Music Genre Classification

John Cast, Chris Schulze, Ali Fauci
CS229 Autumn 2013-2014

Introduction

Music genres allow the categorization of music into broad groups of related songs for use in applications such as song recommender systems based on genre. Many existing songs are unlabeled, and new songs are constantly being released. While hand-labeling them all would be too tedious, classifying them by genre can fit them into categories for use in recommendation. Also, even if the artist or producer labels a song as a specific genre, finding the songs that are most similar in terms of frequency structure rather than subjective labeling might provide recommendation information that the user would enjoy more.

In this project, we attempt to classify songs into a set of genre classes using Mel-frequency cepstral coefficients (MFCCs) from the songs. Because of the high dimensionality that results from using MFCCs, our main focus was to find appropriate methods that could 1) best classify data with such a high dimensionality and 2) reduce this high dimensionality and still retain the meaningful components of the data. We use both unsupervised and supervised algorithms, to cluster songs by genre and predict the genre of a given song, respectively.

Data Set and Features

Data Set: We used the Marsyas 1000 song data set [1], an openly distributed and widely used data set for song analysis. It consists of ten genres each with 100 songs, each 30 seconds in length. Each song is a 22050 Hz Mono 16-bit audio file.

Features: Our features consisted of the MFCCs, which are commonly used in music analysis. The human ear's response is approximately linear below 1kHz and logarithmic above 1kHz. Thus, the Mel scale provides a useful metric to discern the human ear's response to the Hz scale.

Feature Retrieval

We analyzed the data set for 1 and 15 second intervals taken from the middle of each song and segmented them into 20ms chunks, which helps in analyzing statistically stationary segments to build up an effective fingerprint for a song. We smooth the edges of these chunks with a hanning window and take the periodogram, which gives us their power spectral components. We multiply by overlapping triangles spaced according to the Mel scale and take the log, which tells us effectively how much energy (i.e. filter bank energy) is within a given frequency range because the human ear tends to be able to discriminate less accurately between closely spaced frequencies as frequency increases. We then take the discrete cosine transform (DCT) to decorrelate the originally overlapping filter bank energies, choose the lower 15 DCT coefficients (for each frame), and ignore the rest, which gives a slight performance gain since higher coefficients represent fast changes in filter bank energies and often degrade performance.

Data Conditioning

In addition to the 1sec and 15sec data parsing, we applied two PCA conditioning algorithms to the data sets in an attempt to reduce the dimensionality of each song while retaining the meaningful information.
PCA Basic: This representation interpreted the entire collection of songs as a single data set, normalized[1] the data, stacked the frames of all songs, and applied the standard PCA algorithm as described in class, and normalized again. This approach effectively reduced the number of MFCCs per frame from fifteen to ten.

[1] Normalization for a given data set (i.e. xi's) is subtracting off the mean of the entire data set in question and dividing by the component variances of the set.

PCA Map: This conditioning interpreted each song as an individual data set, normalized each data set, and found the top ten principal eigenvectors for a given song by the methods of PCA discussed in class. We then replaced the frames for each song with that song's eigenvectors and normalized. This perspective effectively viewed the eigenvectors of each song as a mapping that preserved information about the ordering and relative (not absolute) values of the MFCCs of a given song.

Data used

Each algorithm was run on the following data sets: original mels (before normalization), normalized mels, and PCA basic mels and PCA map eigenevectors. Our feature vector for each song was the mels for each frame concatenated together (or in the case of PCA map the eigenvectors concatenated together). For the original and PCA basic mels, we also ran the algorithms on the mean mel vectors over all frames of a song (for PCA map, the means lose their meaning, so results were very poor).

All of these feature vectors were retrieved from both 1 second and 15 second segments of the song. After experimenting with multiple combinations of the ten genres with varying success, we focused on a set of three and a set of four genres that gave us the best results: {classical, metal, pop}, and {classical, country, metal, pop}. This choice is consistent with Haggblade et al. [2], who asserted that accuracy of genre classification falls off dramatically for subsets of size greater than four.

Distances

Euclidean Distance: The default distance used for k-means and k-medoids by the Matlab packages, as well as kNN in our implementation is Euclidean, where the distance between two vectors is the norm-2 squared. MFCCs of each frame of a song were concatenated, normalized, and taken as the song's feature vector.

Kullback-Lieber (KL) Divergence: For kNN, we also used KL divergence as a distance metric between songs. 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 compute the following [7]:

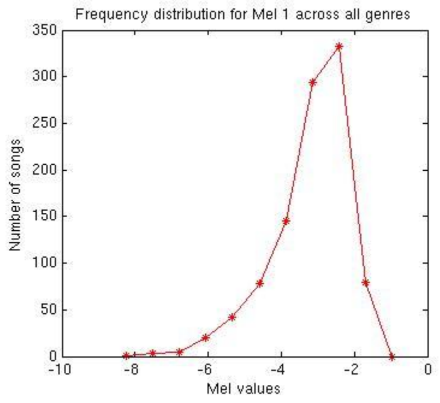
$$KL(p\,\|\,q) = \frac{1}{2}\left(tr(\Sigma_1^{-1}\Sigma_0) + (\mu_1 - \mu_0)^T \Sigma_1^{-1}(\mu_1 - \mu_0) - k - \ln\left(\frac{\det \Sigma_0}{\det \Sigma_1}\right)\right)$$

The KL divergence is not symmetric, so we took the following average:

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

While this divergence is not usually used as a distance metric for music classification, a prior project in CS229 [2] reports success with it. Under this representation, we used the mean mel vectors described above and the covariance matrix of the mels for each song.

Figure 1



Algorithms

Naive Bayes: Since our features are continuous-valued mel coefficients, we used the Gaussian Naive Bayes model, where the probability of a specific feature given a class is given by:

$$p(x_i = v \mid c) = \frac{1}{\sqrt{2\pi\sigma_c^2}} \exp\left(\frac{-(v-\mu_c)^2}{2\sigma_c^2}\right)$$

We are able to use this model because the features are normally distributed, which we determined by graphing the frequency distributions of the mean mel values for all songs (Figure 1 shows one such graph for the first mel).

We used the Matlab Statistics Toolbox NaiveBayes function [6] to fit the model and predict classes. Unlike with k-means, Naive Bayes trains a model and then tests on separate data to predict genres for the test songs. We took 80% of the data as our training set and left 20% for testing. The accuracy is the number of correctly assigned songs over the total number of songs tested.

k-Means: The k-means algorithm is an unsupervised learning algorithm that clusters data based on feature similarities. For our selected genres, we ran the Matlab *kmeans* algorithm in the Statistics Toolbox [3] on all of the songs for the selected genres to cluster them. We ranged the number of clusters (the "k" of k-means) from 1 to 10 times the number of genres (i.e. for 3 genres, we tried 3, 6, 9,..., 30 clusters).

In this algorithm, we wanted to look at whether, using our features, songs could be clustered correctly into their genres (i.e. whether each cluster represented a specific genre rather than a mix of genres). Instead of actually predicting genre based on a given song, this approach can be used to find songs that are similar to each other for applications such as music recommendation. To measure accuracy, we first assigned a label to each cluster by choosing the "majority vote" - which genre was the most common in that cluster. Our total accuracy for the model was given by the total number of songs labeled as the genre (label) of their cluster divided by the total number of songs used.

k-Medoids: K-medoids is a very similar algorithm to k-means, except that the cluster means are assigned not to the average of the points in the cluster at that time but to the closest actual data point to this average [4]. We used an open-source package in Matlab [5] to do the clustering itself, and our genre assignment and calculation of the total accuracy was the same for k-medoids as that for k-means.

k-Nearest Neighbor: The k-nearest neighbor (kNN) classifier is a non-parametric classifier that polls the "k" nearest training points to a given test point and classifies that test point based on the majority vote of these "k" nearest neighbors. kNN relies on a local prominence of data with a certain label, and works well with data that is described by a number of highly dense clusters, with each cluster representing a given label. While this algorithm is simple to implement, it is vulnerable to high dimensional inputs. It can be shown that increasing the dimensionality of the data corresponds to decreasing the effectiveness of kNN - with increasing dimensionality, the "k" nearest neighbors to a given test point become increasingly less local to that test point [4]. Therefore, we implemented the Kullback-Leibler divergence as described in the "Distances" section to represent distance between two songs for kNN in addition to the standard Euclidean distance. We took 90% of the data as our training set and left 10% for testing.

Results

Discussion: Figure 2 shows the peak performance of each algorithm on each of the datasets discussed above. Below we discuss notable results and overall trends for the four algorithms.

Naive Bayes: Accuracies for Naive Bayes were consistently among the highest across both the three and four genre testing and the 1 sec/song and 15 sec/song datasets, with above 90% for all of the three genre tests and most of the four genre ones as well. While results for the original mels vs. the PCA features were different, we found that normalization and PCA mapping did not affect results. For the original and normalized mels of three genres, we got 98% accuracy.

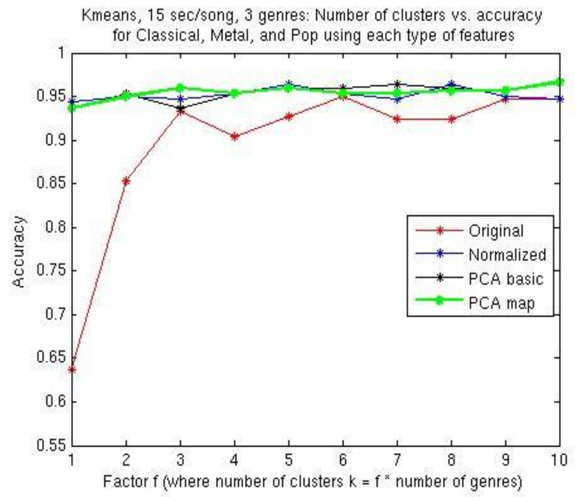
Figure 2

3 Genres	k-means (%)				k-medoids (%)				kNN (%)				Naive Bayes (%)	
classical, metal, pop	1 sec	k	15 sec	k	1 sec	k	15 sec	k	1 sec	k	15 sec	k	1 sec	15 sec
Original	93	24	95	18	87	18	87	27	97	6	86.67	9	98.33	98.33
Normalized	94	15	96	15	89	30	91	21	87	9	70	12	98.33	98.33
PCA basic	87	24	96	30	83	24	91	12	33.3	6	46.47	15	93.33	95
PCA map	86	27	96	30	82	30	91	6	93	7	73.3	10	93.33	95
Original means	93	30	96	18	91	27	96	30	100	6	100	6	98.33	96.67
PCA basic means	87	21	97	24	88	9	96	18	90	6	96.67	8	93.33	90

4 Genres	k-means (%)				k-medoids (%)				kNN (%)				Naive Bayes (%)	
classical, country, metal, pop	1 sec	k	15 sec	k	1 sec	k	15 sec	k	1 sec	k	15 sec	k	1 sec	15 sec
Original	82	40	85	36	78	36	80	32	75	9	72.5	11	90	93.75
Normalized	79	40	86	28	70	32	69	36	57.5	9	50	14	90	93.75
PCA basic	72	36	84	40	65	24	73	24	67.5	9	57.5	12	77.55	91.25
PCA map	73	40	84	40	66	32	72	36	25	6	30	12	77.55	91.25
Original means	81	40	86	24	82	32	86	32	92.5	8	92.5	11	88.75	88.75
PCA basic means	75	36	86	30			86	32	75	8	90	7	75	86.25

k-means: For k-means clustering, we reached 97% accuracy with the 15 second/song data using the three genre set. All accuracies (for each form of the mel vector) for this feature set were 95% or above. Four genres gave accuracies in the low 80% range for this data set as well. Overall, three genres always performed better than four. It is important to note that there is some variation between runs of k-means due to the random initialization of clusters and the possibility of local optima.

Figure 3



As we increased k, starting with the number of genres, we found that the accuracy increased (see Figure 3). However, after a factor of about 4 on the number of genres, the results plateaued and stayed generally level as we increase k. We believe this occurs because, while not all songs of a genre are very similar in structure, groups of songs within each genre may be. Taking notice of these effective "subgenres" for an overarching genre thus helps improve recommendation accuracy. We capped it at a factor of 10 since too many clusters would give too few songs per cluster to maintain significance.

On a related note, we tried spectral clustering (using the Meila and Shi algorithm and the algorithm detailed by Ng et al. [8]) on the data, but obtained poor results, with 55.6% accuracy using 4 genres and the mean mel vectors (using the original data set). Given our results for k-means and kNN, and the fact that k-means clusters data based on "compactness" while spectral clustering uses "connectivity", we hypothesize that our data is quite compact and can thus be effectively clustered using algorithms which rely on compactness.

k-medoids: The results for k-medoids followed the same patterns as for k-means, but were generally about 5% less accurate than the k-means results for each feature set.

kNN: Accuracy for this algorithm was seen to increase with "k" until about k = 9 and then subsequently decrease. This agrees well with expectation, as polling too few neighbors would not give a proper picture of the environment surrounding the test point. Polling too many neighbors, on the other hand, would increase the probability that a significant number of neighbors would not be local to the test point. Accuracy for kNN peaked (100% for 3 genre classification) when using the mean mel vectors (using the original data set). This agrees with expectation when considering dimensionality.

Future Work

PCA k-means: Another data conditioning scheme that could be tried is to first reduce the number of frames per song and then effectively reduce the number of MFCCs per frame. In this approach we would first normalize the data set, and then run the standard k-means clustering algorithm on the song for k centroids where k is less than the number of frames for a song. Then PCA would be applied to this resulting set of frames for a given song using only the first ten principal eigenvectors.
PCA Map: By inspection, the eigenvectors produced by PCA for the songs seem like a reasonable set of attributes to classify the songs. Our methods used unitary weighting of each eigenvector retrieved by PCA. However, this may be a naive approach. It would be interesting to try different weightings for each of the eigenvectors to better reflect their order of importance as determined by PCA. Possible weightings to consider are one over their respective ordering from PCA or respective eigenvalues.

KL and Jensen-Shannon Divergences on all algorithms: We wrote code to evaluate the KL divergence as well as the Jensen-Shannon (JS) divergence as defined in [9]. We also had custom implementations of k-means, k-medoids and Naïve Bayes using the KL divergence as well as k-means using JS divergence. However, due to poor results and problems with invertibility issues surrounding clustering of covariance matrices when using these divergences as distance metrics, we abandoned our custom implementations of these algorithms and hence abandoned using the KL and JS divergences for these algorithms. Further investigation should be performed to find the root causes of these issues. Our attempts to use these divergences showed that there was little difference in using the KL or JS divergences but further work should be performed to provide a better analysis.

Features: More analysis should be performed using other features coupled with MFCCs. For example, sources such as [10], suggest that the use of delta and acceleration coefficients in conjunction with MFCCs improves classification accuracies.

References

[1] *Marsyas.* “Data Sets.” Web 5 Nov 2013. http://marsysas.info/download/data_sets.
[2] *Haggblade, M., Hong, Y., Kao, K..* “Music Genre Classification”. CS229 Course Website. Web 5 Nov 2013. http://cs229.stanford.edu/proj2011/HaggbladeHongKao-MusicGenreClassification.pdf.
[3] “K-means clustering”. Matlab R2013a Statistics Toolbox. Web 1 Dec 2013. http://www.mathworks.com/help/stats/kmeans.html.
[4] *Murphy, Kevin. Machine Learning: A Probabilistic Perspective.* Cambridge: MIT Press. 2012. Print
[5] Chen, Mo. “K-medoids.” Matlab File Exchange. 30 Sept 2010. Web 1 Dec 2013. http://www.mathworks.com/matlabcentral/fileexchange/28898-k-medoids.
[6] “Naive Bayes”. Matlab 2013a Statistics Toolbox. Web 1 Dec 2013. http://www.mathworks.com/help/stats/naive-bayes-classification-1.html.
[7] *Wikipedia.* “Kullback-Leiber divergence” http://en.wikipedia.org/wiki/Kullback–Leibler_divergence
[8] Ng, Andrew et al., “On Spectral Clustering: Analysis and Algorithm”. *NIPS*. 2002. http://ai.stanford.edu/~ang/papers/nips01-spectral.pdf
[9] *Wikipedia.* “Jensen-Shannon divergence” http://en.wikipedia.org/wiki/Jensen–Shannon_divergence
[10] *Karpov, I..* “Hidden Markov Classification for Musical Genres” http://www.cs.utexas.edu/~ikarpov/Rice/comp540/finalreport.pdf