Self-Tuned Kernel Spectral Clustering for Large Scale Networks

Raghvendra Mall
Department of Electrical Engineering
ESAT-SCD
Katholieke Universiteit Leuven
Kasteelpark Arenberg 10
B-3001 Leuven, Belgium
Email: rmall@esat.kuleuven.be

Rocco Langone
Department of Electrical Engineering
ESAT-SCD
Katholieke Universiteit Leuven
Kasteelpark Arenberg 10
B-3001 Leuven, Belgium

Johan A.K. Suykens
Department of Electrical Engineering
ESAT-SCD
Katholieke Universiteit Leuven
Kasteelpark Arenberg 10
B-3001 Leuven, Belgium

***Abstract*—We propose a parameter-free kernel spectral clustering model for large scale complex networks. The kernel spectral clustering (KSC) method works by creating a model on a subgraph of the complex network. The model requires a kernel function which can have parameters and the number of communities k has be detected in the large scale network. We exploit the structure of the projections in the eigenspace to automatically identify the number of clusters. We use the concept of entropy and balanced clusters for this purpose. We show the effectiveness of the proposed approach by comparing the cluster memberships w.r.t. several large scale community detection techniques like Louvain, Infomap and Bigclam methods. We conducted experiments on several synthetic networks of varying size and mixing parameter along with large scale real world experiments to show the efficiency of the proposed approach.**

***Keywords*-kernel spectral clustering, number of clusters, parameter-free spectral clustering**

I. Introduction

In the modern era complex networks are ubiquitous. Their presence ranges from domains like road graphs, social networks, collaboration networks, e-mail networks, trust networks, biological networks to financial networks. Complex networks can be represented as a graph $G(V, E)$, where the vertices V represent the nodes of the graph and the edges are represented by E. The nodes represent the entities in the network and the edges determine the relationship between these entities. Real world networks exhibit community like structure. This means that the nodes which are part of one community tend to be more densely connected within the community and sparsely connected between the communities. The problem of community detection has received a lot of attention [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12] and has often been framed as graph partitioning and graph clustering [13]. In this paper we refer to clusters and communities interchangeably.

Spectral clustering methods [9], [10], [11], [12] are widely used for community detection. Spectral clustering methods work by performing an eigen-decomposition of a normalized Laplacian matrix derived from the affinity matrix of the nodes in the network. The disadvantage of spectral clustering methods is that they require to compute and store the affinity matrix. As the size of network increases it becomes in-feasible to store the $N \times N$ affinity matrix where N is the number of nodes. Also, these methods do not have an out-of-sample extension property. Hence, they cannot scale for very large networks (millions of nodes).

Recently, a kernel spectral clustering formulation based on weighted kernel principal component analysis (PCA) with a primal-dual framework was proposed in [14]. The method was extended for community detection in [7] and [22]. The KSC model requires a kernel function and the number of clusters k has to be detected. For model selection, the proposed approach in [7] and [22] uses the concept of Modularity [16]. The Modularity criteria needs to store a $N \times N$ matrix which makes it in-feasible for large scale networks. An approach to apply KSC for big data networks was proposed in [15]. We use the concepts proposed in [15] to build the model. In this paper we propose to make the kernel spectral clustering approach parameter-free or in other words

- Use the normalized linear kernel function.
- Automatically detect the number of communities (k) in the large scale network.

In section II, a description of the KSC method for large scale networks is given. In section III, we explain the proposed approach for making KSC parameter-free. In section IV, we perform experiments on various synthetic and real world datasets. Section V provides conclusion to the paper.

II. Kernel Spectral Clustering for Large Scale Networks

The KSC [14] and [7] method consists of three steps - a) Building a model on the training data, b) Estimate the model parameters using validation data and c) The out-of-sample extension for unseen test nodes.

For building the model, the KSC method selects a subgraph which captures the inherent community structure of the network. Several approaches have been proposed for sampling a graph including [17], [18], [19], [20]. The Fast and Unique Representative Subset (FURS) selection approach was proposed in [18] and used in [15]. We use the FURS technique for training and validation set selection.

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.

A. Model Building and overcoming any kernel parameters

For V_{tr} training nodes, $\mathcal{D} = \{x_i\}_{i=1}^{N_{tr}}$, such that $x_i \in \mathbb{R}^N$. Given $\mathcal{D}$ and a user-defined k (maximum number of clusters in the network), the primal formulation of the weighted kernel PCA [14] is given by:

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

where $e^{(l)} = [e_1^{(l)}, \ldots, e_{N_{tr}}^{(l)}]^\intercal$ are the projections onto the eigenspace, $l = 1, \ldots, k-1$ indicates the number of score variables required to encode the k 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 which has been shown to be an effective similarity measure for large scale data [25]. 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 $K(x, z) = \frac{x^\intercal z}{\|x\| \|z\|}$ [24]. The clustering model is then represented by:

$$e_i^{(l)} = w^{(l)\intercal} \phi(x_i) + b_l, i = 1, \ldots, N_{tr} \tag{2}$$

where $\phi : \mathbb{R}^N \to \mathbb{R}^{d_h}$ is the mapping to a high-dimensional feature space d_h, b_l are the bias terms, $l = 1, \ldots, k-1$. 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 $k-1$ 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:

$$D_\Omega^{-1} M_D \Omega \alpha^{(l)} = \lambda_l \alpha^{(l)} \tag{3}$$

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 $K : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$ plays the role of similarity function.

B. Out-of-Sample Extension

Ideally when the communities are non-overlapping, we will obtain k well separated clusters 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 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 do overlap and we do not have piece-wise constant eigenvectors.

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

$$\text{sign}(e_{test}^{(l)}) = \text{sign}(\Omega_{test} \alpha^{(l)} + b_l 1_{N_{test}}) \tag{4}$$

where $l = 1, \ldots, k-1$, Ω_{test} is the $N_{test} \times N_{tr}$ kernel matrix evaluated using the test points 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 points is the main advantage of KSC. For the test set we use the entire network. If the network cannot fit in memory, we divide it into chunks to calculate the cluster memberships for each test chunk in parallel.

C. Model Selection or Estimation of k

In [15], the authors proposed a criterion called Balanced Angular Fit (BAF) which tries to estimate the number of clusters using the validation set of nodes. The BAF criterion is based on the principle that the nodes which belong to the same cluster have nearly zero or very small angular distance in the eigenspace w.r.t. each other in comparison to the nodes belonging to different clusters. To determine the number of clusters, the end-user provides a maximum possible number of clusters ($maxk$). The KSC model is built using this $maxk$ i.e. we perform an eigen-decomposition of the $D^{-1} M_D \Omega$ matrix and select the top $maxk$ eigenvectors. The BAF criterion is iteratively evaluated for $k = 2$ to $k = maxk - 1$. The value of k for which the BAF is maximum is provided to the end-user as the ideal number of clusters. The method works well but requires to iterate over all values of k from 2 to $maxk - 1$. Hence, it doesn't automatically determine the ideal value of k in a few steps. Another approach to automatically determine the ideal value of k was proposed in [11] using the structure of the eigenvectors. However, the underlying assumption is that the eigenvector matrix Z comprises k piece-wise constant eigenvectors and there is at most one non-zero entry in every row. Both these assumptions do not hold for the KSC model for real world overlapping networks. Algorithm 1 provides a summary of the Original KSC method.

Algorithm 1: Original KSC Algorithm

Data: Given a graph $G = (V, E)$.

Result: The partitions of the graph G, i.e. divide graph into k clusters.

1 Convert and store the graph G in sparse format.

2 Select the training set of nodes X_{tr} (i.e. maximum size $= 5,000$ nodes) using FURS.

3 Select the validation set of nodes (same size as X_{tr}) after removing the subgraph S corresponding to X_{tr} using FURS.

4 Calculate the kernel matrix Ω by applying cosine similarity operations on sparse adjacency lists of $\forall i, j$ $v_i, v_j \in X_{tr}$.

5 Perform eigen-decomposition of Ω to obtain the model i.e. $\alpha^{(l)}, b_l$.

6 Use out-of-sample extension property to obtain the projection values for the validation set i.e. $e^{(l)}_{valid}$.

7 **Use BAF and $e^{(l)}_{valid}$ to estimate the number of clusters k.**

`// This is the step which is automated by our proposed approach.`

8 After estimating k, use the out-of-sample property to assign clusters to unseen test nodes (We use the entire network as test data).

III. Self-Tuned Kernel Spectral Clustering

In section II-A, we have shown that for large scale networks the normalized linear kernel function is sufficient. The only parameter left is the number of clusters to be identified (k). We propose a novel technique to determine the number of clusters (k) automatically in a few steps. We exploit the projections of the validation nodes in the eigenspace. The proposed approach is based on the simple idea that the projections of the validation nodes which belong to the same cluster have very small or nearly zero angular distance from each other in comparison with the nodes belonging to different clusters.

Before we explain the technique, we mention a few pre-requisites. All the experiments were performed on a single machine with 8 Gb RAM, 2.4Ghz Intel I-7 processor. The maximum size matrix that can be stored in memory is $5,000 \times 5,000$. The training set size is fixed as $N_{tr} = \min(15\%$ of N, $5000)$ i.e. minimum between 15% of the total nodes in the network (N) and $5,000$ nodes. The subset size of 15% of the nodes in the network is chosen as per experimental analysis in [21]. We set the validation set size N_{valid} to be the same as N_{tr}. We put the condition that minimum size of a cluster is $MinCsize = \max(0.01\%$ of $N_{valid}, 5)$ i.e. maximum between 0.01% of the number of nodes in the validation set and 5 nodes. Since for estimation of the number of clusters we use the validation set, we put the minimum size constraint on the validation set. This constraint is to prevent the selection of outlier groups of small size as a cluster. Thus, suppose the training set size is $5,000$ nodes then the minimum size of cluster has to be 5. The $maxk$ value is determined as $\lceil \frac{N_{tr}}{MinCsize} \rceil$. Thus, if the minimum size of the cluster is 5 and maximum size of the validation set is $5,000$ then the maximum number of clusters that can be detected is $1,000$. The runtime complexity of KSC approach is $\mathrm{O}(N_{tr}^3) + \mathrm{O}(N_{tr} \times N_{test})$. However, testing can be done in parallel and the second term in time complexity can be reduced significantly.

A. Identifying the number of clusters

We obtain the projection of the validation nodes $v_j \in V_{valid}$ in the eigenspace using the out-of-sample extension and binarize the score variables as follows:

$$\mathrm{sign}(e^{(l)}_{valid}) = \mathrm{sign}(\Omega_{valid}\alpha^{(l)} + b_l 1_{N_{valid}}) \tag{5}$$

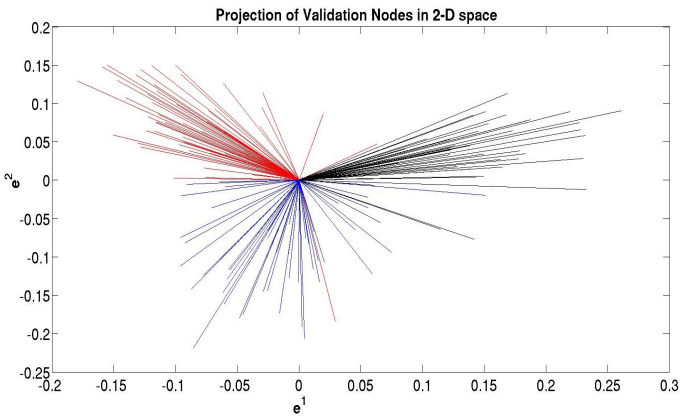
Let the latent variable matrix be represented as E. The size of the this matrix is $N_{valid} \times (maxk - 1)$. The matrix E is defined as $[e_1, \ldots, e_{N_{valid}}]^\intercal$. We then create a new symmetric affinity matrix A from the latent variable matrix E. Since we already stated that the maximum size of the validation set is $5,000$ nodes, the maximum size of the A matrix is $5,000 \times 5,000$. The affinity matrix is build using cosine distance as:

$$A(i,j) = CosDist(e_i, e_j) = 1 - \cos(e_i, e_j) = 1 - \frac{e_i^\intercal e_j}{\|e_i\|\|e_j\|}$$

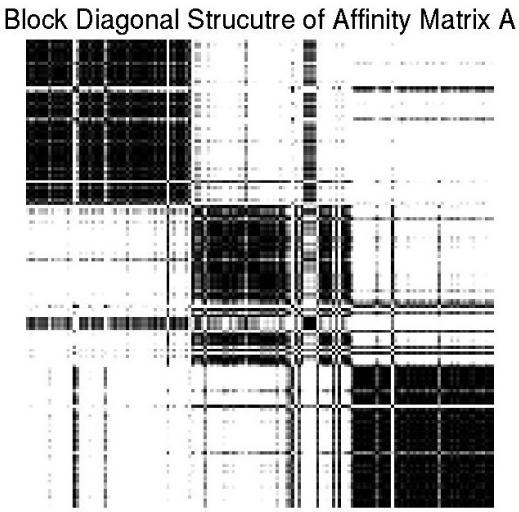
The $CosDist()$ function can take values between $[0, 2]$. We use the concept that nodes which belong to the same community have smaller or nearly zero angular distance w.r.t the nodes within the community and more angular distance w.r.t nodes belonging to different communities. Thus, the nodes which belong to the same cluster will have very small or nearly zero $CosDist(e_i, e_j), \forall i, j$ in the same cluster.

Figure 1 highlights this idea on a synthetic network of $1,000$ nodes generated by the software proposed in [2] with mixing parameter $\mu = 0.1$. The mixing parameter implies the extent of overlap in the communities with smaller values representing less overlap. In Figure 1a, we obtain the projections of the validation nodes in 2-dimensional space as we set $maxk = 3$ for the purpose of visualization. Figure 1b showcases the presence of a block diagonal structure for the affinity matrix A. To obtain this block diagonal matrix we used the groundtruth cluster memberships for the validation set of nodes. We then sort the validation nodes using the groundtruth information on the basis of their cluster affiliation and construct the affinity matrix A. The black shaded regions represent the part of the affinity matrix where the $CosDist(e_i, e_j)$ values are zero and nearly equal to zero. The aim is to identify the number of such block diagonals which determine the number of clusters k in the given network.

Real world networks have overlapping communities. However, the extent of overlap in these communities is not known beforehand. Also the groundtruth is not available in most real world large scale networks. Thus the affinity matrix A, though contains a block-diagonal structure, need not necessarily be ordered in block-diagonal format. We observe from Figure 1a, that projection of nodes which belong to the same cluster do not necessarily have zero angular distance due to presence



(a) Validation node projections



(b) Affinity matrix A w.r.t. groundtruth

Fig. 1: First two steps for identifying the number of clusters in a given network

of overlap. But the angle between two projection vectors say e_i and e_j belonging to the p^{th} cluster (C_p) is mostly less than 90° leading to $0 \leq \cos(e_i, e_j) \leq 1$. After little algebraic manipulation we obtain that $0 \leq CosDist(e_i, e_j) \leq 1$ for two nodes i and j belonging to cluster p (C_p).

Since the extent of overlap in terms of angular distance is not known beforehand we setup a threshold $td = [0.1, 0.2, \ldots, 1]$. We need to investigate only 10 different threshold values which is much less than iteratively solving for each $k = 2$ to $k = maxk - 1$ in increments of 1. Generally as the value of threshold increases the number of clusters decreases as more validation nodes satisfy the threshold and fall in the same cluster. For a given value of threshold $t \in td$, we first identify that validation node (e_i) for which the number of nodes with $CosDist(e_i, e_j) \leq t$ is maximum. We keep the count of this number of nodes in a vector. We obtain the indices of all those validation nodes and shrink the affinity matrix A by removing the rows and columns corresponding to these indices. We repeat this process recursively till it results in an empty matrix. This procedure is equivalent to identifying the block diagonal structure. As a result of this procedure for each value of threshold t, we obtain a vector ($SizeC_t$) containing information about the size of clusters for that value of threshold t.

In order to determine the ideal number of clusters k, we use the notion of entropy and balance w.r.t. the size of the clusters. For each value of threshold t we have the vector $SizeC_t$. We also put the constraint that the minimum size of a cluster is greater than or equal to $MinCsize$. So we remove from $SizeC_t$ vector all the size of clusters which are less than $MinCsize$. The Shannon entropy for a discrete set of probabilities is defined as:

$$H(s)^t = -\sum_{i=1}^{k} p(s_i)^t \log(p(s_i^t)), \forall t \in td \tag{6}$$

where $H(s)^t$ is the entropy corresponding to the vector $SizeC_t$ for the threshold t. Here s_i^t is the size of the i^{th} cluster and $p(s_i^t)$ is defined as ratio between the size of i^{th} cluster corresponding to threshold t and the size of the validation set (N_{valid}). Entropy values are higher when there is a larger number of uniformly sized clusters.

Balance is generally defined as the ratio between the minimum size cluster in $SizeC_t$ and the maximum size cluster in $SizeC_t$. The aim of using balance is to prevent one or more large community from being superior over all other communities. Higher balance promotes uniformly sized clusters and prevents few extremely large communities from being dominant. This leads to more number of communities. In this paper, we use the overall balance criterion which is the sum of balance for all the clusters in $SizeC_t$ and defined as:

$$B(s)^t = \sum_{i=1}^{k} \frac{s_i^t}{\max(s_1^t, \ldots, s_k^t)}, \forall t \in td \tag{7}$$

where s_i^t represents the size of the i^{th} cluster corresponding to threshold t. The denominator in (7) refers to maximum sized cluster for the threshold t. The higher the overall balance in the clusters, the closer the value of $B(s)^t$ is to k.

We calculate the entropy and overall balance for each of these $SizeC_t$ and use the harmonic mean or an F-measure to determine the best of both entropy and overall balance as their values are comparable. The F-measure is defined as:

$$F(s)^t = \frac{2H(s)^t B(s)^t}{H(s)^t + B(s)^t}, \forall t \in td \tag{8}$$

where $H(s)^t$ and $B(s)^t$ represents the entropy and expected balance for threshold t. We evaluate $F(s)^t$ for all values of threshold $t \in td$ and output that value of threshold t say $maxt$ for which $F(s)^t$ is maximum. The number of clusters (k) is obtained from the vector $SizeC_{maxt}$ and is equal to the number of terms in this vector. Algorithm 2 summarizes the proposed method for identifying the number of clusters (k) in a given large scale network.

Algorithm 2: Algorithm to automatically identify k communities

Data: $E = [e_1, e_2, \ldots, e_{N_{valid}}]$
Result: The number of clusters k in the given large scale network.

Construct an affinity matrix A using the projection vectors $e_i \in P$ and the similarity function $CosDist()$.
Set $td = [0.1, 0.2, \ldots, 1]$.
for *each* $t \in td$ **do**
    Save A in a temporary variable B i.e. $B := A$.
    Initialize the $SizeC_t$ as an empty vector.
    while B *is not an empty matrix* **do**
        Locate the projection of validation node e_i for which the number of nodes with $CosDist(e_i, e_j) \leq t$ is maximum.
        Count the number of these nodes and locate the indices of these nodes.
        Append the count of these nodes to the vector $SizeC_t$.
        Remove rows and columns corresponding to these indices.
    end
    Calculate the entropy $H(s)^t$ from $SizeC_t$.
    Calculate the expected balance $B(s)^t$ from $SizeC_t$.
    Calculate the F-measure $F(s)^t$ using $H(s)^t$ and $B(s)^t$.
end
Obtain the threshold corresponding to which the F-measure is maximum as $maxt$.
Estimate k as the number of terms in the vector $SizeC_{maxt}$.

IV. Experiments

We conducted experiments on several synthetic and real world networks. The synthetic networks are generated from the software proposed in [2] with varying the size of the network and the mixing parameter μ. The mixing parameter contains information about the extent of overlap between the communities in the large scale network.

A. Experiments on Synthetic networks

Figures 2 and 3 show the result of our proposed approach for identifying the number of clusters k on synthetic networks made of $5,000$and $50,000$ nodes respectively. We show the results for two different mixing parameters $\mu = 0.1$ and $\mu = 0.5$. For smaller μ the extent of overlap is less in comparison to higher value of μ and detecting communities is less difficult.

From Figures 2a, 2b, 3a and 3b, we show that the proposed method identifies exactly or close to the exact number of clusters present in the network even when the extent of overlap is high. The adjusted rand index (ARI) [26] values are also quite high suggesting that the clusters obtained via KSC are meaningful. We also observe that the threshold corresponding to which we obtain the maximum F-measure is related to the mixing parameter μ. For the smaller mixing parameter (0.1) the threshold value t for which the F-measure is maximum is less than or equal to the threshold value corresponding to higher mixing parameter 0.5 as observed from Figures 2c, 2d, 3c and 3d.

B. Experiments on Real World Datasets

We conducted experiments on several real-world large scale networks ranging from trust networks (PGPnet), collaboration networks (Cond-mat), citation networks (HepPh), communication networks (Enron), social networks (Epinion), web graphs (Youtube), actor networks (Imdb-actor) to road networks (roadCA). Details about these networks are available at http://snap.stanford.edu/data/index.html. Table I provides a few key statistics of each dataset.

Dataset	**Vertices**	**Edges**	**CCF**
PGPnet	10,876	39,994	0.008
Cond-mat	23,133	186,936	0.6334
HepPh	34,401	421,578	0.1457
Enron	36,692	367,662	0.497
Epinions	75,879	508,837	0.2283
Imdb-actor	383,640	1,342,595	0.453
Youtube	1,134,890	2,987,624	0.173
roadCA	1,965,206	5,533,214	0.0464

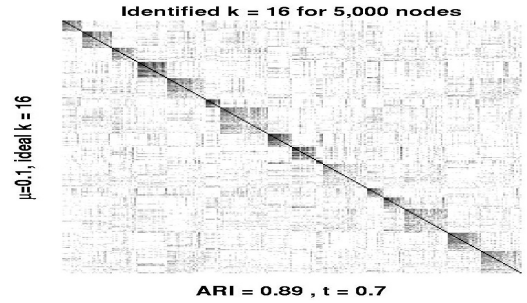
TABLE I: Vertices (V), Edges (E) and Clustering Coefficients (CCF) for each dataset

We use the entire large scale network as test set in our experiments. We use the clusters obtained by Louvain [8], Infomap [6] and Bigclam [27] methods as groundtruth for these networks. We compare the clusters obtained via the Original KSC method w.r.t. these groundtruth clusters. Then we modify the Original KSC method by automatically determining the number of clusters (k). We combine our proposed approach with the original out-of-sample extension property of the KSC model to obtain the Self-Tuned KSC (ST-KSC) method.

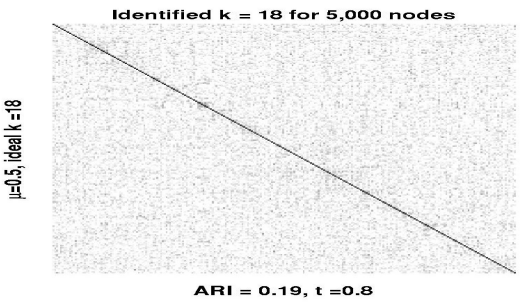
In order to perform a comprehensive evaluation, once we automatically obtain the value of k, we perform k-means on the projections in the eigenspace for the validation set to obtain the centroids. We obtain the projections of the test set in the eigenspace and assign the cluster membership corresponding to the centroid to which it is the closest. This method is referred as Self-Tuned k-means KSC (ST k-means KSC). We then compare the cluster memberships obtained via ST-KSC and ST k-means KSC w.r.t. the community affiliations via Louvain, Infomap and Bigclam method respectively.

For evaluating the cluster memberships we use three information theoretic measures: mutual information (MI), variation of information (VI) and adjusted rand index (ARI). These are standard cluster quality evaluation metrics given two list of cluster memberships and are described in detail in [28].

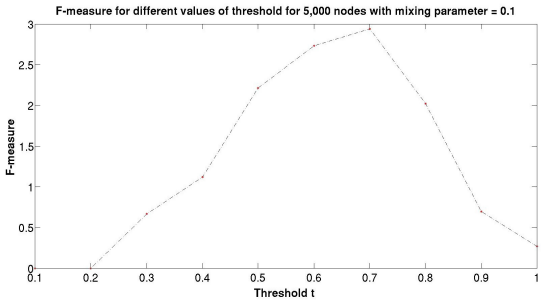
Table II evaluates the clusters obtained via the original KSC method, the proposed ST-KSC method and ST k-means KSC methods w.r.t. the clusters obtained by Louvain, Infomap and Bigclam methods. Since the partition obtained by methods like Louvain and Infomap method are not unique, we perform 10 randomizations and report the mean VI, ARI and MI values. Figure 4a and 4b shows the variations for these evaluation metrics in case of Enron and roadCA networks respectively.



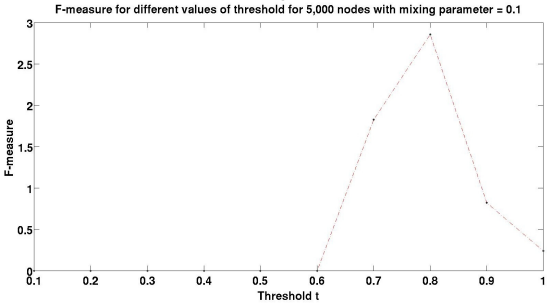
(a) A w.r.t. groundtruth



(b) A w.r.t. groundtruth

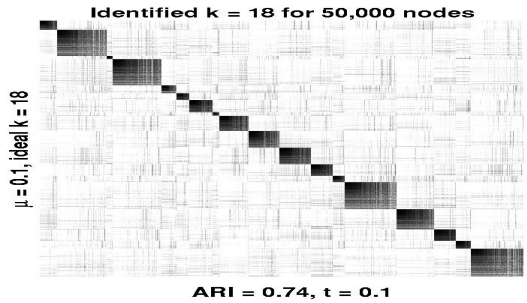


(c) F-measure vs Threshold

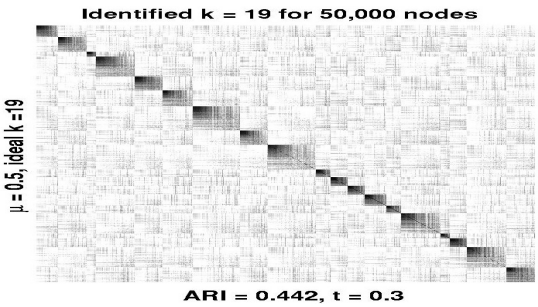


(d) F-measure vs Threshold

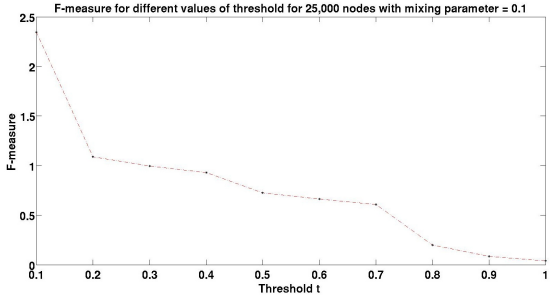
Fig. 2: Self-Tuned KSC on a synthetic network of $5,000$ nodes for mixing parameter values 0.1 and 0.5



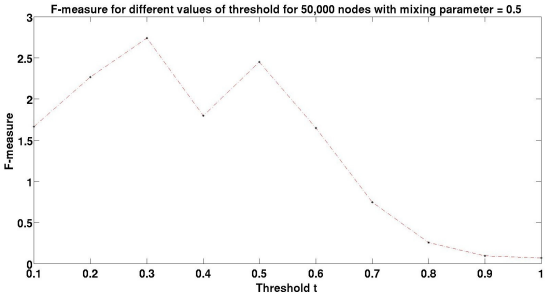
(a) A w.r.t. groundtruth



(b) A w.r.t. groundtruth



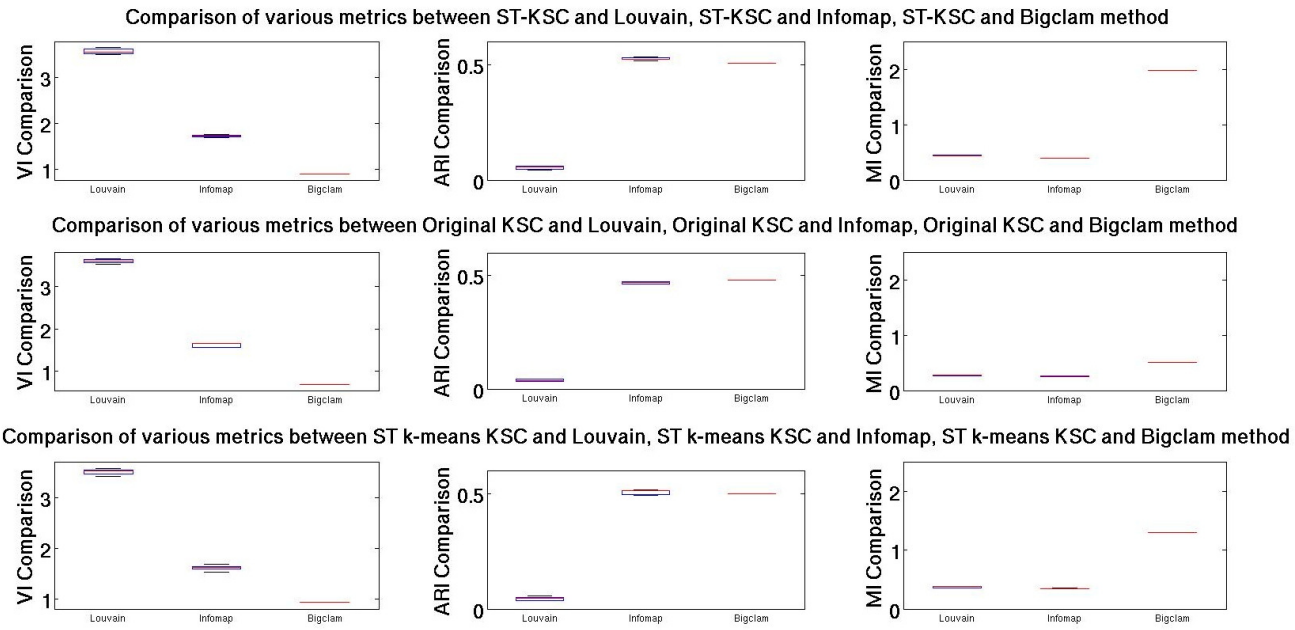
(c) F-measure vs Threshold



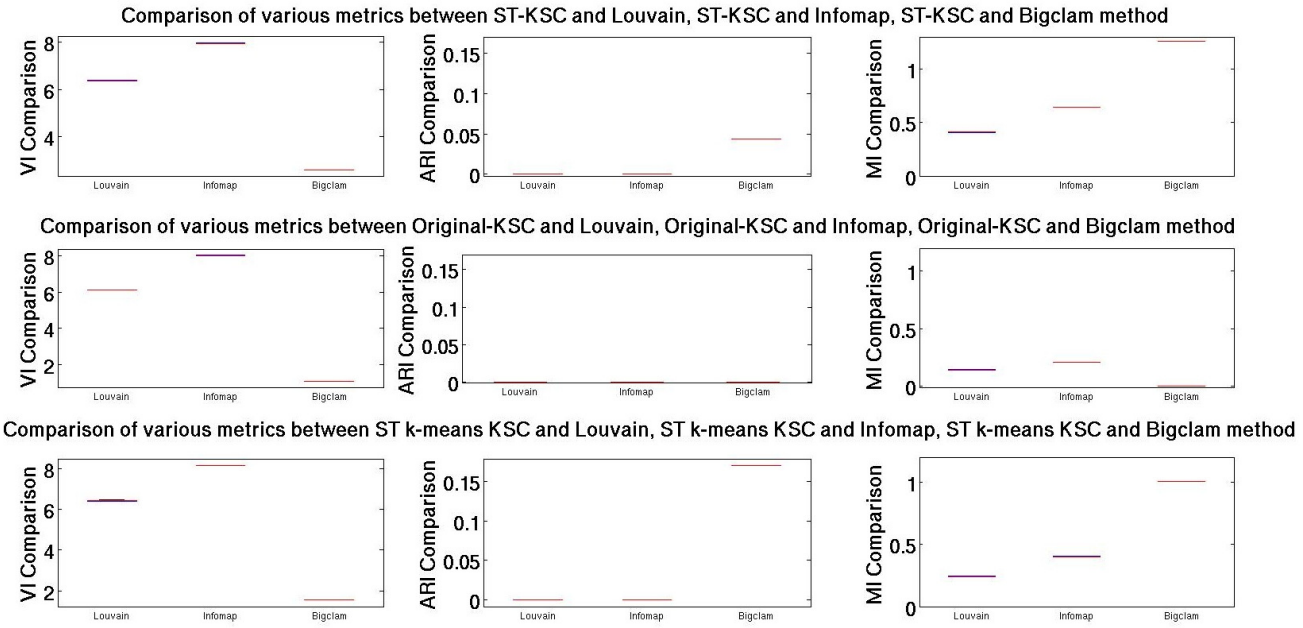
(d) F-measure vs Threshold

Fig. 3: Self-Tuned KSC on a synthetic network of $50,000$ nodes for mixing parameter values 0.1 and 0.5

From Table II we observe that the absolute values of ARI are quite small for most networks as it is heavily dependent on the number of clusters. For methods like Louvain and Infomap method, the number of clusters found are much more than that for KSC methods which lead to smaller ARI. From Table II we observe that clusters obtained by the Original KSC method generally have better variation of information (VI) corresponding to the clusters obtained via Louvain, Infomap and Bigclam methods. This is because the Original KSC method using the BAF evaluation metric is biased to produce small number of clusters as observed from Table II. However, due to this bias it results in poor ARI and MI values for most of the real world large scale networks. The clusters obtained by the two self-tuned methods have better ARI and MI values. The ST-KSC method results in better quality clusters for PGPnet, Enron, Imdb-actor and roadCA networks when the



(a) For Enron network



(b) For roadCA network

Fig. 4: Variation of evaluation metrics (VI, ARI and MI) for various methods for Enron 4a and roadCA 4b network. It shows that ST-KSC method gives best results for Enron network and ST k-means KSC method provides best results for roadCA network.

clusters are evaluated w.r.t. the clusters obtained by Louvain, Infomap and Bigclam methods. The evaluation metrics for which ST-KSC method performs better is ARI and MI. The ST k-means KSC method results in better quality clusters for Cond-mat, HepPh and Youtube networks w.r.t to the quality metrics ARI and MI.

Figure 4 showcases the results of the various quality metrics for the Enron and roadCA networks. Figure 4a shows that the clusters obtained by all the KSC methods are dissimilar from those obtained by Louvain method because of the high VI, low ARI and low MI values. However, the clusters obtained by the KSC methods are more similar to the ones obtained by Infomap and Bigclam methods. We observe this from 4a due to the low VI, high ARI and better MI values. In case of the Bigclam method, the clusters obtained are highly similar to that obtained by all the KSC methods. The clusters produced by the ST-KSC method in particular with VI (0.91), ARI (0.51) and MI (1.97) is the most similar to the clusters produced by

		Original KSC				ST-KSC				ST k-means KSC			
		k	VI	ARI	MI	k	VI	ARI	MI	k	VI	ARI	MI
PGPnet	Louvain	4	4.022	0.004	0.124	30	**3.94**	**0.073**	**1.102**	30	4.08	0.041	0.87
	Infomap	4	3.583	0.006	0.1	30	**3.7**	**0.1**	**0.98**	30	3.77	0.063	0.78
	Bigclam	4	**1.57**	0.011	0.029	30	3.4	**0.12**	**1.026**	30	3.462	0.07	0.78
Cond-mat	Louvain	3	4.021	0.01	0.225	52	**3.91**	**0.011**	0.324	52	4.5	0.04	**0.68**
	Infomap	3	3.132	0.05	0.222	52	**3.1**	0.051	0.32	52	3.63	**0.224**	**0.68**
	Bigclam	3	**0.5**	**0.49**	0.562	52	1.47	0.155	**2.49**	52	1.22	0.331	1.99
HepPh	Louvain	3	**3.22**	0.028	0.093	76	5.7395	0.014	0.216	76	5.23	**0.09**	**0.64**
	Infomap	3	**5.72**	0.006	0.166	76	7.224	0.014	0.78	76	6.3	**0.1**	**1.4**
	Bigclam	3	**0.95**	0.0004	0.0005	76	6.44	0.001	0.51	76	6.25	**0.09**	**0.8**
Enron	Louvain	3	3.6	0.038	0.274	16	**3.57**	**0.053**	**0.442**	16	3.51	0.0441	0.37
	Infomap	3	**1.6**	0.467	0.261	16	1.73	**0.53**	**0.4**	16	1.62	0.51	0.344
	Bigclam	3	**0.7**	0.48	0.51	16	0.91	**0.51**	**1.97**	16	0.94	0.50	1.3
Epinion	Louvain	3	3.1	0.0006	0.005	24	**3.08**	0.001	0.015	24	3.451	**0.03**	**0.06**
	Infomap	3	4.163	0.017	0.007	24	**4.16**	**0.031**	0.013	24	4.47	0.014	**0.1**
	Bigclam	3	**0.11**	**0.1**	0.007	24	0.84	0.02	**0.02**	24	0.74	0.014	0.016
Imdb-actor	Louvain	3	**3.05**	0.0005	0.006	59	4.4	**0.114**	**1.09**	59	3.84	0.093	0.673
	Infomap	3	**2.7**	0.0006	0.004	59	4.2	**0.11**	**0.99**	59	3.58	0.08	0.63
	Bigclam	3	**1.1**	0.0	0.0	59	3.0405	**0.025**	**1.07**	59	3.14	0.02	0.83
Youtube	Louvain	3	**3.64**	0.0003	0.004	21	4.1	0.016	0.047	21	4.55	**0.017**	**0.096**
	Infomap	3	**3.71**	0.0003	0.0035	21	4.16	0.014	0.03	21	4.7	**0.016**	**0.07**
	Bigclam	3	**0.056**	0.0	2.6e-06	21	1.6	0.035	0.06	21	1.25	**0.09**	**0.1**
roadCA	Louvain	3	**6.12**	0.0004	0.141	39	6.38	0.0005	**0.41**	39	6.42	**0.0006**	0.242
	Infomap	3	8.0	7.6e-05	0.21	39	**7.9**	0.0001	**0.64**	39	8.14	**0.00011**	0.40206
	Bigclam	3	**1.075**	0.0	0.0	39	2.62	0.044	**1.26**	39	1.56	**0.17**	1.01

TABLE II: Evaluation of the cluster memberships for Original KSC, ST-KSC and k-means KSC for several real world networks w.r.t. Louvain, Infomap and Bigclam methods. The highlighted numbers represent the best results. Original KSC method performs better w.r.t. VI while the ST-KSC and ST k-means KSC methods perform well w.r.t. ARI and MI.

Bigclam method.

From Figure 4b we observe that the clusters obtained by all the KSC methods have high VI and low ARI values in comparison to the clusters obtained by Louvain and Infomap method. This is because the mean number of clusters obtained by Louvain and Infomap method are $3,000$ and $65,807$ respectively over 10 randomizations. This is much larger in comparison to the number of clusters obtained by Original KSC (3) and 39 obtained by the proposed approach. However, the Bigclam method identifies 40 clusters. The clusters obtained by Bigclam method have best result w.r.t. VI with Original KSC method (1.075), w.r.t. ARI with ST k-means KSC (0.17) and w.r.t. MI with ST-KSC (1.26).

V. Conclusion

In this paper we proposed an approach to make the kernel spectral clustering method free of parameters for large scale networks. We used the normalized linear kernel for large scale networks and devised an approach to automatically identify the number of clusters k in the given network. For achieving this, we exploit the projections of the validation nodes in the eigenspace by creating an affinity matrix which had block-diagonal structure. We used the concepts of entropy and balance to identify these block-diagonals and obtain the number of clusters. We compared the resulting KSC methods with large scale community detection methods like Louvain, Infomap and Bigclam methods. In future work, we plan to evaluate the clusters obtained by the KSC methods using other likelihood based techniques like BIC and AIC.

Acknowledgements

This work was supported by Research Council KUL, ERC AdG A-DATADRIVE-B, GOA/10/09MaNet, CoE EF/05/006, FWO G.0588.09, G.0377.12, SBO POM, IUAP P6/04 DYSCO, COST intelliCIS.

References

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

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

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

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

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

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

[7] Langone, R., Alzate, C., Suykens, J.A.K. Kernel spectral clustering for community detection in complex networks. *In IEEE WCCI/IJCNN*, **2012**, 2596-2603.

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

[9] Ng, A.Y., Jordan, M.I., Weiss, Y. 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, **2002**; pp. 849-856.

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

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

[12] Shi, J., Malik, J. Normalized cuts and image segmentation. *IEEE TPAMI*, **2000**, *22(8)* , 888-905.

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

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

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

[16] Newman, M.E.J. Modularity and community structure in networks. *PNAS*, **2006**, *103(23)*, 8577-8582.

[17] Maiya, A., Berger-Wolf, T. Sampling community structure. *WWW*, **2010**, 701-710.

[18] Mall, R., Langone, R., Suykens, J.A.K. FURS: Fast and Unique Representative Subset selection for large scale community structure, *Internal Report 13-22, ESAT-SISTA, K.U.Leuven*, 2013.

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

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

[21] Leskovec, J., Faloutsos, C. Sampling from large graphs. *KDD*, **2006**.

[22] Langone, R., Alzate, C., Suykens, J.A.K. Kernel Spectral Clustering with Memory Effect. *Physica A: Statistical Mechanics and its Applications*, **2012**, 392(10), 2588-2606.

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

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

[25] Muflikhah, L. Document Clustering using Concept Space and Cosine Similarity Measurement. *ICCTD*, **2009**, 58-62.

[26] Hubert, L., Arabie, P. Comparing partitions. *Journal of Classification*, **1985**, 2, 193-218.

[27] Yang, J., Leskovec, J. Overlapping community detection at scale: a non-negative matrix factorization approach. *In Proceedings of WSDM*, **2013**, 587-596.

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