# Assignment 2
# EQ2341 Pattern Recognition and Machine Learning

**Names**
Oriol Closa and Clara Escorihuela Altaba

**Date**
January 23, 2022

## 1 Introduction

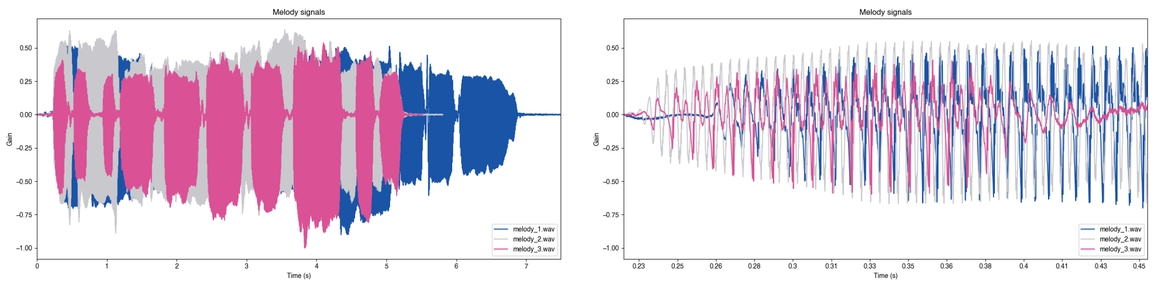
This project aims at developing a system capable of differentiating and recognizing humming melodies based on Hidden Markov Models (HMM), a statistical model which consists of a sequence of hidden states and observations. In this case, the main idea is to design an HMM for each melody, in order to identify them despite the voice, the instrument and the tone of the song.

The first step to proceed with the design of a pattern recognition mechanism is the feature extractor generator. This step is not part of the learning and training process, however it is crucial to have precise predicted results. A melody presents many characteristics, nevertheless only some of them are important for the training process: for that reason, it is crucial to understand which are the relevant features to take into account while generating our model, and create a suitable feature extractor generator on its purpose.

## 2 Data

The initial data for this assignment consists of three humming songs from the initial assignment files themselves. Two of these signals contain the same melody but in a different tone and rhythm, while the third one belongs to a completely different audio which apparently interprets *La Marseillaise*. Our HMM should be able to show a high level of similarity between the first two files, recognising them as the same song, while the third should be determined to be an entirely different audio from the first two due to some high differences.

Figure 1a presents the sound waves of the melodies. As we can see in figure 1b, the period and frequency of the wave changes depending on its tone, which makes it very easy to identify its initial and final time. Moreover, the amplitude of the wave can also give us an intuition of the intensity of the melody, when the gain increases the intensity also does and vice versa.

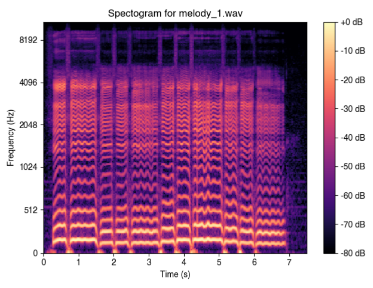


(a) All samples

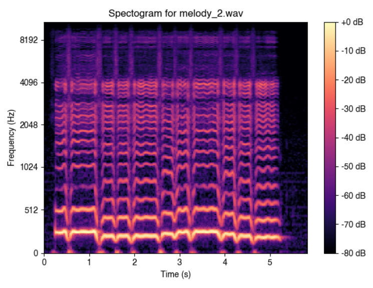
(b) Samples 500-1.000

Figure 1: Audio Signal for $melody_1$, $melody_2$ and $melody_3$

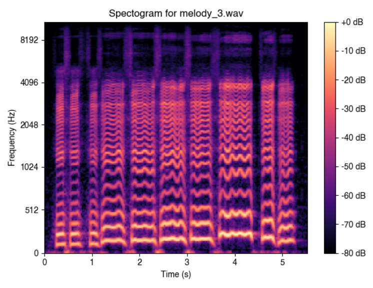
On the other hand, figure 2 exposes the spectogram of $melody_1$, $melody_2$ and $melody_3$. The spectogram represents the average frequency of a 60ms window in the sound signal, therefore it also allows us to identify the specific tone we are singing thanks to knowing the frequency. The window is moved by half its size creating as many as $\frac{duration_{melody_n}}{0{,}03}$ splits. Theoretically, the spectogram of the first and second audio should present different frequency values for each note, but should also contain a similar pattern. On the other hand, the third spectogram should be different in all senses picturing different shapes for the same timestep $t$.



(a) $melody_1$



(b) $melody_2$



(c) $melody_3$

Figure 2: Spectogram for $melody_1$, $melody_2$ and $melody_3$

We can also visualise the fundamental frequency of one of the melodies as seen in figure 3 along with our sampled frequency. We divided the spectrum into a few different octaves as explained in the following section where each of them has 12 different semitones. What we can see here is therefore the original signal along with the processed values that are produced from a saved octave and semitone. This means the accuracy of our processed values is much lower than the original sample allowing us to discretize the original frequency.

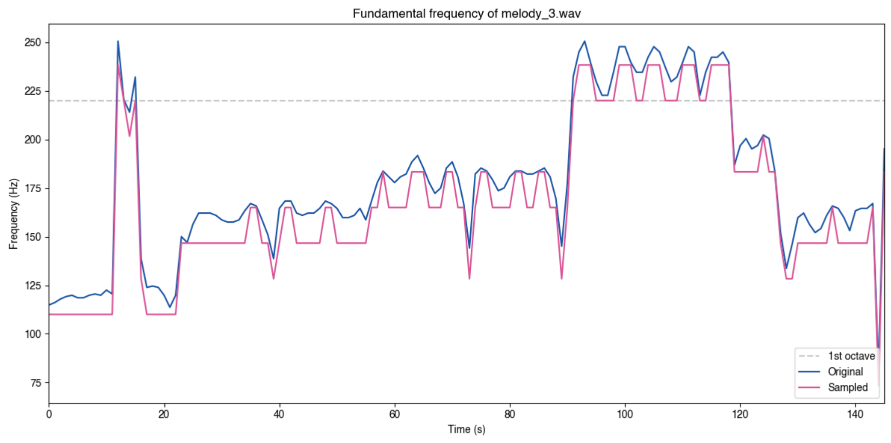


Figure 3: Fundamental frequencies for $melody_3$

## 3 Feature Extraction

As «Acoustic Feature Extraction from Music Songs to Predict Emotions Using Neural Networks» exposes the most relevant features in melodies are timbre, rhythm and dynamics[1], for that reason we propose the feature matrix $F$ as a list of $8$ features, which try to mimic those characteristics for a specific range in our input signal. By using moving windows as explained in the previous section we collect certain aspects of the values within the range and compute the following values.

1. Semitone ($st$): The smallest musical interval we consider within the 12-tone scale. This value ranges $[0, 11]$ and it is always the semitone within a specific octave which is defined in the next point. Thnat is, the semitone $5$ in octave $0$ and $2$ for example will have the same value here, the first could be interpreted as $5$ while the second as $2 \cdot 12 + 5 = 29$ but will be set as $5$ as well.

2. Octave ($o$): The interval between one musical pitch and the following doubling its frequency, or in other words, the interval between the harmonic $t$ and $t+1$ of the harmonic series. For our feature extractor function we consider the frequency ranges $[0, 220, 440, 880, 1.760, 3.520, 7.040, 14.080]$ to

define the octaves $[0, 6]$. Therefore a frequency of $376, 45$Hz would be considered to belong to the second octave, $1$ in our feature matrix.

3. Silence ($s$): This value determines if the segment is considered to be a silence or not. We define this as a silence if the fundamental frequency is a really high value or the intensity is really low, in our case we use $0, 025$ as the threshold.
4. Filtered silence ($sf$): This is a derived value from $s$ where less than 5-length silences have been filtered out. This means that if we have a temporary window where we determined a silence occurred such as in `[False, True, True, False, False]`, $sf$ will be `[False, False, False, False, False]` instead thus avoiding cases where we unintentionally misinterpreted a window as containing a pause. The length requirement of $5$ is completely arbitrary and will depend on the window length, in our initial case we determine the silence has to be at least 300ms as the window size is 60ms.
5. Intensity ($i$): This is related to the dynamics of the song. It returns a value ranged $[0, 1]$ which indicates the sound level relation between different windows. They are directly part of the output from the given `GetMusicalFeatures` function.
6. Tempo ($t$): This defines how fast a segment is performed by counting the number of different semitones and octaves in the corresponding $10$ adjacent windows. This $10$ is again an arbitrary number corresponding to 600ms according to our current window size. As we approximate the frequencies into their closest semitones we ensure they will not be all different between themselves. Therefore the possible value ranges here are $[1, k]$ where $k$ is the number of samples we have in 600ms.
7. Semitone difference ($dst$): This represents the amount of steps between semitones of consecutive segments including negative increments. This value also takes the change of octave into account thus its range is $[-84, 84]$. For example, if the semitone at timestep $t$ is $6$ with octave $0$ and the semitone at $t + 1$ is $10$ at octave $2$ then the value for $dst$ at $t$ will be $(12 \cdot 2 + 10) - (12 \cdot 0 + 6) = 28$.
8. Octave difference ($do$): This is the same as above but it only takes the difference in octaves ranging $[-7, 7]$ in our case.

Therefore, the output feature matrix $F$ ca be represented as following.

$$F = \begin{pmatrix} st_1 & st_2 & \cdots & st_n \\ o_1 & o_2 & \cdots & o_n \\ s_1 & s_2 & \cdots & s_n \\ sf_1 & sf_2 & \cdots & sf_n \\ i_1 & i_2 & \cdots & i_n \\ t_1 & t_2 & \cdots & t_n \\ dst_1 & dst_2 & \cdots & dst_n \\ do_1 & do_2 & \cdots & do_n \end{pmatrix}$$

A way to visualise the different values for a given audio can be seen in figure 4 where the first six features are included. The left axis includes both the semitone and octave as well as the tempo while the right axis displays the intensity of the signal. Silences are displayed as grey areas on top of the plotted data. Light grey areas correspond to original silences while the darker regions belong to filtered silences as well. It is really interesting to see how the very beginning and end are marked as belonging to a silence due to the intensity being really low. Moreover, we can see this pattern repeats every time a tone changes as the air output from the mouth stops briefly to allow changing the shape and tongue position to produce the next tone. Another interesting value is the octave which increases by three for each intermediary silence. This is due to the frequency being really high in those samples and probably due to the microphone picking low intensity noise with a high frequency as there is no humming being pronounced.

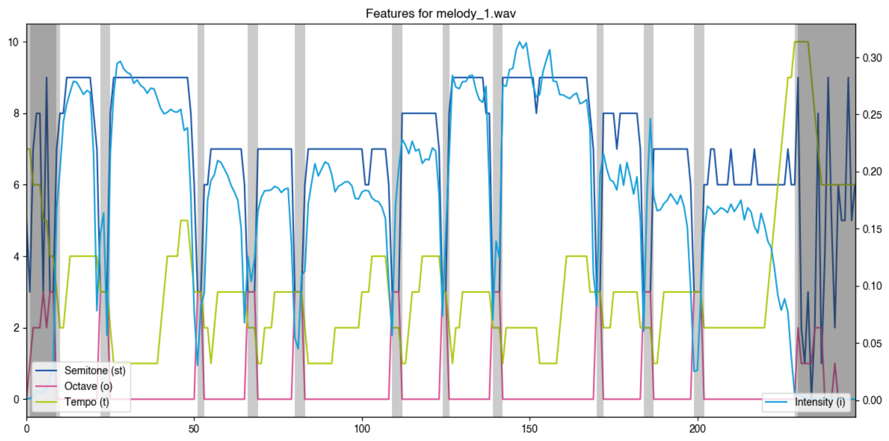


Figure 4: Extracted features for $melody_1$

## 4 Limitations

The following section details some possible issues our feature extractor could present. In the first case, this method is based on detecting the fundamental frequency of each note, and almost all the features are extracted from this value. This frequency is deduced from the yin algorithm, which presents difficulties to detect frequencies in polyphonic melodies. For that reason, if the melody contains two different notes in the same time the obtained frequency could be no as good, and the feature extractor would not work properly.

Another important issue would be the tempo of the songs: If two recordings contain the same melody but the tempo in one of them is much faster, this feature will be different for both of them. However, this problem would be solved if we normalised the tempo previously, by making all the songs to have the same length (to last the same amount of time).

Finally, if the melody goes much faster the difference in each note will not be as precise and we should decrease the window's length to avoid overlapping signals.

## References

1. NAWAZ, Rab; NISAR, Humaira; YAP, Vooi; TANG, Py. Acoustic Feature Extraction from Music Songs to Predict Emotions Using Neural Networks. In: 2018, pp. 166–170. Available from DOI: `10.1109/ICBAPS.2018.8527414`.