# Multimédia TP2

## Multimédia Information Retrieval

**aula-02 - 06/04/2022**

## Objetivos da aula:

## Ex 2.1 e 2.2 - extração de features

**Observações:**

- Façam funções para cada operação
- Organizem os ficheiros em pastas com nomeclatura adequada que facilite o desenvolvimento
- No exercício 2.2, testem com poucas músicas antes de processarem a lista completa (900). Pode levar +- 1h30min para processar as 900. (tempo estimado do processamento em PC Intel Corei7-6700HQ CPU @ 2.60GHz)

2.1. Processar as features do ficheiro top100_features.csv.

- 2.1.1. Ler o ficheiro e criar um array numpy com as features disponibilizadas.
- 2.1.2. Normalizar as features no intervalo [0, 1].
- 2.1.3. Criar e gravar em ficheiro um array numpy com as features extraídas (linhas=músicas; colunas = valores das features).

In [66]:
```
#lista de módulos necessários
import librosa
import librosa.display
import sounddevice as sd
import warnings
import numpy as np
import matplotlib.pyplot as plt
import os
import scipy.stats as st
```

2.1.1 Leitura do ficheiro:

o ficheiro top100_features.csv contém:

- 1ª linha = column labels -> devemos remover
- 1ª coluna = "song id" string -> devemos remover
- última coluna = "quadrante" string -> devemos remover

In [67]:
```
fileName = './datasets/features/top100_features.csv'
top100 = np.genfromtxt(fileName, delimiter=',')
nl, nc = top100.shape
print("dim ficheiro top100_features.csv original = ", nl, "x", nc)
print()
print(top100)

top100 = top100[1:, 1:(nc-1)]
nl, nc = top100.shape
print()
print("dim top100 data = ", nl, "x", nc)
print()
print(top100)
```

```
dim ficheiro top100_features.csv original =  901 x 102

[[        nan         nan         nan ...         nan         nan
          nan]
 [        nan  4.6433e+00  9.2878e+00 ...  2.8810e-01  1.0444e+03
          nan]
 [        nan  1.5461e+00  2.9366e+01 ... -2.6923e-01  0.0000e+00
          nan]
 ...
 [        nan  2.2305e+00  3.4567e+01 ... -9.4595e-02  0.0000e+00
          nan]
 [        nan  3.7033e+00  3.0063e+01 ...  1.5820e-01  1.9645e+03
          nan]
 [        nan  4.3733e+00  2.8490e+01 ... -2.9947e-01  9.8686e+02
          nan]]

dim top100 data =  900 x 100

[[ 4.6433e+00  9.2878e+00  9.2722e-01 ...  5.7884e-01  2.8810e-01
   1.0444e+03]
 [ 1.5461e+00  2.9366e+01  9.8409e-01 ...  6.5528e-01 -2.6923e-01
   0.0000e+00]
 [ 2.1486e+00  4.0437e+01  1.0521e+00 ...  6.8354e-01 -9.5701e-02
   0.0000e+00]
 ...
 [ 2.2305e+00  3.4567e+01  1.6469e+00 ...  6.5526e-01 -9.4595e-02
   0.0000e+00]
 [ 3.7033e+00  3.0063e+01  1.9423e+00 ...  6.6045e-01  1.5820e-01
   1.9645e+03]
 [ 4.3733e+00  2.8490e+01  1.5797e+00 ...  5.9931e-01 -2.9947e-01
   9.8686e+02]]
```

### 2.1.2 Normalização

Cada coluna da matriz de features deve ser normalizada de forma independente. Ou seja, utilizando os valores máximo e mínimo de cada coluna.

In [68]:
```
top100_N = np.zeros(top100.shape)
nl, nc = top100_N.shape

for i in range(nc):
    vmax = top100[:, i].max()
    vmin = top100[:, i].min()
    top100_N[:, i] = (top100[:, i] - vmin)/(vmax - vmin)

print(top100_N)
```

```
[[0.40833396 0.55892695 0.08767653 ... 0.49339478 0.29677785 0.28512149]
 [0.17434797 0.81639588 0.09864405 ... 0.66779986 0.14344785 0.        ]
 [0.21986539 0.95836272 0.11175995 ... 0.73227772 0.19118833 0.        ]
 ...
 [0.22605274 0.8830899  0.22646862 ... 0.66775423 0.19149261 0.        ]
 [0.33731922 0.82533373 0.28343725 ... 0.6795957  0.26104038 0.53630904]
 [0.38793611 0.80516266 0.21350893 ... 0.54009902 0.13512837 0.26941305]]
```

### 2.1.3 Gravar a matriz de features num ficheiro

In [69]:
```
fileName = './datasets/avp_features/top100_features_normalized_data.csv'
np.savetxt(fileName, top100_N, fmt = "%lf", delimiter=',')

#checando
top100_N = np.genfromtxt(fileName, delimiter=',')
nl, nc = top100_N.shape
print("dim ficheiro top100_features_normalized_data.csv = ", nl, "x", nc)
print()
print(top100_N)
```

```
dim ficheiro top100_features_normalized_data.csv =  900 x 100

[[0.408334 0.558927 0.087677 ... 0.493395 0.296778 0.285121]
 [0.174348 0.816396 0.098644 ... 0.6678   0.143448 0.      ]
 [0.219865 0.958363 0.11176  ... 0.732278 0.191188 0.      ]
 ...
 [0.226053 0.88309  0.226469 ... 0.667754 0.191493 0.      ]
 [0.337319 0.825334 0.283437 ... 0.679596 0.26104  0.536309]
 [0.387936 0.805163 0.213509 ... 0.540099 0.135128 0.269413]]
```

2.2. Extrair features da framework librosa.

- 2.2.1. Para os 900 ficheiros da BD, extrair as seguintes features (sugestão: guardar todas as músicas na mesma pasta):
  - Features Espectrais: mfcc, spectral centroid, spectral bandwidth, spectral contrast, spectral flatness e spectral rolloff.
  - Features Temporais: F0, rms e zero crossing rate.
  - Outras features: tempo.
  - Utilize os parâmetros por omissão do librosa (sr = 22050 Hz, mono, window length = frame length = 92.88 ms e hop length = 23.22 ms).
  - Guarde as features num array numpy 2D, com número de linhas = número de músicas e número de colunas = número de feartures
- 2.2.2. Calcular as 7 estatísticas típicas sobre as features anteriores: média, desvio padrão, assimetria (skewness), curtose (kurtosis), mediana, máximo e mínimo. Para o efeito, utilizar a biblioteca scipy.stats (e.g., scipy.stats.skew).
- 2.2.3. Normalizar as features no intervalo [0, 1].
- 2.2.4. Criar e gravar em ficheiro o array numpy com as features extraídas.

2.2.1 Extração de audio features com a librosa

2.2.2 Cálculo das estatísticas

In [70]:
```
filesPath = './datasets/audio/all/'
files = os.listdir(filesPath)
files.sort()
numFiles = len(files)
print("num audio files = ", numFiles)
```

```
num audio files =  900
```

Cálculo do tamanho da matriz resultante:

número de linhas = número de ficheiros de audio

número de colunas = (número de valores de cada feature) x (número de estatísticas) + tempo

| mfcc | sp centroid | sp bandwidth | sp contrast | sp flatness | sp rolloff | F0 | rms | zcr | tempo |
|---|---|---|---|---|---|---|---|---|---|
| 13 | 1 | 1 | 7 | 1 | 1 | 1 | 1 | 1 | 1 |

numero de colunas = $7 \times (13 + 1 + 1 + 7 + 1 + 1 + 1 + 1 + 1) + 1 = 190$

Exemplo de utilização da librosa e scipy.stats

In [76]:
```
#feature extraction options
sampleRate = 22050
useMono = True
warnings.filterwarnings("ignore")
F0_minFreq = 20 #minimun audible frequency
F0_maxFreq = 11025 #nyquist/2
mfcc_dim = 13
spContrast_dim = 7 #(6+1) bands

#extração de features para o 1° file

fileName = filesPath + files[0]
inFile = librosa.load(fileName, sr=sampleRate, mono = useMono)[0]

mfcc = librosa.feature.mfcc(inFile, n_mfcc = mfcc_dim)
print("\n mfcc size = ", mfcc.shape, "\n")
print("\n mfcc coeficients: \n", mfcc)

#calcular as estatísticas para cada linha da mfcc e guardar numa matriz
nl, nc = mfcc.shape
mfcc_meanAndSkew = np.zeros((nl, 2))
for i in range(nl):
    mean = np.mean(mfcc[i, :])
    skew = st.skew(mfcc[i, :])
    mfcc_meanAndSkew[i, :] = np.array([mean, skew])


print("\n mfcc_meanAndSkew:\n", mfcc_meanAndSkew)

#formatar a matrix de linhas para colunas para poder usar 1 linha por música
mfcc_meanAndSkew = mfcc_meanAndSkew.flatten()
print("\n mfcc_meanAndSkew flattened shape = ", mfcc_meanAndSkew.shape)
print("\n mfcc_meanAndSkew flattened:\n", mfcc_meanAndSkew)
```

```
mfcc size =  (13, 1295)


mfcc coeficients:
[[-5.81123352e+02 -5.80946899e+02 -5.75401550e+02 ... -5.47280701e+02
 -5.71498962e+02 -5.81089844e+02]
[ 0.00000000e+00  2.27403581e-01  7.48309231e+00 ...  3.36044388e+01
  1.05559568e+01  4.30819876e-02]
[ 0.00000000e+00  1.64889589e-01  5.78880501e+00 ...  8.77727890e+00
  5.31558323e+00  3.09513584e-02]
...
[ 0.00000000e+00 -1.12017617e-01 -2.65580797e+00 ... -4.58360577e+00
 -3.17881680e+00 -1.92028321e-02]
[ 0.00000000e+00 -1.07866675e-02 -1.67927337e+00 ... -7.95033407e+00
 -4.34999800e+00  5.81501052e-04]
[ 0.00000000e+00  9.21395198e-02 -9.21839297e-01 ... -6.61009216e+00
 -2.75344086e+00  2.02601999e-02]]

mfcc_meanAndSkew:
[[-2.30354630e+02 -7.10278009e-01]
[ 1.00476913e+02 -5.92350300e-01]
[-3.22840118e+01 -4.76053778e-01]
[ 2.87400303e+01  7.85627418e-01]
[-9.85208797e+00 -1.38887354e+00]
[-1.56655493e+01 -5.88064407e-01]
[-5.18498659e+00  6.29570357e-01]
[-1.68022370e+00 -2.18031770e-03]
[-1.43120260e+01 -2.79983158e-01]
[-2.23230267e+00 -6.74988304e-02]
[-5.93310308e+00 -6.38440981e-02]
[ 6.49562550e+00  1.59027078e-01]
[-2.46414280e+00 -3.31705947e-01]]

mfcc_meanAndSkew flattened shape =  (26,)

mfcc_meanAndSkew flattened:
[-2.30354630e+02 -7.10278009e-01  1.00476913e+02 -5.92350300e-01
-3.22840118e+01 -4.76053778e-01  2.87400303e+01  7.85627418e-01
-9.85208797e+00 -1.38887354e+00 -1.56655493e+01 -5.88064407e-01
-5.18498659e+00  6.29570357e-01 -1.68022370e+00 -2.18031770e-03
-1.43120260e+01 -2.79983158e-01 -2.23230267e+00 -6.74988304e-02
-5.93310308e+00 -6.38440981e-02  6.49562550e+00  1.59027078e-01
-2.46414280e+00 -3.31705947e-01]
```

In [75]:
```
! jupyter nbconvert --to html tp2-pl-02.ipynb
```

```
[NbConvertApp] Converting notebook tp2-pl-02.ipynb to html
[NbConvertApp] Writing 298151 bytes to tp2-pl-02.html
```

In [ ]:
```
```