Clustering and modelling structural representations of percussive audio files using Gaussian mixture distributions

Project Report

Mattia Paterna

Aalborg University Copenhagen
Sound and Music Computing

This document has been designed in LaTeX. All the scripts, simulations and plot have been done using Mathworks' MATLAB software. Every effort has been made to ensure that all the information presented is correct.

Sound and Music Computing
Aalborg University Copenhagen
http://www.aau.dk

Title:
Clustering and modelling structural representations of percussive audio files using Gaussian mixture distributions

Theme:
Sound and MusicInnovation

Project Period:
Spring Semester 2016

Project Group:
IX

Participant(s):
Mattia Paterna

Supervisor(s):
Hendrik Purwins

Copies: 1

Page Numbers: 26

Date of Completion:
February 16, 2017

Abstract:

In this project, an unsupervised technique to model and cluster a set of observation has been described and deployed. The evaluation is performed using several percussive audio files, which have been edited for the purpose of the project. The HDP-HMM inference sampler is then run using several data configurations. These configurations come from different mixture models, which have been either fitted to data or built around statistical parameters inferred from it.

Lyd og Musik
Aalborg Universitet København
http://www.aau.dk

Titel:
Clustering and modelling structural representations of percussive audio files using Gaussian mixture distributions

Tema:
Sound and Music Innovation

Projektperiode:
Forårssemestret 2016

Projektgruppe:
IX

Deltager(e):
Mattia Paterna

Vejleder(e):
Hendrik Purwins

Oplagstal: 1

Sidetal: 26

Afleveringsdato:
16. februar 2017

Abstract:

In this project, an unsupervised technique to model and cluster a set of observation has been described and deployed. The evaluation is performed using several percussive audio files, which have been edited for the purpose of the project. The HDP-HMM inference sampler is then run using several data configurations. These configurations come from different mixture models, which have been either fitted to data or built around statistical parameters inferred from it.

Contents

Preface

This report has been prepared in partial fulfilment of the requirement for the theme: *Sound and Music Innovation* in Sound and Music Computing, 3rd semester in the Academic year 2016-2017

The report is the result of a three-month period work during the Fall semester and the month of January, 2017 as well.

I would really like to thank my supervisor, Hendrik Purwins, for his help and dedication despite the quite few time that both of us had. Special thanks should I give to Ricard Marxer, whom I and Hendrik had a meeting with. That meeting has been so helpful to me to understand how to correctly assess my question and guide my work under a new and brighter light.

Aalborg University, February 16, 2017

Mattia Paterna
<mpater15@student.aau.dk>

Chapter 1

Introduction

In the effort to contribute to the development in the field of information retrieval and in its use in musical applications, a promising statistical learning method has been used to analyse structure representations of percussive audio files. This report is intended as a continuation of a first understanding of it, which has been made the previous semester.

Particularly, whereas in the previous work some lack in conducting the experiment could be noticed, here a thorough evaluation has been done. Such an evaluation is now performed using several percussive audio files, which have been edited for the purpose of the project. The HDP-HMM inference sampler is then run using several data configurations. These configurations come from different mixture models, which have been either fitted to data or built around statistical parameters inferred from it.

1.1 Aims and motivations

As the very first step was to understand how the HDP-HMM changes depending upon a particular settings of its parameter, this step is mainly focused in understanding how the *inference sampler* works if it is fed with data that follows a model, which replicates the assumption of such a sampler.

1.1.1 The HDP-HMM in short

From the previous work, the HDP-HMM statistical model can be seen as an Hidden Markov Model with an infinite state space [18], where the number of states is unbounded and computed via Gibbs sampling [7]. Doing so, a Bayesian nonparametric model is used, namely the Hierarchical Dirichlet Process. Given this premise, when running the model a Gaussian underlying base distribution has been always considered. That leads to a model where each of the possible states

is represented by a Gaussian mixture model whose number of component is specified *a priori*[1]. The outcome is a state sequence based on transition probabilities, which are modelled using the given prior. In other words, a clustering process of the time series is computed inside the inference sampler, yielding to a number of states (i.e. the *clusters*) whose members are described using a Gaussian mixture model each.

1.1.2 Research questions

Therefore, it could be possible to show how such mixture models can well approximate a time series depending on the number of parameters to estimate. It could also be possible to generate data from such models and fed the HDP-HMM inference sampler with it. This last proposition might show how well the inference sampler is able to infer the model parameters as they change in their number.

Finally, the research questions can be formulated as follows:

1. *is a model replicating the HDP-HMM assumption able to approximate an audio file?*
2. *is the HDP-HMM able to model a structural representation of an audio file?*
3. *how should data be processed to ensure a good result?*

1.1.3 Objectives

As stated in [7]

> *The motivation for this project is to learn more about these machine learning methods with the purpose of, as a future work, developing a system which a performer could improvise with.*

However, in the use of the HDP-HMM some constraints take place. As an example, the choice of the upper bound in the number of clusters representing a musical structure may rise doubts about the fully effectiveness of such state-of-art batch clustering method[15]. Therefore, some considerations on the possible constraints has to be made. More on this can be found in chapter 3, where the creation of the test files along with their label structure is described.

Finally, a good *trade-off* between the number of incoming observations and the number of parameters to estimate is needed to be achieved as a good rule for the parameter inference.

[1] Albeit it takes always a value of 1 since a problem in the toolbox was stated by the Author's toolbox herself.

1.1.4 Structure of the report

To possibly answer the first question several models are provided, which all are Gaussian mixture models with different types of covariance matrices. Further details are given in chapter 2.

Regarding the second question, this report aims to enlighten the experiment session provided in [1] adding more data and settings, as stated in the foregoing. Nevertheless, the optimal parameters found in [1] have been used throughout in the experimental setup. Further details are given in chapter 3

Moreover, the *dimensionality* of data has been stressed out. That is, data are processed using several techniques in order for it to be arranged in a *low-dimensionality* mapping. That aims to both find a representation of data which could show more insight and, at the same time, reduce the number of parameters to estimate. It is shown how the curse of dimensionality is achieved if the high-dimensionality dataset is taken as input. More information on data and examples is given in chapter 3. Further details on data reduction can be found in chapter 2.

Finally, some remarks about the structure of the report. In chapter 2 an overview of the data reduction techniques and the models is provided. Chapter 3 refers to the experimental setup and some short discussion on it. Chapter 4 provides an organic conclusion to the report.

Chapter 2

Gaussian mixture modelling

In order to address the first research question several models have been built, which are then used to describe the test data[1]. The variability of such models depends on the choice of their parameters. They all are based on the Gaussian mixture model, of which a short theoretical explanation is given below. Two main strategies have been used in deploy the generation:

1. fitting a mixture model to data using the Matlab object *fitgmdist*, which returns a model whose parameters are estimated inside the process;
2. building a mixture model on top of data estimating the parameter manually, *i.e.* sample mean and covariance are computed before and may be altered.

Doing so, it is possible to compare how well the models can approximate data and to simulate data from all of them. These random samples are then used to feed the HDP-HMM inference sampler.

2.1 Motivation

As said in chapter 1, Gaussian mixture models are widely used for data clustering. However, as described in [7], the previous stage of this work is still dependent on the *agglomerative* clustering stage proposed in [14]. This stage is originally performed to yield to a regularity measure in order to build an homogeneous structure, but it is required no longer. Therefore, the clustering process can be worked out by the HDP-HMM itself and computational resources can be saved, avoiding multiple redundant processes.

[1]Throughout all the report, the term *data* is referred to the full-dimensional set of Mel Frequency Cepstrum Coefficients (MFCC). A further explanation about this feature extraction technique is given in [1]

2.2 Theory

The Gaussian mixture model is a particular type of distribution in which a linear superposition of two or more normal distributions aims to describe a data set[2]. In other words, a mixture distribution is made of several *components*, each of it having its probability density function. A mixture model can therefore be intended as a strategy to create more complex probability distribution.

Gaussian mixture model are widely used in many fields of machine learning and pattern recognitions, such as clustering. Their parameters are determined by *maximum likelihood*, using typically the expectation-maximisation (EM) algorithm.

2.2.1 Gaussian mixture models

A mixture of Gaussian can be described using the form

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k) \tag{2.1}$$

where each Gaussian density $\mathcal{N}(x|\mu_k, \Sigma_k)$ is called *component* of the mixture and is describes by its own mean μ_k and covariance Σ_k. The parameters π_k are called *mxing coefficients*. It can be proved that

$$\sum_{k=1}^{K} \pi_k = 1 \tag{2.2}$$

with the requirement that each $\pi_k \geq 0$. That means, each mixing proportion must assume a value bounded between 0 and 1.

The form of the Gaussian mixture model is therefore influenced by the parameter π, μ and Σ, that need to be estimated.

2.2.2 EM algorithm

The Gaussian mixture model described in 2.2.1 can be interpreted under the concept of *latent* variable, *i.e.* a random variable that can not be directly observed. In this case, the latent variable may be the component *identity*. In other words, the assignment of data points to specific component of the mixture has to be defined. For this purpose, the EM algorithm is used.

Such an algorithm is based on a two-stage optimisation that is repeated until convergence. Indeed, the assignment of data points inside a specific cluster is inferred computing the Euclidean distance of such points to the vector $\{\mu_k\}$, which

[2]This section closely follow [2], ch. 9 and partly [12], ch. 3.9 in the general description of the theory behind Gaussian mixture models.

represent the centres of clusters. Doing so, for each point a binary indicator variable r_nk can be introduced. The variable takes value 1 if a point x_n is assigned to the cluster k and 0 otherwise. An *objective* function can then be defined, and it is given by

$$J = \sum_{n=1}^{K} \sum_{k=1}^{K} r_{nk} \|x_n - \mu_k\|^2 \tag{2.3}$$

This function represents the sum of the squares of the distances of each data point x_n to the assigned vector $\{\mu_k\}$, given the indicator variable r_nk and is called *distortion measure*. The aim of the EM algorithm is to find values for r_nk and $\{\mu_k\}$ so that the objective function is *minimised*.

The algorithm first assigns data points to cluster w.r.t the cluster means $\{\mu_k\}$, then computed new values for the means w.r.t r_nk. In other words, one variable is computed while keeping the other fixed. The two stages are repeated until no further change in the assignment is reached, or the maximum number of iterations is exceeded. To give an example, when performing a clustering solution in Matlab using the EM algorithm a maximum number of repetition can be set. Convergence for the algorithm is assured, either to a global or to a local minimum.

A drawback exists such that the convergence to a minimum might depend on the initial condition. In the attempt to mitigate it, a common procedure is to initialise $\{\mu_k\}$ equal to a random subset of K data points, where K is the number of cluster used in the clustering evaluation. Moreover, the *k-means* is often used to initialise the parameters before applying the EM algorithm. This is common practice in order to limit the number of iteration the EM requires to approximate and is often used to find a suitable initialisation for a Gaussian mixture model.

EM for Gaussian mixtures The aim of EM in a given Gaussian mixture model is to maximised the likelihood function (in this case, the *log-likelihood*) w.r.t. the parameter of this model, *i.e.* mixing coefficients, means and covariances for each component. In a very first step, the parameters are initialised via k-means and the log-likelihood is computed. The posterior probabilities are evaluated using the current parameters (the **E** step). The parameter are then re-estimated using the new posterior probabilities (the **M** step). Finally, the log-likelihood is evaluated and convergence is checked.

Comparison to the k-means with the EM algorithm shows some similarity. However, k-means is often referred to be a deterministic approach in which each data point is uniquely associated with one cluster. Therefore, the k-means is said to perform a *hard* assignment of data point. Conversely, the EM is a probabilistic approach, since the assignment is based on the posterior probabilities. A point may be associated with several clusters at the same time. That is referred as a *soft*

assignment technique. It is also worth mentioning how the k-means estimates only the centroids, while the EM requires a full estimates of mean and covariances.

2.3 Dimensionality reduction

In a very first attempt of running the HDP-HMM, the so-called *curse of dimensionality* has experienced. As explained in [16] an increase in dimensionality often requires a large increase in the numerosity of data, as the performance of the algorithm may drastically lower. Several techniques have been formulated to mitigate such a phenomenon. A general advise to improve machine learning performance is to apply some technique to reduce the dimensionality of data.

In [19] *dimensionality reduction* is described as the transformation of the dataset into a low-dimensional mapping, where the original geometry of data has preserved as much as possible. A very first distinction is drawn between *linear* and *non-linear* techniques.

Among the different categories, *feature projection* has been used in this work as an additional stage prior to the clustering process. Together with one of the most popular techniques, namely the *Principal Component Analysis* [16], a recent dimensionality reduction technique called *t-SNE* [13] has been deployed. Both of them are completely unsupervised. Unlike suggested in [13], the dimensionality reduction is carried out by using a two-dimensional mapping instead of using the *intrinsic* dimensionality of data.

Finally, a further motivation in applying such techniques can be also found in the need of data visualisation in order to get some intuition about data arranged in a multidimensional space. The suggestion here is to build a map in which each object is represented by a point. Similar objects are represented by points whose distance is short. In other words, the distances among points in the low-dimensional map reflect the similarities in the original high-dimensional data. Doing so in any dimensionality reduction techniques, an *objective function* has to be minimised.

However, depending on the chosen technique, different strategies are used for this purpose. When using *PCA*, that is much concerned with describing as much of the variance in the data as possible [19], the large pairwise distances in the Euclidean distance are preserved but not the small though. In contrast, the *t-SNE* techniques developed by van der Maaten aims to stress out such small distances, which can also well represent data distributed in non-linear manifolds, via minimising the *Kullback-Leiber* divergence between two sets of probabilities that represent probability distributions over pair of points both in the original data space and in the reduced mapping. For a complete explanation of the algorithm, [13] is highly suggested.

2.4 Clustering stage

Once the dimensionality reduction stage has been computed, the resulting mapping is used for the clustering stage. Such a process makes use of the Gaussian mixture model to provide a meaningful insight into the data distribution, seeking out any pattern inside it.

At first glance, one Gaussian mixture model has been used to describe the whole data, with a number of component equal to the number of unique labels inferred from the *optimal structure* (a thorough explanation is given in chapter 3). That model did not lead to any expected result though. First of all, the number of components is not provided in an unsupervised manner. Moreover, if artificial data is sampled from such a model and compared to the original data, the observations that should be of the same group show way different values for each MFCC dimension.

To overcome this issue, it has been decided to represent each group, *i.e.* the labels that make up the structural representation, with a Gaussian mixture model so to yield to a multiple mixture model configuration for a specific structure. This also reflects closely the structure of the HDP-HMM model.

As said before, two different techniques have been used. The former tries to *fit* a mixture model to data, given specifics about the covariance matrix type, the initialisation stage and a regularisation parameter if needed. The outcome of this technique is a Matlab *gmm* object containing all the details about the mixture model. Such a built-in function computes automatically all the parameters needed and, in case of *ill-conditioned* covariance matrix[3] provides regularisation. On the other hand, it does not let the user choose specific configurations for the covariance matrix, *e.g.* a *isotropic* or a *pooled* covariance matrix. For this purpose, the latter tries to *build* a Gaussian mixture model having a particular type of covariance matrix.

The motivation here is to *reduce the number of parameter to estimate*. As expressed in [17], the structurization of a sample covariance matrix, whose the *diagonal* type is an example, could reduce the number of parameter to be estimated. This is particularly the case when data dimensionality is high and number of samples is limited (p. 246):

> *a ratio learning-set dimensionality plays an essential role in selecting and constructing the classification algorithm*

and that could also apply in building a statistical model.

[3] It is not possible defining a Multivariate Gaussian distribution if the covariance matrix is not belonging to the space of symmetric positive definite *n x n* matrices [8]

2.4.1 Determination of the number of components

An important issue in the mixture modelling is the selection of the number of component. As stated in [10] if a too large number of components per mixture is chosen the model may over-fit the data, while too fewer components may not approximate the true underlying model. A good trade-off has to be achieved.

Although several techniques exist, for this task the *Variance-ratio Criterion* (VRC), or *Calinski-Harabasz* criterion is deployed in conjunction to a *k-means* clustering solution. A Calinski-Harabasz index is computed for a selected numbers of k components and the optimal number of clusters is the solution with the highest VRC_k[4].

2.4.2 Choice of covariance matrix

As expressed in [2] the number of parameters can be a general limitation in the use of Gaussian distribution as density model.

For instance, a general symmetric covariance matrix Σ has $D(D+1)/2$ independent parameter, plus D parameter for the mean μ, leading to a total number of $D(D+3)/2$ parameters to estimate. That means, it has a quadratic grows and for high-dimensionality data space the computational tasks can become hard to solve. Using restricted forms of covariance matrix, or *structurization*, might help in reducing the complexity.

A matrix is said to be *diagonal* if all the entries outside the diagonal are null. A diagonal covariance matrix takes form $\Sigma = diag(\sigma_i{}^2)$ where only the variance σ^2 has to be computed. That lead to a total of *2D* parameter and therefore complexity of $\mathcal{O}(n)$. In turn, the contours of constant density are aligned to the axes.

A even more restricted form of covariance matrix is called *isotropic*, which is just proportional to the identity matrix and takes form $\Sigma = \sigma^2 I)$. Only $D + 1$ parameters have to be inferred for such a covariance matrix. However, the resulting Gaussian will have spherical surfaces of constant density. In other words, the shape of the probability density is much restricted.

Finally, a choice regarding the component inside a Gaussian mixture model can be made. The model could have either separate or shared covariance matrices. The latter introduces the concept of *pooled* covariance matrix, where the mean of the covariance matrix is used throughout all the components.

[4]This measure identifies the partitioning of data, where the larger the VRC_k the more defined the cluster, given a large between-cluster variance SS_B and a smaller within-cluster variance SS_W[3]

Chapter 3

Experimental setup

In chapter 2, an unsupervised technique to model and cluster a set of observation has been described and deployed. This chapter aims to answer to the second research question. To do so, the sets of Mel-Frequency Cepstrum Coefficient (*MFCC*) computed from four different test percussive sequences have been first reduced in their dimensionality, and then used as input for the HDP-HMM inference sampler. Moreover, the sets are used to simulate data sampled from Gaussian mixture model either fitted to or built on top of them. The motivations in using such different samples are

- assess if the HDP-HMM inference sampler improves its accuracy if models with fewer parameter to infer are used;
- assure to some extent a variability and provide different sampling for the estimation procedure since the whole set of MFCC observation is used for each audio file.

3.1 Preliminaries

The HDP-HMM Matlab toolbox by Emily B. Fox [11] is used throughout all the experimental setup. Doing so, unlike described is [7] only one set of hyperparameters α and γ is used, particularly the set of *optimal* parameters inferred in [1]. Indeed, the fixed values for the upper bound in the number of mixtures and for the number of component within each mixture is inherited from [1].

The analysis performance of the HDP-HMM model is observed using four new audio files that have been edited specifically for this purpose[1]. Further details on the editing techniques and the constraints are given in the section below.

[1] These audio files can be published if needed.

3.2 Audio files

Since general conclusions can not be drawn if using only one audio percussive sequence because of the limited number of observations and the consequent risk for over-fitting when trying to apply a model on it, four new test audio files have been selected. Here are provided some observations:

- the tempi for such audio files are *85*, *100* and *138* beat per minute (BPM);
- two audio files in tempo of 100 BPM exist, of which one played in half tempo so to have one pattern repetition per bar with *bass* drum and *snare* drum on the strong beat positions;
- the second audio file in tempo of 100 BPM is played so to have two repetitions of the same rhythmic pattern inside each bar;
- the audio file in tempo of 138 BPM contains some spurious sounds from other instruments that were playing in the same room and may therefore show how MFCC and how much the performance might be affected from such resonances;

Generally speaking none of them contain tempo or meter changes, as well as syncopation within the rhythmical structure. They all are in $\frac{4}{4}$ time signature and contain a regular rhythmic structure based on the repetition of bass drum (BD) and snare drum (SD) on the strong beats, most of the time in conjunction with cymbals. Three of them contain mainly *hi-hat* sounds, while the audio file in tempo of 138 BPM shows a rhythmical structure where the *ride* cymbals take place instead.

As stated in [7] and [1] the three most repeated sequences are extracted from a full inference sampler trial. The *Adjusted Rand Index* (ARI) [20] has been chosed as a measure of similarity between the outcome of the HDP-HMM and the *optimal* label structure[2] that has been made up for each file.

3.2.1 Beat and label structure

In order to get a sequence of timestamps corresponding to the main onsets within an audio file, *Sonic Annotator* [5] has been used. Particularly, the library of Vamp audio feature extraction plugins developed by the Centre of Digital Music at Queen Mary, University of London has been deployed. Moreover, several scripts have been created in the Matlab environment to this purpose. From each audio file, a list of onset times of events using a complex domain onset detection function, which calculates an onset likelihood function for each spectral frame, and picks peaks in a smoothed version of this function [9] and a list of the metrical beat

[2] With *labels* the different events played are intended, *e.g.* bass drum, snare drum etc.

position has been extracted using the beat tracking method explained in [6]. The latter returns both the estimated beat location and the corresponding estimated tempo.

The main goal is to define a beat structure that is matched to the onset annotations, discard the unimportant onsets between the beats and assign a label to each metrical position so that an *optimal* structure representation is given. Doing so, the similarities between such lists are sought, together with the onset times that lie on a metrical position. In the process of finding the similarities, a small interval ϵ is defined and the index of the closest annotation onset to a beat location is defined if

$$idx = abs(beat_i - onsets) < \epsilon$$

where $beat_i$ is the selected beat location and *onsets* is the list of onset times. Doing so, an *optimal* beat structure is created, which closely follow both the beat locations and the tempo fluctuations.

The two lists are then used in conjunction to provide two different label structures for each audio file. The two structures differs in such a way that one has either double or half as much the number of labels of the structure based on the beat locations. That allows for a multiple interpretation of the same audio file depending on the beat measure chosen, *e.g.* 4-th or 8-th note. If needed, the foregoing discarded onsets are used to build the sub-beat structure. Doing so, the inter-onset intervals are computed and then compared to the local interval computed using the local tempo estimates. An onset is considered part of the sub-beat structure if

$$abs(ioi_n - localIoi) < localTempo * 0.2$$

where ioi_n is the inter-onset interval for a selected onset, $localIoi$ is the local inter-onset interval and $localTempo$ is the local tempo estimate[3]. If a structure is needed that considers onsets that lie on a larger beat, *e.g.* a minim, half of the onsets are simply discarded, yielding to an inter-onset interval that is double as much the original value.

Finally, each timestamps is labelled manually and a cluster representation of such a label structure is created, where labels are replaced by a number assuming each category of audio event as a separate group. The assumption here are:

- not to consider minimal changes within the phrase as structural relevant, but *nuances*;
- if two events are played together, *e.g.* a bass drum and a hi-hat, interpret them with respect to the metrical position and assign a *bd* label to them if lying on a strong accent and *hh* otherwise.

[3]Such a multiplication factor has been chosen by heuristics.

Such assumption may introduce some bias in the performance evaluation since the HDP-HMM can interpret specific set of MFCC as belonging to a group that differs from the associated label. Nevertheless, it is worth mentioning that the model is completely *unsupervised* and only the measure of similarity can change, but now the way in which the model is inferring the given state sequence. It is however remarkable to consider all these constraints when assessing any performance measure.

3.2.2 Analysis of a selected audio file

The corresponding label structure obtained from the Matlab scripts is then exported into a *csv* format and then imported in Sonic Visualiser [4] for a comparison. In figure 3.1, the anacrusis of the audio file in tempo of 100 BPM and played in half is shown. The red vertical lines correspond to the detected onsets using the onset detection function plugin, while the blue lines corresponds to the estimated beat positions.

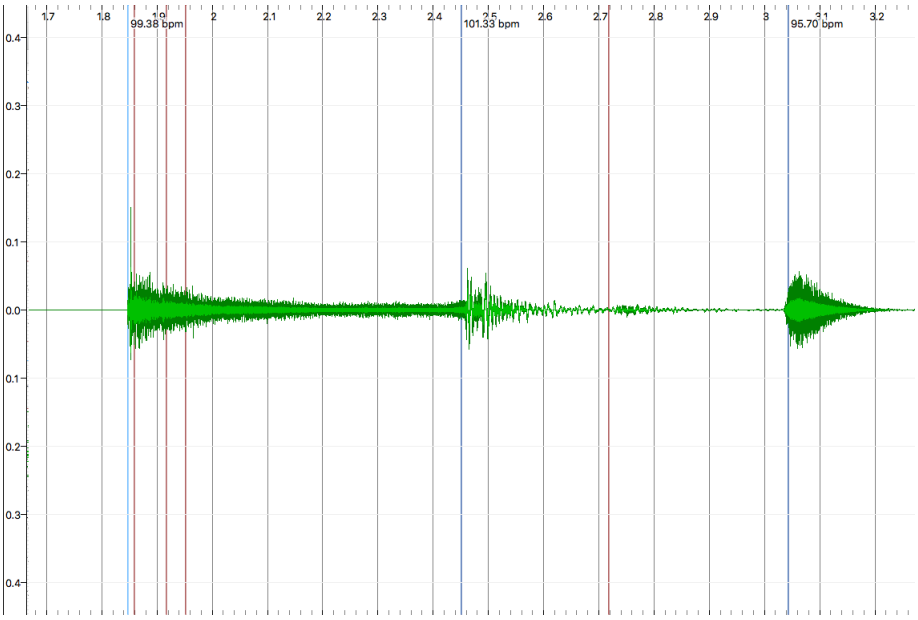


Figure 3.1: Detected onset times and detected beat positions.

Figure 3.2 shows how the resulting sequences of selected onsets and labels look like. It is also noticeable how the onsets that lies in sub-beats positions are not taken into consideration, and so their events. However, there is still room for a possible subdivision of the beats into smaller audio segments.

A remark on tempo fluctuation can be made: the tempo track presents most of its value concentrated around the mean, although some outliers take place.

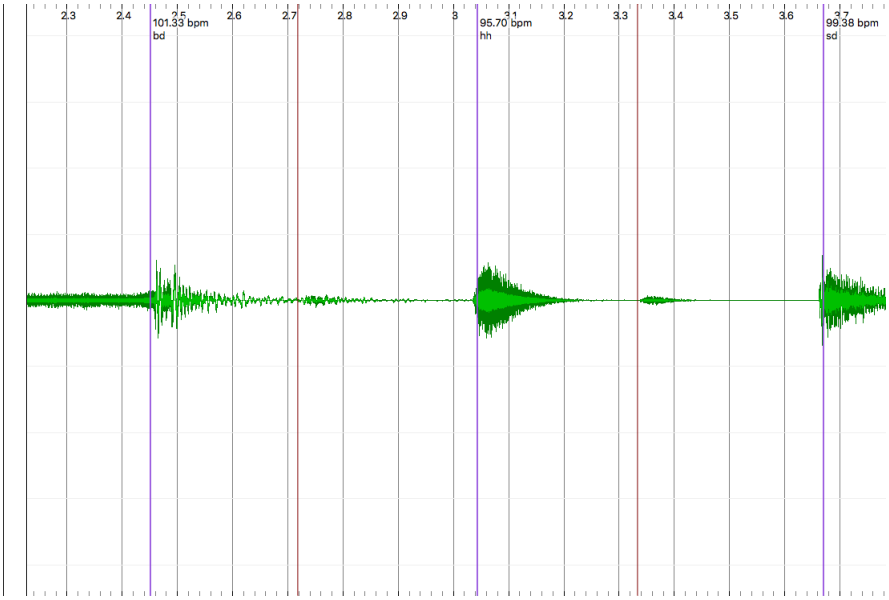


Figure 3.2: Selected onsets and corresponding labels if considering a *4-th* note beat.

The figure 3.3 shows a particular case where there is no matching between the estimated beat position and the onset times around it. In such a situation, no new onset is computed, but directly the beat location is taken in the optimal structure representation instead. Again, the tempo estimation on top left appears quite different compared to the other two shown - that are, in factual act, the same.

3.3 Evaluation

Some of the results of the HDP-HMM inference sampler are provided in table A in Appendix A. The model is run using all MFCC sets for the selected audio files, of which both the structure representations have been used. It is important to remark how:

- for the audio file in tempo of 85 BPM only the structure based on the 8-th note beat division is run, since the 4-th note beat structure has a fewer number of observations (32) compared to the minimum number of parameter to estimate;
- the files in tempo of 85 BPM is reduced to a low-dimensionality mapping using the *PCA* technique, since the *t-SNE* technique showed no meaningful result on it;
- some of them have been used to simulate data using different models with a

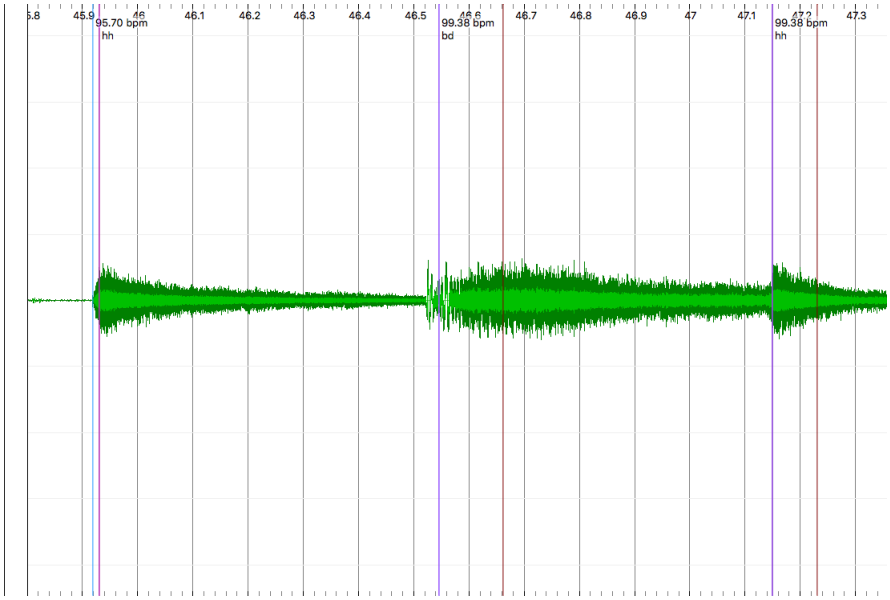


Figure 3.3: Case-scenario if no match is found: the estimated beat location is considered and no onset is computed, differently from [14] and [7].

different number of parameters to estimate.

The ARI values in table A have to be intended as the mean of the values for the three most repeated state sequences inside the trial. A trial was made of 1200 iteration of the Gibbs sampling, of which a *burnout* stage made of the first 200 has been discarded when computing the performance measure.

Table 3.3.1 gives a comparative of the mean ARI values throughout all the audio files using the two dimensionality reduction techniques together with different data set and considering solely their *beat* structure[4].

3.3.1 Discussion

At a first sight, it is noticeable how the largest performance measures have been achieved using either the original MFCC set or a simulation from a fitted mixture models. Nevertheless, the HDP-HMM shows similar performances if the model is fed with a simulation from built mixture models.

[4]The values between parentheses correspond to the mean ARI value is quite bad performances are discarded. See table A for further details.

DR type	sim	ARI
t-SNE	no	0.4498 (*0.6720*)
PCA	no	0.5459 (*0.6892*)
t-SNE	fit	0.9564
PCA	fit	0.6948
t-SNE	build	0.6372
PCA	build	0.6772

Table 3.1: ARI mean values for different case-scenarios.

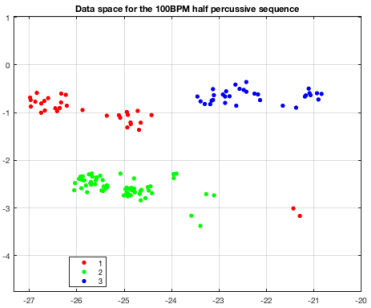
Particularly, t-SNE seems appearing a better technique if a low-dimensionality mapping is then used to fit a mixture model to it. Inversely, PCA seems mapping the data space in such a way that all the samplings from a mixture model with several covariance matrix structurizations make the HDP-HMM performs better. In addition, in two audio files the t-SNE algorithm is not able to lead to a mapping that keeps the original clustering properties of the MFCC set, yielding to a poor performance of the inference sampler. A simple observation on the audio files which t-SNE performed bad on shows a major presence of hi-hat sounds, which are extremely harsh and exhibit long tail. Moreover, in both files a *backbeat* is present such that a *bd* is played upbeat followed by a *hh* on the prominent beat location. As a result of it, such a beat location may contain part of the *bd* tail, explaining a possible misplacement of the MFCC set for that onset time and the consequent unrecognition.

Furthermore, given that the inference sampler computes a full covariance matrix for each mixture model, *i.e.* a state, it can be pointed out that a number of *60* parameters needs to be estimated at each run of the Gibbs sampling, assuming the initial setup[5]. That represents a value roughly between $\sim$ 30 and $\sim$ 50% of the total number of input observation.

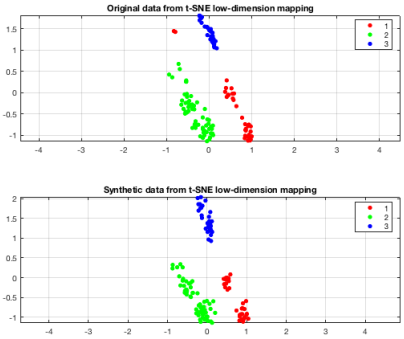
An explanation of the different measures can be found if the distributions are observed. As shown in fig. 3.4a and 3.4b for the second audio file in the table, some *outlier* events are detected both in the original data space and in the reduced mapping but not in the simulation. Moreover, the clusters in the simulation exhibits a smaller within-cluster distance. That may explain an increase of $\sim$ 20% in the similarity measure for the HDP-HMM inference sampler.

If the third audio file in the table is taken as example, the same behaviour in the HDP-HMM inference sampler can be noticed. It is interesting to point out how the fitting using a *full* covariance matrix does make a worse performance than the one using a *diagonal* covariance matrix. Moreover, the frequency of repetition for the most repeated state sequence is quite large. That is, the inference sampler can model such audio file. It is also important to notice how the dimensionality

[5]The initial setup for this experiment is a maximum number of 10 states of 1 component each



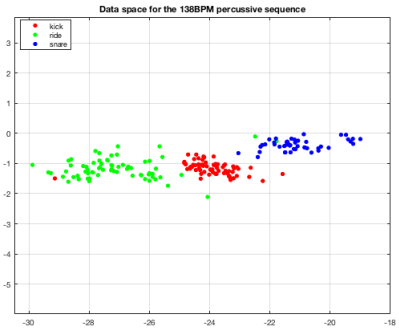
(a) Data space for the MFCC set.



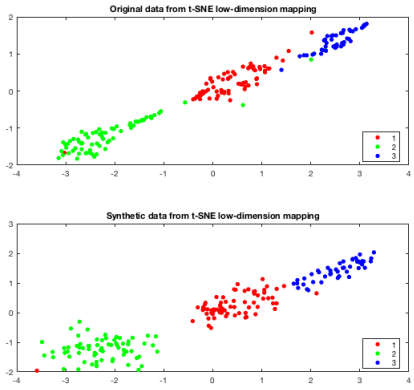
(b) Low-dimensional mapping and sampling from the mixture models fitted to it.

Figure 3.4: The 100 bpm audio file

reduction technique is relevant in defining a good performance measure. Using the same model to simulate data from but a different technique, such as PCA, the ARI measure dramatically decreases. Finally, it is worth mentioning how such the total number of estimated parameter for such a model is in number of *18*, since the original MFCC set can be clustered in 3 groups according to the clustering solution provided by the Calinski-Harabasz criterion. Fig. 3.5 shows both the sampling, while fig 3.6 shows the mixture model fitted to the group of *sd* events.



(a) Data space for the MFCC set.



(b) Low-dimensional mapping and sampling from the mixture models fitted to it.

Figure 3.5: The 138 bpm audio file

If considering the model built on top of the same audio file, the performance measure for the HDP-HMM is good in all cases. Particularly, the frequency is much higher if a *pooled* covariance matrix is used. As shown in fig. 3.7, an isotopic

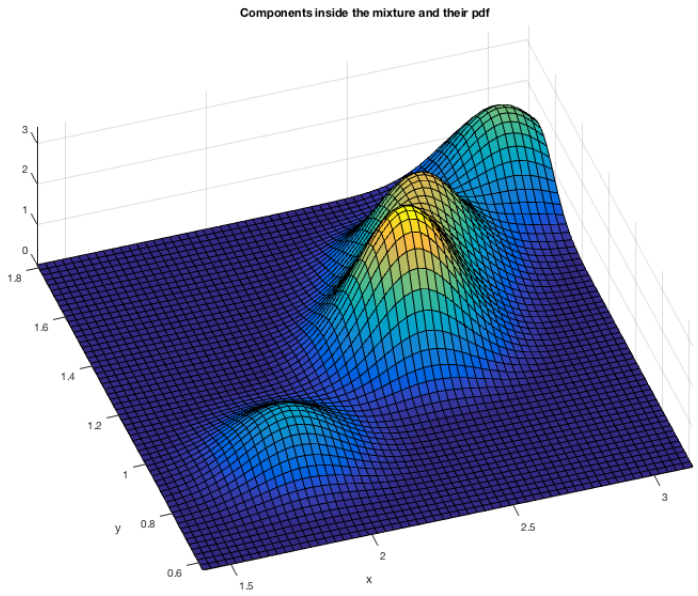


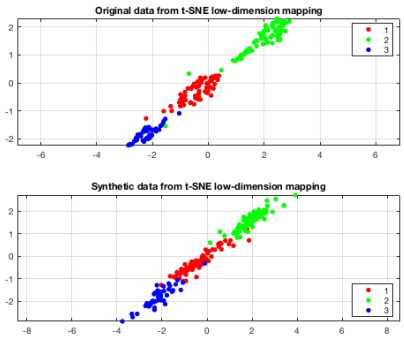
Figure 3.6: Mixture model fitted to the *sd* group. The mixture contains 4 components, which share the same diagonal covariance matrix.

matrix does not represent the low-dimensional mapping as much good for this case. Conversely, the HDP-HMM has lower performance with a model with a pooled covariance matrix for the second audio file. For such models, the number of parameters to estimate range between *10* for the former, *9* for the latter - that is, only the sample mean for each normal distribution plus the variance.

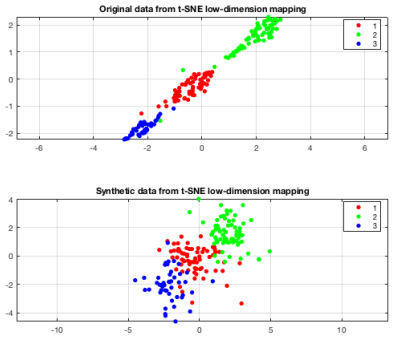
3.3.2 Conclusion

Generally speaking all the original sets of MFCC exhibit some event that are placed far from its cluster. That may be due to the presence of audio material that is different to the expected. Particularly, it can be pointed out how that most of the misplacement cases interest *bd* event. An explanation to this tendency can reside in the presence of cymbals in most of them. An unambiguous example is shown in fig. 3.4a where two *bd* events are located closer to the *hh* group. If the audio file is played, it turns out easily how such events contain a predominance of cymbal sound, in this case a *crash*.

Moreover, a thorough analysis of the results clearly shows that the considered models are able to simulate data according to a model that approximate a low-dimensional mapping of audio features. Given that such models replicate the HDP-HMM inference sampler model, the first research question may have posi-



(a) Using pooled covariance matrix.



(b) Using an isotopic covariance matrix.

Figure 3.7: Data simulation for the 138 bpm audio file

tive answer. Finally, in most cases the HDP-HMM shows performance measures that are not inferior if model with a lower number of parameters to infer is used. Thus, it might be possible to run the inference sampler with different covariance matrix type which could reduce the parameters to infer without losing a successful modelling. Nevertheless, the conjunction of the t-SNE technique and the mixture models fitted to data, each of them using several components, shows an outstanding result. This suggests a possible enhancement in the HDP-HMM performance when using more than one component for each state.

Chapter 4

Conclusion

In this project, an unsupervised technique to model and cluster a set of observation has been described and deployed. In particular, a new set of test audio files has been produced in order to assess the performance of such a statistical model.

A thorough analysis of a specific configuration of the HDP-HMM inference sampler has been replicated and deployed, which involves the use of Gaussian mixture models for clustering data. Several of such models have been either built or fitted to the original set of Mel-Frequency Cepstrum Coefficients extracted from the audio files to prove the opportunity to approximate this audio feature and analyse its effectiveness when using simpler models which require a lower number of parameters to estimate. The use of a technique to reduce the original data space is common practice, nevertheless a linear and a non-linear technique have been deployed. A future investigation is suggested, which may point out which technique is better depending on the audio material.

It has also been shown how a general improvement in the HDP-HMM performance arises if an improved version of the homogenization process described in [14] and later in [1] is used. However, both versions are limited by several constraints that has to be considered when a structural representation is created for an audio file. Finally, audio files with poor recording resolution or that do not exhibit sharp and well-separate percussive event are still not well approximated. The deployment of source separation or filtering techniques could be suggested for this purpose.

Appendix A

Table of results

In this appendix, a table with some of the result of the HDP-HMM inference sampler for different audio files, dimensionality reduction techniques and sampling is shown. The column *DR type* shows the dimensionality reduction technique used to realise the mapping of the MFCC sets in two dimensions, while the columns *sim* and *cov* describe which model has been used to simulate data, either a fitted (*fit*) or a built (*build*) mixture model, and the specifics for the covariance matrix (*sample* = sample mean, *diag* = diagonal, *fixed* = isotopic). Finally, the column *freq* shows the numbers of instances for most repeated state sequence in the inference trial.

Structure	DR type	sim	cov	ARI	freq
085 sub	PCA	no	-	0.1163	411
100 half	t-SNE	no	-	0.7369	*943*
100 half	PCA	no	-	0.6758	556
100 half	t-SNE	fit	shared diag	0.9271	549
100 half	PCA	fit	shared diag	0.7528	672
100 half *sub*	t-SNE	no	-	0.1599	15
100 half *sub*	PCA	no	-	0.1701	33
100 half *sub*	t-SNE	build	sample diag	0.0768	1
100 half *sub*	t-SNE	build	pooled	0.1171	1
100 half *sub*	t-SNE	build	fixed	0.3116	88
100	t-SNE	no	-	*0.0056*	975
100	PCA	no	-	0.7464	150
100	PCA	fit	shared diag	0.7480	250
138	t-SNE	no	-	0.6070	366
138	PCA	no	-	0.6453	97
138	t-SNE	fit	shared full	0.4851	387
138	t-SNE	fit	shared diag	**0.9857**	**954**
138	PCA	fit	shared diag	0.5837	89
138	t-SNE	build	sample full	0.6595	319
138	t-SNE	build	sample diag	0.5794	182
138	t-SNE	build	pooled	0.6450	683
138	t-SNE	build	fixed	0.6648	102
138	PCA	build	sample full	0.6632	149
138	PCA	build	sample diag	0.6223	509
138	PCA	build	pooled	0.7533	2
138	PCA	build	fixed	0.6701	61
138	t-SNE *1D*	no	-	0.4851	387
138	PCA *1D*	no	-	0.6397	49
138 duple	PCA	no	-	*0.9928*	*826*

Table A.1: ARI values for the HDP-HMM inference trials. Results of the HDP-HMM inference for all the audio files using several MFCC set configuration and simulated data. The structure variable contains *sub* when the sub-beat structure and *duple* when the duple beat structure is intended, *i.e.* a beat every minim note.

Bibliography

[1] Jose Luis Diez Antich et al. "Unsupervised Learning of Structural Representation of Percussive Audio Using a Hierarchical Dirichlet Process Hidden Markov Model". In: *Mml 2016 9th International Workshop on Machine Learning and Music*. 2016.

[2] Christopher M Bishop. "Pattern recognition". In: *Machine Learning* 128 (2006), pp. 1–58.

[3] Tadeusz Caliński and Jerzy Harabasz. "A dendrite method for cluster analysis". In: *Communications in Statistics-theory and Methods* 3.1 (1974), pp. 1–27.

[4] C. Cannam, C. Landone, and M. Sandler. "Sonic Visualiser: An Open Source Application for Viewing, Analysing, and Annotating Music Audio Files". In: *Proceedings of the ACM Multimedia 2010 International Conference*. Firenze, Italy, 2010, pp. 1467–1468.

[5] Chris Cannam et al. "Linked Data And You: Bringing music research software into the Semantic Web". In: *Journal of New Music Research* 39.4 (2010), pp. 313–325.

[6] Matthew EP Davies and Mark D Plumbley. "Context-dependent beat tracking of musical audio". In: *IEEE Transactions on Audio, Speech, and Language Processing* 15.3 (2007), pp. 1009–1020.

[7] JL. Diez Antich and Mattia Paterna. *Unsupervised structure analysis of an audio file using Hierarchical Dirichlet Process Hidden Markov Models*. Aalborg University, København, 2016.

[8] Chuong B Do. "The multivariate Gaussian distribution". In: *Section Notes, Lecture on Machine Learning, CS* 229 (2008).

[9] Chris Duxbury et al. "Complex domain onset detection for musical signals". In: *Proc. Digital Audio Effects Workshop (DAFx)*. Vol. 1. Queen Mary University London. 2003, pp. 6–9.

[10] Mario A. T. Figueiredo and Anil K. Jain. "Unsupervised learning of finite mixture models". In: *IEEE Transactions on pattern analysis and machine intelligence* 24.3 (2002), pp. 381–396.

[11] Emily Fox et al. “Nonparametric Bayesian learning of switching linear dynamical systems”. In: *Advances in Neural Information Processing Systems*. 2009, pp. 457–464.

[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT Press, 2016.

[13] Laurens van der Maaten and Geoffrey Hinton. “Visualizing data using t-SNE”. In: *Journal of Machine Learning Research* 9.Nov (2008), pp. 2579–2605.

[14] Marco Marchini and Hendrik Purwins. “Unsupervised analysis and generation of audio percussion sequences”. In: *Exploring Music Contents*. Springer, 2010, pp. 205–218.

[15] Richard Marxer and Hendrik Purwins. “Unsupervised incremental online learning and prediction of musical audio signals”. In: *IEEE Transactions on Audio, Speech and Language Processing* (2015).

[16] Claude Sammut and Geoffrey I Webb. *Encyclopedia of machine learning*. Springer Science & Business Media, 2011.

[17] Aušra Saudargienė. “Structurization of the covariance matrix by process type and block-diagonal models in the classifier design”. In: *Informatica* 10.2 (1999), pp. 245–269.

[18] Yee Whye Teh and Michael I Jordan. “Hierarchical Bayesian nonparametric models with applications”. In: *Bayesian nonparametrics* 1 (2010).

[19] Laurens Van Der Maaten, Eric Postma, and Jaap Van den Herik. “Dimensionality reduction: a comparative”. In: *J Mach Learn Res* 10 (2009), pp. 66–71.

[20] Ka Yee Yeung and Walter L Ruzzo. “Details of the adjusted Rand index and clustering algorithms, supplement to the paper “An empirical study on principal component analysis for clustering gene expression data””. In: *Bioinformatics* 17.9 (2001), pp. 763–774.