Independent Study

Yun-shao Sung*

This is independent study mid-way report

1. Abstract. There are many approaches to understand the structure of music, and this is a project applying spectral graph theory with Laplacian formula to analyze repeated patterns in musical recordings. Starting with low-dimensional encoding of repeating structure, hierarchical relationship among repeating components will be built by changing number of eigenvectors. Further works for this project will include Neural Network and back-propagation to apply thie technique to analyze the structure of wide variety of musics.

2. Baseline Approach. Here is the baseline approach which can be roughtly divided into two parts: the goal of the first part is constructing *sequence-augmented affinity matrix* A from *symmetric matrix* R of the feature similarity at each time points, and the second part is to construct symmetric normalized Laplacian matrix and obtain the top m eigenvectors with the m smallest eigenvalues, and then perform k-means for boundary detection. The high level pseudocode is described in Algorithm 1.

Algorithm 1 Baseline Approach

Input: number of top m smallest eigenvalues

```
M = getSymmetricMatrix() //to get affinity matrix
L = scipy.sparse.csgraph.laplacian() //to get symmetric normalized Laplacian matrix
eigVals, eigVecs = np.linalg.eig(L)
Y = getMthSmallest(eigVals, eigVecs, m)
return boundaryDetection(Y)
```

2.1. Symmetric and Laplacian Matrix. Initially symmetric matrix is constructed manually, and the symmetric properity is ensured by checking $m == m^T$. The following symmetric normalized Laplacian L is defined as:

$$L = I - D^{-1/2} A D^{-1/2} \tag{2.1}$$

The properity of symmetric is important as it shows the correlation between time i and j which is an undirected graph, and the proterity is important to hold to make sure the eigenvectors calculated from Laplacian matrix is done correctly. Figure 2.1 shows the example of symmetrix matrix and its corresponding top-10 eignevectors, and method is implemented from Algoritm 1.

*yss265@nyu.edu

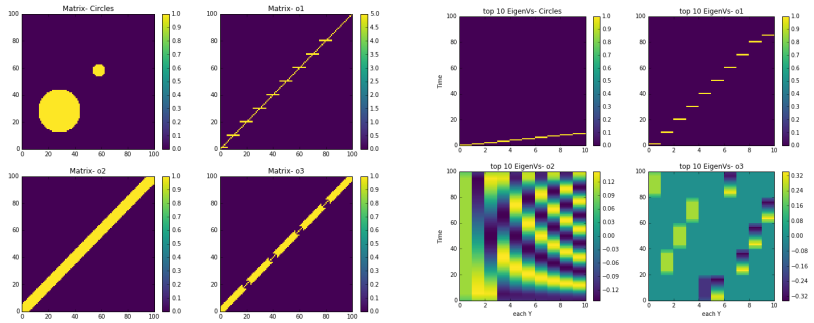

Figure 2.1. *Symmetric matrixs and corresponding eigen vectors. Left: four different type of symmetirc matrix, and Right: the top-10 eigen vectors*

Interestingly, eigenvectors are like the building blocks of the original graph, and the signal of each eigenvector are like showing the correlation of each time point in this building block. As the top-m eigenvectors matrix Y constructed, we can notice from Figure 2.2 that the dot product of YY^T is similar to the original symmetric matrix. I think understand the properity of eigenvector is important, as the each row from Y is similar to the eigen-feature representation at each time point, and therefore we can perform k-means for boundary base on Y as illustrated in Algorithm 2.

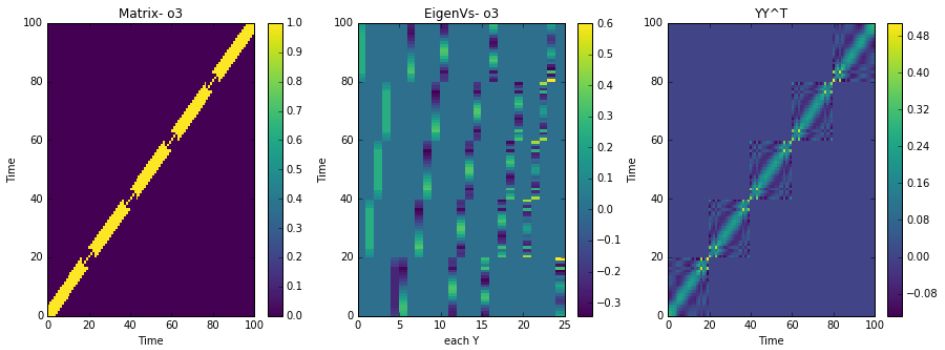

Figure 2.2. *The steps from symmetric matrix to eigenvector*

2.2. Boundary Detection. As the pseudocode in Algorithm 2, once the Laplacian matrix is constructed, each row is the representation of eigen-features at specific time. Therefore, running k-means for eigen-features of all time points will yield the results of which centroids of this time point belongs to, and therefore the place where the $t_i \neq t_{i+1}$ is where the boundary is. As showed in Figure 2.3, boundary is correctly detected at each time point, but when doing experiments I noticed the number of iteration during k-means will affect the correctness of boundary, and the more iteration usually gives better boundary.

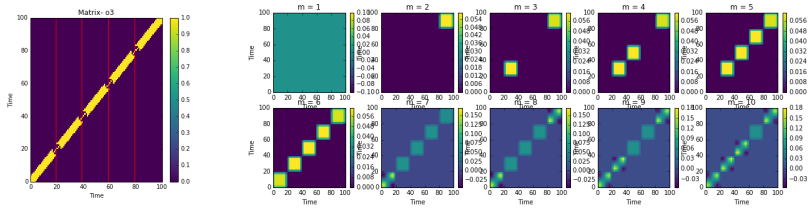

Figure 2.3. *Boundary detection and pair-wise frame similarities (YY^T). Left: Boundary detection of original recurrence graph, and Right: visualization of YY^T using the first 10 eigenvectors*

Algorithm 2 boundaryDetection

Input: Laplacian eigenvectors $Y \in R^{n \times m}$
Output: Boundary b, Centroids c

1: $\overline{y}_i = \frac{Y_i}{\|Y\|}$ //normalize each row Y_i
2: Run k-means on $\{\overline{y}_i\}_{i=1}^{n}$
3: Let c_i denote the cluster containing $\overline{y}_i$
4: $b \leftarrow \{i | c_i \neq c_{i+1}\}$
5: return b, c

2.3. Affinity Matrix from Real Music. As basic process is established above, I started to construct matrix from real music, and the music I started with is *The Beatles - Come Together*. Data is sampled in the rate of 22050Hz, and transformed to .wav format, as it can be imported by scipy. Imported signal has two tracks, and all following analysis is based on the first track, and feature extraction is done by librosa cqt computing the constant-Q transform of an audio signal. As we will get too little information at low frequencies and too much information at high frequencies if we just simply double the frequency for Fourier transformation[Ref 6], and cqt will be better suit for extracting feature from music signal because it spaced the center frequencies of the frequency bins geometrically, and Q-factors are all equal[Ref 7]. As we can see from Figure 2.4, feature extraction from mfcc and cqt is different and seems cqt is able to cpature long-range repeating form.

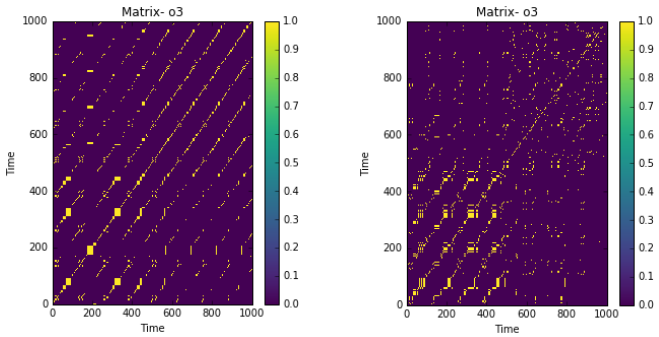

Figure 2.4. *Visualization of matrix in Figure 2.3 after Left: cqt and Right: mfcc feature extraction*

The *affinity matrix* A is constructed by:

$$A_{ij} = \mu R'_{ij} S^{rep}_{ij} + (1-\mu)\Delta_{ij} S^{loc}_{ij} \tag{2.2}$$

where R_{ij} is recurrence matrix and $R[i][j]$ is 1 if time i and j are mutual k-nearest neighbors else is 0, and R is symmetric. $\Delta[i][j]$ is 1 if $|i-j| = 1$ else is 0. S_{ij} is the result of matrix passed to $GaussianKernel$ of feature vectors x_i, x_j:

$$G(M) = S_{ij} = exp\left(\frac{-1}{2\sigma^2}\|x_i - x_j\|^2\right), \textbf{ for each row of i and j in M} \tag{2.3}$$

and $S^{rep}_{ij} = G(R'_{ij})$ and $S^{loc}_{ij} = G(\Delta_{ij})$, and μ in Equation 2.2 is a factor balancing local and global linkage, and optimal of μ is solved by:

$$\mu^* = \frac{< d(\delta), d(R') + d(\delta) >}{\|d(r') + d(\delta)\|^2} \tag{2.4}$$

where we treat the input matrix as vector of degree-sum where $d(\cdot) = [d_i(\cdot)]^n_{i=1}$, and $d_i(G) = \Sigma_j G_{ij}$ is the degree sum at time i. However, after the affinity matrix A being constructed from Equation 2.2, we loss the symmetric proerity, even though $R'_{ij}, S^{rep}_{ij}, \Delta_{ij}, S^{loc}_{ij}$ are symmetric. The product from matrix multiplication from Equation 2.2 is not symmetric, and the addition modification may be required. Figure 2.5 is the visulization for R, A, and Y with the top 25 eigenvectors.

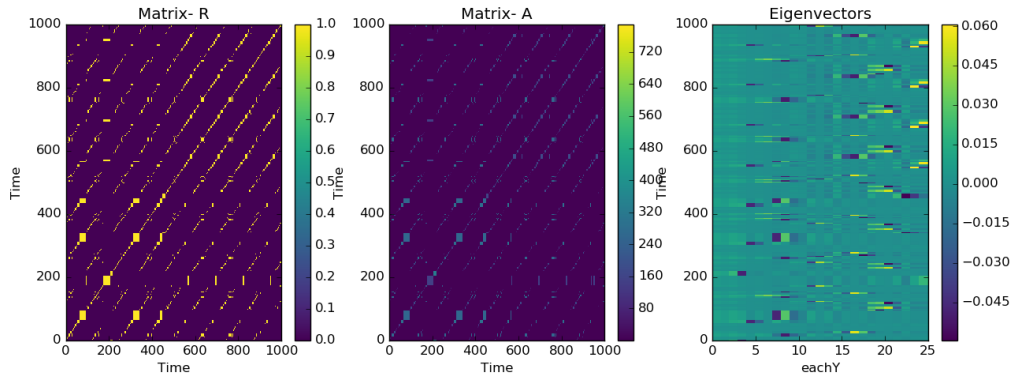

Figure 2.5. *First 1000 time point of recurrence and affinity matrix, and first 25 eigenvectors for The Beatles - Come Together*

2.4. Discussion. Here is the section for discussion about the current issues.
The first issue is about the time complexity. Input signal of a 3 minutes music contains 6291012 data points and still 12288 points after mfcc or cqt, which gives us a recurrent matrix R in shape of (12288, 12288). For a graph matrix like this big shape will give a hard time for visualizing the whole matrix, as well as spending lots of time on calculations like Equation 2.3 which is an $O(n^2)$. Also, np.linalg.eig, line 3 in Algorithm 1 is requiring computation time, and

line 4 in Algorithm 1 also requiring sorting since the returned eigen value form line 3 doen't guarantee in sorted order. I think these complexity is one of the reason that motivate Ref 4 to use encode and decode method, but the advantage of our method is able to understange the hierarchical structure which is different from Ref 4. The majority vote method mentioned in Ref 2 maybe will be a way to reduce the shape of the matrix, but addition clarification may needed to implement this method.
The second issue is about the construction of affinity matrix A in Equation 2.2. We loss the symmetric properity after A constructed, mainly due to the fact that matrix multiplication is not guarantee the symmetric properity.

2.5. Further Works.

REFERENCES

[1] A Tutorial on Spectral Clustering
[2] Analyzing Song Structure with Spectral Clustering
[3] Deep clustering: Discriminative embeddings for segmentation and separation
[4] Learning Deep Representations for Graph Clustering
[5] Hierarchical Evaluation of Segment Boundary Detection
[6] Calculation of a Constant Q Spectral Transformation
[7] Constant-Q Transform Toolbox for Music Processing