Graph Representations using Adjacency Matrix Transforms for Clustering

Nikolaos Tsapanos
Informatics Department
Aristotle University of Thessaloniki
Email: niktsap@aiia.csd.auth.gr

Ioannis Pitas
Informatics Department
Aristotle University of Thessaloniki
Email: pitas@aiia.csd.auth.gr

Nikolaos Nikolaidis
Informatics Department
Aristotle University of Thessaloniki
Email: nikolaid@aiia.csd.auth.gr

***Abstract*—This paper is meant as a proof of concept regarding the application of standard 2D signal representation and feature extraction tools that have wide use in their respective fields to graph related pattern recognition tasks such as, in this case, clustering. By viewing the adjacency matrix of a graph as a 2-dimensional signal, we can apply 2D Discrete Cosine Transform (DCT) to it and use the relation between the adjacency matrix and the values of the DCT bases in order to cluster nodes into strongly connected components. By viewing the adjacency matrices of multiple graphs as feature vectors, we can apply Principal Components Analysis (PCA) to decorrelate them and achieve better clustering performance. Experimental results on synthetic data indicate that there is potential in the use of such techniques to graph analysis.**

I. Introduction

Graphs have been known to mathematicians for centuries. A wide variety of problems can be expressed and solved in graph form. However, the fact that most important basic graph related problems are NP-complete (for example, *clique detection*, *Hamiltonian paths* and *cycles*, *subgraph detection*) and even the *graph isomorphism* problem belongs to its own complexity class (neither shown to be in P nor NP-complete, considered to be somewhere "in-between"). This has discouraged researchers in the past and related work had waned.

With the emergence of the World Wide Web and social networks, graph analysis has begun attracting researchers in the past few years. Due to the aforementioned computational complexity issues, a lot of approaches to various problems are heuristic. An interesting new approach presented in [1] involves detecting anomalous subgraphs by defining and then maximizing the Signal to Noise Ratio of a candidate anomalous subgraph by including or excluding random nodes.

In this paper we will focus on graph clustering, by which we include both the clustering of nodes in a single graph (that we will henceforth refer to as node clustering) and the clustering of different graphs (that we will henceforth refer to as graph clustering). For an extensive review of graph clustering, the interested reader may refer to [2]. We change the order of the nodes through which the adjacency matrix is generated, aiming to maximize the sum of the first k elements of the diagonal of the graph's adjacency matrix DCT and, by doing so, bring nodes of the same strongly connected component (node cluster) in consecutive places in the order of the nodes. In order to cluster different graphs together, we use the accuracy of the nearest neighbor algorithm to measure how helpful applying PCA is in decorrelating the graphs for better clustering performance.

The paper is structured as follows: section II briefly describes the basics of a graph, section III discusses the relation between a graph's adjacency matrix and the 2 dimensional DCT basis and the application of this relation to node clustering, section IV presents the effects of applying PCA on the adjacency matrices of multiple graphs to cluster graphs, while section V concludes the paper.

II. Definition of a graph

A graph $G = (\mathcal{V}, \mathcal{E})$ is formally defined as a pair of sets $\mathcal{V}$ containing the graph's vertices and $\mathcal{E}$ containing the graph's edges. An edge (u, v) exists in $\mathcal{E}$ iff the vertices u and v are connected in G. If the graph is directed, the order of the vertices in the edge matters. The adjacency matrix A of a graph is a $|\mathcal{V}| \times |\mathcal{V}|$ matrix. The element A_{ij} is 1 iff $(i, j) \in \mathcal{E}$ and 0 otherwise. If the graph is undirected, then its adjacency matrix is symmetrical. In this paper we consider graphs to be undirected.

III. DCT for node clustering

To objective of node clustering is to find groups of nodes that are strongly connected with the other nodes in the group and relatively disjoint from the rest of the graph. For an extended review on the subject, readers may refer to [2]. There are several categories of methods for node clustering. Graph cut based methods separate subgraphs into further subgraphs until a subgraph is not easy to further cut. There are several spectral methods that use a combination of the eigenvalues and eigenvectors of the graph's Laplacian matrix to perform the clustering. Incremental methods process the nodes in order and proceed to update an initially empty clustering by adding the current node to one of the existing clusters or into a new one. Other methods use node label propagation algorithms to cluster with the same label. There are also some methods inspired by physics, such as one that model the graph as an electrical circuit and cluster nodes according the their voltage.

The Discrete Cosine Transform [3] is a Fourier-like transform that expresses a sequence of numbers or a signal in terms of sums of sinusoidal functions. Each function has a different

frequency and is multiplied by the proper amplitude, which depends on the signal, in the sum. The DCT has been widely applied to signal processing tasks, such as signal decorrelation and signal compression. There are several DCT variants, but in this paper we will only consider the 2D DCT. This transform receives an input $m \times n$ matrix $\mathbf{X}$ and calculates the matrix $\mathbf{S}$ as given by the following equation:

$$s(i,j) = \sum_{k=1}^{m} \sum_{l=1}^{n} x(k,l) \cos(\frac{\pi}{m}(k+\frac{1}{2})i) \cos(\frac{\pi}{n}(l+\frac{1}{2})j) \quad (1)$$

Since the adjacency matrix of an undirected graph can be viewed as a square, symmetric along the diagonal, binary image, there is no modification required to used 2D DCT to move from the graph's edges domain to the DCT's frequency domain. An illustration of the 2D DCT basis functions for a 8×8 matrix is shown in Figure 1.

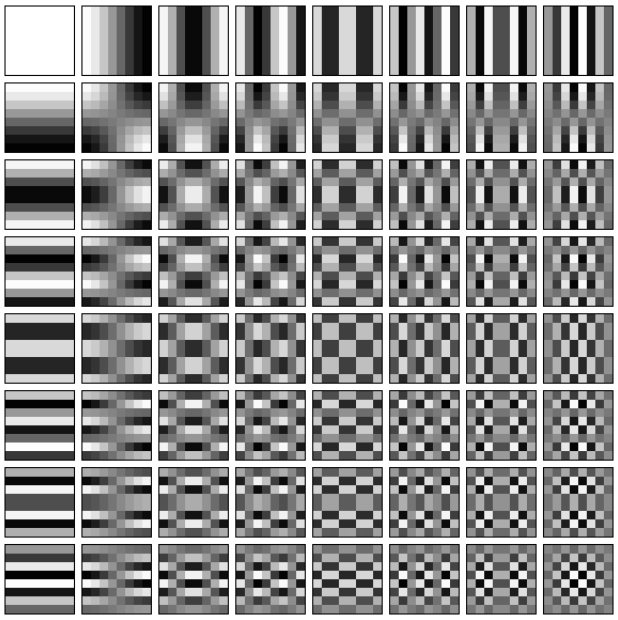

Fig. 1. The basis functions of an 8×8 matrix.

Let us now consider the issue at hand; node clustering. In a graph, a cluster of nodes, or a closely connected component, is a subgraph whose nodes are heavily connected with each other while at the same time being loosely connected to the rest of the graph's nodes. When the nodes of each cluster appear in consecutive order when the adjacency matrix is defined, the clusters can be seen clearly as big squares along the matrix's diagonal. An example of this can be seen in the graph in Figure 2a. When the nodes appear in random order, however, the apparency of the clusters is lost, as illustrated in Figure 2b.

Looking back at Figure 1, one can notice that DCT coefficients that represent "big boxes" in the diagonal of the original matrix are in the diagonal of the DCT matrix. For example, the DCT element $s(2,2)$ represents a graph comprised of two clusters equal in size, while $s(3,3)$ represents a graph comprised of three clusters and so on. We, therefore, expect that we could gather clusters together in the adjacency matrix by maximizing the sum of the first few elements of the DCT diagonal though changing the ordering which the nodes have.

We generated a random graph with 32 nodes to test this hypothesis. The initial graph contained three cliques (fully connected graphs) disjoint with each other, a 16-clique, a 11-clique and a 5-clique. Then we added noise to the adjacency matrix by randomly adding and removing edges from the graph, resulting in the graph in Figure 2a. Finally, we shuffled the adjacency matrix by swapping the ordering of two nodes several times, resulting in the graph in Figure 2b.

We wrote a simulated annealing program in MATLAB [4] that would randomly swap the ordering of two nodes, perform 2D DCT in the resulting graph, calculate the sum of the k first elements of the DCT's diagonal and either directly make the transition, if the new sum was greater than the previous sum, or probabilistically checking whether to make the transition or not, based on the difference of the two sums and the current temperature. Note that k is a parameter of the algorithm that must be provided by the user and it has some interesting effects on the algorithms output, as we will see.

We set the initial temperature to 10, the ending temperature to 0.1 and the temperature was modified by multiplying it with 0.99. We set the number of iterations to 500 at each temperature step. The program returned the adjacency matrix whose DCT yielded the maximum sum of the first k elements of the transform's diagonal. We used the shuffled graph from Figure 2 as input. Sample outputs can be seen in Figure 3.

Overviewing the results, we notice some interesting things concerning the effect that parameter k has on the program's output. When k was set to 3 (second row of images in Figure 3), essentially the number of actual clusters, the program produced the best outputs. Setting $k = 2$ (first row) produces some good results, but sometimes the two smaller clusters are mixed together, which makes sense, considering that element $s(2,2)$ of the DCT transform corresponds to two equal clusters, so including this element in the sum and not $s(3,3)$ guides the program into finding two clusters. For higher values of k (4 and 5, third and fourth row respectively) we can again see that, while sometimes the solution is good, at other times the big cluster or the medium cluster is split into two smaller ones. This can be easily explained by the fact that higher frequencies

(a) (b)

Fig. 2. The same graph under different isomorphisms (edges are marked in black): (a) Each cluster's nodes are in consecutive places in the ordering of the nodes (b) Random node ordering.

in the DCT's diagonal represent more and smaller boxes in the adjacency matrix's diagonal, so the program will be lead to solutions with smaller clusters.

Overall, in order to ensure that the program at its current state will perform well, the parameter k has to apparently "guess" the number and size of the clusters. This is an obvious drawback. It does, however provide evidence to the potential flexibility of this approach. It may be possible to describe a particular adjacency matrix configuration that we are looking for in the DCT's frequency domain though the proper combination of the DCT elements into a cost function. We can then use this program to attempt to detect the target adjacency matrix configuration.

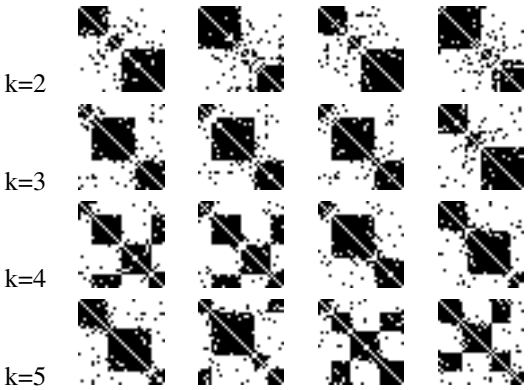

Fig. 3. Sample outputs of the node clustering program. Each row corresponds to results for different values of parameter k.

IV. PCA FOR GRAPH CLUSTERING

We will now focus on the issue of graph clustering, in which we have several graphs and we would like to group them into clusters. Popular graph matching methods use edit cost functions to determine how similar two graphs are by measuring the cost of editing one graph into the other, as well including edit distance based kernel functions into the classification process [5]. Spectral methods use the similarity between the eigenvectors of vertices to establish a correspondence between the vertices of two graphs [6].

Principal Component Analysis [7] is an orthogonal linear transformation that attempts to decorrelate a data set and sort the dimension axes in descending order of variance. The new axes are dictated by the eigenvectors of the data set's covariance matrix, while their order is given by the corresponding eigenvalue. PCA is a popular method of feature extraction through dimensionality reduction, as it is possible to project the data only into the first p dimensions with the highest eigenvalue of the PCA space.

PCA has been used widely to decorrelate real data in various applications. Graphs, however, are not as straight forward to consider as regular feature vectors. In the general case, there are isomorphism concerns, causing two graphs that are isomorphic (in essence the same graph) to appear with wildly different adjacency matrices. In our case, we will assume that the nodes are labeled, so the isomorphism issue is sidestepped. Nodes with the same label are always in the same row and column of the adjacency matrix. This is not entirely implausible, as it could be the same node in a different instance of a graph that is evolving over time. We will also assume that the size of the graph remains the same. We will use synthetic data to test whether using PCA has any meaningful effect on the correlation of the data.

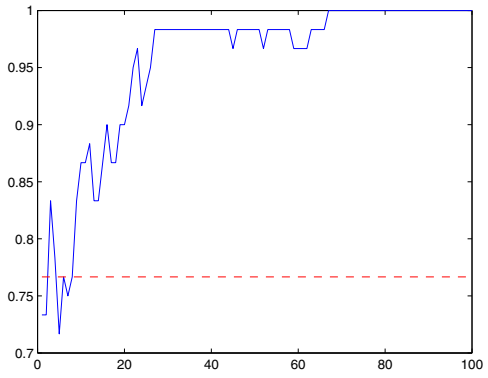

Fig. 4. The performance of the nearest neighbor algorithm (solid line) vs the number of principal components retained. The performance on the raw data (dashed line) provided for reference.

The generated dataset consisted of three clusters. We initially generated three graphs. One graph consists of one big node cluster in the first 16 nodes and two 8-node clusters, another graph with an 8-node cluster in the beginning, one 16-node cluster in the middle and another 8-node cluster in the end, while the third graph consisted of two 16-node clusters. The graphs for each graph cluster were generated by adding considerable noise (40% chance of flipping edges) to the three original graphs. We generated 100 graphs for each cluster and labeled each cluster with $1-3$. We split each cluster into 80 training graphs and 20 test graphs then collected all the training and testing subsets into a training set with 240 graphs and a test set with 60 graphs.

We used the performance of the Nearest Neighbor Classifier (NNC) algorithm to evaluate the effect of using PCA to decorrelate the data. The adjacency matrices of all graphs involved that is to be tested is reshaped into a vector. When a graph was to be tested, we measured the Euclidean distance with every vector in the training set and classified it with the label of the closest match, as per the NNC algorithm. If the resulting labeled was the same as the test label the search was considered successful.

When using the raw data ro set the baseline performance, the accuracy of the NNC was 0.76666. We then projected the data to the PCA space and iteratively used the NNC algorithm on the p first principal components of the projected data. Figure 4 presents the accuracy of the NNC plotted against the number of principal components retained. When using the 70 first principal components and thereafter, the accuracy of

the NNC had reached 1, indicating that PCA was working as intended.

V. Conclusion

In this paper we have demonstrated that using a signal representation technique, DCT in particular, can provide a viable alternative to a graph's adjacency matrix in a node clustering problem. DCT provides us with a way to measure the quality of the current clustering and estimate the effect that swapping two nodes will have on that quality, without directly involving the rest of the graph's nodes. DCT could also provide interesting new ways to describe the desired properties of a graph in the frequency domain of its adjacency matrix and solve more graph related problems in similar fashion. We also provided evidence towards the viability of dimensionality reduction in the vectorized adjacency matrix space. Most probable application of such techniques is the temporal study of changing graphs in order to analyze or even compress them.

References

[1] B. Miller, N. Bliss, and P. Wolfe, "Toward signal processing theory for graphs and non-euclidean data," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, march 2010, pp. 5414 –5417.

[2] S. E. Schaeffer, "Graph clustering," *Computer Science Review*, vol. 1, no. 1, pp. 27 – 64, 2007.

[3] G. Strang, "The discrete cosine transform," *SIAM Review*, vol. 41, pp. 135–147, 1999.

[4]

[5] H. Bunke, C. Irniger, and M. Neuhaus, "Graph matching - challenges and potential solutions," in *ICIAP*, ser. Lecture Notes in Computer Science, F. Roli and S. Vitulano, Eds., vol. 3617. Springer, 2005, pp. 1–10.

[6] T. Caelli and S. Kosinov, "An eigenspace projection clustering method for inexact graph matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, pp. 515–519, 2004.

[7] I. T. Jolliffe, *Principal Component Analysis*, 2nd ed. Springer, Oct. 2002.