Graph Neural Networks For Music Genre Recognition

Shuo Feng sf587
Jingyi Chen jc2498
Cornell University

Introduction

In the last decade or so, the big messy data is well trained by deep learning in many different fields like natural language processing, computer vision, and speech recognition. The innovation speed of deep learning algorithms is very fast while no one network is considered perfect. Most generally, the standard neural network can only capture the dependency information by only regarding the features of nodes, and using Euclidean distance to detect their similarities. However, there are numerous structures of data that are generated from non-Euclidean domains which are represented as graphs with interrelationships and more complex dependencies between various entities. Hence, graph neural networks(GNNs) have emerged in machine learning and demonstrated a superior performance from the message passing between the nodes of graphs by representing information from their neighborhood with arbitrary depth.

Since music genres have been an increasingly important category of discriminative data, on which classification techniques are intensively explored to correctly address the music genre classification problem. In this project, the goal is to discover the knowledge graph structural relations using GNN on music genres classification problem where the input music sample could be represented in a form of vector or matrix in high dimension via spectrograms, MEL spectrogram and MFCC, etc. As preparation for GNN, we will use various algorithms to find sample(node) features, connection(edge) between each node and construct graph data. After implementation of GNN on data set and accuracy analysis, we will conduct several improvements on feature engineering including feature extractions and feature importances, and also try different model inner structures to improve the overall performance of GNN.

Dataset Description

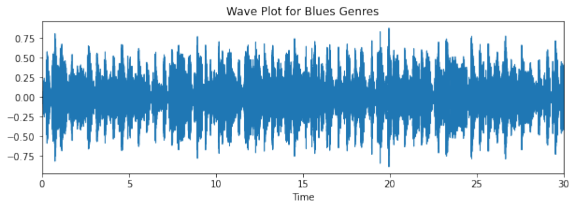
The original data is GTZAN (Tzanetakis and Cook (2002))1 in this project, collected in 2000-2001, which is a widely used dataset for evaluation in machine listening research for music genre recognition. GTZAN provides a clean distribution on a variety of audio sources and conditions in order to represent a variety of recording conditions.

GTZAN consists a file contains 10 files for each music genres, and each file contains 100 samples of 30s music audio sample of that genre.

For the midterm project, we only explore two music genres classification for better discovering the knowledge graph of audio features and pruning the model . Therefore the data set we used contains 'blues' music and 'jazz' music samples. Total data is splitted into training and validation by 77%, 33%, respectively.

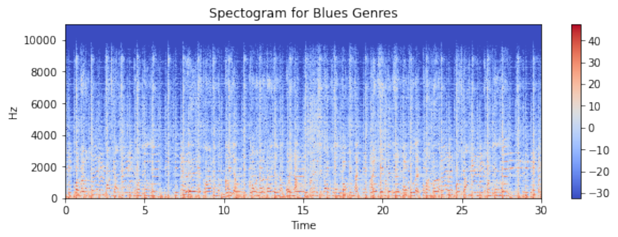
Feature Extraction

For this project, the features of audios are the physical features that refer to mathematical measurements captured from sound waveform. Waveform is formed by sample rate(frequency) and sample data(amplitude).



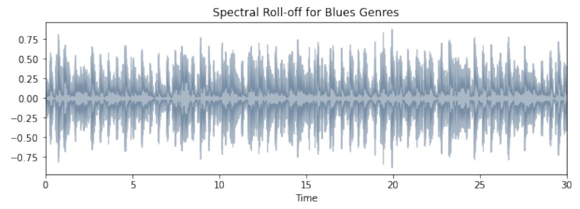
Spectrogram

Spectrogram represents the signal strength(energy) of a signal over time at various frequencies. Time runs along the horizontal axis, the frequency is represented by the vertical axis. The color is amplitude refers to energy.



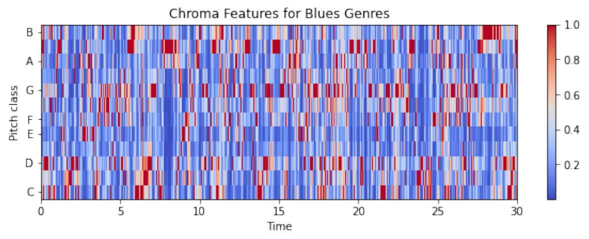
Spectral Roll-Off

Spectral Rolloff is the frequency under some specified percentage of the total spectral energy. This feature can detect the difference between the harmonic which is below the roll off and noisy sounds which is above the roll-off.



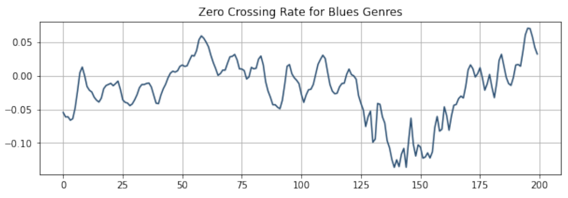
Chroma Features

Chroma Features is related to pitches using the entire spectrum to project onto 12 distinct chroma. For music genres classification, it is important to include the pitches since it can capture harmonic and melodic characteristics of music.



Zero Crossing Rate

Zero crossing is defined by the number of the speech signal changes from positive to zero to negative or inverse.



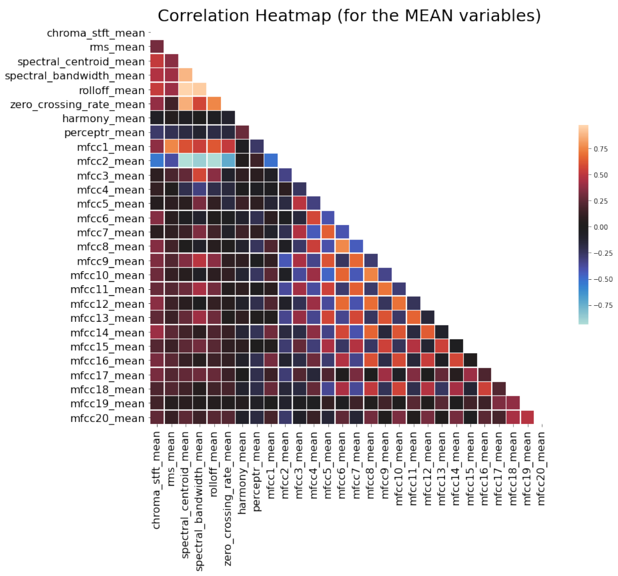
MFCC

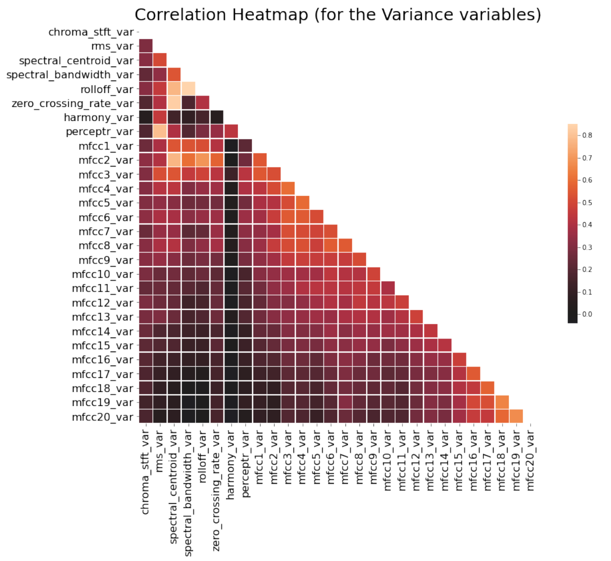
In sound processing, the mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency. Mel-frequency cepstral coefficients (MFCCs) are coefficients that collectively make up an MFC. They are derived from a type of cepstral representation of the audio clip (a nonlinear "spectrum-of-a-spectrum") [1].

For feature extraction, we combined the features that were mentioned before and computed each feature's mean and variance to represent each music genre. Therefore, the total features we have are 57 feature sets.

Data Visualization

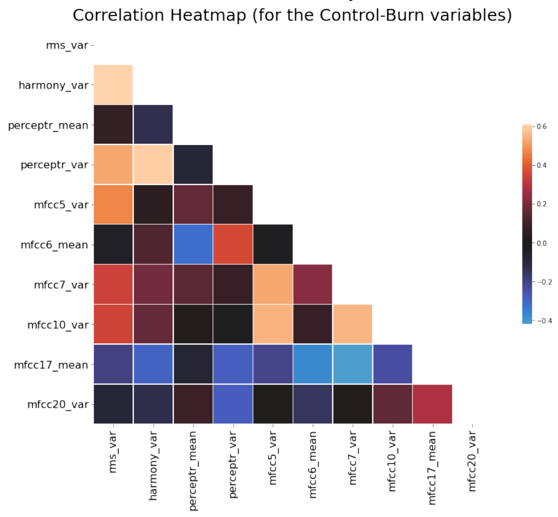
Correlation Heatmap: The correlations between mean variances variables having many high correlated relationships. This might influence the model overfitting issue. We will use control burn technique to get rid of it.





Control Burn

Since there are 57 features for each sample, there might be some correlated features which reinforce these feature groups, decreasing interpretability and increasing complication and overfitting. In such a scenario, control Burn is a powerful algorithm on feature selection. Control Burn uses a weighted LASSO-based feature selection method to prune unnecessary features from tree ensembles and assigns all the feature importance of a correlated group of features to a single feature[2]. We implement Control Burn on the 57 features to prune unnecessary and correlated features, and keep only 10 features as final features. Control Burn has demonstrated that it can largely improve our classification accuracy. The validation set accuracy without Control Burn is 63% while validation set accuracy after Control Burn is 72%.



Graph model

GNN

A graph is a data structure consisting of two components: node and edge. Graph neural network (GNN) is a deep learning method that works in the graph domain, providing an easy way to do node-level, edge-level, and graph-level prediction tasks. GNN is used in this data set to model music features interaction and connection as knowledge graphs. In our case, we are trying to use GNN to apply a node classification problem.

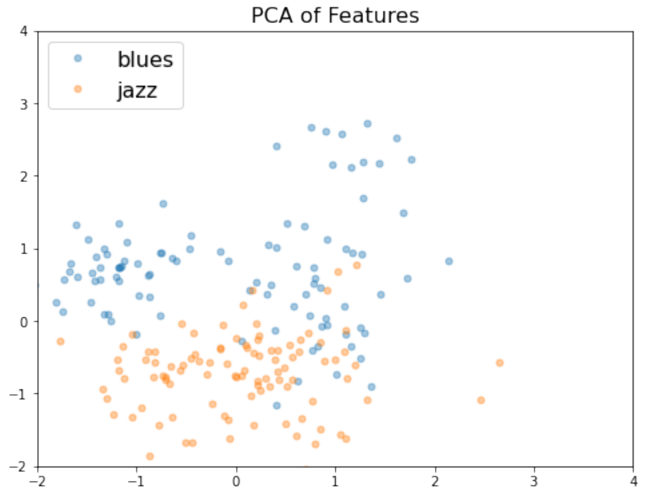
Construct Graph

Since GNN takes a knowledgeable graph as input, we need to first construct the graph data for GNN. Suppose each sample serves as a node, we need to know the edge between each node, in the other words, how those nodes are connected to each other, whether they are connected and how strong is the connection. Unlike other data such as matrix and vector, since graph data works in the non-euclidean space, we need algorithms to compute the edge between each pair of nodes. We implement several steps to compute the edge between nodes.

First, we use Principal Component Analysis(PCA) to reduce each feature vector in dimension of 1×57 to a vector in dimension of 1×2, which enables nodes to be represented in a plane, and then find the edges of those nodes.

Second, after PCA, for each node, we use k-nearest neighbors to find k neighbours of each node, setting k equals 120, that is, for each music sample node, we find the nearest k music sample mode around it, which means for this node, those k nodes are connected to it.

Finally, these edge and node connection need to be represented in a knowledge graph as the input of GNN. To realize this, introduce Adjacency matrix: $A_{i,j} = 1$ if (i, j) is an edge, 0 if it is not. Repeat this process to compute k neighbours for each node and update the adjacency matrix. The last preparation is to convert the Adjacency matrix to network using tools that will be used for input into GraphSage.

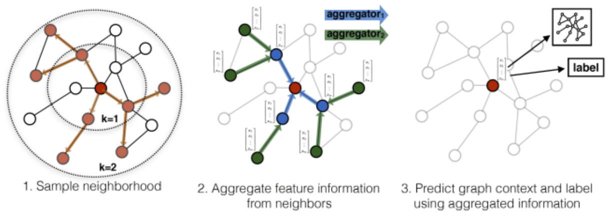


Graph Sage Model

In the node classification problem set up, each node v is characterized by its features x_v, and ground trues label t_v. In our case, x_v is the PCA features in two dimensions. $x_{co[v]}$ denotes the features of the edges connecting with v, and the embedding of the neighboring nodes of v is denoted by $x_{ne[v]}$. In our case, the neighboring nodes are calculated by KNN algorithms. The model digest the information from the given labels graph G to predict the unlabeled nodes by finding the best solution of function f which is denoted as h_v(weights).

$$\mathbf{h}_v = f(\mathbf{x}_v, \mathbf{x}_{co[v]}, \mathbf{h}_{ne[v]}, \mathbf{x}_{ne[v]})$$

To avoid the computationally expensive modifications in an inductive setting which is requiring additional rounds of gradient descent before new predictions can be made, we conduct GraphSAGE model. Instead of training a distinct embedding vector for each node, we train a set of aggregator functions that learn to aggregate feature information from a node's local neighborhood[3].


k=1
k=2
aggregator
aggregator
label
1. Sample neighborhood
2. Aggregate feature information from neighbors
3. Predict graph context and label using aggregated information

The aggregator we used for this project is the mean aggregator which takes the average of the latent vectors of the node and its related neighborhood.

Result

Loss Function

For binary classification, we used cross entropy as loss function which calculated the scores that summarized the average difference between the true label and predicted probability distributions for predicting class 1.

Cross Validation

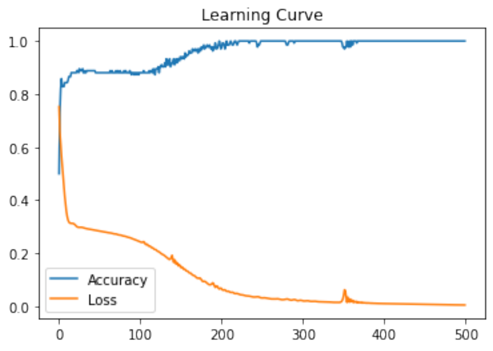
We design 10 times cross validation to have more robust results.

Model Evaluation

Accuracy: The number of corrected labels/ The total number of test set

57 features: Accuracy 0.63 10 features selected by control burn: 0.72

Learning Curve



Improvement

Adding More Labels

Since in the current model the feature space is reduced by PCA to a vector of (1*2), although accuracy is not bad, we will try to include more features to implement the model.

Dimensional Reduction From PCA to TSNE

PCA and TSNE[4] (T-Distributed Stochastic Neighbouring Entities) both are dimension reduction techniques. Difference between them is that PCA uses mathematics, using the eigenvalues and eigenvectors of the data-matrix to provide a minimum number of variables that keeps the maximum amount of variation, while TSNE is a proablistic one, that is, TSNE looks at the original data and looks at how to best represent this data using less dimensions by matching both distributions. Therefore, we will try TSNE and compare it with PCA.

Graph Edge

We might try different method to find node edge.

Adding More Features Set

Feature extraction is also an important part of the model. We might try different methods of feature extraction, such as CNN, Siamese Network , wav2vec, Gempas, and compare classification results.

Cosine Similarity

Our current model uses Euclidean distance in KNN model to compute neighbours, and might try other distance measurement such as Cosine similarity.

References

[1] Min Xu; et al. (2004). "HMM-based audio keyword generation" (PDF). In Kiyoharu Aizawa; Yuichi Nakamura; Shin'ichi Satoh (eds.). Advances in Multimedia Information Processing – PCM 2004: 5th Pacific Rim Conference on Multimedia. Springer. ISBN 978-3-540-23985-7. Archived from the original (PDF) on 2007-05-10.

[2] Liu, Brian and Xie, Miaolan and Udell, Madeleine. ControlBurn. Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. Aug 2021.

[3] Hamilton, Ying, and Leskovec, Inductive Representation Learning on Large Graphs,2017

[4] Laurens van der Maaten, Geoffrey Hinton. Visualizing Data using t-SNE.Journal of Machine Learning Research 9 (2008) 2579-2605. 11/08.