the final partitioning the results are illustrated in figure 4.13. MKSC performs better than ESC in some periods and worse in other ones. Both obtain better performances than KSC in terms of the smoothed Modularity and the NMI between consecutive partitions.

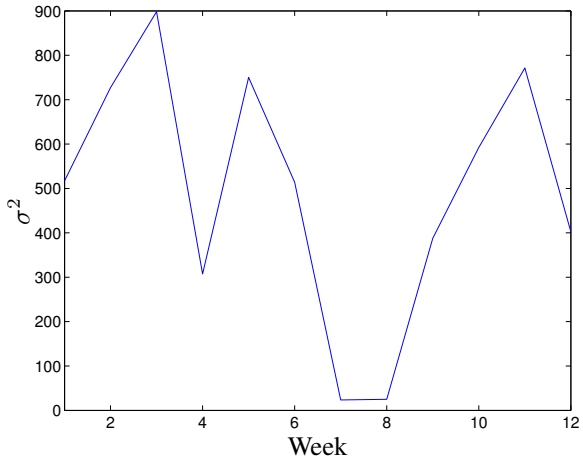


Figure 4.12: **Model selection CellphoneNet**. Optimal σ^2 over time for the cellphone network, related to MKSC. The number of clusters is $k = 2$, $\gamma = 1$ is an optimal value for all the snapshots.

4.4 Framework 2

In this section we describe the experimental results related to networks where nodes are entering and leaving over time and the number of communities is changing. Moreover we do not assume that the clustering results must be smoothed at each time stamp, but the smoothness is activated by tuning the regularization constant ν. Thus, in contrast to *framework 1* we now select ν and we fix $\gamma = 1$. The method that we use can be summarized in algorithm 9. In the experimental results reported later we utilize the Average Membership Strength (AMS) criterion [81] and the Modularity quality function [105, 76] to perform model selection. Moreover, the Fast and Unique Representative Subset Selection (FURS [94]) is used to select the training and validation sets, in the proportion of 15% and 30% respectively. Thus, the results are related to a single run of the MKSC algorithm since the subset selection performed by the FURS method is deterministic.

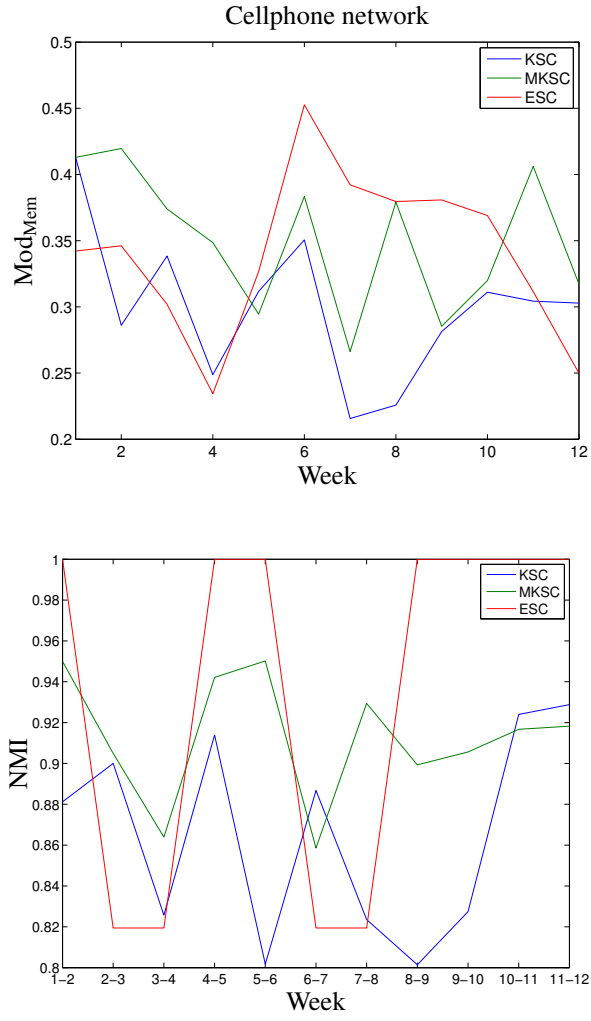


Figure 4.13: **Clustering results CellphoneNet**: performance of MKSC, KSC and ESC on the cellphone network in terms of the smoothed Modularity and the NMI between consecutive clustering results. Also in this case the models with temporal smoothness MKSC and ESC, in most of the time period, perform better than the static KSC.

Algorithm 9: Clustering evolving networks: framework 2 [77]

Data: Training sets $\mathcal{D} = \{x_i\}_{i=1}^{N_{\text{Tr}}}$ and $\mathcal{D}_{\text{old}} = \{x_i^{\text{old}}\}_{i=1}^{N_{\text{Tr}}}$, test sets $\mathcal{D}^{\text{test}} = \{x_m^{\text{test}}\}_{m=1}^{N_{\text{test}}}$ and $\mathcal{D}_{\text{old}}^{\text{test}} = \{x_m^{\text{test,old}}\}_{m=1}^{N_{\text{test}}}$, $\alpha_{\text{old}}^{(l)}$ (the $\alpha^{(l)}$ calculated for the previous M snapshots), positive definite kernel function $K : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ such that $K(x_i, x_j) \rightarrow 0$ if x_i and x_j belong to different clusters, kernel parameters (if any), number of clusters k, regularization constants γ and ν found using algorithm 7.

Result: Clusters $\{\mathcal{C}_1^t, \ldots, \mathcal{C}_p^t\}$, cluster codeset $\mathcal{CB} = \{c_p\}_{p=1}^k$, $c_p \in \{-1, 1\}^{k-1}$.

1 **if** *t==1* **then**
2 Initialization by using kernel spectral clustering (KSC [6]).
3 **else**
4 For every snapshot from $t-1$ to $t-M$ rearrange the data matrices and the solution vectors $\alpha^{(l)}$ as explained in Section 4.4.1.
5 Use steps $4-10$ of algorithm 8.
6 Match the actual clusters $\{\mathcal{C}_1^t, \ldots, \mathcal{C}_p^t\}$ with the previous partitioning $\{\mathcal{C}_1^{t-1}, \ldots, \mathcal{C}_q^{t-1}\}$ using the tracking scheme described in Section 4.4.2 and summarized in algorithm 10.
7 **end**

4.4.1 Object appearing and leaving over time

When performing the clustering for the actual data snapshot present at time t, two possible situations can arise: new data points are introduced or some existing objects may be disappeared. To cope with the first scenario, the rows of the old data matrices corresponding to the new points can be set to zero, as well as the related components of the solution vectors $\alpha_{\text{old}}^{(l)}$. In this way, when solving problem (4.2), the components of $\alpha^{(l)}$ related to the new objects have no influence from the past. On the other hand, data points that were present in the previous snapshots but not in the actual one, can simply be removed in order for the the previous M snapshots[1] $\mathcal{G}_{t-1}, \ldots, \mathcal{G}_{t-M}$ to have the same dimensions as the data matrix $\mathcal{G}_t$.

4.4.2 Tracking the clusters

Several events that happen during the evolution of clusters are continuing, shrinking, growing, splitting, merging, dissolving and forming of clusters. In order to recognize these circumstances a tracking algorithm that matches the partitions found at each time step is needed, like the ones proposed in [49] and [22]. In this realm, we introduce a tracking method that is based on a maximum weight matching mechanism [67], as depicted in algorithm 10.

[1] We remind that the memory M indicates the amount of past information to carry along when clustering the current data matrix.

Algorithm 10: Cluster Tracking Algorithm [77]

Data: At a given time stamp t, take the clustering information of time stamp $t-1$ and t i.e. C^{t-1} and C^t.

Result: A weighted directed network W_N^t tracking the relationship between the clusters at time stamps $t-1$ and t.

foreach $C_j^{t-1} \subset C^{t-1}$ **do**
  foreach $C_k^t \subset C^t$ **do**
    if *Nodes with labels* c_j^{t-1} *at* $t-1$ *have the label* c_k^t *at* t **then**
      Create temporary edge $v^t(j,k)$ between C_j^{t-1} and C_k^t.
      $n(j,k)$ = Number of nodes with labels c_j^{t-1} at $t-1$ which have the label c_k^t at t.
      Weight of the edge $v^t(j,k) = \frac{n(j,k)}{|C_j^t|}$
    end
  end
end
Keep the edge which has maximum weight (in case of multiple edges keep all) w.r.t. C_j^{t-1}.
Add this weighted edge to the graph W_N^t.
foreach $C_k^t \subset C^t$ **do**
  if C_k^t *is isolated* **then**
    Select the edge $v^t(j,k)$ which had maximum incoming weight w.r.t. C_k^t. Add this weighted edge to the graph W_N^t.
    /* This is done in order to prevent isolated nodes in W_N^t. */
  end
end

We generate a directed weighted network W_N^t from the clusters at two consecutive timestamps t and $t+1$. Thus, if we have T timestamps we generate a set $\mathcal{W}_\mathcal{N} = \{W_N^1, \ldots, W_N^{T-1}\}$ of directed weighted networks. Each directed weighted network W_N^t creates a map between the clusters at time stamp t i. e. C^t and time stamp $t+1$ i. e. C^{t+1}, which form the edges of the network. The weight $v^t(j,k)$ of an edge between two clusters is equivalent to the fraction of nodes in cluster C_j at time stamp t which are assigned to cluster C_k at time stamp $t+1$. An edge exists between two clusters C_j and C_k only if $v^t(j,k) > 0$. Thus, if the number of edges going out of a node of W_N^i is greater than 1 it indicates a split whereas if the number of edges entering a node is greater than 1 then it indicates a merge. In case $v^t(j,k) = 1.0$ then the cluster remains unchanged between the 2 time intervals t and $t+1$. In order to tackle the birth and death of clusters, we add a C_0 cluster for each time stamp t. For

the network W_N^t, if C_0^t is isolated then no new clusters were generated at time t and if C_0^{t+1} is isolated then none of the clusters present at time stamp t dissolved in the next snapshot. However, if we have outgoing edges from C_0^t then there was birth of new clusters. Similarly, if we have incoming edges to C_0^{t+1} in W_N^t then some clusters dissolved at time t. In Figure 4.14 an example of the matching mechanism is given.

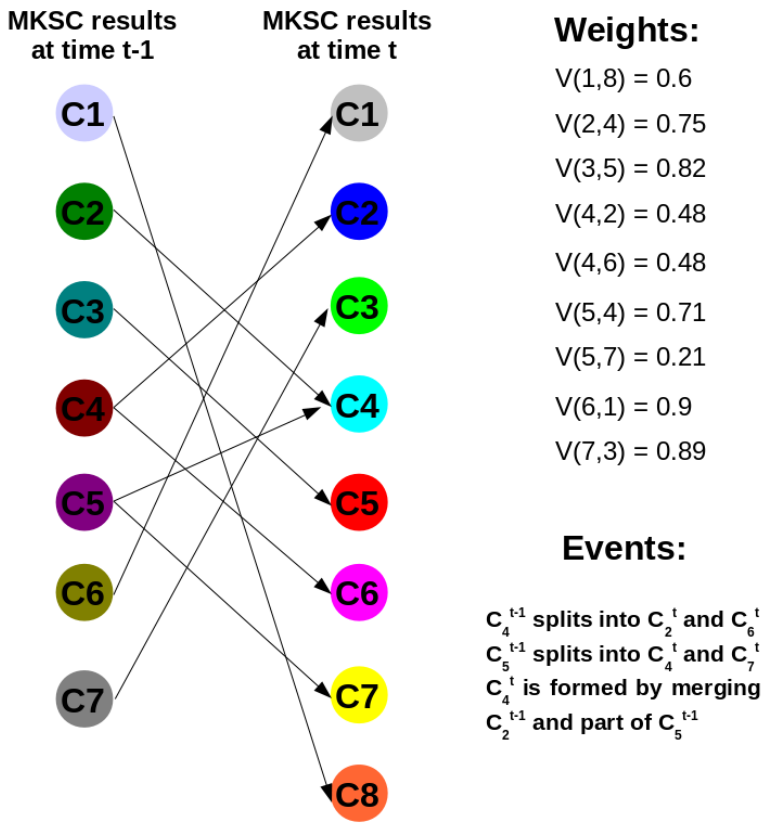


Figure 4.14: **Illustrative example of the cluster matching procedure**. Since the labelling at each time step is arbitrary, the clusters found by MKSC at successive time stamps have to be matched to keep track of their evolution. In this specific case, for instance, it is clear that cluster 3 at time t should be labelled as cluster 7, cluster 5 as cluster 3 and so on.

4.4.3 Description of the data sets

The artificial benchmarks consist of evolving networks generated by the software related to [49]:

- **MergesplitNet**: an initial network of 1000 nodes formed by 7 communities evolves over 5 time steps. At each time step there are 2 splitting events and 1 merging event. The number of nodes remains unchanged.
- **BirthdeathNet**: a starting network with 13 communities experiences at each time one cluster death and one cluster generation, while the number of nodes decreases from 1000 to 866 as time increases from 1 to 5.
- **HideNet**: at each time step a community of an initial network with 1000 nodes and 7 communities dissolves, and the number of nodes also varies over time.

To analyse these data we use the cosine or normalized linear kernel defined as $\Omega_{ij} = \frac{x_i^T x_j}{||x_i|| ||x_j||}$. So at each time step, when performing model selection, we only have to detect the optimal number of clusters k and tune the smoothness parameter ν.

The real-world dataset is the **CellphoneNet** network described in Section 4.3.2. However, unlike for *framework 1* now we consider the complete dataset of 46 weeks since we can handle nodes entering and leaving across time. In total there are 94 nodes, but not all the nodes are present in every snapshot. In particular, the smallest network comprises 21 people and the largest has 88 nodes.

4.4.4 Experiments

In this Section an exhaustive study about the ability of the MKSC algorithm to perform dynamic clustering is performed. First we discuss the model selection issue: different criteria are contrasted and the outcomes are analysed. Then the clustering results are evaluated according to a number of cluster quality measures and the MKSC method is compared with the AFFECT algorithm [139] and the ESC [27] technique. Finally, a simple 3D visualization of the clusters evolution over time is presented.

Model selection

Here the AMS criterion [81] and the Modularity criterion [76] are compared for tuning the number of clusters k and the smoothness parameter ν.

The results related to the selection of k for the synthetic networks are depicted in Figures 4.15-4.17. In general Modularity, AMS or both are able to suggest the

right number of communities or a partitioning close to the ground truth. The results concerning the *CellphoneNet* are depicted in Figure 4.18. Only AMS mostly selects $k = 2$ along all the time period, in agreement with the ground truth suggested in [39] and in [127]. As mentioned before, in the cited works it has been proposed to group students and staff at MIT according to their belonging to the Sloan business school or being co-workers in the same building.

Regarding the smoothness parameter ν, the results related to the *CellphoneNet* are plotted in Figure 4.19. We can notice how the regularization constant has some small peaks around important dates like beginning of fall and winter term and end of winter term. Thus, it seems that ν is behaving as a kind of change indicator measure.

Evaluation of the results

In this Section the clustering results are evaluated according to the Adjusted Rand Index (ARI [58]) when the true memberships are available and the smoothed Conductance (Cond_Mem).

MergeplitNet		
MEASURE	**MKSC**	**AFFECT [139]**
ARI	$\mathbf{0.90 \pm 0.03}$ (MOD)	0.73 ± 0.01 (MOD)
Cond_Mem	0.0112 ± 0.0001 (AMS)	$\mathbf{0.0038 \pm 0.0005}$ (SIL)
BirthdeathNet		
MEASURE	**MKSC**	**AFFECT [139]**
ARI	$\mathbf{0.80 \pm 0.02}$ (MOD)	0.76 ± 0.03 (MOD)
Cond_Mem	$\mathbf{0.036 \pm 0.003}$ (AMS)	0.052 ± 0.002 (MOD)
HideNet		
MEASURE	**MKSC**	**AFFECT [139]**
ARI	$\mathbf{0.97 \pm 0.01}$ (AMS, MOD)	0.85 ± 0.03 (MOD)
Cond_Mem	0.011 ± 0.001 (AMS, MOD)	$\mathbf{0.005 \pm 0.001}$ (SIL)

Table 4.2: **Clustering results synthetic datasets**. Both ARI and Cond_Mem values represent an average over time, i.e. 5 snapshots. Moreover, for each snapshot the smoothed Conductance is obtained by taking the mean over 5 possible values of the parameter η in the range $[0, 1]$. Between parenthesis we indicate with which model selection criterion the related result has been obtained. Although the model selection problem is out of the scope of [139], the authors provide two possible way of tuning the number of clusters, using Silhouette (SIL) or Modularity (MOD). All the results are statistically significant according to a Student t-test for the comparison of two means, with p-value ł0.05.

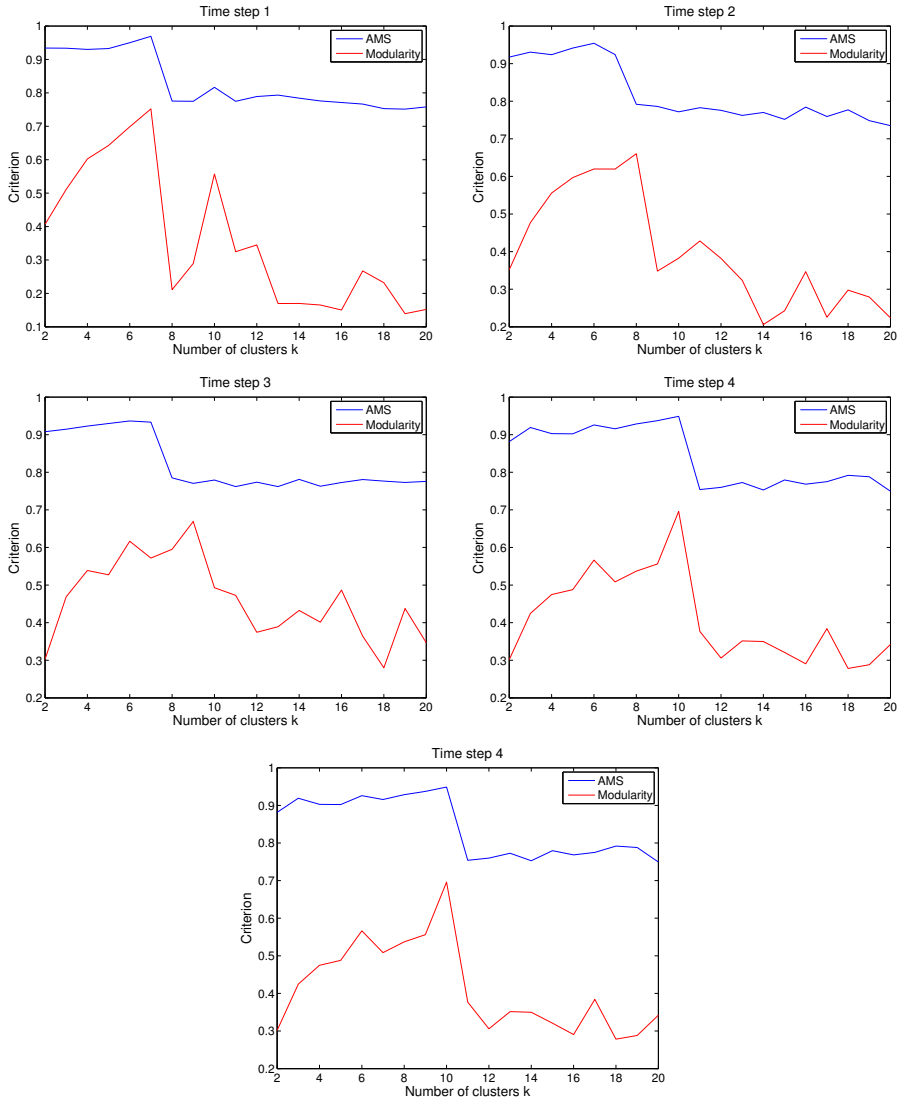


Figure 4.15: **MergesplitNet: selection of the number of clusters**. The true number of clusters over time is 7, 8, 9, 10, 11 at times $T_1, \ldots, T_5$. Modularity (red) perfectly detects the right number of clusters in all the time steps, while AMS (blue) selects 7, 6, 6, 10, 9. In this case ν has been tuned using Modularity and is zero for all time steps except $t = T_4$, where $\nu = 0.1$. This indicates that at each time from $t = T_1$ to $t = T_3$ the current network is not enough different from the previous one to require smoothing. This can be noticed also by looking at the 3D visualization shown in Figure 4.23.

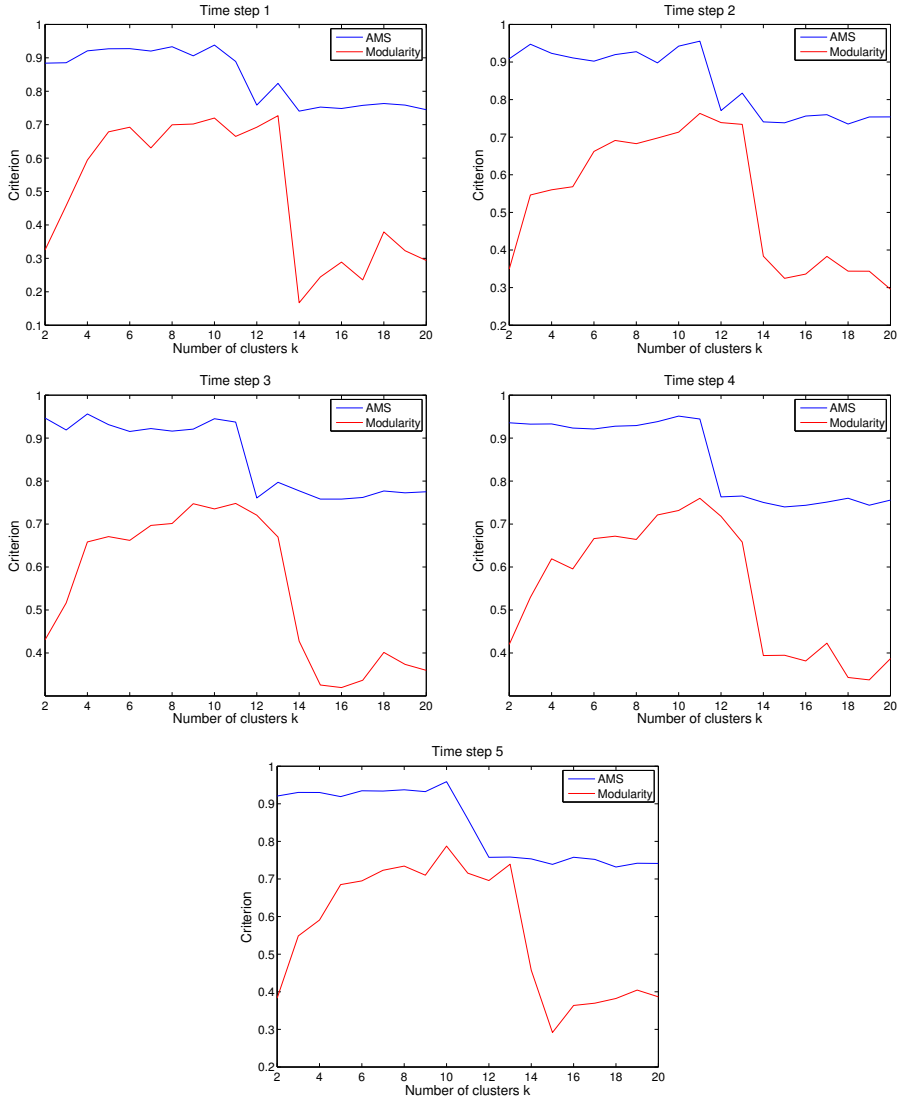


Figure 4.16: **BirthdeathNet: selection of the number of clusters**. The true number of clusters over time is $k = 13$ in every snapshot. Modularity (red) detects 13, 11, 11, 11, 10. AMS (blue) selects 10, 11, 3, 10, 10: however, if we look carefully at the figure, we can notice some peaks also at $k = 13$.

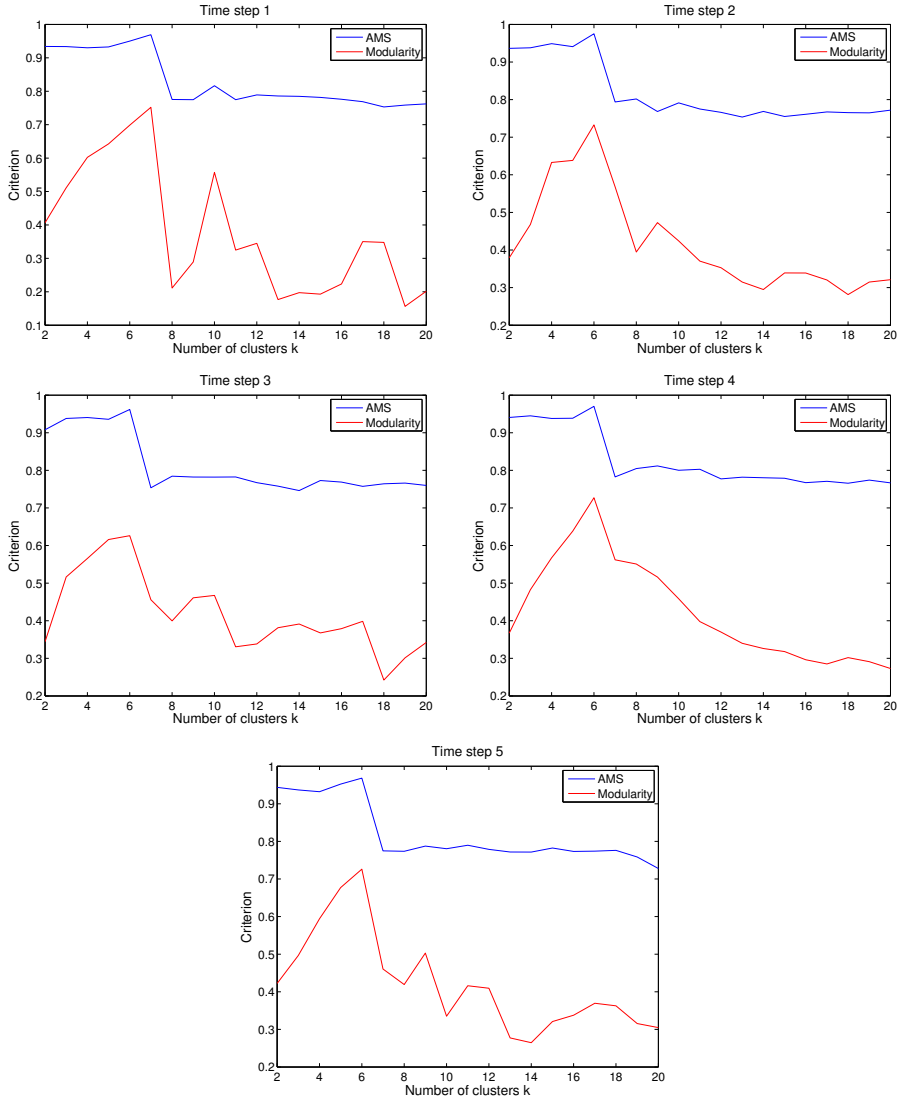


Figure 4.17: **HideNet: selection of the number of clusters**. The true number of clusters over time is 7, 6, 6, 6, 6. Both Modularity and AMS are able to recognize the right sequence.

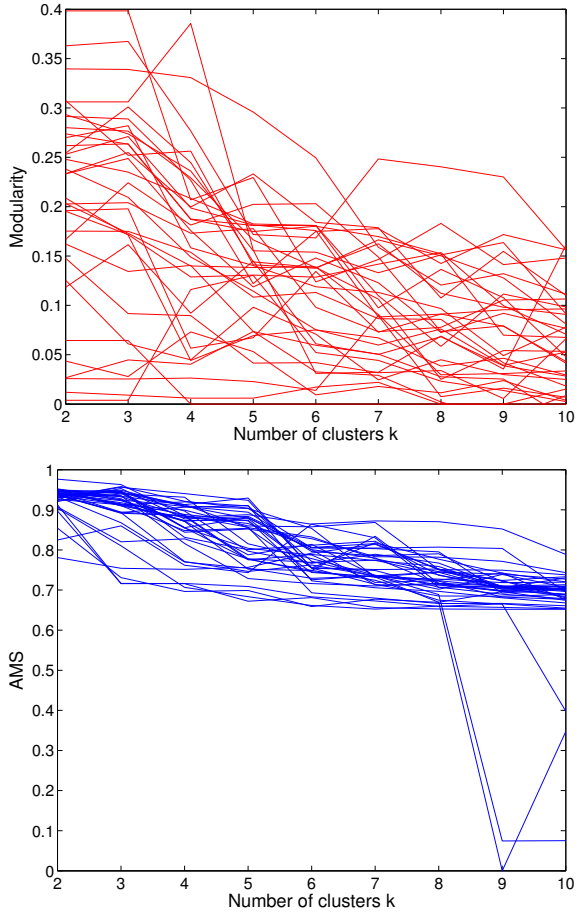


Figure 4.18: **CellphoneNet: selection of the number of clusters**. For this network it has been suggested to consider $k = 2$ as the ground-truth for the entire period of 46 weeks [39]. This is due to the fact that the people representing the nodes of the network belong to 2 different departments at MIT. However we have noticed that the network does not have a clear community structure in every snapshot (for instance in some snapshots the maximum value of the Modularity quality function for $k = 2$ approaches zero). Moreover, both Modularity (top) and AMS (bottom) do not always select $k = 2$: the former mainly detects $k = 4, 5, 6$, and the latter $k = 2$ but also $k = 3$.

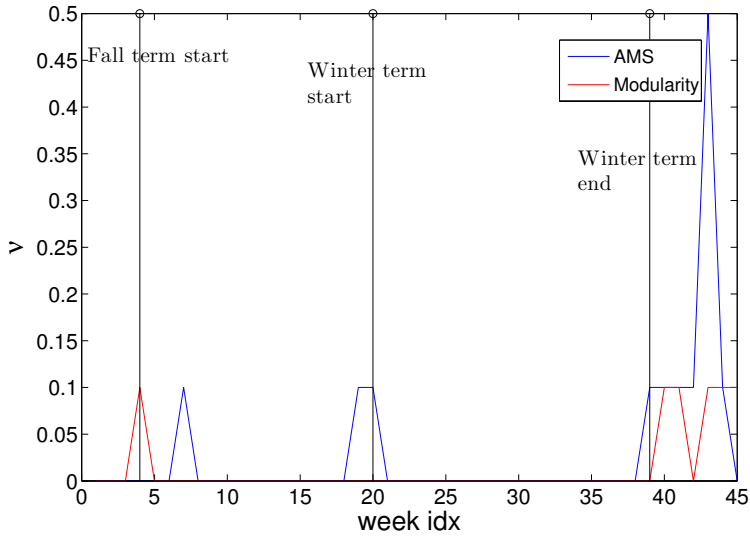


Figure 4.19: **CellphoneNet: selection of the smoothness parameter** ν. The regularization constant ν selected by AMS (blue) and Modularity (red). In both cases some peaks are present around important dates which are labelled in the plot.

CellphoneNet			
MEASURE	**MKSC**	**AFFECT [139]**	**ESC [27]**
ARI	$\mathbf{0.861 \pm 0.051}$ (AMS)	0.763 ± 0.001	–
RI	$\mathbf{0.943 \pm 0.040}$ (AMS)	0.893	0.861
Cond_Mem	$\mathbf{0.0035 \pm 0.0001}$ (AMS)	0.0048 ± 0.0001	–

Table 4.3: *Clustering results CellphoneNet*. We compare MKSC with AFFECT and ESC by reporting the results shown in Tables 5 of [139]. Regarding ESC, the mean between the best results related to the PCQ and PCM frameworks are considered. Moreover, for AFFECT and ESC the number of clusters has been fixed to $k = 2$, while it is fine tuned in case of MKSC. The best performer is MKSC, according to a Student t-test for the comparison of two means with significance level of 0.05 (p-value $\nmid 0.05$).

The MKSC algorithm is compared with Adaptive Evolutionary Clustering (AFFECT [139]) in case of synthetic data and AFFECT and ESC (Evolutionary Spectral Clustering [27]) for *CellphoneNet*. In all the datasets MKSC produces clustering results closer to the ground truth memberships (higher ARI), as it is shown in Tables 4.2 and 4.3. Moreover, the differences between the results of the compared methods are all statistically significant, according to a Student t-test for the comparison of two means with significance level of 0.05. In Figure 4.15 an example of the partition obtained by MKSC for the *mergesplitNet* network is shown.

4.4.5 Visualizing the clusters evolution

In this Section we show the results obtained by our tracking mechanism summarized in algorithm 10 on the networks described in Section 4.4.3. We use the first 3 dual solution vectors of problem (4.2), i.e. $\alpha^{(1)}$, $\alpha^{(2)}$, $\alpha^{(3)}$, to visualize the clusters evolution in 3D. In order to explicitly show the growth and shrinkage events we plot the clusters as spheres, centered around the mean of all the points in that cluster. The radius is equivalent to the fraction of points belonging to that cluster at that time stamp. Each sphere is given a unique colour at time stamp $t = 1$. As the clusters grow or shrink the size of the sphere changes. In case of a split the colour and label of that cluster is transferred to all the clusters obtained as a result of the split. In case of a merge we assign the average colour of the clusters which merge together at time t to the new cluster at time $t + 1$. In case of birth of a new cluster we allocate it a new colour and all the nodes which have disappeared at time interval t are depicted as a blue-coloured sphere centered at the origin (dump).

The visualization of the clusters is shown in Figures 4.20-4.24. For all the synthetic networks (although not perfectly) we are able to grasp the main events occurring at each time step. For instance, in case of the *MergesplitNet*, we can appreciate a certain change in the data at time $t = T_4$. In fact, at this time stamp, the tuning algorithm 7 selected $\nu = 0.1$, while in the other time steps $\nu = 0$. Thus, the memory effect got activated to smooth the clustering results. Regarding the *CellphoneNet*, thanks to the proposed visualization tool, it is possible for the user to have an idea of the clusters evolution discovered by the MKSC model. Finally, concerning the *MergesplitNet* dataset, in figure 4.26 together with the classical representation of the community structure of a network (top) and the proposed 3D illustration in the $\alpha^{(l)}$ space (center), we show another possible kind of visualization (bottom). The latter depicts the affinity matrix of the network W_N^t used by our tracking scheme across time.

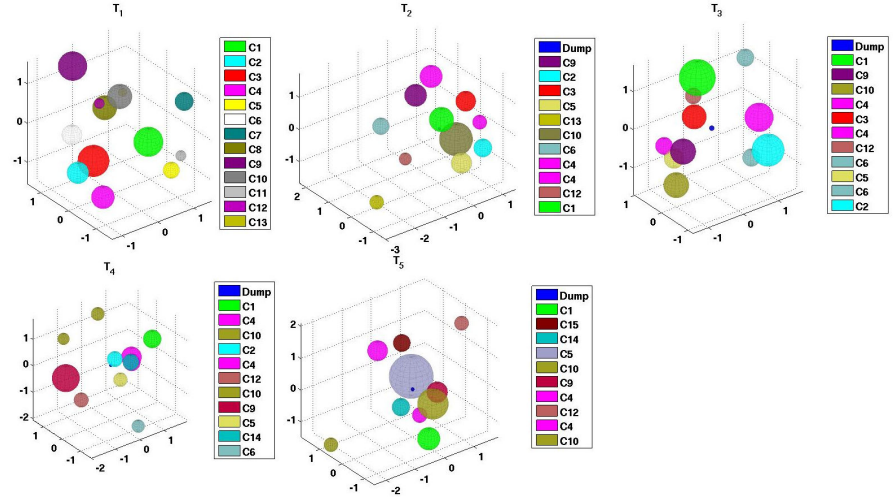


Figure 4.20: The *BirthdeathNet* dataset has 13 clusters at time stamp T_1. At time stamp T_2 cluster $C4$ splits into 2 clusters, clusters $C6$ and $C7$ merge to cluster $C7$, clusters $C8$ and $C10$ merge to cluster $C10$ and $C12$ and $C13$ merge to cluster $C12$. At time stamp T_3 cluster $C10$ and $C13$ merge to cluster $C10$, cluster $C6$ splits into 2 clusters. At time stamp T_4, a new cluster $C14$ appears, cluster $C10$ splits into 2 clusters, cluster $C3$ and $C9$ merge to cluster $C9$ and one of the 2 splits of cluster $C6$ dissolves. At the final time interval cluster $C2$ dissolves, cluster $C15$ appears, cluster $C2$ and $C5$ merge to cluster $C5$. At each time step except T_1 the fraction of nodes which disappeared is represented by blue-coloured sphere. The size of the sphere show the growth of shrinkage of clusters over time.

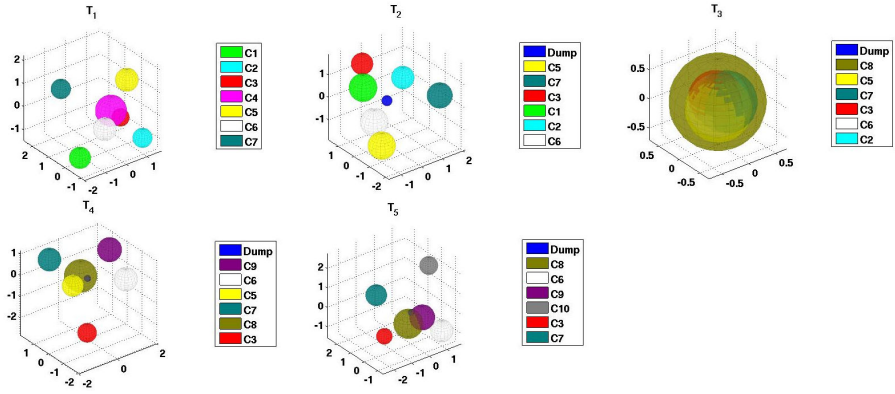


Figure 4.21: The *HideNet* dataset has 7 clusters at time stamp T_1. Cluster $C4$ dissolves at time stamp T_2. Cluster $C8$ is born at time stamp T_3 and cluster $C1$ fades. Cluster $C2$ dies and a new cluster $C9$ is created at time stamp T_4. In the final phase, a new cluster $C10$ is generated and cluster $C5$ is dumped. The size and position of the spheres change at each time interval showing the growth and shrinkage of the clusters.

Figure 4.22: **Evolution of clusters found by the MKSC algorithm for *BirthdeathNet* and *HideNet* datasets.**

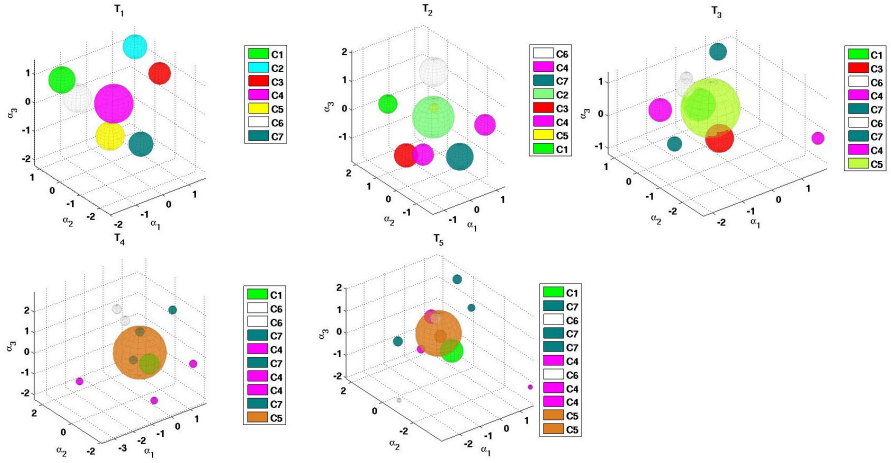


Figure 4.23: The *MergesplitNet* dataset has 7 clusters at time stamp T_1. At time stamp T_2 cluster $C4$ splits into 2 clusters, major part of cluster $C5$ merges with cluster $C2$. At time stamp T_3 cluster $C6$, cluster $C7$ splits into 2 clusters and clusters $C2$ and $C5$ merge to cluster $C5$. At time stamp T_4 clusters $C4$ and $C7$ further split to have 3 clusters each. Cluster $C3$ combines with cluster $C5$. In the final time interval cluster $C5$ splits into 2 clusters.

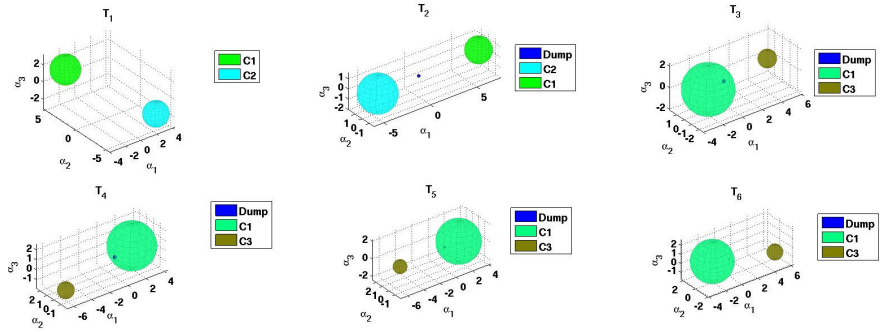


Figure 4.24: The *CellphoneNet* dataset has 2 clusters at time stamp T_1. Some nodes leave the network at time stamp T_2. At time stamp T_3 one part of cluster $C2$ is dumped, another part merges with a new cluster $C3$. After this the clusters remain more of less constant till interval T_6. For the complete evolution of the communities along all the time steps see the video present in the supplementary material.

Figure 4.25: **Visualization of the communities found over time by the MKSC algorithm for the *MergesplitNet* and *CellphoneNet* evolving networks.**

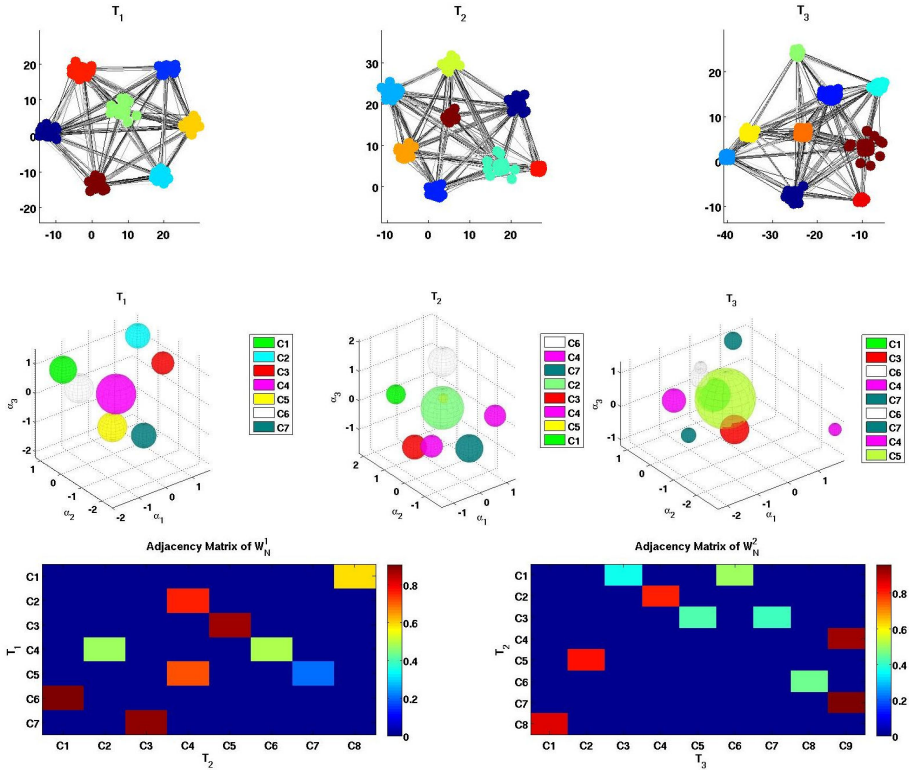


Figure 4.26: **MergesplitNet: three possible ways of illustrating the network evolution**. **(Top)** standard plot of nodes and edges **(Center)** 3D plot in the $\alpha^{(l)}$ space **(Bottom)** Affinity matrix of the network W_N^t constructed by our tracking mechanism. For simplicity, only the first three time steps are considered.

4.5 Flexibility of MKSC

For the analysis of the networks considered in the previous sections we used Modularity or its smoothed version in order to select the optimal parameters for KSC and MKSC. As explained in the appendix, Modularity suffers from some drawbacks like the resolution limit and the high number of local optima [68]. These limitations, however, do not represent an issue in our framework for several reasons:

- the MKSC model described in equation (4.1) is quite general (Modularity is not explicitly optimized as in other algorithms like for example [19], [30])

- Modularity (and its smoothed version) has been used only at the model selection level. However, our framework is quite flexible and allows to plug-in during the validation phase any other quality measure

In order to better understand these issues we present a further analysis based on the Conductance (see appendix). Moreover, in the same way as for BLF, Modularity and AMS we can define the smoothed Conductance Cond_{Mem} of the partition related to the actual snapshot $\mathcal{G}_t$ of an evolving network as:

$$\text{Cond}_{\text{Mem}}(X_\alpha, \mathcal{G}_t) = \eta \text{Cond}(X_\alpha, \mathcal{G}_t) + (1-\eta)\text{Cond}(X_\alpha, \mathcal{G}_{t-1}). \tag{4.6}$$

In table 4.4 we show the mean smoothed Conductance over-time for the 3 networks under investigation related to the partitions found by MKSC, ESC, KSC and Louvain method (LOUV) [19]. The Louvain method is based on a greedy optimization of Modularity and has a runtime that increases linearly with the number of nodes. From the table we can draw the following considerations:

- the methods with temporal smoothness (MKSC and ESC) achieve an equal or better score than the static methods (KSC and LOUV)
- the Louvain method gives the worst results in terms of Conductance. This is not surprising since it is biased toward partitions maximizing the Modularity, which may not be good in terms of Conductance. On the other hand, as already pointed out, MKSC does not suffer of this drawback since Modularity is used only at the validation level. Moreover for model selection every quality function could in principle be used. It is a user-dependent choice.

4.6 Conclusions

In this chapter the problem of community detection in evolving networks has been addressed. We have introduced a novel method named kernel spectral clustering with memory (MKSC), which is designed to incorporate the temporal smoothness of the clustering results over time. Two main frameworks have been devised: in the first one the smoothness is imposed by fixing the regularization constant $\nu = 1$ and the amount of smoothness is tuned by changing the memory M. In the second framework the smoothness get activated automatically only when it is required, that is when an important change in the data structure occurs. As a consequence, ν can also be used as a change detection measure. We have also proposed a tracking mechanism to detect important events characterizing the evolution of the communities over time. Moreover, two visualization tools based on a 3D embedding in the space of the dual solution vectors and on the network constructed by the tracking algorithm,

SwitchingNet	$\mathbf{Cond}_{\mathrm{Mem}}$
MKSC	**0.0020**
ESC	0.0022
KSC	0.0022
LOUV	0.0024
ECNet	$\mathbf{Cond}_{\mathrm{Mem}}$
MKSC	**0.0050**
ESC	0.0051
KSC	0.0051
LOUV	0.0184
CellphoneNet	$\mathbf{Cond}_{\mathrm{Mem}}$
MKSC	**0.0019**
ESC	0.0042
KSC	0.0056
LOUV	0.0153

Table 4.4: **Results network data, Conductance**. Average smoothed Conductance over time for some synthetic graphs and the real network considered in this chapter. ESC, KSC, MKSC and the Louvain Method (LOUV) applied separately in each snapshot are compared.

respectively, are devised. On a number of synthetic and real-life networks we have shown how MKSC is able to produce high-quality partitioning in terms of classical measures such as Modularity and Conductance, and their smoothed version. Finally, comparisons with state-of-the-art methods like evolutionary spectral clustering (ESC), adaptive evolutionary clustering (AFFECT) and the Louvain method (LOUV) have shown that MKSC has a competitive performance.

Chapter 5

Predicting Maintenance of Industrial Machines

This chapter describes an application of KSC to predictive maintenance. In general, an accurate prediction of forthcoming faults in modern industrial machines plays a key role in reducing downtime, increasing the safety of plant operations, and minimizing manufacturing costs. We use clustering on the sensor data collected from a packing machine to recognize in advance when the machine is entering a faulty regime. In this way an early warning can be raised and optimal maintenance actions can be performed.

In this framework we assume stationarity, in the sense that a clustering model is trained off-line in order to distinguish between normal operating condition and abnormal situations. Then we use the model in an on-line fashion via the out-of-sample extension property to recognize these two regimes. Moreover, moderated outputs that mimic the degradation process affecting the machine are also provided. Overall, this improves the interpretability of the results and allows us to have further insights in the problem at hand.

5.1 Problem Description

In industrial processes fault detection, isolation and diagnosis ensure product quality and operational safety. Traditionally, four ways to deal with sensory faults have been used [133],[134],[135]: corrective maintenance, preventive maintenance, manual predictive maintenance, and condition-based maintenance. The first type is performed

only when the machine fails, it is expensive and safety and environment issues arise. Preventive maintenance is based on periodic replacement of components. The rough estimation of parts lifetime causes a non-optimal use of parts, and possible unexpected failures can still occur (with downtime, safety and environmental consequences). In predictive maintenance machines are manually checked with expensive monitoring hardware (termography, motor health, bearing health). In this case the components are replaced according to their real status, but the operations are labour intensive and prone to human errors. Condition-based maintenance is receiving increasing attention due to its many advantages. Machines status is automatically collected and centrally analysed, and maintenance is planned based on the results of the analysis. The continuous monitoring of machine parts leads to reliable and accurate lifetime predictions, and maintenance operations can be fully automated and implemented in a cost efficient way. With the development of information and sensor technology many process variables in a power plant can be sampled, like temperature, pressure, flow rate etc. These measurements give information on the current status of a machine and can be used to predict the faults and plan an optimal maintenance strategy. When a component starts degrading, the related sensor reading shows a deviation from its normal behaviour and this can indicate an incoming failure of the component. So far process models based on the sensor data have been constructed by using exponentially weighted moving average, cumulative sum, principal component analysis (PCA), just to name the most widely used methods [65], [28]. Moreover the problem of discovering the incoming faults can be seen as a special case of outlier detection, since an outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism. In this field supervised, semi-supervised and unsupervised methods are employed [66]. In the study that we will discuss in the next sections, we are given sensor data collected from the sealing jaws of a packing machine. Since the data are highly unbalanced for supervised learning, we use clustering for identifying in advance when the machine enters critical conditions.

5.2 Materials and Methods

The data are collected from a Vertical Form Fill and Seal (VFFS) machine used to fill and seal packages mainly in the food industry. An illustration of such a machine is given in figure 5.1. The VFFS machine supplies film from a roll which is formed into a bag over the vertical cylinder. Sealing jaws close the bag at the bottom before it is filled. At the end of the cycle, the bag is sealed and cut off with a knife. From previous experimental studies the dirt accumulation on the sealing jaws was observed to strongly affect the process quality. For this reason, in the experiments described here the jaws were monitored to predict in advance the maintenance actions.

Maintenance consists of stopping the machine and cleaning the sealing jaws. A total of three experiments have been performed, resulting in three datasets:

- DS_I: this dataset consists of 771 events and 3 external maintenance actions. An event is related to a particular processed bag and takes place every two seconds (i.e. the sampling frequency is $0.5Hz$). Each event is associated with a 150-dimensional accelerometer signal, that is each signal is a vector of length 150 (see top of figure 5.2).
- DS_II: it contains a total of $11\,632$ processed bags and 15 maintenance actions. Here the vibration signals used to monitor the dirt accumulation in the jaws are 190-dimensional time-series (as shown in the center of figure 5.2). This is due to a different setting of the data acquisition system for this experiment.
- DS_III: there are $3\,519$ processed bags and 11 maintenance actions, as depicted at the bottom side of figure 5.2.

Moreover, in order to catch the ongoing deterioration process of the jaws we need to use historical values of sealing quality in our analysis. For this purpose we apply a windowing operation on the data, as illustrated in figure 5.3. Then we develop off-line a KSC model (see chapter 2) fed with the concatenation of a certain number of accelerometer signals. Once properly trained the model will be used on-line to detect the different behavioural regimes experienced by the running machine.

5.3 Results

In this section it is shown how KSC can be used to perform just-in-time maintenance, not too early to take full advantage of component lifetime but also not too late to avoid catastrophic failures and unplanned downtime.

Above all, model selection has been performed. In total we have 3 parameters to determine: the window size (i.e. the number of accelerometer signals to concatenate), the number of clusters k and the RBF kernel parameter σ. According to the BLF criterion the optimal window size is 40 and the optimal number of clusters is $k = 2$ for the three data-sets, while σ is dataset dependent. In figure 5.4 an illustration of the tuning procedure related to the first dataset is given.

After building an optimal KSC model 2 regimes have been identified, where one of them can be interpreted as normal behaviour and the other as critical conditions (need of maintenance). Moreover, a probabilistic interpretation of the results is also provided, which better describes the degradation process experienced by the sealing

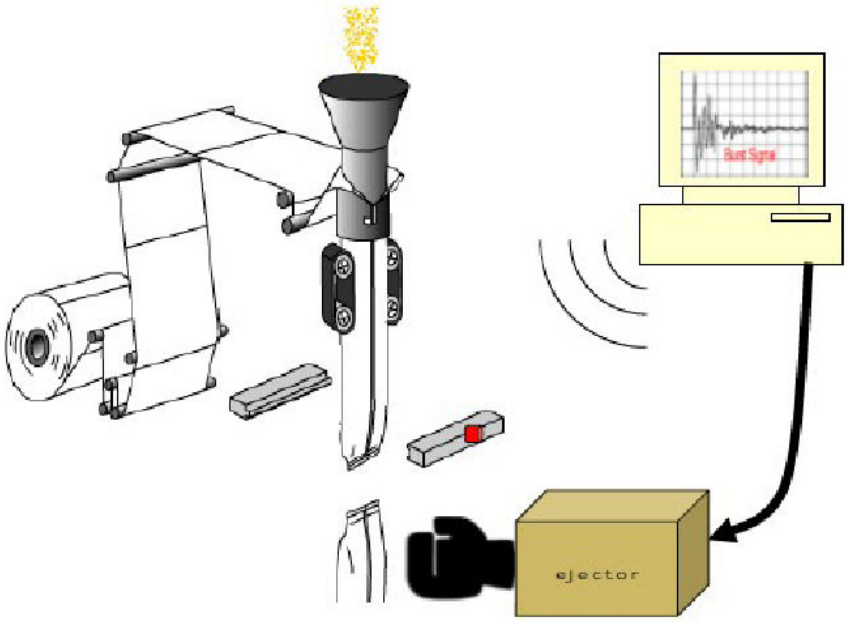


Figure 5.1: **Vertical Form Fill and Seal (VFFS) machine.** Seal quality monitoring in a packing machine.

jaws of the packing machine. Finally, we compare KSC with K-means, which is the most popular method used in industrial applications [17].

5.3.1 Hard Clustering

In figure 5.5 the KSC prediction for the dataset DS_I is shown. We can interpret one of the clusters as normal conditions and the other as maintenance cluster. Notice that the KSC model is able to predict some minutes in advance the maintenance actions before they are actually performed by the operator. Concerning the dataset DS_II we can draw the same comments: KSC is very accurate in predicting the worsening of the packing process around the actual maintenance events (see top of figure 5.6). Finally, figure 5.7 illustrates the results on dataset DS_III. In this case KSC predicts the need of maintenance also in zones where maintenance has not been really performed. As we will see later, surprisingly it would have been more logical to perform maintenance as suggested by KSC and not as actually done by the operator.

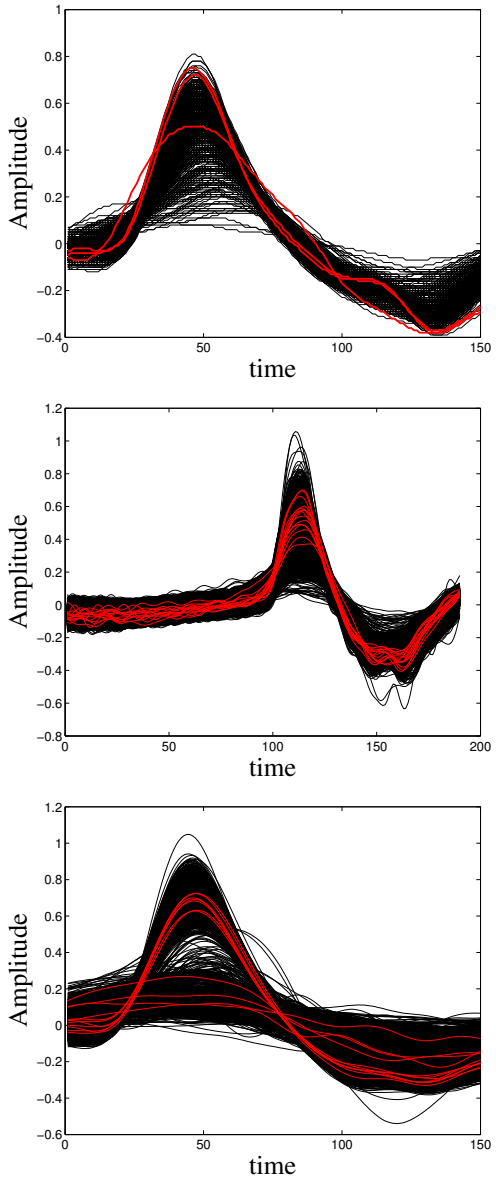


Figure 5.2: **Vibration signals**. **Top:** Vibration signals for the data-set DS_I. **Center:** Vibration signals for the data-set DS_II. **Bottom:** Vibration signals for the data-set DS_III. The signals corresponding to maintenance actions are depicted in red.

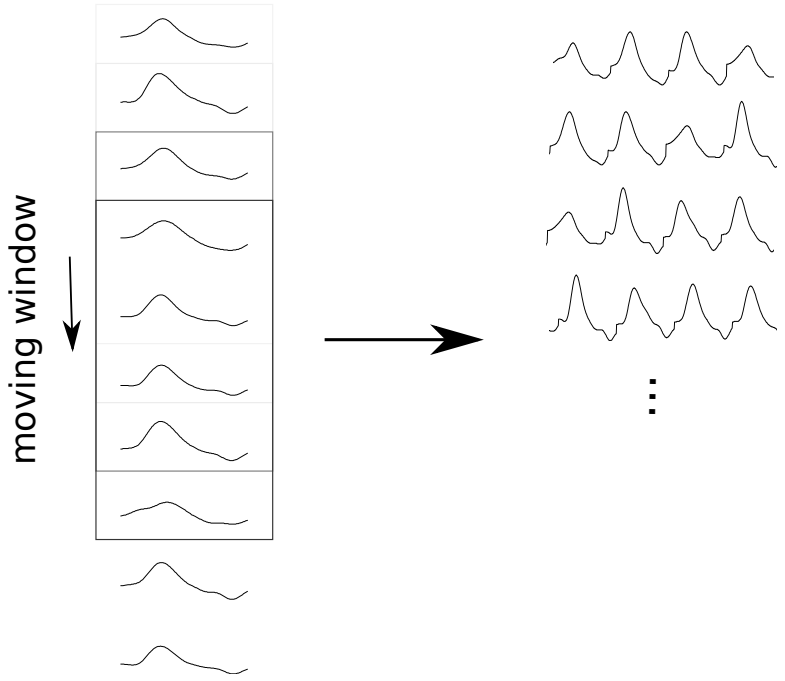


Figure 5.3: **Concatenation of vibration signals**. After the windowing operation, each data-point is now a time-series of dimension $d = 40 \times 150$ for datasets DS_I and DS_III and $d = 40 \times 190$ for the dataset DS_II.

5.3.2 Probabilistic Output

In the previous section we demonstrated the effectiveness of KSC in predicting in advance the maintenance events. Nevertheless the predicted output is binary (it goes suddenly from normal operation to maintenance). An output of this form does not provide a continuous indicator of the incoming maintenance actions. To solve this issue we can use the latent variable $e(x)$ instead of the binarized clustering output $\text{sign}(e(x))$ (see section 2.2.3). The latent variable provides a more informative output which can be analysed in order to produce a better prognostic output. Since the range from which it takes a value depends on many factors (e.g., the kernel and its parameters, the number of training data points), the interpretability might be difficult. To improve it, values are normalized between 0 and 1. This transformation is based on the structure of the latent variable space. As mentioned in section 2.2.4, in this space every cluster is ideally represented as a line. The tips of the lines can be considered prototypes of their cluster, since they have more certainty to belong to

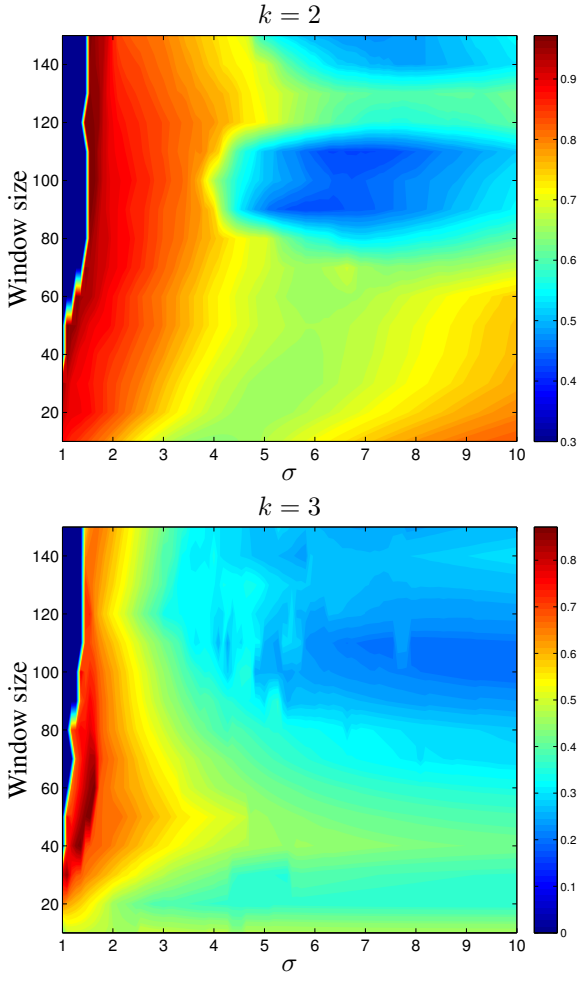


Figure 5.4: **Model selection.** Tuning surfaces for the first dataset, only the results for $k = 2$ and $k = 3$ are shown. If we consider more clusters ($k > 3$) the maximum value of the BLF decreases. The outcome is similar for the datasets DS_II and DS_III.

it because they are further from the decision boundaries[1] [5]. Thus, the Euclidean distance from every point to the cluster prototype can be seen as a confidence measure of the cluster membership. The transformed latent variable is depicted at the bottom of figure 5.5 (dataset DS_I), figure 5.6 (dataset DS_II) and figure 5.7 (dataset DS_III). The value can be considered as a soft membership or probability to maintenance [14]. The latter increases as the number of faulty bags in the window increases. The value can decrease since the window can move onto zones with good seals after a period of bad seals. This is probably due to a self-cleaning mechanism. Maintenance is predicted when the probability reaches the value 1.

For datasets DS_I and DS_II it can be noticed how KSC is able to discover from the vibration signals registered by the accelerometers the dirt accumulation in the jaws that leads to the maintenance actions. This is very surprising because clustering is an unsupervised technique, and thus does not make use of any information on the location of the maintenance actions (like it occurs for classification).

For what concerns dataset DS_III, also regions where no maintenance actions have been really performed appear associated with high probability to maintenance. Before continuing the discussion, we should mention that in the third experiment the machine is also equipped with a thermal camera, which directly measures the dirt accumulation in terms of number of hot area pixels in the acquired images (with hot we mean that the temperature of the sealing jaws is above a user-defined threshold). Then, by rescaling the probability to maintenance such that it varies in the same range as the measured degradation, we can make a comparison between the two. This is illustrated in figure 5.8. We can recognize similar patterns, meaning that KSC was able to catch the real degradation process also in this last experiment, although in a genuinely unsupervised manner. To conclude, probably the operator should have performed the maintenance operations in a different way, being coherent with his behaviour in the first two experiments.

5.4 Comparison with K-means

K-means clustering is a standard method for finding clusters in a set of data-points. After choosing the desired number of cluster centers, the K-means procedure iteratively moves the centers to minimize the total within cluster variance. K-means has several drawbacks:

- the results are strongly influenced by the initialization

[1] However, as we discussed in chapter 2, this procedure is valid only when the amount of overlap between the clusters is negligible. When it is not, it is advisable to use the methodology incorporated in SKSC.

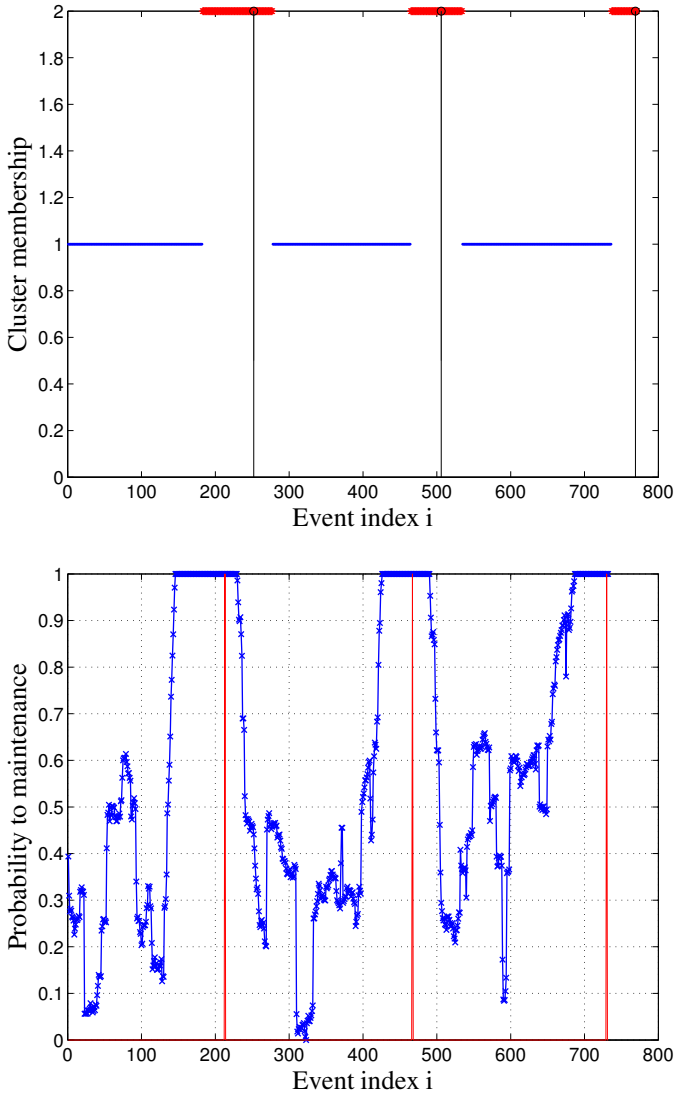


Figure 5.5: **KSC results dataset DS_I. Top:** Hard clustering results for the whole data-set. Cluster 2 represents predicted maintenance events. The vertical black lines show the true maintenance. **Bottom:** Soft clustering results in terms of probability to maintenance.

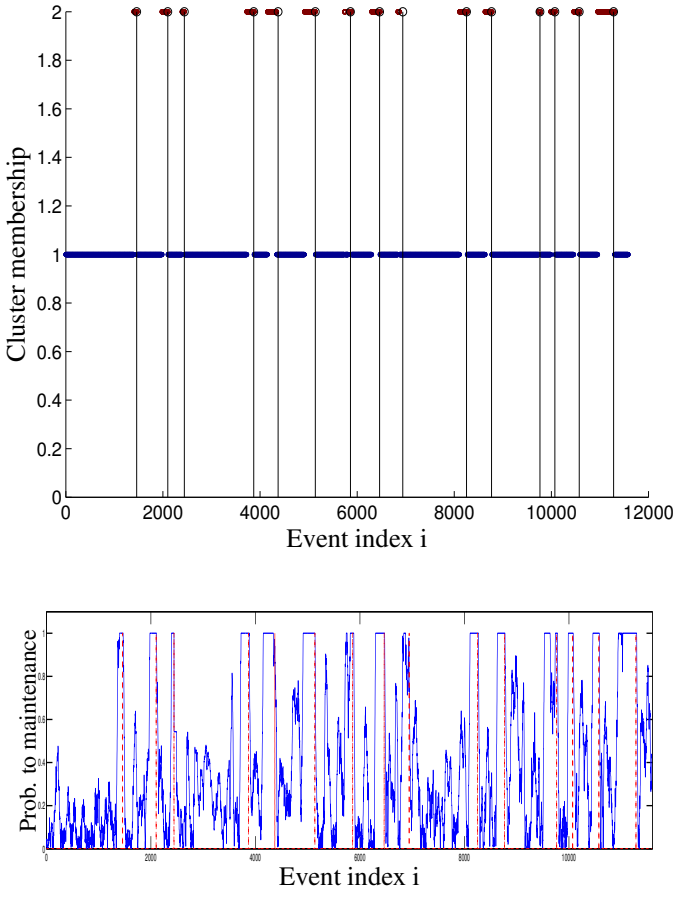


Figure 5.6: **KSC results dataset DS_II. Top:** Hard clustering, cluster 2 represents predicted maintenance events. **Bottom:** Soft clustering in terms of probability to maintenance.

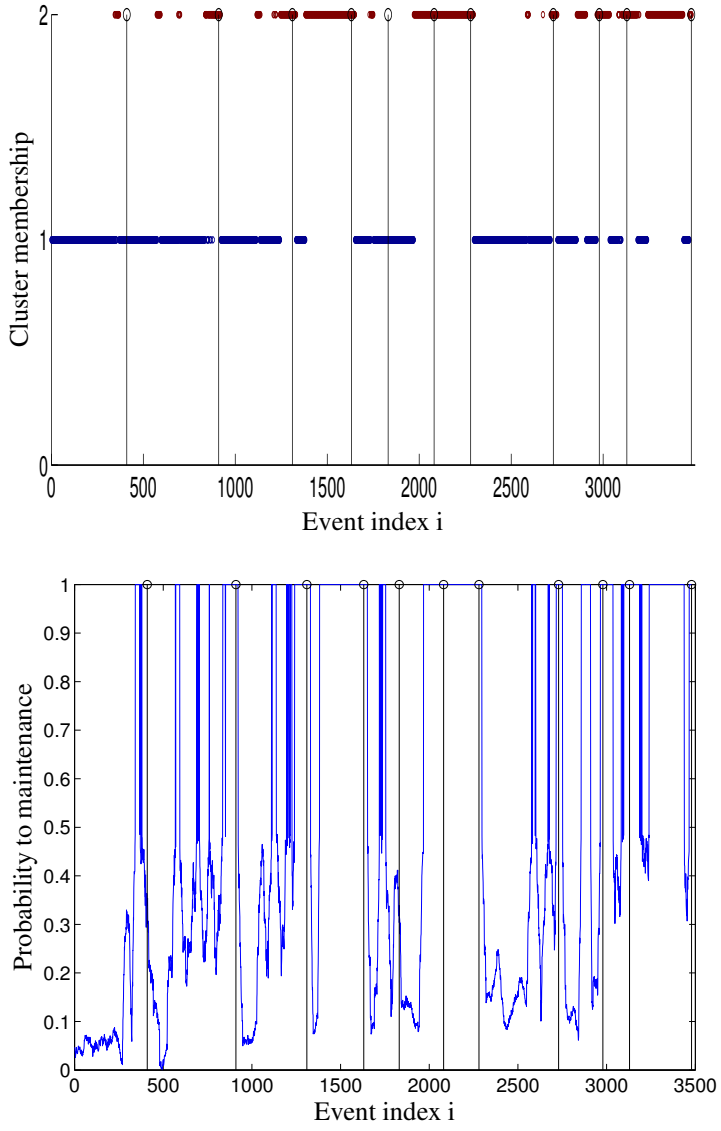


Figure 5.7: **KSC results dataset DS_III. Top:** Hard clustering, cluster 2 represents predicted maintenance events. **Bottom:** Soft clustering in terms of probability to maintenance.

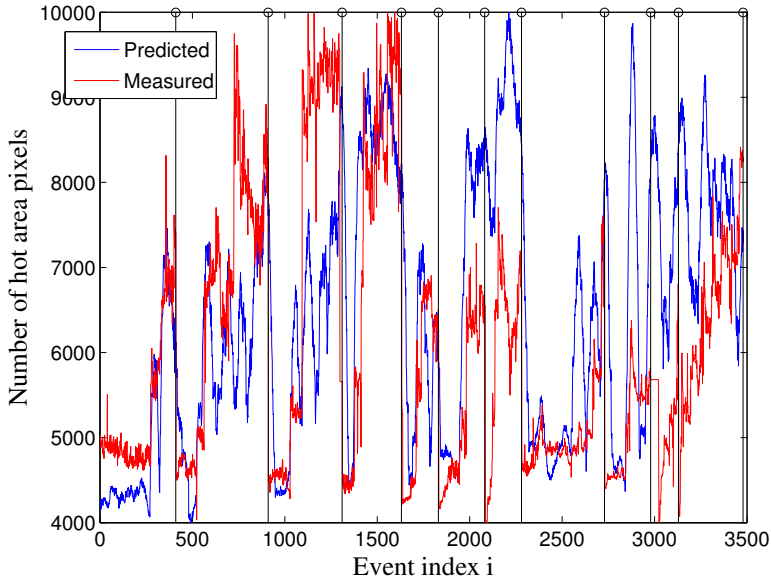


Figure 5.8: **Degradation dataset DS_III**. Degradation inferred by KSC (blue) *Versus* measured degradation (red). We can notice how, although the short-term variability is not detected, KSC can follow the general trend of the measured degradation. This is quite surprising since the clustering model, due to the fact that is unsupervised, did not have any information about the true degradation.

- the number of clusters should be provided by the user (this is done heuristically using a trial and error approach)
- it can discover only spherical boundaries.

Despite these disadvantages, it is still widely used since it works effectively in many scientific and industrial applications. Since it is the most popular method used in the industrial sector, here we present the results of K-means applied on the three datasets under analysis. Before discussing the results, it is worth to mention that, thanks to the model selection scheme of KSC described in the previous section, we give optimal parameters to k-means (number of clusters = 2 and window size of concatenated accelerometer signals = 40). Figure 5.9 visualizes the outcomes of the hard and soft clustering. Similarly to KSC, the soft clustering results are based on the distance between the final centroids and the datapoints in the input space (see [14]). Concerning datasets DS_I and DS_III the results of KSC and K-means are very similar, while in the dataset DS_II analysis K-means performs worse than KSC,

indicating the need of maintenance where it was not really performed. In this case K-means would suggest too many maintenance actions, which is not cost-effective. Finally, in table 5.1 the performance of KSC and K-means are evaluated according to a standard internal cluster quality measures, that is the mean silhouette value (MSV), (see appendix).

5.5 Conclusions

Predictive maintenance of industrial plants is receiving increasing attention in the last years due to advantages like cost efficiency and automation. In this chapter we discussed the usage of the KSC method for maintenance strategy optimization based on real-time condition monitoring of an industrial machine. We used the data collected by accelerometers placed on the jaws of a Vertical Form Fill and Seal (VFFS) machine. After applying a windowing operation on the data in order to catch the deterioration process affecting the sealing jaws, we showed how KSC is able to recognize the presence of at least two working regimes of the VFFS machine, identifiable respectively as normal and critical operating condition. Moreover, we have also proposed a soft clustering output that can be interpreted as "probability" to maintenance. In this way KSC could help to optimize the timing of maintenance actions for the machine under study in order to maximize the production capacity.

Algorithm	**DS_I**	**DS_II**	**DS_III**
KSC	**0.29**	**0.25**	0.27
K**-means**	0.28	0.12	0.27

Table 5.1: **Cluster quality evaluation.** Mean Silhouette value (MSV), with best performance in bold. In the case of dataset DS_II the low value of MSV indicates that K-means does not succeed in correctly separating the normal behaviour and the maintenance cluster, as KSC on the other hand does.

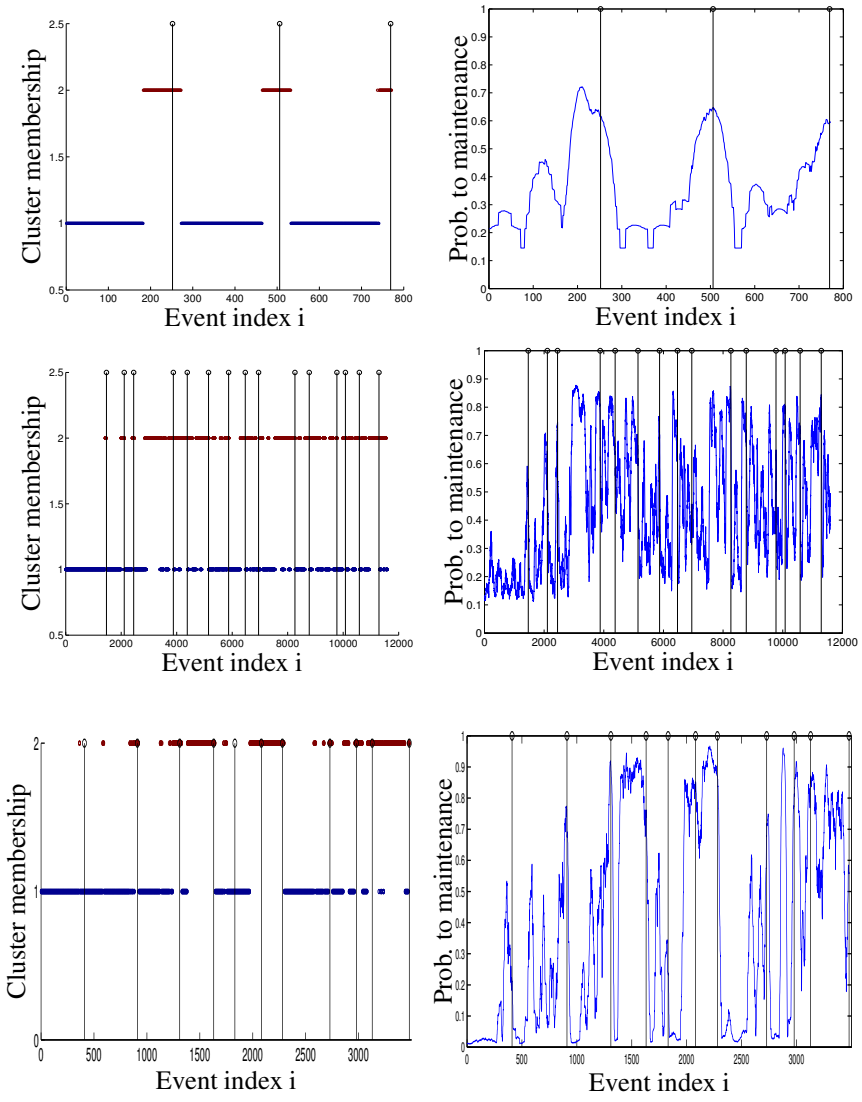


Figure 5.9: K**-means results.** Cluster 1 symbolizes the normal operating condition, cluster 2 represents predicted maintenance events. The vertical black lines show the true maintenance. **Top**: dataset DS_I. The results are similar to KSC outcomes, even if slightly worse since in the end there is a kind of false alarm (a single prediction of maintenance followed by normal behaviour, before the final maintenance cluster). **Center**: dataset DS_II. In this case K-means performs much worse than KSC, suggesting maintenance in regions not corresponding to actual maintenance events. **Bottom**: dataset DS_III. The results of KSC and K-means are very similar. Even if both techniques predict maintenance when it has not been really performed, it is quite likely that in this case the operator should have performed the maintenance actions in a different way he did.

Chapter 6

Clustering Non-Stationary Data

In this chapter we face the problem of clustering data in a non-stationary environment. We already experimented in this context in chapter 4, where the MKSC algorithm was proposed. However, in that case the focus was more on evolving networks and the model was based on a temporal smoothness assumption. Here an adaptive clustering model based on a different principle is proposed. The new method, called Incremental Kernel Spectral Clustering (IKSC), takes advantage of the out-of-sample property of kernel spectral clustering (KSC) to adjust the initial model over time. Thus, in contrast with other existing incremental spectral clustering techniques, we propose a model-based eigen-update, which guarantees high accuracy. On some toy data we will show the effectiveness of IKSC in modelling the cluster evolution over time via drifting, merging, splitting of clusters and so on. Then we analyse a real-world data-set consisting of PM_{10} concentrations registered during a heavy pollution episode that took place in Northern Europe in the end of January 2010. We will see how also in this case IKSC is able to recognize some interesting patterns and track their evolution across time, in spite of dealing with the complex dynamics of PM_{10} concentration.

6.1 General Overview

In many real-life applications, from industrial processes [35] to the analysis of the blogosphere [110] we face the ambitious challenge of dealing with non-stationary data. Therefore researchers perceived the need of developing clustering methods that can

model the complex dynamics of evolving patterns in a real-time fashion. This means that a clustering of the data during time-step t must be provided before seeing any data for time-step $t+1$, with the data-distribution changing across time. Some adaptive clustering models with different inspiration are present in the literature: evolutionary spectral clustering techniques discussed in the previous chapter [25, 27, 80, 82], self-organizing time map (SOTM), where an adaptation of the standard SOM is used to discover the occurrence and explore the properties of temporal structural changes in data [121], dynamic clustering via multiple kernel learning [114], incremental K-means [26], where the algorithm at time t is initialized with the centroids found at time $t-1$, data stream clustering algorithms aiming at analysing massive datasets by using limited memory and a single scanning of the data [50, 1], incremental clustering algorithms, whose main objective is to apply dynamic updates to the cluster prototypes when new data points arrive [23, 52, 109], and many others. Here we focus our attention on the family of the Spectral Clustering (SC) approaches briefly discussed in chapter 2 [108, 136, 29], which has shown its practical success in many application domains. SC is an off-line algorithm, and the above-cited attempts to make it applicable to dynamic data-sets, although quite appealing, are at the moment not very computationally efficient. In [110] and more recently in [37], the authors propose some incremental eigenvalue solutions to continuously update the initial eigenvectors found by SC. In the present study, we follow this direction, but with an important difference. The incremental eigen-update we introduce is based on the out-of-sample extension property of Kernel Spectral Clustering (KSC), introduced in section 2.2.5. The out-of-sample extension alone, without the need of ad-hoc eigen-approximation techniques like the ones proposed in [110] and [37], can be used to accurately cluster stationary data-streams. However, if the data are generated according to some distribution which changes over time (i.e. non-stationary), the initial KSC model must be updated. In order to solve this issue we introduce the Incremental Kernel Spectral Clustering algorithm (IKSC). The IKSC method exploits the research carried out in [7] to continuously adjust the initial KSC model over time, in order to learn the complex dynamics characterizing the non-stationary data.

6.2 Incremental Kernel Spectral Clustering

In contrast with other techniques that compute approximate eigenvectors of large matrices like the Nyström method [138], the work presented in [56] or the above-mentioned algorithms [37] and [110], the eigen-approximation we use in the IKSC method is model-based [7]. This means that based on a training set (in our case the cluster centroids) out-of-sample eigenvectors are calculated using eq. (6.1). These approximate eigenvectors are then used to adapt the initial clustering model over time. In principle, if the training model has been properly constructed, this guarantees high

accuracy of the approximated eigenvectors due to the good generalization ability of KSC and LS-SVM in general [6, 132].

6.2.1 Algorithm

One big advantage of a model-based clustering tool like KSC is that we can use it online in a straightforward way. Indeed, once we built-up our optimal model during the training phase, we can estimate the cluster membership for every new test point by simply applying eq. (2.16) and the ECOC decoding procedure. However, if the data source is non-stationary, this scheme fails since the initial model is not representative any more of the new data distribution. Therefore to cope with non-stationary data the starting code-book must be adjusted accordingly. Here, instead of using the code-book and the ECOC procedure, we propose to express our model in terms of the centroids in the eigenspace and to compute the cluster memberships as measured by the euclidean distance from these centers. In this way it is possible to continuously update the model in response to the new data-stream. In order to calculate the projection in the eigenspace for every new point, we can exploit the second KKT condition for optimality related to the KSC optimization problem which links the eigenvectors and the score variables for training data:

$$\alpha_{\text{test}}^{(l)} = \frac{1}{\lambda_l} D_{\text{test}}^{-1} e_{\text{test}}^{(l)} \tag{6.1}$$

with $D_{\text{test}}^{-1} = \text{diag}(1/\text{deg}(x_1^{\text{test}}), \ldots, 1/\text{deg}(x_{\text{Ntest}}^{\text{test}})) \in \mathbb{R}^{\text{Ntest}} \times \mathbb{R}^{\text{Ntest}}$ indicating the inverse degree matrix for test data. The out-of-sample eigenvectors $\alpha_{\text{test}}^{(l)}$ represent the model-based eigen-approximation with the same properties as the original eigenvectors $\alpha^{(l)}$ for training data. With the term eigen-approximation we mean that these eigenvectors are not the solution of an eigenvalue problem, but they are estimated by means of a model built during the training phase of KSC [7]. To summarize, once one or more new points belonging to a data-stream are collected, we update the IKSC model as follows:

- calculate the out-of-sample extension using eq.(2.16), where the training points x_i are the centroids in the input space $C_1, \ldots, C_k$, and the $\alpha^{(l)}$ are the centroids in the eigenspace $C_1^{\alpha}, \ldots, C_k^{\alpha}$
- calculate the out-of-sample eigenvectors by means of eq (6.1)
- assign the new points to the closest centroids in the eigenspace
- update the centroids in the eigenspace
- update the centroids in the input space

Algorithm 11: IKSC algorithm [75]

Data: Training set $\mathcal{D}_{\text{Tr}} = \{x_i\}_{i=1}^{N_{\text{Tr}}}$, kernel function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ positive definite and localized ($K(x_i, x_j) \to 0$ if x_i and x_j belong to different clusters), kernel parameters (if any), number of clusters k.

Result: Clusters $\{\mathcal{A}_1, \ldots, \mathcal{A}_p\}$, cluster centroids in the input space $C_1, \ldots, C_k$, cluster centroids in the eigenspace $C_1^\alpha, \ldots, C_k^\alpha$.

Acquire N_{Tr} points.
Train the KSC model by solving eq. (2.13).
Obtain the initial centroids in the input space $C_1, \ldots, C_k$ and the initial centroids in the eigenspace $C_1^\alpha, \ldots, C_k^\alpha$.
for $i \leftarrow N_{Tr} + 1$ **to** N **do**
    compute the out-of-sample eigenvectors using eq. (6.1)
    calculate cluster membership for the new point (or the new batch of points) according to the distance between the out-of-sample eigenvectors and the centroids $C_1^\alpha, \ldots, C_k^\alpha$
    update centroids in the eigenspace $C_1^\alpha, \ldots, C_k^\alpha$ using eq. (6.2)
    update centroids in the input space $C_1, \ldots, C_k$ according to eq. (6.2)
    new cluster check
    merge check
    cluster death
end
Outlier elimination.

To update online a centroid C_{old} given a new sample x_{new}, we can use the following formula [64]:

$$C_{\text{new}} = C_{\text{old}} + \frac{x_{\text{new}} - C_{\text{old}}}{n_{\text{old}}} \tag{6.2}$$

where n_{old} is the number of samples previously assigned to the cluster center C_{old}. The same procedure can be used to update the cluster centers in the eigenspace: in this way the initial $\alpha^{(l)}$ provided by KSC are changed over time to model the non-stationary behaviour of the system. A schematic visualization of this procedure is depicted in figure 6.1. Finally, the complete IKSC technique is sketched in algorithm 11. The adaptation to non-stationarities relates to identifying changes in the number of clusters occurring over time by means of some inspections:

- the new cluster check allows to dynamically create a new cluster if necessary. For every new point the related degree d_i^{test} is calculated. If $d_i^{\text{test}} < \epsilon$ where ϵ is a user-defined threshold, it means that the point is dissimilar to the actual centroids. Therefore it becomes the centroid of a new cluster and it is added to the model. The threshold ϵ is data-dependent, and can be chosen before processing the data stream based on the degree distribution of the test kernel matrix, when considering as training set the cluster prototypes in the input space.

- throughout the merge check, if two centroids become too similar they are merged into one center, and the number of clusters is decreased. In particular, the similarity between two centroids is computed as the cosine similarity in the eigenspace, and two centroids are merged if this similarity is greater than 0.5.
- if the centroid of a cluster is not updated any more the algorithm considers that cluster as disappeared (cluster death).

Finally, if one cluster has size smaller than a user-defined threshold it is considered as outlier and it is eliminated in the end of the data-stream acquisition.

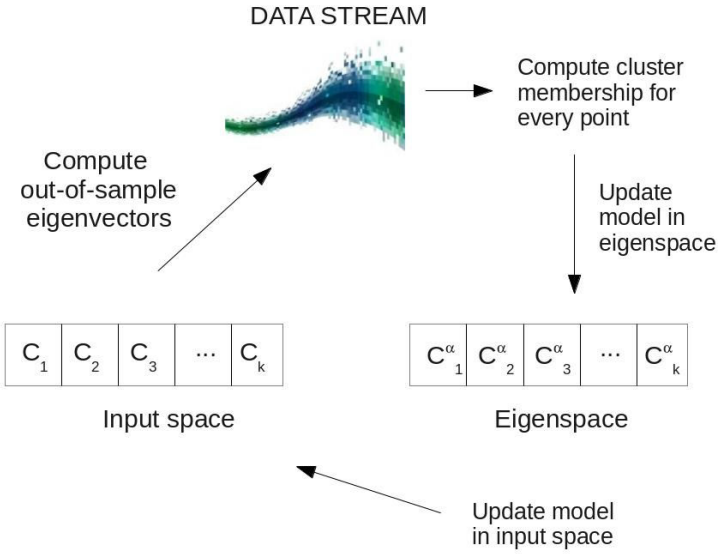


Figure 6.1: **IKSC update scheme** After the initialization phase, when a new point arrives, both the training set and the model (i.e. the cluster centers in the eigenspace) are adapted accordingly.

6.2.2 Computational Complexity

Let us assume that at the beginning of the data-stream acquisition the training set $\mathcal{D} = \{x_i\}_{i=1}^{k}$, $x_i \in \mathbb{R}^d$ is formed by k centroids in the input space $C_1, \ldots, C_k$ and the initial clustering model is expressed by the centroids in the eigenspace $C_1^\alpha, \ldots, C_k^\alpha$, with $C_i^\alpha \in \mathbb{R}^{k-1}$. For every new point of the data-stream, as explained in the previous

section, we have to compute the out-of-sample extension, the corresponding out-of-sample eigenvectors by means of eq. (6.1) and the update of both the model and the training set, that are the clusters centers in the input space. Since the training set can be kept small enough[1], the main contribution to the computational complexity is due to the out-of-sample extension part:

$$e_{\text{test}}^{(l)} = \Omega_{\text{test}}\alpha^{(l)} + b_l 1_{\text{Ntest}}, l = 1, \ldots, k-1. \tag{6.3}$$

The evaluation of the kernel matrix Ω_{test} needs $O(k^2 d)$ operations to be performed. The calculation of the score variables $e_{\text{test}}^{(l)}$ takes then $O(k^2 d + k^2 + k)$ time. This operation has to be repeated for the N_{test} data-points of the data-stream, so the overall time complexity is $O(N_{\text{test}}(k^2 d + k^2 + k))$. This can become linear with respect to the number of data-points ($O(N_{\text{test}})$) when $k \ll N_{\text{test}}$ and $d \ll N_{\text{test}}$, which is the case in many applications. This is comparable with other eigen-updating algorithms for spectral clustering like [110] and [37].

6.3 Synthetic Experiments

6.3.1 Description of the data

Three simulations are performed: the first and the second by reproducing the experiments described in [21], and the third with some computer-generated time-series. In the first simulation two Gaussian distributions evolving over time are created. These two clouds of points drift toward each other with increasing dispersal, as illustrated in figure 6.2. In the second virtual experiment a multi-cluster non-stationary environment is created. In particular, there are two drifting Gaussian clouds that come to merge, some isolated data forming an outlier cluster of 4 points and a static cluster consisting of a bi-modal distribution. This second data-set is depicted in figure 6.3. In order to test the ability of IKSC to dynamically cluster time-series rather than data-points, we generated 20 time-series of three types as depicted in figure 6.4. The idea behind this experiment is that if we cluster in an online fashion the time-series with a moving window approach, we should be able to detect the appearance of a new cluster given the increase in frequency of the signals of the second type at time step $t_1 = 150$. Moreover, when these signals get back to their original frequency at time step $t_2 = 300$, the clustering algorithm must detect this change.

In the first two synthetic experiments (Drifting Gaussians and Merging Gaussians) we use the RBF kernel function defined by $K(x_i, x_j) = \exp(-||x_i - x_j||_2^2/\sigma^2)$.

[1] In this work we assume that the training set consists of k points, where k is the number of clusters. In some situations it could happen that such a small number of training points is not enough to define a proper mapping. Nevertheless, by considering more training points N such that $N \ll N_{\text{test}}$ the overall complexity of the algorithm does not change.

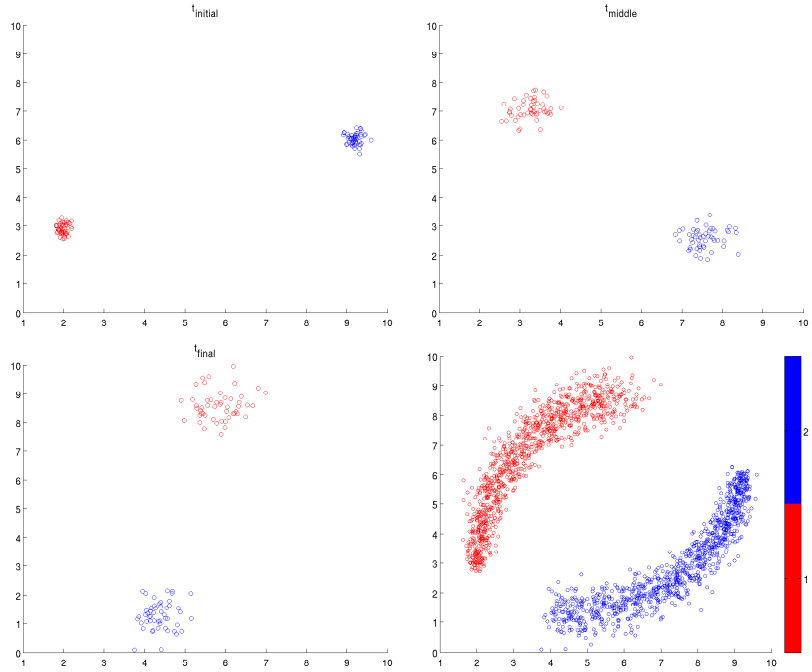


Figure 6.2: **Drifting Gaussian distributions dataset.** Some snapshots of the evolution of the distributions (top and bottom left), and the whole data all at once (bottom right). The axes represent the two dimensions of the N input vectors $x_i \in \mathbb{R}^2, i = 1, \ldots, N$.

The symbol σ indicates the bandwidth parameter and x_i is the i-th data point. In the analysis of the third synthetic data (Synthetic time-series) x_i represents the i-th time-series. In this case to better capture the similarity between the time-series we use the RBF kernel with the correlation distance [86]. Thus $K(x_i, x_j) = \exp(-||x_i - x_j||^2_{\text{cd}}/\sigma^2)$, where $||x_i - x_j||_{\text{cd}} = \sqrt{\frac{1}{2}(1 - R_{ij})}$, with R_{ij} indicating the Pearson correlation coefficient between time-series x_i and x_j. By means of extensive experiments we empirically observed that this kernel is positive definite. Moreover the RBF kernel with Euclidean distance has been mathematically proven to fulfil the positive definitiveness property.

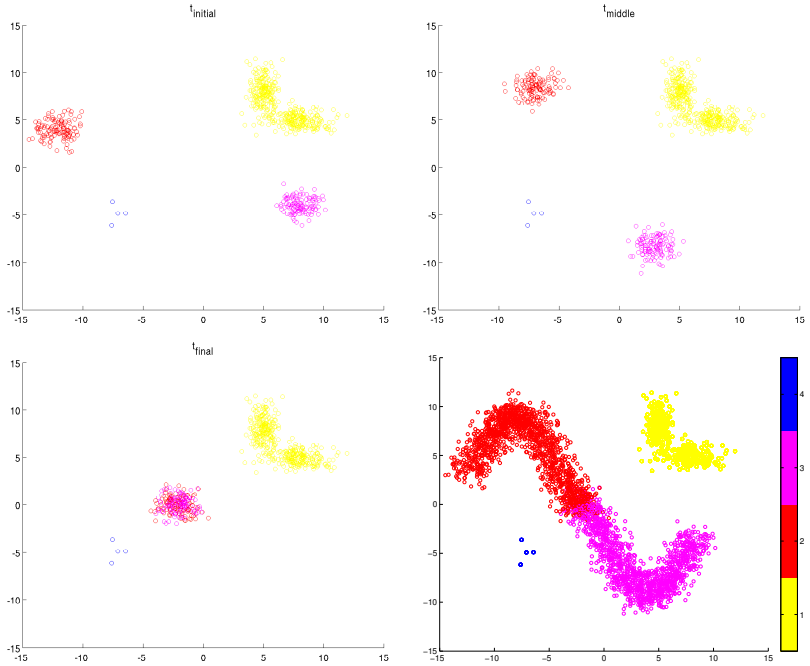


Figure 6.3: **Merging Gaussian distributions dataset.** Some snapshots of the evolution of the distributions (top and bottom left), and the whole data all at once (bottom right). The axes represent the two dimensions of the N input vectors $x_i \in \mathbb{R}^2, i = 1, \ldots, N$.

6.3.2 Simulation results

Here we show how the proposed IKSC model, thanks to its capacity of adapting to a changing environment, is able to model the complex behaviour of evolving patterns of non-stationary data. To evaluate the outcomes of the model, two cluster quality measures are computed [53]: the average cumulative adjusted rand index (ARI) error defined as $1 -$ ARI, like in [37] and the instantaneous mean silhouette value (MSV).

The results of testing the IKSC algorithm on the first synthetic example is presented in figure 6.5. In the initialization phase 30 points are used to construct the model. The IKSC algorithm can perfectly model the two drifting distributions: the average cumulative ARI error is equal to 0. Moreover the quality of the predicted clusters remains very high over time, as demonstrated by the trend of the instantaneous MSV depicted in figure 6.6. The results of the simulation related to the second artificial dataset are depicted in figure 6.7. Similarly to the first artificial experiment, the cluster

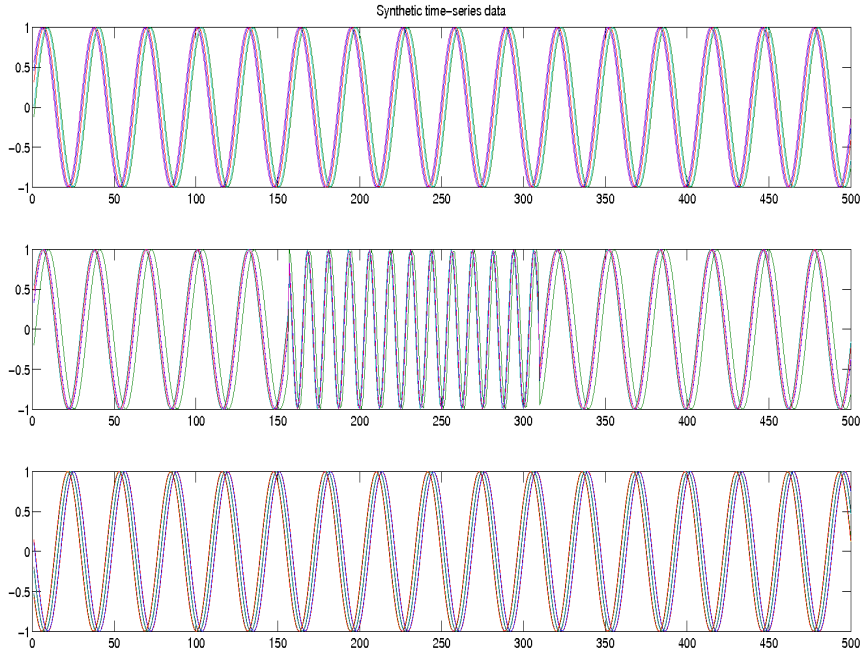


Figure 6.4: **Synthetic time-series.** The X axis indicates the time-step and the Y axis the amplitude of the signals. At $t_1 = 150$ and $t_2 = 300$ two change points (change in frequency) can be observed.

quality stays high over time as shown in figure 6.8, and the partitions found by IKSC are in almost perfect agreement with the ground truth (small ARI error) for the whole duration of the simulation (see figure 6.9). Moreover at time-step $t = 6926$ the two moving Gaussian clouds are merged, as expected. Only in this case, as observed also in [21], there is a small increase in the average cumulative ARI error. The small cluster at the bottom left side of figure 6.3 is detected as outlier after the data acquisition. Finally, we discuss the results of IKSC on the synthetic time-series experiment. In the initialization phase the algorithm recognizes 2 clusters, which are shown in figure 6.10. After some time, we notice that IKSC successfully detects the first change in frequency of the signals of the second type by creating a new cluster at time step $t = 223$, as depicted in figure 6.11. Moreover the second change point is detected at time step $t = 382$, when a merging of two clusters is performed, as illustrated in figure 6.12.

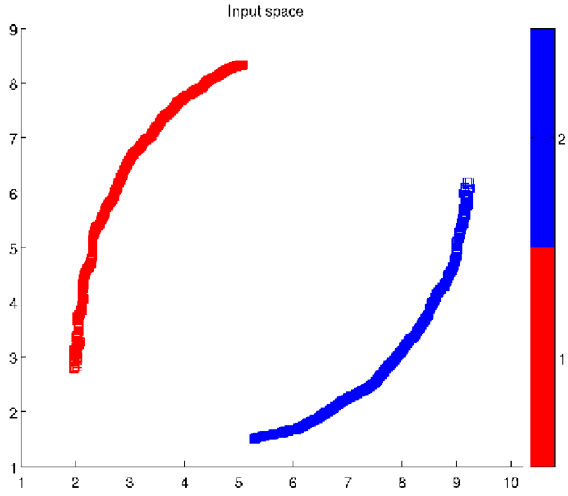


Figure 6.5: **Results of IKSC on the drifting Gaussian distributions.** Evolution of the centroids in the 2D input space. We can notice that the IKSC model can recognize the drifting targets without errors.

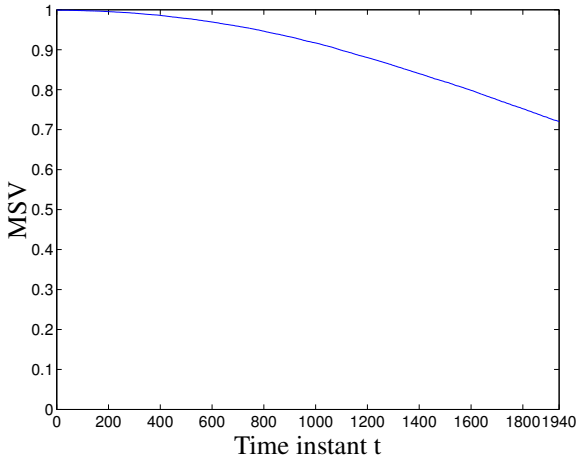


Figure 6.6: **MSV for drifting Gaussian distributions.** The mean silhouette value related to the clusters detected by IKSC stays high over time, meaning that our method is able to model the drift of the distributions.

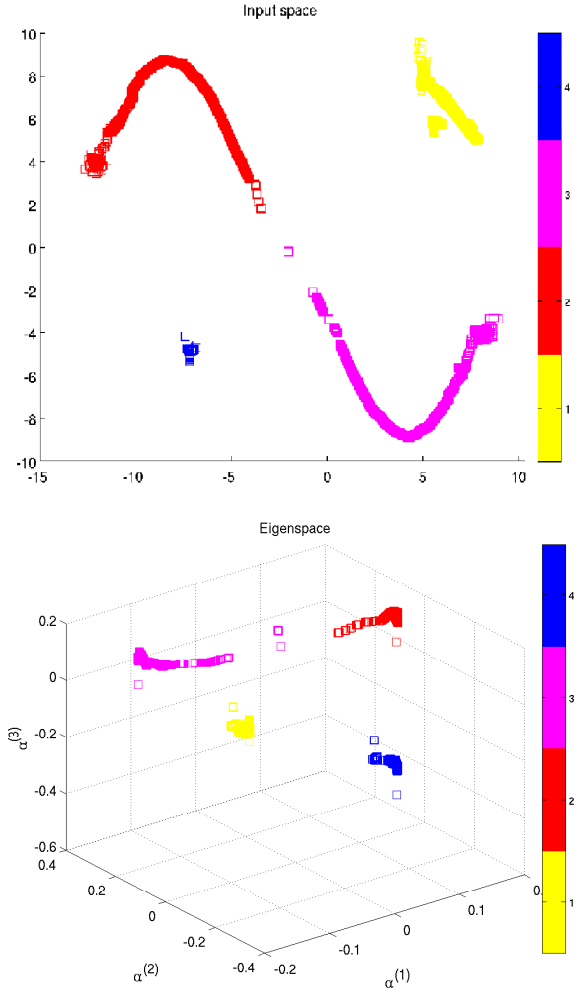


Figure 6.7: **Results of IKSC on the merging Gaussian distributions. Top:** Evolution of the centroids in the 2D input space. **Bottom:** Model evolution in the eigenspace.

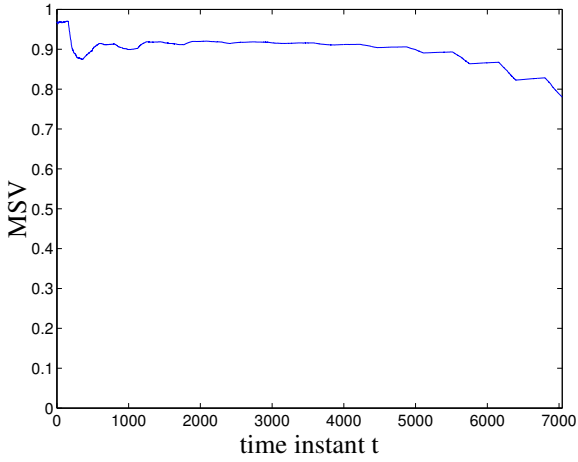


Figure 6.8: **MSV for the merging Gaussian distributions experiment.** The mean silhouette value related to the clusters detected by IKSC remains high over time. Thus, also in this case IKSC manages to properly follow the non-stationary behaviour of the clusters for the whole duration of the experiment.

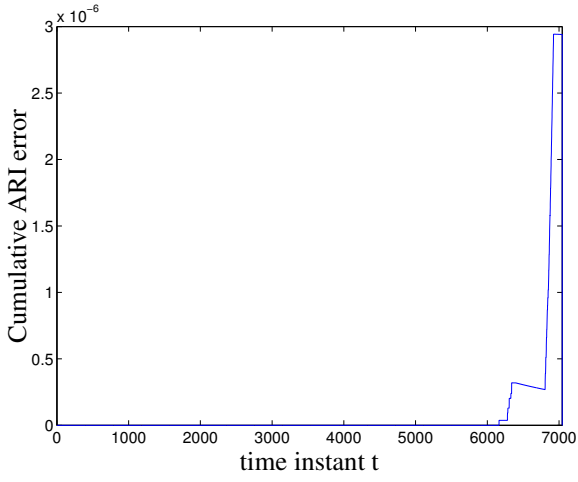


Figure 6.9: **ARI error for Merging Gaussian distributions.** The average cumulative ARI error related to the clusters detected by IKSC is very small over time, with a peak around the merging step at time $t = 6926$, in agreement with what was observed also in [21].

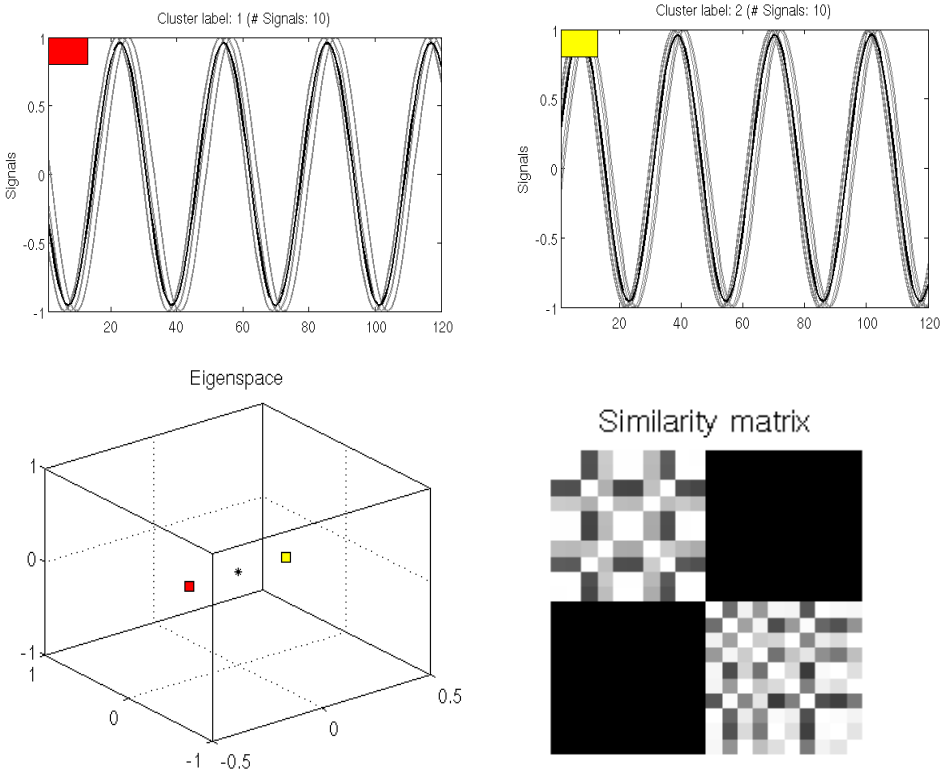


Figure 6.10: **Synthetic time-series initial clustering model. Top:** signals of the two starting clusters. **Bottom left**: data in the eigenspace spanned by $\alpha^{(1)}$ (the points are mapped in the same location as the related centroids, since the eigenvectors are perfectly piece-wise constant). **Bottom right**: kernel matrix with a clear block diagonal structure.

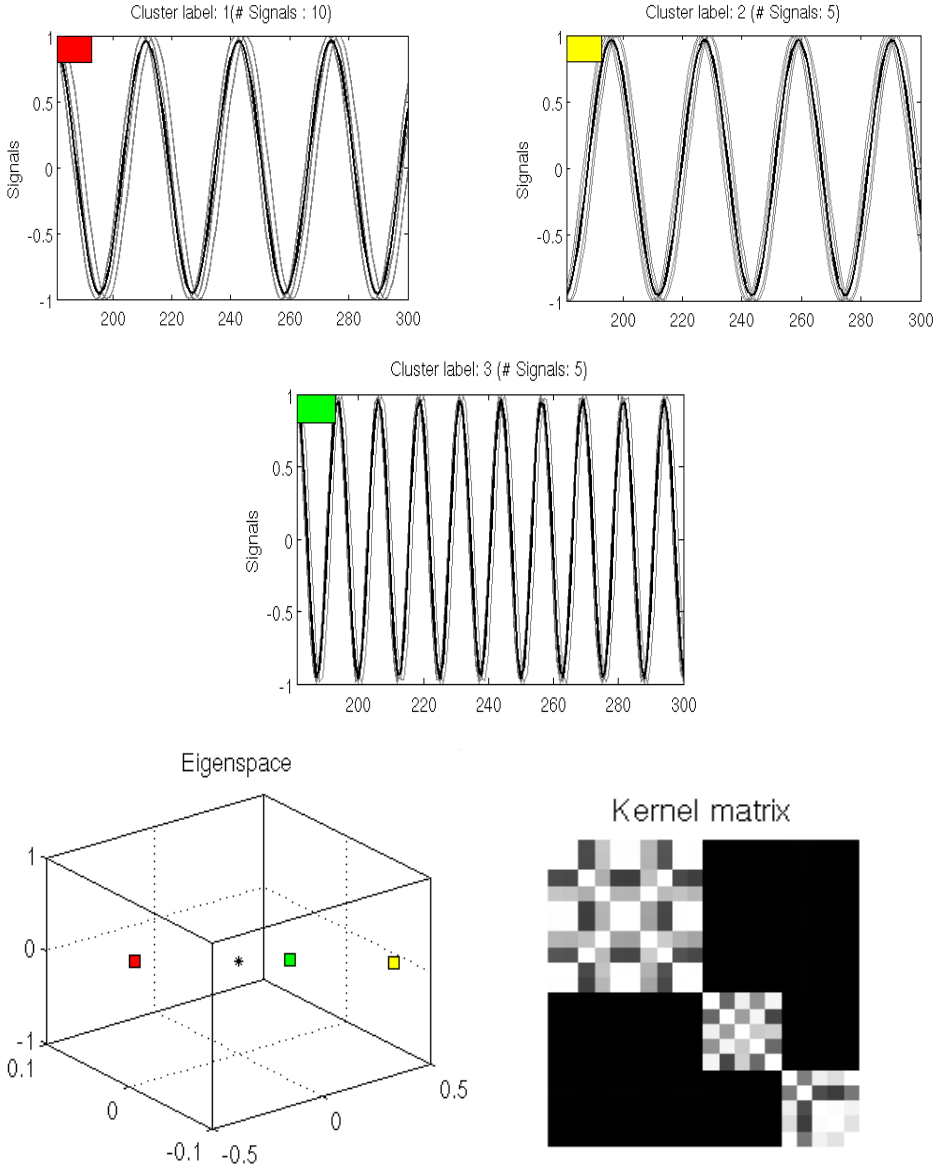


Figure 6.11: **Synthetic time-series clusters after creation. Top and center:** signals of the three clusters after the creation event. **Bottom left** data in the eigenspace spanned by $\alpha^{(1)}$ and $\alpha^{(2)}$ (the points are mapped in the same location as the related centroids, since the eigenvectors are perfectly piece-wise constant). **Bottom right:** kernel matrix.

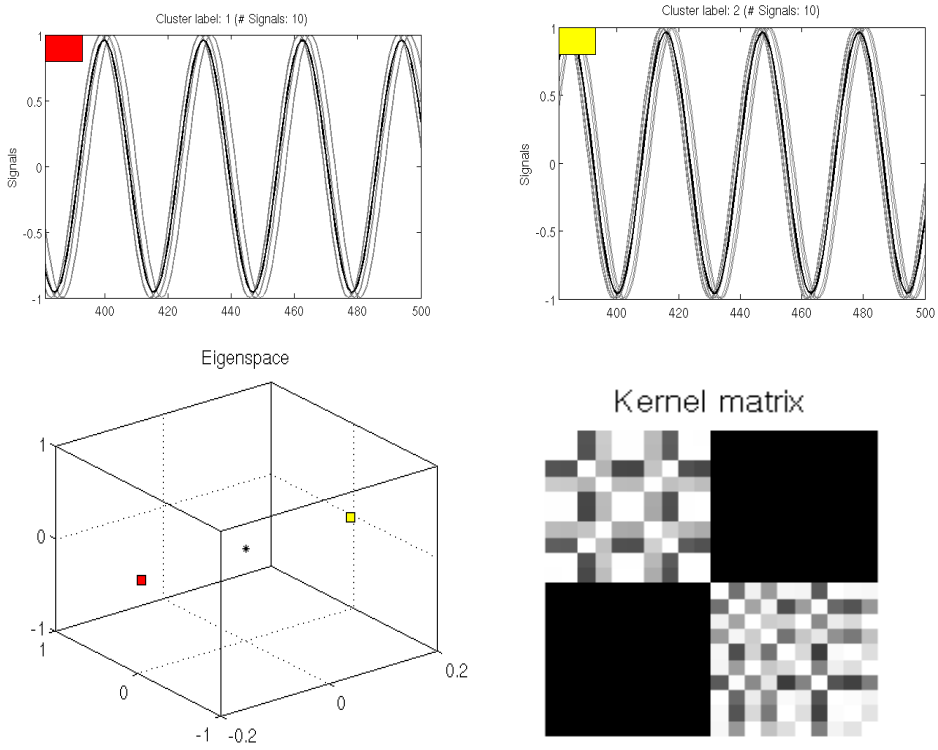


Figure 6.12: **Synthetic time-series final clustering model. Top:** two final clusters after the merging event. **Bottom left:** clustered data in the eigenspace spanned by $\alpha^{(1)}$ (the points are mapped in the same location as the related centroids, since the eigenvectors are perfectly piece-wise constant). **Bottom right:** kernel matrix.

6.3.3 Analysis of the eigenvectors

Here we discuss on the quality of our model-based eigen-updating for kernel spectral clustering. In figure 6.13 the exact and the approximated eigenvector related to the largest eigenvalue of (2.13) for the drifting Gaussians example are shown. We notice that the model-based eigenvectors are less noisy with respect to the exact eigenvectors and a multiplicative bias is present. The first property is quite surprising: basically we are able to recover the perfect separation between the two clusters even when this is somehow masked by the data. This occurs mainly in the end of the simulation when the two Gaussian clouds approach each other. In this case the exact eigenvector is not exactly piece-wise constant due to a small overlap, while the model-based eigenvector is much less perturbed. The multiplicative bias is probably due to the fact that the

out-of-sample eigenvectors are computed using an ultra-sparse training set (only the two cluster centroids). The latter allows to process the data-stream very quickly, but lacks of the information related to the spread of the data-points, which may cause the bias. Similar considerations can be done for the second synthetic experiment, i.e. the merging Gaussians. The three eigenvectors corresponding to the largest eigenvalues of (2.13) are represented in figure 6.14. In the third approximated eigenvector we can notice 4 levels, which are not present in the exact eigenvector. Once again this testifies the tight relation between the clustering model of IKSC (the 4 centroids) and the approximated eigenvectors, which is a unique property of our framework.

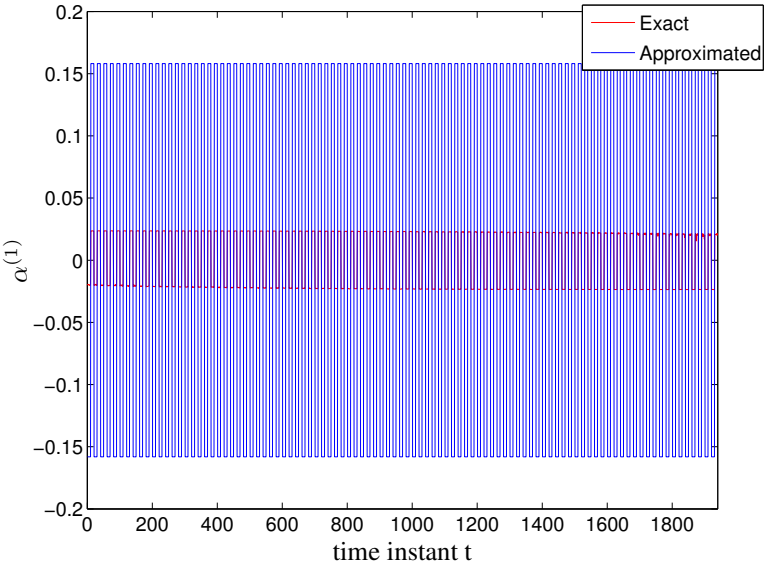


Figure 6.13: **Eigenvector-Drifting Gaussian distributions.** Exact and approximated eigenvector corresponding to the largest eigenvalue of the problem (2.13), for the first synthetic example.

6.4 Real-Life Example

6.4.1 The PM_{10} data-set

Particulate Matter (PM) is the term used for solid or liquid particles found in the air. In particular PM_{10} refers to those particles whose size is up to 10 micrometers

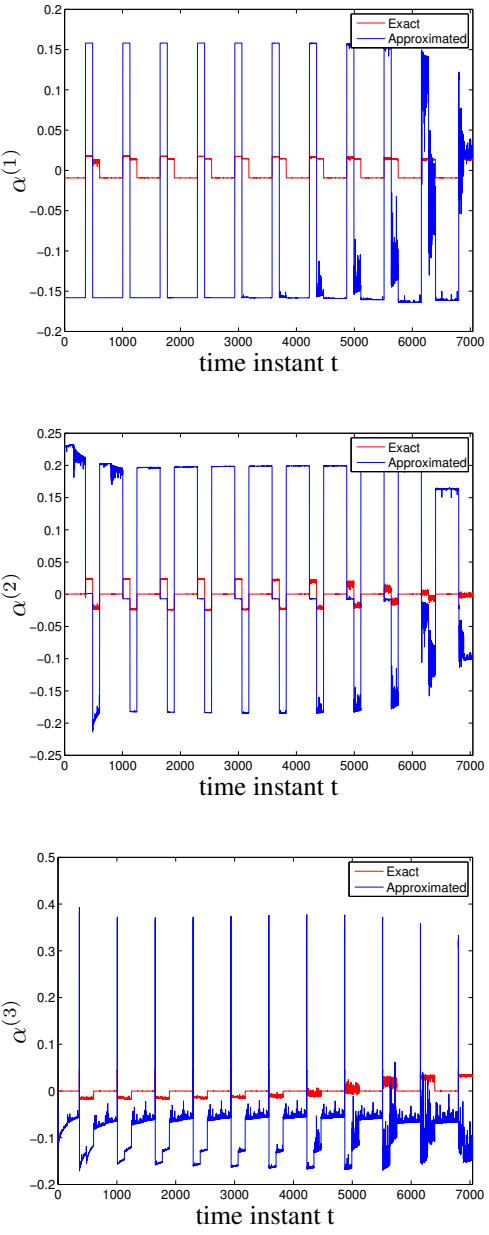


Figure 6.14: **Eigenvectors related to the Merging Gaussian distributions data.** Exact and approximated eigenvectors corresponding to the 3 largest eigenvalues of the problem (2.13), for the second synthetic experiment.

in aerodynamic diameter. The inhalation of these particles is dangerous for human health since it can cause asthma, lung cancer, cardiovascular issues, etc. Accurate measurements and estimation of PM is then of vital importance by the health care point of view. To this aim the European Environmental Agency manages a publicly available database called AirBase [40]. This air-quality database contains validated air quality monitoring information of several pollutants for more than 30 participating countries throughout Europe. In this paper we analyse the PM_{10} data registered by 259 background stations during a heavy pollution episode that took place between January 20th, 2010 and February 1st, 2010. We focus on an area comprising four countries: Belgium, Netherlands, Germany and Luxembourg (see figure 6.15). The experts attributed this episode to the import of PM originating in Eastern Europe [2].

6.4.2 Results of the simulations

In the initialization phase our data-set consists of a time-series of 96 time steps (i.e. four days) for each station. In order to build-up an initial clustering model we tune the number of clusters k and the proper σ for the RBF kernel by using the AMS (Average Membership Strength) model selection criterion explained in chapter 2. After tuning we find $k = 2$ and $\sigma^2 = 0.05$ as optimal parameters, as depicted in figure 6.16. The initial model, based on these parameters, is illustrated in figure 6.17. In this case the 2 centroids in the input space are the time-series representing the two clusters, while in the eigenspace they are points of dimension $k - 1$ (anyway for visualization purposes we always use a 3D plot).

During the online stage, by adopting a moving window approach, our data-set at time t corresponds to the PM_{10} concentrations measured from time $t - 96$ to time t. In this way we are able to track the evolution of the pollutants over time. In fact, after some time the IKSC model creates a new cluster, as depicted in figure 6.18. Later on these three clusters evolve until a merge of two of them occurs at time step $t = 251$ (see figure 6.19). If we analyse more in details the clustering results, we can notice how the new cluster (represented in blue) is concentrated mainly in the Northern region of Germany. Moreover the creation occurs at time step $t = 143$, when the window describes the start of the pollution episode in Germany. Afterwards, the new cluster starts expanding in direction South-West. Basically, IKSC is detecting the arrival of the pollution episode originated in Eastern Europe and driven by the wind toward the West. This ability of our clustering model of detecting the dynamics of the pollution cloud at this level of accuracy is rather unexpected. Indeed, IKSC does not have any information about the spatial localization of the stations and the meteorological conditions. At time step $t = 251$ two clusters are merged. This can be explained by the fact that the window covers the unusually high PM_{10} concentrations as well as the end of the episode, registered by many of the stations.

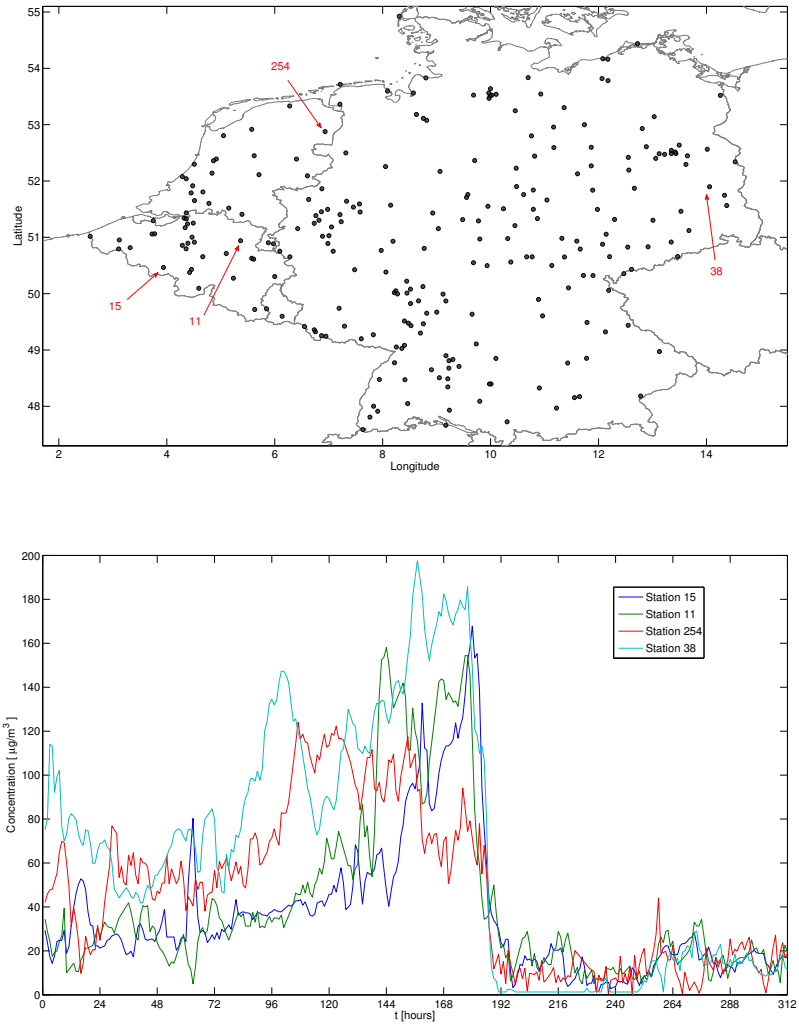


Figure 6.15: **PM_{10} data. Top:** AirBase monitoring stations. **Bottom:** Examples of PM_{10} concentrations time-series in Belgium, Netherlands, Luxembourg and Germany, for the whole period under investigation.

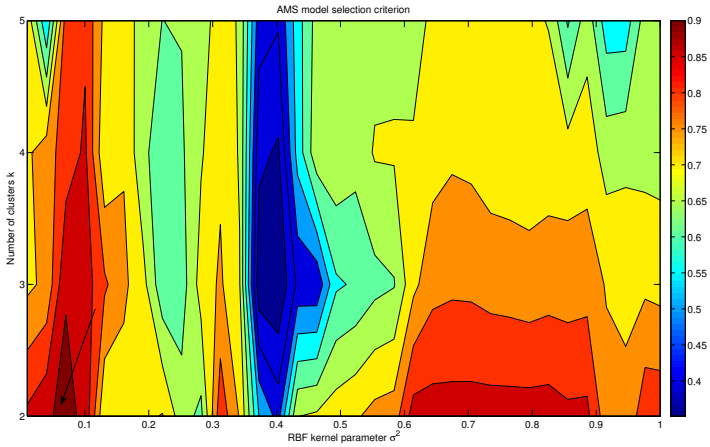


Figure 6.16: **Model selection.** Tuning of the number of clusters and the bandwidth of the RBF kernel in the initialization phase of IKSC for the analysis of the PM_{10} data.

6.5 Incremental K-means Clustering

As mentioned in the previous chapter, one of the most popular data clustering methods in many scientific domains is K-means clustering because of its simplicity and computational efficiency. In its incremental variant, the K-means clustering algorithm is applied online to a data stream. At each time-step Incremental K-means (IKM) uses the previous centroids to find the new cluster centers, instead of rerunning the K-means algorithm from scratch [26].

In table 6.1 a summary of the results regarding all the experiments is presented. The performance of IKSC and IKM are compared in terms of mean ARI and MSV over time. Concerning the experiments with the Gaussian clouds IKSC achieves better cluster accuracy (higher ARI), with a slightly worse MSV with respect to IKM. In the case of the synthetic time-series and the PM_{10} data IKSC outperforms IKM in terms of MSV.

6.6 Conclusions

In this chapter an adaptive clustering model called Incremental Kernel Spectral Clustering (IKSC) has been introduced. IKSC takes advantage of the out-of-sample

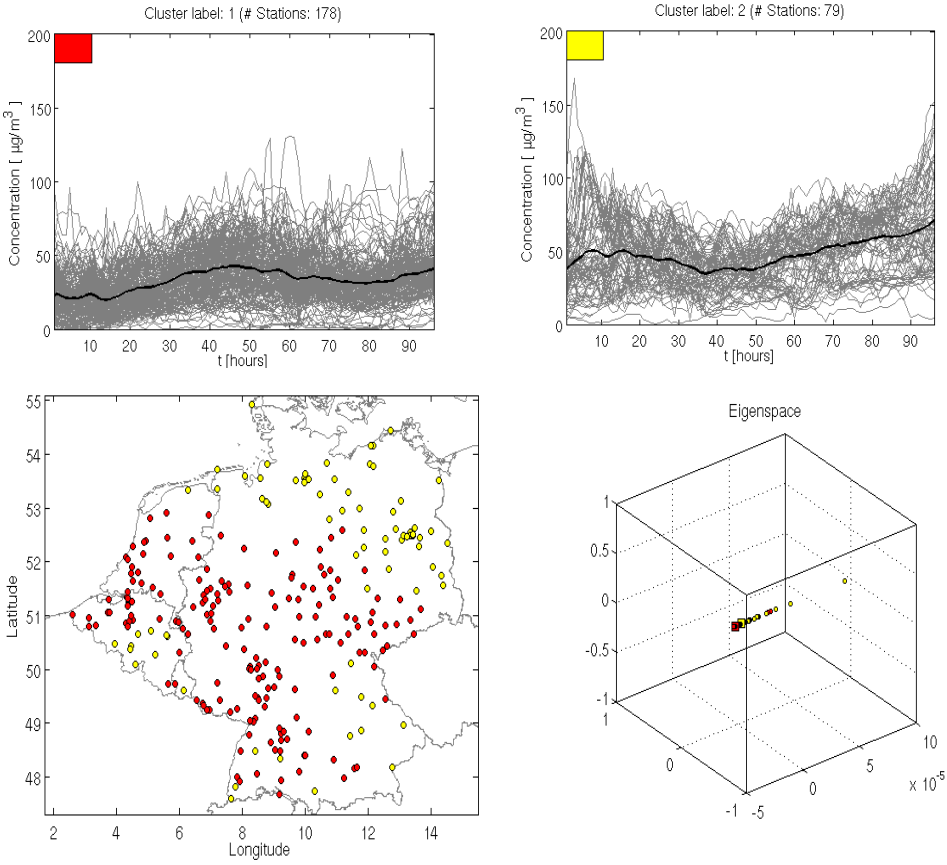


Figure 6.17: **Initial clustering model for the PM_{10} monitoring stations. Top:** signals for the two starting clusters. **Bottom left:** Spatial distribution of the clusters over Belgium, Netherlands, Luxembourg and Germany. **Bottom right:** data mapped in the eigenspace (in this case the space spanned by the eigenvector $\alpha^{(1)}$ related to the highest eigenvalue).

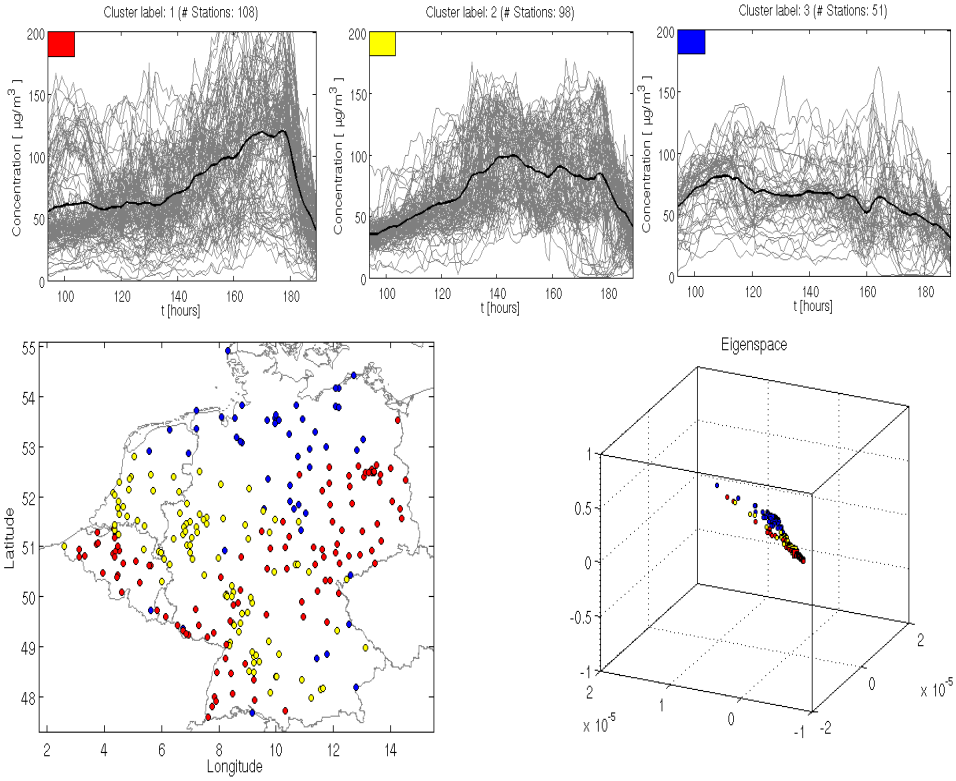


Figure 6.18: **PM_{10} clusters after creation. Top:** signals for the three clusters after the creation event. **Bottom left:** Spatial distribution of the clusters over Belgium, Netherlands, Luxembourg and Germany. Interestingly, the new cluster comprises stations located in the North-East part of Germany, which is the area where the pollutants coming from Eastern Europe started to spread during the heavy pollution episode of January 2010. **Bottom right:** data in the space spanned by the eigenvectors $\alpha^{(1)}$ and $\alpha^{(2)}$.

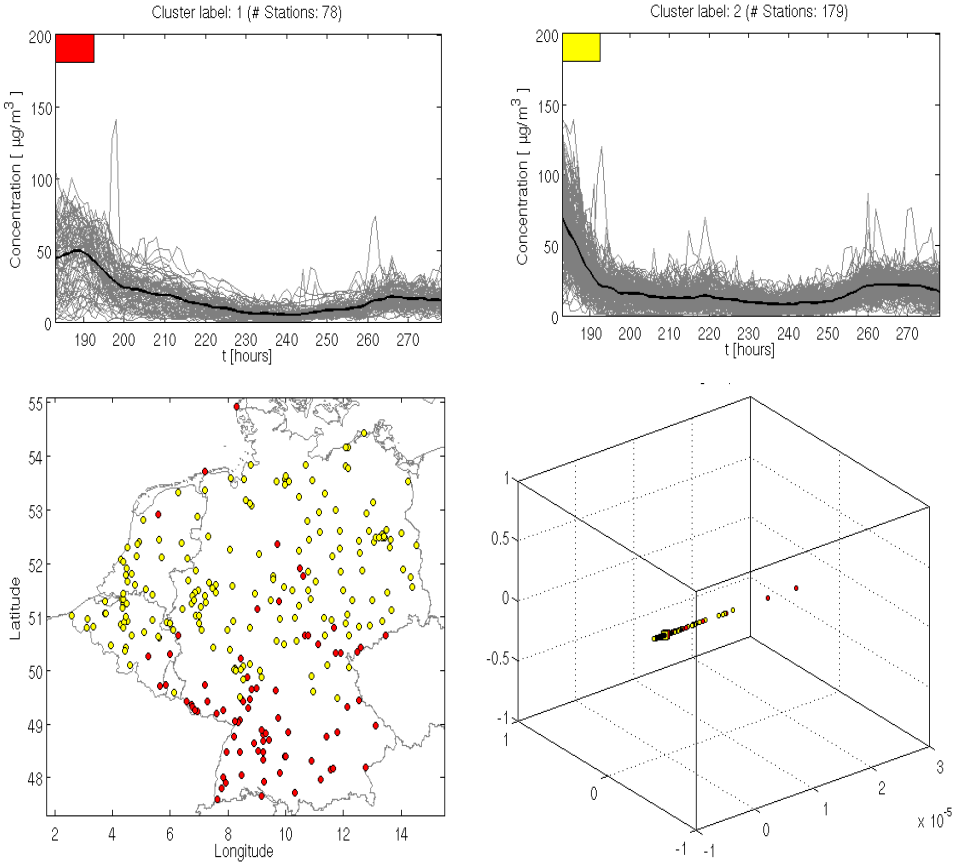


Figure 6.19: **Clustering model of PM_{10} stations after merging. Top:** two clusters left after the merging event occurred at time step $t = 251$. **Bottom left:** Spatial distribution of the clusters over Belgium, Netherlands, Luxembourg and Germany. **Bottom right:** data in the space spanned by the eigenvector $\alpha^{(1)}$.

property of KSC to adjust the initial model over time. Thus, in contrast with other existing incremental spectral clustering techniques, we proposed a model-based eigen-update, which guarantees high accuracy. On some toy-data we have shown the effectiveness of IKSC in modelling the cluster evolution over-time (drifting, merging, cluster appearance etc.) and in recognizing change points. Then we analysed a real-world data-set consisting of PM_{10} concentrations registered during a heavy pollution episode that took place in Northern Europe in January 2010. Also in this case IKSC was able to recognize some interesting patterns and track their evolution over-time, in spite of dealing with the complex dynamics of PM_{10} concentration.

Experiment	Algorithm	MSV	ARI
Drifting Gaussians	**IKM**	**0.89**	1
	IKSC	0.88	1
Merging Gaussians	**IKM**	**0.91**	0.95
	IKSC	0.90	**0.99**
Synthetic time-series	**IKM**	0.90	–
	IKSC	**0.92**	–
PM_{10} data	**IKM**	0.27	–
	IKSC	**0.32**	–

Table 6.1: **Cluster quality evaluation.** Average ARI and/or MSV over time for all the experiments that have been performed. In the case of the synthetic time-series and the PM_{10} only MSV is computed since the true partition is unknown.

Chapter 7

Conclusions and Future Challenges

7.1 General Conclusions

This thesis has addressed the issue of clustering in a changing scenario by means of a variety of techniques cast in the LS-SVM primal-dual optimization framework. In chapter 2 we introduced the basics of spectral clustering (SC), LS-SVM and kernel spectral clustering (KSC), and we discussed a new algorithm called soft kernel spectral clustering (SKSC). In this method, in contrast with KSC, a fuzzy assignment based on the cosine distance from the cluster prototypes in the projection space is suggested, together with a novel model selection technique called average membership strength criterion (AMS). The soft assignment and the AMS are found to produce more interpretable results and an improved clustering performance with respect to KSC, mainly in cases of large overlap between the clusters.

In chapter 3, a whole kernel-based framework for community detection in complex networks is discussed. This methodology is based on five cornerstones, corresponding to the extraction of a small training sub-graph representative of the community structure of the entire network, the model selection issue, the choice of an appropriate kernel function, the out-of-sample extension and the applicability to large-scale data. In particular, we have introduced a model selection criterion based on the Modularity quality function which is more suited for network data. Moreover, the sampling method to select the training set is based on the greedy maximization of the expansion factor. We showed how our approach produces high-quality partitioning both in terms of Modularity and Conductance, and it outperforms spectral clustering using the

Nyström method for out-of-sample extension. As mentioned, we also explained how our technique can be used for large-scale applications and in a distributed computing environment.

The experiments performed for dealing with the static clustering and community detection problems laid the foundations for the development of models for the analysis of evolving data. In chapter 4 a novel technique called kernel spectral clustering with memory effect (MKSC) is proposed in order to cluster evolving networks. An evolving network can be represented as a sequence of snapshots over time, where each snapshot describes the topology of the network at a particular time instant. When community detection is performed at time t, the clustering should be similar to the clustering at the previous time-step $t-1$, and at the same time should accurately incorporate the actual data. In the MKSC algorithm, this sort of temporal smoothness is included in the objective function of the primal problem of KSC through the maximization of the correlation between the actual and the previous models. Moreover, new smoothed measures describing the quality of a partitioning produced at a given time are proposed, and they are used both for model selection and for evaluating the test results. The MKSC method is utilized in two main frameworks: in the first one the smoothness is imposed by fixing the regularization constant $\nu = 1$ and the amount of smoothness is tuned by changing the memory M. In the second framework the smoothness gets activated automatically only when it is required, that is when a big change in the data structure occurs. As a consequence, ν must be properly tuned and it can also be used as a change detection measure. We have also proposed a tracking mechanism to detect important events characterizing the evolution of the communities over time. Moreover, two visualization tools based on a 3D embedding in the space of the dual solution vectors and on the network constructed by the tracking algorithm, respectively, are devised. On a number of synthetic and real-life networks we have shown how MKSC is able to produce high-quality partitioning in terms of classical measures such as Modularity and Conductance, and their smoothed version. Finally, comparisons with state-of-the-art methods like evolutionary spectral clustering (ESC), adaptive evolutionary clustering (AFFECT) and the Louvain method (LOUV) have shown that MKSC has a competitive performance.

In chapter 5, an application of KSC to an industrial use case is considered. In particular, accelerometer data are collected from a packing machine to monitor its conditions. In order to describe the ongoing degradation process due to the dirt accumulation in the sealing jaws, a windowing operation on the data has been applied. Then an optimal KSC model is trained offline to identify two main regimes, interpretable as normal and faulty state, respectively. Thanks to the out-of-sample extension property and a stationarity assumption, this model is used online to raise a warning when the machine is entering critical conditions. Moreover, we proposed also a soft clustering output that can be interpreted as "probability" to maintenance. In this way KSC could help to optimize the maintenances schedule and to maximize the

production capacity.

Finally, in chapter 6 an adaptive model named incremental kernel spectral clustering (IKSC) is introduced in order to work in a non-stationary environment. In the initialization stage a KSC model is constructed, then the latter is updated online according to the acquisition of new points. In particular the clustering model, expressed in terms of cluster prototypes in the eigenspace, is continuously adapted through the out-of-sample eigenvectors calculation. Moreover, the training set is formed only by the cluster centers in the input space, which are also updated in response to new data. In this way, the method has linear complexity and allows to properly track the evolution of complex patterns across time.

7.2 Perspectives

Some possible future research directions may consider the adaptation of the proposed algorithms or the development of new methods dealing with evolving data, in order to cope with challenging applications like video tracking, economic crisis prediction, sentiment analysis etc. Video tracking is the process of locating one or more moving objects over time, by properly associating the target objects in consecutive video frames. It has several uses, like security and surveillance, traffic control, medical imaging and many others. The prediction of an economic crisis is an ambitious challenge, mainly due to the complexity of the economic and financial markets and their non-stationary nature. Nevertheless, unsupervised kernel-based methods could help to catch interesting patterns in the data over time, which may be associated to a forthcoming catastrophe. In this way proper preventive actions can be undertaken to limit the damage. Sentiment analysis aims to determine the attitude of a speaker or a writer with respect to some topic or the overall contextual polarity of a document as positive, negative or neutral. The application to social networks data could allow to track the opinion of millions of people across time on several topics. For instance, this may be a useful tool for the ruling class to understand the impact of some policies and eventually to produce corrective actions.

Also, a new generation of kernel-based methods cast in a deep architecture, which can learn several levels of nonlinearities, can be developed. Deep learning was first proposed in the artificial neural networks (ANN) domain [16, 124, 83] and attracted much attention in the research community. Deep learning allows to learn different levels of abstraction or representations of the original data by stacking together several layers of neurons. Each layer performs unsupervised learning taking as inputs the outputs of the previous layer. In this way a rich understanding of the patterns at different scales is achieved and meaningful features are extracted. As a consequence, if an additional layer is added to the network to perform supervised learning, a

breakthrough accuracy in the learning task can be reached. These recent successful experiments are triggering the interest in exploring more powerful models also in other domains, like kernel methods [15]. In relation to the methods devised in this thesis, a novel hierarchical clustering model based on a multilayer architecture, inspired by [120], could for instance be developed. In particular, the specific models built at each layer can be combined such that the final model produces a result which takes into account the outcomes of the models in the previous layers. This kind of memory effect which propagates the information from layer to layer represents the same principle behind the MKSC algorithm described in chapter 4 in the context of clustering networks over time. Thus, in this case an analogous model where the time is replaced by a scale parameter could be designed.

Appendix A

Appendix

A.1 Cluster Quality Evaluation

In this Appendix we give a description of the main quality measures used in this thesis to evaluate and compare clustering results.

A.1.1 Data Clustering

One of the most important issues in cluster analysis is the evaluation of clustering results to find the partitioning that best captures the underlying structure of the data. Intuitively, any good algorithm should search for clusters whose members are close to each other and well separated. Many validity criteria can be found in the literature [102, 53]. Here we focus on the Silhouette [119] and the Davies-Bouldin Index (DBI).

Silhouette ranges from -1 to 1 and measures how similar data points in the same clusters are to each other compared to data points in other clusters. For each data point the more the Silhouette value is positive the more coherent this data point is to points in the same cluster. Negative values of the Silhouette indicates that this data point should be rather assigned to a different cluster. The definition of the Silhouette for the i-th datum is as follows:

$$\text{Sil}(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

where $a(i)$ is the average dissimilarity of data point i with all other data within the same cluster and $b(i)$ indicates the lowest average dissimilarity of i with the data

belonging to a different cluster. Finally, the mean Silhouette value (MSV) gives an average measure of the coherence of the groups.

In the case of the DBI a measure of similarity R_{ij} between each pair of clusters A_i and A_j is defined based on the ratio between the dispersion of the clusters s_i and s_j and their dissimilarity d_{ij}: $R_{ij} = \frac{(s_i+s_j)}{d_{ij}}$. Then DBI for k clusters becomes:

$$\text{DBI} = \frac{1}{k}\sum_{i=1}^{k} R_i$$

with $R_i = \max_{i\neq j} R_{ij}, i = 1, \ldots, k$. Roughly speaking DBI quantifies the average similarity between each cluster. Since it is desirable for the clusters to have the minimum possible similarity to each other good partitions correspond to a low value of DBI.

A.1.2 Community Detection

As we mentioned in Chapter 3 clustering network data is mainly referred in the literature as community detection. Given a network, we aim to find a partition such that the nodes within a community are densely connected and the connections are sparse between the communities. Also in this case there has been an extensive work related to assessing the quality of a division of a network into communities [84, 33, 70]. In this dissertation, we utilized two well known criteria: Modularity and Conductance.

Modularity is the most popular quality function of a graph introduced in [107, 105]. It is very well considered due to the fact that closely agrees with intuition on a wide range of real world networks. It is based on the idea that a random graph is not expected to have a cluster structure, so the possible existence of clusters can be revealed by the comparison between the actual density of inter-community edges and the density one would expect to have in the graph if the vertices were attached randomly, which defines the null model. Positive high values indicate the possible presence of a strong community structure. Moreover it has been shown that finding the Modularity of a network is analogous to finding the ground-state energy of a spin system [51]. In formulae the definition for an unweighted network[1] is as follows:

$$\text{Mod} = \frac{1}{2m}\sum_{ij}(S_{ij} - F_{ij})\delta_{ij}$$

with i, j indicating a pair of nodes. The sum runs over all pairs of vertices, S is the similarity matrix, m indicates the total number of edges, and F_{ij} represents the expected number of edges between vertices i and j in the null model. The δ_{ij} function

[1] For weighted graphs an analogue definition has been provided in [104].

yields 1 if vertices i and j are in the same community and 0 otherwise. Since the standard null model of Modularity imposes that the expected degree sequence matches the actual degree sequence of the graph, the Modularity can be written as: $\text{Mod} = \frac{1}{2m}\sum_{ij}(S_{ij} - \frac{d_i d_j}{2m})\delta_{ij}$, where we indicate with $d_i = \sum_j S_{ij}$ the degree of the vertex i. Thus, a high value of Mod says that the number of inter-community edges of the graph are few in a statistically significant sense, meaning the presence of a strong community-like structure. Moreover, after some linear algebra calculations [61], we can write Modularity as:

$$\text{Mod} = \frac{1}{2m}\text{tr}(X^T M X)$$

where $M = S - \frac{1}{2m}dd^T$ is the Modularity matrix, $d = [d_1, \ldots, d_N]$ indicates the vector of the degrees of each node and X represents the cluster indicator matrix. Finally, as it has been pointed out in [48] and [43], Modularity suffers from some drawbacks:

- resolution limit: it contains an intrinsic scale that depends on the total number of links in the network. Modules that are smaller than this scale may not be resolved
- exhibits degeneracies: it typically admits an exponential number of distinct high-scoring solutions and often lacks a clear global maximum
- it does not capture overlaps among communities in real networks.

Many of the aforementioned problems of Modularity have been solved by properly modifying its definition, as for example in [9] where a multi-resolution Modularity has been introduced. In this case, however, as pointed out in [71], multi-resolution Modularity has a double simultaneous bias: it leads to a splitting of large clusters and a merging of small clusters, and both problems cannot be usually handled at the same time for any value of the resolution parameter.

Another meaningful quality function is the Conductance. For every community in a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the Conductance is defined as the ratio between the number of edges leaving the cluster and the number of edges inside the cluster [63], [84]. In particular, the Conductance denoted by $\text{Cond}(S)$ of a set of nodes $\mathcal{A} \subset \mathcal{V}$ is:

$$\text{Cond}(S) = c_S / \min(Vol(S), Vol(V \setminus S))$$

where c_S denotes the size of the edge boundary, $c_S = \{(u, v) : u \in S, v \notin S\}$, and $Vol(S) = \sum_{u \in S} d_u$, with d_u representing the degree of node u. In other words the Conductance describes the concept of a good network community as a set of nodes that has a better internal than external connectivity. Therefore the lower the Conductance the better the community structure.

A.1.3 Comparing Partitions

In order to compare two clustering results two measures of agreement between partitions have been utilized in this thesis: the Adjusted Rand Index (ARI) and the Normalized Mutual Information (NMI).

Given two partitions $\mathcal{U}$ and $\mathcal{V}$, the ARI [58] equals 1 when the related cluster memberships agree completely, 0 in case of complete mismatch with equal number of positive and negative mistakes, and -1 for a complete disagreement with all negative mistakes[2]. Its most common use is as measure of agreement between the clustering outcomes of an algorithm and a known grouping which acts like a ground-truth.

The NMI [126] measures the statistical information shared between the partitions $\mathcal{U}$ and $\mathcal{V}$ it also takes values in the range $[0, 1]$. It tells us how much knowing one of these clusterings reduces our uncertainty about the other. The higher the NMI, the more useful the information in $\mathcal{V}$ helps us to predict the cluster memberships in $\mathcal{U}$ and vice versa.

[2] A positive mistake means that if a partition groups a pair of points in the same cluster, the other does not. A negative mistake means that a pair of points is grouped in different clusters according to an algorithm while the points are assigned to the same cluster by the other algorithm.

Bibliography

[1] Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. A framework for clustering evolving data streams. In *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*, VLDB '03, pages 81–92, 2003.

[2] O. M. Agudelo, P. Viaene, L. Blyth, and B. De Moor. Application of data assimilation techniques to the air quality model aurora. *Internal Report 12-134, ESAT-SISTA, KU Leuven (Leuven, Belgium)*, 2012.

[3] Edoardo M. Airoldi, David M. Blei, Stephen E. Fienberg, and Eric P. Xing. Mixed membership stochastic blockmodels. *J. Mach. Learn. Res.*, 9:1981–2014, June 2008.

[4] C. Alzate and J. A. K. Suykens. Sparse kernel models for spectral clustering using the incomplete cholesky decomposition. In *Proc. of the 2008 International Joint Conference on Neural Networks (IJCNN 2008)*, pages 3555–3562, 2008.

[5] C. Alzate and J. A. K. Suykens. Highly sparse kernel spectral clustering with predictive out-of-sample extensions. In *Proc. of the 18th European Symposium on Artificial Neural Networks (ESANN 2010)*, pages 235–240, 2010.

[6] C. Alzate and J. A. K. Suykens. Multiway spectral clustering with out-of-sample extensions through weighted kernel PCA. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(2):335–347, February 2010.

[7] C. Alzate and J. A. K. Suykens. Out-of-sample eigenvectors in kernel spectral clustering. In *Proc. of the International Joint Conference on Neural Networks (IJCNN 2011)*, pages 2349–2356, 2011.

[8] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):898–916, May 2011.

[9] A Arenas, A Fernández, and S Gómez. Analysis of the structure of complex networks at different resolution levels. *New Journal of Physics*, 10(5):053039, 2008.

[10] Sitaram Asur, Srinivasan Parthasarathy, and Duygu Ucar. An event-based framework for characterizing the evolutionary behavior of interaction graphs. *ACM Trans. Knowl. Discov. Data*, 3(4):16:1–16:36, 2009.

[11] F. R. Bach and M. I. Jordan. Learning spectral clustering, with application to speech separation. *Journal of Machine Learning Research*, 7:1963–2001, 2006.

[12] C. Baker. *The numerical treatment of integral equations*. Clarendon Press, Oxford, 1977.

[13] A.-L. Barabasi, S. P. Jeong, H. Mason, and Z. N. Oltvai. Lethality and centrality in protein networks. *Nature*, 6833(411):41–42, 2001.

[14] Adi Ben-Israel and Cem Iyigun. Probabilistic d-clustering. *J. Classif.*, 25(1):5–26, 2008.

[15] Samy Bengio, Li Deng, Hugo Larochelle, Honglak Lee, and Ruslan Salakhutdinov. Guest editors' introduction: Special section on learning deep architectures. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1795–1797, 2013.

[16] Yoshua Bengio. Learning deep architectures for AI. *Found. Trends Mach. Learn.*, 2(1):1–127, January 2009.

[17] P. Berkhin. A survey of clustering data mining techniques. *Grouping Multidimensional Data*, pages 25–71, 2006.

[18] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[19] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[20] András Bóta, Miklós Krész, and András Pluhár. Dynamic communities and their detection. *Acta Cybernetica*, 20(1):35–52, 2011.

[21] Habiboulaye Amadou Boubacar, Stéphane Lecoeuche, and Salah Maouche. Sakm: Self-adaptive kernel machine a kernel-based algorithm for online clustering. *Neural Networks*, 21(9):1287 – 1301, 2008.

[22] Piotr Brodka, Stanislaw Saganowski, and Przemyslaw Kazienko. Ged: the method for group evolution discovery in social networks. *Social Network Analysis and Mining*, 3(1):1–14, 2013.

[23] Fazli Can. Incremental clustering for dynamic information processing. *ACM Trans. Inf. Syst.*, 11(2):143–164, April 1993.

[24] A. Capocci, V.D.P. Servedio, G. Caldarelli, and F. Colaior. Detecting communities in large networks. *Physica A: Statistical Mechanics and its Applications*, 352(24):669 – 676, 2005.

[25] Deepayan Chakrabarti, Ravi Kumar, and Andrew Tomkins. Evolutionary clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 554–560, New York, NY, USA, 2006. ACM.

[26] Sanjay Chakraborty and N.K. Nagwani. Analysis and study of incremental k-means clustering algorithm. In *High Performance Architecture and Grid Computing*, volume 169 of *Communications in Computer and Information Science*, pages 338–341, 2011.

[27] Yun Chi, Xiaodan Song, Dengyong Zhou, Koji Hino, and Belle L. Tseng. Evolutionary spectral clustering by incorporating temporal smoothness. In *KDD '07*, pages 153–162, 2007.

[28] S. W. Choi, C. K. Yoo, and I.-B. Lee. Overall statistical monitoring of static and dynamic patterns. *Ind. Eng. Chem. Res.*, 42:108 – 117, 2003.

[29] F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

[30] Aaron Clauset, M. E. J. Newman, and Cristopher Moore. Finding community structure in very large networks. *Phys. Rev. E*, 70:066111, Dec 2004.

[31] Jeff Clune, Jean-Baptiste Mouret, and Hod Lipson. The evolutionary origins of modularity. *Proceedings of the Royal Society B: Biological Sciences*, 280(1755):20122863, March 2013.

[32] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20(3):273–297, 1995.

[33] Leon Danon, Albert D. Guilera, Jordi Duch, and Alex Arenas. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, (9):P09008–09008, September 2005.

[34] David L. Davies and Donald W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2):224–227, 1979.

[35] B. De Ketelaere, K. Mertens, F. Mathijs, D. Sabin Diaz, and J. De Baerdemaeker. Nonstationarity in statistical process control-issues, cases, ideas. *Applied Stochastic Models in Business and Industry*, 27(4):367–376, 2011.

[36] J. C. Delvenne, S. N. Yaliraki, and M. Barahona. Stability of graph communities across time scales. *Proceedings of the National Academy of Sciences*, 107(29):12755–12760, July 2010.

[37] Charanpal Dhanjal, Romaric Gaudel, and Stephan Clemenccon. Efficient eigen-updating for spectral graph clustering. *arXiv/1301.1318*, 2013.

[38] Luca Donetti and Miguel A Mu noz. Detecting network communities: a new systematic and efficient algorithm. *Journal of Statistical Mechanics: Theory and Experiment*, 2004(10):P10012, 2004.

[39] Nathan Eagle, Alex Sandy Pentland, and David Lazer. Inferring social network structure using mobile phone data. *PNAS*, 106(1):15274–15278, 2009.

[40] Eionet. European topic centre on air and climate change, 2011. [online] `http://air-climate.eionet.europa.eu/databases/airbase`.

[41] T. S. Evans and R. Lambiotte. Line graphs, link partitions, and overlapping communities. *Physical Review E*, 80(1):016105+, 2009.

[42] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, 2010.

[43] Santo Fortunato and Marc Barthélemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41, 2007.

[44] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the Nyström method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):214–225, February 2004.

[45] Katrijn Frederix and Marc Van Barel. Sparse spectral clustering method based on the incomplete cholesky decomposition. *J. Comput. Appl. Math.*, 237(1):145–161, January 2013.

[46] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977.

[47] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[48] B.H. Good, Y.A. De Montjoye, and A. Clauset. Performance of modularity maximization in practical contexts. *Physical Review E*, 81(4):046106, 2010.

[49] Derek Greene, Donal Doyle, and Padraig Cunningham. Tracking the evolution of communities in dynamic social networks. In *Proceedings of the 2010 International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '10, pages 176–183, Washington, DC, USA, 2010. IEEE Computer Society.

[50] Sudipto Guha, Adam Meyerson, Nina Mishra, Rajeev Motwani, and Liadan O'Callaghan. Clustering data streams: Theory and practice. *IEEE Trans. on Knowl. and Data Eng.*, 15(3):515–528, March 2003.

[51] R. Guimera, M. Sales-Pardo, and L.A.N. Amaral. Modularity from fluctuations in random graphs and complex networks. *Physical Review E*, 70(2):025101, 2004.

[52] Chetan Gupta and Robert L. Grossman. Genic: A single-pass generalized incremental algorithm for clustering. In *SDM*. SIAM, 2004.

[53] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17:107–145, 2001.

[54] Matthias Hein, Jean-Yves Audibert, and Ulrike von Luxburg. Graph Laplacians and their convergence on random neighborhood graphs. *Journal of Machine Learning Research*, 8:1325–1370, 2007.

[55] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1996.

[56] L Hoegaerts, L De Lathauwer, Ingeborg Goethals, J A K Suykens, J Vandewalle, and B De Moor. Efficiently updating and tracking the dominant kernel principal components. *Neural Networks*, 20(2):220–229, 2007.

[57] Hsin-Chien Huang, Yung-Yu Chuang, and Chu-Song Chen. Multiple kernel fuzzy clustering. *IEEE T. Fuzzy Systems*, 20(1):120–134, 2012.

[58] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 1(2):193–218, 1985.

[59] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Comput. Surv.*, 31(3), September 1999.

[60] Anil K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recogn. Lett.*, 31(8):651–666, June 2010.

[61] Jeffrey Q. Jiang, Andreas W.M. Dress, and Genke Yang. A spectral clustering-based framework for detecting community structures in complex networks. *Applied Mathematics Letters*, 22(9):1479–1482, 2009.

[62] Yoonseop Kang and Seungjin Choi. Kernel PCA for community detection. In *Business Intelligence Conference*, 2009.

[63] Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clusterings: Good, bad and spectral, 2004.

[64] Donald E. Knuth. *The art of computer programming, volume 2 (3rd ed.): seminumerical algorithms*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997.

[65] Theodora Kourti and John F. MacGregor. Process analysis, monitoring and diagnosis, using multivariate projection methods. *Chemometrics and Intelligent Laboratory Systems*, 28(1):3 – 21, 1995.

[66] H.-P. Kriegel, P. Kroger, and A. Zimek. Outlier detection techniques. *16th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2010.

[67] Harold W. Kuhn. The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly*, 2(1–2):83–97, March 1955.

[68] Renaud Lambiotte. Multi-scale modularity in complex networks. In *Proceedings of the 8th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*, pages 546–553. IEEE, 2010.

[69] Andrea Lancichinetti and Santo Fortunato. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Phys. Rev. E*, 80(1):016118, Jul 2009.

[70] Andrea Lancichinetti and Santo Fortunato. Community detection algorithms: a comparative analysis, 2009.

[71] Andrea Lancichinetti and Santo Fortunato. Limits of modularity maximization in community detection. *Phys. Rev. E*, 84:066122, Dec 2011.

[72] Andrea Lancichinetti, Santo Fortunato, and János Kertész. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics*, 11(3):033015, 2009.

[73] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Phys. Rev. E*, 78:046110, Oct 2008.

[74] C. Lanczos. Iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of Research of the National Bureau of Standards*, 1950.

[75] R. Langone, O. M. Agudelo, B. De Moor, and J. A. K. Suykens. Incremental kernel spectral clustering for online learning of non-stationary data. *Neurocomputing*, (139):246–260, September 2014.

[76] R. Langone, C. Alzate, and J. A. K. Suykens. Modularity-based model selection for kernel spectral clustering. In *Proc. of the International Joint Conference on Neural Networks (IJCNN 2011)*, pages 1849–1856, 2011.

[77] R. Langone, R. Mall, and J. A. K. Suykens. Clustering data over time using kernel spectral clustering with memory. *Internal Report 14-33, ESAT-STADIUS, KU Leuven (Leuven, Belgium)*, 2014.

[78] Rocco Langone, Carlos Alzate, Bart De Ketelaere, and Johan A. K. Suykens. Kernel spectral clustering for predicting maintenance of industrial machines. In *IEEE Symposium Series on Computational Intelligence (SSCI) 2013*, 2013.

[79] Rocco Langone, Carlos Alzate, and Johan A. K. Suykens. Kernel spectral clustering for community detection in complex networks. In *IJCNN*, pages 2596–2603. IEEE, 2012.

[80] Rocco Langone, Carlos Alzate, and Johan A. K. Suykens. Kernel spectral clustering with memory effect. *Physica A: Statistical Mechanics and its Applications*, 392(10):2588–2606, 2013.

[81] Rocco Langone, Raghvendra Mall, and Johan A. K. Suykens. Soft kernel spectral clustering. In *Proc. of the International Joint Conference on Neural Networks (IJCNN 2013)*, pages 1028 – 1035, 2013.

[82] Rocco Langone and Johan A. K. Suykens. Community detection using kernel spectral clustering with memory. *Journal of Physics: Conference Series*, 410(1):012100, 2013.

[83] Quoc V. Le, M. Ranzato, Rajat Monga, Matthieu Devin, Greg Corrado, Kai Chen, Jeffrey Dean, and Andrew Y. Ng. Building high-level features using large scale unsupervised learning. In *ICML*, 2012.

[84] J. Leskovec, K. J. Lang, and M. W. Mahoney. Empirical comparison of algorithms for network community detection. In *Proc. of WWW 2010)*, pages 631–640, 2010.

[85] Jure Leskovec and Christos Faloutsos. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 631–636, 2006.

[86] T. Warren Liao. Clustering of time series data - a survey. *Pattern Recognition*, 38(11):1857 – 1874, 2005.

[87] Frank Lin and William W. Cohen. Power iteration clustering. In *ICML*, pages 655–662, 2010.

[88] Yu-Ru Lin, Yun Chi, Shenghuo Zhu, Hari Sundaram, and Belle L. Tseng. Analyzing communities and their evolutions in dynamic social networks. *ACM Trans. Knowl. Discov. Data*, 3(2):8:1–8:31, 2009.

[89] David Lusseau, Karsten Schneider, Oliver J Boisseau, Patti Haase, Elisabeth Slooten, and Steve M Dawson. The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, 54(4):396–405, 2003.

[90] A. S. Maiya and T. Y Berger-Wolf. Sampling community structure. In *Proc. 19th ACM Intl. Conference on the World Wide Web (WWW '10)*, 2010.

[91] R. Mall, R. Langone, and J. A. K Suykens. Highly Sparse Reductions to Kernel Spectral Clustering. In *5th International Conference on Pattern Recognition and Machine Intelligence*, pages 163–169, 2013.

[92] R. Mall, R. Langone, and J. A. K. Suykens. Agglomerative hierarchical kernel spectral clustering for large scale networks. In *Proc. of the European Symposium on Artitficial Neural Networks (ESANN)*, April 2014.

[93] R. Mall, R. Langone, and J. A. K. Suykens. Multilevel hierarchical kernel spectral clustering for real-life large scale complex networks. *PLOS ONE*, in press 2014.

[94] R. Mall, R. Langone, and J.A.K Suykens. FURS: Fast and unique representative subset selection retaining large scale community structure. *Social Network Analysis and Mining*, 3(4):1–21, 2013.

[95] R. Mall, S. Mehrkanoon, R. Langone, and J. A. K. Suykens. Optimal reduced sets for sparse kernel spectral clustering. In *Proc. of the International Joint Conference on Neural Networks (IJCNN 2014)*, in press 2014.

[96] Raghvendra Mall, Rocco Langone, and Johan A. K. Suykens. Kernel spectral clustering for big data networks. *Entropy (Special Issue on Big Data)*, 15(5):1567–1586, 2013.

[97] Raghvendra Mall, Rocco Langone, and Johan A. K. Suykens. Self-Tuned Kernel Spectral Clustering for Large Scale Networks. In *IEEE International Conference on Big Data*, 2013.

[98] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, 2001.

[99] M. Meila and J. Shi. Learning segmentation by random walks. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*. MIT Press, 2001.

[100] M. Meila and J. Shi. A random walks view of spectral segmentation. In *Artificial Intelligence and Statistics AISTATS*, 2001.

[101] S. Mika, B. Schölkopf, A. J. Smola, K. R. Müller, M. Scholz, and G. Rätsch. Kernel PCA and de-noising in feature spaces. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*. MIT Press, 1999.

[102] Glenn Milligan and Martha Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2):159–179, June 1985.

[103] Peter J. Mucha, Thomas Richardson, Kevin Macon, Mason A. Porter, and Jukka-Pekka Onnela. Community structure in time-dependent, multiscale, and multiplex networks. *Science*, 328(5980):876–878, 2010.

[104] M. E. J. Newman. Analysis of weighted networks. *Phys. Rev. E*, 70(5):056131, Nov 2004.

[105] M. E. J. Newman. Modularity and community structure in networks. *Proc. Natl. Acad. Sci. USA*, 103(23):8577–8582, 2006.

[106] M. E. J. Newman. Communities, modules and large-scale structure in networks. *Nature Physics*, 8(1):25–31, December 2011.

[107] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2), 2004.

[108] 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*, pages 849–856, Cambridge, MA, 2002. MIT Press.

[109] Huazhong Ning, Wei Xu, Yun Chi, Yihong Gong, and Thomas S. Huang. Incremental spectral clustering with application to monitoring of evolving blog communities. In *SDM*. SIAM, 2007.

[110] Huazhong Ning, Wei Xu, Yun Chi, Yihong Gong, and Thomas S. Huang. Incremental spectral clustering by efficiently updating the eigen-system. *Pattern Recogn.*, 43(1):113–127, January 2010.

[111] K. Nowicki and T. A. B. Snijders. Estimation and prediction for stochastic blockstructures. *Journal of the American Statistical Association*, 96(455):1077–1087, September 2001.

[112] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, June 2005.

[113] Gergely Palla, Albert laszlo Barabasi, and Tamás Vicsek. Quantifying social group evolution. *Nature*, 446:2007, 2007.

[114] D. Peluffo, S. Garcia, R. Langone, J. A. K. Suykens, and G. Castellanos. Kernel spectral clustering for dynamic data using multiple kernel learning. In *Proc. of the International Joint Conference on Neural Networks (IJCNN 2013)*, pages 1085 – 1090, 2013.

[115] J. Puzicha, T. Hofmann, and J. Buhmann. Non-parametric similarity measures for unsupervised texture segmentation and image retrieval. In *Computer Vision and Pattern Recognition*, pages 267–272, 1997.

[116] Filippo Radicchi, Claudio Castellano, Federico Cecconi, Vittorio Loreto, and Domenico Parisi. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(9):2658–2663, 2004.

[117] J. Reichardt and S. Bornholdt. Statistical mechanics of community detection. *Phys Rev E Stat Nonlin Soft Matter Phys*, 74(1 Pt 2), July 2006.

[118] Martin Rosvall and Carl T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123, January 2008.

[119] P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20(1):53–65, 1987.

[120] Ruslan Salakhutdinov, Joshua B. Tenenbaum, and Antonio Torralba. Learning with hierarchical-deep models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1958–1971, 2013.

[121] Peter Sarlin. Self-organizing time map: An abstraction of temporal multivariate patterns. *Neurocomputing*, 99:496–508, 2013.

[122] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[123] Herbert A. Simon. The architecture of complexity. In *Proceedings of the American Philosophical Society*, pages 467–482, 1962.

[124] Steve Smale, Lorenzo Rosasco, Jake V. Bouvrie, Andrea Caponnetto, and Tomaso Poggio. Mathematics of the neural response. *Foundations of Computational Mathematics*, 10(1):67–91, 2010.

[125] Tom A. B. Snijders and Krzysztof Nowicki. Estimation and Prediction for Stochastic Blockmodels for Graphs with Latent Block Structure. *Journal of Classification*, 14(1):75–100, January 1997.

[126] A. Strehl and J. Ghosh. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617, 2002.

[127] Jimeng Sun, Christos Faloutsos, Spiros Papadimitriou, and Philip S. Yu. Graphscope: parameter-free mining of large time-evolving graphs. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '07, pages 687–696, New York, NY, USA, 2007. ACM.

[128] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *Least Squares Support Vector Machines*. World Scientific, Singapore, 2002.

[129] J. A. K. Suykens, T. Van Gestel, J. Vandewalle, and B. De Moor. A support vector machine formulation to PCA analysis and its kernel version. *IEEE Transactions on Neural Networks*, 14(2):447–450, March 2003.

[130] J. A. K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Process. Lett.*, 9(3):293–300, 1999.

[131] T. Van Gestel, J. A. K. Suykens, G. Lanckriet, A. Lambrechts, B. De Moor, and J. Vandewalle. Multiclass LS-SVMs: Moderated outputs and coding-decoding schemes. *Neural Process. Lett.*, 15(1):45–58, February 2002.

[132] Tony Van Gestel, Johan A. K. Suykens, Bart Baesens, Stijn Viaene, Jan Vanthienen, Guido Dedene, Bart de Moor, and Joos Vandewalle. Benchmarking least squares support vector machine classifiers. *Machine Learning*, 54(1):5–32, 2004.

[133] V. Venkatasubramanian, R. Rengaswamy, and S.N. Kavuri. A review of process fault detection and diagnosis. part i: Quantitative model-based methods. *Computers and chemical engineering*, 27(3):293–311, 2003.

[134] V. Venkatasubramanian, R. Rengaswamy, and S.N. Kavuri. A review of process fault detection and diagnosis. part ii: Qualitative models and search strategies. *Computers and chemical engineering*, 27(3):313–326, 2003.

[135] V. Venkatasubramanian, R. Rengaswamy, and S.N. Kavuri. A review of process fault detection and diagnosis. part iii: Process history based methods. *Computers and chemical engineering*, 27(3):327–346, 2003.

[136] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[137] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, (393):440–442, 1998.

[138] C. K. I. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*. MIT Press, 2001.

[139] Kevin S. Xu, Mark Kliger, and Alfred O. Hero III. Adaptive evolutionary clustering. *Data Mining and Knowledge Discovery*, pages 1–33, 2013.

[140] Jaewon Yang and Jure Leskovec. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 587–596, 2013.

[141] Kai Yu, Shipeng Yu, and Volker Tresp. Soft clustering on graphs. In *in Advances in Neural Information Processing Systems*, page 05, 2005.

[142] W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33:452–473, 1977.

[143] Lihi Zelnik-manor and Pietro Perona. Self-tuning spectral clustering. In *Advances in Neural Information Processing Systems 17*, pages 1601–1608. MIT Press, 2004.

Publications by the Author

Journal papers

2009

1. A. PASINI, G. SZPUNAR, G. AMORI, R. LANGONE, M. CRISTALDI Assessing Climatic Influences on Rodent Density-A Neural Network Modelling Approach and a Case Study in Central Italy. *Asia-Pacific Journal of Atmospheric Sciences (2009), 45, 319-330.*

2010

1. A. PASINI, R. LANGONE Attribution of precipitation changes on a regional scale by neural network modelling: A case study. *Water* (2010), 2, 3, 321–332.

2. A. PASINI, R. LANGONE, F. MAIMONE, V. PELINO Energy-based predictions in Lorenz system by a unified formalism and neural network modelling. *Nonlinear Processes in Geophysics* (2010), 17, 809–815.

2012

1. A. PASINI, R. LANGONE Influence of Circulation Patterns on Temperature Behavior at the Regional Scale: A Case Study Investigated via Neural Network Modelling. *Journal of Climate* (2012), 25, 2123–2128.

2013

1. R. LANGONE, C. ALZATE, J. A. K. SUYKENS Kernel spectral clustering with memory effect. *Physica A: Statistical Mechanics and its Applications*, (2013), 10, 2588–2606.

2. R. MALL, R. LANGONE, J. A. K. SUYKENS FURS: Fast and unique representative subset selection retaining large scale community structure. *Social Network Analysis and Mining* (2013), 3, 4, 1–21.

3. R. MALL, R. LANGONE, J. A. K. SUYKENS Kernel spectral clustering for big data networks. *Entropy 15 (Special Issue on Big Data)* (2013), 5, 1567–1586.

2014

1. R. LANGONE, O. M. AGUDELO, B. DE MOOR, J. A. K. SUYKENS Incremental kernel spectral clustering for online learning of non-stationary data. *Neurocomputing* (2014), 139, 246-260.

2. R. MALL, R. LANGONE, J. A. K. SUYKENS Multilevel Hierarchical Kernel Spectral Clustering for Real-Life Large Scale Complex Networks. *PLOS ONE* (2014), in press.

Conference proceedings

2007

1. A. PASINI, G. SZPUNAR, G. AMORI, R. LANGONE, M. CRISTALDI Meteo-Climatic influences on rodent density in central Italy: a nonlinear modelling approach. In *The 6th European Conference on Ecological Modelling, ECEM'07* Trieste, Italy, 27-30 November 2007.

2009

1. A. PASINI, R. LANGONE Major Influences of Circulation Patterns on Temperatures in the Italian Side of the Greater Alpine Region: an Investigation via Neural Network Modelling. In *21th Conference on Climate Variability and Change, Seventh Conference on Artificial Intelligence and its Applications to the Environmental Sciences* (2009), Phoenix, Arizona.

2011

1. R. LANGONE, C. ALZATE, J. A. K. SUYKENS Modularity-based model selection for kernel spectral clustering. In *Proc. of the International Joint Conference on Neural Networks (IJCNN)* (2011), 1849–1856.

2012

1. R. LANGONE, C. ALZATE, J. A. K. SUYKENS Kernel spectral clustering for community detection in complex networks. In *Proc. of the International Joint Conference on Neural Networks IJCNN* (2012), 2596–2603.

2. R. LANGONE, J. A. K. SUYKENS Community detection using kernel spectral clustering with memory. *Journal of Physics: Conference Series 410* (2013), 1, 012100.

2013

1. R. LANGONE, R. MALL, J. A. K. SUYKENS Soft kernel spectral clustering. In *Proc. of the International Joint Conference on Neural Networks (IJCNN)* (2013), 1028–1035.

2. R. LANGONE, C. ALZATE, B. DE KETELAERE, J. A. K. SUYKENS Kernel spectral clustering for predicting maintenance of industrial machines. In *IEEE Symposium Series on Computational Intelligence (SSCI)* (2013), 39–45.

3. R. MALL, R. LANGONE, J. A. K. SUYKENS Highly Sparse Reductions to Kernel Spectral Clustering. In *5th International Conference on Pattern Recognition and Machine Intelligence* (2013), 163–169.

4. R. MALL, R. LANGONE, J. A. K. SUYKENS Self-Tuned Kernel Spectral Clustering for Large Scale Networks. In *IEEE International Conference on Big Data* (2013), Santa Clara (USA).

5. D. PELUFFO, S. GARCIA, R. LANGONE, J. A. K. SUYKENS, G. CASTELLANOS Kernel spectral clustering for dynamic data using multiple kernel learning. In *Proc. of the International Joint Conference on Neural Networks (IJCNN)* (2013), 1085–1090.

2014

1. R. MALL, R. LANGONE, J. A. K. SUYKENS Agglomerative Hierarchical Kernel Spectral Clustering for Large Scale Networks. In *ESANN, Special Session: Learning and Modelling Big Data.* (2014).

2. R. MALL, S. MEHRKANOON, R. LANGONE, J. A. K. SUYKENS Optimal Reduced Sets for Sparse Kernel Spectral Clustering. In *Proc. of the International Joint Conference on Neural Networks (IJCNN)* (2014), in press.

Abstracts

2007

1. G. SZPUNAR, G. AMORI, A. PASINI, R. LANGONE, M. CRISTALDI Rodent density and climate in Central Italy: a modelling approach. In *14th Int. Symp. On Environmental Pollution and its impact on Life in the Mediterranean Region*, Sevilla, 2007.

2011

1. R. LANGONE, J. A. K. SUYKENS Community detection via kernel-based methods. In *DYSCO Study Day.* November 29, 2011.

2. R. LANGONE, J. A. K. SUYKENS Kernel-based methods for optimal maintenance of industrial machines. In *Predictive maintenance in industry, final POM workshop.* March 30, 2011, Gent (Belgium).

2012

1. R. LANGONE, J. A. K. SUYKENS Community detection via kernel spectral clustering. In *31st Benelux meeting on Systems and Controls.* March 27-29, 2012.

2. R. LANGONE, J. A. K. SUYKENS Kernel spectral clustering (software demo). In *OPTEC retreat WG2 Meeting.* November 9, 2012.

2013

1. R. LANGONE, J. A. K. SUYKENS Incremental Kernel Spectral Clustering. In *International Conference on Mathematical Modelling in Physical Sciences (IC-MSQUARE)* (2013), Prague, Czech Republic.

2014

1. B. PICCART, R. LANGONE Prediction of remaining useful lifetime of industrial machinery. In *Innovative solutions for predictive maintenance and performance optimization in industry, final workshop POM2 project* (2014), March 25, Leuven.

2. R. LANGONE, R. MALL, J. A. K. SUYKENS Clustering data over time using kernel spectral clustering with memory. In *DYSCO Study Day.* May 16, 2014.

Biography

Rocco Langone was born in Potenza (Italy) in 1983, where he spent his high school years. In 2002 he moved to Rome to study Physics at Sapienza University of Rome. In 2005 he received a Bachelor's degree in Physics and Information Technology. In March 2008 he obtained the Master's degree in Physics with the thesis: *A neural network model for studying the attribution of global circulation atmospheric patterns on the climate at a local scale*. Until September 2008 he worked as researcher at the National Research Council (CNR) in Rome. He received a second Master's degree in Scientific Computing in April 2010, with the thesis: *Stochastic Volatility models for European calls option pricing*. He is currently a doctoral student in machine learning, at the STADIUS research division of the Department of Electrical Engineering (ESAT) of the KU Leuven, under the supervision of prof. Johan A. K. Suykens. During his Phd Rocco served as teaching assistant for the course *Artificial Neural Networks*, included in the master in Artificial Intelligence organized by the KU Leuven. He has also been involved in the *Prognostic for Optimal Maintenance* projects *POM 1 and POM 2*, and as member of the *Optimization in Engineering Center (OPTEC)* of the KU Leuven he actively participated in the meetings and workshops and he also took part in the DYSCO study days. Rocco's scientific interests include kernel methods, neural networks, optimization, unsupervised learning (clustering and community detection), time-series analysis, big data, climate, finance, fault detection.



FACULTY OF ENGINEERING
DEPARTMENT OF ELECTRICAL ENGINEERING (ESAT)
STADIUS CENTER FOR DYNAMICAL SYSTEMS, SIGNAL PROCESSING AND DATA ANALYTICS
Kasteelpark Arenberg 10
B-3001 Heverlee
rocco.langone@esat.kuleuven.be
http://www.esat.kuleuven.be