ABSTRACT

Learning Circles in Social Networks

Debopriya Ghosh, M.S.

Thesis Chairperson: Randal L.Vaughn, Ph.D.

Social networks are ubiquitous. One of the main organizing principles in these real world networks is that of network communities, where sets of nodes organize into densely inked clusters. Identifying such close-knit clusters is crucial for the understanding of the structure as well as the function of these real world networks. We implement an efficient variation of Kernel Spectral Clustering to infer the community affiliation by taking a well represented subgraph of the parent network along with a new notion of cluster mining on feature space to harness the vast amount of rich information stored in users' profile. The proposed method is memory and computationally more efficient than prevalent state-of-art methods. We empirically evaluate our approach against several real world datasets like Facebook, Twitter and Google+ and demonstrate its effectiveness in detecting community affiliations in sparse networks.

Learning Circles in Social Networks

by

Debopriya Ghosh, B.Tech.

A Thesis

Approved by the Department of Information Systems

Timothy R. Kayworth, Ph.D., Chairperson

Submitted to the Graduate Faculty of
Baylor University in Partial Fulfillment of the
Requirements for the Degree of

Master of Science in Information Systems

Approved by the Thesis Committee

Randal L.Vaughn, Ph.D., Chairperson

Jonathan K.Trower, Ph.D.

James D. Stamey, Ph.D.

Accepted by the Graduate School
May 2015

J. Larry Lyon, Ph.D., Dean

Page bearing signatures is kept on file in the Graduate School.

TABLE OF CONTENTS

LIST OF FIGURES vi

LIST OF TABLES vii

ACKNOWLEDGMENTS viii

DEDICATION x

1 Introduction 1

1.1 Network Communities 1

1.2 Communities in Social Networks 1

1.3 Hard Problem to Solve 2

1.4 Scope of the Thesis 2

1.5 Organization of the Thesis 3

2 Related Work 4

3 Kernel Spectral Clustering on Representative Subset 5

3.1 Background 5

3.2 Selection of Representative Subset 5

3.3 Model Description 6

3.3.1 Primal and Dual Formulation 6

3.3.2 Encoding and Decoding 8

3.4 Model Selection 9

4 Feature Space Cluster Mining 13

4.1 Overview 13

4.2 Algorithmic Outline 13
4.3 Model Evaluation 14

5 Experimental Setup and Results 15
5.1 Dataset Description 15
5.2 Feature Construction 16
5.3 FURS on these Dataset 17
5.4 Results and Analysis 17
5.4.1 Evaluation Metrics 17

6 Conclusions and Future Work 20

APPENDICES 22

APPENDIX A PseudoCode 23
A.1 Fast and Unique Representative Subset Computation 23
A.2 Kernel Spectral Clustering 26
A.3 KSC Codebook Computation 30
A.4 Determining Cluster Membership 32
A.5 Feature Space Clustering 33

APPENDIX B Notations 34

BIBLIOGRAPHY 35

LIST OF FIGURES

3.1 *BAF* for different values of k for egonet-698 11

3.2 *BAF* for different values of k for egonet-1912 12

5.1 Feature Construction 16

LIST OF TABLES

5.1 Dataset Attributes 17

5.2 F-score on various egonets 18

ACKNOWLEDGMENTS

I thank my advisor, Professor Randal L. Vaughn, for accepting me as his student and for the unwavering guidance and support. His breadth and depth of knowledge is inspiring, and working with him has been a pleasure. He gave me the freedom to decide what research problems I wanted to work on but he was always there to guide me when I needed it. He always answered my emails where I asked various questions ranging from those on scientific research in general, and on publication of scientific ideas and results, to specific ones pertaining to my research.

I thank the members of my reading committee Dr. Jonathan K. Trower, Dr. James Stamey for their comments on my work which helped to make it better.

I would like to thank Dr. Jonathan K. Trower for inspiring me to take up the thesis track and extend all possible support throughout. Without his influence this work would not have been feasible at all. I also like to thank all my other Professors who have constantly motivated and helped me in my endeavor.Special mention goes to Dr. Timothy R. Kayworth, Dr Gina Green, Dr. Patricia Norman for their invaluable contribution in making this a success.

I would like to thank Mary Reinhardt for all the help that she has provided me all through.

I must acknowledge all my professors at RCC Institute of Information Technology for providing me the basic tools to pursue graduate research. Specifically I would like to thank Professor N.C.Ghosh, Ritabari Roychowdhury and Professor Sujit Kumar Ghosh. Also I express my heartfelt gratitude towards two very important persons in my life, Dr. Asit Chakraborty and Dr. Nabanita Chattopadhay, for motivating and supporting me in every step of my life.

To my loving parents,"Ma and Baba": thank you so much or your constant encouragement and ridiculous amounts of patience. I am greatly indebted to my mom,

Aninda Ghosh for providing me the education and tools which have enabled me to come this far, and for continuing to nudge, cajole, and encourage me to complete my higher education. My dad, Pinaki Kumar Ghosh always demanded the best of me and encouraged me to pursue my goals in life. I also like to thank my in-laws, Reba Talukdar and Bimal Talukdar for supporting me and always loving me even in the worst times of life.

Last, but by no means the least, I would never have achieved all these without the love and inspiration of my husband Tanay Talukdar. Thanks for supporting through all the long hours I spent behind my thesis and keeping up with my tantrums.

I dedicate my thesis to my grandfather and grandma who are no longer with me but has been my inspiration and strength always.

In memory of

Late Shri. Khagendranath Ghosh and Late Smt Nirmala Ghosh,

my inspiration and strength in life.

CHAPTER ONE

Introduction

Social Networks are ubiquitous. With the advent of sites such as Facebook, Twitter, Google+ etc., social networks have reached major popularity and are rich source of data as users populate their sites with personal information. These networks enable people all across the globe to communicate with each other very easily. Online Social Networks allow users to follow streams of posts generated by their friends and acquaintances. Users' friends generate overwhelming volumes of information and to cope with "information overload" users need to organize their personal social networks. Communities serve as organizing principle of nodes in social networks and are created based on shared affiliation, role, activity, social circle, interest or function.

1.1 Network Communities

Network Communities can be defined as sets of nodes organized in to densely linked clusters. Communities in networks often overlap as nodes can belong to multiple clusters at once. Ground truth validation reveals that community overlaps are more densely connected than non overlapping parts, which is in sharp contrast to the conventional wisdom that community overlaps are more sparsely connected than the communities themselves. Communities in networks are thought of as groups of nodes that share a common functional property or role, and the goal of network detection is to identify such sets of functionally related nodes from the unlabeled network alone.

1.2 Communities in Social Networks

In the past few years, community mining has achieved more attention in sociology and data mining. Community detection is useful in social networks because it is more likely that nodes in one community have same properties. Some of the benefits of community detection are that they can be used for content filtering, for privacy

control and for sharing groups of users that others may wish to follow. Currently , users in Facebook, Google+ and Twitter identify their circles either manually or in a naive fashion by identifying friends sharing a common attribute. Neither of the approach is particularly satisfactory, the former is time consuming and does not update automatically as a user adds more friends, while latter fails to capture individual aspects of users' communities and may function poorly when profile information is missing or withheld.

1.3 Hard Problem to Solve

Problem with ego network data (ego and one-step alters, and all interconnections among them) is that they are usually fairly small, many of the methods that are commonly used in cluster detection are efforts to deal with "big data" e.g., spectral, Girvin-Newman, etc. Also we don't know whether two alters might be connected by way of a third party who is not connected to the ego. So the alter may be pretty close to one another sharing ties to the ego, and ties to unobserved others, but not directly to one another. The sparsity of these networks is also a major problem associated with real time networks.

1.4 Scope of the Thesis

This thesis focuses on the problem of automatically discovering users' social circles. In particular, given a single user with her personal social network, the goal is to identify her circles each of which is a subset of her friends. Circles are user specific as each user organize their personal network of friends independently of all others to whom she is not connected. Generally, there are two useful sources of data that help in detecting these communities. The first is the set of edges of the ego network [1]. It is expected that circles are formed by densely connected set of alters. Secondly, it is also expected that each of the circle is not only densely connected but its members

[1] Ego networks consist of a focal node ("ego") and the nodes to whom ego is directly connected to (these are called "alters") plus the ties, if any, among the alters.

also share common properties or traits. Thus we need to explicitly model different dimensions of user profiles along which each circle emerges.

In this work I have tried to implement Kernel Spectral Clustering on a well represented subgraph of the parent network, which had been earlier used to detect community affiliation in only large networks. I have used FURS (Fast-Unique-Representative-Subset) sampling procedure to obtain a subgraph that has higher coverage ration and greater degree distribution compared to other classical sampling methods. This approach gives rise to a powerful property of effectively inferring the community affiliation for out of sample extensions. Computations involving the original large kernel matrix is time consuming and memory inefficient. Selecting a smaller subgraph that preserves the overall community structure to construct the model. It makes use of the out-of-sample extension property for community membership of unseen nodes. I have also introduced a new clustering approach based on the similarity of the feature vector comprised of various profile attributes. It is an iterative approach that attempts to infer clusters in the network by detecting pattern similarity among the feature vectors of each data point (i.e. users' friends).

1.5 Organization of the Thesis

The outline of the thesis is as follows. Chapter 2 discusses related work in the area. Chapter 3, we present the approach of inferring communities based on the cosine similarity in the eigenspace between projected vectors of the nodes in the validation set.Chapter 4 introduces an iterative algorithm of detecting circles based on the Hamming distance of the feature vectors. In Chapter 5, we present the experimental results evaluating both the approaches for solving community detection problem. Having completed the study of the proposed methods, we try to analyze their performance and tread-offs in Chapter 6. Finally we conclude in Chapter 7 with brief note on future work.

CHAPTER TWO

Related Work

Topic modeling have been used to uncover "mixed memberships" of nodes to multiple groups and extensions allow entities to be attributed with text information. Classical algorithms tend to identify communities based on node features or graph structure but rarely use both in concert. Clustering techniques which models membership to multiple communities have also been used in recent past. Among the myriad variety of algorithms for community detection, the Kernel Spectral Clustering (KSC) method is related to the spectral methods. Spectral clustering methods are standard techniques for graph partitioning. The underlying model is based on eigen decomposition of Laplacian Matrix derived from the affinity matrix of the nodes in the community. The major drawback of these spectral clustering methods is the construction of the large affinity matrix (N X N), where N is the number of nodes in the network. As the size of the network increases, the $O(N^2)$ computation and storage of this affinity (N X N) matrix becomes in feasible. Recently spectral clustering formulation based on weighted kernel PCA with primal-dual framework has been proposed. However the model is still computationally and memory inefficient. Attempts had been made to implement spectral clustering on the Laplacian Matrix derived from exponential adjacency matrix. Conceptually, it is gives a better notion of the degree of connectedness of various nodes by smoothing out the inconsistencies in the adjacency matrix.

CHAPTER THREE

Kernel Spectral Clustering on Representative Subset

3.1 Background

Spectral clustering uses the information contained in the affinity matrix to detect structures in the given network. In case of data points the similarity between the points is measured with respect to the mutual distance (e.g. Euclidean, cosine, RBF distance) between the points. Thus the obtained similarity matrix can be considered as weighted graph where each point has a certain extent of similarity with other points in the dataset. In case of undirected graph A, where $A_{ij} = 1$ if there an edge between (v_i,v_j) else $A_{ij} = 0$. Therefore we could have directly applied the spectral clustering on the adjacency matrix but for the KSC method we need to build a graph over the network to represent the similarity between nodes in a kernel based framework.

3.2 Selection of Representative Subset

An inherent requirement of the KSC method is to generate a model on the training set. Since, KSC generates a N_{tr} X N_{tr} size kernel matrix where N_{tr} is the number of training nodes. If the size of N_{tr} becomes too large then the KSC procedure becomes infeasible. Thus we need to select a subset of nodes that is representative for the underlying community structure. The obtained subgraph allows to build a model on it during the training phase and provides a meaningful out-of-sample extension to nodes that are not present in the training set.

Sampling is fundamental to statistics and employed when there is a need to study a population and direct analysis of the entire population is feasible due to sheer size and inaccessibility. In these cases random samples are analyzed, and results are generalized to the population from which they are drawn. Similar approach can be

also applied to networks. Various state-of-the-art techniques for sampling large scale graphs include SlashBurn algorithm (Kang U. 2011), Snowball Expansion sampling (Maiya A. 2010), Metropolis (Raghvendra Mall 2013) and random sampling techniques. FURS (Raghvendra Mall 2013) is computationally less expensive and better preserves the inherent community structure in comparison to the above methods.
The motivation is to select a subset of nodes that approximately maximizes the coverage of the graph under the constraint that the selected nodes belong to different dense regions of the graph. Coverage is defined as the ratio of the number of unique nodes directly reachable from the selected subset to the total number of nodes in the graph. The idea is to first sort the nodes based on their degree in descending order during each iteration and pick the node with highest degree. Once such a node is selected, its immediate neighbors are deactivated (as they can be reached directly from this node) during that iteration and the node is placed in selected subset without affecting the graph topology. We then select the node with highest degree among the active nodes and the process is repeated until either all the nodes are deactivated or we reach the subset size. If all nodes are deactivated before we reach the desired subset size, a new iteration is started and the deactivated nodes are reactivated. FURS result in a subset of nodes that span most or all of the communities in the network.

3.3 Model Description

3.3.1 Primal and Dual Formulation

The KSC method is described by a primal-dual framework. The model is determined during training phase and the parameter of the model, *i.e.,* k (number of clusters), is estimated during validation stage. Finally the model is tested on the test data to provide community affiliation to the unseen nodes.

The training data comprises of the adjacency lists of all the vertices $v_i \in X_{tr}$, where X_{tr} represents set of nodes used to train the model. The cardinality of

the training set X_{tr}, is denoted by N_{tr}. Big data networks are generally stores into memory in sparse format. The adjacency lists can be stored efficiently in the memory as real world networks are highly sparse and there are very few connections for each node $v_i \in X_{tr}$. The maximum length of the adjacency list of a node can be N, when the node is connected to all other nodes in the network. During the test phase, the cluster memberships for each of the unseen nodes can be predicted by uploading the adjacency list of the test node in the memory and using the out-of-sample extension property of the model. Unlike other approaches it does not require the entire graph to be stored in the main memory and hence memory efficient. The time complexity of the model is given by $O(N_{tr}\ N)$, the time required to compute the similarity between the sparse adjacency lists of the nodes in the training set.

Given X_{tr} training nodes $D = \{x_i\}_i^{N_{tr}}$, $x_i \in R^N$ and $x_i \in X_{tr}$. Here x_i represents the adjacency list of the i^{th} training node and the number of nodes in the training set is N_{tr}. Given D and the number of communities k, the primal problem can be formulated as (Alzate C. 2010):

$$\begin{aligned} \min_{\omega^{(l)}, e^{(l)}, b_l} \tfrac{1}{2}\Sigma_{l=1}^{k-1}\omega^{(l)T}\omega^{(l)} - \tfrac{1}{2N}\Sigma_{l=1}^{k-1}\gamma_l\, \mathrm{e}^{(l)\,T}\, \mathrm{D}_\Omega^{-1}\, \mathrm{e}^{(l)} \\ \textit{such that } \mathrm{e}^{(l)} = \Phi\omega^{(l)} + b_l 1_{N_{tr}} l = 1, \ldots, \mathrm{k\text{-}1} \end{aligned} \tag{3.1}$$

where $\mathrm{e}^{(\mathrm{l})} = [\mathrm{e}_1^{(\mathrm{l})}, \ldots, \mathrm{e}_{\mathrm{k\text{-}1}}^{(\mathrm{l})}]^T$ are the projections onto the eigenspace, $\mathrm{l} = 1, \ldots, \mathrm{k\text{-}1}$ indicates the number of score variables required to encode k communities, $\mathrm{D}_\Omega^{-1} \in \mathrm{R}^{\mathrm{N_{tr}} \times \mathrm{N_{tr}}}$ is the inverse of the degree matrix associated with the kernel matrix Ω. Φ is the $\mathrm{N}_{tr} \times \mathrm{d}_h$ feature matrix, $\Phi = [\phi(\mathrm{x}_1)^T, \ldots, \mathrm{x}_{N_{tr}})^T]$ and $\gamma_l \in \mathrm{R}^+$ are the regularization constants. Note that $\mathrm{N}_{tr} \ll \mathrm{N}$ *i.e.,* the number of nodes in the training set, is much less than the total number of nodes in the big data 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 $\Omega_{ij} = \mathrm{K}(\mathrm{x_i}, \mathrm{x_j}) = \phi(\mathrm{x}_i)^T\phi(\mathrm{x}_j)$, is obtained by calculating the cosine similarity between the adjacency

lists of x_i and x_j. Therefore,$\Omega_{ij} = K(x_i, x_j) = \frac{x_i^T x_j}{|x_i||x_j|}$ which is a normalized function. The clustering can thus be represented by:

$$e^{(l)} = \omega^{(l)T}\phi(x_i) + b_l \; i = 1,.....,N_{tr} \tag{3.2}$$

where $\phi : R^N \rightarrow R^{d_h}$ is the mapping to a high-dimensional feature space d_h, b_l are the bias terms,$l = 1, \ldots,$ k-1. The projections $e_i^{(l)}$ represent latent variables of a set of *k-1* binary cluster indicators given by $sign(e_i^{(l)})$, which can be combined with the final groups using an encoding/decoding scheme. The dual problem corresponding to the above primal formulation is:

$$D_\Omega^{-1} M_D \Omega_\alpha^{(l)} = \lambda_l \alpha^{(l)} \tag{3.3}$$

where M_D is the centering matrix, which is defined as:

$$M_D = I_{N_{tr}} - (\frac{1}{1_{N_{tr}}^T D_\Omega^{-1} 1_{N_{tr}}})(1_{N_{tr}} 1_{N_{tr}}^T D_\Omega^{-1}) \tag{3.4}$$

The $\alpha^{(l)}$ are the dual variables and the kernel function $K : R_N \times R_N \rightarrow R$ plays the role of similarity function.

3.3.2 Encoding and Decoding

When the communities are non-overlapping we obtain k well separated clusters and the matrix $D^{-1} M_D \Omega$ has k-1 piecewise constant eigenvectors. The multiplicity of the largest eigen value (*i.e.*, 1) is k-1. In the eigenspace every cluster A_p, $p = 1, \ldots, k$ is a point represented with a unique codebook vector $c_p \in \{-1, 1\}^{k-1}$. The codebook $\boldsymbol{CB} = \{c_p\}_{p=1}^k$ is obtained by transforming the rows of the projected vector matrix obtained from the training data and mapping to binary values, *i.e.*,$[sign(e^{(l)}, \ldots, e^{(k-1)})]$. The codebook set $\boldsymbol{CB}$ is also obtained by selecting the top k most frequent codebook

vectors. Due to the centering matrix M_D, the eigenvectors have zero mean and the optimal threshold for binary mapping of the projected vector matrix is self-determined (equal to 0). Since the first eigenvector $\alpha^{(1)}$ already provides a binary clustering, the number of score variables needed to encode k clusters is *k-1*. The decoding scheme involves comparing the cluster indicators obtained in the validation/test stage with the codebook and selecting the nearest codebook vector based on Hamming distance. This approach is used in out-of-sample extensions also. The proposed 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 mapping the score variables to binary values as follows:

$$\text{sign}(e_{test}^{(l)}) = \text{sign}(\Omega_{test}\alpha^{(l)} + b_l\, 1_{Ntest}) \tag{3.5}$$

where $l = 1, \ldots,$ k-1. Ω_{test} is the $N_{test} \times N_{tr}$ kernel matrix evaluated using test points with entries $\Omega_{test,ri} = K(x_r^t\, est, x_i), r = 1, \ldots, N_{tr}$. Here N_{tr} represents the number of nodes in the test set.

3.4 Model Selection

Model selection is a crucial step in KSC. In order to obtain cluster parameters for the model we use the concept of angular similarity (Raghvendra Mall 2013). Many a times, Modularity has been chosen as model selection criterion. However the validation matrix required for Modularity calculation can blow up as the size of the network increases and might be infeasible to store in memory. Therefore we opt to work with the sparse adjacency lists of the nodes in the validation set instead of creating a square validation matrix. Since we are using cosine similarity metric to estimate each element of the kernel matrix Ω, the model is free of a tuning parameter σ unlike the original formulation of KSC (Alzate C. 2010) when using Gaussian RBF

kernel. Thus the only parameter need to be determined is the number of clusters k in the network.

We use Balanced Angular Fit proposed by the authors in (Raghvendra Mall 2013). The criteria exploits the projections of training and validation nodes in the eigenspace.For a given value of k $\rangle$ 2, the cluster membership of the nodes is estimated based on the codebook ***CB***. Each training node is assigned to the cluster corresponding to the codebook vector for which its Hamming distance is minimum. Thus for a given value of k $\rangle$ 2, a clustering $\Delta = \{ \mathrm{P_1},...,\mathrm{P_k}\}$ is obtained, where P_i contains the set of training nodes belonging to the i^{th} cluster. Cluster mean μ_i for each cluster is calculated with respect to projections of training nodes belonging to the i^{th} cluster in the eigenspace.

Once the cluster means are obtained for all the clusters, the projections of validation nodes in the eigenspace are used. The idea is to calculate the angular similarity between the projection of each validation node and each of the cluster means. The cluster mean that makes the least angle with the projection of the validation node is the closest for that node. Thus, the cluster corresponding to that cluster mean is assigned to the given validation node. For each validation node ($valid_i$), we want to obtain $max_{\mathrm{j}} cos(\theta_{\mathrm{j,valid_i}})$, where

$$\cos(\theta_{j,\mathrm{valid}_i}) = \frac{\mu_j \, \mathrm{e}_{valid_i}}{\|\mu_j\| \|e_{valid_i}\|} \; j = 1,.....,k \tag{3.6}$$

The closer the projection of the validation node is to that of a given cluster mean, the smaller the angle it has with it and larger the cosine value of that angle. Therefore, for each validation node the cluster to which it is assigned has the maximum cosine similarity value. The cosine similarity value for each validation node is maintained in a dictionary *MaxSim*. The dictionary is maintained as $MaxSim\ (valid_i) = \cos(\theta_{j,valid_i})$, where $\cos(\theta_{j,valid_i}) = max_j \cos(\theta_{j,valid_i})$, $j = 1,...,k$ is the maximum cosine similarity value for the validation node ($valid_i$). The obtained clustering is $\Delta_{valid} = \{\mathrm{Q_1},...,\mathrm{Q_k}\}$,

where Q_i contains the set of validation nodes belong to cluster *i*. *Balanced Angular Fit(BAF)* can be defined as:

$$\mathrm{BAF(k)} = \Sigma_{i=1}^{k} \Sigma_{valid_j \in Q_i} \frac{1}{k} \frac{\mathrm{MaxSim(valid_j)}}{\|Q_i\|} \tag{3.7}$$

The *BAF* simply sums up the cosine similarity values of all validation nodes belonging to each cluster divided by the cardinality of the cluster. It then divides the overall value by the total number of clusters k in the network. The range of values *BAF* can take is [-1,1] since the maximum similarity value between a validation node and a cluster mean is 1(i.e., the angle between them is zero). Also that we divide the sum of the cosine similarity values of all the validation node in a cluster by the cardinality of that cluster, the metric is inherently balanced and the fraction can not exceed one. The process is repeated for each cluster, and in order to normalize the metric, it is divided by the number of clusters in the network. Therefore, in the worst case scenario, when all validation nodes are wrongly assigned to clusters, the angle it makes with the respective cluster mean is π and the *BAF* end up being -1. *Figure 3.1 and 3.2* shows the model selection or identification of the number of clusters k for the different egonets.

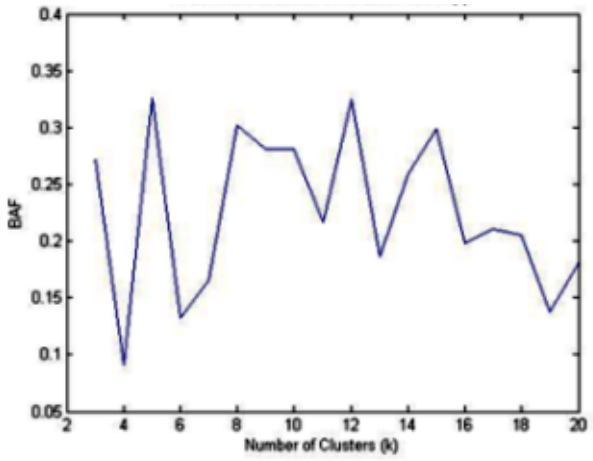

Figure 3.1. *BAF* for different values of k for egonet-698

It has been observed that the BAF values are high for small number of clusters. Thus, we make an exception and select the peaks close to each other.

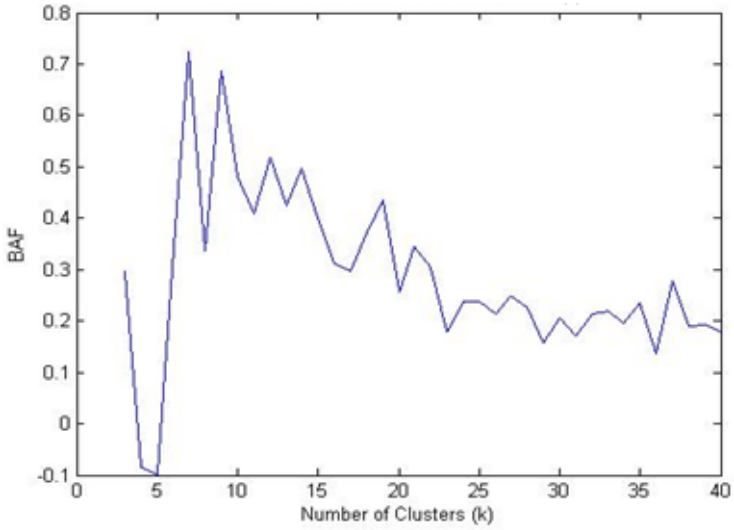

Figure 3.2. BAF for different values of k for egonet-1912

The dominant peaks for each dataset with the specified range of k is plotted on the graph. Currently, the peak selection is based on a adhoc procedure. We sort the BAF values and select the maximum and restrict from selecting peaks in the immediate neighborhood as we might miss out the hierarchical structure in that case.

CHAPTER FOUR

Feature Space Cluster Mining

4.1 Overview

Having performed Kernel Spectral Clustering on the similarity matrix computed on the basis of cosine similarity of the adjacency lists of the pair of nodes, attempt has been made to harness the power of the feature matrix composed on the basis of the various attribute values of user profiles. The feature matrix is basically a binary matrix F (N X F_r) where N is the number of nodes in the network and F_r is the dimension of the feature vector. If $F_{ik} = F_{jk} = 1$,it indicates that the nodes i and j has same values for the attribute k. The proposed method computes the similarity between the feature vectors in a iterative manner based on the Hamming distance between a pair of nodes. The distance matrix reveals an interesting pattern, nodes with similar attribute value that tends to form a cluster are equidistant from the basis vector.

4.2 Algorithmic Outline

The algorithm was adopted from the "Multi-K Algorithm" for semantic compression and pattern extraction with fascicles (Madar 1999). Given a relation we first sort the relation based on a random column. Then we choose the first row to be the basis vector and iterate over the remaining records of the relation and compute the hamming distance between the basis vector and each of the remaining feature vectors. The distance vector returned as the result shows an interesting property that nodes, represented by the rows of the relation tend to form a cluster on account of the similarity of their feature vectors and are thus approximately equidistant from the basis vector. The number of unique distance measure obtained indicates the number

of clusters in the network and the frequency of each of the distance measure accounts for the cardinality of each cluster.

4.3 Model Evaluation

When compared to ground-truth communities the above method performs satisfactorily. But if the number of nodes in the network is large and also the dimension of feature vector is considerably large then the performance of the method degrades. Given a relation that does not fit in main memory, each record sampled would require one disk access. To minimize the amount of I/O activity performed, a commonly used technique "block sampling" can be used, where an entire disk page is read into memory and all records on the page are used. We propose to adopt this approach as well, but there is an important difference. In traditional sampling, possible correlation between co-related records can be compensated by increasing the sample size. In our case, the specific ordering of records in the sample is of critical importance. Increasing the sample size would not be a help at all. For example, suppose we have a relation which comprises of the players' statistics of National Hockey League. For each player, his record describes the position he played, the number of points he scored, the number of minutes he was on ice and was in the penalty box. Now if we sort the relation by number of minutes and store on disk, when we consider successive records, they will very likely have minutes played as a compact attribute to the potential determinant of other possible compact attributes.

To address these concerns, we read into memory as many randomly sampled blocks of the relation as our buffer space permits. Now we can work purely with the sample of the relation in memory.

CHAPTER FIVE

Experimental Setup and Results

Experiments were conducted on several large real world datasets obtained from platforms like Facebook, Twitter, and Google+.

5.1 Dataset Description

This dataset consists of "circles" (or "friends lists") from Facebook. Facebook data was collected from survey participants using the Facebook App. The dataset includes node features (profiles), circles, and ego networks.

Facebook data has been anonymized by replacing the Facebook-internal ids for each user with a new value. Also, while feature vectors from this dataset have been provided, the interpretation of those features has been obscured. For instance, where the original dataset may have contained a feature "political=Democratic Party", the new data would simply contain "political = anonymized feature 1". Thus, using the anonymized data it is possible to determine whether two users have the same political affiliations, but not what their individual political affiliations represent. Data is also available from Google+ and Twitter.

The dataset comprised of the following files:

(1) nodeId.edges : The edges in the ego network for the node "nodeId". Edges are undirected for facebook, and directed (a follows b) for twitter and gplus. The "ego" node does not appear, but it is assumed that they follow every node id that appears in this file.

(2) nodeId.circles : The set of circles for the ego node. Each line contains one circle, consisting of a series of node ids. The first entry in each line is the name of the circle.

(3) nodeId.feat : The features for each of the nodes that appears in the edge file.

(4) nodeId.egofeat : The features for the ego user.

(5) nodeId.featnames : The names of each of the feature dimensions. Features are 1 if the user has this property in their profile, and 0 otherwise. This file has been anonymized for facebook users, since the names of the features would reveal private data.

5.2 Feature Construction

Profiles are tree structured, and the features were constructed by comparing paths in those trees by the authors in. From Facebook they collected data from 26 categories, including hometowns, birthdays, colleagues, political affiliations, etc. A difference vector was described to encode relationships between two profiles. They have identified ways for representing the compatibility between different aspects of profiles for two users. Examples of trees for two users x and y are shown in *Figure 5.1.* Two schemes for constructing feature vectors from the profiles are also depicted in the figure.

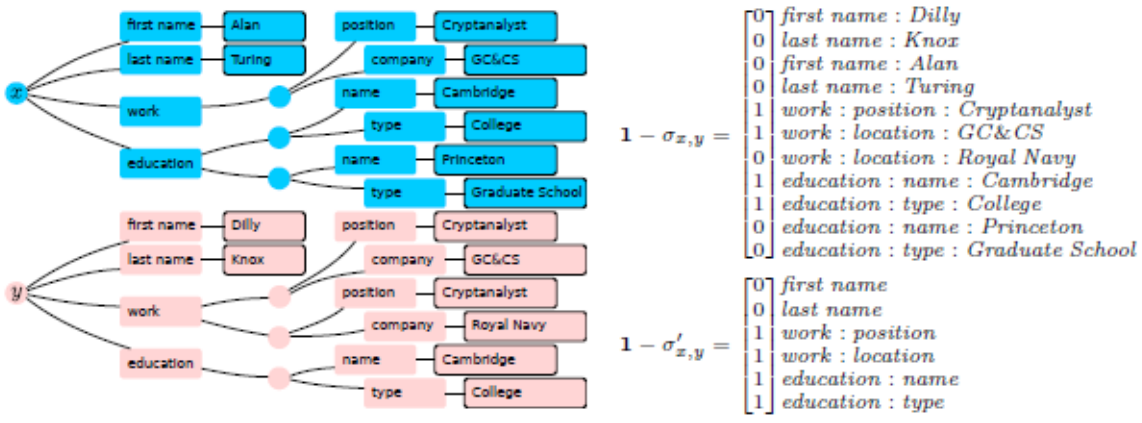

Figure 5.1. Feature Construction

Firstly, binary indicators were constructed to measure the difference between leaves in the two trees, *e.g.*, work→position→Cryptanalyst appears in both trees. Secondly, Summation over the leaf nodes in the first scheme is carried out, maintaining the fact that the two users worked at the same institution, but discarding the identity of the institution.

5.3 FURS on these Dataset

We first apply the FURS technique on these datasets to subset a training set and validation set of nodes. The first step that we follow is to first select the training set using FURS, then these set of nodes are removed from the graph. Then another iteration of FURS is run to select the the validation set of nodes. *Table 5.1* depicts the values of various attributes of these dataset.

Table 5.1. Various characteristic attributes of each datasets

Dataset	Nodes	Edges	Clustering Coefficient
Facebook	4039	88234	0.6055
Twitter	81306	1768149	0.5653
Google Plus	107614	13673453	0.4901

The maximum value of clustering coefficient can be 1. The higher the value, better is the quality of subset selected.

5.4 Results and Analysis

5.4.1 Evaluation Metrics

Although our method is unsupervised, we can evaluate it on ground-truth data by examining the maximum-likelihood assignments of the latent circles $C = \{C_1 \dots C_k\}$ after convergence. The goal is that for a properly regularized model, the latent variables will assign closely with the human labeled ground-truth circles.

To measure the alignment between a predicted circle C and a ground-truth circle C', we compute the number of users that need to be removed from C is the type I error (false positives) and the numbers of users that need to be added to C that are in C' is the *type II* error (false negatives). The union of these two sets is the symmetric difference between C and C'. So the error considered by the evaluation metric between these two circles is just the size of the symmetric difference. But, since we don't know which predicted circle should match with which ground truth circle, we need to find the assignment of circles that will minimize the total error. A naive search over all possible assignments goes as O($N!$) where N is the number of circles which is quickly intractable. The classic solution to this problem is the so-called Hungarian Algorithm which solves the minimization in O(N^3) time.

We describe a precise evaluation criteria we employ to asses the performance of the model. We compute a composite score which is the harmonic mean of the fraction of communities of the sample ($FRAC$)and the cost of optimal assignment of predicted clusters ($COST$). The fraction of communities in the sample is a normalized value ranging from 0 to 1 (higher the values are better). For, ease of illustration the cost is also converted to an accuracy score ranging from 0 to 1 by normalizing and subtracting from one (also resulting in higher value being better).

$$\text{Composite} = \frac{2*FRAC*PART}{FRAC+PART} \tag{5.1}$$

Table 5.2. F-score on various egonets.

EgoNet ID	KCS	Feature Clustering	Training Time	Testing time
698	0.57	0.92	2.66 seconds	0.21 seconds
3980	0.51	0.67	0.36 seconds	0.33 seconds
1912	0.89	0.62	12.85 seconds	23.64 seconds

In terms of absolute performance our model achieves F-scores of 0.65 on Facebook, 0.52 on Google+ and 0.46 on Twitter. The lower F-scores on Google+ and Twitter are explained by the fact that many circles have not been maintained since they were initially created whereas the Facebook data is complete,in the sense that survey participants manually labeled every circle in their ego-networks. Secondly, the 26 profile categories available from Facebook are more informative than the 6 categories from Google+ or the tweet based profiles built from Twitter. A more basic difference lies in the nature of the networks themselves: edges in Facebook encode mutual ties, whereas edges in Google+ and Twitter encode follower relationships, which changes the role that circle serve.

CHAPTER SIX

Conclusions and Future Work

While the problem of community detection has received a lot of attention in the past, most of the state-of-art approaches are based on the assumption that the entire network can fit in main memory. However, with the increasing amount of information, the size of the networks will only increase and big data networks may not fit in the main memory.

The KSC method employs the optimization based framework to construct the model, which has a very useful out-of-sample extension property. The model that is constructed should adhere to memory restrictions. FURS selection procedure is used to select a representative subgraph of the big data network on which the model can be built. It selects nodes from different dense regions of the graph while maximizing the coverage and preserves the inherent community structure of the big data network. In order to obtain the model parameters (*i.e.,* the number of clusters k in the network), we use a novel metric *Balanced Angular Fit* which works with a codebook CB and the projections of the validation set on the eigenspace to determine the ideal number of clusters k in the big data network. The metric is both memory and computationally efficient compared to well known Modularity metric. The out-of-sample extensions property allows inferring community affiliation for unseen nodes and allow to process large scale networks relatively easily.

The profile attributes are rich source of data that can depict pattern similarity based on the feature vector comprised of those individual attributes. Our proposed method has out performed many prevalent graph-partition and other classical clustering methods like K-means, hierarchical clustering etc.

Future work may focus on automatically determining the dominant peaks in BAF versus the number of clusters curve. Using the combination of node and edge

information simultaneously, by integrating individual models to harness the power of both network topology and profile information, performance can be significantly improved. Accounting for heterogeneity combined with homophily can outperform most present state-of-art methods.

APPENDICES

APPENDIX A

PseudoCode

A.1 Fast and Unique Representative Subset Computation

```
function [ S ] = FastAndUniqueRepresentativeSubset( L, M, A, N)
% Selects subset of nodes that approximately maximizes the coverage of the
% graph under the constraint that the selected nodes belong to different
% regions of the graph.
% PARAMETERS:
%   L = (V,f(V)), list of nodes with their corresponding degree.
%   M = Median degree of the graph?
%   A = Adjacency Matrix containing information about neighbors Nbr(Vi),
%   Vi    V
%   N = number of nodes in the network
% RESULT:
%   S,subset of nodes of the given graph G = (V,E) whose cardinality is Ns

% Author: Debopriya Ghosh ( debopriya_ghosh@baylor.edu )

 L_New = [size(L,1),2];

 for i = 1: size(L,1)
     if( L(i,2) > M )
         L_New(i,:) = L(i,:);
     end
 end

 L_New( ~any(L_New,2), : ) = [];
```

```
L_New = sortrows(L_New,-2);
Ns = .15* N;
if(Ns > size(L_New,1))
    Ns = size(L_New,1);
end
S = [];
L_Deactive = [size(L_New,1),2];
k = 1;

while(size(S) < Ns)?

    % Reactivation
    if( isempty(L_New))
    L_New = L_Deactive;
    L_New = sortrows(L_New,-2);

    end
    % Hub selection
    v1 = L_New(1,1); % pop out the highest degree node

    S(k) = v1;  % add to the output set
    k = k+1;
    Nb = find(A(v1,:)); % Neighboring nodes of v1
    L_New(1,:) = [];
    L_New = sortrows(L_New,-2);

    for j = 1: size(Nb,2)
       if(~ismember(Nb(1,j),L_Deactive(:,1)))
```

```
    L_Deactive = L(L(:,1)== Nb(1,j),:);


        end
     end

 end

end
```

```
function [model] = KernelSpectralClustering( V, X_train,k)
%   KERNELSPECTRALCLUSTERING Executes spectral clustering algorithm
%   Data: Given a graph G = (V,E), which might not necessarily be stored in
%   the memory.
%     Inputs: Xtrain: N x d matrix of training data
%             params: kernel parameters (e.g., sig2)
%             k:      number of desired clusters
%          (*) Xtest: Nt x d matrix of test data
%             mode:   0, train
%                     1,test
%   Result: The patitions of the graph G, i.e., divide graph into k
%   clusters.
%   Author: Debopriya Ghosh

% computing kernel matrix

 NORMROWS_THR = 5e-2; % Minimum norm allowed for the rows of the test
 N=size(X_train,2);

  % computing kernel matrix for train nodes
 model.Omegatrain = [ length(X_train), length(X_train)];

  for i = 1: length(X_train)
      x = V(X_train(i),:);

      for j = 1: length(X_train)
          y = V(X_train(j),:);
          model.Omegatrain (i,j) = dot(x,y)/(norm(x,2)*norm(y,2));
```

```
    end
  end

 model.Omegatrain (isnan(model.Omegatrain )) = 0;
 Omegatrain_sparse = sparse(model.Omegatrain );

% calculate degree matrix
 degs = sum(Omegatrain_sparse, 2);
 d=sum(model.Omegatrain)';
 model.dinv=1./d;

 D    = sparse(1:size(Omegatrain_sparse, 1), 1:size(Omegatrain_sparse, 2),
     degs);

% avoid dividing by zero
 degs(degs == 0) = eps;

% calculate inverse of D
 D = spdiags(1./degs, 0, size(D, 1), size(D, 2));

 % calculate centering matrix
 I = eye(length(X_train));
 l = ones(1,length(X_train));
 Md = I - ( 1/(l*D*l'))*(l*l'*D);

 %calculate the operator matrix
 L = D * Md* Omegatrain_sparse;

% eigen decomposition of kernel matrix
```

```
opts.disp=0;
[model.alpha,model.lambda] = eigs(L, k+4,'lm',opts);
model.alpha = real(model.alpha);
model.lambda = real(diag(model.lambda));

[temporary,a]=sort(model.lambda,'descend');

if (length(model.lambda)<k)
    fprintf('Cannot find more eigenvalues\n');
    return;
 end;

model.lambda = model.lambda(a(1:k-1));
model.alpha=model.alpha(:,a(1:k-1));

% calculating b
model.b =
    -(1/(sum(model.dinv)))*(model.dinv'*model.Omegatrain*model.alpha);

% Compute the score variables, codebook and cluster membership
% for training data;

  model.etrain=model.Omegatrain*model.alpha+repmat(model.b,[N 1]);

  [N,d]=size(model.etrain);
  k=d+1;
```

```
[model.CB,model.qtrain,model.mqtrain,model.alphaCenters] =
    KSCcodebook(model.etrain,model.alpha);

model.Cextra = cell(k-1,1);
model.mqtrainExtra = cell(k-1,1);
model.alphaCentersExtra = cell(k-1,1);

for j=2:k
    [model.Cextra{j-1},model.qtrainExtra(:,j-1),model.mqtrainExtra{j-1},
        model.alphaCentersExtra{j-1}] =
            KSCcodebook(model.etrain(:,1:j-1),model.alpha(:,1:j-1));
end;
end
```

```
function [C,qtrain,mqtrain,alphaCenters] = KSCcodebook(etrain,alpha)

  % constructing the code book, by binarizing the projected vector matrix
  % obtained from training data. The code book is obtained by selecting top
  % k most frequent codebook vectors

  [N,d]=size(etrain);
  k=d+1;

  betabin=sign(etrain);
  [C,m,uniquecw]=unique(betabin,'rows');

  cwsizes = zeros(length(m),1);
  for i=1:length(m)
      cwsizes(i) = sum(uniquecw==i);
  end;

  [temp,j]=sort(cwsizes,'descend');

  if length(m)<k
      k = length(m);
  end;

  C = C(j(1:k),:);
  qtrain=zeros(N,1);
  for i=1:k
      qtrain(uniquecw==j(i))=i;
  end;
```

```
  find_groups = [alpha qtrain];
  alphaCenters=0;

  [qtrain,mqtrain] = KSCmembership(etrain,C);

end
```

```
function [qtest,mqtest]=KSCmembership(etest,CB)

% Determines the cluster membership based on the hamming distance of the
    project vector
% and the codebook vectors

% Author: Debopriya_Ghosh

etest2=sign(etest);

% compute Hamming distances between the test encoding vectors and codebook
hamdists = pdist2(etest2,CB,'hamming');

% Assign to the cluster codeword with minimal Hamming distance
[ymin,qtest]=min(hamdists,[],2);
mqtest = num2cell(qtest);


end
```

```
function [ dist ] = findCluster( Data )
%Finds the cluster on basis of similarity of profile attributes.
% Input: The FeatureMatrix
% Output: Vector containing the hamming distance of each node
% from the basis vector
% Author: Debopriya Ghosh

% The basis vector
  X = Data(1,:);

  dist =[];
  dist(1) = 0;
  for i = 2: size(Data,1)
      Y = Data(i,:);
      dist(i) = pdist2(X, Y,'Hamming');
  end
end
```

APPENDIX B

Notations

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

(2) The cardinality of the set V is denoted as N.

(3) The cardinality of the set E is denoted as e.

(4) The matrix A is a $N \times N$ matrix and represents the affinity or similarity matrix.

(5) For unweighted graphs, A is called the adjacency matrix and $A_{ij} = 1$ if $(v_i, v_j) \in E$, otherwise $A_{ij} = 0$.

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

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

(8) The degree matrix is represented as D, a diagonal matrix with diagonal entries $d_{i,i} = \Sigma_j A_{ij}$.

(9) The adjacency list corresponding to each vertex $v_i \in V$ is given by $A(i;:)$.

(10) The neighboring nodes of a given node v_i are represented by $Nbr(v_i)$.

(11) The median degree of the graph is represented as M.

BIBLIOGRAPHY

Agrawal R., Imielinski T., S. A. (1993). Mining association rules between set of items in large databases. In *SIGMOD*, pp. 207–216.

Alzate C., S. J. A. K. (2010). Multiway spectral clustering with out-of-sample extensions through weighted kernel pca. *IEEE Trans. Pattern Anal. Mach. Intell. 32*, 335–347.

Azizifard, N. (2014). Social network clustering. *I. J. Information Technology and Computer Science 01*, 76–81.

Baylis, D. J. (1998). Error correcting codes: A mathematical introduction. Chapman Hall/CRC Mathematics Series 15, CRC Press. Boca Raton, FL, USA.

Fukunaga, K. (1990). *Introduction to statistical Pattern Recognition*. Academic Press.

Kang U., F. C. (2011, December). Beyond "caveman communities": Hubs and spokes for graph compression and mining. In *IEEE 11th International Conference on Data Mining*, Vancouver, Canada, pp. 300–309.

Langone R., Alzate C., S. J. A. K. (2013). Kernel spectral clustering with memory effect. *Phys. Stat. Mech. Appl.*

Leskovec J., F. C. (2006, August). Sampling from large graphs. In *12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Philadelphia, USA, pp. 631–636.

Luxberg, V. (2007). A tutorial on spectral clustering. *Stat. Comput.*, 395–416.

Madar, J. C.-S. (1999, April). Semantic compression and pattern extraction with fascicles. Master's thesis, The University of British Columbia.

Maiya A., B. W. T. (2010, April). Sampling community structure. In *WWW' 10, International Conferene on World Wide Web*, Raleigh, NC, USA, pp. 701–710.

Mall R., Langone R, S. J. (2013). Fast and unique representative subset selection for large scale community structure. *Internal Report; ESAT-SISTA,K.U. Leuven: Leuven Belgium*, 13–22.

McAuley, J. and J. Leslovec (2012). Discovering social circles in ego-network. NIPS 2012.

Muflikhah, L. and B. Baharudin. Document clustering using concept space and cosine similarity measurement. In *International Conference on Computer Technology and Development*, Kota Kinabalu, Malaysia, pp. 58–62. International Conference on Computer Technology and Development.

Raghvendra Mall, Rocco Langone, J. A. S. (2013, May). Kernel spectral clustering for big data networks. *Entropy, Special Issue: Big Data 13*(5), 1567–1586.