TERM PROJECT

Music Genre Classification

Christian Ruiz

CSCI 6660: Artificial Intelligence

UNIVERSITY OF NEW HAVEN
Tagliatela College of Engineering

March 27, 2014

TO: Ted Markowitz

FROM: Christian Ruiz

RE: Term Project – Music Genre Classification

Introduction:

As a big music connoisseur, I enjoy both listening to music and creating it. However, there may be times when I may be unsure what the genre a song is. Most of the time I may know what general genre it should classify as, however do not know exactly what subgenre it would categorize under. Different genres of music are categorized using many different methods though many of them are complicated to realize programmatically.

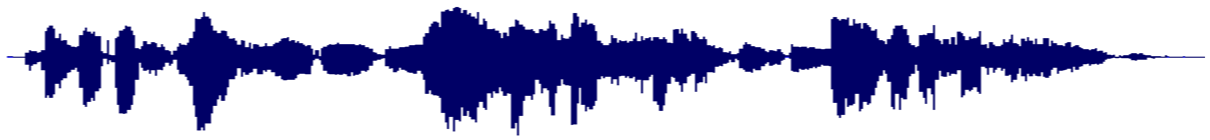
Some important elements of music genre are its tempo, melody, key and progression. These elements are difficult to analyze because of how audio is organized electronically. The program must look at the waveform of the audio signal and apply various filters to determine some of the key elements that may instantly change at any moment of the song.

When the song's genre is known, it may be simpler to determine its subgenre. This is because some of the major elements that classify a genre between the other main genres may be ignored to focus on an important element that can determine the song's subgenre: its tempo. Tempo is the speed of a song measured in beats per minute. The tempo is one of the bigger features of a song that can classify an Electronic song from being either Techno or Drum and Bass. Thus, the following program will explore a song's tempo in comparison to other songs in a database to determine its subgenre given its main genre has already been selected.

Programs like Spotify or Amazon would use data from their huge user database. This collaborative learning would suggest a song or an item because of what other user may have listened to or what they have bought in the past. This is similar from what this program will do, where it will have previously surveyed information (the training set) of average tempo ranges for each subgenre.

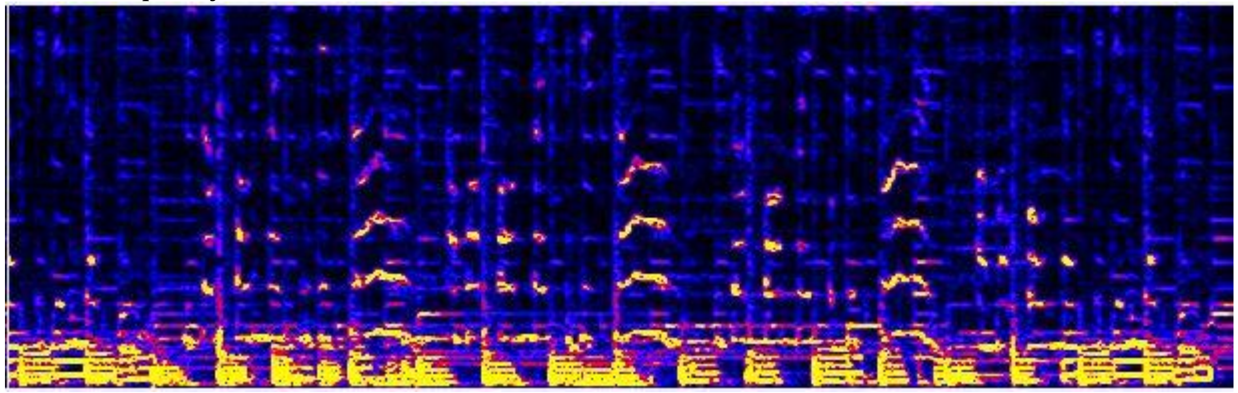
Analyzing Tempo:

There are multiple methods at determining an audio file's tempo. This can be implemented by finding the tempo of smaller slices of the song and putting them into subbands for further analysis. However, before this can be done, the audio file's waveform must be converted from the time domain into the frequency domain. This is done using a Fast Fourier Transform.



The above image displays an audio data's initial waveform in the time domain.

The Fast Fourier Transform (FFT) is an algorithm to compute the Discrete Fourier transform (DFT). This algorithm changes the audio's data from the time domain to the frequency domain. When the data is in the frequency domain, one can determine the intensity of the frequency at various frequency levels.



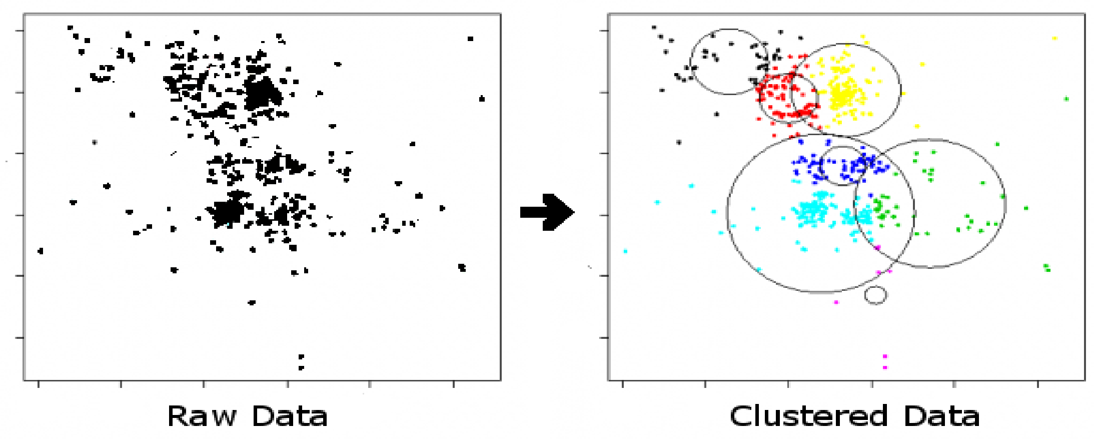
The above image is an audio data's representation in the frequency domain. The color defines the intensity of the frequency (from black to blue to red and to yellow). As one progresses up the chart the value of the frequency increases.

Various patterns can be spotted out of the data in the frequency domain. A method for determining the tempo of the audio file may be to see when the frequency intensity at some range reaches a peak value or when it is lower than a certain value to determine the rhythmic groove of the audio. Thus the tempo can be determined by the amount of times a frequency peaks and goes back down to its minimal value over an interval of time to come up with the beats per minute for that section of audio. These intervals may be different depending on the song, however can be stored in subbands to determine what is the most common tempo or the average tempo. Some autocorrection may be done to make the tempo results more accurate. Some methods include looking at the patterns near a section to determine if it was something consistent or a momentary change that should not be accounted for. Additionally, the algorithm may look at tempos around different sections of the song to see if that tempo fits the song as a whole or it should be disregarded.

There are two main issues with analyzing tempo. First, a song may have various tempos and may have sections where the tempo accelerates or decelerates where it can be close to impossible to determine the tempo. This may skew the results into something that is not ideal for subgenre categorization. Another problem of this method of generalization is the determined tempo may be a multiple off from the actual tempo. User supervision or different methods of analyzing the tempo would be wise for creating a good system based on tempo. These issues are minimal, where it happens for small sections of the song's total life span, thus in general tempo continues to be a good way of determining the song's subgenre.

<u>Analyzing Subgenres:</u>

Tempo is a good starting point for determining a song's subgenre. Each subgenre tends to be in a specific tempo range compared to other subgenres in the same main genre. However, it would not be wise for this to be the only element that categorizes the song. A good method for determining the subgenre is clustering. Clustering is a machine learning concept that groups points together that relate to each other through an assigned label. A program can iterate a clustering algorithm where it will combine clusters together until a desired number of clusters has been reached.



The above charts shows how one can cluster data points together from raw data.

Clustering usually has an unlabeled input put onto the feature vector and this point tries to determine what its label should be by computing a distance between itself and other points or clusters. Thus, as the database grows, the potential for determining the subgenre accurately grows. Each feature of the feature vector acts as a different dimension, thus if one wants to look at n features to determine where a data point should go, there will be n dimensions.

The feature vector may be filled will all unlabeled data, however it can be fed some training data to set the clustering to a good initial starting point. The tempo ranges for the subgenres prove to be a good training data set, where the tempo ranges will be plotted on the feature vector with a given label corresponding to its subgenre. The feature vector for the subgenres will be composed of two dimensions, one for tempo determined by analyzing the FFT data for frequency peaks and one for tempo that has been autocorrected by comparing the tempo data in relation with itself at different points of the song. Once a new song is added to the database to be analyzed, it will look at clusters near it and join it.

There are multiple methods to determining the distance of the nearest clusters such as the nearest data points, the nearest cluster determined by its nearest data point or centroid, the average distance between the data points within a cluster to a different cluster and many other methods. Each method have their benefits and drawbacks, and require much testing on different datasets to determine the best method. K-Nearest Neighbors is used to cluster the subgenres.

K-Nearest Neighbors looks at the k closest data points to the data point in question. The distance between the data point and the k closest data points determine the weight of their subgenre label on the data point that is being analyzed. As the database grows with new data points, the classification becomes more accurate, where there will be more points closer to each other that can merge into clusters and determine alter the initial subgenre clusters (the training set).

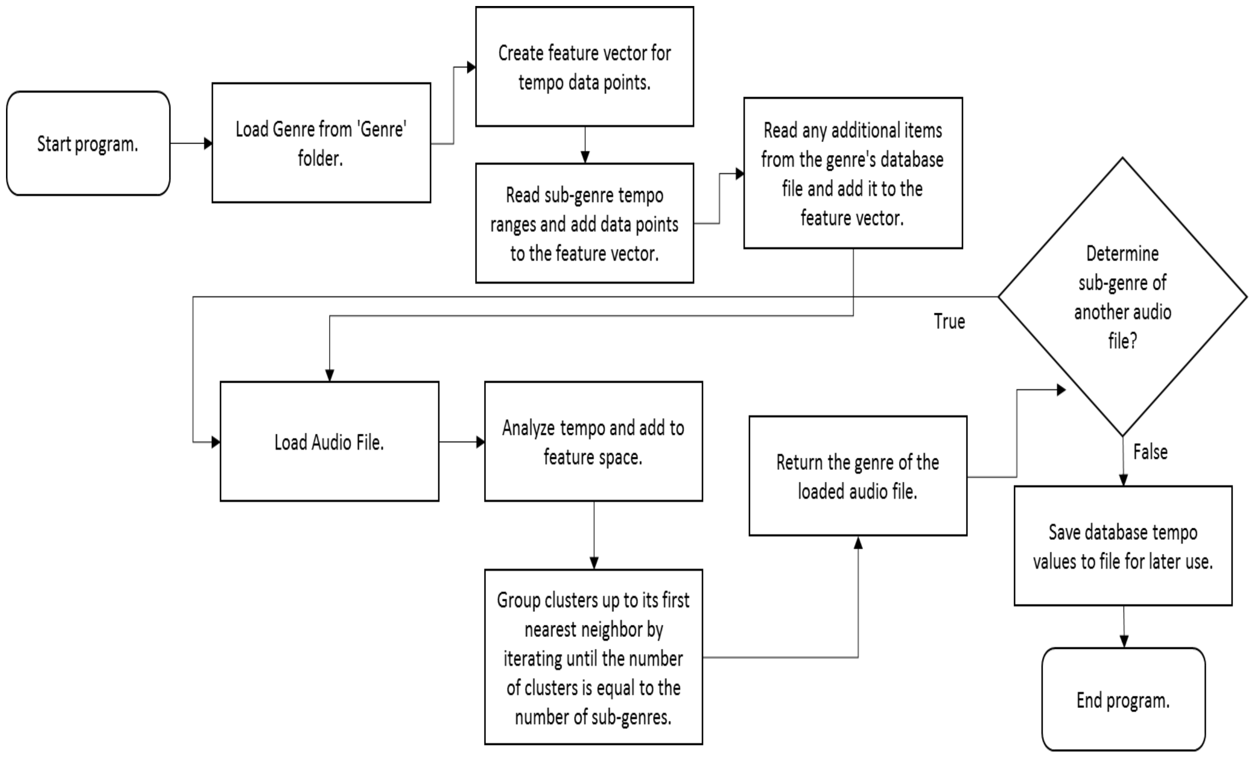
Libraries:

There is one main library that will be used in this program, the libZPlay library. libZPlay is a multimedia library for handling audio files. It can analyze, alter, and play audio files. The main function that will be used is zplay_DetectBPM that takes two parameters: the handler that points to the audio file and the tempo detection method. The handle is created using the CreateZPlay method where a ZPlay object is initialized and this object can open the file path of the audio file to be analyzed. The detection method is a parameter that selects weather the function will find the tempo using the FFT data's peaks or through an autocorrection method.

A smaller library named dirent will be used to display all the items in a specific directory. This is used to determine what main genre the program will be focusing on. The opendir, readdir and closedir functions is used to determine what is in a directory. opendir initializes a DIR object by passing the file path of the directory. A dirent structure is also created to load all information about the directory. readdir is called repeatedly to get the information about an item in the directory, where we are interested in the name of the text files in the 'Genres' folder. After all the items have been read and processed, the directory must be closed using the closedir function.

Implementation:

To put it simply, the program is a command prompt program that provides the user an interface where they navigate using numerical inputs and audio files are selected using an open dialog box. The program's execution is described by the flow chart below.


Start program.
Load Genre from 'Genre' folder.
Create feature vector for tempo data points.
Read sub-genre tempo ranges and add data points to the feature vector.
Read any additional items from the genre's database file and add it to the feature vector.
Determine sub-genre of another audio file?
True
False
Load Audio File.
Analyze tempo and add to feature space.
Group clusters up to its first nearest neighbor by iterating until the number of clusters is equal to the number of sub-genres.
Return the genre of the loaded audio file.
Save database tempo values to file for later use.
End program.

First, the program starts by asking the user to choose a genre. This is done in the loadGenre() function. The dirent library is used to enumerate the files in the ‘Genres’ folder, where each file indicates a major genre category. Each genre file has a subgenre per line where on the line the format is “Subgenre name, minimum tempo, maximum tempo”. Once the genre is selected, the program must read the file for all the subgenres and its tempo ranges. These subgenres are stored as Genre objects and added to a vector of Genres.

Next, the feature vector will be created using the createGenreFeatureVector() function. The feature vector will be a two dimensional array where each element is a vector of integers. The integer represents what cluster an audio file or training point belongs to. When there are the minimum number of clusters (before database items are added or after all the items in the database has been labeled) the cluster number – 1 corresponds to the subgenre in the subgenre vector. The subgenre’s range of tempos is added to the feature vector, for example if the range is 120-140 the elements [120][120] to [140][140] are filled with the value of the subgenre and since it is a two dimensional tempo feature vector it will create a box that translates to a nice circular cluster. The database items are added iteratively after they are read from a database file and appended to the correct location with a cluster number equal to the number of clusters (including itself). The database file is specific to each genre and where each line corresponds to an audio file’s tempo with the format “peak_tempo AC_tempo”. The insertion process is quick because each array slot corresponds to a specific tempo that increases linearly by one.

After the feature vector is initially created, the user is now allowed to select an audio file. An open dialog box will come up so that the user can easily navigate their computer for the desired audio file to be analyzed and added to the feature vector. This is done in the loadAudio() function. The un4seen BASS audio library presented the implementation for opening up the dialog box to select a file. This code is contained in the spectrum.c file under the c examples the BASS library provides.

Once the desired audio file has been selected and loaded, its tempo can be analyzed using the FFT methods as described previously. This is done using the libZPlay library in the analyzeAudio() function. The two tempos using the peaks and autocorrection methods are added to an AudioData object that contain variables for the two tempos, a string for the later determined genre, a subscript for their location on the feature vector, and other information the user may like to know such as the song name and the song length. The location of the AudioData object can be determined by looking up the feature vector with the subscripts [peakTempo][acTemp][location].

Next, the clustering can commence in the analyzeGenre() function. It will iterate through the database to cluster up the unlabeled clusters with other database items and the subgenre tempo ranges. The item that is currently being processed will look above, below, to the right and to the left of its location on the feature vector. It will continuously do this until the first closest cluster is found. It will look around its location at a specified radius where said radius would increase if as it tries to find a corresponding cluster. This will continue to iterate until the number of clusters is equal to the number of subgenres.

After the clustering process completes, the audio file that the user initially selected should have a cluster assigned to it, thus it can determine what subgenre it is. The cluster number – 1 can be passed as a subscript in the subgenres vector to return the audio file's subgenre.

The user has the option of loading another file and doing the whole process again. When the user is done with the program and decides to quit, the program will save the database to the 'Database' folder in a text file where it can then terminate. The interface gives the user a menu where it can load a genre, load an audio file, view the audio file's information, and quit the program.

Conclusion:

This method of clustering seems to work fine for this general purpose. Many things can help improve the database such as having a much bigger database of tempo information. I am currently limited to the music files that I have on my computer and thus am not able to have a large about of input to create a big feature vector; however, this is implemented so that it can do so and would get better with time. Additionally, other features can be added to the feature vector to classify the subgenres more uniquely. When looking for close clusters, it would be better to look for clusters in all angles from the initial location instead of the simple up, down, right, or left. Some probability theory can be applied to determine which cluster to choose if clusters happen to be equidistant away from the cluster that is being processed.

<u>References:</u>

waveform_lotos.gif. N.d. Photograph. floomWeb. 27 Mar 2014. <http://www.floom.com/images/waveform_lotos.gif>.

kmeans. N.d. Photograph. webzeestWeb. 27 Mar 2014. <http://www.webzeest.com/articleimages/8613985ec4_1378814431139kmeans.png>.

Bernhardsson, Erik. "Collaborative Filtering at Spotify."*slideshare*. N.p., 03 AUGUST 2013. Web. 27 Mar 2014. <http://www.slideshare.net/erikbern/collaborative-filtering-at-spotify-16182818>.

"Fast Fourier transform." <http://en.wikipedia.org/wiki/Fast_Fourier_transform>.

"k-nearest neighbors algorithm." <http://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm>.

Chu, Mei-Lan. "Automatic Music Genre Classification ." Diss. National Taiwan University, Web. <http://djj.ee.ntu.edu.tw/music_genreclass.pdf >.

"dirent.h - format of directory entries." n.pag. *The Open Group*. Web. 27 Mar 2014. <http://pubs.opengroup.org/onlinepubs/009695399/basedefs/dirent.h.html>.

“BASS audio library” < http://www.un4seen.com/>

“dirent.h download directory” < http://softagalleria.net/download/dirent/?C=M;O=D>

“libZPlay library” < http://libzplay.sourceforge.net/WELCOME.html>