Supervised Learning in Genre Classification

Mohit Rajani and Luke Ekkizogloy
{i.mohit,luke.ekkizogloy}@gmail.com
Stanford University, CS229: Machine Learning, 2009

Introduction & Motivation

Now that music collections can easily exceed many thousands of songs, organization and classification of these collections are more important than ever. Song genre tends to be an attribute that can help in selecting songs to play. However, in MP3s the genre metadata tends to be non-present. It would be very useful to be able to determine genre by analyzing the music directly. Song genre tends to be quite subjective and genre tends to be applied in a general fashion to an artist as a whole. It would be more advantageous to group songs together that had the same timbre (a.k.a. shape or color of sound). The associated genre of music tends to be closely associated with timbre and is not necessarily an attribute of the artist. We try to address the problem of genre classification using only the audio content of a music file and techniques from machine learning.

Feature Vectors and Learning Algorithms

Mel-frequency Cepstrum Coefficients (MFCCs)

MFCCs are a short-time spectral decomposition of an audio signal that conveys the general frequency characteristics important to human hearing. MFCCs are commonly used in the field of speech recognition. Recent research[2] shows that MFCCs are capable of capturing useful sound characteristics of music files as well. Our premise is that MFCC's contain enough information about the timbre of a song to perform genre classification. To compute these features, a sound file is subdivided into small frames of about 20 ms each and then MFCCs are computed for each of these frames. Since the MFCCs are computed over short intervals of a song, they do not carry much information about the temporal attributes of a song, such as rhythm or tempo.

Process for converting waveforms to MFCCs

The general process for converting a waveform to its MFCCs is described in Logan[2] and roughly explained by the following pseudocode:

$$\mathrm{DCT}(\log_{10}(\mathrm{abs}(\mathrm{FFT}(Signal)))) \Rightarrow \mathrm{MFCCs}$$

1. Take the Fourier Transform of a frame of the waveform.
2. Smooth the frequencies and map the spectrum obtained above onto the mel scale.
3. Take the logs of the powers at each of the mel frequencies.
4. Take the discrete cosine transform of the list of mel log powers, as if it were a signal.
5. The MFCCs are the amplitudes of the resulting spectrum.

We compute N MFCC coefficients for all short duration frames of a wav file and store them in a F x N size matrix, where F is the number of frames. Figure 1 displays the MFCCs for sample songs in each genre, with $N = 13$.

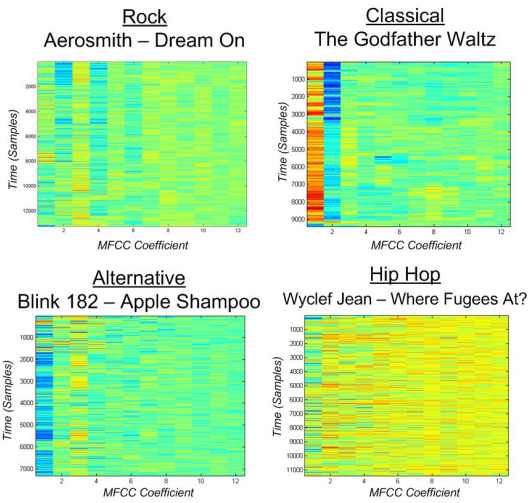

Figure 1: Graphical representations of MFCCs for each genre

Song Level Modeling

Once the MFCC feature vectors for each frame of a song have been computed, an N-dimensional gaussian is fit to this data (where $N = 13$)[1]. Hence all MFCC vectors of each song are modeled as coming from a multivariate gaussian distribution. Using maximum likelihood estimation, the optimal gaussian is calculated by simply taking the mean and covariance of the F x N MFCC matrix. Thus, ultimately each song is reduced to a 1 x N mean vector and an N x N covariance matrix. This greatly reduces the size of training data.

Distance Function for Songs

Since each song is modeled as gaussian probability distribution, it is natural to compute distance between songs using Kullback-Liebler divergence[1]. KL divergence is defined for any two probability distributions. In the case where both distributions are gaussian of dimension d, it can be computed using the following closed form equation:

$$p(\mathrm{x}) = \mathcal{N}(\mathrm{x};\mu_p,\Sigma_p),\ q(\mathrm{x}) = \mathcal{N}(\mathrm{x};\mu_q,\Sigma_q)$$

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

One thing to note is that KL divergence is not symmetric, whereas a distance metric needs to be symmetric. To overcome this, we define our distance function to be:

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

Supervised Learning - K Nearest Neighbors

We use a simplistic approach for classification which gives surprisingly good results. During training, we compute the gaussian model for each song, and store these for all training samples for each genre. To guess the genre of a new song, we compute its distance, as defined above, with all samples in the training set, pick the k closest neighbors and assign it the predominant genre found amongst its nearest neighbors. The results are summarized in a later section.

Unsupervised Learning 'Kernelized' K-Means

We thought that by using the feature vectors and distance function as defined above could help discover interesting relationships between songs, artists, genres, etc. So we devised a modified form of the K-Means algorithm to cluster songs. Each iteration of K-Means assigns clusters to each of the data points and then computes the centroid for each cluster. However, in our case, each point is a probability distribution and there doesn't seem to be a clear way to define the centroid of a set of probability distributions. So while we do have a distance function between the data points, we don't have a good way to compute the centroids for each cluster.

We overcome this, by not computing the centroids at all. Instead we store the set of points for each cluster, and let the centroid vector be implicit. If S_i is the set of points in the cluster i, then we define the distance between the centroid of this cluster and any other point X_p as :

$$Distance(Centroid(S_i), X_p) = Mean(Distance(X, X_p)) where X \in S_i$$

Using this approach we only need to know the distance between all pairs of samples points. This can be passed to the K-Means functions in the form a matrix K, where $K_{ij} = <X_i, X_j>$.

We were only able to do a limited amount of testing with this approach due to time constraints. When we used songs from only two genres, classical and rock, the results looked promising. There were two clusters, one containing mainly classical and another mainly rock. However when we tested with songs from all four genres, the results weren't that good. There was one neat cluster with just classical music, while the other clusters had a mix of genres.

Visualization of Data in 3D with SVD

To help visualize the differences between the 4 genres, we used single value decomposition to project the high-dimensional data onto 3 dimensions for visualization. We did not expect to see any separation at these low dimensions, however there is a significantly visible separation between genres. Hip-hop and classical show the most obvious separation at 2D. Alternative and rock show much less separation but its still noticeable in 3D. Figure 2 shows these cases.

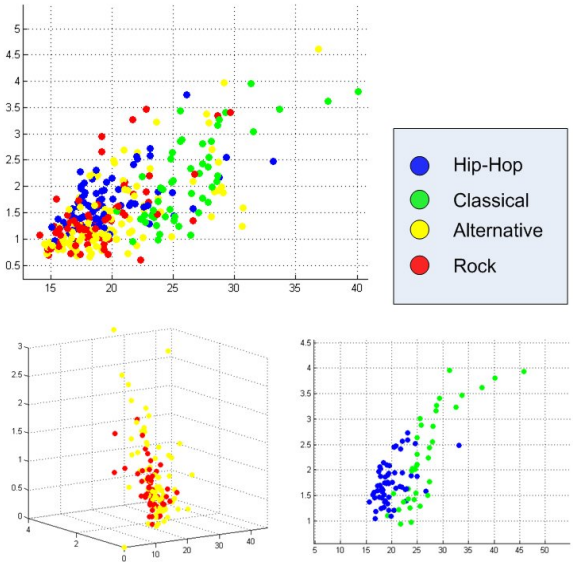

Figure 2: SVD projection of MFCC song data

Classification Results

Training/Testing Data Set

Testing and training data was collected from our music libraries by randomly choosing 476 songs spanning 4 genres. We wrote a script to extract the ID3 tags (metadata) from the files, and populate a master file with song filename, and genre. The script then converted the MP3 file to a WAV file, started MATLAB and ran a script to extract the MFCCs and store them in a file. In order to speed up training and testing, we created a simple test definition file format that identifies songs to process, their respective genres, and whether to treat that file as a training file or a testing file. This structure allows us to create customized train/test cases with subsets of the data we have and run a large number of experiments easily.

Evaluation Procedure

We evaluated the performance of our system in three different ways. The first approach is aimed at measuring performance in the average case whereas the other two simulate unfavorable scenarios for our system. Each train/test cycle generates a confusion matrix illustrating the algorithm's guesses and the actual song label for each song. These are shown in Figure 3.

Randomized K-Fold Cross Validation: 70% of the song data was randomly selected as training data and 30% was used to test the learned model. The absolute accuracy across a large number of runs ranged from 74% to 86%.

New Artist Effect: Since songs from the same artist tend to be similar, using KNN tends to work well given that some training examples from that artist exist. However, it is important to measure the accuracy of the model when presented with a new artist. Two artists, Queen and Pink Floyd, were completely removed from the training set and placed in the testing set. The result was 57% accurate. Also a new classical artist test was performed by moving all Christopher O'Reilly data to the testing set. This is unique since Christopher O'Reilly arranges classical versions of popular rock/ alternative songs. The available data include all his performances covering an alternative band Radiohead. The accuracy of testing on a new classical artist was 100%. Finally, to cover the new artist effect on a large scale, 143 songs from completely new artists were tested, with the result of 57% accuracy.

New Genre Effect: In addition to the 'new artist effect', the ability to introduce a new genre with limited data is important as well. To test the accuracy of the algorithm when presented with a new genre, all Hip-Hop songs were removed from the data set. 35 2Pac songs and 5 Wu-Tang Clan songs were chosen. Two tests were performed, one where the 2Pac songs were included in the training set and the Wu-Tang songs were tested, this resulted in 100% accuracy. The roles of the artists were then reversed and the 5 Wu-Tang songs were used in the training set, and the 2Pac songs were tested resultqing in 54% accuracy.

Overall, these results are extremely encouraging especially since many of the errors tend to occur between the rock and alternative categories. This was expected since the differences between rock and alternative is extremely vague.

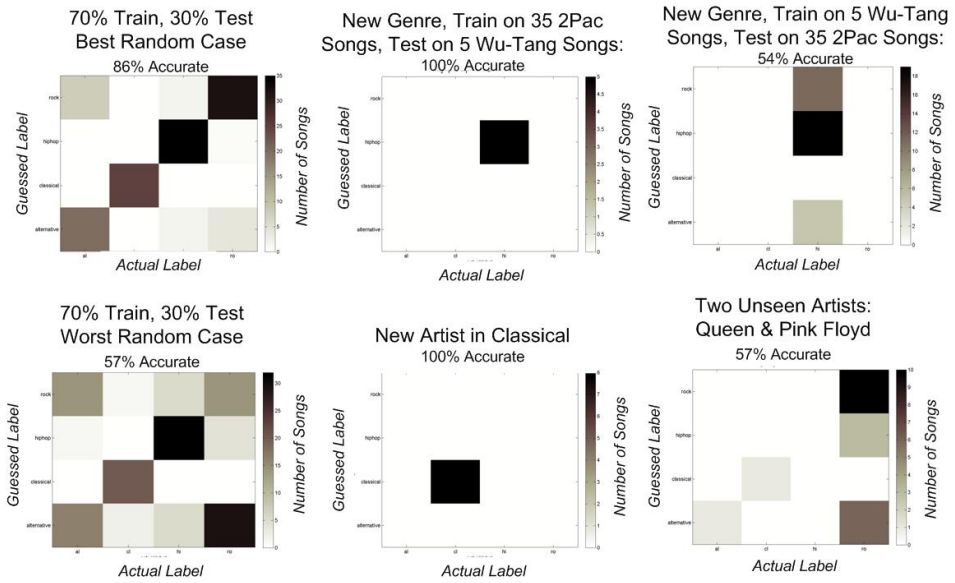

Figure 3: Confusion matrices for various tests

Lessons Learned & Future Work

It turns out that music genres are very subjective. Individual tastes, the artist's previous work, along with the song time period can drive genre classification. Possibly a more successful approach would be to create an entirely new classification language that would identify and classify each song based on timbre. The work here shows a reasonably successful algorithm, that can separate music based on timbre. Some work has already be done in analyzing the content of timbre using different auditory features[6] and perhaps this can be expanded to provide timbre classification for popular music.

It would be interesting if this work can be extended to include more genres of music. The ID3 standard specifies 126 different genres for music. Here we tackled only 4 genres of music due to time constraints. Since genre is such a subjective classifier, additional research in unsupervised learning possibly could yield some very interesting results illustrating some kind of timbral grouping of music.

References

1. Mandel and Ellis, *Song-level features and support vector machines for music classification*, http://www.ee.columbia.edu/~dpwe/pubs/ismir05-svm.pdf
2. Logan, *Mel Fequency Cepstral Coefficients for Music Modeling*, 2000, http://ismir2000.ismir.net/papers/logan_paper.pdf
3. Xu et al. , *Support Vector Machine Learning for Music Discrimination*, http://www.springerlink.com/content/4n32wf0pxfxubt4p/fulltext.pdf
4. Laurier and Herrera, *Audio Music Mood Classification Using Support Vector Machine*, http://www.mtg.upf.edu/files/publications/b6c067-ISMIR-MIREX-2007-Laurier-Herrera.pdf
5. Stanley, Auditory Toolbox v2, http://cobweb.ecn.purdue.edu/~malcolm/interval/1998-010/
6. De Poli and Prandoni, *Sonological Models for Timbre Characterization*, http://lcavwww.epfl.ch/~prandoni/documents/timbre2.pdf