Independent Study – Learning Music Structure by Laplacian Formula

Peter Yun-shao Sung*

Abstract. There are many approaches to analyzing music structure by features extracted from dimension of time series. With contruction of similarity matrix, repeated pattern can be captured which is the building block for large-scale structure. This is the work based on the Laplacian Matrix, which is essential start point of spectral clustering. We introduce variables that are trainable to reduce the cost of Laplacian Matrix from true lable, and we run this method on wide variable of music recordings. Finally, we demonstrate using these trained variable for performing music segmentation.

1. Laplacian formula. Normalized Laplacian matrix is the essential start point for identifying music segmentation, and the correct boundary detection is done in my baseline approach (Ref 2). For proper boundary detection, we woule like to train the initial laplacian matrix (L) close to true laplacian (L^*) from true inerval annotation from SALAMI dataset. Therefore, to train and update the model, this section is for deriving the L and $\frac{\partial L}{\partial w_{i,j}}$, which $W_{i,j}$ representing the recurrence weighting between time point i and j.
Given the definition of normalized laplacian matrix:

$$L := I - D^{\frac{1}{2}} W D^{\frac{1}{2}} \tag{1.1}$$

D is degree matrix defined as the diagnal matrix with degrees $d_1, d_2, \ldots, d_n$, which d_i is accumulation of $w_{i,j}$ which defined as followed:

$$d_i = \sum_{j \neq i}^{n} w_{ij} \tag{1.2}$$

After multiplication and the result of equation 1.1 can be rewrite as:

$$L := I - D^{\frac{1}{2}} W D^{\frac{1}{2}} = \begin{pmatrix} 1 & \frac{-w_{12}}{\sqrt{d_1 d_2}} & \cdots & \frac{-w_{1n}}{\sqrt{d_1 d_n}} \\ \frac{-w_{21}}{\sqrt{d_2 d_1}} & 1 & \cdots & \frac{-w_{2n}}{\sqrt{d_2 d_n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{-w_{n1}}{\sqrt{d_n d_1}} & \frac{-w_{n2}}{\sqrt{d_n d_2}} & \cdots & 1 \end{pmatrix} = \begin{cases} \frac{-w_{i,j}}{\sqrt{d_i d_j}} & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases} \tag{1.3}$$

To take the derivative of L w.r.t $w_{i,j}$. Results is as follow and detail derivation is in appendix 2.1:

$$\frac{\partial L}{\partial w_{i,j}} = \begin{cases} 0 & \text{, if } i = j \\ \frac{-1}{\sqrt{d_i d_j}} + \frac{w_{i,j}(d_i + d_j)}{2(d_i d_i)^{\frac{3}{2}}} & \text{, for position } (i,j),(j,i) \\ \frac{w_{l,k}}{2\sqrt{d_k} d_l^{\frac{3}{2}}} & \text{, for all position } (l,k),(k,l), \text{ where } k \neq i \& j \text{ and } l = i \| j \\ 0 & \text{, for any other position} \end{cases} \tag{1.4}$$

*yss265@nyu.edu

The key idea of this derivation is that, when taking derivative of $w_{i,j}$ all elements on the i-th and j-th row and column will be altered because of the degree from equation 1.2 changed.

2. Methods. Here is the section describing methods being used for data and matrix process.

2.1. Data. The data I used is mainly from SALAMI dataset. Here I have 333 audio files in mp3 format, and each music has segment annotation based on functions, uppercase, and uppercase, and most of time there are two annotators. Here I only used uppercase annotation from first annotator.

2.2. Feature Extraction. Each 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(Ref4), 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 5).

2.3. Recurrent Matrix. Each cqt-proccessed signal will be normalized followed by librosa_beat_beat_track and librosa_feature_sync. The purpose of beat_track function is to pick peaks in onset strength that is approximately consistent with estimated tempo. The sync function is to use those beats to synchronous aggregate cqt signals for dimension reduction, and here I use medium as aggregation method. One issue will be raised during this process: since all music segment annotation is labeled on time-domain, how to convert the label from time to frame, and further from frame to beat will need to considered. Conversion from time to frame is not hard, given the sampleing rate and hop_length, librosa already has fusion performing task for this purpose. By check the source code of librosa_feature_sync, I modified the code and return not only aggregated-cqt but also a beat-frameInterval map, and input this to my function called loadInterval2Frame to perform the task of time-frame-beat conversion. Furthermore, random seed is fixed, and therefore different models can be compared.

3. Models. We like to build the model minimize the loss J between L and L^*, which defined as followed:

$$J := \frac{1}{2}||L^* - L||_2^2 = \frac{1}{2}\sum_i \sum_j (L^*_{i,j} - L_{i,j})^2 \tag{3.1}$$

With the loss function defined, we would like to design our model with trainable variables θ that minimizing the loss fuction during the update:

$$\theta^{t+1} = \theta^t - \alpha \frac{\partial J}{\partial \theta} \tag{3.2}$$

3.1. Model 1– train on sigma. Here we design our model with trainable varialbe $\sigma_{i,j}$, which is gaussian width to define the similarity coefficient $w_{i,j}$ between features x_i and x_j:

$$w_{i,j} = exp(-(\frac{||x_i - x_j||_2}{\sigma_{i,j}})^2) \tag{3.3}$$

As we like to minimize the loss during each of the $\sigma_{i,j}$ update (eqation 2.2), the derivitive need to expand by chain rule:

$$\frac{\partial J}{\partial \sigma_{i,j}} = sum[(L - L^*) \odot \frac{\partial L}{\partial w_{i,j}}] \cdot \frac{\partial w_{i,j}}{\partial \sigma_{i,j}} \tag{3.4}$$

Where $\odot$ and sum are element-wise multiplication and summation respectively. The idea of this derivation is when there is slight changes on $\sigma_{i,j}$, it directly affcts $w_{i,j}$ and also all elements on i-th and j-th row and column by the degree in equation 1.2. Figure 3.1 is the test result and the relative error between numerical and analytical method are all below 1e-5.

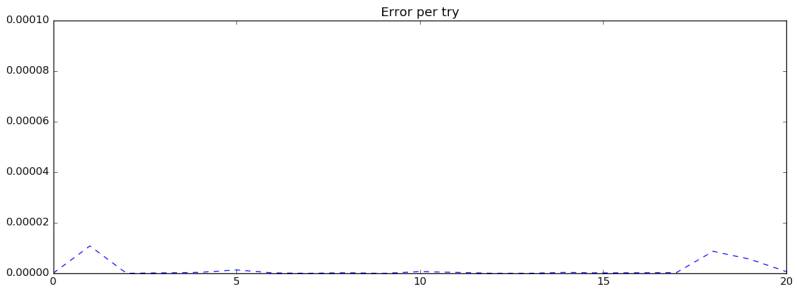

Figure 3.1: Deriative of loss w.r.t $\sigma_{i,j}$. Relative error $\frac{|f_a - f_n|}{max(|f_a|,|f_n|)}$ per each try

Combining equation 2.4 and 2.2, we performed the update for each of $\sigma_{i,j}$. We are training the variables for minimizing loss on laplacian, but since laplacian is normalized version of recurrent matrix that values are hard to visualize, therefore we all showing the result on recurrence matrix. Figure 3.2 is the recurrent matrix after sigma being trained. With boundary detection (Ref 2) performed, the detected boundaries are identical. The video of recurrence matrix updates over each steps is also available in Ref 3.

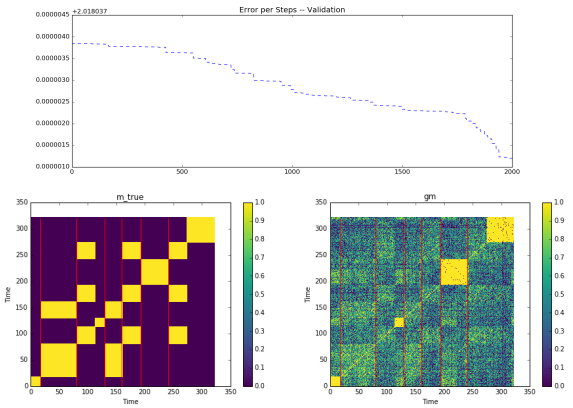

Figure 3.2: Top: Loss per step. Recurrence matrix of Bottom-left: true sigment annotation, and Bottom-right: feature with trained sigma

3.2. Model 2– train on Q. Training on Model 1 is not scalable. First is due to the traing is on $\sigma_{i,j}$, which is the similarity between time point i and j and cannot apply to other song. Second is due to time interval will be different for each songs. Therefore, traing on time scale is not suitable to various songs. Here we propos the other method that is trainning on cqt bin and define as follow:

$$\begin{aligned} W_{i,j} &= S_{i,j}^T \cdot Q \cdot S_{i,j} = \Sigma_k (S_{i,j}^k)^2 \cdot q_k \\ S_{i,j} &= cqt[i] - cqt[j] \end{aligned} \tag{3.5}$$

$Q = [q_1, q_2, ..., q_n]$ is diagnoal matrix with the shape of number of cqt bins. Therefore, we can rewrite our objective function to be:

$$\arg\min_Q J(L) = \arg\min_Q \frac{1}{2}||L^* - L||_2^2 \tag{3.6}$$

And the update rule for each of q_i will be:

$$q_k = q_k - \alpha \frac{\partial J}{\partial q_k} \tag{3.7}$$

which:

$$\frac{\partial J}{\partial q_k} = \sum_{i,j} (L_{i,j} - L_{i,j}^*) \frac{\partial L_{i,j}}{\partial q_k} \tag{3.8}$$

changing each q_k is like changing the weight of k-th cqt frequency bin, and therefore this will affect all elements of recurrent matrix, and $w_{i,j}$, d_i, and d_j from equation 1.3 are all the function of q_k:

$$\begin{aligned} \frac{\partial L_{i,j}}{\partial q_k} &= \frac{-1}{\sqrt{d_i d_j}} \frac{\partial w_{i,j}}{\partial q_k} + \frac{w_{i,j}}{2(d_i d_j)^{\frac{3}{2}}} (\frac{\partial d_i}{\partial q_k} d_j + d_i \frac{\partial d_j}{\partial q_k}) \\ \frac{\partial w_{i,j}}{q_k} &= (s_{i,j}^k)^2 \\ \frac{\partial d_i}{\partial q_k} &= \sum_l (S_{l,i}^k)^2 \end{aligned} \tag{3.9}$$

The consistancy between numerical and anlytica method are confirmed by relative differnce smaller than 1e-5, and shown in figure 3.3.

Figure 3.3: Deriative of loss w.r.t q_k. Relative error $\frac{|f_a - f_n|}{max(|f_a|, |f_n|)}$ per each try

We first test this model on training single one song. We batch update the elements in Q all at once, and as we can see from figure 3.4 there is a noticable drop in the beginning 25 epochs with step size of 5. After that there is oscillation but slightly dropping curve. Maybe model has reached to minimum, but maybe the slightly dropping curve implies the possibility at saddle point and further improvement is possible by different update method, proper step size, or different Q initiation. From Ref 6 and Ref 7 we can see the video of recurrence matrix and Q during each epoch. After Q was trained, we can see from figure 3.4 that much of the noise from initial recurrent matrix was smoothed, and the pattern in the trained recurrent matrix became similar to ture label.

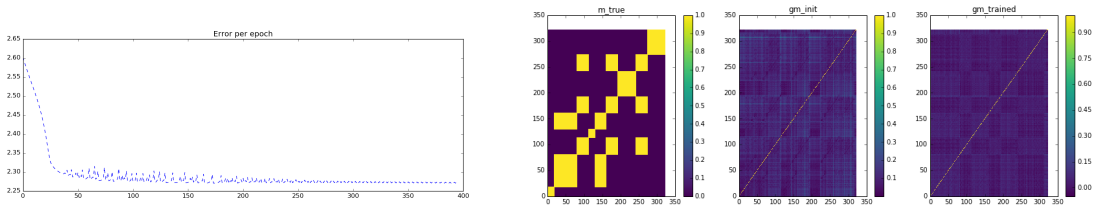

Figure 3.4: Left: loss function during each epoch. Right: recurrence matrix of true label, matrix with initial Q, and matrix with trained Q (True label vs Trained model)

Although matrix after training is cleaner and similar to true label, there are still many regions are not the same as true label. Therefore, this effect reflect on the matrix of top10 eigenvector matrix in figure 3.5. As we can see maybe the top 3 eigenvector is clearly showing some features, but there are too many noise for the remaining eigenvectors. Therefore, these noise on eigenvector will make boundary detection confused for which centroid it should belong, and therefore lots of boundaries were detected. Even there existed many noise boundaries, there still many clear boundaries were detected that are the same as true label, for example the boundary at time 10, 75, 110, 160, 200, and 240, which I think is mainly due to the contribution of clearn region in figure 3.4.

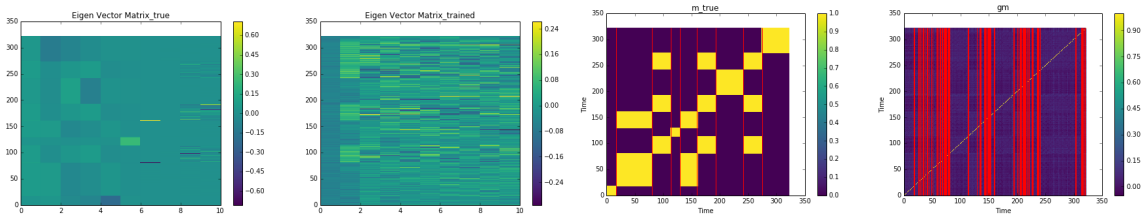

Figure 3.5: Left: matrix of top-10 eigenvectors. Right: boundary detection on true and train recurrence matrix (True label vs Trained model)

As the result of insterested in further understand the energy landscape, I did some tests on the update for single one song. As shown in figure 3.6, initially model was trained by different learning rate α. Small α lead to slow loss drop, but large α oscillates the curve.

This pre-train was done for about 900 epochs, and then followed by post-training with same update method but different/same learning rate. Basically, if pre- and post-training is using the sample learning rate, the curve is keep dropping and oscilation can still be oberved in $\alpha = 5$, but in a very small scale, potentionally suggesting model is at a state with very slight gradient. Furthermore, if model is pre-trained with $\alpha = 1$, and followed by post-train with $\alpha = 5$, the loss curve will just jump out and stay the same. This implys althought loss is dropped to similar value with different learning rate, but models are now at very different energy territory, and therefore change to different learning rate will make the model jump out of its delicate state and not coming back. The test here is the same model that showing in figure 3.5, which the boundary detection is fine for some parts but not all. Therefore, this is just testing on single one song and energy trap can be observed already, then better update model or deeper layer of traing modle might required for skipping the trap and accerating the training.

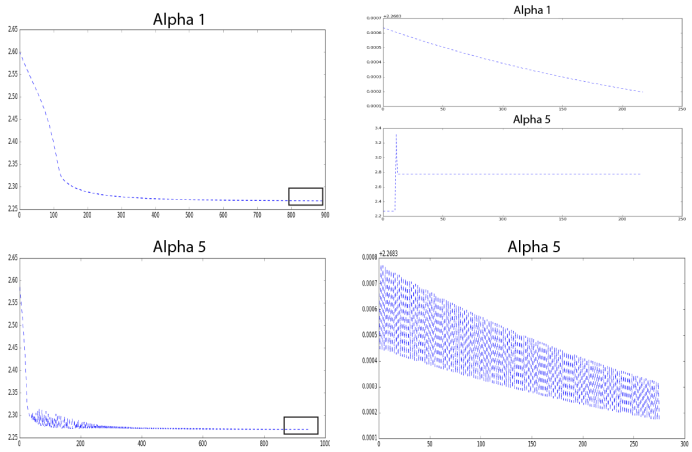

Figure 3.6: Loss function pre- and post-training during each epoch. Top: loss function during first 900 epochs with alpha of 1, and post training using alpha 1 and 5. Bottom: loss function during first 900 epochs with alpha of 5, and post training using alpha 5.

REFERENCES

[1] A Tutorial on Spectral Clustering
[2] "https://github.com/jfriend08/IS/blob/master/reports/midway/report.pdf"
[3] "https://youtu.be/h-JTtPNF8nA"
[4] Calculation of a Constant Q Spectral Transformation
[5] Constant-Q Transform Toolbox for Music Processing
[6] "https://youtu.be/bEbaQwhDSbM"
[7] "https://youtu.be/qNF4cmplpyk"

4. Appendix.

4.1. Differential of Laplacian Matrix. If we would like to take derivative of Laplacian w.r.t variable $w_{i,j}$ in the symmetric matrix W. Basically, except for $L_{i,j}$ the components of $L_{i,k}$, $L_{k,i}$, $L_{k,j}$, $L_{j,k}$ will need to consider.
For position $L_{i,j}$:

$$\begin{aligned} L_{i,j} = L_{j,i} &= \frac{-w_{i,j}}{\sqrt{d_i d_j}} \\ \frac{\partial L_{i,j}}{\partial w_{i,j}} = \frac{\partial L_{j,i}}{\partial w_{i,j}} &= \frac{-1}{\sqrt{d_i d_j}} + \frac{w_{i,j}}{2(d_i d_i)^{\frac{3}{2}}}(\frac{\partial d_i}{\partial w_{i,j}} d_j + d_i \frac{\partial d_j}{\partial w_{i,j}}) \\ &= \frac{-1}{\sqrt{d_i d_j}} + \frac{w_{i,j}(d_i + d_j)}{2(d_i d_i)^{\frac{3}{2}}} \end{aligned} \tag{4.1}$$

For position $L_{l,k}$, $L_{k,l}$, where $k \neq i \& j$ and $l = i \| j$:

$$\begin{aligned} L_{k,l} = L_{l,k} &= \frac{-w_{k,l}}{\sqrt{d_k d_l}} \\ \frac{\partial L_{k,l}}{\partial w_{i,j}} = \frac{\partial L_{k,l}}{\partial w_{i,j}} &= \frac{w_{k,l}}{2\sqrt{d_k} d_l^{\frac{3}{2}}} \end{aligned} \tag{4.2}$$