Classifying Parts Of Songs

Nick Colonnese

December 10,2010

1 Problem Statement

Audio visualizers are programs that take in music and produce images that correspond to the music in some pleasing way. In most songs there are several different parts: the verse, chorus, a solo, and so on. Ideally, a music visualizer would change its output to match the different parts of the song. While it is quite easy for humans to detect the difference between different parts of songs the task is not straightforward for computers. This project aims to use machine learning techniques to find if different parts of a song can be classified. The goal would be that before a song is played it could be processed by the algorithm and the transitions times from one part of a song to the next could be detected with the visualizer changing its display accordingly. A similar problem is approached using a different method in [2].

2 Data Collection

To get the music data into a form that is workable songs are converted from mp3 into wav format and then loaded into matlab. The songs are converted from mp3's to wav format with free online software such as 'mp3 converter simple.' The songs are converted to mono, so only one channel is considered. Once the data is in wav format it can be loaded into matlab at a specific sample rate. This is done using the built in function 'wavread.' The data representing a song takes the form of a long vector.

3 Approach

To classify different parts of a song the following approach was taken:

1. Sample from the song every couple of seconds the region around the sample time.
2. Extract features from the samples containing information about which part of the song the sample was taken.
3. Create points in which an element of a point is a feature.
4. Run k-means clustering on the points to find "clusters" for various values of k. These clusters represent different parts of the song.
5. Choose the value of k corresponding to the minimum of an objective function which attempts to discover the "true" value of k which should correspond to the number of parts in the song.
6. Given a sample from a song and an estimate for the number of different parts of a song classify the sample by determining which cluster it is closest to.

3.1 Feature Selection

For ease of implementation as well as visualization only two features were used. First, five second regions were sampled throughout the length of the song. This is displayed in Figure 1.

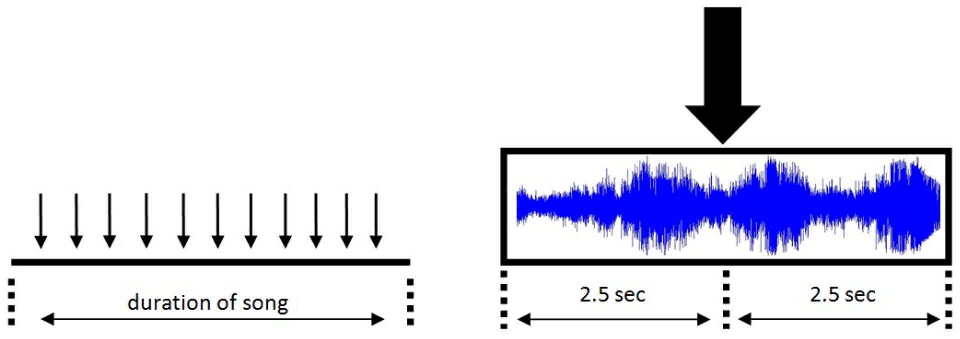

Figure 1: Sampling throughout the length of song

Then, for each sample the spectral content was estimated and then smoothed. This was done with the pseduospectrum function in matlab. The first feature was the frequency at which there was the highest spectral content. The second feature was the standard deviation of the spectrum a fixed amount around the maximum. A similar method for gathering features is in [3]. The feature selection method is displayed in Figure 2.

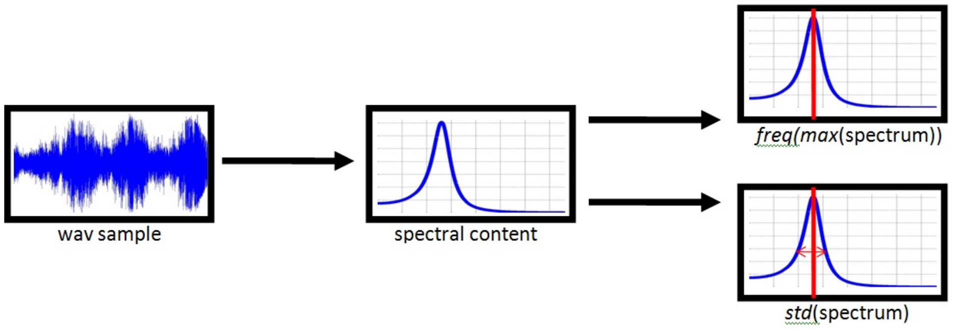

Figure 2: Feature extraction

3.2 Choosing the Number of Clusters

To use k-means clustering to classify different parts of a song the number of clusters k is required. Different options were explored to deal with this. One quick and dirty method was to try to find k by finding a balance between the number of clusters and the average variance of the clusters. This was done by setting k equal to the arg min of the following function:

$$\arg\min_k \left[\frac{\sum_i^k var(cluster^i)}{k} + C * k\right]$$

Where *var* represents the variance and C is a set constant. This method did not work usually because it is extremely dependent on the constant C. A better solution was seen in [1] in which many clusters are tried and their fit is judged with bayesian information scoring, but is complex. For this research k was set by hand. More is said on this choice in the conclusion.

4 Results

The algorithm was tested on several songs. The classification error was in the range of **20-40 percent** for real songs. Figure 3 contains scatter-plots of the features found by the algorithm for two songs. The colors correspond to different parts of the song. The x-axis of the points corresponds to the the frequency with the highest spectral content, while the the y-axis corresponds to the standard deviation.
The plot on the left is of a test song that consists of only three distinct parts, but these parts are interchanged and repeated many times. The plot of the right is of a real song called "3's and 7's" by Queens of the Stone Age.

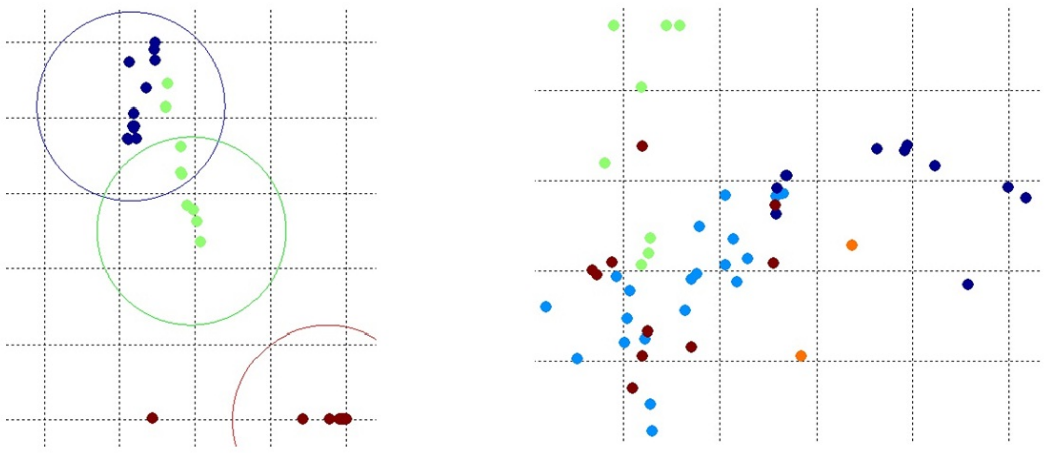

Figure 3: Test song (Left) and 3's and 7's (right) scatter-plot of features

The scatter-plot of the features for the test song (left) are separated fairly well into clusters. Circles representing the centroid's locations are shown with circles. However, for the real song (right) the data does note separate as well. The scatter-plot on the right for "3's and 7's" was typical of other real songs put through the algorithm: some parts of the song could be classified well, but other parts were usually too similar to be differentiated using this method.

5 Conclusion and Future Work

This algorithm works to a certain degree, but could be improved greatly if instead it was recast using EM instead of k-means clustering. It would be more appropriate because EM would allow for probabilities of a certain sample, not "hard" classifications like k-means. It would also allow for non-circular boundaries, and perhaps most importantly would obviate the need to find the k in k-means.

In terms of the audio visualizer application this algorithm sometimes still works well. The algorithm classifies different sounding samples of songs well, which may be more appropriate for knowing when to change an audio visualizer than classic parts of song classification. Also, time was not a feature, and if time was included the algorithm could probably do much better in this respect.

The algorithm was developed in matlab. For a real time application the algorithm would need to be ported to C or similar language.

References

[1] Andrew Moore Dan Pelleg. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 727–734, San Francisco, 2000. Morgan Kaufmann.

[2] Kiran Murthy Eleanor Crane, Sarah Houts. Musical hit detection. *CS 229 Final Project*, 2008.

[3] Greg Sell Song Hui Chon, Gauthum Mysore. Musical instrument detection. *Center for Computer Research in Music and Acoustics*, 2006.