AUTHOR GUIDELINES FOR ICASSP 2018 PROCEEDINGS MANUSCRIPTS

Mattia Lecci, Federico Mason, Victor Cercos Llombart

UPC ETSETB - MET

ABSTRACT

In this report we try to perform chord recognition, meaning the extraction of the name of a musical chord from an audio file. After describing the techniques used, we show the performance obtained in two different scenarios: the first one is a simple single-chord single instrument kind of scenario, while the second one considers full songs.

Through the implementation of classical *Machine Learning* (ML) techniques and *Hidden Markov Models* (HMM) we illustrate the difficulties and show advantages of fairly complicated algorithms over the simple ones.

1. INTRODUCTION

A chord is defined as two or more notes sounding simultaneously .(1) This is normally achieved by playing the different notes that form the chor at the same time, in other cases like arpeggios and broken chords, the notes are played separated and successively. (FOTO??) The harmonic content of a music piece is defined by the chords and the progressions of them. This aspect is very important to understand and analyse tonal music.(2) This is an important applications in cover songs identification where the harmonic analysis is more robust. (3) The chord names vary depending on the number of notes that form them: "Triad","Seventh", "Ninth",... The most commonly used chords in music pieces are the "triads", in which this project focusses. This kind of chord is composed by three different notes. To avoid problems with the different octaves and their pitch, "chromas" are used instead of distinct notes.(4)
(Foto) The notes of a chord are selected according to the musical distance between pitches of simoultaneously sounding notes, the so called musical interval. The chords are commonly classified as "minor", "major", "dimished" and "augmented". This project focusses on minor and major chords. The name of the chord is driven from the root note of it, usally this note is the lowest one. Taking the triad as an example the second note is called third, because it's the third chroma from the root note on, so the last note is called fifth. In the minor triad only the 2 noe changes, to a minor third interval. Some chords present inversions, where the root note is not in the lowest position.

2. TECHNIQUES USED

2.1. Features Extraction

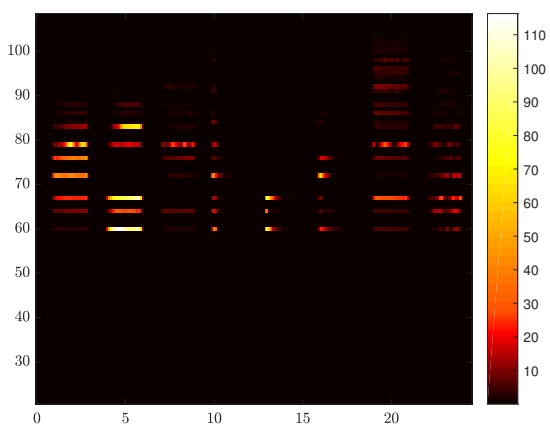
Feature extraction is performed using *Chroma Toolbox* [**?**], a MATLAB implementation of many different features under GPL license.
At first, the audio signal is decomposed into 88 frequency bands centered in the frequencies corresponding to the pitches of A0 up to C8. A multirate filter bank using elliptic filters is used and finally *Short-Time Mean-Square Power* (STMSP) is used to extract the useful information. In order to ignore the the different harmonicities given by different instruments, the energies referred to the same notes are pooled together resulting in 12-dimensional vectors.
The different features are then obtained operating in different ways on the extracted pitches. In this project we used three of them:

- CLP (*Chroma Features with Logarithmic Compression*): bands from different octaves are summed together, a logarithmic function is applied (to emulate our logarithmic perception of sound intensity) and then the vector is normalized to L_2-norm equal to 1.
- CENS (*Chroma Energy Normalized Statistics*): in order to account for dynamics, timbre, articulation, execution of note groups, and temporal micro-deviations, smart logarithmic quantization, temporal smoothing, downsampling and normalizations are performed.
- CRP (*Chroma DCT-Reduced log Pitch*): they include logarithmic compression, smoothing and timbre invariance by using a DCT-based technique.

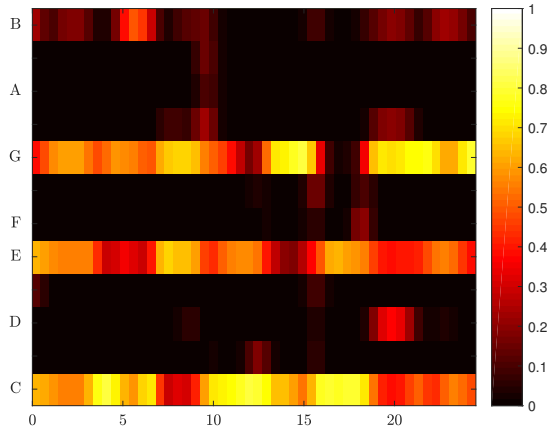
2.2. Template-based chord recognition

2.3. Gaussian Mixture Models (GMMs)

Gaussian Mixture Models are a statistic technique used to individuate subpopulations within an overall population. At the end of the process, each subpopulation is assigned to a Multivariate Gaussian Distribution (MGD). As it is widely knows, MGD is defined only by a mean vector and a covariance matrix. We notice that given a MGD it is very simple determine the probability of that an object belongs to the distribution.



(a) Example of pitch extraction



(b) Example of CENS features

Fig. 1: Example of feature extraction from a track containing the same chord played by 8 different instrument. You can see in (a) the different harmonic content while in (b) the instruments are basically indistinguishable

We just need to implement the *Mahalanobis distance* which measures how many standard deviations the selected object is distant from the distribution mean.
MATLAB defines a class called *gmdistribution* which allows to construct an object that contain one or more MGDs. This object can be built either by adding every distributions by ourself (supervised method) or fitting the entire population with a chosen number of distributions (unsupervised method). MATLAB offers also the method *mahal* to compute the Mahalanobis distance between an object X and a GMM.
GMMs represent a very simple and effective technique, which can be easily implemented with the pre-established MATLAB methods. In our work we focus on the supervised approach of GMM, creating the singles subpopulations one at time during the training phase.

2.4. Multiclass Support Vector Machines (MC-SVM)

Support Vector Machines (SVM) are a well known and widely used *Machine Learning* algorithm for binary classification. Before Neural Networks became so popular, they generally outperformed most of the other standard classification (and also regression) algorithms. Some good advantages over them is the possibility of choosing non-linear kernels (the most famous ones are the *Gaussian/RBF* and the *Polynomial*). Also, they are far easier to program and optimize than Neural Networks (less hyperparameters) and they tend to overfit less.
Their main problem, though, is that they were conceived for binary classification. Their extension to multiclass classification is not trivial but there are some standards ways to do it. One of them is already implemented in Matlab and it's called *Error-Correcting Output Codes* (ECOC). It works by creating multiple classifiers and assigning to each of them some classes to be considered positive and some other to be negative and by assigning weights to these binary classifiers, the final decision is taken.
For our tests we used the *One-vs-One* method, meaning that we create a classifier for every couple of classes ($O(K^2)$) assigning one to the positive and one to the negative, while ignoring everything else. This is usually a good tradeoff between complexity and precision, but other options could be explored in future works.

2.5. Hidden Markov Models (HMMs)

Hidden Markov Models are an extremely powerful technique that has been widely used in speech recognition application for twenty years. A HMM consistis in an extension of the classical Markov Model scenario, in which observations are considered probabilistic functions of states. The final result is a pair of stochastic processes connected to each other; one process is directly observable, the other process is not and therefore is called "hidden". A more detailed theory about HMM working can be found in [**?**].
In speech recognition field, HMMs are used as statistical method in order to recover specific sequences of sound. The validity of HMMs in the the particular case of chords transcription has already been demostrated several times, as in [**?**] and [**?**]. Both these works utilize the HMM approach proposed in [**?**], training HMM with the *expectation-maximization algorithm*. As our work is not limited to implementing HMM, we choose to take a different approach. In particular we combined HMM to the SVM technique previously presented.
As suggested in [**?**], HMM working can be supported by *Viterbi algorithm*. Given a HMM scenario and a sequence of observables, Viterbi algorithm aims to find the most likely sequence of states. It works in a recursive mode, first finding the probability of each state for each observable, then identyfing the most probable states sequence. Given a total number

of states N and a sequence of observables T long, the Viterbi algorithm shows a complexity equal to $O(T \times N^2)$. In our work the implementation of Viterbi algorithm represented one of the most difficult challenges. Codes already existing raised problems and therefore they needed to be carefully adapted to the experiment setup.

3. EXPERIMENT SETUP

Our experimentation was split in two parts: the first was about recognizing a single chord from a single chord track, the second one was about recognizing the chords of full songs.
For the first part, we used the dataset `jim2012Chords` [**?**] created for their paper [**?**]. It's composed of a total of over 2.000 recordings of 10 guitar chords (both major and minor triads). Four different Guitars are used, as well as Piano, Violin and Accordion. Some tracks were recorded in an anechoic chamber, some other in a noisy environment. For each one of the tracks we obtain a 12-dimensional feature vector using *Chroma Toolbox* and taking the maximum value for each note. Looking at Fig. 1(b), the idea is that we are interested in high values of specific groups of notes in order to guess the chord and the maximum value is a simple way of doing this. It's biggest disadvantage is that impulsive noisy environment can easily mask the useful information introducing high values in the wrong notes. Finally, we run the different methods on these obtained 12D vectors.
For the second part we used as dataset *The Beatles*' discography, which has been professionally transcribed by Christopher Harte. This dataset contains the chords transcription of all The Betales's songs. Harte's work has proven to be extremely high quality and is an excellent resource for speech processing research. Details on how the chords transcription was achieved can be found in [**?**] and in [**?**]. In his work Harte performed a very detailed transcription, obtaining a wide set of chords. For reduce computational cost we mapped the labels transcripted by Harte into 24 labels, which coincide with the 12 notes of the chromatic scale in the major and minor version. For similar reason we selected only 150 Beatles songs out the 180 trasncribed by Harte, keeping anyway a big number of songs to work on. From now on, we indicate our labels dataset with Λ and our songs dataset with $\mathcal{D}_{Beatles}$.
The second part of the experiment was divided in three steps which are *pre-processing*, *training* and *testing*. As machine learning approach, we randomly divided our dataset in two subset that we call *training dataset* $\mathcal{D}_{train}$ and *testing dataset* $\mathcal{D}_{test}$. We call r_{train} the ratio between $|\mathcal{D}_{train}|$ and $|\mathcal{D}_{Beatles}|$ and r_{test} the ratio between $|\mathcal{D}_{train}|$ and $|\mathcal{D}_{Beatles}|$. During the experiment we varied the value of $r_{test} \in \{0.1, 0.2, 0.3\}$, consequently obtaining r_{train} as $1 - r_{test}$. Every song of both the subsets was processed according to one of the technique offered by *Chorma toolbox*. We noticed that we set toolbox parameters so that we didn't get an exaggerated number of frames per song. In *results* section we will see that this choice could make outputs weaker, especially in CLP and CRP cases. After features extraction was completed, we obtained a sequence of *frame-features* for each song in $\mathcal{D}_{Beatles}$; the length of the sequence depended both on the song duration and the chosen featuring process $(CENS, CLP, CRP)$. Harte's work gave us information about the chord periods in each songs; we wanted to make compatible this information with the data obtained by features extraction. With this purpose we divided the periods computed by Harte in sub-periods with equal time-length to the frames computed with Chroma toolbox. For each song we obtained a sequence of *frame-labels* with equal length to the sequence of *frame-features*; from now on we indicate the *frame-labels* sequence for a generic song with $\mathcal{L}_{Harte}$. Finally we noticed that almost all the songs presented periods with no sounds; in the dataset these period were marked with the 'N' label. Obviously 'N' frames were out of our interested and therefore were discarded.
At the end of the *pre-processing* phase we obtained two sets of songs $(\mathcal{D}_{train}, \mathcal{D}_{test})$, in wich every songs is assigned to both a sequence of *frame-features* and a sequence of *frame-labels*. The data contained in $\mathcal{D}_{train}$ were then used to train the MC-SVM and subsequently build the HMM. MC-SVM training was achieved using one chord at time, as done in the first part of the experiment, and using as kernel the *polynomial* one. HMM training was achieved testing the data contained in $\mathcal{D}_{train}$ with the MC-SVM previously built. The resulting errors of this process allowed us to establish values for the *emission probabilities*. Analyzing directly all the $\mathcal{L}_{Harte}$ *transition probabilities* and *initial probabilities* values were computed.
Once all our methods were trained, the *testing* began. We took the *frame-features* of $\mathcal{D}_{test}$ and we processed them first only with MC-SVM methods, then also with HMM. Therefore for each song we obtained two different outputs, that we call $\mathcal{L}_{SVM}$ and $\mathcal{L}_{HMM}$. $\mathcal{L}_{SVM}$ corresponds to the *frame-labels* sequence produced by the MC-SVM. In this process every frame was assigned to a chord, without taking in account the others frame componing the song. $\mathcal{L}_{HMM}$ corresponds to the *frame-labels* sequence produced by the HMM, which took MC-SVM's output and tried to improve it. The HMM's output was obtained implementing the Viterbi algorithm, which allowed to taking into account the whole sequence of chords that compone the song.

4. RESULTS

For the first part of the experiment we simply run all the different classification methods explained in Section 3 randomly splitting the dataset into 70% training and 30% testing. The results can be clearly seen in Tab. 1. Note that the first approach (templates) yields very poor results even in such an idealized scenario, although the more advanced *Harmonic Templates* improve a little bit performance over the simple *Bi-*

Classification	CLP	CENS	CRP
Template (binary)	24.17%	20.08%	24.96%
Template (harmonic)	23.04%	17.79%	20.38%
GMM	14.58%	6.94%	11.81%
SVM (linear)	13.61%	6.39%	12.64%
SVM (poly)	10.56%	4.17%	7.92%
SVM (RBF)	10.00%	4.31%	7.78%

Table 1: Error results of the first part of the tests (single chord recognition).

nary ones. The jump to the more advanced GMMs is pretty clear as much as the one to SVMs. Overall, CENS features have by far the best error rates.

For the second part of the experiment we got two different kind of outputs for each song, which are $\mathcal{L}_{SVM}$ and $\mathcal{L}_{HMM}$. In order to verify the quality of ours results we compared them with $\mathcal{L}_{Harte}$. Given $\mathcal{L}_{SVM}$ ($\mathcal{L}_{HMM}$) and $\mathcal{L}_{Harte}$ for a certain value of r_{test} and for a certain extracted feature we computed the error probability counting all the matching between $\mathcal{L}_{SVM}$ ($\mathcal{L}_{HMM}$) and $\mathcal{L}_{Harte}$.

In Tab. 2 we can look at the error probabilities obtained after the first step of testing, before processing HMM. In Tab. 3 we can look at the results obtained after the second step of testing, after processing HMM. In both cases we show different outputs depending on the testing rate r_{test} and on the chosen extracted feature.

Table 2: Results of the second part before processing HMM

r_{test}	CENS	CLP	CRP
0.1	49.46%	47.39%	62.06%
0.2	46.57%	60.00%	63.76%
0.3	47.65%	56.91%	58.03%

Table 3: Results of the second parts after processing HMM

r_{test}	CENS	CLP	CRP
0.1	44.39%	45.02%	56.41%
0.2	42.96%	59.19%	59.67%
0.3	41.79%	54.84%	52.04%

Observing the results we make the following observations. First, we got best results with CENS features than with CLP and CRP features. This is reasonable since also in the first part of the experiment we got better results with CENS than others type of features. Second, HMM proved to be able to enhance better results than MC-SVM alone in all the cases. This is not obvious since we worked with features of multi-instrumental sounds including also vocal parts. Indeed the chords prediction of some songs often isn't improved but is aggravated. We show this by comparing the two different results obtained with the same $\mathcal{D}_{train}$ as done in Fig. 2.

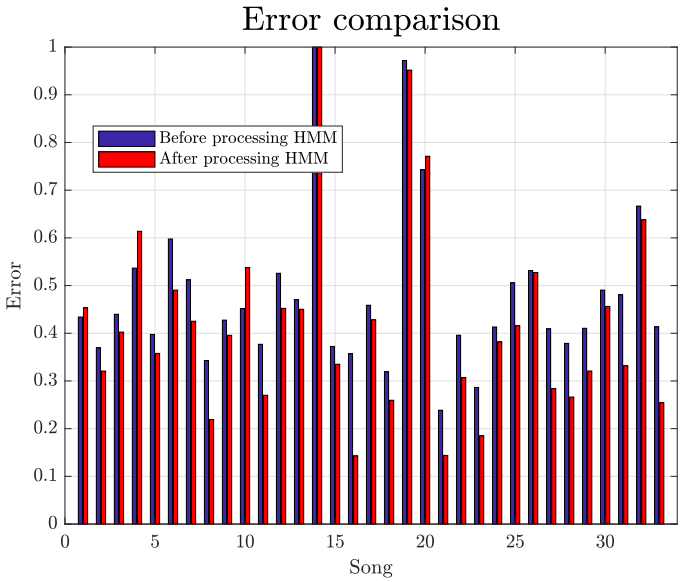


Fig. 2: Error comparison using CENS features and $r_{test} = 0.3$

We concluded that HMM don't always improve our result for every single song. It is able to do so only in average. However what HMM is able to do for each song is reduce the chords variation. We define chord variation as the number of times that a chord sequence varies within the same song, therefore the number of times that $\mathcal{L}(i) \neq \mathcal{L}(i-1)$. This value decreases, since HMM tends to correct the most improbables outputs generated by MC-SVM. We find confirmation to this assumption looking at Fig. 3 and Fig. 4. In particular in Fig. 4 we can see how HMM removes the chord-labels that are not repeated costantly over time and therefore result less likely.

5. CONCLUSIONS

After seeing the results in Sec. 4, it should be clear that the more advanced techniques described in Sec. 2 improve the results very significantly over the oldest and simplest ones.

In particular, CENS features are able to perform really well with all classifiers but the most advanced ones (SVM with non-linear kernels) clearly outperform all the others.

Using this information, we tried to predict chords in full songs having, of course, poorer results, due to the increased complexity of the problem. The HMM-based approach, again, was able to improve performance over the trivial *per-frame* one.

Future works might try to use different type of features and classifiers for the general chord recognition problem. When dealing with entire songs, instead, it might be useful to either extract or subtract specific instruments (e.g. drums) and voice, since we believe they are the main cause of the high error rates obtained.

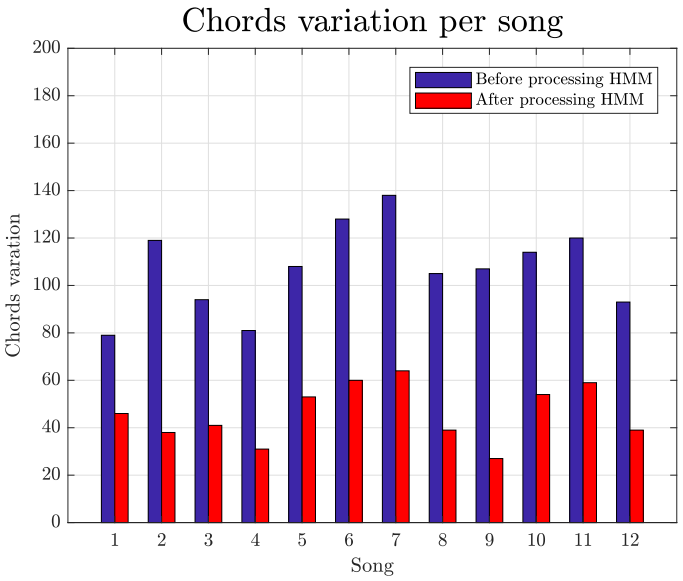


Fig. 3: Chords variation per song using CENS features and $r_{test} = 0.1$

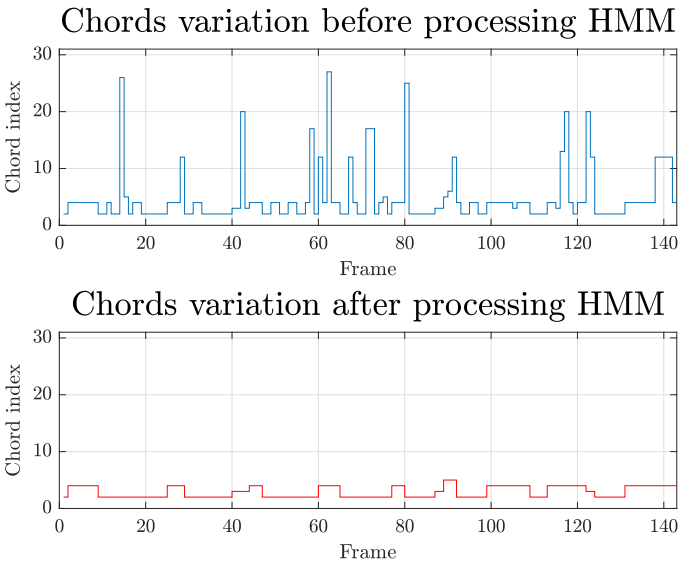


Fig. 4: Chord variation in a single song using CENS features and $r_{test} = 0.1$