**Lab session 6**

Audio analysis and feature extraction

# Introduction

Audio analysis has a wide range of applications in different fields, like entertainment (audio editing, identification), biology (monitor endangered species through audio ID), health care (early detection of diseases like Parkinsons), psychology (audio emotion recognition), and many more.

Audio classification systems often follow a similar structure to image classification ones:

1. Data gathering
2. Data preprocessing
3. Feature extraction
4. Classification

Therefore, a classification model needs to be trained using certain features that represent the samples (in this case: audio signals).

This lab exercise will focus on the third step of the audio classification system, i.e., feature extraction. Incidentally, this step can be reused for other types of audio analysis tools, such as audio matching techniques, since it basically consists in representing the audio signal through different descriptors.

# 2. Project description

In this lab session, we will address a binary classification problem consisting of two classes: major (happy songs) and minor chords (sad songs). In other words, we will use recordings of major chords and minor chords to train a system that will later be expected to hear a sound of either a major or a minor chord and correctly classify it into their type. The audios for this lab session come from two instruments, guitar and piano.

## 2.1. Database description

The provided database consists of 859 recordings of major and minor chords played with a guitar or a piano. The audios have the same length (2 seconds) but different characteristics. The database contains 502 major chord recordings and 357 minor chord recordings, and has been obtained from Kaggle:

* *https://www.kaggle.com/datasets/deepcontractor/musical-instrument-chord-classification*

This dataset needs to be divided into a training set and a test set. The optimal percentage of data devoted to each process is left as a choice for the students to build their models.

## 2.2. Feature extraction

This is the key stage of the process, since it is where the students will perform all the implementations for the basic project requirements. The feature extraction stage is devoted to representing the audio signal using different techniques.

For this stage, let us remember that in order to analyze an audio signal, we often divide this signal into different frames. There are several window types that can be employed to extract the frames (e.g., hamming).

A basic template is provided where 2 basic audio features are extracted:

* Average value of the entropy of the energy of the audio signal’s frames.
* Maximum value of the entropy of the energy of the audio signal’s frames.

Therefore, in this template, a Nx2 feature matrix is generated (where N is the number of samples that are fed to our feature extraction module). Additional features can be extracted to increase the size of the feature matrix. To this end, the features studied in class can serve as a reference for the students. Some examples of potential features to be extracted are listed below:

* Spectral entropy of the audio signal
* Zero-crossing rate
* Mel-frequency Cepstral Coefficients
* Spectral centroid of the audio signal
* Spectral spread of the audio signal
* Harmonic ratio
* Chromagram from a waveform (Chroma STFT)
* Constant-Q chromagram

The obtained features need to be then normalized and fed to the classification system to train the model.

## 2.3. Classification

This stage fits the model with the training data using a Support Vector Machine with RBF kernel. No implementation is required in this stage.

Note that we have specified a positive class (label = 1) for the class ‘major’. However, this decision is completely arbitrary, since this binary classification problem faces two antagonic classes with equal weight. In other words, it differs from a problem where “something to detect” is clear and unambiguous (e.g., ‘disease’ vs ‘no disease’).

After the model is fitted, we use our test data to evaluate its performance.

## 2.4. Evaluation

The test data is fed to the system and we obtain a set of predicted scores as an output. These scores represent the probability of each sample belonging to the positive class (the probability of the audio source to be a major chord).

In order to evaluate the system without establishing a point of operation (i.e., a threshold for the scores that would yield hard binary labels), we make use of the Receiver Operating Characteristic (ROC) curve. In order to have a numeric representation of this performance, we observe the area contained under this curve as a reference (AUC).

Try running the template with the default features specified in the previous section to visualize this curve. This action should provide a figure similar to the following:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Note that the curve is quite close to the diagonal and, therefore, its AUC is nearing its minimum possible value (0.5). This can be considered as a baseline system to beat.

# 3. Experimentation and Result Analysis

In this section, we will conduct a series of experiments to understand the impact of individual features on the major-minor chord classification problem.

The executable code for these experiments is encapsulated in the *template.py* script. The *audio\_features.py* file contains the corresponding functions to extract each audio feature. It is essential to note that the **librosa library** is used in this practice. Make sure you have this package installed in your Python environment, otherwise you can install it in the terminal by executing the following command: pip install librosa.

## Experiment 1: Testing Zero Crossing Feature

Explore the impact of the zero-crossing feature on the major-minor chord classification. To do this, modify the extract\_features function of the *template.py* file, setting only the zero-crossing rate feature.

What classification results did you achieve? Why do you believe the zero-crossing feature may not contribute significantly to this problem? For what other problem do you think this feature could be useful?

|  |
| --- |
|  |

## Experiment 2: Implementing Chroma STFT Feature

Enhance the chord classification model by implementing the Chroma Short-Time Fourier Transform (STFT) feature extraction function in the *audio\_features.py* file. Utilize the resources available in the **[librosa](https://librosa.org/doc/latest/index.html)** library to aid in this implementation.

1. **Changing the frame length:**

Experiment with different frame lengths (flen) to discern the influence on the chord classification model's performance. Specifically, explore frame lengths of 512 and 2048.

How do the classification results change with varying frame lengths during the experimentation? What hypotheses can you formulate to explain the observed variations in performance with different frame lengths?

|  |
| --- |
|  |

1. **Analyse the output:**

Examine the output shape of the Chroma STFT. What do you think the value of the dimensions represents?

|  |
| --- |
|  |

## Ultimate Goal: Maximizing AUC in Major-Minor Chord Classification

The overarching objective of this practical session is to attain the highest possible AUC (Area Under the Curve) in major-minor chord classification.

In the *audio\_features.py* file, different functions have been provided to extract features from audio signals. The student should implement the functions to extract the missing features listed in the practice, in [Section 2.2](#_6xdog9na9x2o). Since all the features listed in this practice may be subject to evaluation in the lab test.

A clear grasp of each featured element is crucial for success in this practice. Different combinations of these features can be tried but, understanding these elements is not just beneficial but necessary for making informed decisions during the feature implementation process.

**Hint on how to obtain a better AUC:**

The first we should do, is to analyse the classes we want to differentiate; major and minor chords, we should wonder: ¿What is a chord?

A chord is a group of three or more notes (pitches) played together to create a harmonious sound. Chords are often formed by stacking notes on top of each other, and the specific combination of these notes determines the type of the chord.

The basic components of a chord are:

* Root: The foundational note upon which the chord is built. The other notes in the chord are named in relation to the root.
* Third: The interval (distance) between the root and the third determines whether the chord is major or minor. If the third is a major third interval (which means that there is a distance of 4 pitches between the root and the third), the chord is major; if it's a minor third (which means that there is a distance of 3 pitches between the root and the third), the chord is minor.
* Fifth: The interval between the root and the fifth determines the stability and consonance of the chord.

In addition to these basic components, chords can be extended with more notes, and alterations can be made to create various chord qualities, such as seventh chords, ninth chords, suspended chords, and more.

**Important Note:**

The key to achieving optimal AUC will not be disclosed, and no implementation response will be provided to the students. Instead, students are tasked with independently achieving a commendable metric. A reference point for a robust model is considered to be a value exceeding 0.8.