# History of audio classification

[History of digital recording]

Even though digital audio became available in 1938 as telephone technology, it wasn’t until the 60s that mankind was able to record digital audio and store it in a computer.

Digital audio became possible after Harry Nyquist and Claude Shannon discovered what was known as Nyquist-Shannon Sampling Theorem, which was also discovered by E. T. Whittaker, Vladimir Kotelnikov and others whose name hasn’t been catalogued.

This theorem was, and still is, used to convert an analog signal (continuous) into a digital signal (discrete), dividing the analog signal into smaller pieces called “samples” and analysing every sample to get a value, that will represent all frequencies in the signal.

Years later, in the 1950s and 1960s, the technology to record digital audio kept improving, but it was still too expensive to be used for the great public.

It wasn’t until the 70s, that digital audio started to become mainstream, thanks to Thomas Stockham who, in 1976, built which is considered the first digital audio recorder: a 4-channel, 16-bit system that sampled at 50KHz.

[History of music analysis]

[Music classification nowadays]

With the appearance of Machine Learning, classification tasks have become easier,

# Set

For the realization of the project, we needed a large set of songs and genres to be able to train our algorithm in a proper way.

Initially, we wanted to use a relatively small amount of songs (100) of 4 different genres, all of them royalty free, taken from Free Music Archive. The problem was that the set we ended up with was too small to make the program work as intended.

We decided to change the set to an already made one, so we looked for data sets build for our purpose and ended up finding Marysas, a website in which we could find 1000 songs of 10 different genres (100 songs per genre), all of them 30 seconds long and with a similar set of properties (which will be explained later).

All the music in the data set is available for everyone and it can be used for investigation without any charge.

## Properties

All songs are “.au” files, which is a format used by the program Audacity.

To work with them, we need to know a few basics of digital audio, so I will explain what each one of the terms we will need when we extract the features of each song.

* Audio frame: Contains information in a given time.
* Sample rate: Number of samples taken from a continuous signal in order to produce a discrete signal.
* Channels: Number of streams in which the audio is sent.
* Frame size: Size of each frame. Sample rate \* # of channels.
* Frame rate: Number of frames per second. Frame size / s.

In our data set, all songs have the following properties:

* Sample rate: 22050Hz
* Channels: 1 (Mono)
* Frame rate: 22050 fps

To make the program able to work with other formats and songs, we will take all this information when we extract the features.

This is accomplished forcing the load function from librosa to take the Sample Rate as 22050 and converting the signal to Mono-channel.

# Audio Analysis

After investigation, we found out that most of the projects involving audio analysis, were using MFCC to extract features from the audio files.

MFCC (Mel Frequency Cepstral Coefficients) is usually used to extract features from human talk, but has been used lately for all kinds of sound.

## MFCC

MFCC were defined by Paul Mermelstein and S. Davis in 1980.

Although it was first developed to recognize monosyllabic words in spoken form, its characteristics make it useful for all kinds of sounds.

The algorithm works as follows:

1. Divide the signal in several same-sized intervals.
2. Take the Fourier transform of each interval.  
   This step, converts a wave into all the frequencies that made it, using the following formulas:
3. Convert the values obtained before to Mel Scale using the following formula:
4. Calculate the logarithm of every mel frequency.
5. Apply the discrete cosine transform to all mel logs.

Using this, we will end up with a matrix which size will be determined by the number of coefficients we want and the length of the audio sample.

## librosa

Considering the difficulty of this steps, we looked into ways of applying them in a simpler way, and we ended up finding that most of the projects involving MFCC use a Python library named “librosa”.

This library gives us the majority of audio analysis tasks already built in, so we only need to tweak the parameters we need to get the information we need out of every song.

The functions we will use for our project are the following:

* librosa.load(): This function loads the audio file, modifying the properties of the file we need to have all files following the same standards.  
  The most important parameters we need are:
  + sr: changes the sample rate
  + mono: converts the file to mono-channel
  + duration: crops the song into a smaller length.  
    The size of the matrix depends on the length of the file, so we need to make all songs last the same to work with them.
* librosa.feature.mfcc(): calculates the MFCC of the audio file we have loaded.  
  The function automatically tweaks all the parameters it needs to make a small enough matrix, but without losing huge amounts of information.  
  In this case, each interval is about 0.02 seconds long.

# Classification

Now that we know how to extract characteristics from the audio files, we can start to classify them.

The problem is that the amount of information we have using “librosa” is too big in comparison with the amount of songs we are using, so we have to find other methods that reduce the size of the data, but without losing information.

In order to do that, we will be using two different methods: reducing the size of the vector using PCA, and creating a histogram with all those values.

For each method, we will be explaining its procedure, as well as the results.

## Raw vector + PCA [nombre temporal]

As we will be doing in both methods, the first we are going to do is extract the MFCC of all the 1000 songs we have in our dataset.

This is done using the librosa function we described before, which gives us a matrix with as many rows as features we want to get and 1292 columns.

Before we store it, we will remove the first column, because it gives no meaningful information. Once we have remove it, we transform the matrix into a 1-dimensional array.

After that, we store that array in an array of tuples, where we will have the array and the genre, which will be an integer.

### PCA

Once we have the array of tuples, we have to reduce the size of each MFCC array.

To do that, we will be using a function given by sklearn library, which makes it automatically.

## Histogram

# Bibliography

<http://marsyasweb.appspot.com/download/data_sets/>

<http://freemusicarchive.org/>

<https://stackoverflow.com/>

<https://en.wikipedia.org/>

<https://github.com/librosa/librosa/issues/219/>