EXECUTIVE SUMMARY

This project is related to music and machine learning. In the previous project they use the self-similarity matrix to analyze the musical form. We start with their work, and extend it by adding more theoretical topics. We talk about spectral algorithm and dimensionality reduction, and their relationship to SVD. We use SVD to decompose the similarity matrix of music and analyze the meaning of it. By doing some experiments, we research the performance of our method and discuss the reason of its advantage. We also study some important kinds of SVD implementation algorithms, and compare them by designed experiments. Our software implementation can be seen as an updated version of the software in previous project by adding SVD analysis functions and some more applications.

- I researched and discussed some interesting topics in spectral algorithm and dimensionality reduction, include relationship between PCA and SVD and LSI, Laplacian Eigenmap and related problems, etc.

- I successfully used SVD in musical structure discovery. This topic has been researched by few researchers and my method performs very well for some specific kinds of music

- I implemented an updated version of the software, added SVD analysis in it and some more functions are implemented.

- I compare some popular algorithms of calculating SVD, see page 52-59

ACKNOWLEDGEMENTS

Firstly I would like to thank my supervisor Perter Flach. He is always very serious to my project and his guidance helps me solve many problems. He makes me know that math can also be as interesting as art.

And I have to thank my friends, who help me treat printing things when I am out of UK.

Finally, thanks to my parents, they are always kind to me and support me.

TABLE OF CONTENTS

1. INTRODUCTION

This section includes a statement of the aims and objectives of the project and then I explain the structure of this thesis.

1.1 Aims and Objectives

The project is related to music. The primary aim of this project is to research the methods of matrix decompositions and come up with a general method of structure discovery in music using matrix decompositions. And this includes two aspects, the theoretical analysis of the general method and the software implementation as an application of musical structure analysis.

1.1.1 Theoretical Analysis Objectives

a) Compare different methods of matrix decomposition. Discuss their advantages and disadvantages.
b) Research the mathematical meaning of SVD, include its meaning and benefit for matrix decomposition.
c) Study the different algorithm of SVD, compare their performances and complexity. Discuss the constrain conditions of them.
d) Find out how about the performance of SVD when it is used in musical structure discovery.
e) Study how to model the bar and build the self-similarity matrix.
f) Research the matrix selection to discover what will happen if we build different kinds of similarity matrix instead of self-similarity matrix.
g) Compare the known methods of musical structure analysis.
h) Discuss some other applications of the result of SVD related to musical structure.

1.1.2 Software Implementation Objectives

a) Implement the algorithm of SVD.
b) The software can model the bars and build the matrix of MIDI music.
c) It use SVD automatically analyze the structure.
d) Implement the Graphical User Interface. It can accept MIDI file as input and output a detail report of the music and a graphical matrix.
e) Users can set the thresholds and resize the graphical matrix or window. They can also change the colors.
f) Users can get some temporary data include numerical data and some graphs like heat map.
g) The software should be fast and stable. It should have good frame in the source code since this software is designed as a general system of musical structure analysis.

1.2 Structure of Thesis

Firstly we state the aims and objectives of this project include theoretical aims and software aims in Chapter 1. And we briefly introduce the structure of the thesis.

Since this project is about music, we should first study the fundamental musical background. In Chapter 2 we introduce the basic knowledge of music theory, especially the musical form knowledge. And we represent music as MIDI files in this project.

Then we consider how to connect music and math In Chapter 3 we introduce the theoretical background includes spectral algorithms and dimensionality reduction methods. We not only describe the definition and conditions of them, we also discuss the mathematical meaning of them and how they are related to musical structure discovery. And we use derivations to prove their advantages.

In Chapter 4 we survey some relevant works include music generation, the previous project [4] and musical structure analysis. Among this works, the previous project [4] is the most important one, since our project is based on the topics and software framework in this project and extent to SVD analysis.

In Chapter 5 we describe the details of how we design the experiments, include vector modeling, matrix building and SVD application. We introduce the basic experiment framework in our project and give a simple example. We also briefly introduce the software implementation which is an updated version of the previous one.

In Chapter 6 we analyze the experiment result. We discuss the parameter selection and model selection, which include how to choose the matrix and distance function. We compare the performance between our method and the other methods include the one used in the previous project [4].

In Chapter 7 we focus on SVD itself. We firstly research the mathematical meaning of SVD, and then introduce the definition and properties of it. We also list some applications of SVD except musical structure analysis. Then we detailed the algorithms of SVD. We give some important algorithm frameworks of it and briefly compare their performance.

In Chapter 8 we state the evaluation and further work. This is a short chapter since there are no too many technical topics and experiments.

2. MUSICAL BACKGROUND

Since this project is based on music, I will first introduce basic musical theory. This section includes some concepts and notations of music, and some related musical structure knowledge. There is a conceptual model of music structure below; from this graph we can overview the concept levels.

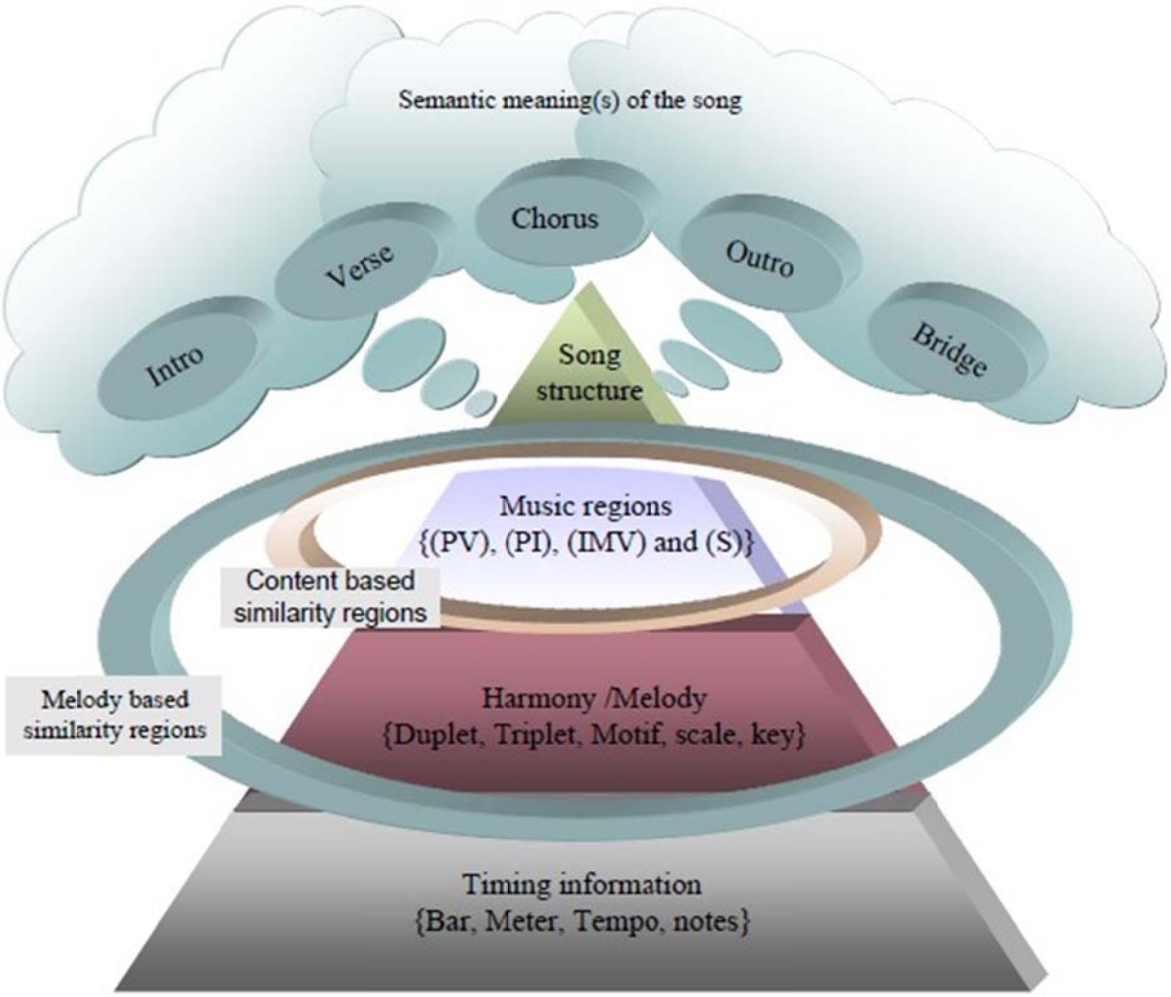
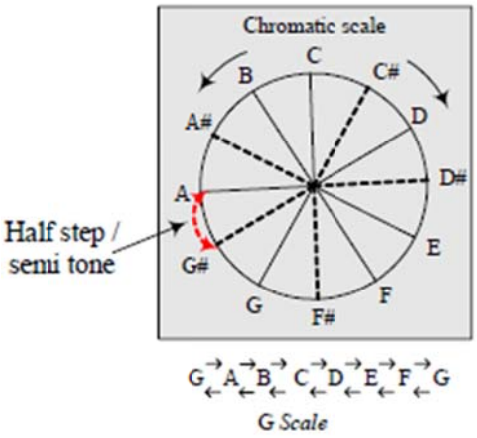


Figure 1 The framework of music structure [1]

2.1 Basic Terms

i. Pitch: How human feel the frequency of sound, how high or how low about the sound. A kind of perceptual analog of frequency.
ii. Note: A notation of pitch of a sound.
iii. Scale: A sequence of notes.



G Scale	Notes in the C - Scale							
	I	II	III	IV	V	VI	VII	I
Major	G	A	B	C	D	E	F#	G
Natural Minor	G	A	A#	C	D	D#	F	G
Harmonic Minor	G	A	A#	C	D	D#	F#	G
Melodic Minor	G	A	A#	C	D	E	F#	G

Figure 2 Succession of music notes and music Scale [2, 3]

iv. Interval (Duration): Distance between two notes, how long they last. Include Whole, Half, Quarter, Eighth, sixteenth, etc.

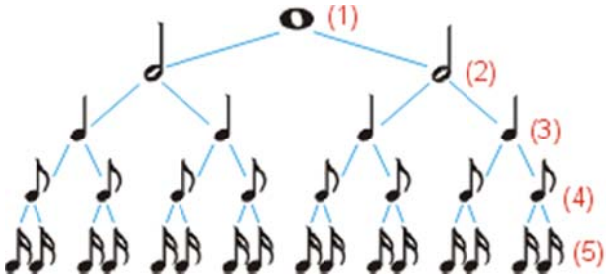


Figure 3 A hierarchical representation of Durations

v. Beat: Basic units represent pulse.
vi. Loudness: How loud human hear. A kind of perceptual analog of amplitude.
vii. Rhythm: the pattern of durations

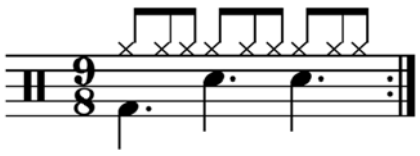


Figure 4 An example of rhythm: Compound triple drum pattern

viii. Bar (Measure): A segment of time in a duration of given number of beats. The pattern of recurring beats.

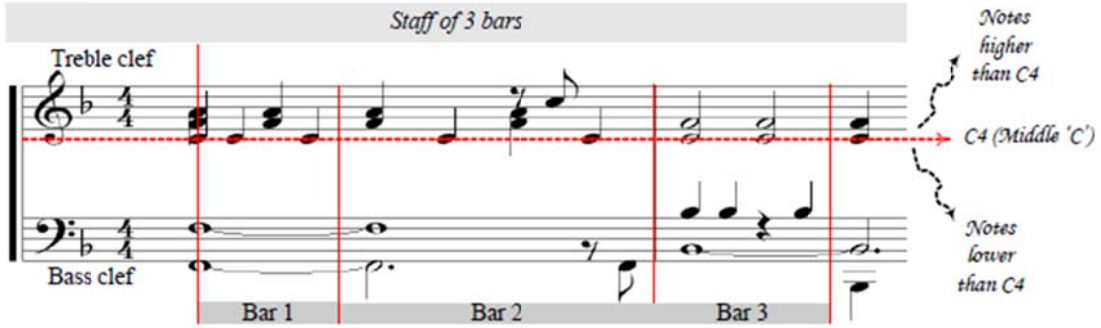


Figure 5 Three example Bars [1]

ix. Meter: type of measure, written as fraction like 4/4, 2/2, etc.
x. Chord: A set of notes
xi. Harmony: Combination of chords or some notes sound in the same time

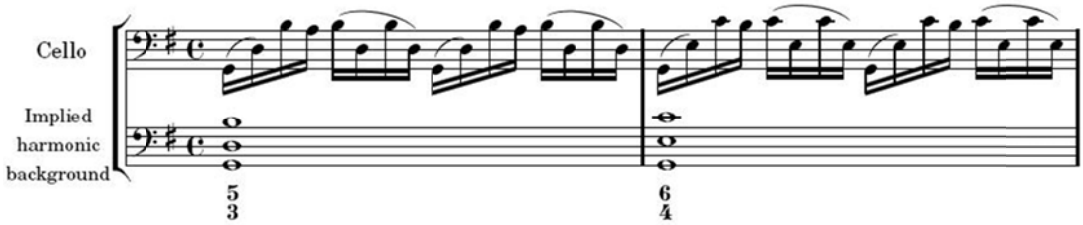


Figure 6 An example of implied harmonies in a monodic line. From J.S. Bach Cello Suite no. 1 in G, BWV 1007

2.2 Harmonic Analysis

Music is an art form which exhibit rich structure. Harmonic analysis is to find the right chord which fit a given piece of notes without giving any harmonic information, and mark the chord symbols beside the notes if necessary.

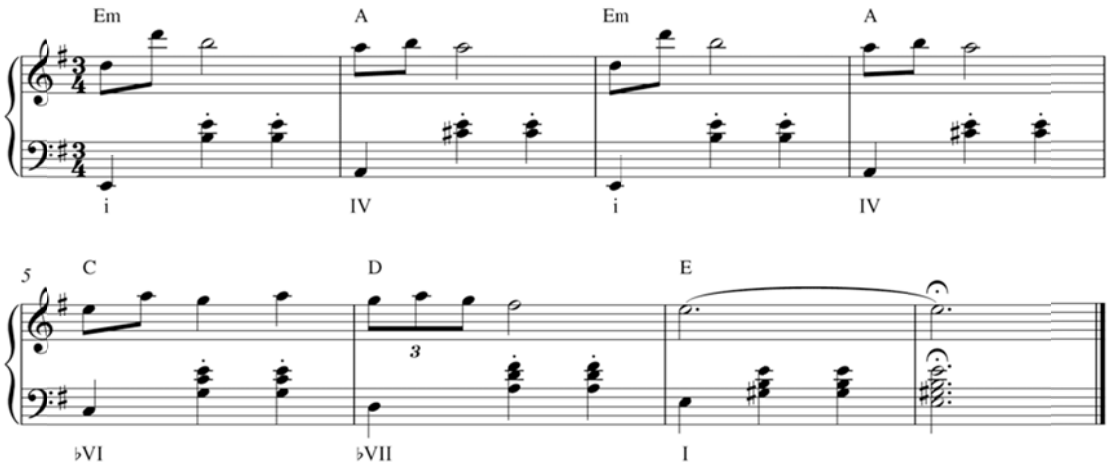


Figure 7 An example harmonic analysis of “Minuet of Forest” from <Zelda>

Harmonic analysis is usually a complex job. We need compare all the known chords and find the fittest one. It is a kind of searching of bars. It is even the branch of mathematics now, and related to Fourier series and Fourier transforms. Interestingly harmonic analysis has more applications in some other subjects like physics, signal system and neuroscience.

2.3 Musical Form

Musical form is one of the most important aspects of music structure. Simply, they are patterns which often occur in music. It describes music structure of time. There are some different kinds of music pieces; they are introduction, verse, chorus, bridge and ending. Introduction and ending often appear just one or zero time in one song, so we do not consider them. A simple example is just verse-chorus [4]. Most short pop songs are in this form. And this is actually “Binary Form”, can be written as AB.

Name	**Symbolic Representation**	**Notes**
Rondo	A B A C A D A	A common returning theme played between contrasting sections.
Binary	A B	
Rounded Binary	A B A	Only part of A played second time.
Bar Form	A A B	
Ternary	A B A'	
Theme and Variations	A A' A"	
Song Form	A A B A	B is often called the ‘bridge’.
Sonata	Exposition-Development-Recapitulation	Include Baroque, Classical, Romantic

Table 1 Some common musical forms [4]

For popular songs, their structure contains Intro, Verse, Chorus, Bridge, Middle eighth, INST-instrumental sections and Outro. Based on the similarity of the melody and content, we can get the clusters. There are many rules in Musical form, and we can find the corresponding different patterns for different songs. In the figure below there is an example.

In this project we will mainly focus on classic music. In Table 1 there are some common musical forms for classic music.

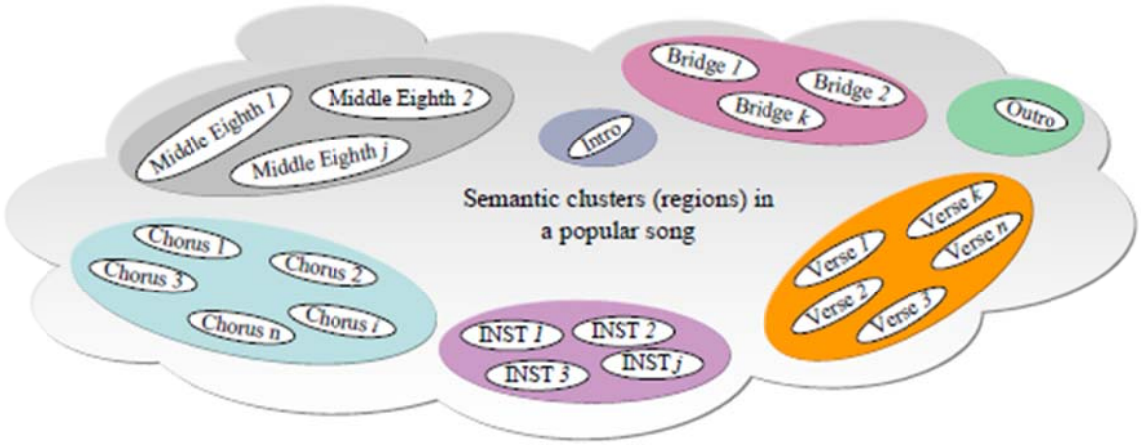


Figure 8 An example of clustering based on semantic similarity [1]

2.4 MIDI

MIDI is the short of Musical Instrument Digital Interface, it is Developed in 1983 as protocol and firstly for synthesizers to communicate/control each other

To be more general and compatible with the previous project [4], we use the standard MIDI (Musical Instrument Digital Interface) files as input. So we will never consider the audio wave signal and the noises. We can just focus on the musical notes and musical forms.

The MIDI file just includes all the events, not the waves. So we need not worry about the distortions. And we can also build some test files by typing, need not make any sound. MIDI files include notes, times, scales, tempo… and all the information we need in the project. But MIDI files do not include any signal information and wave based information since there are only instructions for real or virtual synthesizer. And we will use the MIDI reader component from the previous project [4]. Later we will discuss how to exact information and how we will process and use them.

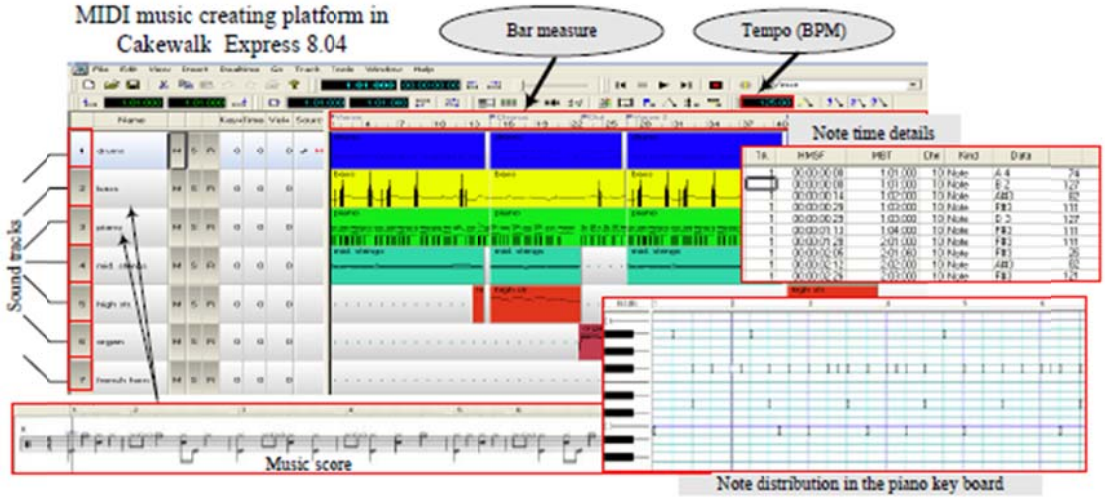


Figure 9 MIDI music generating platform (Cakewalk) [1]

In the SMF (Standard MIDI Files) files, pitch is represented by key number, and the most frequently used events are “note on” and “note off”. Here is an example of the SMF file:

```
MFile 1 2 1024
MTrk
0 TimeSig 2/4 24 8
0 KeySig 0 major
0 Tempo 500000
16385 Meta TrkEnd
TrkEnd
MTrk
0 Meta TrkName "Acoustic Grand
   Piano"
0 PrCh ch=1 p=0
0 On ch=1 n=72 v=64
0 On ch=1 n=48 v=64
1024 Off ch=1 n=72 v=0
1024 On ch=1 n=72 v=64
1024 Off ch=1 n=48 v=0
1024 On ch=1 n=60 v=64
2048 Off ch=1 n=72 v=0
2048 Off ch=1 n=60 v=0
2048 On ch=1 n=79 v=64
2048 On ch=1 n=64 v=64
3072 Off ch=1 n=79 v=0
3072 On ch=1 n=79 v=64
3072 Off ch=1 n=64 v=0
3072 On ch=1 n=60 v=64
4096 Off ch=1 n=79 v=0
```

Figure 10 An example of SMF file in text format

2.5 Others

Here are some kinds of music which have interesting structure, these special music prove that the music structure could be very logical and there should be rich information hidden in it.

Fuge: Using mathematical language, Fuge music has a fractal structure. Bach invented “counterpoint” to achieve this.

Crab Canon: By Bach again, a fantastic music with two duality parts. If you play it both from beginning to end and from end to beginning, you will find that you played exactly the same music.

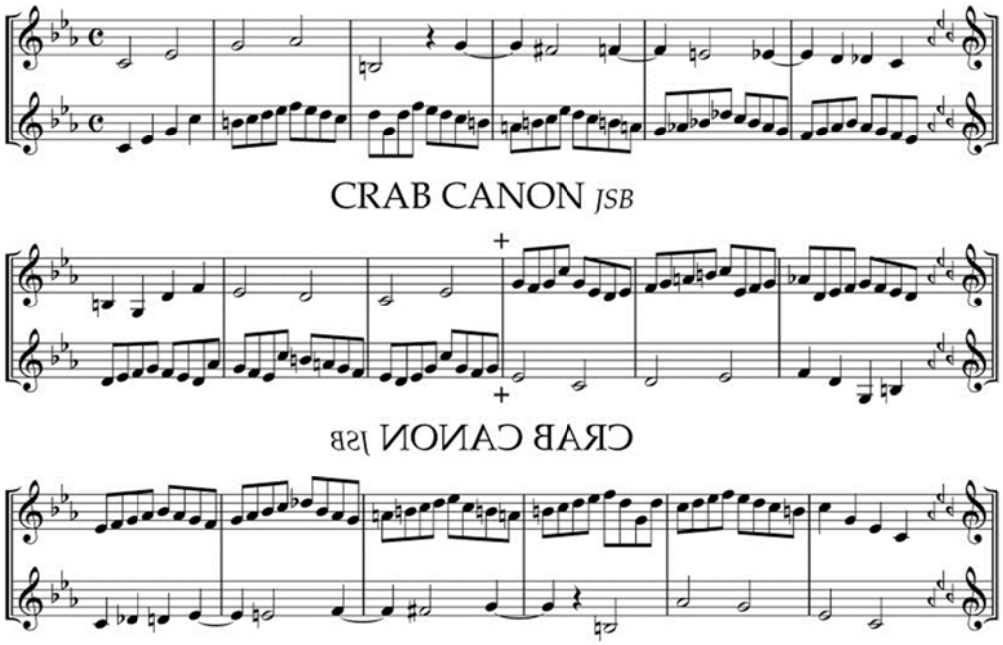


Figure 11 J.S. Bach's Crab Canon based on retrograde transposition (Hofstadter, 1980)

3 THEORETICAL BACKGROUND

3.1 Dimensionality Reduction

3.1.1 Introduction

There are many method of dimensionality reduction, and different ways are for different aims. Matrix decomposition is a kind of method which can reduce the dimension of data. There are some classic method of reducing the dimension include: “Principal component analysis (PCA)”, “Linear discriminant analysis (LDA)”, “Independent component analysis (ICA)”, “Vector quantization (VQ)”, and “Singular value decomposition (SVD)”. In the mentioned methods, the original large matrixes (which in high dimension space) are all decomposed into low rank matrixes (which in low dimension space). But the low rank matrixes have no negativity conditions. While we analyze the matrixes, we often cannot explain the negative values since it seems meaningless. So there is another special method “Non-negative matrix factorization (NMF)”. In this section I will introduce the mathematical theories about these methods.

3.1.2 Feature

In many applications of machine learning, there are different kinds of data we need to deal with, like graphs, texts, audio, video; or experiments’ data of physics, chemistry, biology and business. We certainly will not design a special algorithm for each type of data, so there are some standard formats of data in machine learning. The standard formats are called features. One common format is one point of data corresponding to one dimension in Euclidian space.

If the original data cannot be directly used, we should do feature extraction. Actually, feature extraction is implemented by dimensionality reduction. We collect the data points, and drop the less important ones, leave the most important ones. For example, in text processing, when we do sentiment analysis, one method is “Tf-idf” [5]. Tf is Term Frequency, and idf is Inverse Document Frequency. It is a weighted score to measure how important a piece of text is.

$$idf_t = log \frac{|D|}{|\{d: t \in d\}|}$$

Feature selection is actually a special case of dimensionality reduction. Tf-idf is not strictly a kind of feature selection, since it does not really drop any low weighted dimentions.

The aim for dimensionality reduction is to reduce the high dimensionality and save as many as possible the important features. One idea to measure it is reconstruction error,

which is [6]:

$$\frac{1}{N}\sum_{i=1}^{N}\|x_i - \tilde{x}_i\|^2$$

$\tilde{x}_i$ is the high dimension reconstruction of the low dimension approximation of x_i, in other words, the decompression of the compressed term. If the compression method is lossless, the reconstruction error will be equal to zero. One aim of dimensionality reduction is to let it be as small as possible.

Another method is simply use variance to measure it. For example, if we need reduce the D-dimension vectors to 1-dimension, we could maximize the variance, which is written as [7]:

$$\underset{f}{argmax}\frac{1}{N}\sum_{i=1}^{N}\left(f(x_i) - \overline{f(x_\iota)}\right)^2$$

Here f is a dimension reducing function. For 2-dimension case, the second dimension should make sure to maximize the variance when it is orthogonal to the first dimension.

When we restrict the dimension reducing function f as linear function, these two methods will lead to the same result.

3.1.3 Eigenvalue

If a vector v is an eigenvector of matrix A, it can be written as:

$$Av = \lambda v$$

Here λ is the eigenvalue of the eigenvector v. A group of eigenvectors is orthogonal. Eigenvalue decomposition is to decompose a matrix as the form [8]:

$$A = Q\sum Q^{-1}$$

Matrix Q includes the eigenvectors of matrix A. $\sum$ is a diagonal matrix includes all the eigenvalues.

The Matrix multiplication is actually a linear transformation. For example, if we look at the matrix M:

$$M = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

It is also a linear transformation like:

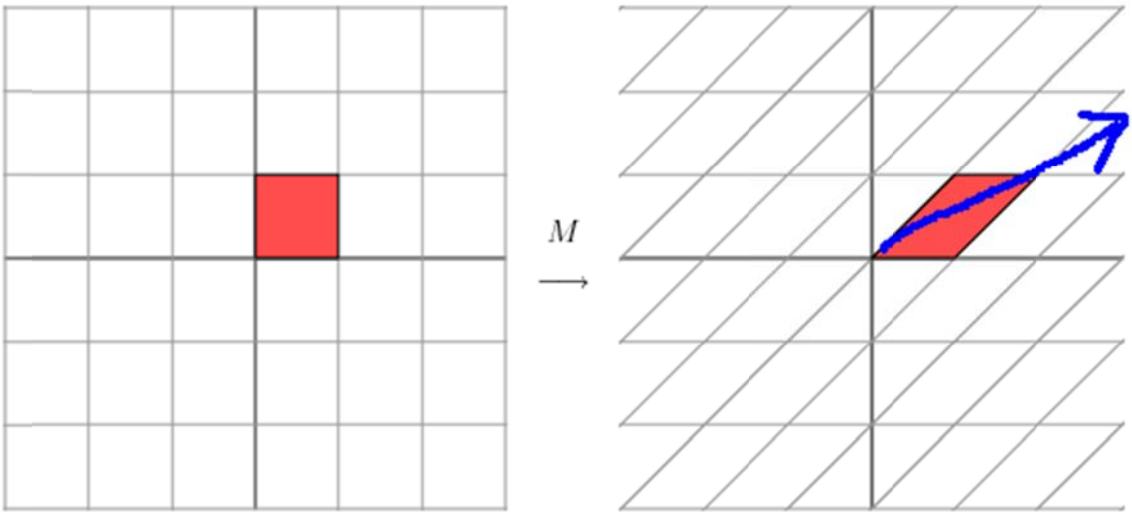


Figure 12 The graphical representation of linear transformation

This is actually a stretching of the coordinate axis in the 2-D plane. Blue arrow means the mainly direction of the transformation, sometimes we could just use the arrow represent the complex transformations.

So, the eigenvectors mean the stretching "arrows" and the eigenvalues corresponding to the eigenvectors mean the importance of the arrows.

3.1.4 Vector Quantization (VQ)

VQ is widely used in signal processing and data compression. Actually, in the ISO standard JPEG and MPEG-4, they all include the VQ step [9]. It is more like a kind of coding than a kind of dimensionality reduction.

VQ codes the sets of points, use the subsets of them to represent the original sets. It performs fast but the accuracy is low, and this method has too much limitation for our project.

3.1.5 Linear discriminant analysis (LDA)

LDA is a kind of supervised learning. In some paper it is also called "Fisher's Linear Discriminant", since it is invented by Ronald Fisher in 1936 [10]. In many kinds of Bayes methods, we may need use some probability data like prior and posterior probability. The word "Discriminant" means a model needs no probability methods to train and predict data. It is a very classic and popular algorithm in machine learning and data mining area.

The basic think of LDA is projecting the tagged data (point) to lower dimension space, and the projected data are separated into clusters. It is a kind of linear classifier. For a K-classification problem, there are k linear functions like [10]:

$$y_k(x) = w_k^T x + w_{k0}$$

When for all j, we have $y_k > y_j$, we can say that x belong to classification k.

This function is actually a kind of projection. It projects a high dimension point to a high dimension line. When k=2, see Figure 13 [4]:

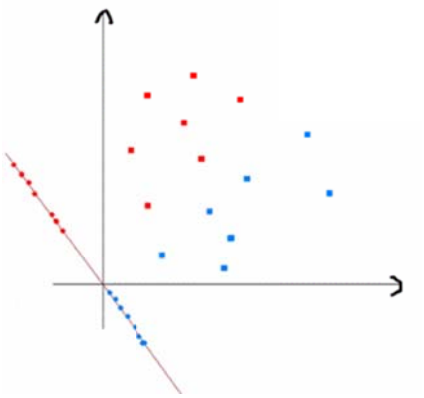


Figure 13 Linear Classifier when k=2

Red points are in Class A and Blue points are in Class B, we project them into a high dimension line, and we can see that it separated as the original classification in the line.

First let us focus on k=2 situation. Assume the projection function is [10]:

$$y = w^T x$$

The LDA algorithm requires the distance between different classes is far. So we need define some key points.

The original central point of Class i is [11]:

$$m_i = \frac{1}{n_i} \sum_{x \in D_i} x$$

Here D_i means the points that belong to Class i.

After the projection, the central point of Class i is [11]:

$$\tilde{m}_i = w^T m_i$$

And the variance of the points in Class i is [11]:

$$\tilde{s}_i = \sum_{y \in Y_i} (y - \tilde{m}_i)^2$$

Now we can get the loss function of the projection in LDA [11]:

$$J(w) = \frac{|\tilde{m}_1 - \tilde{m}_2|^2}{\tilde{s}_1{}^2 + \tilde{s}_2{}^2}$$

The denominator is an addition of variances, which mean the dispersion of the points in the sets. And the numerator is the distance between central points, so maximize J(w) we can get the best w.

Now we define a matrix about the dispersion degree of the original data [12]:

$$S_i = \sum_{x \in D_i} (x - m_i)(x - m_i)^T$$

Then the denominator is [12]:

$$\tilde{s}_i = \sum_{x \in D_i} (w^T x - w^T m_i)^2 = \sum_{x \in D_i} w^T (x - m_i)(x - m_i)^T w = w^T S_i w$$

$$\tilde{s}_1{}^2 + \tilde{s}_2{}^2 = w^T (S_1 + S_2) w = w^T S_w w$$

And the numerator is [12]:

$$|\tilde{m}_1 - \tilde{m}_2|^2 = w^T (m_1 - m_2)(m_1 - m_2)^T w = w^T S_B w$$

So the loss function can be formed as: [12]

$$J(w) = \frac{w^T S_B w}{w^T S_w w}$$

Now we can use the Lagrange multiplier method. We limit the length of the denominator as 1 to avoid infinite solution.

$$c(w) = w^T S_B w - \lambda (w^T S_w w - 1)$$

$$\Longrightarrow \frac{dc}{dw} = 2 S_B w - 2 \lambda S_w w = 0$$

$$\Longrightarrow S_B w = \lambda S_w w$$

Now the problem is an eigenvalue problem and easy to solve.

Now consider K-Classification (k>2) problem, we have the similar form [13]:

$$S_W = \sum_{i=1}^{c} S_i$$

$$S_B = \sum_{i=1}^{c} n_i (m_i - m)(m_i - m)^T$$

$$S_B w_i = \lambda S_w w_i$$

This is also an eigenvalue problem, if we solve the i-th eigenvector, it is the correspondingW_i.

Eigenvalue problem is a general problem, its time cost is $O(D^3)$. We will not discuss the detail of the algorithm.

3.1.6 Principal component analysis (PCA)

PCA has a close relationship of LDA. The difference is that input data for PCA is not tagged. So it is an unsupervised learning method.

We consider two ways to get the same PCA algorithm. First we just consider maximize the variance. Assume we still project a point into a vector, the original central point is [13]:

$$\bar{x} = \frac{1}{N} \sum_{n=1}^{N} x_n$$

If we defineu_1 is the vector we project to, the variance is [13]:

$$\frac{1}{N} \sum_{n=1}^{N} \{u_1^T x_n - u_1^T \bar{x}\}^2 = u_1^T S u_1$$

Similarly, use the Lagrange multiplier method [13]:

$$u_1^T S u_1 + \lambda_1 (1 - u_1^T u_1)$$

After the derivation, we have:

$$Su_1 = \lambda_1 u_1$$

This is an eigenvalue problem, u_1 is the eigenvector and λ is the eigenvalue. Maximizing it is indeed finding the maximum λ_1.

Another way is to minimize the loss. First we consider a D-dimension space, assume there are D vectors which are orthogonal, we have [13]:

$$x_n = \sum_{i=1}^{D} \alpha_{ni} u_i$$

By using approximation method, after the projection we have:

$$\tilde{x}_n = \sum_{i=1}^{M} z_{ni} u_i + \sum_{i=M+1}^{D} b_i u_i$$

Now we have reduced the dimension to M, and the loss function is:

$$J = \frac{1}{N} \sum_{n=1}^{N} \|x_n - \tilde{x}_n\|^2$$

To minimum this function, we let [13]:

$$\frac{\partial J}{\partial z_{nj}} = 0 \Longrightarrow z_{nj} = x_n^T u_j$$

$$\frac{\partial J}{\partial b_j} = 0 \Longrightarrow b_j = \bar{x}^T u_j$$

Then:

$$x_n - \tilde{x}_n = \sum_{i=M+1}^{D} \{(x_n - \bar{x}) u_i\} u_i$$

$$J = \frac{1}{N} \sum_{n=1}^{N} \sum_{i=M+1}^{D} (x_n^T u_i - \bar{x}^T u_i)^2 = \sum_{i=M+1}^{D} u_i^T S\, u_i$$

By using the Laplace multiplier method [14], we get:

$$Su_i = \lambda_i u_i$$

It is the same as we get in the first way.

As an example, let us consider a 2-D space, in Figure 14 it is a represent of PCA. The dotted lines are the vectors we project to. pc1 means the maximum eigenvector and pc2 means the second maximum eigenvector, they are orthogonal.

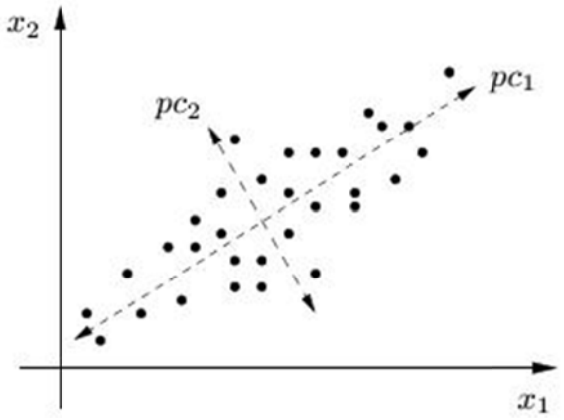


Figure 14 A 2-D PCA example [15]

PCA is widely used in many areas. But it is a linear transformation. One extended method is Kernel PCA, which use Kernel trick to extend PCA to nonlinear problems. And maximizing the covariance will not always valid.

3.1.7 Singular value decomposition (SVD)

In PCA and LDA we introduce them based on eigenvalues. But SVD is a method to decompose matrix based on singular values. It includes obviously physical meaning. It represents the original matrix as the multiplication of some smaller matrixes and approximation can be used. In machine learning area, many topics include PCA, LSI, data compression are related to SVD.

In the SVD chapter we will discuss the details of it.

3.1.8 Summary

We decide to use SVD method, since it is stable and fast. In this project we can use this method to analyze the similarity matrix. The singular values imply the patterns in the music structure. Since we have such a powerful tool the plan of increasing the size of the matrix is also considerable.

3.2 Spectral Clustering

3.2.1 Introduction

We know there have been many methods of clustering. Classical methods include K-means, GMM, etc. Generally, all the algorithms which use SVD or eigenvalue decomposition can be called as spectral algorithm. It has many advantages comparing to classical algorithms:

i. Spectral algorithms only requires the similarity matrix (similar to K-medoids algorithm), the data represents in N-Dimension space is not required as in K-means
ii. Since spectral algorithms focus on the major features, and ignore the unimportant features, it is more stable and robust than the classical algorithms. It is not very sensitive to the irregular error in data. And many experiments have proved that spectral algorithms have better performance. Actually, in most of the comparison experiments, K-means algorithm is used as the baseline.
iii. The algorithmic complexity is better than K-means, especially for high dimensional video/audio/image data. K-means is simple, but it is NP-hard, using EM (Expectation-Maximization) iterative method may fall in local optimum. Theoretically SVD has the unique solution, and it also has many support theoretical result and surrounding mathematical properties.

There are many branches in spectral algorithms:

i. PCA/LDA as dimensionality reduction: which we mentioned in previous sections.
ii. Spectral Clustering/Embedding: which will be discussed later
iii. Low-rank approximation

And here is an example experiment result of comparing the performance between K-means and spectral clustering from [19]:

k	TDT2		Reuters-21578	
	K-means	SC	K-means	SC
2	0.989	0.998	0.871	0.923
3	0.974	0.996	0.775	0.816
4	0.959	0.996	0.732	0.793
...				
9	0.852	0.984	0.553	0.625
10	0.835	0.979	0.545	0.615

Figure 15 Selected experiment result of K-means and spectral clustering [19]

TDT2 and Reuters-21578 are widely used text dataset. We can obviously see that spectral clustering is much better than K-means.

Mathematically, clustering problem is equivalent to Graph Partition problem [20], which means how to partition the vertex set into some disjoint subsets given a graph $\mathrm{G} = (\mathrm{V}, \mathrm{E})$, and meanwhile the partition is "best". The two difficult points are:

i. It is difficult to define "best". Some rules are mentioned in [16]
ii. Even though we have defined a good rule, it is also difficult to optimize it.

There are many ideas about the problem i:

a) Normalized Cut (Graph Cut problem), and some variation like Ratio Cut and Min/max cut
b) Minimum conductance which is highly related to Algebraic Graph Theory [16]
c) Algorithms without general rules, but can be proved [17]
d) Not based on graph, but reformulate known clustering methods, to let the problem be able to solved by SVD [18]

And here is a simple framework of spectral clustering:

a) Construct a graph, each vertex in the graph corresponding to a data point. Connect the similar points. The weight of the edge indicates the similarity between the points. Write the adjacency matrix as W
b) Sum each row of W and get N numbers. Put them on the diagonal line to construct a $N \times N$ matrix, written as D and let $L = D - W$
c) Calculate the first k eigenvalues of L as $\{\lambda\}_{i=1}^{k}$ and corresponding eigenvectors $\{v\}_{i=1}^{k}$
d) Combine the k eigenvectors together to construct a $N \times k$ matrix, treat each row as a vector in k-dimension space, and use K-means to cluster them. The result is so related to the original data points.

3.2.2 Laplacian Eigenmaps

Laplacian eigenmaps (LE) [21] is the key of spectral clustering. It is actually a kind of Dimensionality reduction method, spectral clustering uses it to reduce the dimension, and then do K-means. This method only requires the similarity matrix, but also requires that the similarity matrix S should have some local properties, which means, for example, if point i is too far away from point j, S_{ij} should be equal to zero. There are two methods for the local properties:

i. Use a threshold, and set the similarity value to zero if it is lower than the threshold. This means we consider locality in ϵ-domain

ii. For each point, select the nearest k points as its neighbors. And set the similarity values to other point to zero. Note that LE requires the similarity matrix as symmetric matrix. If i is one of the k nearest neighbors of j, or/and j is one of the k nearest neighbors of i, we keep the S_{ij} value. Otherwise set it to zero.

After we build S, we consider reducing the dimension. We start with the simplest case, to reduce the dimension down to 1, as $x_i \to y_i$. So we can minimize the following function [21]:

$$\sum_{ij} (y_i - y_j)^2 W_{ij}$$

The meaning of this function is: If x_i is close to x_j, W_{ij} will be large. After mapping, if there are large difference between y_i and y_j, it will be amplified by W_{ij}. Minimizing the function makes sure that the mapped points will not be too far away from each other when the original points are close.

Let D be a diagonal matrix which is the sum of each row of W, and $L = D - W$ is called Laplacian matrix [21]. We should solve the eigenvalue problem:

$$Lf = \lambda Df$$

Obviously the smallest eigenvalue is zero. The smallest eigenvalue except zero corresponding to eigenvector is the result of projection. The eigenvector is N-Dimension column vector. After rotation, it is the result of projecting N-Dimension original data to one-Dimension.

Extend to M-Dimension case, the only difference is we need to select the smallest M eigenvalues except zero. A Matlab code (use KNN to build the similarity matrix) of this algorithm is in Appendix A

Actually, Laplacian eigenmap assumes data is distributed on a low dimension manifold nested in the high dimension space. And Laplacian matix L is a discrete approximation of manifold Laplace Beltrami operator. We will not discuss the detail of it; just consider an interesting example Swiss Roll.

Swiss Roll is just like doughnut. It could be treated as a 2D manifold nested in 3D space. In Figure 16 we can see it in the left. In the right there are some randomly selected points.

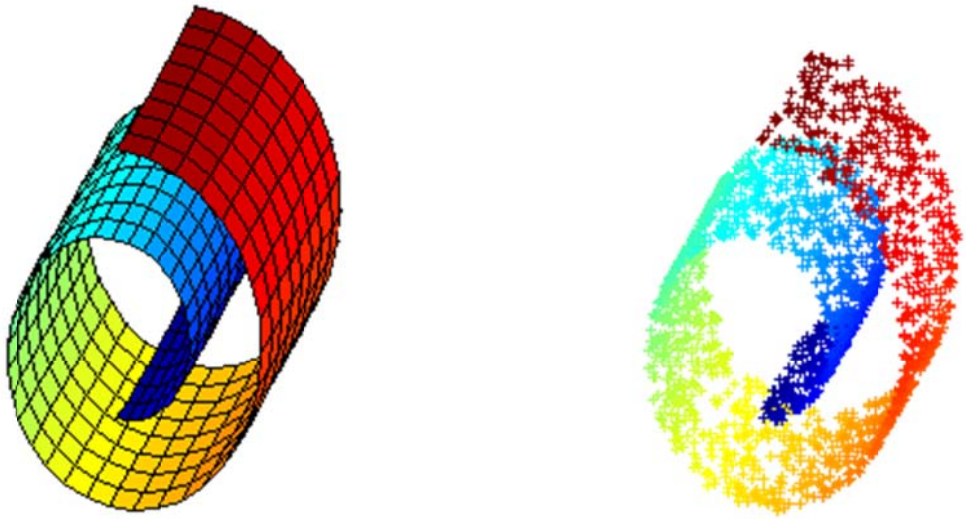


Figure 16 Swiss Roll and random points

Now we use both Laplacian eigenmap and PCA to reduce the dimension of Swiss Roll. We can see that in the graph LE can easily project the different points to different place, but PCA project the points to a mixed up graph.

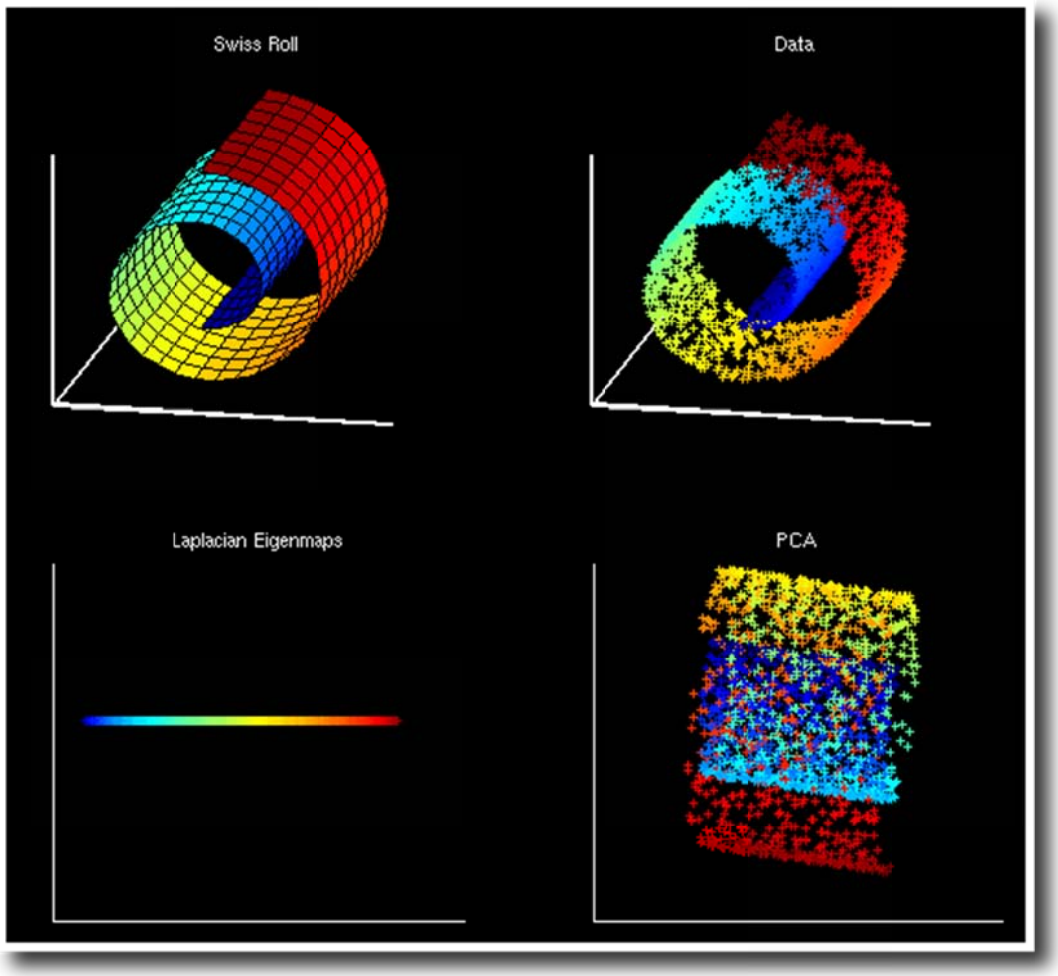


Figure 17 PCA vs LE on Swiss Roll

Otherwise, there is another dimensionality reduction method called Locally Linear Embedding [22]. It also uses the assumption of manifold as LE, and assumes the smooth manifold has local linear property. The points can be refactored by the local near points. It firstly minimizes:

$$\sum_i \left\| x_i - \sum_j W_{ij} x_j \right\|^2$$

For solving the best local linear refactored matrix W. W_{ij} will be zero if point i is far from point j. Then we minimize:

$$\sum_i \left\| y_i - \sum_j W_{ij} y_j \right\|^2$$

Here y is the vector after projection. If x_i can be refactored, the data after projection should also keep this property. After some transformation, the solving method is similar to LE, which is to solve a eigenvalue problem. The theoretical comparison of LE and LLE can be found in [23], and here is the experiment about their performance on Swiss Roll.

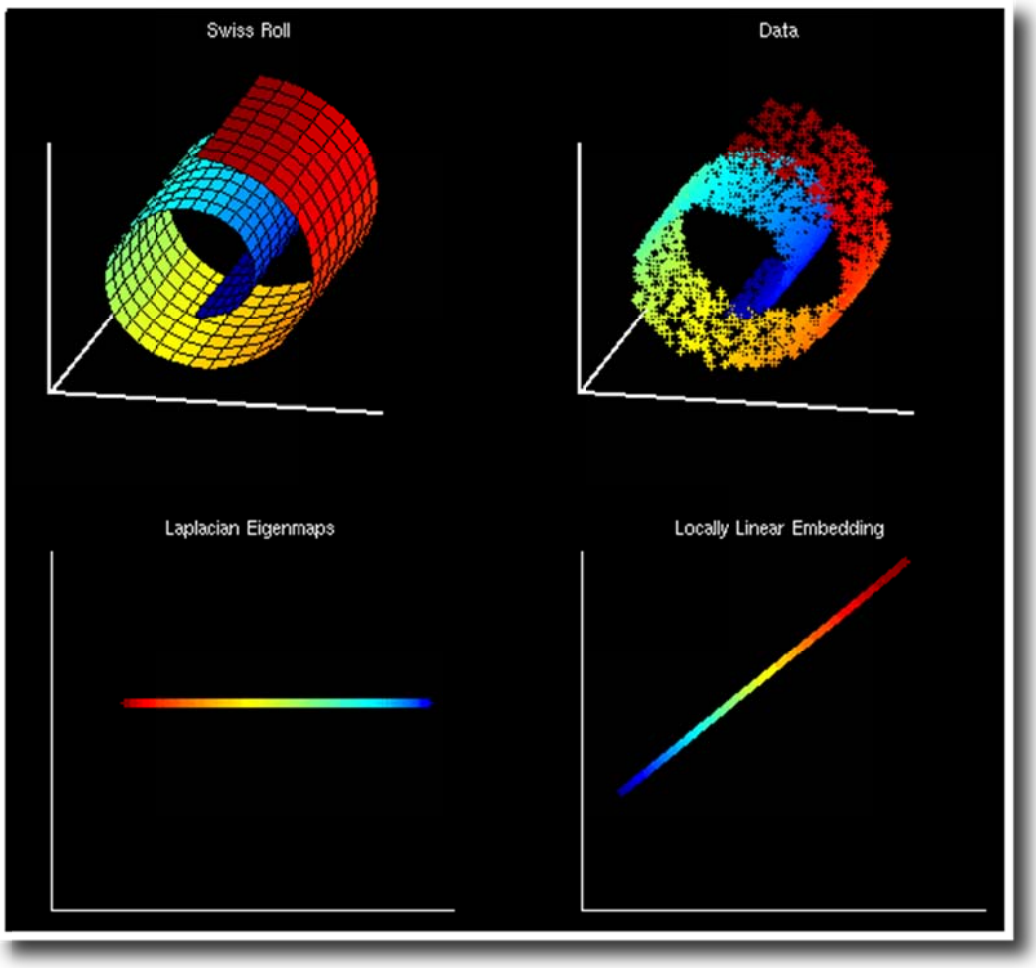


Figure 18 LLE vs LE on Swiss Roll

Note that LLE and LE are all nonlinear methods. And the result of PCA is a projection matrix. But LLE and LE directly give the result of data, without the "dimensionality reduction function". For new data, they cannot directly use a function to deal with. So people found some linear LE and LLE forms, called Locality Preserving Projection and Neighborhood Preserving Embedding.

In LPP, the dimensionality reduction function is a linear transformation, represented as a dimensionality reduction matrix A. So the function of LE becomes [23]:

$$\sum_{ij}\left(A^T x_i - A^T x_j\right)^2 W_{ij}$$

Similar to pervious work, the eigenvalue problem we need to solve is [22]:

$$X^T L X a = \lambda X^T D X a$$

And get the sorted eigenvectors to build projection matrix A. In LE, the eigenvalue problem has sparseness, so we can efficiently solve it when only several smallest eigenvalues are needed. But here after multiplying X, it is not sparse again. To solve this problem we can use a method called Spectral Regression [24]. If we use Kernel trick to non-linearize LPP, we will go back to LE.

Otherwise, although LE is an unsupervised method, we can still use the tag information. When we build the similarity matrix, the element belong to the same tag will have higher similarity value.

In a word, LE is fast, but not always performs best. One special advantage is: when outlier occurs, its robustness is especially good.

3.2.3 Graph Cut

Graph cut problem is a hot point in spectral clustering. Briefly, it is to cut some edges, reconstruct the graph into some independent sub-group. The sum of the weight of the cut edges is called Cut-value. For example, imagine segmentation problem will be equivalent to cutting the graph into sub-graphs, and we require minimizing the Cut-value. In music processing, the music segmentation problem is similar, also can be treat as Graph Cut problem.

Actually, Minimum cut is the most famous problem of Graph Cut. But sometimes the original minimum cut is not appropriate for practical problem, since the measure

methods are different. For this reason, there are also many different methods, like Ratio Cut, Normalized Cut, etc.

Now let us use some formula to describe Graph Cut problem [25]:

First represent the graph as adjacency matrix, written as W, and w_{ij} is the weight of the edge between vertex i and vertex j. If no edge, set the weight to zero. Let A and B be two subsets (without intersection) of graph. So the Cut-value can be defined as:

$$Cut(A, B) = \sum_{i \in A, j \in B} w_{ij}$$

For the simplest case, if we cut the graph into two parts, Minimum cut requires minimizing $Cut(A, \bar{A})$, which $\bar{A}$ means the complement of A. But in this case there are some isolated points. So we also define RatioCut [20]:

$$RatioCut(A, B) = \frac{Cut(A, \bar{A})}{|A|} + \frac{Cut(A, \bar{A})}{|\bar{A}|}$$

And NormalizedCut [20]:

$$NCut(A, B) = \frac{Cut(A, \bar{A})}{vol(A)} + \frac{Cut(A, \bar{A})}{vol(\bar{A})}$$

Here $|A|$ means the size (number of element) of A, and $vol(A) = \sum_{i \in A} w_{ij}$. So Adding this terms can avoid the isolated points and the cut will be more average. One PAMI paper [20] use Normalized cut in image segmentation.

RatioCut and NormalizedCut have very close relationship with spectral clustering. But they are still NP-hard. So we can do some deformation [20].

Let V be the set of all the vertex in graph. Firstly define a N-Dimension vector as f:

$$f_i = \begin{cases} \sqrt{\frac{|\bar{A}|}{|A|}} & if \ v_i \in A \\ -\sqrt{\frac{|A|}{|\bar{A}|}} & if \ v_i \in \bar{A} \end{cases}$$

Remember that in spectral clustering framework, we defined a matrix $L = D - W$, its name is Graph Laplacian. And it is not the only matrix called this name, many authors called their matrix the graph Laplacian.

L has a propertiy [20]:

$$f'Lf = \frac{1}{2}\sum_{i,j=1}^{N} w_{ij}\left(f_i - f_j\right)^2$$

Related to the pervious definition of f, we can have [20]:

$$\begin{aligned} f'Lf &= \sum_{i,j=1}^{N} w_{ij}\left(f_i - f_j\right)^2 \\ &= \sum_{i\in A, j\in \bar{A}} w_{ij}\left(\sqrt{\frac{|\bar{A}|}{|A|}} + \sqrt{\frac{|A|}{|\bar{A}|}}\right)^2 + \sum_{i\in \bar{A}, j\in A} w_{ij}\left(-\sqrt{\frac{|\bar{A}|}{|A|}} - \sqrt{\frac{|A|}{|\bar{A}|}}\right)^2 \\ &= 2Cut(A,\bar{A})\left(\frac{|\bar{A}|}{|A|} + \frac{|A|}{|\bar{A}|} + 2\right) \\ &= 2Cut(A,\bar{A})\left(\frac{|A| + |\bar{A}|}{|A|} + \frac{|A| + |\bar{A}|}{|\bar{A}|} + 2\right) \\ &= 2|V| \cdot RatioCut(A,\bar{A}) \end{aligned}$$

Define I as the vector which all the elements in it are equal to 1, we have: $f'I = \sum f_i = 0$ and $\|f\|^2 = \sum f_i^2 = n$, since $|V|$ is constant, minimizing RatioCut implies minimizing $f'Lf$, with two conditions $f \perp I$ and $\|f\| = \sqrt{n}$

And we define Rayleigh quotient as [20]:

$$R(A,x) = \frac{x'Ax}{x'x}$$

The maximum and minimum values of it are the largest eigenvalue and smallest eigenvalue of matrix A. When x equal to the corresponding eigenvector, it reaches the extreme value. Since $f'f = \sqrt{n}$ is constant, minimizing $f'Lf$ implies minimizing $R(L,f)$. But the smallest eigenvalue of L is zero, so the related eigenvector is just I. This does not satisfy the condition $f \perp I$, so we use the second smallest eigenvalue and the corresponding eigenvector v.

Till now, it seems that we have solved the NP-hard problem, but this is just a trick: The original problem is NP-hard since the vector f can be just assigned by two value, $\sqrt{|\bar{A}|/|A|}$ and $-\sqrt{|A|/|\bar{A}|}$, so that it is a discrete problem. But in our trick, the vector v can be any real number, which means we relaxed the restrictions. To get the original solution, a simple method is to pay attention on the sign symbols of the terms

of v, and corresponding plus and minus symbols to the dicreate terms $\sqrt{|\bar{A}|/|A|}$ and $-\sqrt{|A|/|\bar{A}|}$. A more complex method is to use K-means as $k = 2$, to cluster v into two groups.

In brief, what we have done is to first calculate the eigenvalues and do K-means to find the clusters. Actually, if we generalize the binary case to k-class problem, we can get the simple flow of spectral clustering: calculate the eigenvalues and select the first k small values, arrange the corresponding eigenvectors to construct matrix and cluster them by row using K-means. Here we use a short Matlab code to present spectral clustering flow:

```
function idx = spectral_clustering(W, k)
    D = diag(sum(W));
    L = D-W;

    opt = struct('issym', true, 'isreal', true);
    [V dummy] = eigs(L, D, k, 'SM', opt);

    idx = kmeans(V, k);
end
```

So now we have an approximation solution, it always performs well in practical application, but there is no support thesis which has strictly proved this point. Moreover, Normalized Cut and Markov chain have very close relationship [26], the measure method can be treat as a random walk in graph.

3.2.4 Other Topics

3.2.4.1 Minimum conductance

Minimum conductance is another rule of graph partition, which seems similar to Normalized cut. But the analysis method is different.

Minimum conductance is a constant in graph, it is named Cheeger constant [27] in Algebraic Graph Theory. It traverse all the partition (A, B), get the value of:

$$\frac{Cut(A,B)}{min(vol(A), vol(b))}$$

It seems that it would be more difficult than Normalized Cut, since there is a $min()$

function. But interestingly, in Algebraic Graph Theory there is a conclusion, which claims that, the second eigenvalue of the Laplace matrix on the graph is an approximation of Cheeger constant [25].

Let Cheeger constant be C [27]:

$$2C \leq \lambda \leq \frac{C^2}{2}$$

So we can use SVD to solve the last step. This approximation performs worse than Minimum conductance, but the difference is not too much, and there are strict theoretical boundaries of it in [16].

3.2.4.2 Reformulate the classical clustering methods

Spectral clustering is interesting since not only there are many algorithms about it, but also the thinking of it. It combines SVD into optimization problem, which is named spectral relaxation. For example, [18] used this thinking to reformulate K-means algorithm. He discretizes the continuity and relaxes the problem, such that it can be solved by SVD finally.

3.2.4.3 Others

For some methods, it is hard to describe their rules, and some authors only gave the algorithms and proofs.

[16] gives a recursive algorithm, keep cutting the graph into two parts, until the terminal conditions are satisfied. It can deal with K-class problem. And [16] also gives the proof of boundaries which cannot be calculated for Normalized Cut.

[17] gives a method to solve K-class problem directly. It does not include the rules of the aim function, but it uses the Matrix perturbation theory to prove the algorithm works in some cases.

In [28], spectral algorithm is used to cut the point sets as trees, and use other rules (like K-means) to merge the leaf. So that it cluster the data. And for trees, dynamic programming can be used to solve NP-hard problems.

Semi-definite Programming is another way, if we relax some problems on SDP, the result will be better. But in [18], we cannot find the difference by the experiments. And SDP is much more complex.

4 RELEVANT WORKS

4.1 The Previous Project

This project is based on the previous project “Finding harmonic structure in music files” by Tom Lyner [4]. Some of the ideas are from his project, include the 12-dimensional vector, note and bar modeling, similarity matrix, using MIDI files, etc. In our project, we are more focus on the decomposition of matrix part, research of the SVD method. We hope by using SVD, we can add more functions and do more experiments about music structure, not just focus on the harmonic similarity, because SVD can break many limitations in the previous project [4]. Tom had implemented software by JAVA, so we also decide to use JAVA for the implementation, and then we can reuse the basic part of Tom’s software. Since our project is more about the SVD method, we may not cost too much time on implementation of the software, and we may also use Matlab, R language or other tools to do some independent experiment and some quick tests.

4.2 Music Generation

In the early time, music generation was the primary topic in music information retrieval area. Programs are created to generation different kinds of music. And some composers used computer as a music band, they created and computer played. But people found that music generation was actually the most difficult problem in music information retrieval area. Most programs can just make the similar music or mix some different piece of music to try to create a “new” music.

Generating accompaniment then was mentioned. It is easier since it just need computer listen to the music and generate some music that fit it. Since computer is fast, so we can let it real time generate the accompaniment. The harmonic structure analysis will occur in this work, computers see the music as structure include patterns.

4.3 Music structure analysis

People can listen music and “feel” it, what about computers? Some people had tried to make computer to be able to “feel” the music, not just treat music as data. Dannenberg (2002) tried to let computer classify the different style of the music, and the techniques he used is from machine learning area. Y Shiu and H Jeong(2006) used the similarity matrix; they analyze the music by Viterbi Algorithm, and they can recognize some chords in the music. Many people have work on similarity matrix, like Dannenberg and Hu’s (2002). They use the matrix to find the clusters. Foote and Cooper’s (2001) built a visualization of similarity matrixes; they first used FFT to filter the signal and then tried to find the self-similarity part in the music. Our project can treat this relevant work as reference, and focus on how to construct a general method of structure discovery in music.

5 METHOD

In this section we describe and discuss the details of our experiments and software implementation.

5.1 Introduction

In previous project [4], we decide to use bar as the basic unit, since a bar can be represented as a chord, that means a bar is seen as a harmonic “idea”. By using the method by Paiement et. al. (2005) we consider representing a note as a 12-dimensional (one for every semitone) space vector. So we can also represent the whole bar as a vector addition of all the notes in the bar. While every bar is just a vector, we can calculate the distance of each pair of the bars. Then we can build a similarity matrix which measures the difference of these bars. And we can find some patterns in this matrix.

We decide to keep this method at first, but this method has some problem, we will try to improve it. It mixes all the notes in one bar as one vector. That is good for reducing the size of the matrix. But the notes in a bar lost their order information. It is just like in each bar we press all the notes in the same time. This may induce losing much information. So we can try to use note or segment or other things as the basic unit to construct the matrix. Another problem is the method to compute distance. In previous project we use Euclidean distance [4]. In this project we can try some more kinds of distance.

And we will use SVD as a matrix decomposition method to analyze the musical matrix. This is the main innovation point of this project, since in previous project [4] the segmentation and clustering process are done by simple algorithms and the result of SVD is very interesting to musical structure discovery. We can use it to cluster the segments based on the segment-segment self-similarity matrix. We research some applications of SVD in musical matrix decomposition, and analyze the mathematical meaning and musical meaning of them. And we study and analyze the details of algorithms of SVD, to know the theoretical derivation of SVD and why it can be used in musical structure discovery. We also design some experiments and survey the performance of SVD.

There are some interesting topics we discuss. The first is matrix selection. We know that many applications about music analysis are based on matrix, so different matrixes are used for different researches. For example, Segment-Segment self-similarity matrix can be used for clustering, but Song-Note similarity matrix can be used for musical retrieval. Other topics include distance function selection and vector space selection, by using different spaces and distance function; we can solve different problems about music.

5.2 Modeling

As that mentioned in the previous project [4], when we listen to one octave, there are actually many frequencies in it instead of one (which we generally believe in). And we can fold harmonics. We decide to use the method that representing each note by 12 semitones with different strength in the octave. An distribution graph of the strength of the 12 dimensions are shown in Figure 19:

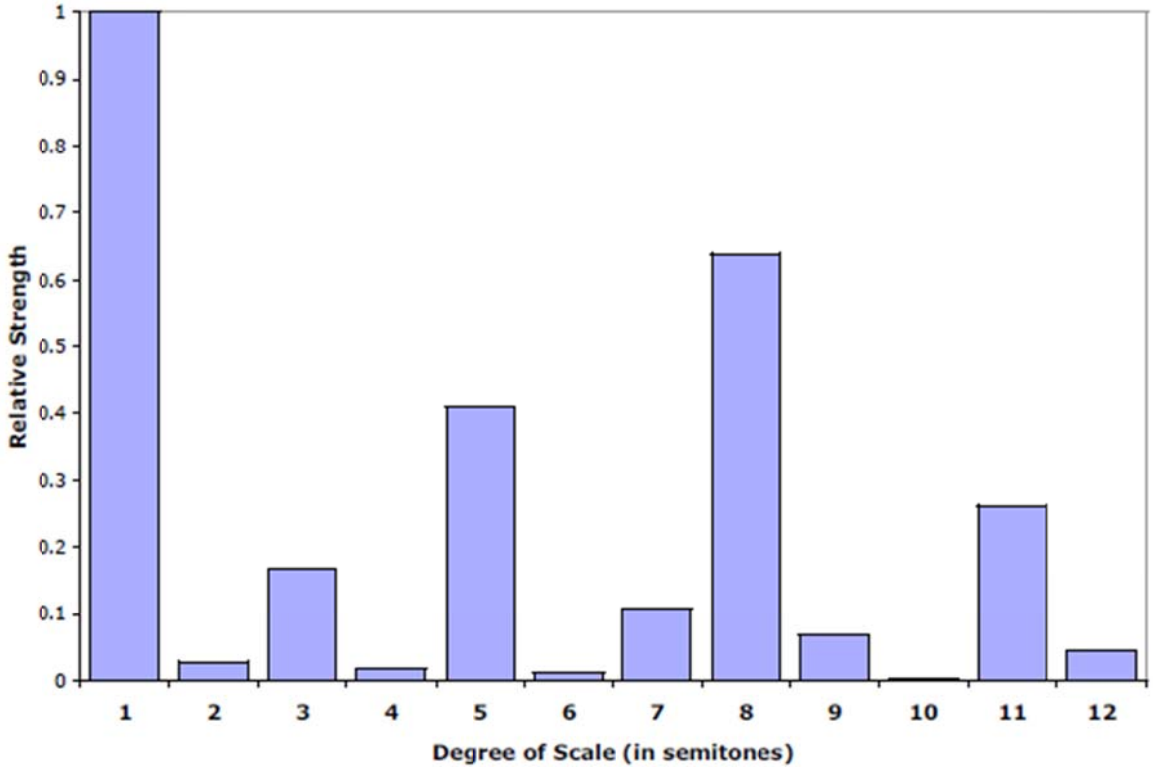


Figure 19 A distribution of the strength of the 12 dimensions [4]

Then we can define a 12-dimensional space [4]. In this space each note can be represented as a vector. The first dimension of the space is note C, and the second is note C#, etc. In the illustration we can see note C and not D as example.

Note that the vector of note D is very similar to which of note C. It just shifted two positions to right. The highest value appears in C for note C and in C# for note C#, etc. So there are totally 12 kinds of vectors.

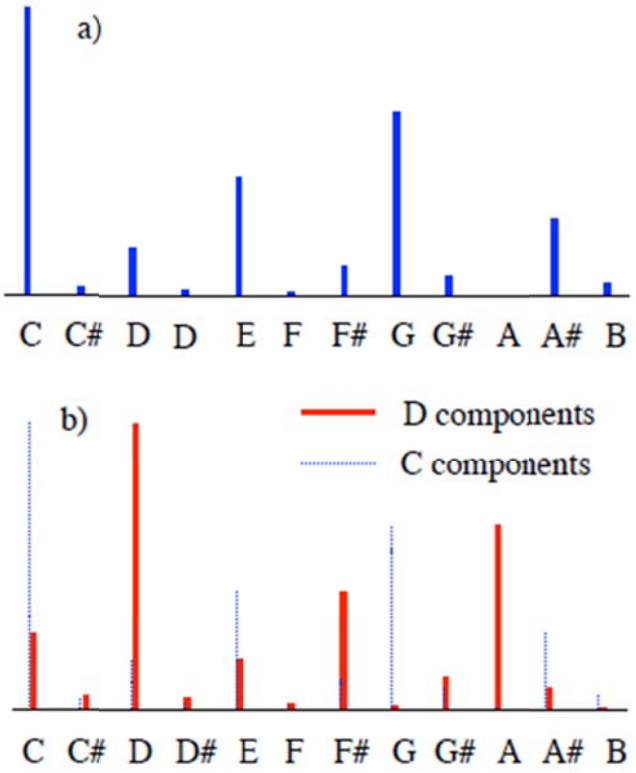


Figure 20 They represent the vectors of Note C in a) and Note D in b) [4]

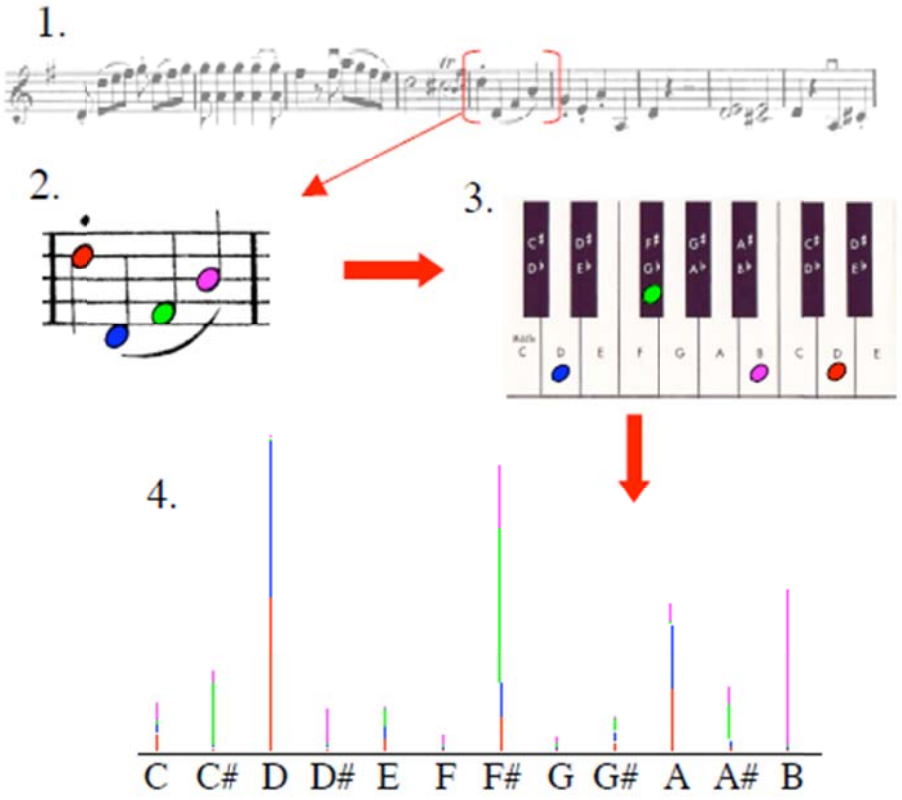


Figure 21 The selected notes build a bar's vector [4]

If we want to use bar as the basic unit, we can also model the bars. Here we simply consider only the notes information without timing information. So we add all the notes in a bar together to calculate the vector of a bar, as which shown in Figure 21. The long notes and the short notes will be treated equally.

And here we have another choice, we can consider not represent the notes as distributions, just leave the strongest semitone as 1, set others to 0, and do some experiments to compare the difference.

5.3 Building the matrix

By using the Euclidean distance function we can calculate each pair of the bars in one song. Here is the function: [4]

$$D(x,y) = \sqrt{(x_1-y_1)^2 + (x_2-y_2)^2 + \cdots + (x_n-y_n)^2} = \sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}$$

So we can try to build a self-similarity matrix. The self-similarity matrix compares the musical sequence with itself. If we define n is the length of self-similarity matrix, the matrix is $n \times n$. The value of (x, y) in the matrix is the comparison between x and y by calculating the distance or similarity. Each basic unit in the sequence will be compared to every unit include itself [4].

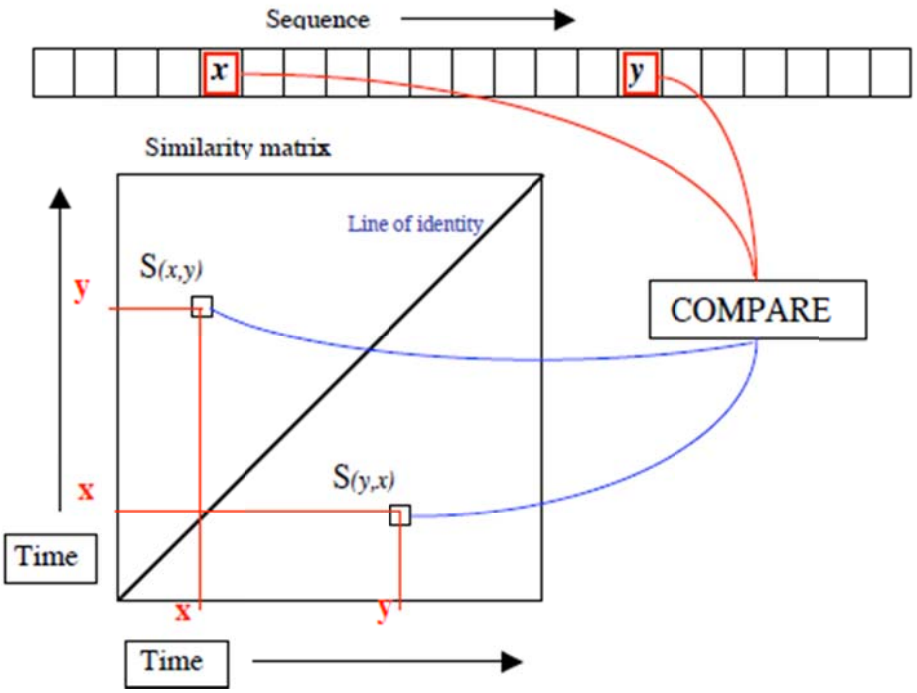


Figure 22 A diagram of comparison between two basic units x and y [4]

The matrix then is symmetrical along the diagonal line. So we only need half of the memory to keep the matrix. The reasons of using similarity matrix is we can easily see the patterns by eyes, and this kind of matrix can be analyzed be using many mathematical tools.

We have mentioned that we will try some other distance functions. There are many distance functions used in different subjects like Euclidean distance, Hamming distance, Chebyshev distance, Minkowski distance, etc. We will discuss and compare them in the discussion section.

We will also try to build some other matrix, not just self-similarity matrixes. For example we can compare too different songs instead the same songs, or search a small piece of music in the database.

5.4 Clustering

As a simple example, we will research the similarity based clustering of segments.

5.4.1 Segmentation

To cluster the music, segmentation will be done firstly. There are many algorithms about audio/music segmentation. Here we just use a simple method. Note that there are checkerboard patterns [29] when the segments change. In other words, the boundary between the segments will show as a checkerboard pattern. To find all the patterns in the matrix, we use a matched filter by using a Gaussian-tapered checkerboard kernel [30], an example is shown in Figure 23.

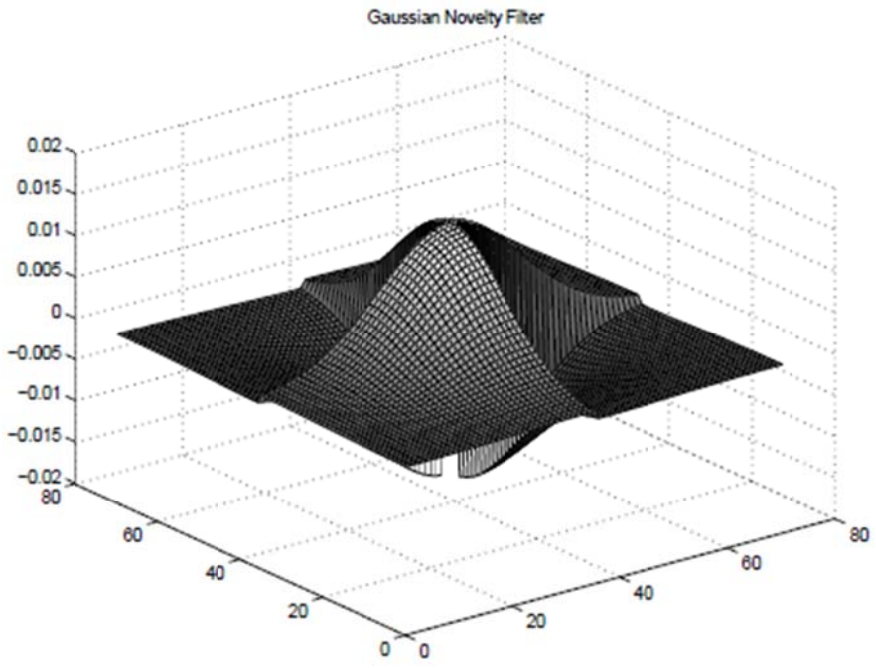


Figure 23 An example checkerboard kernel [29]

By using the kernel along the diagonal of the matrix, we can calculate the "novelty value" which means how novelty the local pieces are. And Figure 24 shows a result of the calculation by the kernel for time-indexed sequence.

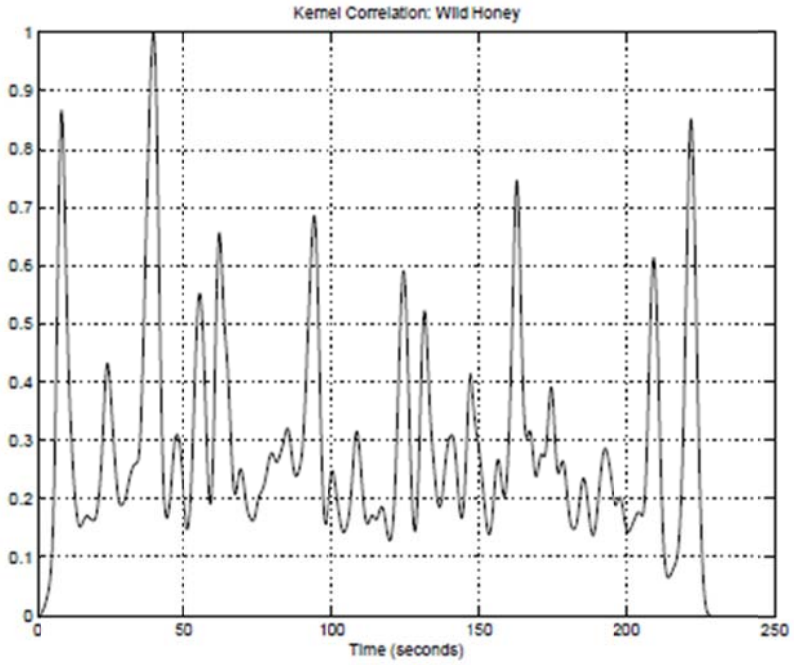


Figure 24 Novelty score computed by the kernel on the similarity matrix [29]

Since this method only requires the diagonal strip of the matrix, we do not need to keep all the original matrix, but construct a resized $N \times K$ matrix $\hat{S}$ [31]:

$$\hat{S}(i,l) = S\left(i, i+l-\left\lfloor\frac{K}{2}\right\rfloor\right) \quad i = 1,..,N, \;\; l = 1,..,K$$

Actually we do this in a "lag domain" to save time and memory. When K is small and appropriate, the reduction is very appreciable.

5.4.2 Application of SVD

In section 5.3, we talk about matrix building. We use bars as the basic units. And we use cosine distance as the distance measure:

$$d_{cos}(v_i, v_j) = \frac{\langle v_i, v_j \rangle}{|v_i|\,|v_j|}$$

Of course some other methods are also valid, such as non-negative exponential distance measure:

$$d_{exp}(v_i, v_j) = \exp\left(1 - d_{cos}(v_i, v_j)\right)$$

For our method, cosine measure is good enough. After we calculate the self-similarity matrix, we can detect the boundaries of segments. But the lengths of segments are different, so we have to design a method to measure the similarity between the variable segments.

We need to design a segment-indexed similarity matrix [31], which measures the similarity of segments, written as S_S. This similarity is computed by the Kullback-Leibler (KL) distance of the Gaussian density. The KL distance of B-dimensional normal densities $\mathbb{G}(\mu_i, \Sigma_i)$ and $\mathbb{G}(\mu_j, \Sigma_j)$ is [31]:

$$d_{KL}\left(G(\mu_i, \Sigma_i) \parallel G(\mu_j, \Sigma_j)\right)$$

$$= \frac{1}{2} log\left(\frac{|\Sigma_j|}{|\Sigma_i|}\right) + \frac{1}{2} Tr(\Sigma_i {\Sigma_j}^{-1}) + \frac{1}{2}(\mu_i - \mu_j)^t \Sigma_j^{-1}(\mu_i - \mu_j) - \frac{B}{2}$$

Tr means the matrix trace. Because $\mathrm{d}_{KL}()$ is nonsymmetrical, so we modify it to another form which is symmetrical. We just add two KL distance terms together and the sum is [31]:

$$\hat{d}_{KL}\left(G(\mu_i, \Sigma_i) \parallel G(\mu_j, \Sigma_j)\right)$$

$$= d_{KL}\left(G(\mu_i, \Sigma_i) \parallel G(\mu_j, \Sigma_j)\right) + d_{KL}\left(G(\mu_j, \Sigma_j) \parallel G(\mu_i, \Sigma_i)\right)$$

$$= \frac{1}{2} Tr(\Sigma_i {\Sigma_j}^{-1}) + \frac{1}{2} Tr(\Sigma_j {\Sigma_i}^{-1})$$

$$+ \frac{1}{2}(\mu_i - \mu_j)^t (\Sigma_j^{-1} + \Sigma_i^{-1})(\mu_i - \mu_j) - B$$

So that we can represent segments by mean μ and covariance $\sum$. And define the similarity of segments p_i and p_j as [31]:

$$d_{seg}(p_i, p_j) = exp\left(-\hat{d}_{KL}\left(G(\mu_i, \Sigma_i) \parallel G(\mu_j, \Sigma_j)\right)\right)$$

It can be equal to the value between zero and one. And it is symmetrical now, which is required by the similarity matrix construction. Now the matrix can be built as [31]

$$S_S(i, j) = d_{seg}(p_i, p_j)$$

$$i, j = 1, .., P$$

By using SVD, we can write the similarity matrix S_S as the form [32]:

$$S_S = U\Lambda V^t$$

Here we will not discuss the meaning of SVD and other details. We put these in the discussion section. Note that U and V are all orthogonal. Matrix Λ only includes the singular values sorted along the diagonal line. So singular value $\lambda_i = \Lambda_{ii}$. We can now write the form as the sum of sub matrixes [32]:

$$S_S(i,j) = \sum_{p=1}^{P} \lambda_p \, U(i,p)V(j,p)$$

And we can define a series of matrix B_p as [31]:

$$B_p = \lambda_p U(i,p)V(j,p)$$

$$\text{So } S_S(i,j) = \sum_{p=1}^{P} B_p$$

To cluster the segments, we calculate the measure function b_p [31]:

$$b_p(j) = \sum_{i=1}^{P} B_p(i,j) \qquad p,j = 1,\dots,P$$

The character p means the p-th group. So there are totally P groups. The value of $b_p(j)$ measures the similarity of the segment j to all the segments in the p-th group. This thinking is very similar to K-means, which also measure the distances from each point to each cluster. So we can indicate the segment j to group (cluster) p which has the highest value of $b_p(j)$.

Till now we can give a simple algorithm description [32]:

Algorithm 1 Clustering by SVD

1. Calculate and construct a Bar-Bar self-similarity matrix S
2. Build a segment-indexed similarity matrix S_S
3. Use SVD algorithm to decompose the matrix S_S and calculate all $b_p(j)$
4. Indicate the segment j to group c when:

$$c = \underset{p=1,..,P}{ArgMax}\, b_p(j)$$

5.4.3 Example Experiment

Let us use a piece of music as an example of this algorithm. We choose *Bach- Suite No. 1 in G major - BWV 1007* since it is short and well structured. First we build the self-similarity matrix:

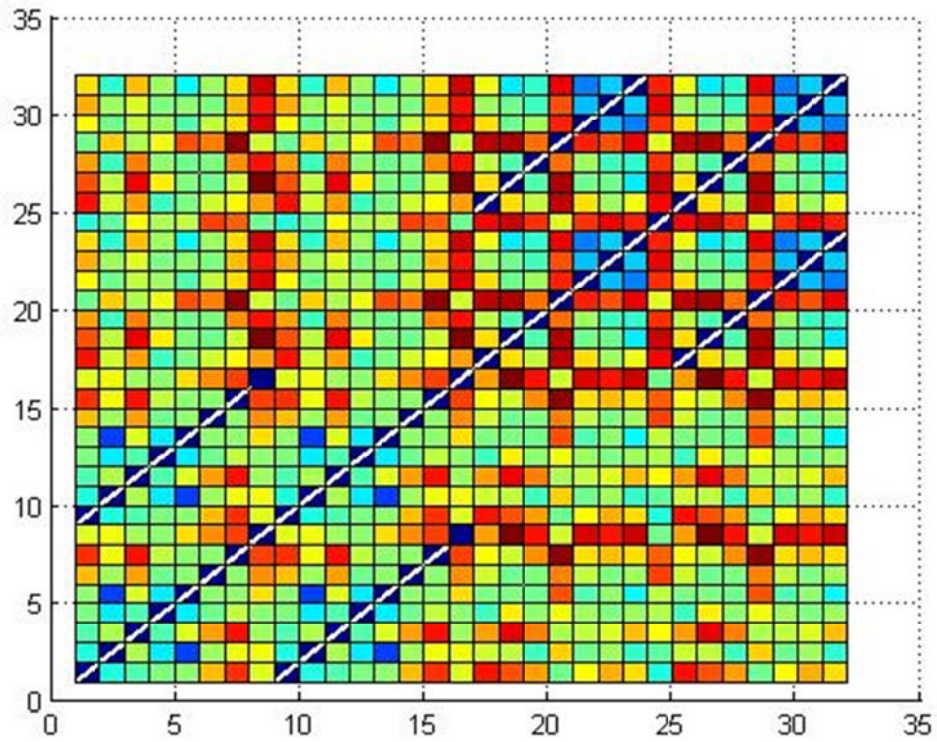


Figure 25 The self-similarity matrix of *Bach- Suite No. 1 in G major - BWV 1007*

Since the first several sub-matrixes can represent the major of the original matrix, here we only select the first 7 sub-matrixes which are marked as $A_1, A_2, A_3, A_4, A_5, A_6, A_7$ and sorted by the singular values. This can also be seen as a kind of low dimensional approximation of the original matrix.

Note that the first sub-matrix sometimes can be ignored since the most important component may not represent the latent meaning of data. But in our project, the first component is also valid, since it is based on the quadratic programming problem, and similar to K-means, each component with significant singular value should be considered.

By watching the seven graphs in Figure 26, we can see that each sub-matrix has its pattern, which indicate to the similar segments. The patterns are significant in first two or three groups, and for A_6 and A_7 the patterns become vague. That is because the singular values are decreasing when the numbers of sub-matrixes are increasing.

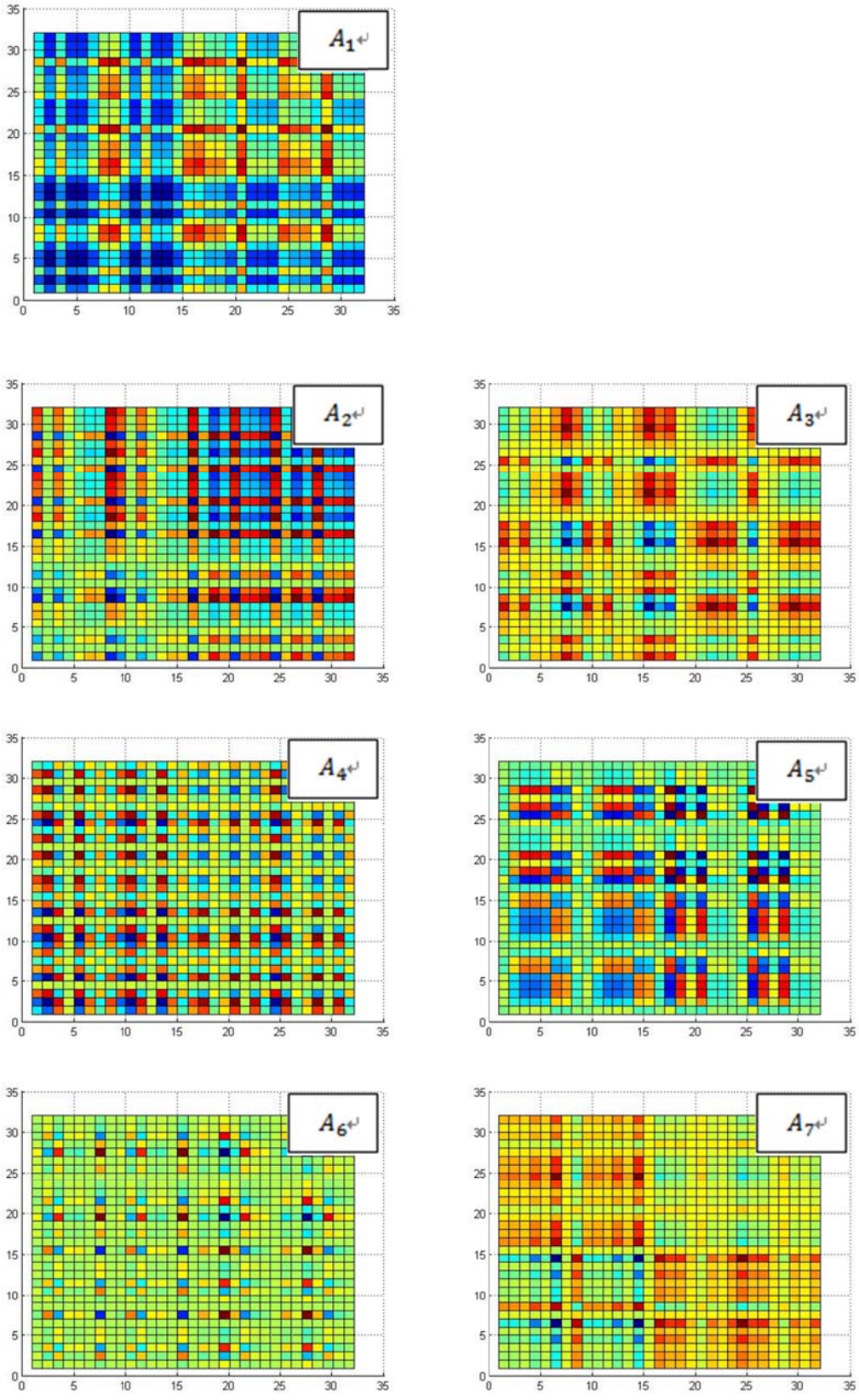


Figure 26 The first 7 sub-matrixes computed by SVD

And by constructing the matrix B_p and matrix $b_p(j)$ using the method described in Algorithm 1, we can calculate the measures of the "clustering scores". Here we only watch b_p values related to A_2, A_3, A_4, A_5 in Figure 27.

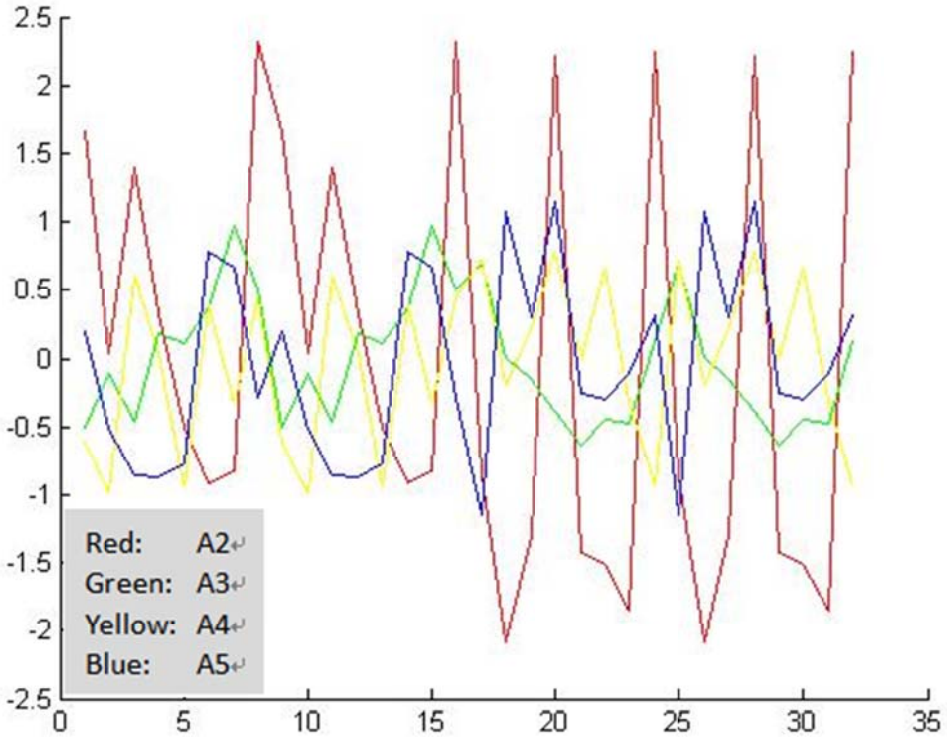


Figure 27 The b_p values of A_2, A_3, A_4, A_5

And we can now finish the clustering. We just indicate each segment to the lowest value in the graph.

5.4.4 Software Implemetation

The software is based on the previous project [4] by Tom Lyner. He implements the basic framework of MIDI input and graph drawing function. Since this project is not mainly focus on the software implementation, we implement an updated version of the software.

Figure 28 shows the Class diagrams of the basic framework form Tom Lyner's project. The detailed description can be found in his report [4], and we will not discuss too many implementation details in this report. But the implementation algorithm will be analyzed in the discussion section.

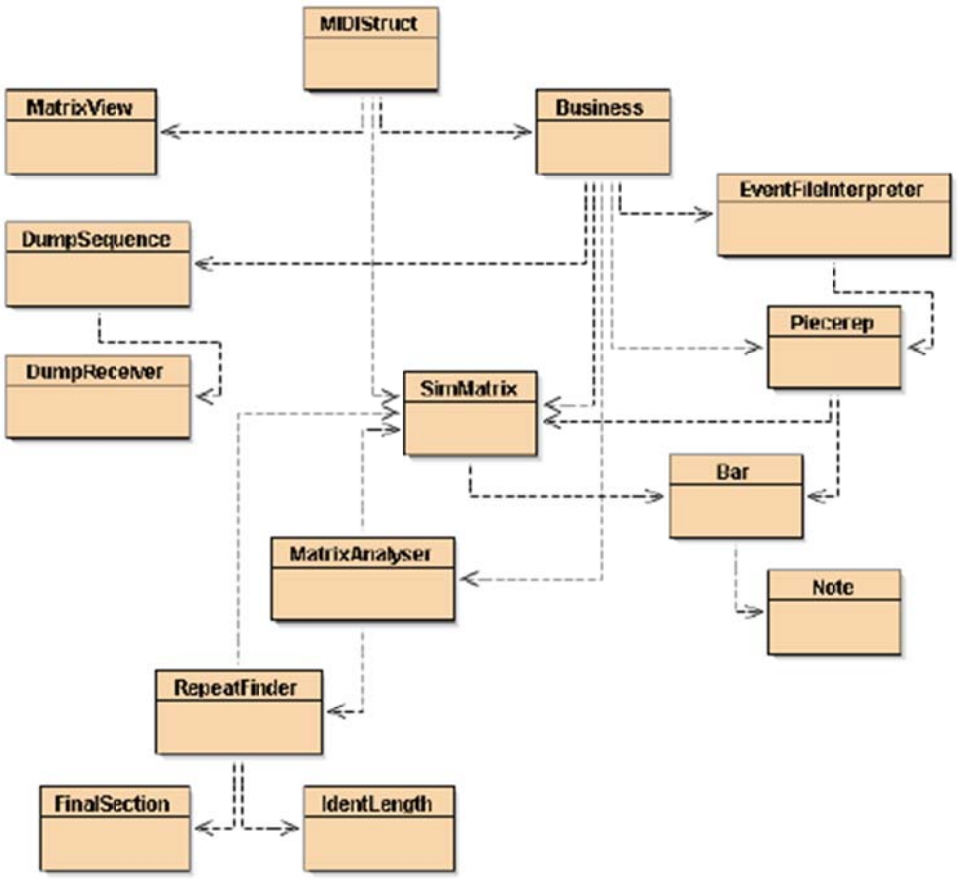


Figure 28 Class diagram for the software [4]

And we redesign the GUI to show SVD sub-matrixes, as shown in Figure 29:

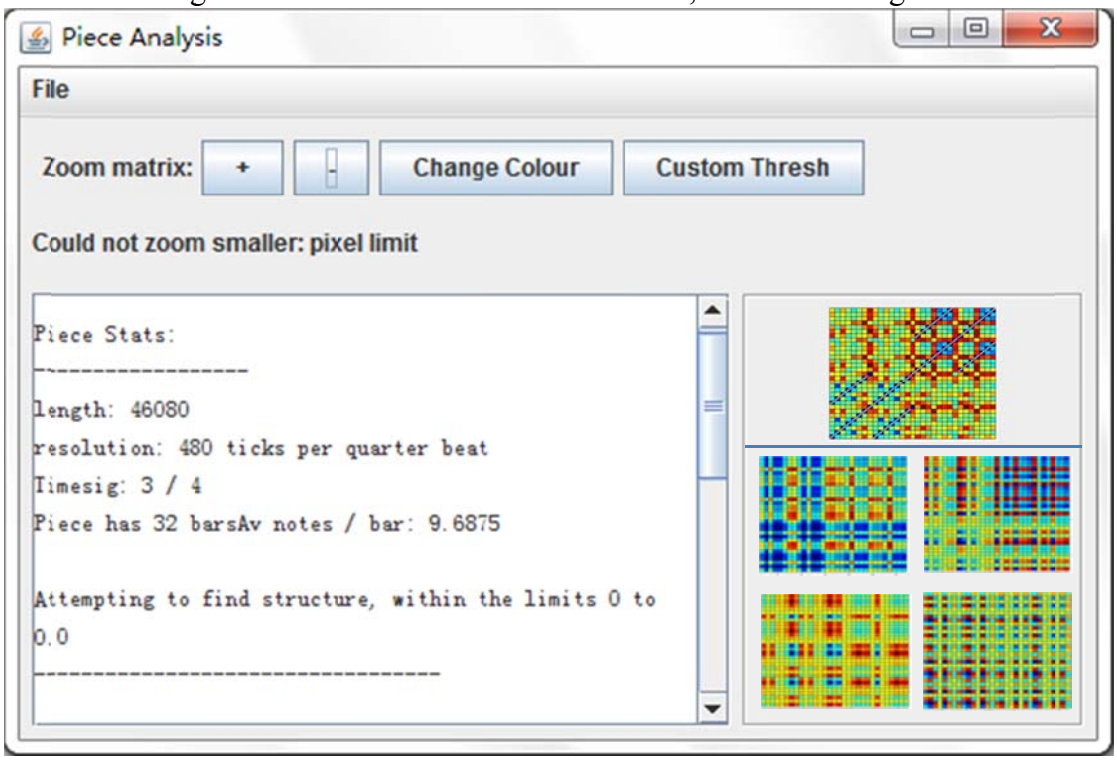


Figure 29 The GUI of the software

5.4.5 More Applications

There are many other applications of SVD in musical structure discovery. In this section we will study some of them.

5.4.5.1 Musical Retrieval

We know that in text retrieval, there is a method called Latent Semantic Indexing (LSI). It uses SVD to decompose the document-term matrix and the text searching can be easily done. We consider use the variation of this method in musical structure discovery.

In LSI, we construct the document-term matrix. For our project, we could use a Bar-Feature matrix, like the form shown in Figure 30:

Feature / Bar	C	C#	D	…
Bar 1	…	…	…	
Bar 2	…	…	…	
Bar 3	…	…	…	
…				

Figure 30 The form of a Bar-Feature matrix

So we can use SVD to compose the matrix and calculate the latent properties, and project the Bars to a lower dimensional space as shown in Figure 31 to do the retrieval. This can be used in the music database, or music piece search.

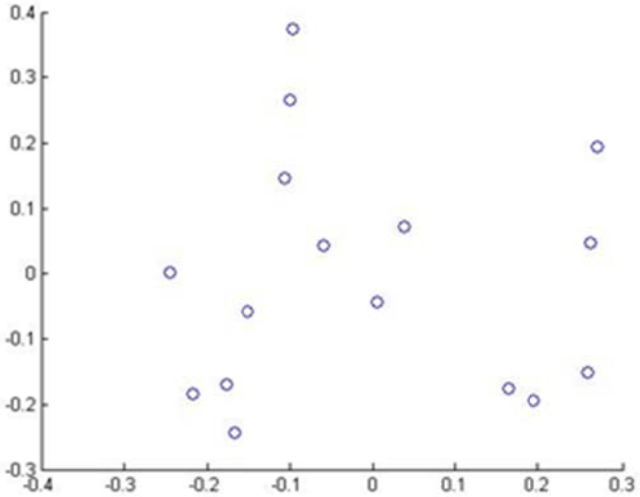


Figure 31 The lower space after the dimensionality reduction by SVD

5.4.5.2 Automatic Summarization

After we get the clusters of the segments by using algorithm 1, we can design a method to summarize the pieces. In first step we select the two clusters with first two biggest singular values. For major cluster i, we can define index j_i^* by calculating the terms of b_i [31]:

$$j_i^* = \underset{j=1,\ldots,P}{ArgMax}\, b_i(j)$$

In other word, we can summarize each cluster by using this index. And if we can use only one segment represent each cluster, we summarize the whole song. This can also be used to predict the type of clusters, like verse, chorus, etc.

5.4.5.3 Audio Segmentation

In our project the segmentation process are done by a simple method. Actually, SVD can also be used to segment the music. In the paper [33] they mentioned a method. This method use Markov Normalization to convert the matrix to probability matrix, and use SVD to compose the self-similarity matrix.

5.4.5.4 Others

In above we have introduced many applications about SVD in music structure analysis. Note that the main different of them is the matrix. Using different matrix we could solve different problems. For example, Bar-Bar and Segment-Segment self-similarity matrix are used for clustering. Bar-Feature and Segment-Feature matrix are used for musical retrieval. Song-Note can be used to musical feature extraction. In our project we will do experiments most about clustering, which is an extension of the previous project [4].

6 RESULT

6.1 Clustering Experiments

We use some pieces of music to do the clustering experiments. The selected pieces are:

1. Menuet 1& Menuet 2, ‘Cello Suite no.1, J. S. Bach
2. Blues in E (Piano, bass, guitar, drums), Anon.
3. Canon in D, J. Pachelbel
4. Allegro, Eine Kleine Nachtmusik, W. A. Mozart
5. Andante, Piano Sonata in G, K.283, W. A. Mozart
6. Allegro, Symphony no. 6 ‘Pastorale’, L. v. Beethoven.

They are the same as those in the previous project [4], such that we can compare the performance between our project and the previous project.

Piece 1
Menuet 1& Menuet 2, ‘Cello Suite no.1, J. S. Bach

Repeating sections are:

Bars 9 - 16 match 1 - 8: length of 8
Bars 97 - 104 match 1 - 8: length of 8
Bars 97 - 120 match 9 - 32: length of 24
Bars 33 - 48 match 17 - 32: length of 16
Bars 105 - 120 match 33 - 48: length of 16
Bars 53 - 54 match 49 - 50: length of 2
Bars 57 - 64 match 49 - 56: length of 8
Bars 61 - 62 match 49 - 50: length of 2
Bars 57 - 58 match 53 - 54: length of 2
Bars 61 - 62 match 57 - 58: length of 2
Bars 81 - 96 match 65 - 80: length of 16
The form of this piece is:

A : bars 1 to 8 (8 bars long)
A : bars 9 to 16 (8 bars long)
B : bars 17 to 32 (16 bars long)
B : bars 33 to 48 (16 bars long)
C : bars 49 to 56 (8 bars long)
C : bars 57 to 64 (8 bars long)
D : bars 65 to 80 (16 bars long)
D : bars 81 to 96 (16 bars long)
A : bars 97 to 104 (8 bars long)
B : bars 105 to 120 (16 bars long)

Clusters:

A:
bars 1 to 8 (8 bars long)
bars 9 to 16 (8 bars long)
bars 97 to 104 (8 bars long)
B:
bars 17 to 32 (16 bars long)
bars 33 to 48 (16 bars long)
bars 105 to 120 (16 bars long)
C:
bars 49 to 56 (8 bars long)
bars 57 to 64 (8 bars long)
D:
bars 65 to 80 (16 bars long)
bars 81 to 96 (16 bars long)

The form of this piece is:

AABBCCDDAB

Style Prediction:

Sonata

The report of previous project [4] is on the left and that of our project is on the right. We can see our method and the previous method performs the same, but our method is even faster. Additionally, our method gives the style prediction which based on the musical form. This is a well-structured music, so most algorithms can perform well and get the right musical form.

Piece 6
Allegro, Symphony no. 6 ‘Pastorale’, L. v. Beethoven

The form of this piece is:

A : bar 1 (1 bar)
B : bars 2 to 16 (15 bars long)
C : bar 17 (1 bar)
B : bars 18 to 32 (15 bars long)
D : bars 33 to 53 (21 bars long)
E : bars 54 to 56 (3 bars long)
? : bars 57 to 59 (3 bars long)
F : bars 60 to 72 (13 bars long)
? : bars 73 to 75 (3 bars long)
G : bars 76 to 80 (5 bars long)
? : bars 81 to 83 (3 bars long)
H : bar 84 (1 bar)
I : bars 85 to 88 (4 bars long)
? : bars 89 to 91 (3 bars long)
J : bars 92 to 98 (7 bars long)
? : bars 99 to 102 (4 bars long)
K : bars 103 to 106 (4 bars long)
? : bars 107 to 109 (3 bars long)
L : bars 110 to 114 (5 bars long)
? : bars 115 to 117 (3 bars long)
M : bars 118 to 138 (21 bars long)
? : bars 139 to 143 (5 bars long)
N : bars 144 to 191 (48 bars long)
O : bar 192 (1 bar)
? : bars 193 to 202 (10 bars long)
P : bars 203 to 207 (5 bars long)
Q : bars 208 to 221 (14 bars long)
R : bar 222 (1 bar)
? : bars 223 to 252 (30 bars long)
S : bars 253 to 254 (2 bars long)

Clusters:

A:
bars 1 to 16 (16 bars long)
bars 17 to 32 (16 bars long)
bars 57 to 75 (19 bars long)
bars 118 to 143 (26 bars long)
bars 223 to 254 (32 bars long)

B:
bars 33 to 56 (24 bars long)
bars 33 to 48 (16 bars long)
C:
bars 76 to 96 (21 bars long)
bars 97 to 117 (21 bars long)
D:
bars 144 to 191 (48 bars long)
bars 192 to 222 (31 bars long)

The form of this piece is:

AABACCADDA

Style Prediction:

Rondo

Now let us skip over piece 2 to piece 5, and watch piece 6 directly. This is indeed a hard piece for the previous project. Tom Lyner claimed that there are problems when he tried to identify the musical form. And he finally only found one pattern “B”. And look at our result, our method cluster the segments into four groups and successfully predict the musical style.

Why our method performs much better than the previous method? The only reason is SVD. Since this music is written by Beethoven, it includes less obvious structure than the music written by Bach like piece 1. And this song is longer and more complex, include many musical composition tricks. So SVD can reduce the dimension of the data and find the latent relationship and clusters.

We compare our method and the method in the previous project [4], and we use the manual musical form and segmentation as reference. The statistical results for the six pieces are in the form below:

Songs	Old Method		Our Method		
	Segmentation precision	Musical form precision	Segmentation precision	Musical form precision	Style Prediction
Piece 1	94.1%	100%	97.2%	100%	right
Piece 2	81.7%	66.6%	77.2%	80%	wrong
Piece 3	61.4%	46.5%	83.6%	100%	right
Piece 4	92.3%	100%	89.7%	100%	right
Piece 5	45.6%	66.6%	70.2%	90%	wrong
Piece 6	21.9%	16.5%	93.9%	100%	right

Table 2 Comparison between our method and the old method

Here the precision means the average precision. For example, if the right (manual) musical form is ABCAB, and the result of one method is ABCAA, the precision will be 80%.

From the form we can see that our method is much better than the previous one. And the improvement is over 100% of the original method.

We also compare our method to other methods, but most experiments are designed for pop music, and our method is designed for classical music. So in the experiments of pop music our method performs worse and in the experiments of classical music our method performs better than most methods.

6.2 Distance function Selection

In earlier sections we mentioned the distance functions. Here we do some experiments to test their performance. By replacing the cosine distance function with some other functions, we modify the similarity matrix constructed in Chapter 5. And the evaluation method is just as we used in the clustering experiments, to measure the musical form precision.

We test five functions:
Cosine distance
Euclidean distance
Hamming distance
Chebyshev distance
Minkowski distance

We use cosine distance as the base line, and the details about how there functions defined can be found in [34]. And we use the open source implementation of these distance functions to do the experiments. The sample musical pieces are also the six pieces.

Distance Function	Piece 1	Piece 2	Piece 3	Piece 4	Piece 5	Piece 6
Cosine distance	100%	80%	100%	100%	90%	100%
Euclidean distance	90%	90%	80%	80%	100%	90%
Hamming distance	60%	50%	60%	80%	100%	80%
Chebyshev distance	90%	100%	100%	90%	60%	100%
Minkowski distance	100%	100%	100%	80%	100%	70%

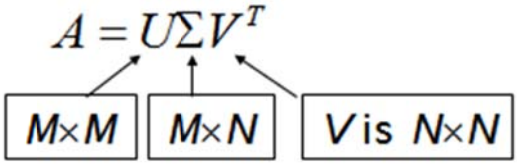
Table 3 Comparison of different distance measures

From form 2 we can see that cosine distance is averagely good enough for most cases, and it is the most stable method. Hamming distance is not appropriate for this method. Chebyshev distance and Minkowski distance is good for most samples, but they are not stable enough. Euclidean distance is stable, but only one result is 100%, seems not good enough. So using cosine distance is really a good choice in our project.

7 DISCUSSION

7.1 Meaning of SVD

In Chapter 3 we introduce eigenvalues. But SVD is based on singular values. The most obvious difference between eigenvalue and singular value is the accepted matrix. Eigenvalue problem can only treat the square matrix, and sometimes requires the symmetry. But for most general problems, they cannot be converted to square matrixes. SVD is designed for these cases; it can deal with matrix in any size. It can be shortly written as [35]:



These equations for $M \times N$ matrix A you may have seen before. And you will see it many times in the future since it is very important for SVD. And we can also represent SVD by illustrations:

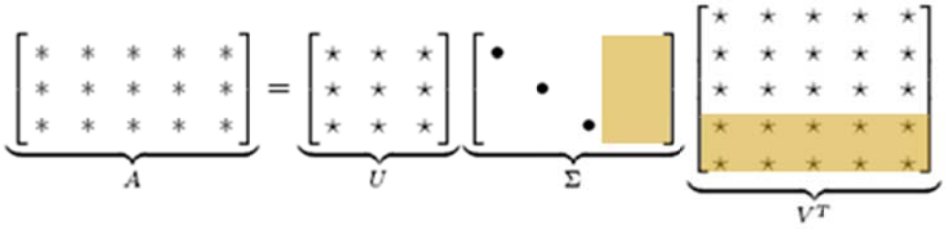


Figure 32 Illustration of SVD

Note that the dark area means the sparseness of the matrix. So we can do the low-rank approximation by the sparseness as:

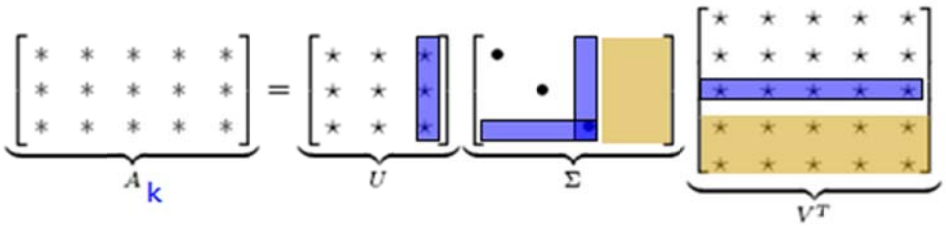


Figure 33 Illustration of Reduced SVD

The interesting approximation means we can represent the high dimensional data in a lower space. For example, in the musical segments clustering section, we can represent the segments in the lower (select the first two dimensions) space as:

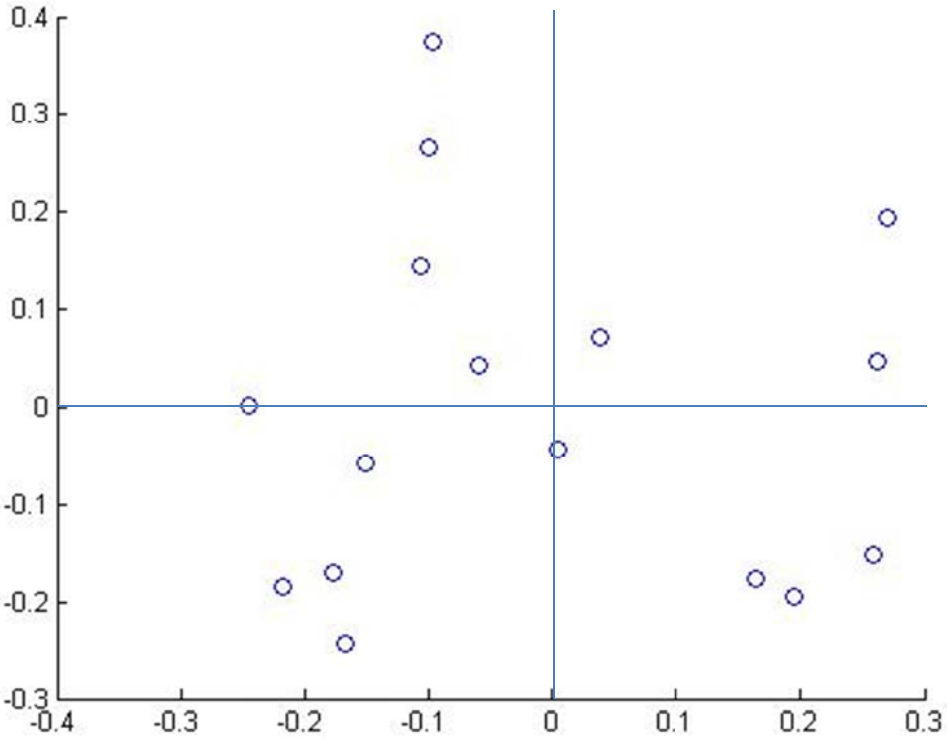


Figure 34 Segments in the low space

Now we have the geometric meaning of SVD: it can project the high dimensional data to lower dimensional space. And some data can be easily separated in the dimensional space.

7.2 SVD Analysis

7.2.1 Introduction

Definition 1. Assume $A \in R^{m \times n}$, the Nonnegative square root of the eigenvalue of $A^T A$ is the singular value of A, all the singular values of A is written as $\sigma(A)$. [36]

Theorem 1. Let $A \in R^{m \times n}$, there must exist orthogonal matrixes [36]:

$$U = [u_1, ..., u_m] \in R^{m \times m} \quad \text{And} \quad V = [v_1, ..., v_n] \in R^{n \times n}$$

$$\text{Such that: } U^T AV = \begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix} \begin{matrix} r \\ m-r \end{matrix} \quad (1)$$
$$\qquad\qquad\qquad\qquad\quad r \quad n-r$$

$$\text{Here } \Sigma_r = diag(\sigma_1, ..., \sigma_r), \sigma_1 \geq ... \geq \sigma_r > 0$$

The equation ϕ is called the Singular value decomposition (SVD) of matrix A, σ_i is the singular value of A. u_i and v_i are the i-th left singular vector and the i-th right singular vector. From this, we can get some useful reasoning.

Reasoning 1. Let $A \in C_r^{m \times n}$, the number of the non-zero singular values of A is $r = rank(A)$,

$v_{r+1}, ..., v_n$ is a standard orthogonal basis of the null space(or kernel) of A

$u_1, ..., v_r$ is a standard orthogonal basis of the domain space of A

$A = \sum_{i=1}^{r} \sigma_i u_i v_i^H$ is the full rank SVD of A.

And we define the following notations:

σ_i is the i-th singular value of A
σ_{max} is the maximum singular value of A
σ_{min} is the minimum singular value of A

Now let us consider the geometric meaning of the singular value. Assume m=n, we have [36]:

$$E_n = \left\{ y \in C^n : y = Ax, x \in C^n, \|x\|_2 = 1 \right\}$$

It is a hyper-ellipsoid. There are n semi axis which can be seen as the n singular values $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n \geq 0$ of A.

Let us assume $m \geq n$ (if $m < n$, we can use the transposition of A, and do the SVD as well, then calculate the transposition of the result of SVD). If we represent U as:

$$U = (U_1, U_2),$$

Then we can get a new SVD equation:

$$A = U_1 \Sigma V^T$$

Here U_1 has n m-Dimension orthogonal vectors, $\Sigma = diag(\sigma_1, \sigma_2, ..., \sigma_n)$, $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n \geq 0$ is the singular values of A [36]

7.2.2 The application of SVD

1) Determine the rank of matrix

When the rank of A is r, we have:

$$\sigma_1 \geq ... \geq \sigma_r > \sigma_{r+1} = ... = \sigma_n = 0 \quad [37]$$

Then if we find $\sigma_r \neq 0$ and $\sigma_{r+1} = ... = \sigma_n = 0$, we can determine the rank of matrix.

Actually, the singular values are almost all non-zero. We should also determine the approximation rules.

2) Determine the projection operator [38]

If the rank of A is r, we can represent U_1 as

$$U_1 = \left(U_1^{(1)}, U_2^{(1)}\right)$$

Here $U_1^{(1)}$ is a $m \times r$ matrix, and the column vectors of it form the orthogonal vector base of the column space of A

So $P_A = U_1^{(1)} U_1^{(1)T}$ is the orthogonal projection operator on the column space of A and $P_A^{\perp} = (U_2^{(1)}, U_2)(U_2^{(1)}, U_2)^T$ is projection on the orthogonal complement space of the column space of A

3) Least squares (LS) problem [39]

The LS problem is: Let $A \in R^{m \times n}\ (m > n), b \in R^m$, find $x \in R^n$ that:

$$\left\| Ax - b \right\|_2 = \min \{ \left\| Av - b \right\|_2 : v \in R^n \}.$$

If we have had the SVD $U \Sigma V^T$ of A, then we can get:

$$\begin{aligned} Ax - b &= U \Sigma V^T x - b \\ &= U(\Sigma V^T x) - U(U^T b) .....(2.2) \\ &= U(\Sigma y - c) \end{aligned}$$

Here $y = V^T x$, $c = U^T b$

Since U is an orthogonal matrix, we have $\|Ax-b\|_2 = \|U(\Sigma y - c)\|_2 = \|\Sigma y - c\|_2$, then the problem become finding a value of y which can minimize $\|\Sigma y - c\|_2$

If the rank of A is r, we have:

$$\Sigma y = \begin{bmatrix} \sigma_1 y_1 \\ \vdots \\ \sigma_r y_r \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \qquad \Sigma y - c = \begin{bmatrix} \sigma_1 y_1 - c_1 \\ \vdots \\ \sigma_r y_r - c_r \\ -c_{r+1} \\ -c_{r+2} \\ \vdots \\ -c_m \end{bmatrix}$$

So $y_i = c_i / \sigma_i, (i = 1, 2, ..., r)$ makes $\Sigma y - c$ minimum as $\left[\sum_{i=r+1}^{m} c_i^2\right]^{1/2}$. We define $\sum^+$ as the transposition of $\sum$ and inverse all the non-zero values. Then the first r elements of $y = \Sigma^+ c$ will be $c_i / \sigma_i, (i = 1, 2, ..., r)$, and others are 0, and from $y = V^T x$, $c = U^T b$, we have:

$$x = V \sum^+ U^T b$$

This is the minimum norm solution of LS problem.

In [40] there is a more general solution:

$$x = V \sum^+ U^T b + V_2 w$$

Vector w can be any $n-r$ dimension vector.

4) pseudo-inverse [41]

Let $A^+ = V \sum^+ U^T$, by using LS, the solution is $x = A^+ b$. It is very similar to the solution $x = A^{-1} b$ of the linear function set $Ax = b$. So the singular value can be used to represent the pseudo-inverse of A as $A^+ = V \sum^+ U^T$ [41]

7.2.3 SVD Algorithm

First let us see a survey of the history of SVD. The earliest work was done by Beltrami in 1873. At that time lots of theory work had been done. That work is stated in Stewart's paper [35]. But the numerical computation of SVD is never been researched until 1965, Golub and Kahan got some dramatic breakthroughs [35]. And designed a stable algorithm [42] (QR-iteration algorithm) in 1969. The main idea of the algorithm is using orthogonal transformation to transform the matrix to double diagonal matrix. And use double diagonal matrix iteration as the QR decomposition.

In an unpublished report, Kahan proved that the singular value of the double diagonal matrix can be precisely calculated. Moreover, Demmel and Kahan gave a zero-shift QR algorithm in 1990. It has a significant relative precision for calculating the singular value of the double diagonal matrix [43]. Then the corresponding singular vector also can be precisely calculated [44]. Fernando and Parlett used qd algorithm on the singular value problem in 1994, and constructed a new algorithm which was faster and more precise for calculating the singular value than the zero-shift QR algorithm [45, 46].

In [47], Demmel and Veselic introduced how to apply Jacobi method to get more precise singular value and singular vector, but it is much slower than DK algorithm. And [48] did some improvement on Jacobi method and made it as fast as DK alforithm.

There are also divide and conquer type method in [44, 45], it is can significant decrease the total computation cost.

Bisection algorithm [41, 42] is also an effective one.

Now we will state the detail of the classic QR iterationg algorithm [41, 42, and 43].

Let $A \in R^{m \times n} (m \geq n)$. SVD can be calculated from the Schur decomposition of the real symmetric matrix $C = A^T A$ [41]. So we can first use symmetrical QR method to calculate the Schur decomposition of C. But getting C will cost much time, and $C = A^T A$ will cause much error. Golub and Kahan stated another stable method in 1965, it can impliedly apply symmetrical QR method on $A^T A$, but not calculate the form of $A^T A$.

The first step is to get orthogonal matrixes U_1 and V_1, such that:

$$U_1^T A V_1 = \begin{bmatrix} B \\ 0 \end{bmatrix}_{m-n}^{n}$$

And

$$B = \begin{bmatrix} \delta_1 & \gamma_2 & 0 & 0 & 0 \\ 0 & \delta_2 & \gamma_3 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & 0 \\ 0 & 0 & 0 & \ddots & \gamma_n \\ 0 & 0 & 0 & 0 & \delta_n \end{bmatrix}$$

We can use Householder transformation to get it. Represent A as:

$$A = \underset{1}{\begin{bmatrix} v_1 \end{bmatrix}} \quad \underset{n-1}{\begin{bmatrix} A_1 \end{bmatrix}}$$

Calculate m-rank Householder transformation P_1:

$$P_1 v_1 = \delta_1 e_1 (\delta_1 \in R, e_1 \in R^m)$$

And represent as:

$$P_1 A_1 = \begin{bmatrix} u_1^T \\ \tilde{A}_1 \end{bmatrix} \begin{matrix} 1 \\ m-1 \end{matrix}$$

Then calculate (m-1)-rank Householder transformation $\tilde{H}_1$:

$$\tilde{H}_1 u_1 = \gamma_2 e_1 (\gamma_2 \in R, e_1 \in R^{n-1})$$

And represent as:

$$\tilde{A}_1 \tilde{H}_1 = \underset{1}{\begin{bmatrix} v_2 \end{bmatrix}} \quad \underset{n-2}{\begin{bmatrix} A_2 \end{bmatrix}}$$

Then for $k = 2, 3, ..., n-2$:

a. Calculate (m-k+1)-rank Householder transformation $\tilde{P}_k$:

$$\tilde{P}_k v_k = \delta_k e_1 (\delta_k \in R, e_1 \in R^{m-k+1})$$

And represent as:

$$\tilde{P}_k A_k = \begin{bmatrix} u_k^T \\ \tilde{A}_k \end{bmatrix} \begin{matrix} 1 \\ m-1 \end{matrix}$$

b. Calculate (n-k)-rank Householder transformation $\tilde{H}_k$;

$$\tilde{H}_k u_k = \gamma_{k+1} e_1 (\gamma_{k+1} \in R, e_1 \in R^{n-k})$$

And represent as:

$$\tilde{A}_k \tilde{H}_k = \begin{bmatrix} v_{k+1} & A_{k+1} \\ 1 & n-k-1 \end{bmatrix}$$

After we get $k = n-2$, calculate (m-n+2)-rank Householder transformation $\tilde{P}_{n-1}$:

$$\tilde{P}_{n-1} v_{n-1} = \delta_{n-1} e_1 (\delta_{n-1} \in R, e_1 \in R^{m-n+2})$$

And represent as:

$$\tilde{P}_{n-1} A_{n-1} = \begin{bmatrix} \gamma_n \\ v_n \end{bmatrix} \begin{matrix} 1 \\ m-n+1 \end{matrix}$$

Then calculate (m-n+1)-rank Householder transformation $\tilde{P}_n$:

$$\tilde{P}_n v_n = \delta_n e_1 (\delta_n \in R, e_1 \in R^{m-n+1})$$

Now let:

$$P_k = diag\left(I_{k-1}, \tilde{P}_k\right), k = 2, \ldots n$$

$$H_k = diag\left(I_k, \tilde{H}_k\right), k = 1, 2, \ldots n-2$$

$$U_1 = P_1 P_2 \ldots P_n \text{ , } \quad V_1 = H_1 H_2 \ldots H_{n-2}$$

$$B = \begin{bmatrix} \delta_1 & \gamma_2 & 0 & 0 & 0 \\ 0 & \delta_2 & \gamma_3 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & 0 \\ 0 & 0 & 0 & \ddots & \gamma_n \\ 0 & 0 & 0 & 0 & \delta_n \end{bmatrix}$$

Then we have:

$$U_1^T A V_1 = \begin{bmatrix} B \\ 0 \end{bmatrix} \begin{matrix} n \\ m-n \end{matrix}$$

Till now, the first step is finished.

Then we start the symmetrical QR iteration with Wilkinson shift for tridiagonal matrix $T = B^T B$, note that we need not calculate the whole T matrix. First get the principal sub matrix in the 2×2 area of the lower right corner of $T = B^T B$

$$\begin{bmatrix} \delta_{n-1}^2 + \gamma_{n-1}^2 & \delta_{n-1}\gamma_n \\ \delta_{n-1}\gamma_n & \delta_n^2 + \gamma_n^2 \end{bmatrix}$$

Use the closest eigenvalue of $\delta_n^2 + \gamma_n^2$ as the shift μ, and then determine the Givens transformation $G_1 = G(1, 2, \theta)$. Here $c = \cos(\theta), s = \sin(\theta)$ can satisfy:

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix}^T \begin{bmatrix} \delta_1^2 - \mu \\ \delta_1 \gamma_2 \end{bmatrix} = \begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} \delta_1^2 - \mu \\ \delta_1 \gamma_2 \end{bmatrix} = \begin{bmatrix} \sigma \\ 0 \end{bmatrix}$$

Here $\delta_1^2 - \mu$ and $\delta_1 \gamma_2$ are the only two non-zero elements in the positions (1, 1) and (1, 2) of the first row of $T - \mu I$

Now we determine orthogonal matrix Q such that $Q^T (G_1^T T G_1) Q$ is a Symmetric tridiagonal matrix, which means bidiagonalize BG_1. We can premultiplication and postmultiplication some Givens transformations, as form: $J_{n-1} J_{n-2} ... J_1 (BG_1) G_2 ... G_{n-1}$, to finish the bidiagonalization.

Algorithm 2

Input: x, y

Output: $c = \cos(\theta), s = \sin(\theta),$ such that $\begin{bmatrix} c & s \\ -s & c \end{bmatrix}^T \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}$

$function: [c \quad s \quad r] = Givens(x, y)$

if $y = 0$

$c = 1, s = 0;$

else

if $(|y| > |x|)$

$\tau = -x / y; s = \sqrt{1 + \tau^2}, r = -y * s; s = 1 / s; c = s\tau;$

else

$\tau = -y / x; c = \sqrt{1 + \tau^2}, r = x * c; c = 1 / c; s = c\tau;$

end

end

Algorithm 3

Input: c, s, v_1, v_2
Output: none

Update(c, s, v_1, v_2)

for i = 1 to n

$t = v_1(i)$

$v_1(i) = c * t - s * v_2(i)$

$v_2(i) = s * t + c * v_2(i)$

endfor

Every time it updates, multiple the singular matrixes with a Givens matrix $\begin{bmatrix} c & s \\ -s & c \end{bmatrix}$

Algorithm 4 One QR iteration

Input: diagonal elements $\delta_{\underline{i}} \ldots \delta_{\bar{i}}$ *and sub diagonal elements* $\gamma_{\underline{i}+1} \ldots \gamma_{\bar{i}}$

Output: none

$d = \left[\left(\delta_{\bar{i}-1}^2 + \gamma_{\bar{i}-1}^2 \right) - \left(\delta_{\bar{i}}^2 + \gamma_{\bar{i}}^2 \right) \right] / 2$,

$\mu = (\delta_{\bar{i}}^2 + \gamma_{\bar{i}}^2) + d - sign(d)\sqrt{d^2 + \delta_{\bar{i}-1}^2 \gamma_{\bar{i}}^2}$,

$x = \delta_{\underline{i}}^2 - \mu, y = \delta_{\underline{i}} \gamma_{\underline{i}+1}, k = \underline{i}$,

$Q = I, P = I$;

*

$[c, s, r] = Givens(x, y)$;

Update(c, s, q_k, q_{k+1})

If $k > \underline{i}$

$\gamma_k = r$

Else

$[c, s, r] = Givens(x, y);$

$\delta_k = r$

$Update(c, s, p_k, p_{k+1})$

End

If $k < \bar{i} - 1$

$$\begin{bmatrix} x & y \\ \delta_{k+1} & \gamma_{k+2} \end{bmatrix} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}^T \begin{bmatrix} \gamma_{k+1} & 0 \\ \delta_{k+1} & \gamma_{k+2} \end{bmatrix}$$

$k = k + 1$

Goto *

Else $\begin{bmatrix} \gamma_{\bar{i}} \\ \delta_{\bar{i}} \end{bmatrix} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}^T \begin{bmatrix} \gamma_{\bar{i}} \\ \delta_{\bar{i}} \end{bmatrix}$

End

Algorithm 5 Classic SVD iteration

Input: $A \in R^{m \times n} (m \geq n)$ *and error threshold* ε

Output: SVD result $A = U \Sigma V^T$

1. *Calculate Householder transformation* $P_1, \cdots, P_n$, $H_1, \cdots, H_{n-2}$, *such that:*

$$(P_1 \cdots P_n)^T A (H_1 \cdots H_{n-2}) = \begin{bmatrix} B \\ 0 \end{bmatrix} \begin{matrix} n \\ m-n \end{matrix},$$

$$And \quad B = \begin{bmatrix} \delta_1 & \gamma_2 & & 0 \\ & \ddots & \ddots & \\ & & \ddots & \gamma_n \\ 0 & & & \delta_n \end{bmatrix}$$

$U := P_1 P_2 \cdots P_n,$

$V := H_1 H_2 \cdots H_{n-2}.$

2. *Set all the j to 0 which satisfy:*

$$\left|\gamma_j\right| \le \varepsilon\left(\left|\delta_j\right| + \left|\delta_{j-1}\right|\right)$$

If $\gamma_j = 0,\ j = 2,\cdots,n$

Exit algorithm;

Else $\gamma_1 := 0$ *;*

End

Determine positive integers $p < q$ *which satisfy:*

$$\gamma_p = \gamma_{q+1} = \cdots = \gamma_n = 0, \quad \gamma_j \neq 0, \quad p < j \le q$$

If there exist a i that $p \le i \le q-1$ *satisfy* $\left|\delta_i\right| \le \varepsilon \left\|B\right\|_\infty$

$$\delta_i := 0, x := \gamma_{i+1}, y := \delta_{i+1}, \gamma_{i+1} := 0, l := 1$$

Else goto step 4
End
*
$[c, s, r] = Givens(x, y);$
$\delta_{i+l} := \sigma,\ \ U := UG(i, i+l, \theta)^T;$

If $l < q - i$

$$x := s\gamma_{i+l+1},\ \ \gamma_{i+l+1} := c\gamma_{i+l+1}$$

$$y := \delta_{i+l+1},\quad l := l+1$$

*Goto **

Else goto step 2
End

3. *SVD iteration: apply Algorithm 3 on:*

$$B_1 = \begin{bmatrix} \delta_p & \gamma_{p+1} & & & 0 \\ & \delta_{p+1} & \gamma_{p+2} & & \\ & & \ddots & \ddots & \\ & & & \ddots & \gamma_q \\ 0 & & & & \delta_q \end{bmatrix}$$

Get:

$$B_1 := P^T B_1 Q,\ \ U := U diag(I_p, P, I_{n-p-q}),\ \ V := V diag(I_p, Q, I_{n-p-q})$$

Goto step 2

Cost analysis: This iteration algorithm is Cubical fast convergence [42], so the time cost of it is just $O(n^3)$ [43]. In [49] the time cost is about $20n^3$

7.2.4 Method comparison

There are many different algorithms, but we decide to firstly finish the classical one, and then finish more algorithms to research their effect. From the survey of the SVD algorithms (some results are listed in the forms below), Classical QR algorithm performs stable but low accuracy and speed; zero-shift QR mix algorithm is a good tradeoff between the accuracy and speed. qd algorithm seems the best, but there is no one implementation of it. Jacobi algorithm perform the best accuracy, but not fast enough. Divide and conquer method perform unstable accuracy but very fast, and some detail of it is lacked.

Table 4 includes some experiment results of DK and oqd. The values in it mean the increment of each cycle.

	DK	oqd
cabs	2	1*2
Divisions	2	1*2
Multiplication	6	2*2
Conditionals	1	0
Assignments	7	3*2
Auxiliary variables	6	1

Table 4 Comparison between Dk algorithm and oqd algorithm

Form 4 includes the experiment results of three kinds of qd algorithms. The meaning of values in it is the same as which in form 3.

	oqd	dqd	qd
cabs	1	0	0
Divisions	2	1	1
Multiplication	4	2	1
Additions	1	1	1
Subtractions	0	0	1
Assignments	3	3	2
Auxiliary variables	1	1	0

Table 5 Comparison of three kinds of qd algorithms

8 CONCLUSION

8.1 Evaluation

In this report we gave a detailed description and discussion of our project. We think we achieve our aims of this project, and find some additional interesting topics. We introduced some music background and the modeling of the music structure. And we stated some relevant work about music information discovery and matrix decomposition. We researched and described many methods of the matrix decomposition. And we focus on one of the methods, SVD; introduce the algorithms of SVD and why it is useful for music structure discovery. We design some experiments and analyze the result of it. Then we discuss the algorithm performance and compare different algorithms.

For the theoretical aims, we completed the comparison of different methods of matrix decomposition and analyzed their advantages and disadvantages. We also introduce an interesting topic, spectral algorithms. We deduced the equations and overview the applications. Combining dimensionality reduction and spectral clustering, we lead to SVD. We studied the mathematical meaning of SVD. We also introduced many SVD algorithms, and gave the basic frames. We also successfully used SVD for musical similarity decomposition, and compare our method with the method in the previous project [4]. The result shows our method is significantly better than most known methods for classical musical form determining. We keep the bar modeling method and similarity matrix building method from the previous project [4], but we compared different distance measures and introduce more kinds of similarity matrixes and their applications.

For the software implementation aims, since we mainly focus on the analysis of method, theory and result, we just briefly describe the software details. And the basic frame is also from the previous project [4], we just update it and add SVD component into it. And we also implement some interesting additional functions include musical style prediction. And we redesigned the GUI to show more information about SVD decomposition.

8.2 Further Work

HMM

In the research process, we find HMM is a powerful tool for musical structure analysis. So we could do some more research about HMM

Spectral Algorithm and manifold

In the background research section, we find two subjects: spectral algorithm and manifold. They are both the hot points in machine learning area. More and more projects about them are created now. The normalization method and regularization rule are desired to research.

Other Similarity Matrix

In our project we just introduce some kinds of similarity matrix. There are more kinds of matrix desired to study. We should systemically collect and research their applications and analyze their performance. Some result may be very interesting.

REFERENCE

[1] Content-based music structure analysis NC MADDAGE - 2006 - scholarbank.nus.edu.sg

[2] Rudiments and Theory of Music. The associated board of the royal schools of music, 14 Bedford Square, London, WC1B 3JG, 1949.

[3] Krumhansl, C. L. (1979). The Psychological Representation of Musical Pitch in a Tonal Context. In Journal of Cognitive Psychology, 1979, Vol.11, No. 3, pp. 346-374.

[4] Tom Lyner, Finding harmonic structure in music files (2009), UoB

[5] Salton, G. and M. J. McGill (1983). Introduction to modern information retrieval. McGraw-Hill. ISBN 0070544840

[6] W. Huffman, V. Pless, Fundamentals of error-correcting codes, Cambridge University Press, ISBN 9780521782807, 2003

[7] Samet, H. (2006) Foundations of Multidimensional and Metric Data Structures. Morgan Kaufmann. ISBN 0123694469

[8] Friedberg, Stephen H.; Insel, Arnold J.; Spence, Lawrence E. (1989), Linear algebra (2nd ed.), Englewood Cliffs, NJ 07632: Prentice Hall, ISBN 0-13-537102-3

[9] "Vorbis I Specification". Xiph.org. 2007-03-09. Retrieved 2007-03-09.

[10] Ronald Fisher (1936)The Use of Multiple Measurements in Taxonomic Problems In: Annals of Eugenics, 7, p. 179—188

[11] McLachlan (2004)Discriminant Analysis and Statistical Pattern Recognition In: Wiley Interscience

[12] Pattern recognition and machine learning CM Bishop - 2006 - library.wisc.edu

[13] Nicholas J. Higham, Recent Developments in Dense Numerical Linear Algebra (2000), http://citeseer.ist.psu.edu/higham00recent.html

[14] Bae, D-S; Haug EJ (1988). "A Recursive Formulation for Constrained Mechanical System Dynamics: Part I. Open Loop Systems". Mechanics of Structures and Machines 15: 359–382

[15] Shaw PJA (2003) Multivariate statistics for the Environmental Sciences, Hodder-Arnold.

[16] Ravi Kannan and Adrian Vetta, On clusterings: good, bad and spectral. Jour nal of the ACM (JACM) 51(3), 497--515, 2004.

[17] A.Y. Ng, M.I. Jordan, and Y. Weiss. On spectral clustering: Analysis and a n algorithm. Proc. Neural Info. Processing Systems (NIPS 2001), 2001.

[18] H. Zha, C. Ding, M. Gu, X. He, and H.D. Simon. Spectral relaxation for K-m eans clustering. Advances in Neural Information Processing Systems 14 (NIPS 20 01). pp. 1057-1064, Vancouver, Canada. Dec. 2001.

[19] Deng Cai, Xiaofei He, Jiawei Han, "Document Clustering Using Locality Preserving Indexing," IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 12, pp. 1624-1637, Dec. 2005, doi:10.1109/TKDE.2005.198

[20] J. Shi and J. Malik. Normalized cuts and image segmentation. IEEE. Trans. on Pattern Analysis and Machine Intelligence, 22:888--905, 2000.

[21] M. Belkin and P. Niyogi. Laplacian Eigenmaps and Spectral Techniques for E mbedding and Clustering, Advances in Neural Information Processing Systems 14 (NIPS 2001), pp: 585-591, MIT Press, Cambridge, 2002.

[22] John A. Lee, Michel Verleysen, Nonlinear Dimensionality Reduction, Springer, 2007

[23] Laplacian Eigenmaps for Dimensionality Reduction and Data Representation Mikhail Belkin, Partha Niyogi, Neural Computation June 2003, Vol. 15, No. 6, Pages 1373-1396: 1373-1396.

[24] Spectral Regression: A Unified Approach for Sparse Subspace Learning, Deng Cai; Xiaofei He; Jiawei Han; Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on

[25] F.R.K. Chung. Spectral Graph Theory. Amer. Math. Society Press, 1997.

[26] M. Meila and J. Shi. A random walks view of spectral segmentation. Int'l W orkshop on AI & Stat (AI-STAT 2001)

[27] Jeff Cheeger, A lower bound for the smallest eigenvalue of the Laplacian. Problems in analysis (Papers dedicated to Salomon Bochner, 1969), pp. 195–199. Princeton Univ. Press, Princeton, N. J., 1970 MR0402831

[28] A Divide-and-Merge Methodology for Clustering (D. Cheng, R. Kannan and G. Wang) Proc. of the ACM Symposium on Principles of Database Systems, 2005.

[29] Jonathan T. Foote, "Content-based retrieval of music and audio", Proc. SPIE 3229, 138 (1997)

[30] Jonathan T. Foote, An overview of audio information retrieval, MULTIMEDIA SYSTEMS Volume 7, Number 1, 2-10

[31] J. Foote and M. Cooper. Media segmentation using self-similarity decomposition. In Proc. SPIE Storage and Retrieval for Media Databases, volume 5021, pages 167-175, Santa Clara, California, USA, 2003.

[32] P. A. Businger and G. H. Golub, Algorithm 358: Singular value decomposition of a complex matrix, Assoc. Comp. Math, 12:564-565, 1969.

[33] S. Dubnov and T. Appel, "Audio segmentation by singular value clustering," in Proc. of Int.Conf. on Computer Music (ICMC), 2004.

[34] Steen, Lynn Arthur; Seebach, J. Arthur Jr. (1995) [1978], Counterexamples in Topology, Dover, ISBN 978-0-486-68735-3, OCLC 32311847

[35] G. W. Stewart, On the early history of the singular value decomposition, SIAM Rev., 35:551-566,1993.

[36] G. H. Golub and W. Kahan, Caculating the singular values and pseudo-inverse of a matrix, SIAM J. Numer. Anal., 2:205-224, 1965

[37] P. A. Businger and G. H. Golub, Algorithm 358: Singular value decomposition of a complex matrix, Assoc. Comp. Math, 12:564-565, 1969.

[38] Nicholas J. Higham, Recent Developments in Dense Numerical Linear Algebra (2000), http://citeseer.ist.psu.edu/higham00recent.html

[39] James W. Demmel and W. Kahan, Accurate singular values of bidiagoanl matrices, SIAM J. Sci. Stat. Comput, 11(5):873-912, 1990.

[40] Percy Deift, James W. Demmel, Luen-Chau Li, and Carlos Tomei, The bidiagonal singular value decomposition and Hamiltonian mechanics, SIAM J. Numer. Anal, 28:1463-1516, 1991.

[41] K. Vince Fernando and Beresford N. Parlett, Accurate singular values and differential qd algorithms, Numer. Math, 67:191-229, 1994.

[42] Beresford N. Parlett, The new qd algorithms, In Acta Numerica, Cambridge University Press, 1995, pages 459-491.

[43] Demmel James and Veselic Kresimir, Jacobi's method is more accurate than QR, SIAM J. Matrix Anal. Appl. 13 (1992), no. 4, 1204--1245.

[44] Z.DRMAC, A posteriori computation of the singular vectors in a preconditioned Jacobi SVD algorithm, IMA J. Numer. Anal., 19(1999), pp191-213.

[45] Ming Gu and Stanley C. Eisenstat, A divide-and-conquer algorithm for the bidiagonal SVD, SIAM Journal on Matrix Analysis and Applications Volume 16, Number 1 pp. 79-92, 1995.

[46] I. Dhillon, J. Demmel, M. Gu, Efficient Computation of the Singular Value Decomposition with Applications to Least Squares Problems, LBL Report #36201, 1994, TECHNICAL REPORT.

[47] J. Barlow and J. Demmel, Computing accurate eigensystems of scaled diagonally dominant matrices, SIAM J. Numer. Anal., 27(1990), pp762-791

[48] K. V. Fernando and B. N. Parlett, Accurately counting singular values of bidiagonal matrices and eigenvalues of skew-symmetric tridiagonal matrices, SIAM J. Matrix Anal. Appl., 20(1998),pp373-399

[49] Jesse L. Barlow, More Accurate Bidiagonal Reduction for Computing the Singular Value Decomposition, SIAM Journal on Matrix Analysis and Applications, Volume 23, Number 3 pp. 761-798, 2002.

[50] E. Anderson, Z. Bai, C. H. Bischof, S. Blackford, J. W. Demmel, J. J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. C. Sorensen, Lapack Users' Guide, 3rd edn. SIAM Press, Philadelphia PA, 1999.

APPENDICES

Appendix A: Laplacian Eigenmap

```
% Laplacian Eigenmap ALGORITHM (using K nearest neighbors)
%
% [Y] = le(X,K,dmax)
%
% X = data as D x N matrix (D = dimensionality, N = #points)
% K = number of neighbors
% dmax = max embedding dimensionality
% Y = embedding as dmax x N matrix

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [Y] = leigs(X,K,d)

[D,N] = size(X);
fprintf(1,'LE running on %d points in %d dimensions\n',N,D);


% STEP1: COMPUTE PAIRWISE DISTANCES & FIND NEIGHBORS
fprintf(1,'-->Finding %d nearest neighbours.\n',K);

X2 = sum(X.^2,1);
distance = repmat(X2,N,1)+repmat(X2',1,N)-2*X'*X;

[sorted,index] = sort(distance);
neighborhood = index(2:(1+K),:);



% STEP2: Construct similarity matrix W
fprintf(1,'-->Constructing similarity matrix.\n');

W = zeros(N, N);
for ii=1:N
    W(ii, neighborhood(:, ii)) = 1;
    W(neighborhood(:, ii), ii) = 1;
end

% STEP 3: COMPUTE EMBEDDING FROM EIGENVECTS OF L
fprintf(1,'-->Computing embedding.\n');

D = diag(sum(W));
```

```
L = D-W;

% CALCULATION OF EMBEDDING
options.disp = 0; options.isreal = 1; options.issym = 1;
[Y,eigenvals] = eigs(L,d+1,0,options);
Y = Y(:,2:d+1)'*sqrt(N); % bottom evect is [1,1,1,1...] with eval 0


fprintf(1,'Done.\n');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```