Genre Classification: A Transductive, Inductive and Deep Approach

Group 32:
Isinsu Katircioglu, Sena Kıcıroğlu
George Adaimi, Okan Altingovde

1) Introduction

Many real world problems can be represented by graphs and analyzed for the task of graph, node or edge label prediction. In this project we tackle the Music Genre Classification (MRC) problem. Classifying genre has lately become prominent especially with the current popularity of streaming music services. Potential applications can be forming playlists from massive amounts of data, song recommendation based on genre, etc.

Throughout the project, we make use of the publicly available Free Music Archive (FMA) dataset [1]. First, we attempt to get to know the dataset and the graphs we can produce using the features. Afterwards, we propose machine learning and graph based algorithms to estimate the genre for each songs. Ultimately, we attempt to train a graph convolutional neural network (GCN), a method that makes use of both the feature vectors of each node and the adjacency matrix of the entire graph. Since our data is plentiful, our intuition is that a data-driven approach such as training a deep neural network will be useful for our task.

Our results are reported in tables and figures in the results section.

2) Exploration

2.1) The FMA Dataset

We use the Free Music Archive Dataset, containing 106,574 songs from 16,341 artists. As mentioned in [1], this dataset has three versions, namely small, medium and large, that are generated based on the size. We construct a more balanced subset for these three versions by selecting the songs that correspond to the eight genres containing the most songs. The train and test split is done as explained in [1].

The dataset contains both audio tracks and readily-calculated feature vectors for each song. We have made use of the mel-frequency cepstral coefficients (MFCC) features in our project, as they are widely recognized in the genre classification community to be descriptive. In order to form the adjacency matrix, we calculate the pairwise distances between features using the "correlation" metric.

We threshold the edge values in the adjacency matrix so that we only keep the values above 0.9. We have deliberately set the threshold to a high number, resulting in the graph being sparser. When doing this operation, we sometimes end up with disconnected nodes (nodes that are not connected to any other node in the graph). We simply chose to remove such nodes from the graph. This operation is necessary to run the graph neural network, since otherwise we run into GPU memory problems.

2.2) Growth Analysis and Graph Statistics

The statistics of our graph is given in Table 1. We see that our graph is fairly sparse, however the average degree (48) is quite large compared to real-world graphs as we have seen in the course. We have many connected components (214), but our largest component contains most of our nodes (96%). Figure 2.2) shows the degree distribution of our graph is logarithmic scale. This distribution most resembles a power-law distribution.

Number of Nodes	18648
Number of Edges	447280
Sparsity	0.2%
Average Degree	48.0
Number of Connected Components	214
Size of Largest Connected Component	18084
Average Clustering Coefficient	0.3396480816933378

Table 1: Statistics for the large subset

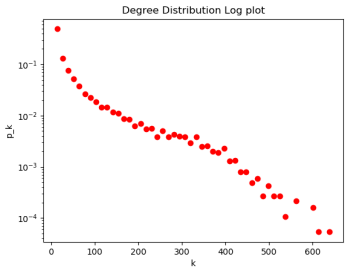
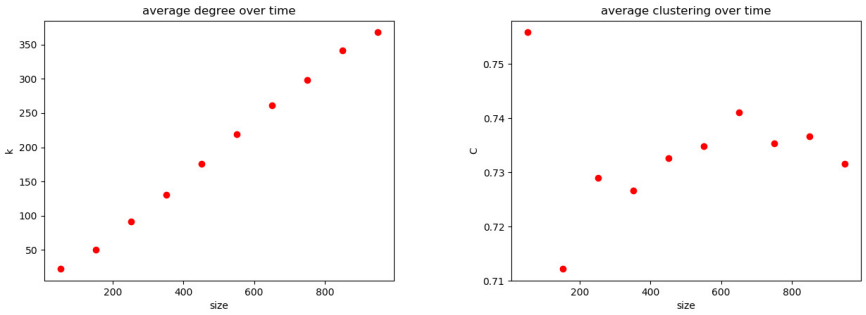


Figure 1: The degree distribution of our graph in log scale.

We have access to the release dates of the majority of music tracks which allows us to analyze the growth of the graph over time. Inspection of time variant or invariant aspects of the data provide interesting information, as in our case, for better understanding.

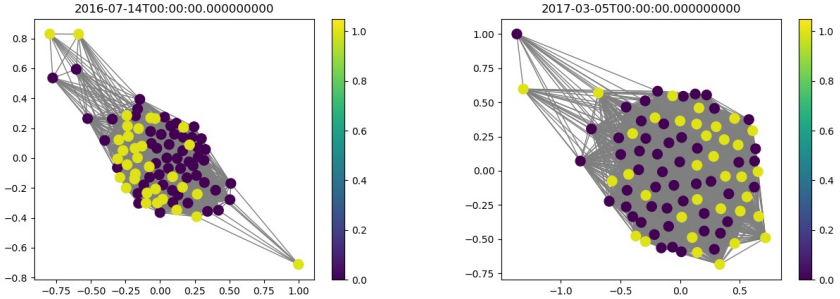
In order to analyze growth of our graph, we take multiple time instances of the graph and calculate statistics to compare between different instances. We calculate average clustering coefficient and average degree for every instances of the graph. Since our graph is based on similarity of its nodes, we found these properties particularly interesting for us to understand the limits and possible exploits of data. We find from the plots in 2 that average degree increases linearly with graph size while average clustering coefficient does not change.



(a) Average degree over time. (b) Average clustering coeff. over time.

Figure 2: These plots display the growth of our graph as we add nodes according to the release date of each track.

Another property that we think is interesting is the change in the distribution of features for a specific genre through time. Though this is not directly related to genre classification, it could be another direction to explore in a future project. To this end, we choose a genre and split its nodes into two classes of songs released before and after 2012 in our time-line. We plot the resulting graph to see if they also form clusters which is sometimes the case. In Figure 3 we show sub-genre clusters formed in time for Rock and Hip-Hop classes respectively. For the Rock genre, we can observe that the old songs and new songs form clusters among each other, whereas we do not observe this happening in Hip-Hop. This is reasonable considering that Rock has been around for a longer time than Hip-Hop and has had more time to evolve within itself.



(a) The distribution of rock songs (b) The distribution of hip-hop songs

Figure 3: The distribution of songs within a single genre according to release year. The data has been labelled as 0 (purple) for older songs and 1 (yellow) for newer songs.

3) Exploitation

3.1) Evaluation Methods

During our training we use K-fold cross validation with K set to 5. In order to do cross validation, we have combined the training and validation splits already provided by the dataset. We loop through this data 5 times, each time selecting a different portion for our validation. In order to test our data, we use the test split of the dataset. The results and confusion matrices are reported in the results section.

3.2) Pre-processing

In order to get a representative graph, we have tested the following operations on our feature vectors.

- The first method is to normalize the data and run PCA on it, reducing the dimensionality to 120. This method is used to remove unnecessary feature vectors from the dataset.
- The second method is to use a multi-layer perceptron (MLP) to learn good features. These features are then used to form the adjacency, and since they are formed to specifically perform well on genre recognition, we can say that they help in forming a graph that is more tailored for genre recognition. We refer to these features throughout this report as the MLP-features.

3.3) Classical Machine Learning Methods

We evaluate our dataset using several classical machine learning models. These methods only utilize the feature vectors of the data, meaning that we do not make use of our graph knowledge for the most part. We have tested the methods support vector machines (SVM), random forests (RF), K-nearest neighbours (KNN), and multi-layer-perceptrons (MLP).

3.4) Transductive Learning

Inductive models, such as the classical machine learning methods we have discussed above aim to learn a general model from training data, then use this general model to evaluate test data. In contrast, transductive learning aims to learn from training data in order to evaluate specific test data. In order to do this, transductive models utilize not only the connectivity of the training data, but of the entire graph. Even though we only know the labels of the training data, we still use our knowledge of the entire graph's layout to predict the unknown labels.

We evaluate several transductive learning methods, listed below.

- Gaussian fields and harmonic functions (GFHF): This method tackles the unlabeled nodes with a kernel classifier and minimizes a harmonic energy function obtained through a Gaussian random field model defined on the graph [2]. It finds the harmonic solution by heat diffusion on the unlabeled subgraph.
- OMNI-Prop: This approach is based on an iterative approach that updates a self-score and a follower score in a Bayesian inference manner [3]. It relies on the hypothesis that if the majority of the followers of a node have a label k then the rest also have the same.
- Partially absorbing random walks (PARW): This stochastic process defines a diffusion during which a random walk is absorbed by probability p at a vertex i and flows with a probability $1 - p$ [4]. As the process continues the flow stored will sum up to 1.

3.5) Graph Neural Networks

In this section, we leverage the supervised learning of deep neural networks and combine it with the structural information embedded in our graph to propose a graph convolutional neural network (GCN) based approach for genre classification. Formally, consider a graph $G = (V, E)$ where V and E are the set of nodes and edges respectively. Let $X \in \mathbf{R}^{\mathbf{N}\times\mathbf{M}}$ be the matrix containing N nodes with their features of dimension M and each row x_n is the feature vector of node n. We also introduce an adjacency matrix A and its degree matrix D. The goal of our model is then to learn a function of features on our graph defined as

$$Y = f_{gcn}(X, A) \tag{1}$$

where $y_n \in \{0, 1\}^C$ to predict one of C many genres for each node. The adjacency matrix serves as a way of aggregating the feature vectors from the neighborhood of a node. However, as the original graph G does not contain self-loops, we add identity matrix to A in order to take into account the feature vector of the central node. We normalize this new adjacency matrix $\hat{A}$ with its degree matrix $\hat{D}$ by $\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}$. We can build

our network by stacking L many layers where each layer in our transforms the given features into a new set of features by taking the average of the neighbouring features for a particular node as follows:

$$Z^{l+1} = f_l(Z^l, A) = \sigma(\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}Z^l W^l) \tag{2}$$

where $Z^0 = X$ and $Z^L = Y$.

3.5.1) K-Hop Graph Convolution

We define the k-hop neighborhood as $\{v_j \in V | d(v_i, v_j) <= k\}$ for each node i. We can compute the k-hop adjacency matrix as the k-th power of A such that

$$\hat{A} = (\prod_{j=1}^{k} A + I) \tag{3}$$

and we can acquire the normalized adjacency matrix as explained in the previous section. Exploiting the k-hop adjancency map together with the one-hop adjancecy map, A, in parallel and combining their features through a linear layer allows us reach out the features of the distant neighbours of a node while doing classification.

3.5.2) Implementation Details

We take a 1 layer GCN with randomly initialized weights and we use ReLU as the activation function. The size of the latent feature vector is set to 1000. We use Adam with learning rate 0.01 to optimize the parameters. To avoid overfitting we apply dropout of 0.1 probability after each graph convolutional layer and do early stopping based on our validation accuracy.

In the k-hop graph neural network, we select k to be 2 and we concatenate the output features coming from the layers using the one-hop and two-hop adjacency matrices and regress them to the class probabilities through a fully connected layer of size 16.

4) Results

We evaluate our results on the 8 genres containing the most songs for each dataset. Note that this does not necessarily mean the same genres for both datasets.

In Table 2 we show the results evaluated on the medium and large datasets. Our error values are lower for the medium size dataset than the large one. On the medium dataset, GCN-KHop performs the best, whereas on the large dataset RF gives the lowest error.

	Medium Dataset		Large Dataset	
Method	Error (%)	Training Time (s)	Error (%)	Training Time (s)
SVM	20.38	85	37.13	160
RF	21.44	46	**36.46**	36
KNN	21.51	39	37.07	48
MLP	20.59	68	38.02	90
GCN	20.53	256	37.78	241
GCN-KHop	**20.24**	731	37.56	1655
GFHF	22.15	93	38.81	140
OMNI-Prop	22.43	303	40.15	339
PORW	26.26	0.7	38.36	0.635

Table 2: Comparison of our methods on the medium and large test sets for 8 genre classification.

We evaluate using PCA and MLP-features as pre-processing methods in Table 3. We use our classical methods since they are relatively faster to train and test. From the results, we find that performing PCA by reducing the dimensionality to 120 and finding the MLP-features on top of this overall improves the results. Therefore, we report all of our results in Table 2 with PCA and MLP-features.

5) Discussion and Conclusion

As for future problems, it could be interesting to explore the possibility of learning in what year a song was released. We have already seen from our exploration of the genres that visually, songs dating before a certain

	SVM	RF	KNN
% Error with MLP & PCA	**20.38**	**21.44**	21.51
% Error w/o MLP, with PCA	21.86	26.48	**21.13**
% Error w/o PCA w/o MLP	-	26.87	27.95

Table 3: Ablation study for MLP and PCA using classical methods

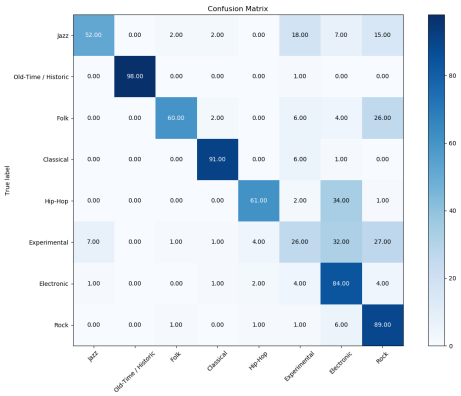


Figure 4: Confusion matrix of results of SVM run using PCA and MLP features on the medium dataset. The confusion matrix shows us the accuracy of each genre within the dataset. We can see that we perform very well with the genres "Old-time / Historic", "Classical", and "Rock". However, the genres "Experimental" and "Hip-Hop" are confused with "Electronic". This is reasonable, considering that humans also sometimes have difficulty with these genres.

year and after a certain year are well clustered among themselves. This could be an interesting direction to explore.

In this project we have explored the FMA dataset and used different methods for genre classification. It was very interesting for us to understand the properties of the graph we found, as well as the process of designing a GCN.

References

[1] M. Defferrard, K. Benzi, P. Vandergheynst, X. Bresson. FMA: A Dataset for Music Analysis. arXiv preprint arXiv:1612.01840, 2017.

[2] X. Zhu, Z. Ghahramani, J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. ICML, 2003.

[3] Y. Yamaguchi, C. Faloutsos, H. Kitagawa. OMNI-Prop: Seamless Node Classification on Arbitrary Label Correlation. AAAI, 2015.

[4] X.M. Wu, Z. Li, A. M. So, J. Wright, S. F. Chang. Learning with partially absorbing random walks. NIPS, 2012.