10701 Final Report

Huan Zhang
School of Music
Carnegie Mellon University
Pittsburgh, PA 15213
huanz@andrew.cmu.edu

Jingyuan Xing
Department of Statistics and Data Science
Carnegie Mellon University
Pittsburgh, PA 15213
jingyuax@andrew.cmu.edu

1 Introduction and Problem Setup

In this project, we choose **task 5: Unsupervised classification of datasets**, aiming to investigate the stylistic trends of Western classical music composers by clustering their compositions and analyze the clustering result. Over the course of classical music history, composers have developed distinctive styles while influencing each other. Low-level quantifiable features like intervals, rhythms, tonal characteristics may be not prominent in human perception of music, but significantly contributes to the style of different composers. Thus we attempt to cluster compositions by different composers by applying clustering algorithms to these features extracted from patterns in music. There are previous works that attempted labeled classification of composers, but few of them have done unsupervised clustering, which we believe will provides more insights on the similarities of stylistic trends between composers.

In this project, we are going to form our own dataset. The music compositions in the dataset are in the form of MIDI encoding, that contains information about musical notes, instrumentation, etc. Our datasets contains 4702 pieces of compositions, spanning over 20 composers in from 13^{th} to 20^{th} century. More statistics of the dataset will be in the following sections.

The problems is as follows: Given the input of MIDI encoding of compositions, output the k clustering result where k is the number of composers. The analysis of clustering results will include a discussion of different music representation models, as well as musical insights.

2 Data

The raw data we used for our project is a symbolic music file format, MIDI (Music Instrument Digital Interface). Each MIDI file can represent a musical composition by storing information like note pitch, length, tempo and so on. The MIDI files are obtained from the following sources: The website http://www.kunstderfuge.com and Classical Archives. All the musical compositions are open domain, but we are not going to release the MIDI data to respect the wish of the archives.

2.1 Data Statistics

The content of our data consists of 20 western classical composers. A wide range of compositions are chosen. Especially, the pieces are not limited to piano solo, but from a variety of instruments. Also, to avoid encountering statistical bias which can created by too big and too small sample size, we controlled the sample sizes for each composers to be more than 50 compositions. Here are the statistics of each composer

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Alkan	205	Bach	996	Beethoven	496	Brahms	119	Buxtehude	72
Byrd	102	Chopin	154	Dandrieu	65	Debussy	179	Dvorak	67
Faure	72	Handel	284	Haydn	452	Mozart	373	Scarlatti	555
Schubert	188	Schumann	68	Scriabin	86	Shostakovich	104	Soler	65

Music is an relative abstract art form, thus a large task we are addressing is how to obtain a good latent representation of musical composition, so that they are distinctive when clustering. To represent the raw music data, we tried two different methods in representing music:

1. **Music Theory features** (776 dim)
2. **Pianoroll Representation** (128000 dim)

1.We used an open source MIDI analytics software, **jSymbolic**, which is able to extract music theory related statistics. From the theoretic contents, we are able to represent each piece of the composition as feature vectors and numerically represent the piece. The information includes: **Pitch Statistics**, **Melodic Intervals**, **Vertical Intervals** , **Rhythm and Tempo**, **Instrumentation**, **Voice Leading**.

2.2 Music Theory Representation

For data prepossessing, we used an open source MIDI analytics software, **jSymbolic**, which is able to extract music theory related statistics.[5] From the theoretic contents, we are able to represent each piece of the composition as feature vectors and numerically represent the piece.

Among the 1494 types of features that provided by jSymbolic, we hand-selected and prioritized the features based on music theory knowledge and designed experiments accordingly. As a brief overview, the features roughly include the following categories: **Pitch Statistics**, **Melodic Intervals**, **Vertical Intervals** , **Rhythm and Tempo**, **Instrumentation**, **Voice Leading**. Under each category there are specific type of information that we are extracting. After feature representation, a piece may look like this:

Name	Pitch Variability	Chromatic Motion	...	Vertical Dissonance
barcarolle 101	12.84	0.0626	...	0.246

At current stage, we are using **51** hand-picked features with a dimension of 776 (some features have higher dimensions). Of course these might not be the most representational feature. In the second part of the project, we are going to investigate the feature representation in detail, using dimension reduction techniques to obtain the best feature space.

2.3 Piano Roll Representation

We also represent MIDI as a piano roll matrix - a 128 * 1000 matrix that pictures the note onset/offset and volume in the axis of time. 128 stands for the fact that there are 128 pitches in MIDI representation, and we evenly sample 1000 points on the axis of time for any length of music. In this way, music of different length is represented as same dimension. Inside the matrix, the volume of the pitch is represented as a number between 0 and 127.

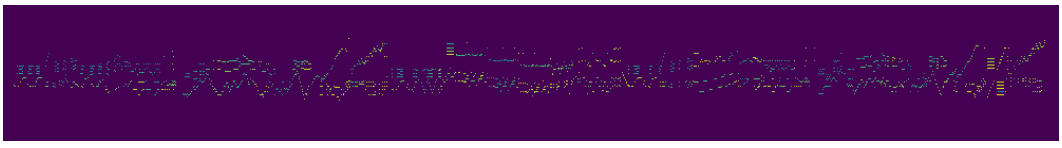

Figure 1: Pianoroll: Mozart Piano Sonata K310, first movement

3 Background

For the midway report, we compared the binary clustering task using KMeans and Spectral Clustering. For discussion on algorithms, the Spectral Clustering with rbf affinity (similarity metrics when constructing the graph) does slightly better than KMeans overall.

Before the midway, we used four metrics for clustering: **adjusted mutual info score**, **adjusted rand score**, **homogeneity score**, **completeness score**. The first one calculates the mutual information between cluster label and true label. Adjusted rand score is base on the Rand Index. Homogeneity score and Completeness score are analogous to the Precision and Recall in the classification task.

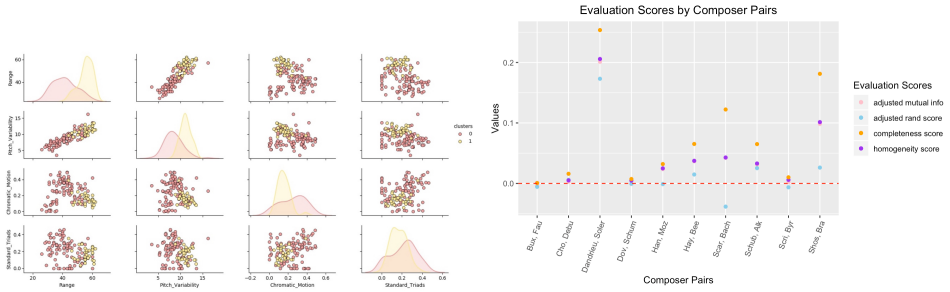

Figure 2: pair-wise scatter plot based on several main attributes in experiment type 1, group 1

Figure 3: Four evaluation scores of 10 composer pairs (KMeans)

For now, we have calculated the metrics for experimental type 1, for all 10 pairs, and plot them in Figure 2. It's true that none of them are close to 1, which indicates a perfect clustering based on the ground truth of composer label.

One of the main reason that accounts for the result is feature representation. It's hard to visualize the 777-dimension feature representation of the composition graphically, but we visualized 4 features pairwise in Figure 1. This is the first binary experiment with composers Faure and Buxtehude using K Means clustering, with color indicating the clustering result. The data points of these features are pretty close. Also, given that there are different genres of compositions among each individual composer's repertoire, that also creates a discrepancy.

4 Related Work

Most of the previous works that related to composition clustering, takes a *similarity metric* approach that first define a distance between pair of music or composers, and then cluster based on the distance. Little machine learning algorithms are involved. Georges[1] defined the metric of centralised cosine measure to cluster the composers. In [2], an information theoretic approach was applied to hierarchical clustering of the music. They did not extract specific features from music but viewed MIDI data as a whole, and utilized the normalized compression distance (NCD) as a similarity metric to cluster composition.
However, there exists multiple works in labeled composition classification that uses machine learning approach. Jain et al. [3] converted the MIDI into image representation and fed it into CNN, so that the composition classification problem is turned into an image classification problem. They also experimented with other algorithms like SVM with extracted features, softmax regression, and they found gradient boosting using decision stumps yields the best result.

5 Methods

Our goal is to obtain the most meaningful and interpretative clustering. There are several variables that we can control when performing clustering: The clustering algorithm being used, features being extracted, and selection of data. For the data, We have designed several types of experiments to investigate aspect of composers' styles, and perform both algorithms on them.

5.1 Clustering from the Raw Representation

For the baseline, we clustered the raw data in two different ways of music representation: **music theory features** (776 dimension) and **piano roll representation** (128000 dimension). We experimented with traditional clustering algorithms like **K Means Clustering**, **Spectral Clustering**, and **Hierarchical Clustering**.

5.2 Autoencoder Latent Representation

As the raw representation of the music data may not be optimal (for example, the pianoroll is a very sparse matrix), we aim to use **autoencoder** and **convolutional autoencoder** as a feature selection/dimension reduction technique, in order to obtain a better representation and use it clustering.

1. For the music theory vector representation, we used an autoencoder to transform the data with a non-linear mapping $f_\theta : X \to Z$, where θ are learnable parameters and Z is the latent feature space. The autoencoder is structured as follows: *Input (776 dims) - Dense (128 dims) - Code (32 dims) - Dense (128 dims) - Output (776 dims)*.
We use rectified linear units (ReLUs) in all layers. Training is performed by minimizing the least-squares loss $||x - y||^2$. The final result is a multilayer autoencoder with a bottleneck coding layer in the middle. We then discard the decoder layers and use the encoder layers as our mapping between the data space and the feature space.
In this way, autoencoder plays a role of dimension reduction, that strives to replicate the original data from a compressed representation: In our case, a 32 dimension vector. After that, we used K Means clustering on this latent 32 dimension representation and compare with the ground truth label for the clustering accuracy.

2. Building up on the previous autoencoder idea, we experimented with more complicated network structure. A convolution component come to mind as the piano roll matrix has a graphic-like interpretation. The structure is as pictured in the following diagram: 4 convolution layers (first 2 5x5 filter, second 2 3x3 filter) each with a maxpooling (2x2 filter) layer. Then the feature map was send into 3 dense layers that reduce to a representation 32-dim vector. As it's a convolutional autoencoder, the rest of layers mirrors the previous ones. The training objective, is again, reproducing the original piano roll representation.

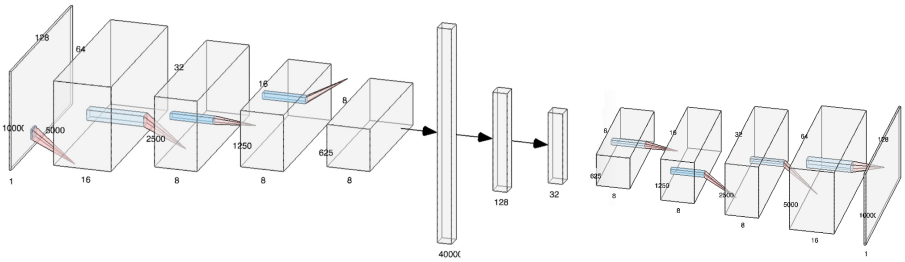

Figure 4: Convolutional Autoencoder Structure

5.3 Deep Embedded Clustering

In the autoencoder latent representation, the representation step and clustering step is separate. Thus, the autoencoder only functions as a feature extraction and dimension reduction technique, without gaining too much insights with the clustering objective. Naturally, one approach to improve the model is to reconcile the two steps.
Deep Embedded Clustering (DEC) is a deep clustering model proposed by Xie et al. [4] that

simultaneously improve clustering assignment and feature representation. After the training of an autoencoder model as in previous section, it improves the representation in the following way:

1. Compute a soft assignment between the embedded points and the cluster centroids.
2. Update the autoencoder representation and refine the cluster centroids by learning from current high confidence assignments using an auxiliary target distribution: $p_{ij} = \frac{q_{ij}^2/f_j}{\sum_j q_{ij}^2/f_j}$, where p is the soft assignment representing the probability to assign to each cluster.

This process is repeated until a convergence criterion is met, and we use the final representation for clustering. In short, the improvement come from a self training concept that adjusts the representation in a way that confirms high confidence predictions in clustering.

5.4 Experiment Design

- **Experimental Type 1: Pairwise Composers**
  For each pair in the 20 composers, we cluster them binarily. Binary label greatly reduces the uncertainty, so the results may serve as a baseline for analyzing and comparing each model's performance.
- **Experimental Type 2: Contrasting style Composers** We have hand picked composers that are known to have contrasting styles based on 4 stylistic period in classical music history: Baroque, Classical, Romantic, Post-Romantic.

Scarlatti, Mozart, Faure, Shostakovich	Byrd, Handel, Chopin, Scriabin
Bach, Beethoven, Schubert, Debussy	Buxtehude, Haydn, Brahms, Alkan

  This means that we are expecting to see clearly distinctive clusters.
- **Experimental Type 3: Similar style Composers** We have hand picked composers that we known to have similar styles and tries to differentiate them. The means that they are harder to separate.

Scarlatti, Bach, Byrd, Buxtehude	Beethoven, Handel, Mozart, Haydn
Brahms, Chopin, Schubert, Faure	Scriabin, Shostakovich, Debussy, Alkan

- **Experimental Type 4: All composers**

6 Results

6.1 Evalutation Metrics

We use the standard unsupervised evaluation metric and protocols for evaluation. For all the clustering algorithms that we used, let the number of clusters be the number of ground-truth categories and evaluate performance with *unsupervised clustering accuracy (ACC)*, which takes a cluster assignment from an *unsupervised* algorithm and a ground truth assignment and then finds the best matching between them. ACC is defined as following:

$$ACC = max_m \frac{\sum_{i=1}^{n} 1\{l_i = m(c_i)\}}{n}$$

where l_i is the ground-trugh label, c_i is the clustering assignment produced by the algorithm, and m ranges over all possible one-to-one mappings between clusters and labels.

6.2 Accuracy heatmap for experiment type 1: pair-wise composers

The above confusion matrix shows the pairwise composer clustering accuracy under different models. Overall, we are surprised to find out that clustering trained base on raw features has higher accuracy than clustering trained base on selected features, since the heat map shows that the overall color is a lot darker for raw features on the top row. However, the clustering accuracy using raw features is a lot more random and scattered than using selected piano roll features. From the heat map on the right

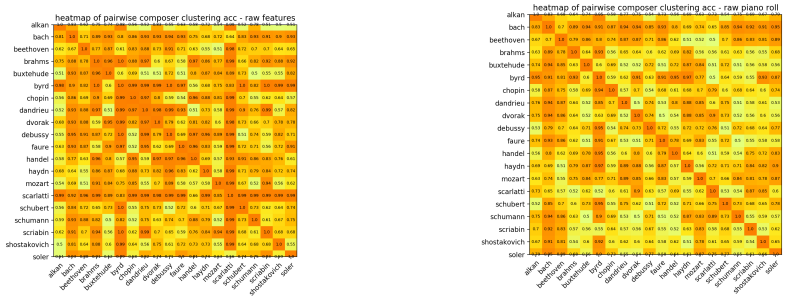

Figure 5: pairwise accuracy for: Left: raw jSymbolic features, Right: raw pianoroll representation

Figure 6: Left: autoencoder jSymbolic features, Right: convolutional ae pianoroll representation

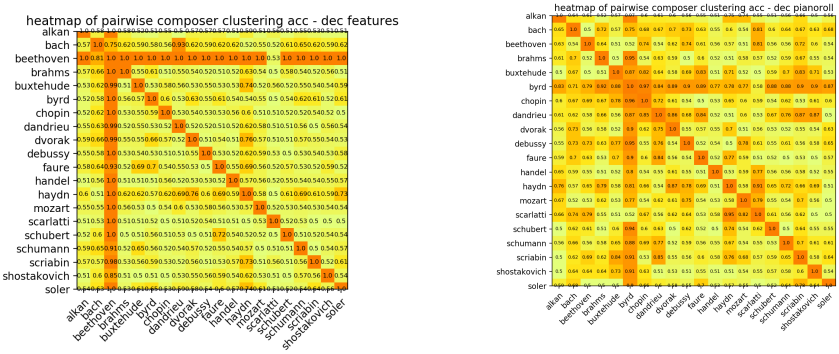

Figure 7: Deep Embedded Clustering of: Left: jSymbolic features, Right: pianoroll

we can see that, the accuracy is significantly higher for composers Beethoven (average around 0.97), Bach (average around 0.85) than for other composers, which means the piano roll method is working great for composers with large amount of data. This is an important finding since this means that as long as we can obtain more data, we can achieve very high unsupervised clustering accuracy using the piano roll selected features. Also, it's interesting to note that the clustering accuracy for composer Scarlatti is very high for both methods, which indicates that Scarlatti is very distinguishable from all other composers.

6.3 Accuracy for other types of experiments

The following statistics come from taking the average of all experimental groups among each experimental type. See 5.4 for details in experimental design.

model	Pairwise	Contrasting Composers	Similar Composers	Avg. accuracy
raw jSymbolic	0.78	0.58	0.52	0.63
ae jSymbolic	0.70	0.38	0.36	0.48
dec jSymbolic	0.62	0.39	0.45	0.49
raw pianoroll	0.71	0.56	0.41	0.56
cae pianoroll	0.67	0.46	0.33	0.49
dec pianoroll	0.65	0.47	0.43	0.52

6.4 t-SNE Representation visualization

We applied t-SNE algorithm to project the high dimensional data into a 2D space. As t-SNE algorithm preserves the relative distance, it's helpful for visualizing all compositions: The following comparison demonstrates that rather than diversifying the data, the autoencoder representation actually compresses the data into a more homogeneous space that no longer helps with clustering.

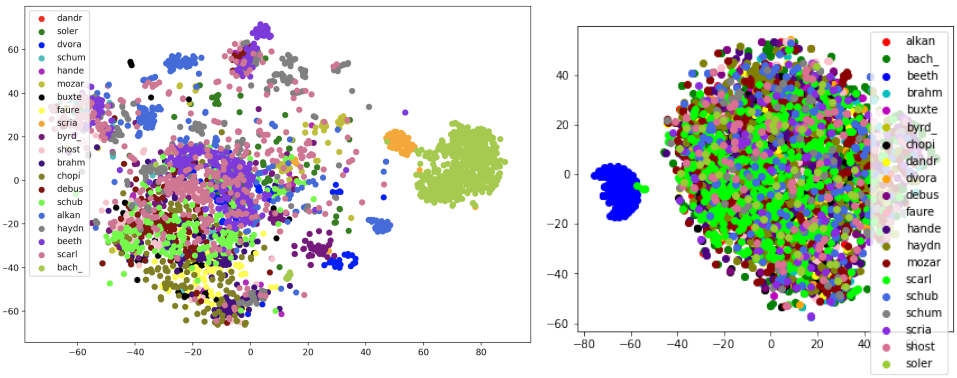

Figure 8: all works from 20 composers, L: raw jSymbolic representation, R: ae transformed jSymbolic

7 Discussion and Analysis

To summarize, in this project we tried three clustering models from low level to high level, with two versions of raw representation: MIDI pianoroll and jSymbolic music theory features. Given the subjectivity of music data, our models achieved convincing results for raw data clustering (but there is also few composition clustering research that we can compare with). However, it's clear from the accuracy summary that after feature representation and deep clustering, the result is not as good as original.

Another thing that suffers is the interpretability of results. Even the jSymbolic music theory features are interpretable in their original form, after feature transformation, they can't be related back to their original meaning.

One assumption we made is for the autoencoder training set: Given the self-reproducing objective, we used all composer's data to train the autoencoder or convolutional autoencoder. However, this might result in biased latent representation given the imbalance of each composer's data in the dataset.The latent representation for Beethoven is quite distinctive and result in great clustering accuracy while other composer's data suffers.

7.1 Thoughts on Deep clustering

However, we noticed that deep embedded clustering doesn't always increase the accuracy. The following graphic shows the accuracy change of all the pairwise experiment of jSymbolic representation: among the deep clustering step the accuracy fluctuates, and a lot of them suffers from a decrease. This means that the autoencoder in the pretraining step obtains a quite good clustering result, but the cluster center fine-tuning make things worse. After all, it's from a self-training perspective that enhances the high-confidence cluster assignment. Given the homogenity of musical data, it's likely that the wrong assignment gets enhanced.

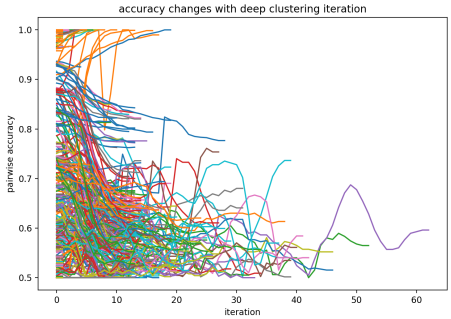

Figure 9: Accuracy Change with deep clustering iteration, pairwise experiment, jSymbolic representation

7.2 Musical Insights

With models experimented on both ways of music representation, we can reach the conclusion that jSymbolic music theory features (776 dim) performs better than the sparse pianoroll matrix (128000 dim), especially when differentiating similar composers. However, music theory features also suffers from a greater decrease after latent representation: One possible explanation is that they can be viewed as a higher level representation than pianoroll matrix.

The contrasting - similar multiple composer experiments also yields interesting result: We where expecting the contrasting composers to be more differentiable when it comes to clustering. This expectation is generally being met in all models, but not as large as we think. The all Baroque (Bach, Buxtehude, Byrd, Scarlatti) experiment group even performed much better than contrasting groups. This leads to the discovery that:

Composer with homogeneous repertoire (i.e most of Chopin's music are written for piano, all of the Byrd's music in the dataset are for solo harpsichord, and all of the Scaratti's works are in a very specific sonata form) tends to perform better in clustering. Baroque composers, due to their stylistic restrictions at an earlier period, tends to have more homogeneous corpus and thus perform well in clustering. If large-scaled symphony and art songs all simultaneously exist in one composer's corpus, then a greater proportion is going to be mistaken.

It is true that upon investigating the dataset, we realized that the the MIDI files are hardly standardized in terms of volume, length, format, due to the fact that our dataset is collected by multiple teams. For example, sketches of musical ideas are placed as part of Beethoven's corpus, which can hardly be considered a musical 'piece' with stylistic information.

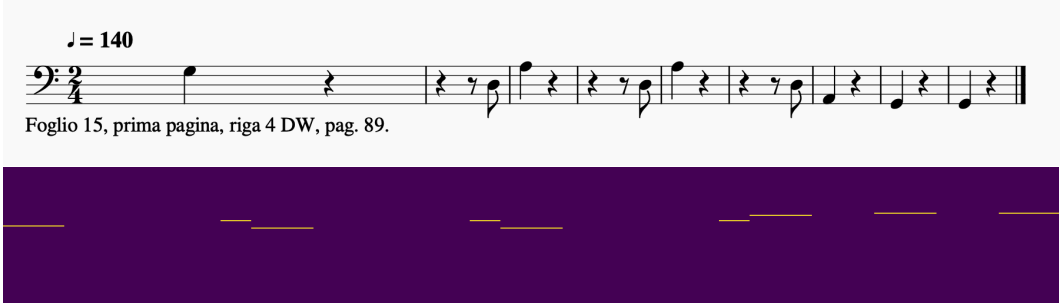

Figure 10: A full 'piece' in Beethoven's dataset

Thus the success of Beethoven in representation (both using Autoenoder and Deep Embedded Clustering) is really intriguing, given it contains such diverse type of music data.

References

[1] Georges, P., Nguyen, N. Visualizing music similarity: clustering and mapping 500 classical music composers. *Scientometrics 120*, 975–1003 (2019).

[2] R. Cilibrasi, P. Vitanyi and R. de Wolf, Algorithmic clustering of music. *Proceedings of the Fourth International Conference onWeb Delivering of Music*, 2004. EDELMUSIC 2004., Barcelona, Spain, 2004, pp. 110-117.

[3] S. Jain, A. Smit, T. Yngesjo, Analysis and Classification of Symbolic Western Classical Music by Composer.

[4] Junyuan Xie, Ross Girshick, and Ali Farhadi. 2016. Unsupervised deep embedding for clustering analysis. In Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48 (ICML'16). JMLR.org, 478–487.

[5] McKay, Cory, Ichiro Fujinaga. 2006. jSymbolic: A Feature Extractor for MIDI Files. ICMC (2006).