Music Genre Classification

Michael Haggblade **Yang Hong** **Kenny Kao**

1 Introduction

Music classification is an interesting problem with many applications, from Drinkify (a program that generates cocktails to match the music) to Pandora to dynamically generating images that complement the music. However, music genre classification has been a challenging task in the field of music information retrieval (MIR). Music genres are hard to systematically and consistently describe due to their inherent subjective nature.

In this paper, we investigate various machine learning algorithms, including k-nearest neighbor (k-NN), k-means, multi-class SVM, and neural networks to classify the following four genres: classical, jazz, metal, and pop. We relied purely on Mel Frequency Cepstral Coefficients (MFCC) to characterize our data as recommended by previous work in this field [5]. We then applied the machine learning algorithms using the MFCCs as our features.

2 Our Approach

2.1 Data Retrieval Process

Marsyas (Music Analysis, Retrieval, and Synthesis for Audio Signals) is an open source software framework for audio processing with specific emphasis on Music Information Retrieval Applications. Its website also provides access to a database, GTZAN Genre Collection, of 1000 audio tracks each 30 seconds long. There are 10 genres represented, each containing 100 tracks. All the tracks are 22050Hz Mono 16-bit audio files in .au format [2]. We have chosen four of the most distinct genres for our research: classical, jazz, metal, and pop because multiple previous work has indicated that the success rate drops when the number of classifications is above 4. [4] Thus, our total data set was 400 songs, of which we used 70% for training and 30% for testing and measuring results.

We wrote a python script to read in the audio files of the 100 songs per genre and combine them into a .csv file. We then read the .csv file into Matlab, and extract the MFCC features for each song. We further reduced this matrix representation of each song by taking the mean vector and covariance matrix of the cepstral features and storing them as a cell matrix, effectively modeling the frequency features of each song as a multi-variate Gaussian distribution. Lastly, we applied both supervised and unsupervised machine learning algorithms, using the reduced mean vector and covariance matrix as the features for each song to train on.

2.2 Mel Frequency Cepstral Coefficients (MFCC)

For audio processing, we needed to find a way to concisely represent song waveforms. Existing music processing literature pointed us to MFCCs as a way to represent time domain waveforms as just a few frequency domain coefficients (See Figure 1).

To compute the MFCC, we first read in the middle 50% of the mp3 waveform and take 20 ms frames at a parameterized interval. For each frame, we multiply by a hamming window to smooth the edges, and then take the Fourier Transform to get the frequency components. We then map the frequencies to the mel scale, which models human perception of changes in pitch, which is approximately linear below 1kHz and logarithmic above 1kHz. This mapping groups the frequencies into 20 bins by calculating triangle window coefficients based on the mel scale, multiplying that by the frequencies, and taking the log. We then take the Discrete Cosine Transform, which serves as an approximation of the Karhunen-Loeve Transform, to decorrelate the frequency components. Finally, we keep the first 15 of these 20 frequencies since higher frequencies are the details that make less of a difference to human perception and contain less information about the song. Thus, we represent each raw song waveform as a matrix of cepstral features, where each row is a vector of 15 cepstral frequencies of one 20 ms frame for a parameterized number of frames per song.

We further reduce this matrix representation of each song by taking the mean vector and covariance matrix of the cepstral features over each 20ms frame, and storing them as a cell matrix. Modeling the frequencies as a multi-variate Gaussian distribution again compressed the computational requirements of comparing songs with KL Divergence.

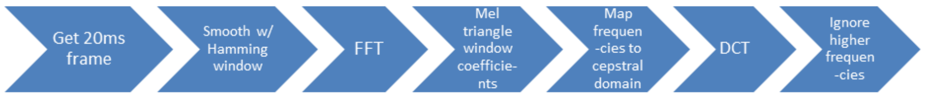

Get 20ms frame
Smooth w/ Hamming window
FFT
Mel triangle window coefficie-nts
Map frequen-cies to cepstral domain
DCT
Ignore higher frequen-cies

Figure 1: MFCC Flow

3 Techniques

3.1 Kullback-Lieber (KL) Divergence

The fundamental calculation in our k-NN training is to figure out the distance between two songs. We compute this via the Kullback-Leibler divergence [3]. Consider p(x) and q(x) to be the two multivariate Gaussian distributions with mean and covariance corresponding to those derived from the MFCC matrix for each song. Then, we have the following:

$$2KL(p||q) = \log \frac{|\Sigma_q|}{|\Sigma_p|} + Tr(\Sigma_q^{-1}\Sigma_p) + (\mu_p - \mu_q)^T \Sigma_q^{-1}(\mu_p - \mu_q) - d$$

However, since KL divergence is not symmetric but the distance should be symmetric, we have:

$$D_{KL}(p, q) = KL(p||q) + KL(q||p)$$

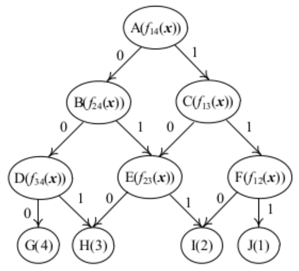
3.2 k-Nearest Neighbors (k-NN)

The first machine learning technique we applied is the k-nearest neighbors (k-NN) because existing literature has shown it is effective considering its ease of implementation. The class notes on k-nearest neighbors gave a succinct outline of the algorithm which served as our reference.

3.3 k-Means

For unsupervised k-means clustering to work on our feature set, we wrote a custom implementation because we had to determine how to represent cluster centroids and how to update to better centroids each iteration. To solve this, we chose to represent a centroid as if it were also a multi-variate Gaussian distribution of an arbitrary song (which may not actually exist in the data set), and initialized the four centroids as four random songs whose distances (as determined by KL divergence) were above a certain empirically determined threshold. Once a group of songs is assigned to a centroid, the centroid is updated according to the mean of the mean vectors and covariance matrices of those songs, thus represented as a new song that is the average of the real songs assigned to it. Finally, as random initialization in the beginning and number of iterations are the two factors with notable influence on the cluster outcomes, we determined the iteration number empirically and repeatedly run k-means with different random initial centroids and pick the best, as determined by the calculated total percent accuracy.

3.4 Multi-Class Support Vector Machine (DAG SVM)



A directed acyclic graph of 2-class SVMs

SVM classifiers provide a reliable and fast way to differentiate between data with only two classes. In order to generalize SVMs to data falling into multiple classes (i.e. genres) we use a directed acyclic graph (DAG) of two-class SVMs trained on each pair of classes in our data set (eg $f_{14}(x)$ denotes the regular SVM trained on class 1 vs class 4) [1]. We then evaluate a sequence of two-class SVMs and use a process of elimination to determine the output of our multi-class classifier.

3.5 Neural Networks

We tried neural networks because it has proved generally successful in many machine learning problems. We first pre-process the input data by combining the mean vector and the top half of the covariance matrix (since it is symmetric) into one feature vector. As a result, we get $15+(1+15)*\frac{15}{2}$ features for each song. We then process the output data by assigning each genre to an element in the set of the standard orthonormal basis in R^4 for our four genres, as shown in the table below:

Genre	Classical	Jazz	Metal	Pop
Vector	(1, 0, 0, 0)	(0, 1, 0, 0)	(0, 0, 1, 1)	(0, 0, 0, 1)

We then split the data randomly by a ratio of 70-15-15: 70% of the data was used for training our neural network, 15% of the data was used for verification to ensure we dont over-fit, and 15% of the data for testing. After multiple test runs, we found that a feedforward model with 10 layers for our neural network model gives the best classification results.

4 Results

Table 1: DAG SVM Results

		Actual			
		Classical	Jazz	Metal	Pop
Predicted	Classical	29	4	1	1
	Jazz	1	20	1	0
	Metal	0	4	26	0
	Pop	0	2	2	29
Accuracy		97%	67%	87%	97%

Table 2: Neural Network Results

		Actual			
		Classical	Jazz	Metal	Pop
Predicted	Classical	14	0	0	0
	Jazz	1	12	4	0
	Metal	0	0	13	0
	Pop	1	0	0	19
Accuracy		88%	100%	76%	100%

Table 3: k-Means Results

		Actual			
		Classical	Jazz	Metal	Pop
Predicted	Classical	14	16	0	0
	Jazz	2	27	1	0
	Metal	0	0	27	3
	Pop	0	1	1	28
Accuracy		88%	61%	93%	90%

Table 4: k-NN Results

		Actual			
		Classical	Jazz	Metal	Pop
Predicted	Classical	26	9	0	2
	Jazz	4	20	4	1
	Metal	0	1	24	0
	Pop	0	0	2	27
Accuracy		87%	67%	80%	90%

Classification accuracy varied between the different machine learning techniques and genres. The SVM had a success rate of only 66% when identifying jazz , most frequently misidentifying it as classical or metal. The Neural Network did worst when identifying metal with a 76% success rate. Interestingly, the Neural Network only ever misidentified metal as jazz. k-Means did well identifying all genres but Jazz, which was confused with Classical 36% of the time. k-NN had difficulty differentiating between Metal and Jazz in both directions. Of its 33% failures identifying

Jazz, it misidentifies as Metal 90% of the time. Similarly, k-NN incorrectly predicted that Metal songs would be Jazz in 66% of all its failed Metal identifications.

Overall we found that k-NN and k-means yielded similar accuracies of about 80%. A DAG SVM gave about 87% accuracy and neural networks gave 96% accuracy.

5 Conclusion

5.1 Discussion

Our algorithms performed fairly well, which is expected considering the sharply contrasting genres used for testing. The simpler and more naive approaches, k-NN (supervised) and k-Means (unsupervised), predictably did worse than the more sophisticated neural networks (supervised) and SVMs (unsupervised). However, we expected similar performance from SVMs and neural networks based on the papers we read, so the significant superiority of neural networks came as a surprise. A large part of this is probably attributable to the rigorous validation we used in neural networks, which stopped training exactly at the maximal accuracy for the validation data set. We performed no such validation with our DAG SVM.

5.2 Future Work

Our project makes a basic attack on the music genre classification problem, but could be extended in several ways:

Our work doesn't give a completely fair comparison between learning techniques for music genre classification. Adding a validation step to the DAG SVM would help determine which learning technique is superior in this application.

We used a single feature (MFCCs) throughout this project. Although this gives a fair comparison of learning algorithms, exploring the effectiveness of different features (i.e. combining with metadata from ID3 tags) would help to determine which machine learning stack does best in music classification.

Since genre classification between fairly different genres is quite successful, it makes sense to attempt finer classifications. The exact same techniques used in this project could be easily extended to classify music based on any other labelling, such as artist.

References

[1] *Chen, P., Liu, S..* ”An Improved DAG-SVM for Multi-class Classification” `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=0566976`.

[2] *Marsyas*. ”Data Sets” `http://marsysas.info/download/data\_sets`.

[3] *Mandel, M., Ellis, D..* ”Song-Level Features and SVMs for Music Classification” `http://www.ee.columbia.edu/~dpwe/pubs/ismir05-svm.pdf`.

[4] *Li, T., Chan, A., Chun, A..* ”Automatic Musical Pattern Feature Extraction Using Convolutional Neural Network.” IMECS 2010. `http://www.iaeng.org/publication/IMECS2010/IMECS2010\_pp546-550.pdf`.

[5] *Fu, A., Lu, G., Ting, K.M., Zhang, D..* ”A Survey of Audio-Based Music Classification and Annotation” IEEE Transactions on Multimedia. `http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5664796&tag=1`.