KSC-net: Community Detection for Big Data Networks

Raghvendra Mall[1] and Johan A.K. Suykens[2]

KU Leuven - ESAT/STADIUS

Kasteelpark Arenberg 10, bus 2446, B-3001 Leuven - Belgium

Abstract

In this chapter we demonstrate the applicability of the Kernel Spectral Clustering (KSC) method for community detection in big data networks. We give a practical exposition of the KSC method [1] on large scale synthetic and real world networks with up to 10^6 nodes and 10^7 edges. The KSC method uses a primal-dual framework to construct a model on a smaller subset of the big data network. The original large scale kernel matrix cannot fit in memory. So we select smaller subgraphs using a fast and unique representative subset (FURS) selection technique as proposed in [2]. These subsets are used for training and validation respectively to build the model and obtain the model parameters. It results in a powerful out-of-sample extensions property which allows inferring the community affiliation for unseen nodes.

The KSC model requires a kernel function which can have kernel parameters and the needs to identify the number of clusters k in the network. A memory and computationally efficient model selection technique named balanced angular fitting (BAF) based on angular similarity in the eigenspace was proposed in [1]. Another parameter-free KSC model was proposed in [3]. In [3], the model selection technique exploits the structure of projections in eigenspace to automatically identify the number of clusters and suggests that normalized linear kernel is sufficient for networks with millions of nodes. This model selection technique uses the concept of entropy and balanced clusters

for identifying the number of clusters k.

We then describe our software called KSC-net which obtains the representative subset by FURS, builds the KSC model, performs one of the two (BAF and parameter-free) model selection techniques and uses out-of-sample extensions for community affiliation for the big data network.

1 Introduction

In the modern era with the proliferation of data and ease of its availability with the advancement of technology, the concept of big data has emerged. Big data refers to massive amount of information which can be collected by means of cheap sensors and wide usage of Internet. In this chapter we focus on big data networks which are ubiquitous in current life. Their omnipresence is confirmed by the modern massive online social networks, communication networks, collaboration networks, financial networks etc.

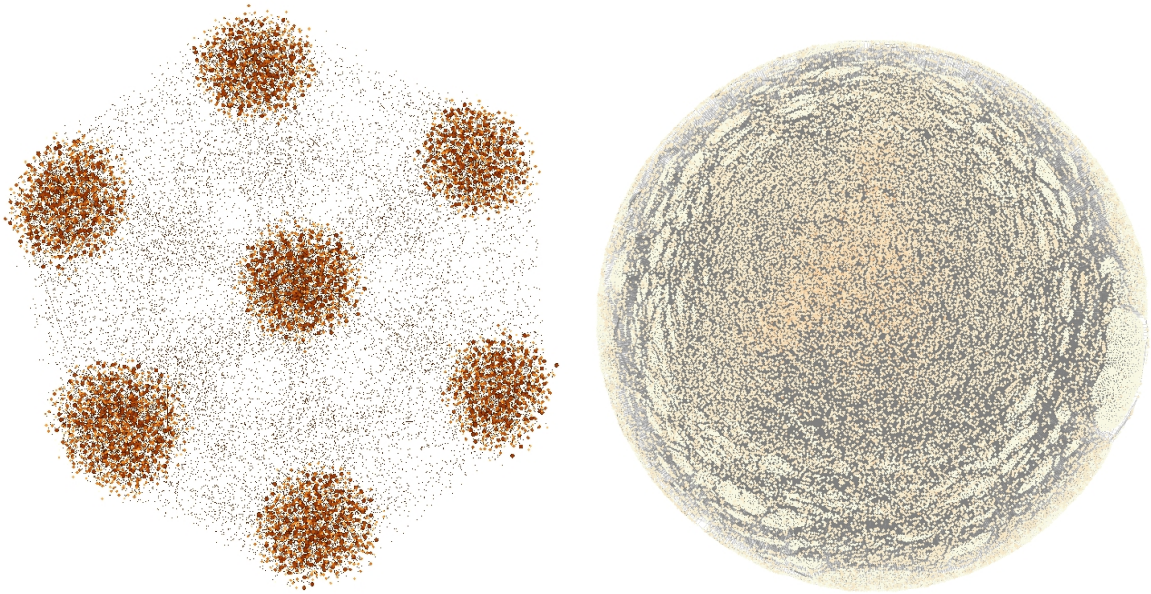


Figure 1: A Synthetic network and a real world Youtube network

Figure 1 represents a synthetic network with $5,000$ nodes and a real world Youtube network with over a million nodes. The networks are visualized using the gephi software (`https://gephi.org/`).

Real world complex networks are represented as graphs $G(V, E)$ where the vertices V represent the entities and the edges E represent the relationship between these entities.

For example in a scientific collaboration network the entities are the researchers and the presence or absence of an edge between two researchers depicts whether they have collaborated or not in the given network. Real life networks exhibit community like structure. This means nodes which are part of one community are densely connected to each other and sparsely connected to nodes belonging to other communities. This problem of community detection can also be framed as graph partitioning and graph clustering [4] and off-late has received wide attention [5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]. Among these techniques, one class of methods used for community detection is referred as spectral clustering [13, 14, 15, 16].

In spectral clustering an eigen-decomposition of the graph Laplacian matrix (L) is performed. L is derived from the similarity or affinity matrix the nodes in the network. Once the eigenvectors are obtained, the communities can be detected using k-means clustering technique. The major disadvantage of spectral clustering technique is that we need to create the large $N \times N$ kernel matrix for the entire network to perform the unsupervised community detection. Here N represents the number of nodes in the large scale network. However, as the size of network increases, calculating and storing $N \times N$ matrix becomes computationally in-feasible.

A spectral clustering method based on weighted kernel PCA with primal-dual framework was proposed in [17]. The formulation resulted in a model built on a representative subset of the data with a powerful out-of-sample extensions property. This representative subset captures the inherent cluster structure present in the dataset. This property allows community affiliation for previously unseen data points in the dataset. The KSC method was extended for community detection on moderate size graphs in [11] and for large scale graphs in [1].

We use the fast and unique representative subset (FURS) selection technique introduced in [2] for selection of the subgraphs on which we build our training and validation models respectively. An important step in the KSC method is to estimate the model parameters i.e. kernel parameter, if any and the number of communities k in the large scale network. In case of large scale networks the normalized linear kernel is an effective kernel as shown in [18]. Thus, we use the normalized linear kernel which is parameter-less in our experiments. This saves us from tuning for an additional kernel parameter. In order to estimate the number of communities k, we exploit the eigen-projections in eigenspace

to come up with a Balanced Angular Fitting (BAF) criterion [1] and another self-tuned approach [3] where we used the concept of entropy and balance to automatically get k.

In this chapter we give a practical exposition of the KSC method by first briefly explaining the steps involved and then showcasing its usage with the KSC-net software. We demonstrate the options available during the usage of KSC-net by means of two demos and explain the internal structure and functionality in terms of the KSC methodology.

2 Kernel Spectral Clustering for Big Data Networks

We first provide a brief description of the KSC methodology along with the FURS selection technique. We also explain the BAF and self-tuned model selection techniques. The notations used in this chapter are explained below:

2.1 Notations

1. A graph is represented as $G = (V, E)$ where V represents the set of nodes and $E \subseteq V \times V$ represents the set of edges in the network. The nodes represent the entities in the network and the edges represent the relationship between them.

2. The cardinality of the set V is given as N.

3. The cardinality of the set E is given as E_{total}.

4. The adjacency matrix A is a $N \times N$ sparse matrix.

5. Affinity/Kernel matrix of nodes is given by Ω and affinity matrix of projections is depicted by S.

6. For unweighted graphs, $A_{ij} = 1$ if $(v_i, v_j) \in E$ else $A_{ij} = 0$.

7. The subgraph generated by the subset of nodes B is represented as $G(B)$. Mathematically, $G(B) = (B, Q)$ where $B \subset V$ and $Q = (S \times S) \cap E$ represents the set of edges in the subgraph.

8. The degree distribution function is given by $f(V)$. For the graph G it can written as $f(V)$ while for the subgraph B it can be presented as $f(B)$. Each vertex $v_i \in V$ has a degree represented as $f(v_i)$.

9. The degree matrix is represented as D. It is a diagonal matrix with diagonal entries $d_{i,i} = \sum_j A_{ij}$.

10. The adjacency list corresponding to each vertex $v_i \in V$ is given by $x_i = A(:, i)$.

11. The set of neighboring nodes of a given node v_i is represented by $Nbr(v_i)$.

12. The median degree of the graph is represented as M.

The KSC methodology comprises of three major steps:

1. Building a model on the training data.

2. Identifying the model parameters using the validation data.

3. The out-of-sample extension to determine community affiliation for the unseen test set of nodes.

2.2 Fast and Unique Representative Subset (FURS) Selection

Nodes which have higher degree centrality or more connections represent the presence of more influence and have the tendency to be located in the center of the network. The goal of the FURS selection technique is to select several such nodes of high degree centrality from different dense regions in the graph to capture the inherent community structure present in the network. However, this problem of selection of such a subset (B) is NP-hard and is formulated as:

$$\begin{aligned} \max_{B} \quad & J(B) = \sum_{j=1}^{s} D(v_j) \\ \text{s.t.} \quad & v_j \in c_i, \\ & c_i \in \{c_1, \ldots, c_k\} \end{aligned} \tag{1}$$

where $D(v_j)$ represents the degree centrality of the node v_j, s is the size of the subset, c_i represents the i^{th} community and k represents the number of communities in the network which cannot be determined explicitly beforehand.

The FURS selection technique is a greedy solution to the aforementioned problem. It is formulated as an optimization problem where we maximize the sum of the degree centrality of the nodes in the selected subset B, such that neighbors of the selected node

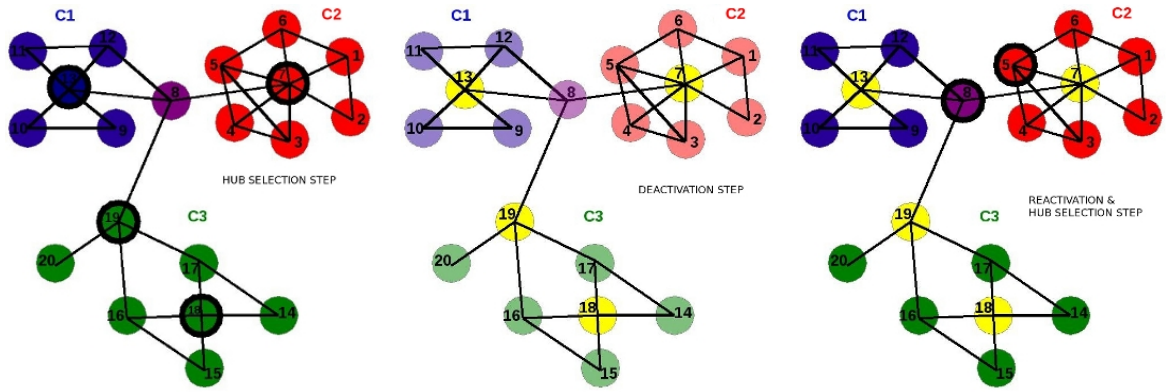


Figure 2: Steps of FURS for a sample of 20 nodes.

are deactivated or cannot be selected in that iteration. By deactivating its neighborhood we move from one dense region in the graph to another, thereby approximately covering all the communities in the network. If all the nodes are deactivated in one iteration and we have not yet reached the required subset size s, then these deactivated nodes are reactivated and the procedure is repeated till we reach the required size s. The optimization problem is formulated below:

$$
\begin{aligned}
& J(B) = 0 \\
& \text{While } |B| < \text{s} \\
& \max_{B} \qquad J(B) := J(B) + \sum_{j=1}^{s^t} D(v_j) \qquad (2) \\
& \text{s.t.} \qquad Nbr(v_j) \rightarrow \text{deactivated, iteration t,} \\
& \qquad\quad Nbr(v_j) \rightarrow \text{activated, iteration t+1,}
\end{aligned}
$$

where s^t is the size of the set of nodes selected by FURS during iteration t.

Several approaches have been proposed for sampling a graph including [2, 19, 20, 21]. The Fast and Unique Representative Subset (FURS) selection approach was proposed in [2] and used in [1]. A comprehensive comparison of various sampling techniques has been explained in detail in [2]. We use the FURS selection technique in KSC-net software for training and validation set selection.

2.3 KSC Framework

For large scale networks, the training data comprises the adjacency list of all the nodes $v_i, i = 1, \ldots, N_{tr}$. Let the training set of nodes be represented by V_{tr} and the training

set cardinality be N_{tr}. The validation and test set of nodes are represented by V_{valid} and V_{test} respectively. The cardinality of these sets are N_{valid} and N_{test} respectively. These sets of adjacency lists can efficiently be stored in the memory as real world networks are highly sparse and there are limited connections for each node $v_i \in V_{tr}$. The maximum length of the adjacency list can be equal to N. This is the case when a node is connected to all the other nodes in a network.

2.3.1 Training Model

For V_{tr} training nodes selected by the FURS selection technique, $\mathcal{D} = \{x_i\}_{i=1}^{N_{tr}}$, such that $x_i \in \mathbb{R}^N$. Here x_i represents the adjacency list of the i^{th} training node. Given $\mathcal{D}$ and a user-defined $maxk$ (maximum number of clusters in the network), the primal formulation of the weighted kernel PCA [17] is given by:

$$\begin{aligned} \min_{w^{(l)},e^{(l)},b_l} \quad & \frac{1}{2}\sum_{l=1}^{maxk-1} w^{(l)\intercal}w^{(l)} - \frac{1}{2N_{tr}}\sum_{l=1}^{maxk-1}\gamma_l e^{(l)\intercal}D_\Omega^{-1}e^{(l)} \\ \text{such that} \quad & e^{(l)} = \Phi w^{(l)} + b_l 1_{N_{tr}}, l = 1,\ldots,maxk-1 \end{aligned} \tag{3}$$

where $e^{(l)} = [e_1^{(l)},\ldots,e_{N_{tr}}^{(l)}]^\intercal$ are the projections onto the eigenspace, $l = 1,\ldots,maxk-1$ indicates the number of score variables required to encode the $maxk$ communities, $D_\Omega^{-1} \in \mathbb{R}^{N_{tr}\times N_{tr}}$ is the inverse of the degree matrix associated to the kernel matrix Ω. Φ is the $N_{tr} \times d_h$ feature matrix, $\Phi = [\phi(x_1)^\intercal;\ldots;\phi(x_{N_{tr}})^\intercal]$ and $\gamma_l \in \mathbb{R}^+$ are the regularization constants. We note that $N_{tr} \ll N$ i.e. the number of nodes in the training set is much less than the total number of nodes in the large scale network. The kernel matrix Ω is obtained by calculating the similarity between the adjacency list of each pair of nodes in the training set. Each element of Ω, denoted as $\Omega_{ij} = K(x_i,x_j) = \phi(x_i)^\intercal\phi(x_j)$ is obtained by calculating the cosine similarity between the adjacency lists x_i and x_j. Thus, $\Omega_{ij} = \frac{x_i^\intercal x_j}{\|x_i\|\|x_j\|}$ can be calculated efficiently using notions of set unions and intersections. This corresponds to using a normalized linear kernel function $K(x,z) = \frac{x^\intercal z}{\|x\|\|z\|}$ [22]. The clustering model is then represented by:

$$e_i^{(l)} = w^{(l)\intercal}\phi(x_i) + b_l, i = 1,\ldots,N_{tr} \tag{4}$$

where $\phi : \mathbb{R}^N \to \mathbb{R}^{d_h}$ is the mapping to a high-dimensional feature space of dimension d_h, b_l are the bias terms, $l = 1,\ldots,maxk-1$. However, for large scale networks we can utilize the explicit expression of the underlying feature map and $d_h = N$. Since

the KSC formulation is valid for any positive definite kernel, we use a normalized linear kernel function to avoid any kernel parameter. The projections $e_i^{(l)}$ represent the latent variables of a set of $maxk-1$ binary cluster indicators given by $\text{sign}(e_i^{(l)})$ which can be combined with the final groups using an encoding/decoding scheme. The dual problem corresponding to this primal formulation is:

$$D_\Omega^{-1} M_D \Omega \alpha^{(l)} = \lambda_l \alpha^{(l)} \tag{5}$$

where $M_D = I_{N_{tr}} - (\frac{(1_{N_{tr}} 1_{N_{tr}}^\intercal D_\Omega^{-1})}{1_{N_{tr}}^\intercal D_\Omega^{-1} 1_{N_{tr}}})$. The $\alpha^{(l)}$ are the dual variables and the kernel function $K : \mathbb{R}^N \times \mathbb{R}^N \rightarrow \mathbb{R}$ plays the role of similarity function.

2.3.2 Model Selection

KSC-net provides the user with the option to select one of the two model selection techniques i.e. BAF criterion [1] or the self-tuned procedure [3].

Data: Given codebook $\mathcal{CB}$, $maxk$ and $P = [e_{valid_1}, e_{valid_2}, \ldots, e_{valid_N}]$.

Result: Maximum BAF value and optimal k.

foreach $k \in (2, maxk]$ **do**

- Calculate the cluster memberships of training nodes using codebook $\mathcal{CB}$.
- Get the clustering $\Delta = \{C_1, \ldots, C_k\}$ & calculate cluster mean u_i for each C_i.
- For each validation node $valid_i$ obtain $\max_j \cos(\theta_{j,valid_i})$, where $cos(\theta_{j,valid_i}) = \frac{\mu_j^\intercal e_{valid_i}}{\|\mu_j\| \|e_{valid_i}\|}, j = 1, \ldots, k.$
- Maintain dictionary $MaxSim(valid_i) = \max_j \cos(\theta_{j,valid_i}), j = 1, \ldots, k.$
- Obtain clustering $\Delta_{valid} = \{C_{valid_1}, \ldots, C_{valid_k}\}$ for the validation nodes.
- Define BAF as: $BAF(k) = \sum_{i=1}^{k} \sum_{valid_j \in C_i} \frac{1}{k} \cdot \frac{MaxSim(valid_j)}{|C_i|}$.

end

Save the maximum BAF value along with the corresponding k.

Algorithm 1: Balance Angular Fitting (BAF) model selection criterion

2.3.3 Out-of-Sample Extension

Ideally when the communities are non-overlapping, we will obtain k well separated communities and the normalized Laplacian has k piece-wise constant eigenvectors. This is because the multiplicity of the largest eigenvalue i.e. 1 is k as depicted in [23]. In case of

Data: $P = [e_{valid_1}, e_{valid_2}, \ldots, e_{valid_N}]$.

Result: The number of clusters k in the given large scale network.

Construct S using the projection vectors $e_{valid_i} \in P$ & their CosDist().

Set $td = [0.1, 0.2, \ldots, 1]$.

foreach $t \in td$ **do**

    Save S in a temporary variable R i.e. $R := S$ & $SizeC_t = []$.

    while *S is not an empty matrix* **do**

        Find e_{valid_i} with maximum number of nodes whose cosine distance is $< t$.

        Determine the number of these nodes & append it to $SizeC_t$.

        Remove rows and columns corresponding to the indices of these nodes.

    end

    Calculate the entropy & expected balance from $SizeC_t$ as in [3].

    Calculate the F-measure $F(t)$ using entropy and balance.

end

Obtain the threshold corresponding to which F-measure is maximum as $maxt$.

Estimate k as the number of terms in the vector $SizeC_{maxt}$.

Algorithm 2: Algorithm to automatically identify k communities

KSC due to the centering matrix M_D the eigenvectors have zero mean and the optimal threshold for binarizing the eigenvectors is self-determined (equal to 0). So we need k-1 eigenvectors. However, in real world networks the communities exhibit overlap and don't have piece-wise constant eigenvectors.

The decoding scheme consists of comparing the cluster indicators obtained in the validation/test stage with the codebook $\mathcal{CB}$ and selecting the nearest codebook based on Hamming distance. This scheme corresponds to the ECOC decoding procedure [24] and is used in out-of-sample extensions as well.

The out-of-sample extension is based on the score variables which correspond to the projections of the mapped out-of-sample nodes onto the eigenvectors found in the training stage. The cluster indicators can be obtained by binarizing the score variables for the out-of-sample nodes as:

$$\text{sign}(e^{(l)}_{test}) = \text{sign}(\Omega_{test}\alpha^{(l)} + b_l 1_{N_{test}}) \tag{6}$$

where $l = 1, \ldots, k-1$, Ω_{test} is the $N_{test} \times N_{tr}$ kernel matrix evaluated using the test nodes

with entries $\Omega_{r,i} = K(x_r, x_i)$, $r = 1, \ldots, N_{test}$ and $i = 1, \ldots, N_{tr}$. This natural extension to out-of-sample nodes is the main advantage of KSC. In this way the clustering model can be trained, validated and tested in an unsupervised learning procedure. For the test set we use the entire network. If the network cannot fit in memory, we divide it into blocks to calculate the cluster memberships for each test block in parallel.

2.4 Practical Issues

In this chapter all the graphs are undirected and unweighted unless otherwise specified. We conducted all our experiments on a machine with 12Gb RAM, 2.4Ghz Intel Xeon processor. The maximum cardinality allowed for the train (N_{tr}) and the validation set (N_{valid}) is $5,000$ nodes. This is because the maximum size kernel matrix that we can store in the memory of our PC is $5,000 \times 5,000$. We use the entire network as test set V_{test}. We perform the community affiliation for unseen nodes in an iterative fashion by dividing V_{test} into blocks of $5,000$ nodes. This is because the maximum size Ω_{test} that we can store in the memory of our PC is $5,000 \times 5,000$. However, this operation can easily be extended to a distributed environment and performed in parallel.

3 KSC-net software

The KSC-net software is available for Windows and Linux platform at `ftp://ftp.esat.kuleuven.be/SISTA/rmall/KSCnet_Windows.tar.gz` and `ftp://ftp.esat.kuleuven.be/SISTA/rmall/KSCnet_Linux.tar.gz` respectively. In our demos we use benchmark synthetic network obtained from [9] and anonymous network available freely from the Stanford SNAP library `http://snap.stanford.edu/data/index.html`.

3.1 KSC Demo on Synthetic Network

We first provide a demoscript to showcase the KSC methodology on synthetic network which has $5,000$ nodes and $124,756$ edges. The network comprises 7 communities which are known beforehand and act as ground truth. We provide the demo script below: Figure 3 is also obtained as an output of the demoscript to show the effectiveness of the BAF model selection criterion. From Figure 3 we can observe that the maximum BAF value is 0.9205 and it occurs corresponding to $k = 7$ which is equal to the actual number

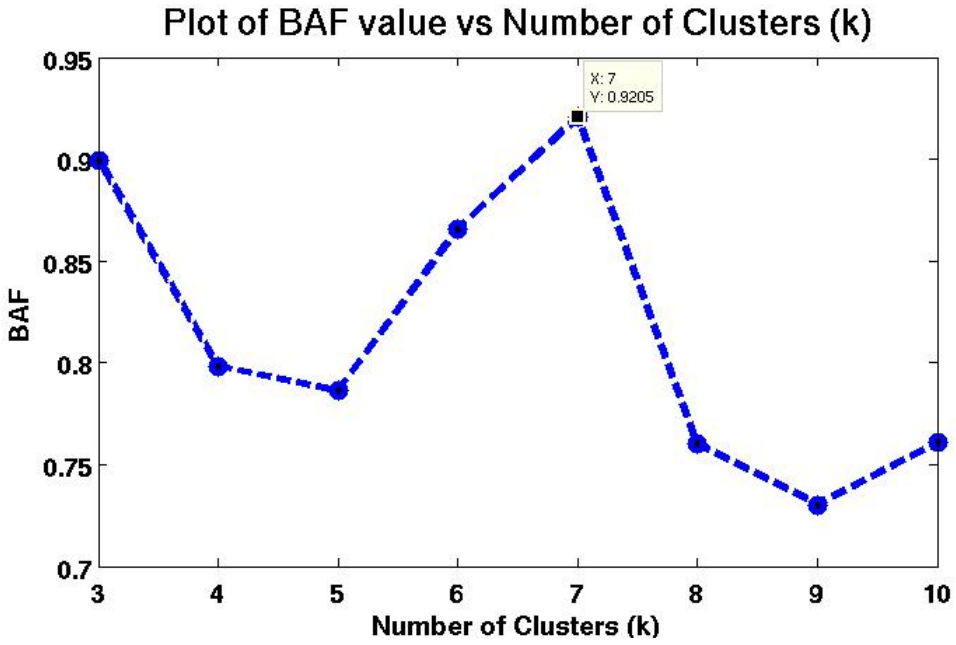


Figure 3: Selection of optimal k by BAF criterion for Synthetic network.

of communities in the network. The ARI (adjusted rand index [25]) value is 0.999 which means the community membership provided by the KSC methodology is as good as the original ground truth.

If we put mod_sel = 'Self' in the demoscript of Algorithm 3 then we perform the self-tuned model selection. It uses the concept of entropy and balance clusters to identify the optimal number of clusters k in the network. Figure 4 showcases the block diagonal affinity matrix S generated from the eigen-projections of the validation set using the self-tuned model selection procedure. From Figure 4 we can observe there are 7 blocks in the affinity matrix S. In order to calculate the number of these blocks the self-tuned procedure uses the concept of entropy and balance together as an F-measure. Figure 4 shows that the F-measure is maximum for threshold $t = 0.2$ for which it takes the value 2.709. The number of clusters k identified corresponding to this threshold value is 7 which is equivalent to the number of ground truth communities.

3.2 KSC Sub-functions

We briefly describe some of the sub-functions that are called within the KSCnet() function. Firstly, the large scale network is provided as edgelist with or without the ground truth information, we convert the edgelist into a sparse adjacency matrix A. The

Data:

netname = 'network' ; `// Name of the file and extension should be '.txt'`

baseinfo = 1 ; `// Ground truth exists`

cominfo = 'community' ; `// Name of ground truth file and extension should be '.txt'`

weighted = 0 ; `// Whether a network is weighted or unweighted`

frac_network = 15 ; `// Percentage of network to be used as training and validation set`

maxk = 10 ; `//` $maxk$ `is maximum value of` k `to be used in eigen-decomposition, use` $maxk = 10$ `for` $k <= 10$ `else use` $maxk = 100$

mod_sel = 'BAF' ; `// Method for model selection (options are 'BAF' or 'Self')`

output = KSCnet(netname,baseinfo,cominfo,weighted,frac_network,maxk,mod_sel);

Result:

output = 1 ; `// When community detection operation completes`

netname_numclu_mod_sel.csv ; `// Number of clusters detected`

netname_outputlabel.csv ; `// Cluster labels assigned to the nodes`

netname_ARI.csv ; `// Valid only when ground truth is known, provides information about ARI and number of communities covered`

Algorithm 3: Demoscript for Synthetic Network using BAF criterion

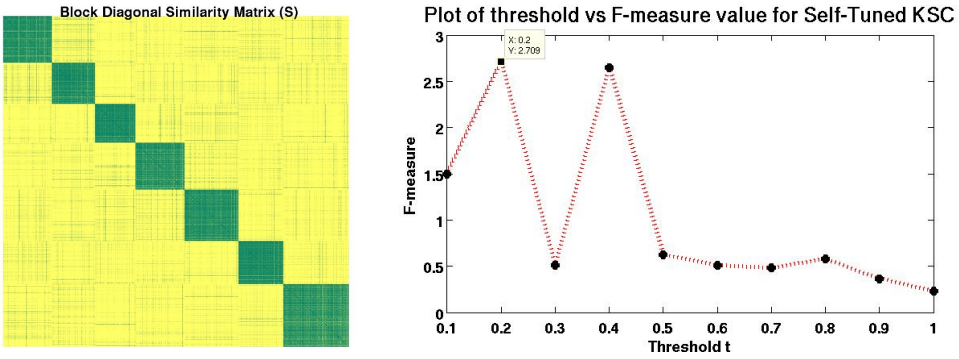


Figure 4: Selection of optimal k by self-tuned criterion for Synthetic network.

frac_network provides information about the what percentage of total number of nodes in the network is to be used as training and validation set. We put a condition to check that the N_{tr} or N_{valid} cannot exceed maximum value of $5,000$. We then use 'system' com-

mand to run the FURS selection technique as it is implemented in the scripting language Python. The python pre-requisites are provided in the 'ReadMe.txt'. Figure 5 demonstrates the output obtained when the FURS selection technique is used for selecting the training and validation set.

```
>> demoscript
First Step of the Sampling Process Completed
Degree Matrix Empty for the 1 time
Degree Matrix Empty for the 2 time
0.960400104523 seconds
Elapsed time is 1.072561 seconds.
First Step of the Sampling Process Completed
Degree Matrix Empty for the 1 time
Degree Matrix Empty for the 2 time
0.637423992157 seconds
Elapsed time is 0.743813 seconds.
```

Figure 5: Output of FURS for selecting training and validation set.

From Figure 5 we can observe that the statement 'First Step of Sampling Process Completed' appears twice (once for training and once for validation).

- KSCmod - The KSCmod function takes as input a boolean state variable to determine whether it's the training or test phase. If its 0 then we provide as input the training and validation set otherwise the input is training and test set. It also takes as input maxk and the eigenvalue decomposition algorithm (options are 'eig' or 'eigs'). It calculates the kernel matrix using a Python code and then solves the eigen-decomposition problem to obtain the model. Output is the model along with its parameters, training/validation/test set eigen-projections and codebook $\mathcal{CB}$. For the validation set we also obtain the community affiliation corresponding to $maxk$ but for the test set we obtain cluster membership corresponding to optimal k.

- KSCcodebook - Takes as input the eigen-projections and eigen-vectors. Takes the sign of these eigen-projections and calculates the top k most frequent codewords and outputs them as codebook. It also estimates the cluster memberships for the training nodes using the KSCmembership function.

- KSCmembership - Takes as input the eigen-projections of the train/validation/test set and codebook . Takes a sign of these eigen-projections and calculates the hamming distance between the codebook and the sign(e_i). Assigns the corresponding

node to that cluster with which it has minimum hamming distance.

In case of 'BAF' model selection criterion we utilize the KSCcodebook and KSCmembership functionality to estimate the optimal number of communities k in the large scale network. However, the 'Self' model selection technique works independent of these functions and uses concept of entropy and balance to calculate the number of block diagonals in the affinity matrix S generated by the validation set eigen-projections after re-ordering by means of the ground truth information. Figure 6 represents snippet which we obtain when trying to identify the optimal number of communities k in the large scale network.

```
Tuning of the number of clusters
/usr/lib64/python2.6/site-packages/scipy/maxentropy/__init__.py:19: DeprecationWarning:
The scipy.maxentropy module is deprecated in scipy 0.10, and scheduled to be
removed in 0.11.

If you are using some of the functionality in this module and are of the
opinion that it should be kept or moved somewhere - or you are even interested
to maintain/improve this whole module - please ask on the scipy-dev mailing
list.

The logsumexp function has already been moved to scipy.misc.
  DeprecationWarning)
Elapsed time is 17.605555 seconds.
/usr/lib64/python2.6/site-packages/scipy/maxentropy/__init__.py:19: DeprecationWarning:
The scipy.maxentropy module is deprecated in scipy 0.10, and scheduled to be
removed in 0.11.

If you are using some of the functionality in this module and are of the
opinion that it should be kept or moved somewhere - or you are even interested
to maintain/improve this whole module - please ask on the scipy-dev mailing
list.

The logsumexp function has already been moved to scipy.misc.
  DeprecationWarning)
Elapsed time is 17.627895 seconds.
```

Figure 6: Output snippet obtained while identifying k.

In Figure 6 we observe a 'DeprecationWarning' which we ignore as it is related to some functionality in 'scipy' library which we are not using. We also observe that we obtained 'Elapsed time ... ' at two places. Once it occurs after building the training model and the second time it appears after we have performed the model selection using validation set. Figure 7 represents the time required to obtain the test set community affiliation. Since, there are only $5,000$ nodes in the entire network and we divide the test set into blocks of $5,000$ nodes due to memory constraints, we observe 'I am in block 1' in Figure 7. Figure 8 showcases the community based structure for the synthetic network. We get the same result in either case when we use the 'BAF' selection criterion or the 'Self' selection technique.

```
I am in block 1
/usr/lib64/python2.6/site-packages/scipy/maxentropy/__init__.py:19: DeprecationWarning:
The scipy.maxentropy module is deprecated in scipy 0.10, and scheduled to be
removed in 0.11.

If you are using some of the functionality in this module and are of the
opinion that it should be kept or moved somewhere - or you are even interested
to maintain/improve this whole module - please ask on the scipy-dev mailing
list.

The logsumexp function has already been moved to scipy.misc.
  DeprecationWarning)
Elapsed time is 110.644138 seconds.
```

Figure 7: Output snippet obtained while estimating the test cluster membership.

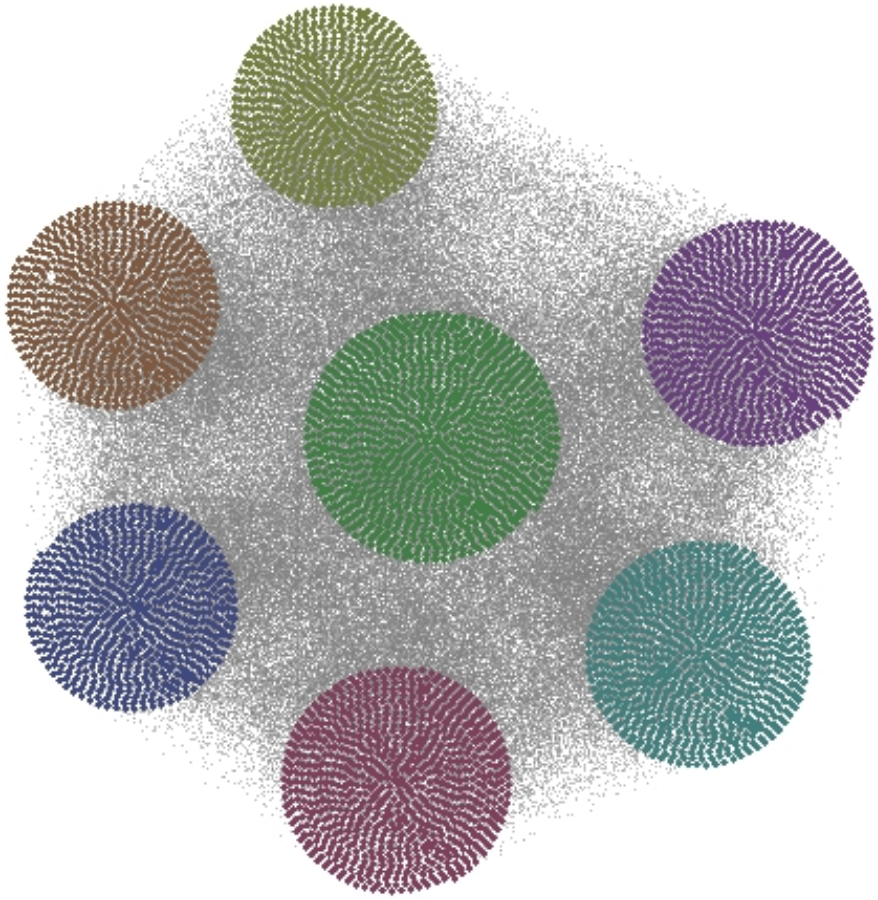


Figure 8: Communities detected by KSC methodology for the Synthetic Network

3.3 KSC Demo on Real-life Network

We provide a demoscript to showcase the KSC methodology on a real-life large scale Youtube network which has $1,134,890$ nodes with $2,987,624$ edges. The network is highly sparse. It is a social network where users form friendship with each other and can create groups which other users can join. We provide the demoscript below:

Figure 9 shows the effectiveness of BAF model selection criterion on the Youtube

Data:

netname = 'youtube_mod' ; `// Name of the file and extension should be '.txt'`

baseinfo = 0 ; `// Ground truth does not exists`

cominfo = [] ; `// No ground truth file so empty set as input`

weighted = 0 ; `// Whether a network is weighted or unweighted`

frac_network = 10 ; `// Percentage of network to be used as training and validation set`

maxk = 100 ; `//` $maxk$ `is maximum value of` k `to be used in eigen-decomposition, use` $maxk = 10$ `for` $k <= 10$ `else use` $maxk = 100$

mod_sel = 'BAF' ; `// Method for model selection (options are 'BAF' or 'Self')`

output = KSCnet(netname,baseinfo,cominfo,weighted,frac_network,maxk,mod_sel);

Result:

output = 1 ; `// When community detection operation completes`

netname_numclu_mod_sel.csv ; `// Number of clusters detected`

netname_outputlabel.csv ; `// Cluster labels assigned to the nodes`

Algorithm 4: Demoscript for Youtube network using BAF criterion

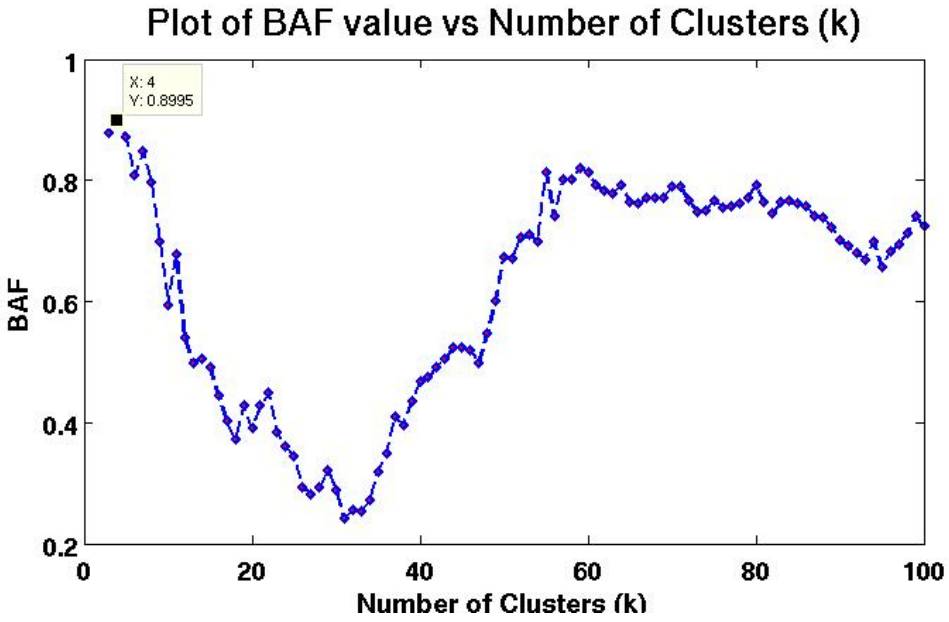


Figure 9: Selection of optimal k by BAF criterion for Youtube network.

social network. From Figure 9 we observe that there are multiple local peaks on the BAF versus k curve. However, we select the peak corresponding to which the BAF value is maximum. This occurs at $k = 4$ for the BAF value of 0.8995. Since, we provide the cluster memberships for the entire network to the end user, the end user can use various internal quality criteria to estimate the quality of the resulting communities.

If we put mod_sel = 'Self' in the demoscript of Algorithm 4 then we would be using the parameter-free model selection technique. Since the ground truth communities are not known before hand, we try to estimate the block diagonal structure using the greedy selection technique mentioned in Algorithm 2 to determine the $SizeC_t$ vector. Figure 10 shows that the F-measure is maximum for $t = 0.1$ for which it takes the value 0.224. The number of communities k identified corresponding to this threshold value is 36.

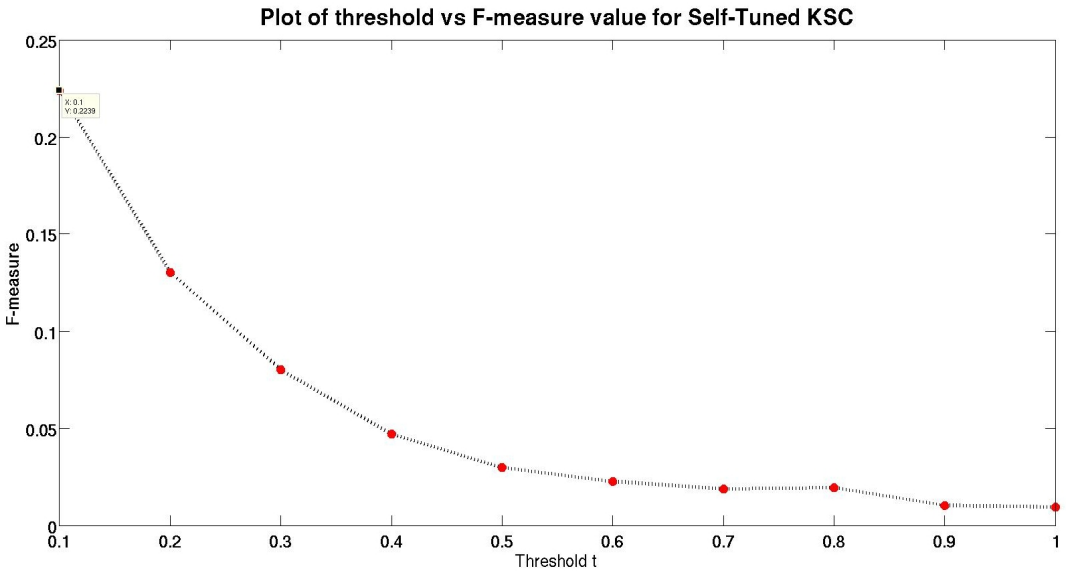


Figure 10: Selection of optimal k by self-tuned criterion for Youtube network.

4 Conclusion

In this chapter we gave a practical exposition of a methodology to perform community detection for big data networks. The technique was built on the concept of spectral clustering and referred as kernel spectral clustering (KSC). The core concept was to build a model on a small representative subgraph which captured the inherent community structure of the large scale network. The subgraph was obtained by FURS [2] selection technique. Then, the KSC model was built on this subgraph. The model parameters were obtained by one of the two techniques: a) Balance Angular Fitting (BAF) criterion or b)

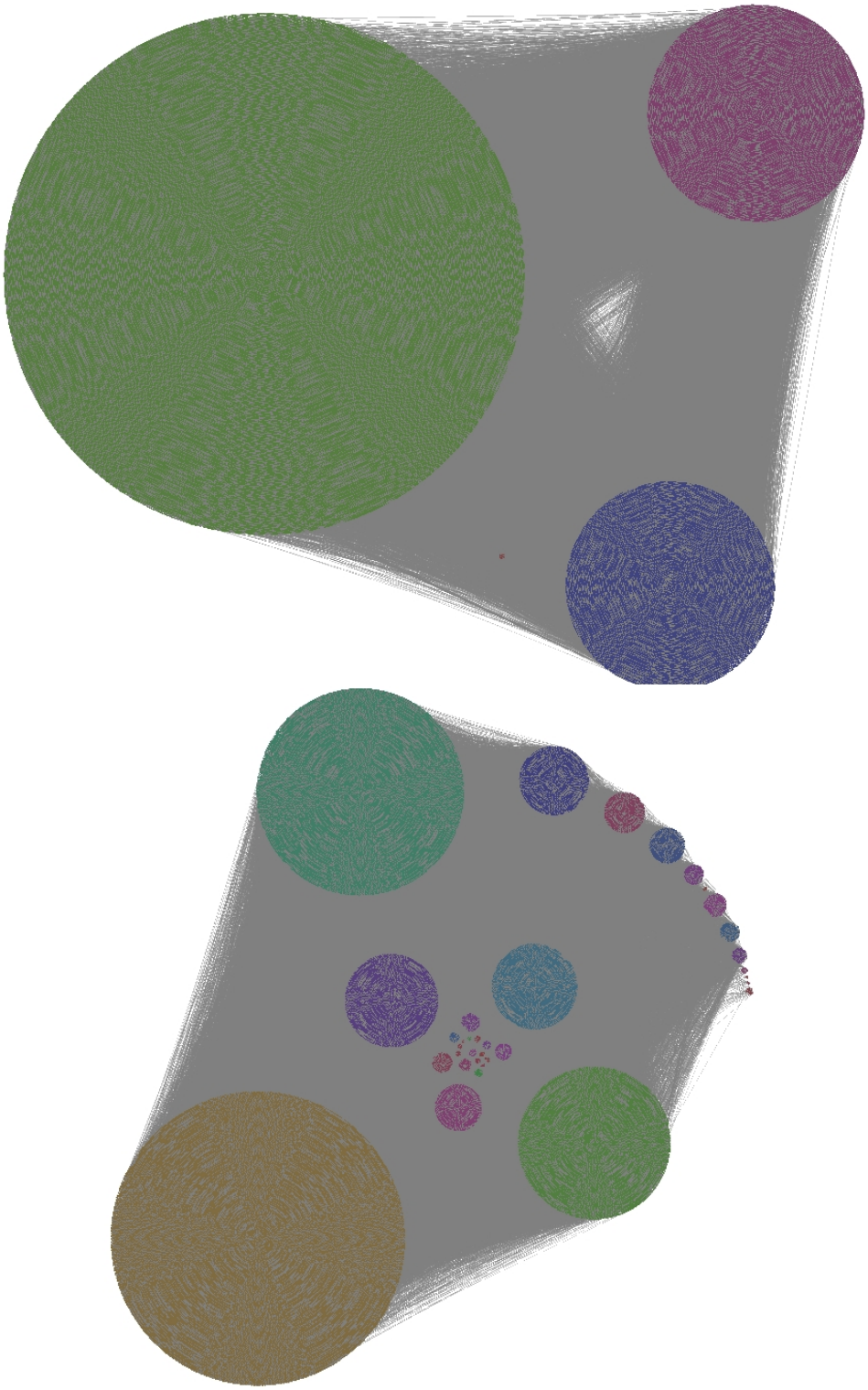


Figure 11: Community structure detected for the Youtube network by 'BAF' and 'Self' model selection techniques respectively visualized using the software provided in [26].

Self-tuned technique. The KSC model has a powerful out-of-sample extensions property. This property was used for community affiliation of previously unseen nodes in the big data network. We also explained and demonstrated the usage of the KSC-net software

which uses the KSC methodology as the underlying core concept.

Acknowledgements

EU: The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC AdG A-DATADRIVE-B (290923). This chapter reflects only the authors' views, the Union is not liable for any use that may be made of the contained information. Research Council KUL: GOA/10/09 MaNet, CoE PFV/10/002 (OPTEC), BIL12/11T; PhD/Postdoc grants. Flemish Government: FWO: projects: G.0377.12 (Structured systems), G.088114N (Tensor based data similarity); PhD/Postdoc grants. IWT: projects: SBO POM (100031); PhD/Postdoc grants. iMinds Medical Information Technologies SBO 2014. Belgian Federal Science Policy Office: IUAP P7/19 (DYSCO, Dynamical systems, control and optimization, 2012-2017.)

References

[1] R. Mall, R. Langone and J.A.K. Suykens; Kernel Spectral Clustering for Big Data Networks. *Entropy (Special Issue: Big Data)*, 15(5):1567-1586, 2013.

[2] 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.

[3] R. Mall, R. Langone and J.A.K. Suykens; Self-Tuned Kernel Spectral Clustering for Large Scale Networks. *Proceedings of the IEEE International Conference on Big Data*, (IEEE BigData 2013), October 6-9, Santa Clara (U.S.A), 2013.

[4] S. Schaeffer; Algorithms for Nonuniform Networks. *Phd thesis*, Helsinki University of Technology, 2006.

[5] L. Danaon, A. Diáz-Guilera, J. Duch and A. Arenas; Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 09(P09008+),2005.

[6] S. Fortunato; Community detection in graphs. *Physics Reports*, 486:75-174, 2009.

[7] A. Clauset, M. Newman and C. Moore; Finding community structure in very large scale networks. *Physical Review E*, 70(066111), 2004.

[8] M. Girvan and M. Newman; Community structure in social and biological networks. *PNAS*, 99(12):7821-7826, 2002.

[9] A. Lancichinetti and S. Fortunato; Community detection algorithms: a comparitive analysis. *Physical Review E*, 80(056117), 2009.

[10] M. Rosvall, and C. Bergstrom; Maps of random walks on complex networks reveal community structure. *PNAS*, 105:1118-1123, 2008.

[11] R. Langone, C. Alzate and J.A.K. Suykens; Kernel spectral clustering for community detection in complex networks. *In IEEE WCCI/IJCNN*, pp. 2596-2603, 2002.

[12] V. Blondel, J. Guillaume, R. Lambiotte and L. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statisitcal Mechanics: Theory and Experiment*, 10(P10008), 2008.

[13] A.Y. Ng, M.I. Jordan, and Y. Weiss; On spectral clustering: analysis and an algorithm. *In proceedings of the Advances in Neural Information Processing Systems*; Dietterich, T.G., Becker, S., Ghahramani, Z., editors ;MIT Press: Cambridge, MA, pp. 849-856, 2002.

[14] U. von Luxburg; A tutorial on Spectral clustering. *Stat. Comput*, 17:395-416, 2007.

[15] L. Zelnik-Manor and P. Perona; Self-tuning spectral clustering. *Advances in Neural Information Processing Systems*; Saul, L.K., Weiss, Y., Bottou, L., editors; MIT Press: Cambridge, MA, pp. 1601-1608, 2005.

[16] J. Shi and J. Malik; Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Intelligence*, 22(8):888-905, 2000.

[17] 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, 2010.

[18] L. Muflikhah.; Document clustering using concept space and concept similarity measurement. *ICCTD*, pp. 58-62, 2009.

[19] A. Maiya and T. Berger-Wolf; Sampling community structure. *WWW*, pp. 701-710, 2010.

[20] U. Kang and C. Faloutsos; Beyond 'caveman communities': Hubs and Spokes for graph compression and mining. *In Proceedings of ICDM*, pp. 300-309, 2011.

[21] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller; Equation of state calculations by fast computing machines. *Journal of Chem. Phys.*, 21(6): 1087-1092, 1953.

[22] 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.

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

[24] J. Baylis; Error Correcting Codes: A Mathematical Introduction. *Boca Raton, FL: CRC Press*, 1988.

[25] R. Rabbany, M. Takaffoli, J. Fagnan, O.R. Zaiane, and R.J.G.B. Campello; Relative Validity Criteria for Community Mining Algorithms. *International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 258-265, 2012.

[26] A. Lancichinetti, F. Radicchi, J.J. Ramasco and S. Fortunato; Finding statistically significant communities in networks. *Plos One*, 6(e18961), 2011.