Kernel Spectral Clustering with Memory Effect

Rocco Langone, Carlos Alzate and Johan A. K. Suykens

Department of Electrical Engineering ESAT-SCD-SISTA, Katholieke Universiteit Leuven, Kasteelpark Arenberg 10, B-3001 Leuven, Belgium
Email: {rocco.langone, carlos.alzate, johan.suykens}@esat.kuleuven.be.

Abstract

Evolving graphs describe many natural phenomena changing over time, such as social relationships, trade markets, methabolic networks etc. In this framework, performing community detection and analyzing the cluster evolution represents a critical task. Here we propose a new model for this purpose, where the smoothness of the clustering results over time can be considered as a valid prior knowledge. It is based on a constrained optimization formulation typical of Least Squares Support Vector Machines (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 history. We also propose new model selection criteria in order to carefully choose the hyper-parameters of our model, which is a crucial issue to achieve good performances. We successfully test the model on four toy problems and on a real world network. We also compare our model with Evolutionary Spectral Clustering, which is a state-of-the-art algorithm for community detection of evolving networks, illustrating that the kernel spectral clustering with memory effect can achieve better or equal performances.

Keywords: kernel spectral clustering, community detection, evolving networks, temporal smoothness, memory.

1. Introduction

In many practical applications we face the problem of community detection in dynamic scenarios, which recently gained much attention in the science community [26, 25, 6, 4, 29]. We aim to cluster objects whose characteristics change over time and we wish to obtain a meaningful clustering result at each time step. This situation arises for instance in segmentation of moving objects in a dynamic environment, community detection of evolving networks such as in the blog data, telephone traffic data, time-series microarray data etc. A desirable feature of a clustering model which has to capture the evolution of communities over time is the temporal smoothness between clusters in successive timesteps. In this way the model is able to track the long-term trend and at the same time it smoothens out short-term variation due to noise, similarly to what happens with moving averages in time-series analysis. This issue was first addressed in [8] and [7].

In this paper our starting point is the kernel spectral clustering (KSC) model, which represents a spectral clustering formulation as a weighted kernel PCA problem with primal and dual representations [1]. The dual problem is an eigenvalue problem, related to spectral clustering. As in classical spectral clustering, if we are given data points we build-up a weighted graph where

each node represents a point and the weights are related to the similarity between the points, according to a particular similarity measure. In what follows for the sake of simplicity we will refer to nodes and networks (or graphs) either if we are given data points or we start our analysis directly from a graph.

The KSC has two main advantages:

- a systematic model selection scheme for the tuning of the parameters
- the extension of the clustering model to out-of-sample nodes.

The clustering model can be trained on a subset of the whole network (by solving an affordable eigenvalue problem) and then be applied to the rest of the network in a learning framework. The out-of-sample extension allows then to predict the memberships of a new node thanks to the model learned during the training phase. This is a unique feature in the community detection field, since most of the algorithms work only at the training level and the out-of-sample extension (if any) is done in a heuristic way. Also in [11], the out-of-sample extension performed by the kernel *k*-means model is based on the same assignment rule used for training, but there is not an underlying predictive model like in the KSC case.

The KSC model is static, in the sense that there is no temporal information in it. Then, in order to cluster an evolving network, we can apply the model to each snapshot separately. This procedure, as we discussed, is not advisable if we want to track the long-term drift of the communities by ignoring meanwhile the short-term fluctuations.

In this framework, we propose an extension of the KSC model that we call kernel spectral clustering with memory effect (MKSC). In this new formulation we incorporate the desired temporal smoothness at the primal level, as a kind of prior knowledge incorporation. This leads to a dual problem which is no longer an eigenvalue problem but a set of linear systems.

The rest of this paper is organized as follows: Section 2 summarizes the KSC model. Section 3 introduces our new contribution, that is the MKSC model. In Section 4 we propose new cluster quality measures, we describe the model selection issue for KSC and MKSC and we briefly review the evolutionary spectral clustering algorithm. Some simulation results on both real-life and artificial datasets are presented in Section 5: we compare the MKSC model with KSC and the evolutionary spectral clustering. In Section 6 we discuss the drawbacks of Modularity and we describe another well known quality function, namely the Conductance. Moreover we interpret the results shown in Section 5 also in terms of Conductance. Section 7 contains a discussion on the computational complexity of our method. Finally, in Section 8 we draw some conclusions and we suggest some future work.

2. Kernel spectral clustering model

Given a graph (weighted or unweighted), several properties of it can be explained through spectral graph theory, which is the study of the eigenspectrum of graph Laplacian matrices [9, 33, 28]. Spectral clustering methods make use of the eigenvectors of the Laplacian to find useful partitions of the data, and they have been reported to be successful in cases where classical clustering schemes such as *k*-means and hierarchical clustering fail.

Given training data $\mathcal{D} = \{x_i\}_{i=1}^N$, $x_i \in \mathbb{R}^d$ and the number of clusters k, the primal problem of spectral clustering via weighted kernel PCA is formulated as follows [1]:

$$\min_{w^{(l)},e^{(l)},b_l} \frac{1}{2}\sum_{l=1}^{k-1} w^{(l)^T} w^{(l)} - \frac{1}{2N}\sum_{l=1}^{k-1} \gamma_l e^{(l)^T} D^{-1} e^{(l)} \tag{1}$$

$$\text{such that } e^{(l)} = \Phi w^{(l)} + b_l 1_N \tag{2}$$

where $e^{(l)} = [e_1^{(l)}, \ldots, e_N^{(l)}]^T$ are the projections, $l = 1, \ldots, k-1$ indicates the score variables needed to encode the k clusters to find, $D^{-1} \in \mathbb{R}^{N\times N}$ is the inverse of the degree matrix D, Φ is the $N \times d_h$ feature matrix $\Phi = [\varphi(x_1)^T; \ldots; \varphi(x_N)^T]$ and $\gamma_l \in \mathbb{R}^+$ are regularization constants. The clustering model is expressed by:

$$e_i^{(l)} = w^{(l)^T} \varphi(x_i) + b_l, i = 1, \ldots, N \tag{3}$$

where $\varphi : \mathbb{R}^d \to \mathbb{R}^{d_h}$ is the mapping to a high-dimensional feature space, b_l are bias terms, $l = 1, \ldots, k-1$. The projections $e_i^{(l)}$ represent the latent variables of a set of $k-1$ binary clustering indicators given by $\text{sign}(e_i^{(l)})$. 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. The dual problem related to this primal formulation is:

$$D^{-1} M_D \Omega \alpha^{(l)} = \lambda_l \alpha^{(l)} \tag{4}$$

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 $M_D = I_N - \frac{1}{1_N^T D^{-1} 1_N} 1_N 1_N^T D^{-1}$, the $\alpha^{(l)}$ are dual variables. The kernel function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$ plays the role of the similarity function of the graph: it has to be positive definite, to take positive values and to be localized (such that the similarity between points belonging to different clusters tends to 0). Finally, the eigenvalue problem (4) is related to spectral clustering with random walk Laplacian L_{RW}. In this case, the clustering problem can be interpreted as finding a partition of the graph in such a way that the random walker remains most of the time in the same cluster with few jumps to other clusters, minimizing the probability of transitions between clusters. The stochastic transition matrix P of a random walker on a graph can be obtained by normalizing the similarity matrix S associated to the graph such that its rows sum to 1. The ij-th entry of P represents the probability of moving from node i to node j in one step of the process. This transition matrix can be defined as $P = D^{-1}S$. The corresponding eigenvalue problem becomes $Pr = \xi r$. The latter is equivalent to problem (4) (exept for the presence of the centering matrix M_D), where the kernel matrix acts as the similarity matrix S.

The out-of-sample extension to new nodes is done by an Error-Correcting Output Codes (ECOC) decoding procedure. The decoding scheme consists of comparing the cluster indicators obtained in the validation/test stage with the codebook and selecting the nearest codeword in terms of Hamming distance. The cluster indicators can be obtained by binarizing the score variables for out-of-sample nodes as follows:

$$\text{sign}(e_{\text{test}}^{(l)}) = \text{sign}(\Omega_{\text{test}} \alpha^{(l)} + b_l 1_{\text{Ntest}}) \tag{5}$$

with $l = 1, \ldots, k-1$. Ω_{test} is the $N_{\text{test}} \times N$ kernel matrix evaluated using the test nodes with entries $\Omega_{\text{test,ri}} = K(x_r^{\text{test}}, x_i)$, $r = 1, \ldots, N_{\text{test}}$, $i = 1, \ldots, N$. This natural extension to out-of-sample points corresponds to the main advantage of the KSC framework. In this way, the clustering model can be trained, validated and tested in an unsupervised learning scheme.

3. Kernel spectral clustering with memory effect

3.1. Introduction

The KSC model summarized in the previous section has been designed to cluster static graphs. In this section we present an extension of such a model that can produce cluster results more consistent over time when applied to changing networks. A dynamic network $\mathcal{G}$ is a sequence of networks $G_t(V_t, E_t)$, i.e. $\mathcal{G} = (G_1, ..., G_t, ...)$, where t indicates the time index. The symbol V_t indicates the set of nodes in the graph G_t and E_t the related set of edges. In what follows we assume that $|V_t|$ is constant, that is all the graphs in the sequence have the same number of nodes (the symbol $|V_t|$ indicates as usual the cardinality of a set). The new model implements a trade-off between the current clustering and the previous partitioning and is referred as kernel spectral clustering with memory effect or MKSC. Finally, for simplicity we assume that the total number of communities does not change over time and we introduce a memory of one snapshot.

3.2. Mathematical formulation

The primal problem of the MKSC model can be stated as follows:

$$\min_{w^{(l)},e^{(l)},b_l} \frac{1}{2}\sum_{l=1}^{k-1} w^{(l)^T} w^{(l)} - \frac{\gamma}{2N}\sum_{l=1}^{k-1} e^{(l)^T} D_{\text{Mem}}^{-1} e^{(l)} - \nu \sum_{l=1}^{k-1} w^{(l)^T} w_{\text{old}}^{(l)} \tag{6}$$

$$\text{such that } e^{(l)} = \Phi w^{(l)} + b_l 1_N \tag{7}$$

The symbols have the same meaning as for the KSC, but here we have only one regularization constant γ instead of multiple γ_l. Moreover $w_{\text{old}}^{(l)}$ represents the l_{th} binary model developed for the previous snapshot of the evolving network (as mentioned we consider a memory of one snapshot in the past), ν is an additional regularization constant and D_{Mem}^{-1} is a weighting matrix whose meaning will be clear soon. Since we consider only one snapshot in the past, we say that our model has a memory of one snapshot. The term $w^{(l)^T} w_{\text{old}}^{(l)}$ describes the correlation between the actual and the previous model, which we want to maximize. In this way we are able to introduce temporal smoothness in our formulation, such that the current partition does not deviate too drammatically from the recent history. The clustering model is expressed, as before, by:

$$e_i^{(l)} = w^{(l)^T} \varphi(x_i) + b_l, i = 1, \ldots, N \tag{8}$$

The Lagrangian of the problem (6), (7) is:

$$\mathcal{L}(w^{(l)}, e^{(l)}, b_l; \alpha^{(l)}) = \frac{1}{2}\sum_{l=1}^{k-1} w^{(l)^T} w^{(l)} - \frac{\gamma}{2N}\sum_{l=1}^{k-1} e^{(l)^T} D_{\text{Mem}}^{-1} e^{(l)} - \nu \sum_{l=1}^{k-1} w^{(l)^T} w_{\text{old}}^{(l)} + \alpha^{(l)^T}(e^{(l)} - \Phi w^{(l)} - b_l 1_N)$$

The KKT optimality conditions are:

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

From $w_{\text{old}}^{(l)} = \Phi_{old}^T \alpha_{\text{old}}^{(l)}$, the bias term becomes $-\frac{1}{1_N^T D_{\text{Mem}}^{-1} 1_N} 1_N^T D_{\text{Mem}}^{-1}(\Omega \alpha^{(l)} + \nu \Omega_{\text{new-old}} \alpha_{\text{old}}^{(l)})$.

Eliminating the primal variables $e^{(l)}$, $w^{(l)}$, b_l leads to the following set of linear systems in the dual problem:

$$(D_{\text{Mem}}^{-1} M_{D_{\text{Mem}}} \Omega - \frac{I}{\gamma})\alpha^{(l)} = -\nu D_{\text{Mem}}^{-1} M_{D_{\text{Mem}}} \Omega_{\text{new-old}} \alpha_{\text{old}}^{(l)}, l = 1, \ldots, k-1 \tag{9}$$

where $\Omega_{\text{new-old}}$ captures the similarity between the nodes of the current graph and the nodes of the previous snapshot and has the $ij-th$ entry $\Omega_{\text{new-old},ij} = K(x_i^{new}, x_j^{old}) = \varphi(x_i^{\text{new}})^T \varphi(x_j^{\text{old}})$. $D_{\text{Mem}}^{-1} \in \mathbb{R}^{N \times N}$ is the inverse of the degree matrix $D_{\text{Mem}} = D + D_{\text{new-old}}$, which is the sum of the actual degree matrix D and the degree matrix with entries $D_{\text{new-old},ii} = \sum_j \Omega_{\text{new-old},ij}$. $M_{D_{\text{Mem}}}$ is the centering matrix equal to $M_{D_{\text{Mem}}} = I_N - \frac{1}{1_N^T D_{\text{Mem}}^{-1} 1_N} 1_N 1_N^T D_{\text{Mem}}^{-1}$. Finally, the cluster indicators for the training points become:

$$\text{sign}(e^{(l)}) = \text{sign}(\Omega\alpha^{(l)} + \nu\Omega_{\text{new-old}}\alpha_{\text{old}}^{(l)} + b1_N). \tag{10}$$

The out-of-sample extension is done in a similar way, but considering the test kernel matrices instead of the training kernel matrices:

$$e^{(l)} = \Omega^{\text{test}}\alpha^{(l)} + \nu\Omega_{\text{new-old}}^{\text{test}}\alpha_{\text{old}}^{(l)} + b_l 1_N. \tag{11}$$

Finally, the MKSC algorithm can be summarized as follows:

Algorithm MKSC Kernel Spectral Clustering with Memory Effect algorithm

---

Input: Training sets $\mathcal{D} = \{x_i\}_{i=1}^N$ and $\mathcal{D}_{\text{old}} = \{x_i^{old}\}_{i=1}^N$, 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 snapshot), 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, regularization constants γ and ν.
Output: Clusters $\{\mathcal{A}_1, \ldots, \mathcal{A}_p\}$, cluster codeset $\mathcal{CB} = \{c_p\}_{p=1}^k$, $c_p \in \{-1, 1\}^{k-1}$.

1. Initialization by solving eq. (4).
2. Compute the solution vectors $\alpha^{(l)}$, $l = 1, \ldots, k-1$, related to the linear systems $(D_{\text{Mem}}^{-1} M_{D_{\text{Mem}}} \Omega - \frac{I}{\gamma})\alpha^{(l)} = -\nu D_{\text{Mem}}^{-1} M_{D_{\text{Mem}}} \Omega_{\text{new-old}} \alpha_{\text{old}}^{(l)}$ (eq. (9)).
3. Binarize the solution vectors: $\text{sign}(\alpha_i^{(l)})$, $i = 1, \ldots, N$, $l = 1, \ldots, k-1$, and let $\text{sign}(\alpha_i) \in \{-1, 1\}^{k-1}$ be the encoding vector for the training data point x_i, $i = 1, \ldots, N$.
4. Count the occurrences of the different encodings and find the k encodings with most occurrences. Let the codeset be formed by these k encodings: $\mathcal{CB} = \{c_p\}_{p=1}^k$, $c_p \in \{-1, 1\}^{k-1}$.
5. $\forall i$, 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.
6. 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}, $m = 1, \ldots, N_{\text{test}}$.
7. $\forall m$, assign x_m^{test} to A_{p^*}, where $p^* = \text{argmin}_p d_H(\text{sign}(e_m), c_p)$.

---

4. Cluster quality

Since clustering is an unsupervised learning procedure, the partitioning produced by an algorithm requires an evaluation regarding its validity [17]. There are two main cluster quality measures:

- internal evaluation: these methods usually assign the best score to the partitions with high similarity within a cluster and low similarity between clusters
- external criteria: they measure the similarity between two partitions. They are useful to compare the results of two different algorithms or to assess the similarity between the partition produced by a clustering method and a golden standard for the evaluation.

4.1. Static measures

In this paper, we will consider as internal measure for the evaluation of our experimental results (see section 5) the Modularity criterion and the BLF criterion and as external measure the adjusted Rand index (ARI).

4.1.1. Modularity

Modularity is the most widely accepted quality function of a graph introduced in [27]. 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, regardless of community structure. Positive high values[1] 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 [16]. Modularity can be written as $mod = X^T M X$, where $M = S - \frac{1}{2m} d d^T$ is the modularity matrix or Q-Laplacian, S indicates the similarity matrix, $d = [d_1, \ldots, d_N]^T$ indicates the vector of the degrees of each node and X represents the cluster indicator matrix.

4.1.2. ARI

The Adjusted Rand Index [19] is used as a performance measure when comparing two clustering results. The ARI equals 1 when the cluster memberships agree completely. Its most common use is as measure of agreement between clustering results of a model and a known grouping which acts like a ground-truth.

4.2. New smoothed measures

Here we introduce new measures in order to assess the quality of a partition related to an evolving network, taking inspiration from [8] (see also Section 4.4). 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 an higher score the more consistent with the past it is (for a particular value of η).

[1] In the literature values above 0.2 are considered already good to indicate a meaningful community structure of a network.

4.2.1. Smoothed modularity

We define the smoothed modularity mod_{Mem} of the partition related to the actual snapshot G_t as:

$$mod_{\text{Mem}}(X_\alpha, G_t) = \eta mod(X_\alpha, G_t) + (1-\eta) mod(X_\alpha, G_{t-1}). \tag{12}$$

It is equivalent to writing:

$$mod_{\text{Mem}}(\alpha, G_t) = \eta X_\alpha^T M_t X_\alpha + (1-\eta) X_\alpha^T M_{t-1} X_\alpha. \tag{13}$$

where X_α indicates the cluster indicator matrix calculated by using the current solution vectors $\alpha^{(l)}$. With η we indicate 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.

4.2.2. Smoothed ARI

The smoothed adjusted Rand index ARI_{Mem} is:

$$ARI_{\text{Mem}}(X_\alpha, G_t) = \eta ARI(X_\alpha, G_t) + (1-\eta) ARI(X_\alpha, G_{t-1}). \tag{14}$$

The symbols have the same meaning as in the previous Section.

4.3. NMI

Given two partitions $\mathcal{U}$ and $\mathcal{V}$, the NMI (Normalized Mutual Information) [30] measures the information that $\mathcal{U}$ and $\mathcal{V}$ share and it 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 viceversa. NMI has been proven to be reliable and is currently used in testing community detection algorithms. We will use the NMI to compare two consecutive partitions over time found by the KSC, MKSC and ESC algorithms, in Section 5: the higher the NMI, the smoother and the more consistent over time can be considered the clustering results.

4.4. Evolutionary spectral clustering

The evolutionary spectral clustering (ESC) was proposed in [8] and aims to optimize the cost function:

$$C_{tot} = \alpha_{\text{ESC}} C_{\text{temp}} + (1-\alpha_{\text{ESC}}) C_{\text{snap}}.$$

C_{snap} describes the classical spectral clustering objective (i.e. ratio cut, normalized cut etc.) related to each snapshot of an evolving graph. C_{temp} can be defined in two ways, in the Preserving Cluster Quality (PCQ) framework or in the Preserving Cluster Membership (PCM) framework. From now on, we will always refer to the former. In PCQ, C_{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.

4.5. Model selection

A proper way of choosing the tuning parameters in a kernel model is critical for determining the success of the model in dealing with a given problem. For KSC some model selection criteria have been proposed, based on the modularity statistic (discussed in [23]), Balanced Line Fit (BLF) [1], or the Fisher criterion (see [2]). In the experiments described later we use the BLF and the modularity criteria to select the parameters of the KSC model. For what concern the

MKSC model, in the same way as we did in the previous subsection, we can define a smoothed version of the BLF criterion BLF_{Mem} to judge the cluster quality over time:

$$BLF_{\text{Mem}}(X_\alpha, G_t) = \eta BLF(X_\alpha, G_t) + (1-\eta)BLF(X_\alpha, G_{t-1}). \quad (15)$$

So, to summarize, for the MKSC model we use a model selection scheme based on the smoothed modularity and BLF criteria, which can be described as follows:

Algorithm MS Model selection algorithm for MKSC

Input: training sets and validation sets (actual and previous snapshots), kernel function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$ positive definite and localized.
Output: selected number of clusters k, kernel parameters (if any), γ, ν.

For every snapshot:

1. Define a grid of values for the parameters to select
2. Train the related kernel machines using the training subgraph
3. Compute the cluster indicator matrix corresponding to the nodes of the validation graph
4. For every partition calculate the related score (by using mod_{Mem} or BLF_{Mem})
5. Choose the model with the highest score.

This scheme is quite general but often we do not need to tune all the parameters, due for example to some prior knowledge we can take advantage of. Moreover it can also be that the same value of some parameters is optimal for all the snapshots, so we do not need to tune for every timestep. Finally some parameters selected for KSC can also be used in the MKSC model.

5. Experimental Results

In this section we show the results of MKSC model on four toy problems and one real dataset. We compare the performance of MKSC with respect to KSC applied separately on each snapshot of the evolving network under investigation, in terms of the new smoothed quality measures introduced in section 4.2 and the NMI between two consecutive partitions as described in 4.3. As expected, in general the outcome suggests that the simple approach based on applying a static clustering model (in our case KSC) at each timestep is less able than a model with temporal smoothness (in our case the MKSC model) to produce a consistent and meaningful partitioning over time. Moreover, we compare MKSC with the ESC algorithm, which is considered among the state-of-the-art methods for community detection on evolving networks. We obtain comparable or better clustering performances, as we show further.

5.1. Data sets

Four synthetic datasets and one real world network have been studied:

- **two moving Gaussians**: the setup for this experiment is shown in Fig.1. We generate 1000 samples from a mixture of two 2-D Gaussians, with 500 samples drawn from each component of the mixture. From timesteps 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 2 Gaussians as nodes of a weighted network (as briefly discussed in section 1) where the weights of the edges change over time.

- **three moving Gaussians**: three merging Gaussian clouds, shown in Fig. 2.
- **switching network**: we build up 9 snapshots of a network of 1000 nodes formed by 2 communities. At each timestep some nodes switch their membership between the two clusters. We used the software related to [15] to generate this benchmark.
- **expanding/contracting network**: a network with 5 communities experiences over-time 24 expansion events and 16 contractions of its communities.
- **cellphone network**: this dataset records the cellphone activity for students and staff from two different labs in MIT [12]. 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.

5.2. The choice of the kernel functions

The choice of the appropriate kernel function used to depict the similarity between the nodes is a critical issue. Here we use the radial basis function (RBF) kernel for the two and three Gaussians datasets and the cellphone network, and the community kernel [20] for the unweighted artificial evolving networks. Indeed, for the weighted networks we change representation by considering each row of the adjacency matrix as a datapoint in an Euclidean space and we apply the RBF kernel as similarity function in the usual way. Even if the dimension of each datapoint can appear large, it is not the case since most of the real graphs are very sparse. The RBF kernel is characterized by the bandwidth parameter σ while the community kernel does not have any parameter to tune. In this case the similarity K_{ij} between two nodes i and j is defined as the number of edges connecting the common neighbors of these two nodes: $K_{ij} = \sum_{k,l \in \mathcal{N}_{ij}} A_{kl}$. Here $\mathcal{N}_{ij}$ is the set of the common neighbors of nodes i and j, A indicates the adjacency matrix of the graph, K is the kernel function. As a consequence, even if two nodes are not directly connected to each other, if they share many common neighbors their similarity k_{ij} will be set to a large value.

5.3. Model selection

For the KSC we use the BLF criterion [1] to tune k and σ in the two and three moving Gaussian problem. The results are shown in Fig. 3 and 4 and refer to the first snapshot (for the other snapshots the plots are similar). For the switching network, in Fig. 5, we illustrate how the modularity-based model selection correctly identifies the possible presence of two communities (in this case k is the only parameter to tune since the community kernel is parameter-free as explained in Section 5.2). Also in the case of the expanding/contracting artificial network our model selection technique detects the correct number of clusters, which is 5 in this case (see Fig. 6). Regarding the cellphone network, we have a partial ground truth, namely the affiliations of each participant. In particular, as observed in [12] and in [31], 2 dominant clusters could be identified from the Bluetooth proximity data, corresponding to new students at the Sloan business school and coworkers who work in the same building. Then for this experiment we will 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 Fig. 7).

For what concerns 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 Fig. 3 and 4. In Fig. 5 the optimal value of γ over time for the the switching network suggested by our model selection scheme is shown. Fig. 6 depicts the optimal γ for the expanding/contracting synthetic network. For the cellphone network $k = 2$ and $\gamma = 1$ are optimal hyper-parameters for each of the 12 weeks, while the values of σ^2 over time are reported in Fig. 7.

5.4. Final results

Here we present the simulation results. For what concerns the models with temporal smoothness (MKSC or ESC) the first partition is found by applying the corresponding static model (KSC or spectral clustering) to the first snapshot since we do not have any information from the past. Then we move along the next snapshots one by one since we consider a memory of one snapshot, as explained in section 3.2. In Fig. 8, 9, 10, 11 and 12 we present the performance of KSC, MKSC and ESC in analyzing the five evolving networks under study, in terms of the smoothed cluster quality measures introduced in section 4.2. In all the experiments we use $\eta = \alpha_{ESC} = 0.5$. Moreover, for what concerns the two and three Gaussian 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:

- two Gaussians 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.
- three Gaussians 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).
- switching network: 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 community structure is quite different from snapshot to snapshot and while MKSC is flexible in adapting to this situation, ESC is not.
- expanding/contracting graph: as expected the models with temporal smoothness (MKSC and ESC) obtain better results than the static KSC model. MKSC produces the best performances.
- cellphone network: MKSC performs better than ESC in some periods and worse in other ones. Both obtain better performances than KSC.

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 Fig. 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, our model (MKSC) does not suffer from this drawback.

6. Flexibility of our framework

6.1. Limitations of Modularity

For the analysis of the networks considered in this paper we used Modularity in order to select the optimal parameters of our model. As it has been pointed out in [14] and [13], 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.

All these limitations, however, do not represent an issue in our framework for several reasons:

- the MKSC model described in equation (6) is quite general (Modularity is not explicitly optimized as in other algorithms like for example [5], [10])
- 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
- many of the abovementioned drawbacks of Modularity have been solved by properly modifying its definition, as for example in [3] where a multi-resolution Modularity has been introduced. In this case, however, as pointed out in [22], 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.

6.2. Conductance

In order to better understand these issues we present a further analysis based on another quality function called Conductance. For every community, the Conductance is defined as the ratio between the number of edges leaving the cluster and the number of edges inside the cluster [21], [24]. In particular, Conductance $C(S)$ of a set of nodes $S \subset V$ is $C(S) = c_S / \min(Vol(S), Vol(V \setminus S))$, 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; thus the lower the Conductance the better the community structure. Moreover, in the same way as for Modularity in section 4.2, we can define the smoothed Conductance $Cond_{\text{Mem}}$ of the partition related to the actual snapshot G_t as:

$$Cond_{\text{Mem}}(X_\alpha, G_t) = \eta Cond(X_\alpha, G_t) + (1 - \eta) Cond(X_\alpha, G_{t-1}). \tag{16}$$

In Table 1 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) [5]. 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.

7. Computational complexity

In this section a brief discussion about the computational complexity of the MKSC method is given. In fact nowadays many large data-sets are available and the algorithms for community detection are required to scale well. In order to test the runtime of our method we generated some switching network (see section 5.1) of increasing size, namely from 10^3 to 10^5 nodes. In Fig. 13 we show the results. We can see that the MKSC model runs faster than ESC. Moreover the latter cannot be applied to networks with more than 10.000 nodes because of memory problems. However, the complexity of MKSC appears to be exponential and then needs to be improved. By performing the profiling of our code, we can notice that about 99% of the time is spent to calculate the train (8%) and test (91%) kernel matrices. We believe that an optimized C++ implementation of the code (the actual one is done in Matlab) could allow our method to achieve a much faster runtime. In fact also the Louvain method runtime, in its Matlab implementation, is higher than MKSC. On the other hand it is well known that the C++ implementation of the Louvain method allows the latter to scale linearly with the number of nodes of the entire graph and then to be considered among the fastest state-of-the-art algorithms. Moreover the calculation of the kernel matrices is easily parallelizable.

Finally, for very large networks ($\geq 10^6$ nodes) it can happen that the solution of the linear system (9) requires a considerable amount of time. In this case a possible way to overcome this issue is to use the Woodbury matrix identity, also known as matrix inversion lemma [18]. Suppose that we have to perform community detection of a network formed by 10^6 nodes. We can select for example a training set consisting of $N_{tr} = 50000$ nodes. In order to find the solution vector α we have to solve the dual problem (9). This implies to find the inverse of a $N_{tr} \times N_{tr}$ matrix. Now, instead of calculating this inverse, we could select for instance 1000 nodes from the training set, solve the small linear system of dimension 1000×1000, and then calculate iteratively the $N_{tr} \times 1$ solution vector α by using the Woodbury formula.

8. Conclusions and perspectives

In this paper we faced the problem of community detection on evolving networks in the case that we can include prior knowledge of temporal smoothness. This is an emerging research topic and represents a challenge for actual clustering algorithms. In fact, a desirable property is that the latter should be able to catch the long-term drifts characterizing the evolution of the communities and neglect short-term variations due to noise. As mentioned, this can be thought as a kind of temporal smoothness, in the same sense as in time-series analysis. We proposed a new model, the kernel spectral clustering with memory effect or MKSC, casted in the LS-SVM [32] framework. We explicitly designed our clustering model in order to incorporate temporal smoothness. The clustering is performed by solving a set of linear systems for every snapshot of

the evolving graph and the model is able to predict the memberships of new nodes by means of the out-of-sample extension. We also introduced new cluster quality measures and a related model selection scheme that are more suitable to this kind of problem. We tested our method on four benchmark datasets and on a real graph representing the proximity of MIT students and faculty members during one academic year as recorded by Bluetooth devices. The obtained results are successful and comparable or better than those obtained by using kernel spectral clustering and evolutionary spectral clustering. In future work we aim to extend the MKSC model in order to deal with a changing number of cluster and nodes over time and a longer memory (more than one snapshot in the past). In this way, given a dynamic network, one will be able to obtain a general algorithm in order to discover a meaningful community evolution by catching events like merging, splitting, birth, death etc. Moreover, since scalability is an important issue, we aim to implement a faster version of our algorithm which can be used in an effective way to analyze large evolving networks.

Acknowledgements

This work was supported by Research Council KUL: ERC AdG A-DATADRIVE-B, GOA/11/05 Ambiorics, GOA/10/09 MaNet , CoE EF/05/006 Optimization in Engineering(OPTEC), IOF-SCORES4CHEM, several PhD/postdoc & fellow grants;Flemish Government:FWO: PhD/postdoc grants, projects: G0226.06 (cooperative systems and optimization), G0321.06 (Tensors), G.0302.07 (SVM/Kernel), G.0320.08 (convex MPC), G.0558.08 (Robust MHE), G.0557.08 (Glycemia2), G.0588.09 (Brain-machine) G.0377.12 (structured models) research communities (WOG:ICCoS, ANMMM, MLDM); G.0377.09 (Mechatronics MPC) IWT: PhD Grants, Eureka-Flite+, SBO LeCoPro, SBO Climaqs, SBO POM, O&O-Dsquare; Belgian Federal Science Policy Office: IUAP P6/04 (DYSCO, Dynamical systems, control and optimization, 2007-2011); EU: ERNSI; FP7-HD-MPC (INFSO-ICT-223854), COST intelliCIS, FP7-EMBOCON (ICT-248940); Contract Research: AMINAL; Other:Helmholtz: viCERP, ACCM, Bauknecht, Hoerbiger. Carlos Alzate is a postdoctoral fellow of the Research Foundation - Flanders (FWO). Johan Suykens is a professor at the KU Leuven, Belgium. The scientific responsibility is assumed by its authors.

References

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

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

[3] Arenas, A., Fernndez, A., Gmez, S., 2008. Analysis of the structure of complex networks at different resolution levels. New Journal of Physics 10 (5), 053039.

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

[5] Blondel, V. D., Guillaume, J.-L., Lambiotte, R., Lefebvre, E., 2008. Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment 2008 (10), P10008.

[6] Bóta, A., Krész, M., Pluhár, A., 2011. Dynamic communities and their detection. Acta Cybernetica 20 (1), 35–52.

[7] Chakrabarti, D., Kumar, R., Tomkins, A., 2006. Evolutionary clustering. In: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining. KDD '06. ACM, New York, NY, USA, pp. 554–560.

[8] Chi, Y., Song, X., Zhou, D., Hino, K., Tseng, B. L., 2007. Evolutionary spectral clustering by incorporating temporal smoothness. In: KDD. pp. 153–162.

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

[10] Clauset, A., Newman, M. E. J., , Moore, C., 2004. Finding community structure in very large networks. Physical Review E, 1– 6.

[11] Dhillon, I., Guan, Y., Kulis, B., 2007. Weighted graph cuts without eigenvectors a multilevel approach. Pattern Analysis and Machine Intelligence, IEEE Transactions on 29 (11), 1944 –1957.

[12] Eagle, N., Pentland, A. S., Lazer, D., 2009. Inferring social network structure using mobile phone data. PNAS 106 (1), 15274–15278.
[13] Fortunato, S., Barthlemy, M., 2007. Resolution limit in community detection. Proceedings of the National Academy of Sciences 104 (1), 36–41.
[14] Good, B., Montjoye, Y. D., Clauset, A., 2010. Performance of modularity maximization in practical contexts. Physical Review E 81 (4), 046106.
[15] Greene, D., Doyle, D., Cunningham, P., 2010. 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. IEEE Computer Society, Washington, DC, USA, pp. 176–183.
[16] Guimera, R., Sales-Pardo, M., Amaral, L., 2004. Modularity from fluctuations in random graphs and complex networks. Physical Review E 70 (2), 025101.
[17] Halkidi, M., Batistakis, Y., Vazirgiannis, M., 2001. On clustering validation techniques. Journal of Intelligent Information Systems 17, 107–145.
[18] Higham, N. J., 1996. Accuracy and Stability of Numerical Algorithms. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.
[19] Hubert, L., Arabie, P., 1985. Comparing partitions. Journal of Classification 1 (2), 193–218.
[20] Kang, Y., Choi, S., 2009. Kernel PCA for community detection. In: Business Intelligence Conference.
[21] Kannan, R., Vempala, S., Vetta, A., 2000. On clusterings: Good, bad and spectral.
[22] Lancichinetti, A., Fortunato, S., Dec 2011. Limits of modularity maximization in community detection. Phys. Rev. E 84, 066122.
[23] Langone, R., Alzate, C., Suykens, J. A. K., 2011. Modularity-based model selection for kernel spectral clustering. In: Proc. of the International Joint Conference on Neural Networks (IJCNN 2011). pp. 1849–1856.
[24] Leskovec, J., Lang, K. J., Mahoney, M., 2010. Empirical comparison of algorithms for network community detection. In: Proceedings of the 19th international conference on World wide web. WWW '10. ACM, New York, NY, USA, pp. 631–640.
[25] Lin, Y.-R., Chi, Y., Zhu, S., Sundaram, H., Tseng, B. L., 2009. Analyzing communities and their evolutions in dynamic social networks. ACM Trans. Knowl. Discov. Data 3 (2).
[26] Mucha, P. J., Richardson, T., Macon, K., Porter, M. A., Onnela, J.-P., 2010. Community structure in time-dependent, multiscale, and multiplex networks. Science 328 (5980), 876–878.
[27] Newman, M. E. J., 2006. Modularity and community structure in networks. Proc. Natl. Acad. Sci. USA 103 (23), 8577–8582.
[28] Ng, A. Y., Jordan, M. I., Weiss, Y., 2002. On spectral clustering: Analysis and an algorithm. In: Dietterich, T. G., Becker, S., Ghahramani, Z. (Eds.), Advances in Neural Information Processing Systems 14. MIT Press, Cambridge, MA, pp. 849–856.
[29] Palla, G., lszl Barabsi, A., Vicsek, T., Hungary, B., 2007. Quantifying social group evolution. Nature 446, 2007.
[30] Strehl, A., Ghosh, J., 2002. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. Journal of Machine Learning Research 3, 583–617.
[31] Sun, J., Faloutsos, C., Papadimitriou, S., Yu, P. S., 2007. 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. ACM, New York, NY, USA, pp. 687–696.
[32] Suykens, J. A. K., Van Gestel, T., De Brabanter, J., De Moor, B., Vandewalle, J., 2002. Least Squares Support Vector Machines. World Scientific, Singapore.
[33] von Luxburg, U., 2007. A tutorial on spectral clustering. Statistics and Computing 17 (4), 395–416.

Switching network	Smoothed Conductance
MKSC	**0.0020**
ESC	0.0022
KSC	0.0022
LOUV	0.0024

Expanding network	Smoothed Conductance
MKSC	**0.0050**
ESC	0.0051
KSC	0.0051
LOUV	0.0184

Cellphone network	Smoothed Conductance
MKSC	**0.0019**
ESC	0.0042
KSC	0.0056
LOUV	0.0153

Table 1: Average smoothed Conductance over time for the **switching network** (top), the **expanding/contracting network** (middle) and the **cellphone network**.

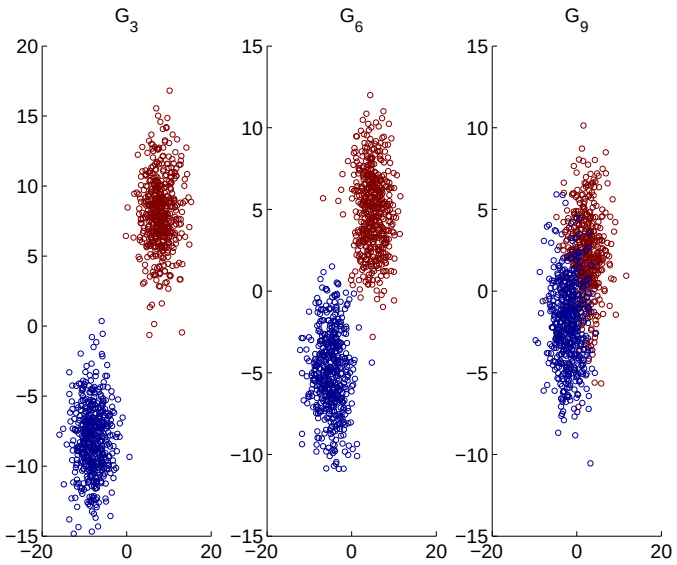


Figure 1: **Two moving Gaussians** dataset. Only the snapshots 3, 6 and 9 are shown.

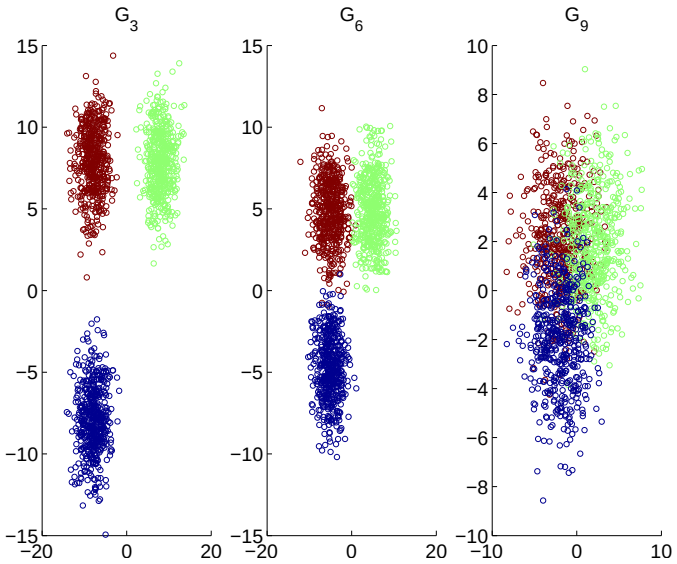


Figure 2: **Three moving Gaussians** dataset. Snapshots 3, 6 and 9 are depicted.

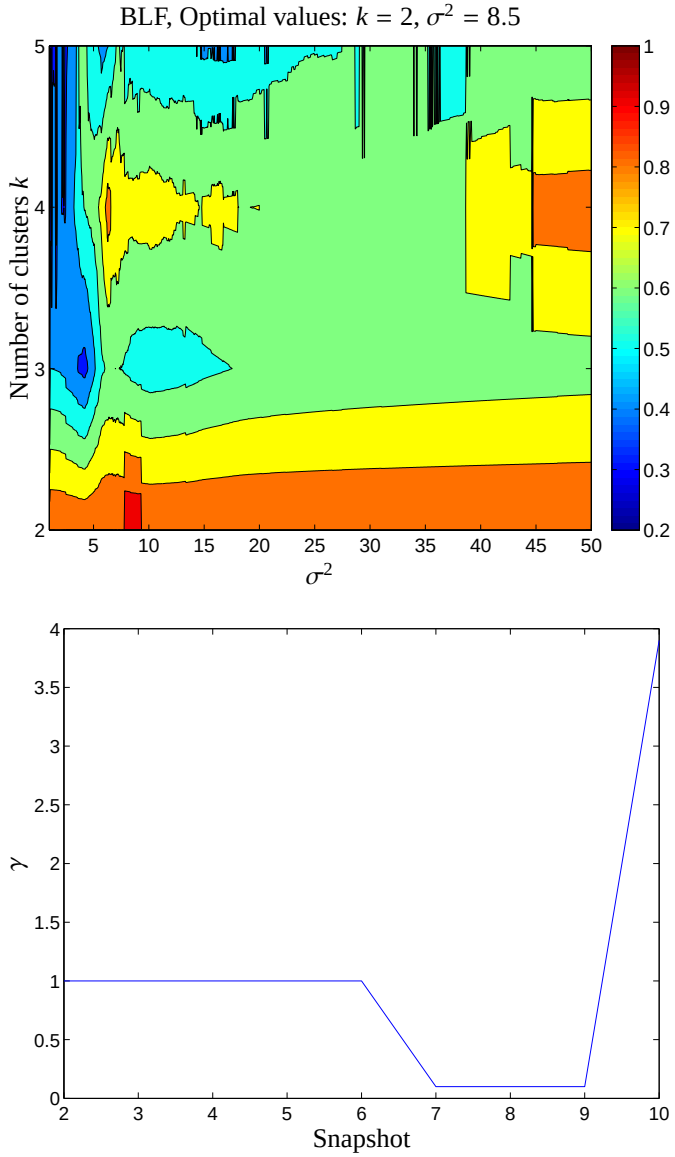


Figure 3: **Model selection two moving Gaussians**. Top: tuning of the number of cluster k and the RBF kernel parameter σ^2 related to the first snapshot of the two moving Gaussians experiment, for KSC. The optimal σ^2 does not change over time (the model selection procedure gives similar results also for the other snapshots). Moreover, this value will also be used for the MKSC model. Bottom: optimal value of γ over time for MKSC, tuned using the BLF_{Mem} method.

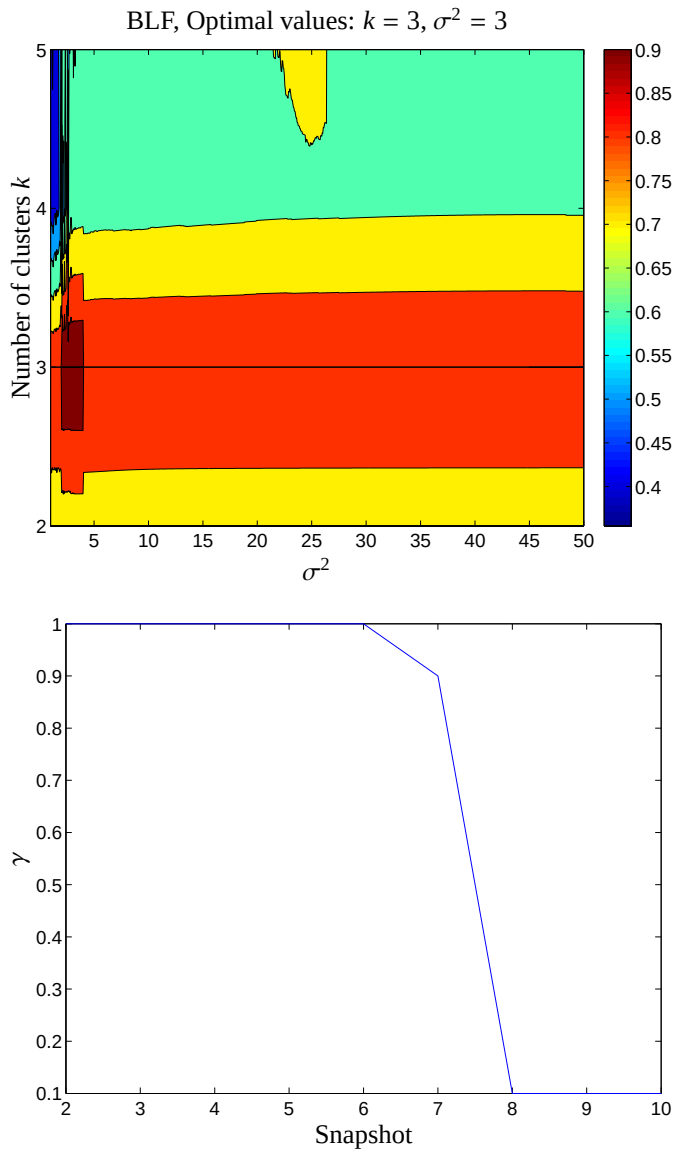


Figure 4: **Model selection three moving Gaussians**. Top: tuning of the RBF kernel parameter σ^2 and the number of clusters k related to the first snapshot of the three moving Gaussians experiment, for KSC. Bottom: tuning of γ for MKSC. The same comments made for Fig. 3 are still valid here.

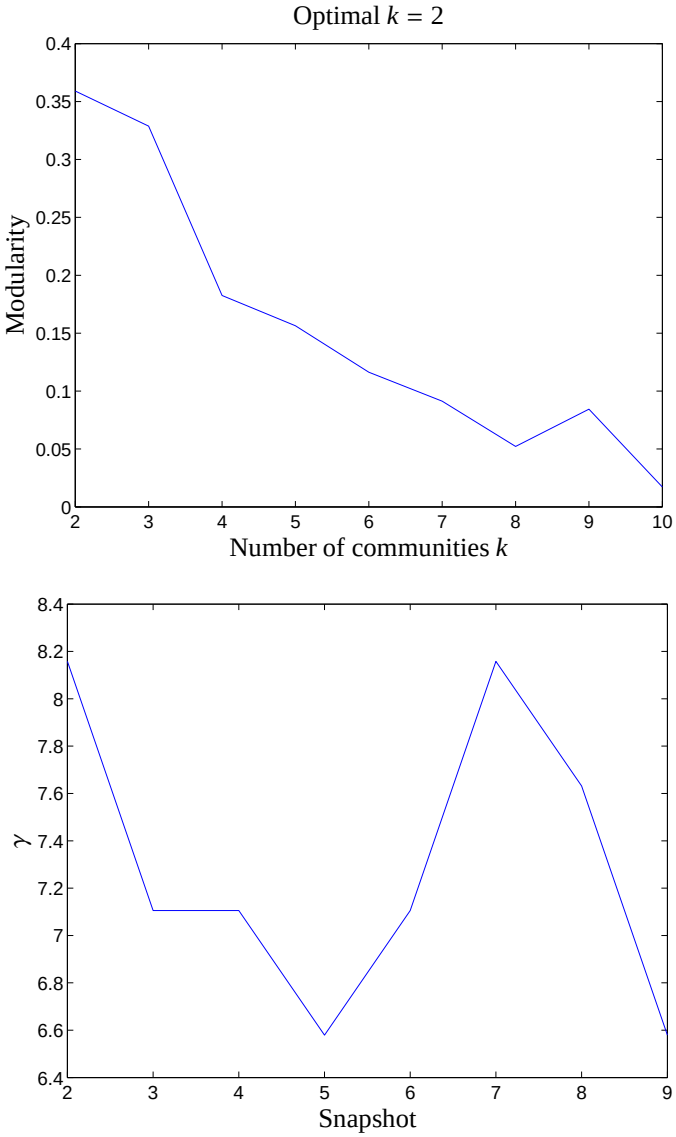


Figure 5: **Model selection switching network**. Top: tuning of the number of clusters k for the switching network, related to the first snapshot and the KSC model. The results are similar for the other snapshots and this k will be used also for the MKSC algorithm. Bottom: optimal value of γ over time for the MKSC model, selected by using the Mod_{Mem} criterion.

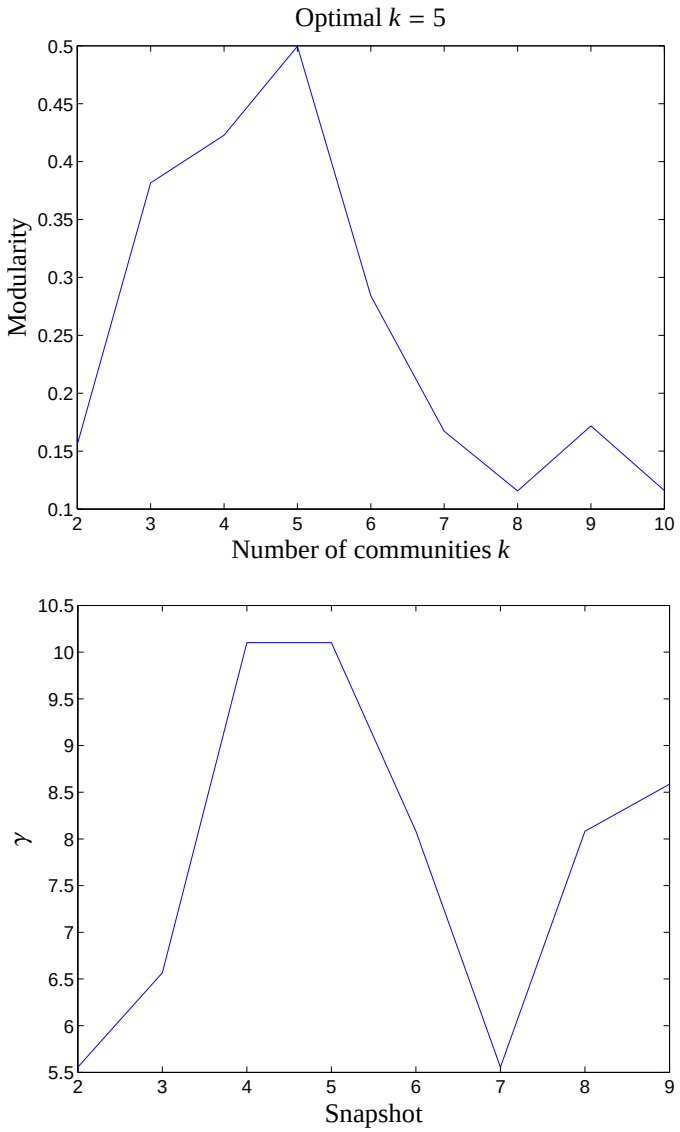


Figure 6: **Model selection expanding/contracting network**. Top: tuning of the number of clusters k for the expanding/contracting network, related to the first snapshot and the KSC algorithm. Bottom: tuning of γ for MKSC. The comments made for Fig. 5 hold also in this case.

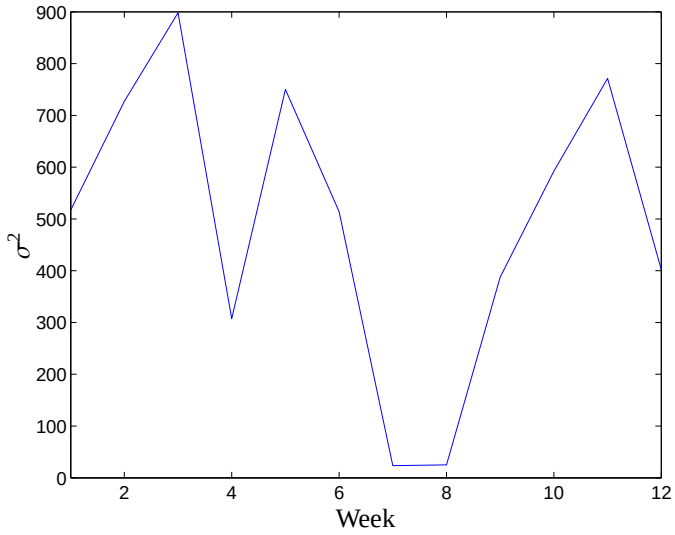


Figure 7: **Model selection cellphone network**. Optimal σ^2 over time for the cellphone network, related to MKSC. The number of clusters is $k = 2$, $\gamma = 1$ is an optimal value for all the snapshots.

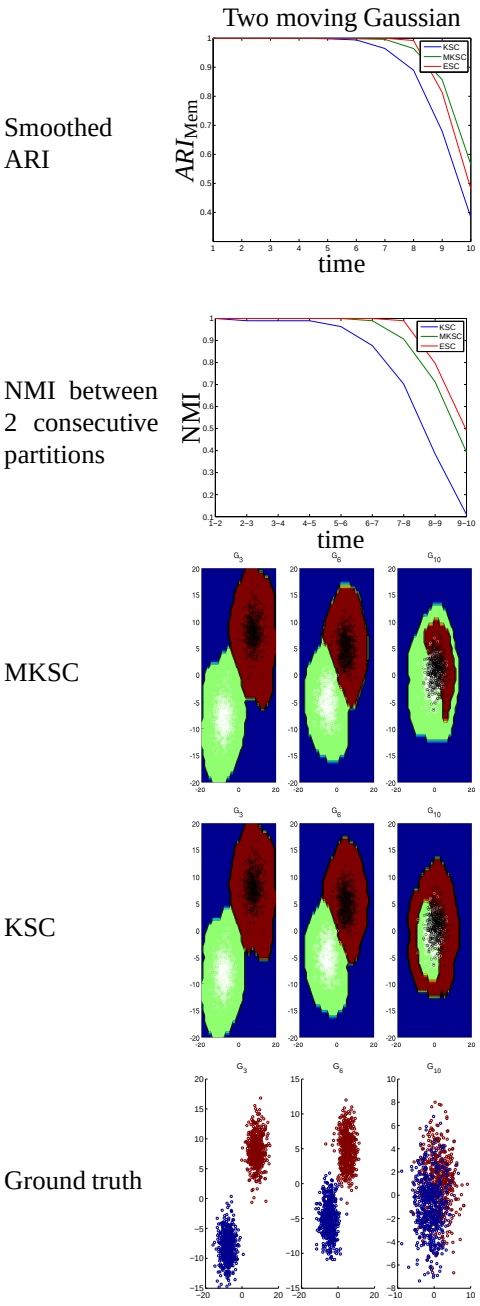


Figure 8: **Clustering results two moving Gaussians**: performance of MKSC, KSC and ESC in the two moving Gaussians experiment (first row) and out-of-sample plot for MKSC and KSC (only the results related to snapshots 3, 6 and 10 are shown). The true partitioning is depicted in the fifth row. The smoothed 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 (in the sense explained in Sections 4.2 and 4.3). However, in the out-of-sample plot we cannot visually appreciate the better performance of MKSC with respect to KSC.

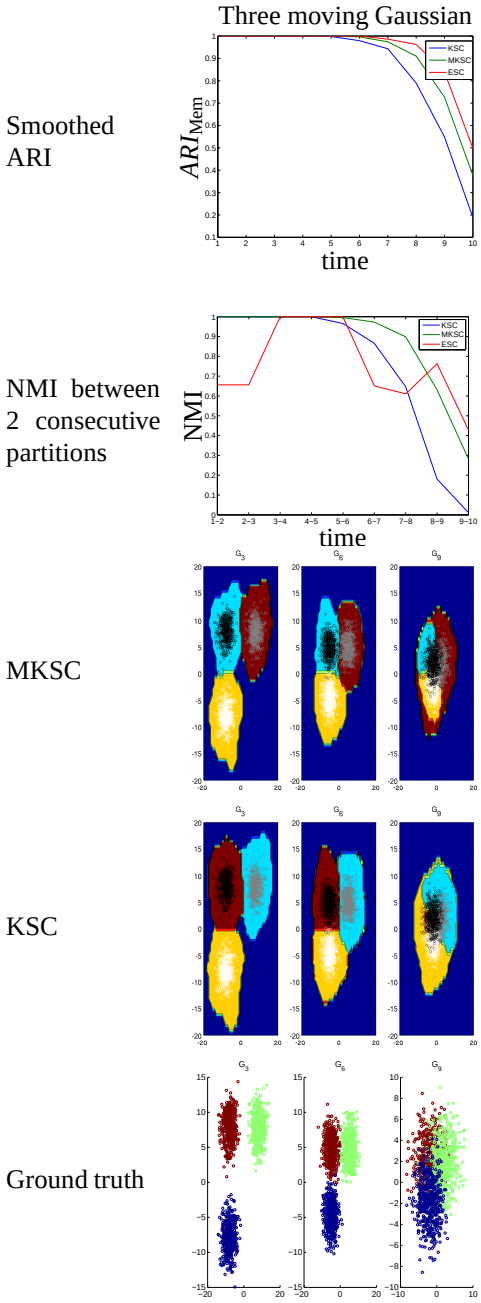


Figure 9: **Clustering results three moving Gaussians**: Performance of MKSC, KSC and ESC in the three moving Gaussians experiment in terms of smoothed ARI and NMI between two consecutive partitions, and out-of-sample plot for MKSC and KSC (only the results related to snapshots 3, 6 and 9 are shown). The true partitioning is depicted in the last row. The same observations made for Fig 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, as mentioned in Section 5.4.

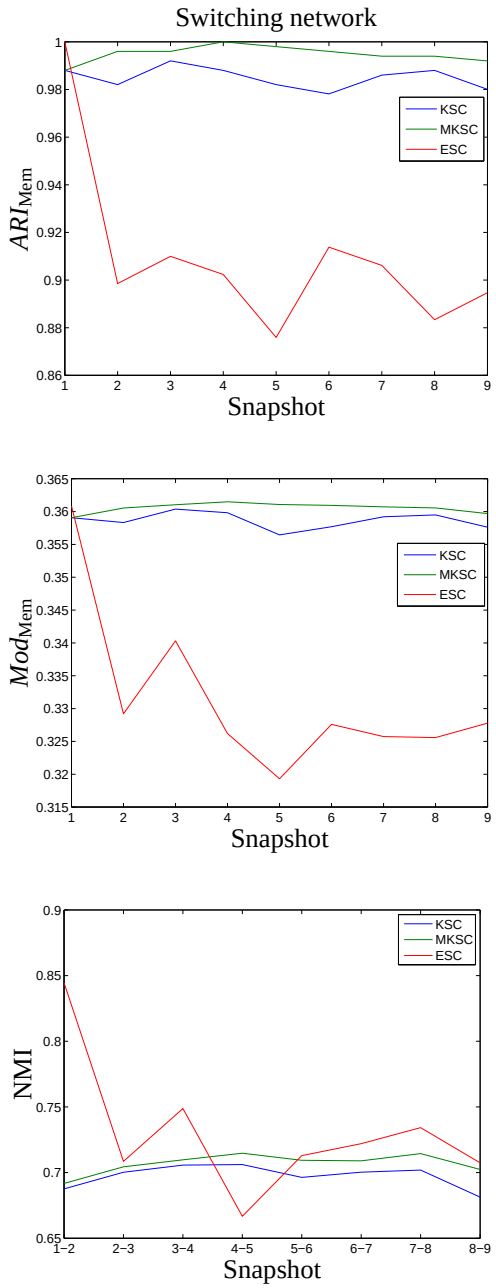


Figure 10: **Clustering results switching network**: performance of MKSC, KSC and ESC on the artificial evolving network with 2 communities in terms of the new smoothed cluster measures explained in Section 4.2. Here, surprisingly, KSC produce better results than the ESC model according to the smoothed ARI and modularity. However MKSC performs better than KSC.

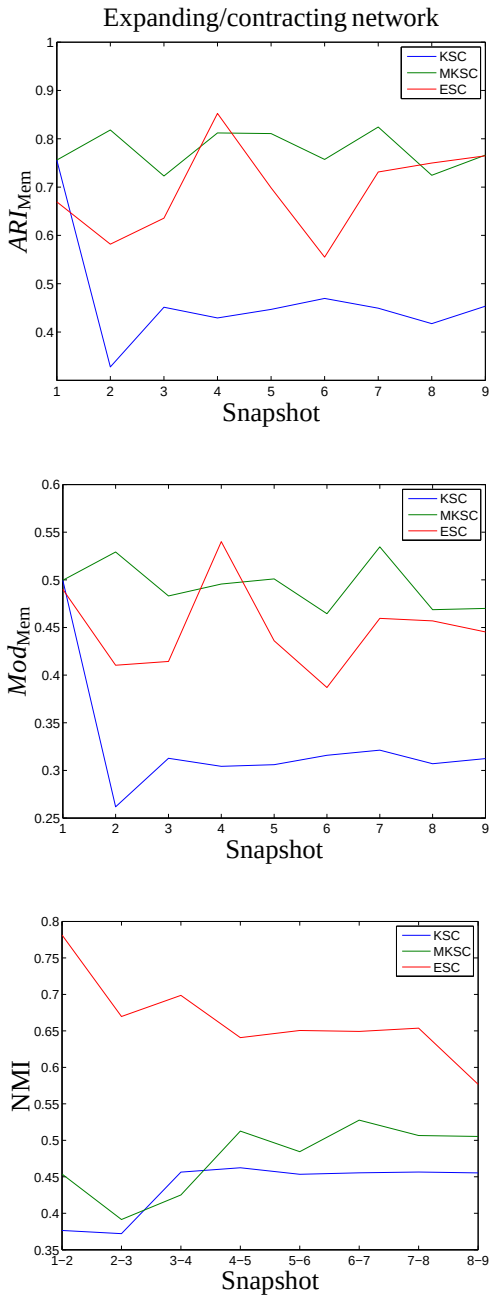


Figure 11: **Clustering results expanding/contracting network**: performance of MKSC, KSC and ESC on the artificial evolving network with 5 communities. The models with temporal smoothness produce partitions of higher quality than KSC (according to the smoothed measures introduced in Section 4.2 and the NMI discussed in Section 4.3), 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.

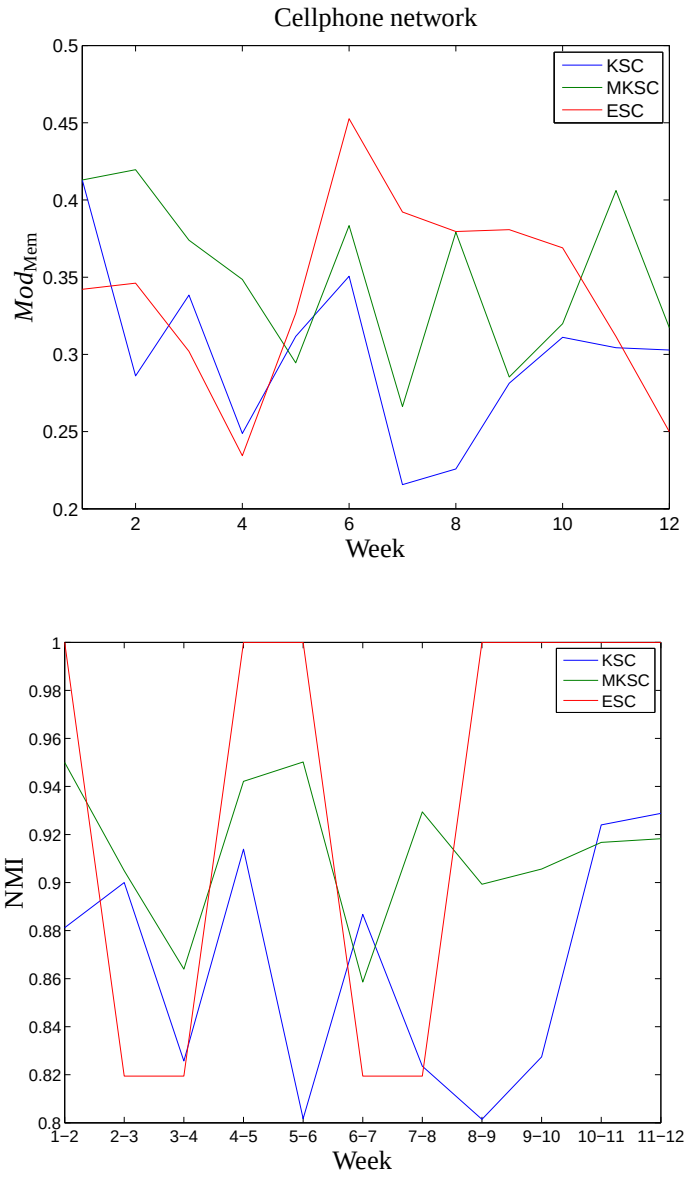


Figure 12: **Clustering results cellphone network**: performance of MKSC, KSC and ESC on the cellphone network in terms of the smoothed modularity and the NMI between consecutive clustering results. Also in this case the models with temporal smoothness MKSC and ESC, in the most of the time period, perform better than the static KSC.

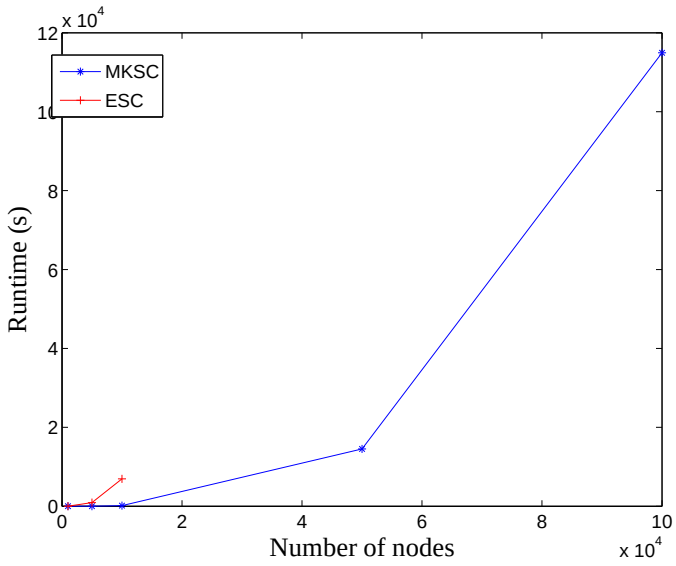


Figure 13: Evolution of the speed with the size of the benchmark network for ESC (until 10^4 because of memory problems) and MKSC. For MKSC the training set size is the 10% of the size of the whole network.