Transcription and classification of audio data by sparse representations and geometric methods

Michal Genussov

Transcription and classification of audio data by sparse representations and geometric methods

Research Thesis

In Partial Fulfillment of the Requirements for
the Degree of Master of Science

Michal Genussov

Submitted to the Senate of the Technion—Israel Institute of Technology

Kislev 5771 Haifa November 2010

Acknowledgement

The research thesis was done under the supervision of Professor Israel Cohen in the Department of Electrical Engineering. I would like to thank him for his dedicated guidance and support throughout all the stages of this research.

I would also like to thank my colleagues Asaf Elron, Ronen Talmon and Sivan Gleichman, for fruitful discussions and helpful comments. Special thanks to Dr. Yizhar Lavner for sharing with me his knowledge in phonetics, and specifically in classification of phonemes, and special thanks to Dr. Ron Rubinstein, for sharing with me his knowledge in sparse representations, for fruitful talks and for his helpful insights.

Finally, I express my deep gratitude to my family - Rina, Ron, Merav and Nadav, for their constant love, encouragement and support.

The Generous Financial Help Of The Technion Is Gratefully Acknowledged.

Contents

List of Figures

List of Tables

List of papers

- Genussov, M.; Cohen, I., "Musical genre classification of audio signals using geometric methods", 18th European signal processing conference (EUSIPCO-2010), Aalborg, Denmark, August 2010.

- Genussov, M.; Lavner, Y.; Cohen, I., "Classification of unvoiced fricative phonemes using geometric methods", IWAENC international workshop on acoustic echo and noise control, Tel-Aviv, Israel, August-September 2010.

- Genussov, M.; Cohen, I., "Transcription of polyphonic music based on sparse representations in a structured dictionary", in preparation.

Abstract

Transcription of music and classification of audio and speech data are two important tasks in audio signal processing. Transcription of polyphonic music involves identifying the fundamental frequencies (pitches) of several notes played at a time. It is an intriguing task which has attracted researches to confront with for the last decades. Its difficulty stems from the fact that harmonics of different tones tend to overlap, especially in western music. This causes a problem in assigning the harmonics to their true fundamental frequencies, and in deducing spectra of several sounds from their sum.

Classification of audio and speech data includes classification of music by genre and identification of speech phonemes. Traditional classification methods consist of two main stages: the first is feature extraction, in which relevant features (usually temporal and spectral) are extracted from the signal, and the second is classification according to these features. The problems with these methods are that they are usually not well-adjusted to the non-linear structure of the feature vectors, and they don't consider the redundancy of the features, leading to unsatisfactory classification results and to high computational complexity.

In this thesis, we introduce transcription and classification methods which are based on representation of the data in a meaningful manner. For transcription of polyphonic music we present an algorithm based on sparse representations in a structured dictionary, suitable for the spectra of music signals. Thanks to the structured dictionary, the algorithm does not require a diverse or a large data set, and is computationally more efficient than alternative methods.

For classification of audio data we propose to integrate into traditional classification methods a non-linear manifold learning technique, namely ”diffusion maps”. In this technique, a graph is built from the feature vectors, and the distances in the graph are

mapped to Euclidean distances, so using Euclidean distances for classification after the mapping becomes meaningful.

Finally, we examine empirically the performances of the proposed solutions. We show that our structured-based dictionary transcription system outperforms existing methods in several tasks of transcription, especially in the difficult case of a small data set with multiple overlaps of harmonics. In classification of musical pieces by genre and in identification of unvoiced fricative phonemes by diffusion maps, most of the samples are classified correctly. However, comparing to classification using the features without the mapping, and comparing to mapping with principal components analysis (PCA), we find that the classification results are not improved. This implies that the assumption of the non-linear redundancy between the features depends on several factors, including the application and the efficiency of classification by the features before mapping.

Nomenclature

Abbreviations

2D	2-Dimensional
ACF	Autocorrelation Function
ADSR	Attack, Decay, Sustain, Release
CQT	Constant Q Transform
CSM	Cosine Similarity Measure
DM	Diffusion Maps
FFT	Fast Fourier Transform
FN	False Negatives
FOCUSS	FOcal Underdetermined System Solver
FP	False Positives
GMM	Gaussian Mixture Model
HPS	Harmonic Product Spectrum
ISA	Independent Subspace Analysis
k-NN	k-Nearest Neighbors
K-SVD	K - Support Vector Machine
LDA	Linear Discriminant Analysis
LLE	Locally Linear Embedding
LS-OMP	Least-Squares OMP
MAP	Maximum A-Posteriori
MFCC	Mel-Frequency Cepstral Coefficients
MIR	Music Information Retrieval
MOD	Method of Optimal Directions
MP	Matching Pursuit

MS dictionary	Musically-Structured dictionary
MS K-SVD	Musically-Structured K-SVD
MS MOD	Musically-Structured MOD
NMF	Non-negative Matrix Factorization
OMP	Orthogonal Matching Pursuit
PCA	Principal Component Analysis
QDA	Quadratic Discriminant Analysis
RMS	Root Mean Square
SVD	Singular Value Decomposition
STFT	Short-Time Fourier Transform
SVM	Support Vector Machine
TP	True Positives
ZCR	Zero Crossing Rate

Notation

$x(n)$	time-domain signal
$X(f)$	discrete-time Fourier transform of signal x
x	scalar
$\mathbf{x}$	column vector
$\mathbf{A}$	matrix
$\mathbf{A}^{-1}$	inverse matrix
$\mathbb{R}$	The set of real numbers
$\|\cdot\|_p$	ℓ_p norm
$\mathbf{x}_{(k)}$	$\mathbf{x}$ at the k_th iteration
S	the support of a vector
$(\cdot)^T$	transpose operation
$(\cdot)^\dagger$	Moore-Penrose pseudoinverse
$\mathbf{a}_j$	the j_{th} column of the matrix $\mathbf{A}$
$p(\cdot)$	probability function
$p(\cdot\|\cdot)$	conditional probability
$E\{\cdot\}$	expectation
$Var(\cdot)$	variance
$Tr(\cdot)$	trace
$< \cdot, \cdot >$	inner product
$\|\cdot\|$	absolute value
$\mathcal{M}$	manifold
$W[k,n]$	time-frequency coefficient of W
$\mathbf{O}^+$	the set intersecting the positive orthant
$\mu(\cdot)$	mutual coherence
$\rho(\cdot)$	one-sided coherence
$\mathbf{1}$	column vector of ones
$\mathbf{I}$	identity matrix
$M_t[n]$	magnitude spectrum at analysis frame t and frequency bin n
sign(·)	sign function

λ_i	i_{th} eigenvalue
$\psi_i(j)$	i_{th} eigenvector at the j_{th} entry
$\Psi_t(\mathbf{x}_j)$	diffusion map of $\mathbf{x}_j$ at scale t
δ	relative accuracy
$D_t(\mathbf{x}_i, \mathbf{x}_j)$	diffusion distance at scale t between $\mathbf{x}_i$ to $\mathbf{x}_j$
$\delta_{i,j}$	Kronecker delta
$./$	element-wise division

Chapter 1

Introduction

1.1 Transcription of polyphonic music

Transcription of music is defined as the process of identifying the parameters of an acoustic musical signal, which are required in order to write down the score sheet of the notes [59]. For each note, these parameters are the *pitch*, which is represented in written music by the note symbol, the onset time and the duration, which are represented in written music by a different length of the note. There are also other parameters which are sometimes determined for larger parts of the musical piece and not for individual notes - the timbre of the sound (its 'color'), and the loudness.

There are numerus motivations for transcribing music, which include:

1. **Structured audio coding**: The transcription allows saving only a limited number of parameters, instead of saving all of the samples of the audio signal. This is called the WAV to MIDI transformation. If, for example, we have a one-minute song sampled at 16 kHZ, and each sample is represented by 2 Bytes, it will capture 16k*60*2B=1.92 MB if saved as a WAV file. However, if we keep only the fundamental frequencies ,onset and offset times, timbre and loudness of each tone, then assuming that there are, e.g., 300 tones in the song, and each parameter consumes 2 Bytes of the memory, the whole song saved in a MIDI format will capture only 300*5*2B= 3KB. This is a compression ratio of 1/640!

2. **A helpful tool for musicians**: Automatic transcription of music leads to the

vision of *computer hearing*, i.e. that the computer, equipped by an appropriate software, could 'hear' a song played and produce the music score of the song as an output. This can save the tedious and time-consuming work of listening to music and writing down the score.

3. **Modifying, rearranging and processing music at a high abstraction level**: For example, transposing the scale, changing the rhythm of the song, and many more possible operations.

4. **A pre-stage for information retrieval and classification**: The written score sheet of a piece contains useful features for the purpose of classifying the piece into different categories, such as musical genre or audio mood identification.

5. **Interactive music systems**: For example, a system that generates an automatic accompaniment to the singing or playing of a soloist [28, 74].

In this work we will only focus on the task of identifying the *pitch*. The pitch is the perceived fundamental frequency of a tone. For convenience, we will refer to the task of pitch identification as "Transcription", although the original meaning is that of identifying also the other parameters mentioned before.

First we need to define and explain the term "pitch", in the context of musical instruments. When a sound is produced from a musical instrument, the instrument acts as a resonant system. This means that it vibrates in several frequencies, where the lowest one is the *fundamental frequency* and the other frequencies are called *overtones*. When the musical instrument utilizes strings or air columns, such as wind or string instruments, then the overtones are multiple integers of the fundamental frequency (or close to it), and they are termed as *harmonics*. Such instruments are termed harmonic musical instruments, as opposed to inharmonic ones such as percussion instruments. Open air columns (e.g., a flute) and strings (e.g., a guitar, a piano), produce all harmonics, and closed air columns (e.g., a clarinet) produce only the odd harmonics. An illustration of the first 7 partials (i.e., fundamental frequency and 6 first harmonics) of an ideal string is shown in Figure 1.1.

Pitch is a perceptual attribute of the sound, defined as the frequency of a sine wave that is matched to the target sound in a psychoacoustic experiment [85]. Simply, this

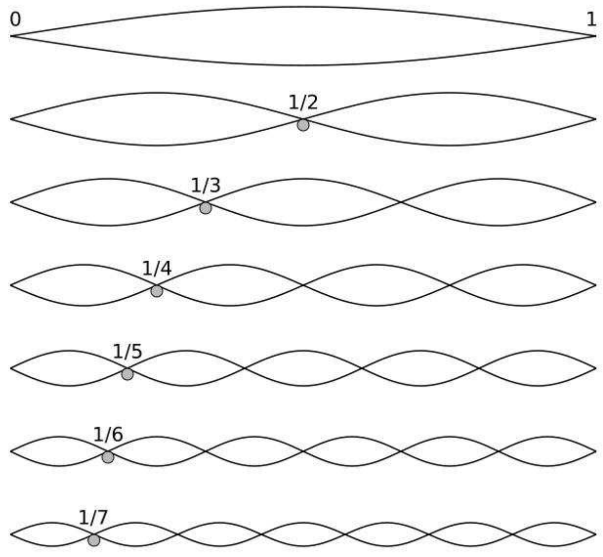


Figure 1.1: The first 7 partials of an ideal string

is the frequency that we perceive as being played by the musical instrument, where the fundamental frequency is its corresponding physical term. In most cases the pitch and the fundamental frequency are the same, and we perceive the pitch as the lowest oscillation frequency. There are some cases where this identity is wrong, for example, when the fundamental frequency is missing (missing fundamental) [47], but we still perceive it as the pitch frequency, because it is the frequency difference between the overtones. We shall not relate to such rare cases, and assume from here on that the pitch equals the fundamental frequency for our analysis.

When dealing with the task of pitch identification, we need to distinguish between two cases - transcription of monophonic music and transcription of polyphonic music. Monophonic music is the case in which a single sound is played at each time instant. For example, a single person is singing, someone is playing a piano with a single finger, a single person is playing a flute and more. For this case, automatic transcription is practically a solved problem. Several proposed algorithms are reliable, commercially applicable and operate in real time. Algorithms for pitch determination can be categorized as time domain, frequency domain, or time-frequency domain algorithms.

Examples for two monophonic transcription methods which yield good identification results, are the autocorrelation pitch tracking [13], and the harmonic product spectrum

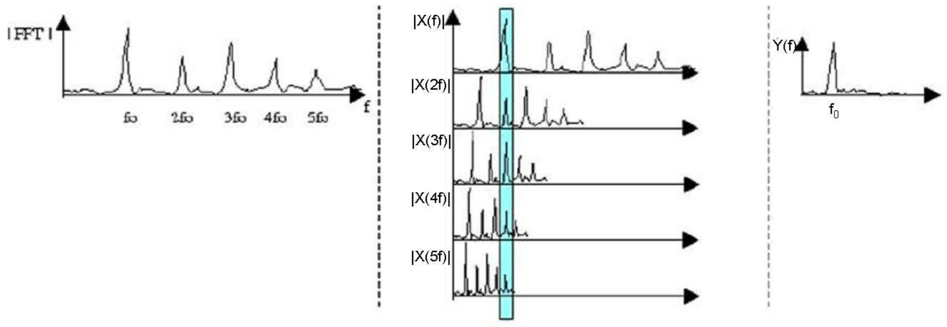


Figure 1.2: Overview of the HPS algorithm

(HPS) [67]. The first method, which operates in the time domain, exploits the periodicity of the sound wave. The autocorrelation function (ACF) for an N-length time window is defined as

$$\phi(\tau) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)x(n+\tau)$$

where $x(n)$ is the sound wave at time n. Because a periodic signal will correlate strongly with itself when offsetting by the fundamental period, we can expect to find a peak in the ACF at the value corresponding to that period. The second method (HPS), which operates in the frequency domain, exploits the harmonic nature of musical sound waves, i.e., the fact that the overtones of the fundamental frequency are approximately multiple integers of that frequency. The method measures the maximum coincidence for the harmonics, according to:

$$Y(f) = \prod_{r=1}^{R} |X(fr)|$$

$$f_0 = argmax_{f_i}\{Y(f_i)\}$$

where $X(fr)$ is a spectral frame, down-sampled by the integer factor r. Because of the harmonic nature of the musical waves, the product of the down-sampled versions of the spectral frame has a maximum value in the fundamental frequency. An illustration of the method is presented in Figure 1.2, which was copied from De La Cuadra et al. [30].

Transcription of polyphonic music, in which more than one note is played at a time, is much more complicated. As far as we know, today there is no algorithm which can identify multiple pitches in an accuracy close to 100%. This is somehow not intuitive to understand, since when a trained human (such as a musician) listens to a polyphonic music piece, he can distinguish and identify the different notes, although played simultaneously. The difficulty in solving this task automatically stems from the fact that most often, especially in western music, the frequency bands of different sounds overlap, and several harmonics of different sounds overlap. This causes a difficulty in assigning the harmonics to their true fundamental frequencies, and in deducing spectra of several sounds from their sum [50, 51]. An example of the spectrum of the three notes composing the C major chord - C4, E4, G4, is presented in Figure 1.3.

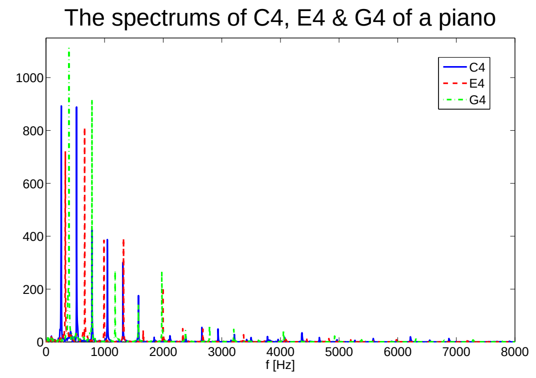


Figure 1.3: The spectrum of C major chord

Since the 1970s, when Moorer built a system for transcribing duets [64], there has been a growing interest in transcribing polyphonic music, and different algorithms were offered for this task. As in transcription of monophonic music, the algorithms can be divided here also into three main groups: time-based, frequency-based and time-frequency-based algorithms.

The time-based group includes methods which are based on the autocorrelation function [29, 62, 63] and on the periodicity of the signal [17, 58, 72]. Since a pitched sound is semi-periodic in time (except of a change in its amplitude), its autocorrelation function is semi-periodic as well and it has local maximums in integer multiples of the time interval of the pitch. The first (and the largest) local maxima corresponds to the time interval of the pitch. Cemgil et al. [17] developed a Bayesian framework in which high level (cognitive) prior information on music structure was coupled with low level (acoustic physical) information in a principled manner. The sound of each note was modeled as a damped oscillator, and the coupling was obtained using a Kalman filter model. Marolt [58] performed transcription using two steps - first the signal was filtered by an auditory model, and then the notes were identified using networks of adaptive oscillators, where each network corresponded to another note in the piano and consisted of oscillators for several harmonics of the fundamental frequency.

The frequency-based group includes methods which are based on typical harmonic patterns in the frequency domain [52, 73], which can be mapped to a logarithmic scale to better fit the human auditory system [13, 14, 25, 26]. The harmonic pattern of harmonic instruments, which was described before, is used in order to identify the pitch. Poliner & Ellis [73] identified notes in polyphonic music by support vector machines, applied on spectral features derived from the signal, and applied a temporally constraint on the outputs via hidden Markov models.

The combined time-frequency-based group includes methods which use a time-frequency image, such as spectrogram or scalogram [84], or cepstrum analysis [20], in which the Fourier transform of the log amplitude of the Fourier transform of the signal is used for pitch detection [66]. Intuitively, The cepstral coefficients characterize the change of the frequency components with time. Saito et al. [78] defined the term “specmurt”, which is very similar to cepstrum, except of the fact that the log function is applied on the frequency axis of the Fourier transform instead of on its amplitude. Saito et al. assumed that all tones in a polyphonic sound have a common harmonic structure, and therefore the fundamental frequencies can be found by de-convolving the observed log-frequency spectrum with the assumed common harmonic structure.

An interesting work on which we would like to expand our talk is by Klapuri [52]. Klapuri suggested a computational model which simulates the auditory system, followed by an iterative pitch estimation, taking into consideration the polyphonic nature of the signal. The system processes the signal as follows:

1. An input signal $x(n)$ is passed through a bank of linear bandpass filters which models the frequency selectivity of the inner ear.

2. The signal $x_c(n)$ at band c is subjected to nonlinear processing to obtain a signal $y_c(n)$ which models the level of neural activity in the auditory nerve fibers representing channel c.

3. An equivalent function to spectral density is calculated over all the channels.

4. The fundamental frequency at each time window is estimated using a method which utilizes the harmonic nature of the signal, and is based on the autocorrelation function of the signal.

5. The previous stage is repeated in an iterative manner in order to find all the fundamental frequencies in the current time window. After each iteration the harmonics of the found fundamental frequency are deleted, under the assumption that their spectral shape follows a constant function.

The idea of using *sparse representations* as a time-based or frequency-based method for transcription of polyphonic music, was first suggested by Abdallah and Plumbley [2]. It was later improved and expanded [1, 71, 72], and inspired other works, which used non-negative matrix factorization [83], independent subspace analysis (ISA) [89] and sparse coding [90]. The idea of "sparse representations" means writing a signal as a linear combination of very few underlying functions, which are contained in a *dictionary* of underlying functions. This is implemented by multiplying the dictionary of the underlying functions by a *sparse* vector (a vector that contains very few non-negative elements compared to its length), giving the method its name.

In played music, only a small number of notes is played simultaneously compared to the number of notes available. This is the motivation for applying Sparse Representations of music signals. In the time domain sparse representations is applied by modeling each sample as a sum of few scaled and shifted versions of some underlying functions. In the frequency domain, the approach is based on the idea that power spectra of different notes approximately add, assuming random phase relationships.

There are several drawbacks and insufficiencies in former transcription methods:

1. Some works [52,78] assumed that the spectral shape of the harmonics can be modeled by a constant function, and then deduced spectrums of several notes from their combination based on this assumption. This is very inaccurate, since the spectral shape changes as a function of many factors, which include:

   - The type of musical instrument - Musical instruments differ from each other by their spectral shape, allowing the listener to identify them and giving the instruments their unique *timbre* ('color' of sound). The ideal spectral shapes (with an infinite time window) of a double bass and a violin, at different fundamental frequencies, are presented in Figure 1.4. It can be seen that the spectral shape changes as a function of the instrument and as a function of the fundamental frequency.

- The total intensity of the tone - The higher the intensity level is, the greater is the number of harmonics generated by the instrument.
- The fundamental frequency - The number of harmonics tends to be larger as the fundamental frequency is lower, and the spectral shape changes as well.
- The stage in the time envelope of the sound - The amplitude of a tone can be modeled in time as an envelope which is composed of four stages - attack, decay, sustain, release (ADSR). The total intensity of the tone is different in each of the stages, leading to a different composition of harmonics, which depends on the stage of the time window we chose for analyzing the spectral shape. The ADSR envelope is presented in Figure 1.5. Amplitudes in high frequencies tend to have lower energy, and tend to decrease faster than amplitudes in lower frequencies [69].

2. The former algorithm based on sparse representations for transcription [1] requires a large and diverse database of notes, in which each note is played at least once by itself, or else, the learned dictionary will not represent the individual notes, but their combinations. Other methods [25, 58, 73, 77] are supervised methods, i.e. require a training set of pieces in order to transcribe another musical piece.

We develop an algorithm for transcription of polyphonic music, based on sparse representations with a parametric dictionary suitable for the spectra of music signals. Our algorithm overcomes the problems above by learning the spectral shape of the harmonics from the signal, and by imposing some restrictions on the structure of the dictionary, such that it would represent individual notes even if they are not being played individually, in an unsupervised manner.

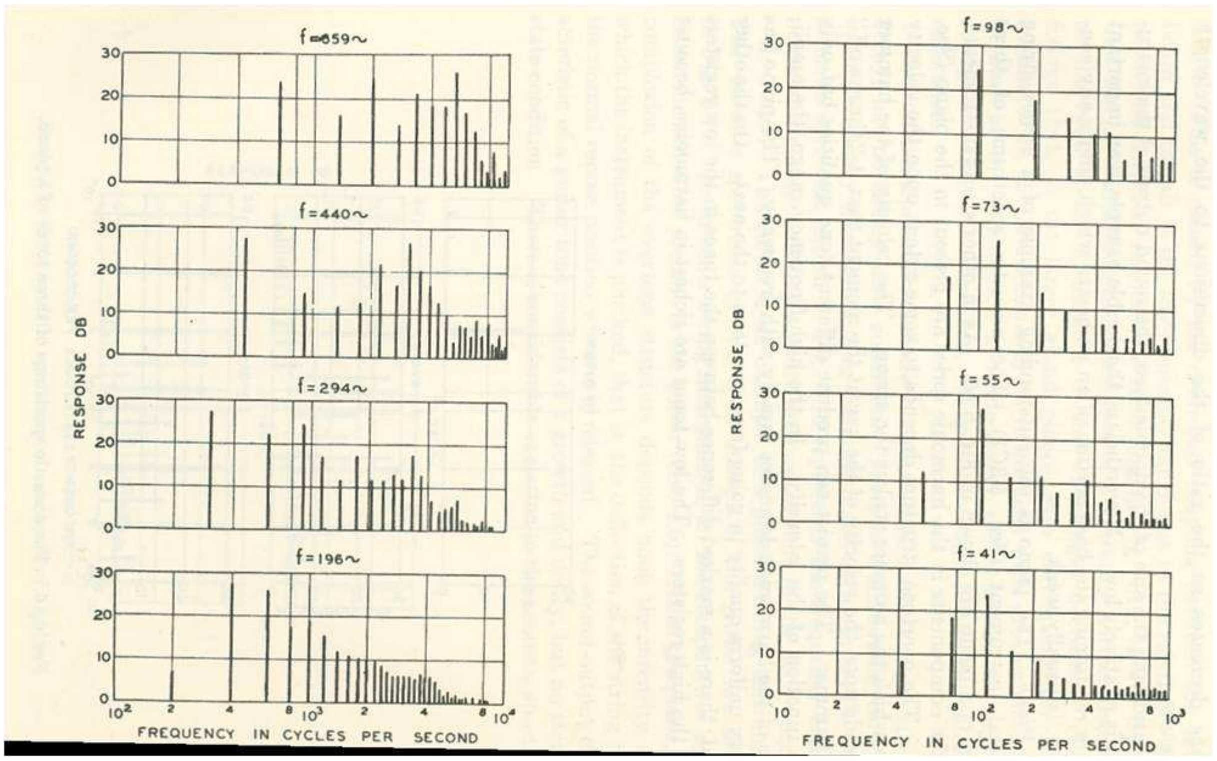


Figure 1.4: Ideal acoustic spectrums of the four open strings of a violin (left) and of the four open strings of a double bass (right). The spectrums of the different musical instruments and of the different fundamental frequencies, differ from each other in the spectral shape. The figure is taken from [69].

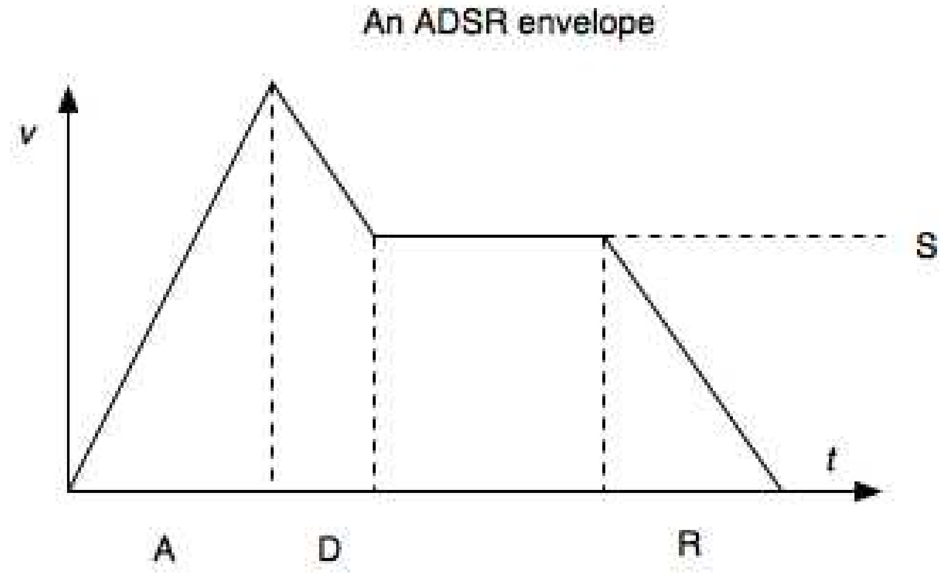


Figure 1.5: An ADSR (attack, decay, sustain, release) envelope which models the amplitude of signal as a function of time.

1.2 Classification of audio and speech data

Automatic classification of audio and speech data is a key stage in many algorithms and applications in signal processing. The types of audio classifications are various and include identification of musical instruments, artist/composer, musical mood and musical genre. Audio classification is a task in musical information retrieval, which is a field with growing interest that holds annual competitions (http://www.music-ir.org/mirex/wiki/2010:MIREX_HOME). Speech data classification includes identification of speakers, mood or phonemes, distinction between singing and speaking and between males and females. In this work we focus on two main classification applications and show the advantages of adding a non linear manifold learning stage named "Diffusion Maps" to traditional classification methods. Traditional classification methods include two stages:

1. Feature extraction - relevant spectral and temporal features, for characterizing the classes.

2. Classification using methods such as k-nearest neighbors (k-NN) [27], linear discriminant analysis (LDA) and quadratic discriminant analysis (QDA) [61], support vector machines (SVM) [24] or Bayesian approaches [18], which use a maximum a posteriori (MAP) decision rule.

We examine the performance of classification using Diffusion Maps in two audio and speech classification tasks - classification of songs by genre, and classification of unvoiced fricative phonemes. We shall describe the applications and explain our motivation for focusing on them.

The first application is classification of musical pieces according to their genre. Musical genres are labels created and used for categorizing and describing the vast universe of music. Different genres differ from each other in their instrumentation, rhythmic structure and pitch content of the music. They include, for example - classic music, jazz, rock etc. Automatically extracting musical information is gaining importance in recent years as a way to structure and organize the increasingly large numbers of music files available digitally on the web. In addition, features evaluated by automatic genre classification can be used for tasks as similarity retrieval, segmentation and audio thumbnailing.

Tzanetakis [88] performed classification of music by genre according to three types of features: timbral texture features, pitch content features and rhythmic content features, which were calculated over short time windows, ranging from 23 ms to 1.5 s, depending on the type of the feature. Clustering of the songs in the training set was conducted using Gaussian mixture models (GMM) with initialization using k-means, and classification of a new song was conducted according to labeled samples by k-nearest neighbors. Li et al. [55] extracted features capturing the local and global information of music signals simultaneously by computing histograms on their Daubechies wavelet coefficients, and used them for classification by k-NN, SVM, LDA and GMM. Benetos and Kotropoulos [10] performed non-negative tensor factorization on tensors composed of concatenation of time-dependant feature matrices, and classified them using cosine similarity measure (CSM). Holzapfel and Stylianou [48] developed a similar system, which used non-negative matrix factorization (NMF) on the extracted features and classification using a statistical model. Such non-negative factorizations constitute linear dimensionality reduction techniques. Panagakis et al. [70] extracted features based on the auditory temporal modulation (using constant Q transform) of the signal and used sparse representations for the classification, where each atom in the dictionary matrix represented the features of a different genre. Works which used non-linear classification include that of Xu et al. [91], which used SVM and McKay and Fujinaga [60], which used neural networks.

The second application is classification of the unvoiced fricative phonemes /s/, /sh/, /th/, /f/. Classification of phonemes is the process of finding the phonetic identity of a short section of a spoken signal [31]. It is a key stage in many speech processing algorithms and applications, such as spoken term detection, continuous speech recognition and speech coding, but it can also be useful on its own, for example in selective processing of phonemes for the hearing impaired, or in the professional music industry. The unvoiced fricative phonemes are specifically important since they tend to be indistinguishable for the hearing impaired [6].

Ali and Van der Shpiegel [5] classified fricative consonants (both voiced and unvoiced) according to their duration, intensity, spectral shape and formant transitions. The classification was obtained via a binary decision tree. Bauer et al. [6] classified the phonemes /s/,

/z/, /C/, /t/ (in German) based on spectral and temporal features, followed by a range check based on the probability functions of the phonemes, a likelihood-based classification, and finally post-processing by a majority based decision according to the classification of the short time windows in each phoneme. Fu et al. [42] clustered the unvoiced fricatives /s/,/sh/,/f/,/th/ and /h/ according to spectral moments - the classification was applied according to the first and second order moments of the spectrum at all time window. Gordon et al. [43] studied the distinguishing features between voiceless fricatives in seven different languages, and found that the overall spectral shape, the center of gravity and the duration are distinguishing features, in a decreasing order of effectiveness. Frid and Lavner [41] used classification with SVM based on relevant spectral and temporal features for automatic classification of fricatives. They performed classification in two stages - on the first stage they classified to two groups - sibilants (/s/, /sh/) and non-sibilants (/f/, /th) and on the second stage they classified the phonemes in each group according to a limited set of features.

There are two fundamental problems in traditional methods for classification of audio and speech data:

1. The number of samples required in order to capture the nature of the signals and in order to differ between them efficiently, grows larger as the complexity of the problem grows. Due to "the curse of dimensionality", the complexity of the problem is related to the dimensionality of the feature vectors. As the number of features increases, the computational complexity and the sample complexity increase as well, leading to the need in a dimensionality reduction technique. Since usually the features are dependant and redundant, such dimensionality reduction is reasonable.

2. Traditional classification techniques are not adaptive to the intrinsic geometry of the feature vectors. We assume that the feature vectors of natural audio and speech data lie on a non linear, low dimensional manifold. For example, some classification techniques are based on Euclidean distances, which do not represent intrinsic distances between the feature vectors on the manifold. This problem is demonstrated in Figure 1.6, presenting a synthetic example of a non-linear manifold, called *Swiss roll*. Each point on the manifold represents a feature vector of a different data input.

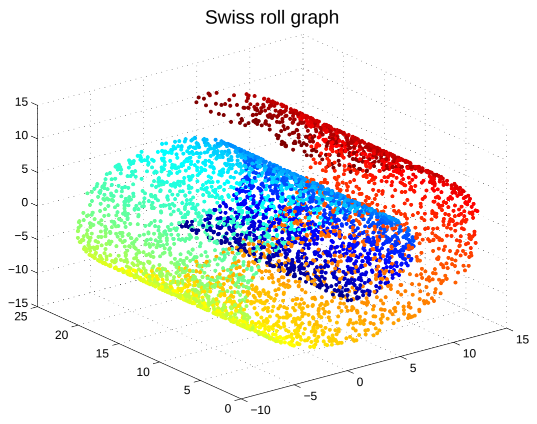


Figure 1.6: Swiss roll - an example for a synthetic low-dimensional manifold

> If we wish to classify, for example, the inputs represented by a red point and a blue point, an algorithm which is based on Euclidean distances might assign both points to the same class, since the Euclidean distance between them is small. However, the distance over the manifold, which is expressed by the number of points separating between them, is very large. This is the intrinsic distance, which should be used for classification.

Some of the methods described before use dimensionality reduction or non-linear classification, but do not use manifold learning. Because of the assumption that the feature vectors lie on a non-linear manifold, we claim that manifold learning is a natural way to learn the intrinsic geometry of the feature vectors. This assumption can only be tested empirically, by comparing to other dimensionality reduction techniques and to other classification methods. In this work we aim to learn the shape of the manifold in order to classify the data based on the intrinsic distances between the feature vectors, and in order to reduce the dimensionality of the problem, if possible.

Manifold learning techniques aim to discover the non-linear nature of the manifold on which the data lies, in order to characterize it better. We use a technique called “Diffusion Maps”, which maps the connections on the manifold to Euclidean distances, leading to an efficient classification based on Euclidean distances. The “Diffusion Maps” technique leverages the relationship between the diffusion operator on the manifold and a Markov transition matrix operating on functions defined on the graph whose vertices were sampled from the manifold.

1.3 Overview of the thesis

This thesis deals with two problems in the field of audio signal analysis, and for both problems we develop algorithms which map the audio data to a space in which it is more meaningful. The first problem is transcription of polyphonic music, and the second is classification of audio and speech data.

For dealing with the transcription problem, we develop a transcription method based on sparse representations with a parametric dictionary. The structure of the dictionary is suitable for the spectra of musical signals and its learned parameters are adaptive to the *timbre* of the signal. This method can be viewed as an unsupervised learning system for identification of notes of a musical polyphonic piece. It improves former classification techniques based on sparse representations in two manners:

1. Better representation of polyphonic music, leading to better transcription - does not require a diverse or a large data set in order to avoid over-fitting, and adjusts to the timbre of the signal.

2. Lower computational complexity and lower sample complexity, due to reduction of the number of learned parameters.

By learning the timbre of the signal, our method avoids the inaccurate assumption which some former works [17, 52] rely on, that the spectral shape of the harmonics is constant.

We describe the motivation for developing our algorithm and its details, examine its performance, compare it to other transcription methods and show its advantages theoretically and empirically.

For solving the classification problem, we add a manifold learning technique stage named "diffusion maps" (DM) [22] to traditional classification techniques, for non linear mapping of the feature vectors. The motivation for this utilization is the assumption that feature vectors of natural audio and speech data lie on a low-dimensional, non-linear manifold. We prove theoretically that the method of DM provides an approximated equivalence between "diffusion distances" between the feature vectors, to Euclidean distances between their representations after the mapping.

We examine empirically the classification of music by genre, and the identification of unvoiced fricative phonemes with diffusion maps. We compare the results to classification without diffusion maps, to other classification methods and to other dimensionality reduction techniques.

1.4 Organization

The organization of the thesis is as follows. In Chapter 2, we explain the concept of sparse representations, and describe existing related methods. In Chapter 3, we present our algorithm for transcription of polyphonic music by sparse representations, and examine its performance empirically. In Chapter 4, we describe manifold learning techniques. In Chapter 5, we present the utilization of diffusion maps as an intermediate stage for classification of audio and speech data, and in Chapter 6, we summarize the work and provide directions for future research.

Chapter 2

Sparse representations

2.1 Introduction

Sparse representations is a way to represent a signal as a linear combination of a small number of elementary signals called atoms. These atoms are arranged as columns of a dictionary, and the linear combination of the small number of atoms is defined by multiplying the dictionary by a *sparse* vector, i.e., a vector whose number of non-zero elements is very small compared to the number of zero elements. Often, the dictionary is over-complete, such that the number of atoms exceeds the dimension of the signal space, so that any signal can be represented by more than one combination of different atoms.

The applications of sparse representations include:

- Compressed Sensing [35] - a sampling method which was developed in the last few years. According to this method, if a signal has a representation which is sparse enough, it can be reconstructed perfectly using a very low sampling rate, significantly lower than that imposed by the Nyquist/Shannon theorem.

- Noise reduction - when allowing a certain error for the sparse representation of a signal, this can be very useful for denoising the signal. For example, see Elad & Aharon's work [38], where zero-mean white and homogeneous Gaussian additive noise was removed from a given image by sparse representations. They showed that even when a dictionary is trained over the noisy image, the denoising performs very well.

- Compression - if we save only the non-zero elements, their locations in the sparse vector, and the transformation which the dictionary represents, then we consume a large amount of storage space, compared to saving the signal itself.

- Feature extraction and pattern classification - the linear combination of atoms which is specific for each signal can characterize features of the signal, and so can be used efficiently for classification, such as in [70], where sparse representations were used to classify musical pieces by their genre.

- Blind source separation - when each source is represented by a single atom, the sparse representations can be used for their extraction and identification. Transcription of polyphonic music is a particular case of this problem, in which we aim to separate single notes from a mix of notes.

Sparse representation of a signal y is formulated in the problem denoted by P_0:

$$(P_0) : \min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{subject to} \quad \mathbf{A}\mathbf{x} = \mathbf{y}. \tag{2.1}$$

We actually search for the sparsest vector $\mathbf{x}$ which satisfies $\mathbf{A}\mathbf{x} = \mathbf{y}$. $\mathbf{A} \in \mathbb{R}^{n \times m}$ is the dictionary composed of the *atoms* as its columns, and $\mathbf{x} \in \mathbb{R}^m$ is coding the linear combination of the atoms from the dictionary. This problem is non-convex, therefore uniqueness is not automatically guaranteed. Solving it entails a combinatorial search, making this problem NP-hard. Therefore, its solution has to be approximated by a pursuit algorithm. There are several methods for approximating the solution to the P_0 problem, and they will be described in the next section.

The chapter is organized as follows. In Section 2.2, several approximations to the solution of the P_0 problem are described, in Section 2.3, the choice of the dictionary matrix is explained, in Section 2.4, an existing method of transcription of polyphonic music using sparse representations is described, followed by a summary in Section 2.5.

2.2 Approximation of the P_0 problem

Pursuit algorithms for solving the P_0 problem can be divided into two main groups - Greedy algorithms and Convex relaxation techniques. We will briefly describe each of these groups.

2.2.1 Greedy algorithms

The logic behind these kind of algorithms, is that once the support of the solution is known, its non-zero values can be easily found by solving a least-squares problem (this is the "oracle" problem). These algorithms operate in an iterative manner, where the initial support of the sparse vector $\mathbf{x}$ is empty ($\mathbf{x}_{(0)}$ is a vector of zeros), and at each iteration one more element is added to the support, i.e., another atom from the dictionary matrix is added to the representation. The algorithm will stop at the k_{th} iteration, when the ℓ_2 norm of the error - $\|\mathbf{y} - \mathbf{A}\mathbf{x}_{(k)}\|$ falls below a specific threshold.

There are several greedy methods including - Least-Squares Orthogonal Matching Pursuit (LS-OMP), Orthogonal Matching Pursuit (OMP), Matching Pursuit (MP), Weak-MP and Thresholding [37, 87]. Each one satisfies a certain compromise between accuracy and complexity. A good compromise between both is settled in the Orthogonal Matching Pursuit (OMP) algorithm. The algorithm is described in Table 2.1.

More details on this algorithm and on other greedy algorithms can be found in [37].

2.2.2 Convex relaxation techniques

In order to render the problem (P_0) to a problem which is easier to solve, a regularization of the ℓ_0 norm can be applied by replacing it with a continuous or even smooth approximation. Examples of such approximations can be ℓ_p norms with $p \in (0, 1]$ or even by smooth functions as $\sum_{i=1}^{m} 1 - e^{-\alpha x_i^2}$ or $\sum_{i=1}^{m} x_j^2/(\alpha + x_j^2)$. One has to keep in mind that such ℓ_p for $p < 1$ are no longer formal norms since they do not satisfy the triangle inequality. Nevertheless, we shall use the term norm for these functions as well.

Replacing the ℓ_0 norm by an ℓ_p norm with $p \in (0, 1]$, leads to the relaxed problem P_p:

$$(P_p) : \min_{\mathbf{x}} \|\mathbf{x}\|_p \quad \text{subject to} \quad \|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2 = 0. \tag{2.2}$$

Promotion of sparse solutions by minimizing the ℓ_p norm with $p \leq 1$ can be explained by examining from a geometric point of view the problem P_p. The solution to this problem can be approximately found by "blowing" an ℓ_p balloon centered around the origin, and stopping its inflation when it first touches the feasible set, which is a hyperplane defined by $\mathbf{A}\mathbf{x} = \mathbf{y}$. When $p \leq 1$, the intersection takes place at a corner of the ball, leading to

Table 2.1: Orthogonal Matching Pursuit - a greedy algorithm for approximating the solution of (P_0)

Task: Approximate the solution of (P_0): $\min_{\mathbf{x}} \|\mathbf{x}\|_0$ subject to $\mathbf{Ax} = \mathbf{y}$.

Parameters: We are given the matrix $\mathbf{A}$, the signal $\mathbf{y}$ and the error threshold ε_0.

Initialization: Initialize $k = 0$, and set

- The initial solution $\mathbf{x}_{(0)} = 0$.
- The initial residual $\mathbf{r}_{(0)} = \mathbf{y} - \mathbf{Ax}_{(0)} = \mathbf{y}$.
- The initial support of the solution $\mathrm{S}_{(0)} = Support\{\mathbf{x}_{(0)}\} = \emptyset$.

Main iteration: Increment k by 1, and apply the following steps:

- **Sweep:** Compute the errors $\varepsilon(j) = \min_{z_j} \|\mathbf{a}_j z_j - \mathbf{r}_{(k-1)}\|_2^2$ for all j using the optimal choice $z_j^* = \mathbf{a}_j^T \mathbf{r}_{(k-1)} / \|\mathbf{a}_j\|_2^2$.
- **Update Support:** Find a minimizer j_0 of $\varepsilon(j) : \forall j \notin \mathrm{S}_{(k-1)}, \varepsilon(j_0) \leq \varepsilon(j)$, and update $\mathrm{S}_{(k)} = \mathrm{S}_{(k-1)} \cup \{j_0\}$.
- **Update Provisional Solution:** Compute $\mathbf{x}_{(k)}$, the minimizer of $\|\mathbf{Ax} - \mathbf{y}\|_2^2$ subject to $Support\{\mathbf{x}\} = \mathrm{S}_{(k)}$.
- **Update Residual:** Compute $\mathbf{r}_{(k)} = \mathbf{y} - \mathbf{Ax}_{(k)}$.
- **Stopping Rule:** If $\|\mathbf{r}_{(k)}\|_2 < \varepsilon_0$, stop. Otherwise, apply another iteration.

Output: The Proposed solution is $\mathbf{x}_{(k)}$ obtained after k iterations.

a sparse solution. When $p > 1$, the intersection doesn't take place a corner, leading to non-sparse solutions. Figure 2.1 demonstrates in two dimensions some intersections of ℓ_p balloons with a hyperplane $\mathbf{Ax} = \mathbf{y}$.

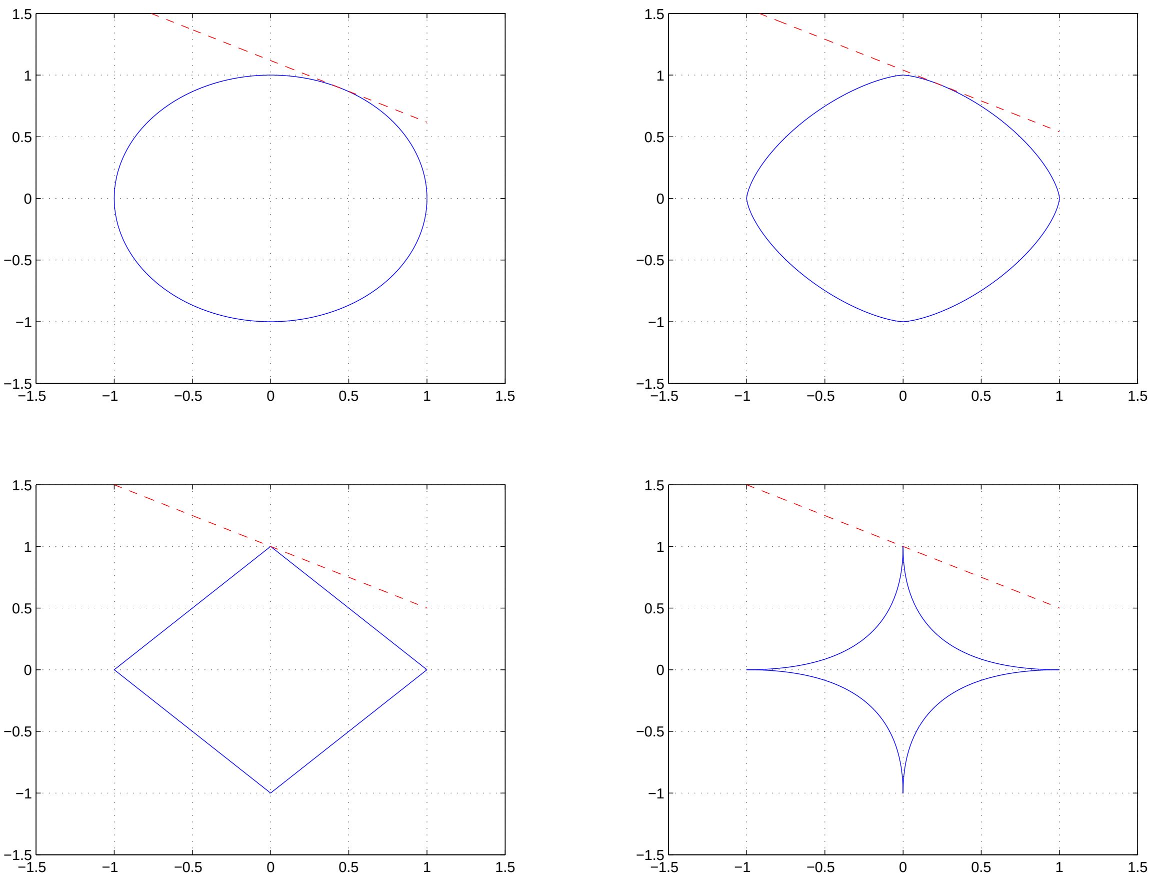


Figure 2.1: The intersections between different ℓ_p balls (solid blue line) and the set $\mathbf{Ax} = \mathbf{y}$ (dashed red line), demonstrated in 2D, for $p = 2$ (top left), $p = 1.5$ (top right), $p = 1$ (bottom left) and $p = 0.5$ (bottom right). When $p \leq 1$, the intersection takes place at a corner of the ball, leading to a sparse solution.

An algorithm developed by Gorodinsky and Rao, named FOCUSS (FOcal Underdetermined System Solver) [44], allows solving the problem in an iterative way, with ℓ_p where $0 < p \leq 1$, by representing the ℓ_p norm as a weighted ℓ_2 norm. Another popular strategy is to replace the ℓ_0 norm by the ℓ_1 norm. This is the best convex approximate, and it leads to the *basis pursuit* algorithm [19], which uses linear programming for approximating the solution of P_0.

An error-tolerant version of P_0 which is more suitable for real-world problems and which can be used, e.g., for denoising, is as follows:

$$(P_0^{\varepsilon}) : \min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{subject to} \quad \|\mathbf{A}\mathbf{x}-\mathbf{y}\|_2 \leq \varepsilon, \tag{2.3}$$

Or in a modified version:

$$\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x}-\mathbf{y}\|_2^2 \quad \text{subject to} \quad \|\mathbf{x}\|_0 < K. \tag{2.4}$$

Since the methods mentioned before are approximations of the real solution, it is interesting to ask which are the conditions that guarantee success, i.e. that guarantee finding the correct solution. Elad proves in [37] that theoretically, success is guaranteed only when the solution is extremely sparse, which is a situation that doesn't occur in many cases. However, he also shows empirically that in most cases theory is far from practice, and in many problems of sparse representations the correct solution is found although the required bound on the sparsity is not obtained. Therefore we might expect good identification results for our problem as well.

2.3 Choosing a dictionary

The sparse coding algorithms described in Section 2.1 do not deal with the problem of choosing the dictionary matrix **A**. The dictionary can be chosen in three different manners:

1. Analytic dictionaries.
2. Learned dictionaries.
3. Parametric dictionaries.

We shall describe here each one of the choices.

2.3.1 Analytic dictionaries

There are several transforms which when applied on natural signals, specifically images, lead to a sparse result. Such transforms are the Curvelet transform [15,16], the Contourlet transform [33,34] the Short time Fourier transform or the Wavelets transform [57]. Therefore, dictionaries which constitute an inverse (or pseudo-inverse) transform matrix of one of the mentioned transforms, would probably be a good choice to sparsely represent natural signals.

The advantage of using such a pre-defined dictionary, is that then the sparse coding algorithm is fast and doesn't require the computation of the dictionary. However, predefined dictionaries are limited in their accuracy in representing the data, because they are not data-driven, and they are based on generic mathematical models.

2.3.2 Dictionary learning

The growing interest in *machine learning* techniques first led to the idea of learning dictionaries by Olshausen and Field in 1996 [68]. They claimed and showed empirically, that learning a dictionary from natural images leads to atoms which are very similar to the receptive fields of simple cells in the visual cortex. They suggested adding to the sparse coding algorithm a stage of dictionary learning, in an iterative and alternative manner.

In the P_0 problem mentioned before, we discussed the sparse representation of a single signal $\mathbf{y}$. We would like to expand our problem of interest to the case of a *set* of signals or samples $\mathbf{y}_{i,\{1\leq i\leq M\}}$, arranged as columns of a matrix $\mathbf{Y}$. In order to represent them we obtain a *set* of sparse vectors $\mathbf{x}_{i,\{1\leq i\leq M\}}$ which are arranged as columns of a matrix $\mathbf{X}$. Formally, the optimization problem P_0^ε turns into:

$$\min_{\{\mathbf{x_i}\}_{i=1}^M} \sum_{i=1}^{M} \|\mathbf{x}_i\|_0 \quad \text{subject to} \quad \|\mathbf{A}\mathbf{x}_i - \mathbf{y}_i\|_2 < \varepsilon, \quad 1 \leq i \leq M. \tag{2.5}$$

Or in its modified version:

$$\min_{\{\mathbf{x_i}\}_{i=1}^M} \sum_{i=1}^{M} \|\mathbf{A}\mathbf{x}_i - \mathbf{y}_i\|_2^2 \quad \text{subject to} \quad \|\mathbf{x}_i\|_0 < K, \quad 1 \leq i \leq M. \tag{2.6}$$

Learning the dictionary $\mathbf{A}$ means adding the term $\mathbf{A}$ to the argument of the optimization:

$$\min_{\mathbf{A},\{\mathbf{x_i}\}_{i=1}^{M}} \sum_{i=1}^{M} \|\mathbf{x}_i\|_0 \quad \text{subject to} \quad \|\mathbf{A}\mathbf{x}_i - \mathbf{y}_i\|_2 < \varepsilon, \quad 1 \leq i \leq M. \tag{2.7}$$

Or:

$$\min_{\mathbf{A},\mathbf{X}} \|\mathbf{Y} - \mathbf{A}\mathbf{X}\|_F^2 \quad \text{subject to} \quad \|\mathbf{x}_i\|_0 < K, \quad 1 \leq i \leq M. \tag{2.8}$$

The first term in (2.8) is minimization of the Frobenius norm and it equals to the first term in (2.6). The solution to the problems (2.7), (2.8) can be approximated by *dictionary learning* algorithms. Such algorithms are iterative and consist of two stages - a sparse coding stage and a dictionary update stage. The advantage of learned dictionaries is that they are data-driven and therefore represent the data well. Their disadvantages are high computational complexity, slower running time compared to sparse coding with analytic dictionaries, and a requirement for a large data set in order to avoid over-fitting. We shall describe in the next subsections two main algorithms for dictionary learning - Method of Optimal Directions (MOD) [39] and K-SVD [4].

The MOD Algorithm

In this algorithm, developed by Engan et al. [39], the dictionary update stage is conducted using least squares:

$$\mathbf{A}_{(k)} = \underset{\mathbf{A}}{\operatorname{argmin}} \|\mathbf{Y} - \mathbf{A}\mathbf{X}_{(k)}\|_F^2 = \mathbf{Y}\mathbf{X}^{\dagger} \tag{2.9}$$

where $\mathbf{X}^{\dagger}$ is the Moore-Penrose pseudo-inverse of $\mathbf{X}$. After an initialization of the dictionary matrix, the matrix of sparse columns $\mathbf{X}$ and the dictionary matrix $\mathbf{A}$ are updated alternately at each iteration, until the change at the k_{th} iteration of $\|\mathbf{Y} - \mathbf{A_{(k)}}\mathbf{X_{(k)}}\|_F^2$ is small enough. The algorithm is described in Table 2.2.

Table 2.2: The MOD dictionary-learning algorithm

Task: Train a dictionary $\mathbf{A}$ to sparsely represent the data $\{\mathbf{y}_i\}_{i=1}^{M}$,
by approximating the solution to the problem posed in (2.8) .

Initialization: Initialize $k = 0$, and

- **Initialize Dictionary:** Build $\mathbf{A}_{(0)} \in \mathbb{R}^{n\times m}$,either by using random entries, or by using m randomly chosen examples.
- **Normalization:** Normalize the columns of $\mathbf{A}_{(0)}$.

Main iteration: Increment k by 1, and apply

- **Sparse Coding Stage:** Use a pursuit algorithm to approximate the solution of $\widehat{\mathbf{x}}_i = \underset{\mathbf{x}}{\operatorname{argmin}} \|\mathbf{y}_i - \mathbf{A}_{(k-1)}\mathbf{x}\|_2^2$ subject to $\|\mathbf{x}\|_0 \leq K$.
  obtaining sparse representations $\widehat{\mathbf{x}}_i$ for $1 \leq i \leq M$. These form the matrix $\mathbf{X}_{(k)}$.
- **MOD Dictionary Update Stage:** Update the dictionary by the formula $\mathbf{A}_{(k)} = \underset{\mathbf{A}}{\operatorname{argmin}} \|\mathbf{Y} - \mathbf{A}\mathbf{X}_{(\mathbf{k})}\|_F^2 = \mathbf{Y}\mathbf{X}^{\dagger}$.
- **Stopping Rule:** If the change in $\|\mathbf{Y} - \mathbf{A}_{(\mathbf{k})}\mathbf{X}_{(\mathbf{k})}\|_F^2$ is small enough, stop. Otherwise, apply another iteration.

Output: The desired results are $\mathbf{A}_{(k)}$ and $\mathbf{X}_{(k)}$.

The normalization stage of the atoms in the MOD (and in the K-SVD) is intended to make the implementation of the sparse coding stage simpler when using a greedy method, and it does not change its solution.

The K-SVD Algorithm

A different update rule for the dictionary was proposed by Aharon et. al., leading to the K-SVD algorithm [4]. In this algorithm, the atoms (i.e., columns) in the dictionary $\mathbf{A}$ are handled sequentially. The dependency on the atom $\mathbf{a}_{j_0}$ in (2.8) is isolated by rewriting the term $\|\mathbf{Y}-\mathbf{AX}\|_F^2$:

$$\|\mathbf{Y}-\mathbf{AX}\|_F^2 = \left\|\mathbf{Y}-\sum_{j=1}^{m}\mathbf{a}_j\mathbf{x}_j^T\right\|_F^2 = \left\|\left(\mathbf{Y}-\sum_{j\neq j_0}\mathbf{a}_j\mathbf{x}_j^T\right)-\mathbf{a}_{j_0}\mathbf{x}_{j_0}^T\right\|_F^2 . \tag{2.10}$$

In this description $\mathbf{x}_j^T$ stands for the j-th *row* of $\mathbf{X}$, i.e., the coefficients which correspond to the j_{th} atom. We define the term

$$\mathbf{E}_{j_0} = \mathbf{Y}-\sum_{j\neq j_0}\mathbf{a}_j\mathbf{x}_j^T \tag{2.11}$$

as the error matrix corresponding to the atom $\mathbf{a}_{j_0}$. We restrict it only to the columns that correspond to the signals (columns) in $\mathbf{Y}$ which use the atom $\mathbf{a}_{j_0}$, and denote the restricted error matrix as $\mathbf{E}_{j_0}^R$. Both $\mathbf{a}_{j_0}$ and the non zero elements in $\mathbf{x}_{j_0}^T$, which are denoted by $(\mathbf{x}_{j_0}^R)^T$, are updated in this algorithm, by minimizing the term in (2.10), using a rank-1 approximation of the error matrix $\mathbf{E}_{j_0}^R$. This approximation is obtained via singular value decomposition (SVD).

The name K-SVD stems from the similarity to the K-Means algorithm. K-Means can be considered as a particular case of K-SVD, in which $K = 1$ (The cardinality of the sparse vectors), and the representation coefficients are constrained to be binary (1 or 0). Then the problem reduces to a simple clustering task. The K-SVD algorithm is described in Table 2.3.

2.3.3 Parametric dictionaries

The idea of parametric dictionaries is a compromise between pre-defined analytic dictionaries to data-driven learned dictionaries. It is more data-driven than analytic dictionaries, and more computationally efficient than explicit learned dictionaries.

Table 2.3: The K-SVD dictionary-learning algorithm

Task: Train a dictionary $\mathbf{A}$ to sparsely represent the data $\{\mathbf{y}_i\}_{i=1}^{M}$,
by approximating the solution to the problem posed in (2.8) .

Initialization: Initialize $k = 0$, and

- **Initialize Dictionary:** Build $\mathbf{A}_{(0)} \in \mathbb{R}^{n\times m}$,either by using random entries, or by using m randomly chosen examples.
- **Normalization:** Normalize the columns of $\mathbf{A}_{(0)}$.

Main iteration: Increment k by 1, and apply

- **Sparse Coding Stage:** Use a pursuit algorithm to approximate the solution of
  $\widehat{\mathbf{x}_i} = \underset{\mathbf{x}}{\text{argmin}} \|\mathbf{y}_i - \mathbf{A}_{(k-1)}\mathbf{x}\|_2^2$ subject to $\|\mathbf{x}\|_0 \leq K$.
  obtaining sparse representations $\widehat{\mathbf{x}}_i$ for $1 \leq i \leq M$. These form the matrix $\mathbf{X}_{(k)}$.
- **K-SVD Dictionary Update Stage:** Update each atom $\mathbf{a}_{j_0,\{j_0=1,2,\ldots,m\}}$
  in the dictionary matrix by rank-1 approximation of its error matrix $\mathbf{E}_{j_0}^{R}$, using SVD.
  $\mathbf{E}_{j_0}^{R}$ is the restriction of the matrix $\mathbf{E}_{j_0} = \mathbf{Y} - \sum_{j\neq j_0} \mathbf{a}_j\mathbf{x}_j^T$ to the columns that correspond to the samples in $\mathbf{Y}$ which use the atom $\mathbf{a}_{j_0}$.
  After applying the SVD $\mathbf{E}_{j_0}^{R} = \mathbf{U\Delta V^T}$, update the dictionary atom by $\mathbf{a}_{j_0} = \mathbf{u}_1$
  and the non zero elements of the representation coefficients by $(\mathbf{x}_{j_0}^{R})^T = (\mathbf{\Delta}[1,1]\mathbf{v}_1)^T$.
- **Stopping Rule:** If the change in $\|\mathbf{Y} - \mathbf{A_{(k)}X_{(k)}}\|_F^2$ is small enough,
  stop. Otherwise, apply another iteration.

Output: The desired results are $\mathbf{A}_{(k)}$ and $\mathbf{X}_{(k)}$.

Examples for parametric dictionaries include the double-sparsity model [76], in which the dictionary is imposed to be sparse as well, based on the observation that most often, dictionaries which represent natural images are sparse under certain bases. Another example is the image signature dictionary [3], in which the dictionary $\mathbf{A} \in \mathbb{R}^{n\times m}$ is composed of *patches* of a small dictionary (the "signature" dictionary) $\mathbf{a}_0 \in \mathbb{R}^{\sqrt{m}\times\sqrt{m}}$, which is trained on the image, saving the computational complexity of training the big dictionary.

The complexity of parametric dictionaries is reduced compared to that of explicit dictionaries, because it is proportional to the number of coefficients to be learned.

2.4 Transcription of polyphonic music by sparse representations

Abdallah and Plumbley [1] were the first to use sparse representations for transcription of polyphonic music. They divided the music signal into 46 ms time-windows with an overlap of 50% , applied Fourier transform and took the magnitude, based on the idea that power spectra of different notes approximately add, assuming random phase relationships. They aimed to deduce the magnitude into magnitude spectrums of individual notes by applying MOD dictionary learning, where the solution of the sparse coding problem was approximated using basis pursuit. Their probabilistic motivation for this is as follows:

Each sample $\mathbf{y}$ from the signals we examine, can be written as

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{e} \tag{2.12}$$

where $\mathbf{e}$ is a zero-mean Gaussian random noise. The components of $\mathbf{x}$ are statistically independent, implying

$$p(\mathbf{x}) = \prod_{j=1}^{m} p(x_j). \tag{2.13}$$

The Gaussian model implies that the conditional density $p(\mathbf{y}|\mathbf{x}, \mathbf{A})$ is

$$p(\mathbf{y}|\mathbf{x}, \mathbf{A}) = [\frac{det\Lambda_{\mathbf{e}}}{(2\pi)^n}]exp(-\frac{1}{2}\mathbf{e}^T\Lambda_{\mathbf{e}}\mathbf{e}) \tag{2.14}$$

where $\mathbf{e} = \mathbf{y} - \mathbf{Ax}$ and $\Lambda_{\mathbf{e}} = E\{\mathbf{ee}^T\}^{-1}$. The MAP estimate for the sparse coding is $\widehat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmax}}\{p(\mathbf{x}|\mathbf{y}, \mathbf{A})\}$, or equivalently, from Bayes' Theorem

$$\widehat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}}\{-\log p(\mathbf{y}|\mathbf{x}, \mathbf{A}) - \log p(\mathbf{x})\} \tag{2.15}$$

which under the assumptions of the model, becomes:

$$\widehat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}}\{\frac{1}{2}\lambda\|\mathbf{y} - \mathbf{Ax}\|_2^2 - \sum_{j=1}^{m} \log p(x_j)\}. \tag{2.16}$$

The second term in (2.16) becomes the ℓ_1 norm $-\|\mathbf{x}\|_1$, if we use a Laplacian prior $p(\mathbf{x_j}) = \frac{1}{2}exp(-|x_j|)$, which leads to the basis pursuit relaxation. The approximation $\widehat{\mathbf{A}} = \underset{\mathbf{A}}{\operatorname{argmin}}\|\mathbf{y} - \mathbf{Ax}\|_2^2$ stems from the same MAP estimation that led to (2.16):

$$\widehat{\mathbf{A}} = \underset{\mathbf{A}}{\operatorname{argmin}}\{\frac{1}{2}\lambda\|\mathbf{y} - \mathbf{Ax}\|_2^2 - \sum_{j=1}^{m} \log p(x_j)\}, \tag{2.17}$$

but now the second term is irrelevant.

We show in Chapter 3 that transcription of polyphonic music can be applied as well by K-SVD dictionary learning.

2.5 Summary

We have presented in this chapter the idea of sparse representations. It consist of two stages - the first is a sparse coding stage, i.e., approximation of the solution to the problem P_0, which can be solved by greedy methods or by relaxation methods. The second stage is choosing the dictionary, which can be pre-defined, fully trained or partially trained if the dictionary is parametric.

The advantages of parametric dictionaries are low computational complexity and faster convergence compared to explicit dictionaries, as well as better representation of the signal compared to analytic dictionaries. In Chapter 3 we develop a parametric dictionary which is suitable for the spectra of music signals, and has further advantages over explicit and implicit dictionaries when it comes to transcription of polyphonic music. We also offer modifications of existing dictionary learning schemes for the learning of this parametric dictionary.

Chapter 3

Sparse representations of music signals

3.1 Introduction

Transcription of music is defined as the process of identifying the parameters of an acoustic musical signal, which are required in order to write down the score sheet of the notes [59]. As mentioned in Chapter 1, we are interested in this work only in finding the *pitch* of the notes.

Harmonic instruments act as periodic oscillators, leading to vibrations in a fundamental frequency, and in its integer multiples, which are called *harmonics*. In polyphonic music, especially in western music, several harmonics of different sounds tend to overlap. This causes a difficulty in assigning the harmonics to their true fundamental frequencies, and in deducing spectra of several sounds from their sum [50, 51]. Therefore transcription of polyphonic music is considered a hard problem, which hasn't received a satisfying solution.

Algorithms that aim to solve this problem can be divided into three main groups: time-based, frequency-based and time-frequency-based algorithms.

The idea of using *sparse representations* as a time-based or frequency-based method for transcription of polyphonic music, was first suggested by Abdallah and Plumbley [2]. It was later improved and expanded [1, 71, 72], and inspired other works, which used non-negative matrix factorization [83], Independent Subspace Analysis (ISA) [89] and sparse coding [90].

In played music, only a small number of notes is played simultaneously compared to the number of notes available. This is the motivation for transcription of music signals by sparse representations. In the time domain, sparse representations is applied by modeling each sample as a sum of few scaled and shifted versions of some underlying functions. In the frequency domain, the approach is based on the idea that power spectra of different notes approximately add, assuming random phase relationships. Then the representation is applied for the magnitude of the spectrum.

There are some drawbacks and insufficiencies in the existing methods which were described in Chapter 1 and 2. In methods based on sparse representations with an explicit dictionary, such as that of Abdallah and Plumbley [1], learning the dictionary of underlying functions is based on the expectation that each atom (column) in the dictionary would represent a single note. Yet, unless restricted, this will not always be the case. Only if the data set of notes is large, and each note is played alone at least once in the transcribed musical piece, the dictionary can be learned properly. Otherwise, there would be an over-fitting to the data. If, for example, a certain note is played only as part of a chord, an atom that represents this individual note will not be learned, but only an atom that represents the whole chord.

In addition, former methods such as those conducted by Saito et al. [78] and by Klapuri [52], assumed that the spectral shape of the harmonics can be modeled by a constant function, and then deduced spectrums of several notes from their combination based on this assumption. This is very inaccurate, since the spectral shape, which is the *timbre* ('color') of the signal, changes as a function of several factors which were mentioned in Chapter 1.

In this chapter, we develop an algorithm for transcription of polyphonic music, based on sparse representations with a parametric dictionary suitable for the spectra of music signals. Our algorithm overcomes the problems above by learning the spectral shape of the harmonics from the signal, and by imposing some restrictions on the structure of the dictionary, such that it would represent individual notes even if they are not being played individually, in an unsupervised manner.

The chapter is organized as follows. In Section 3.3, our algorithm is described and examined, in Section 3.4, simulation results are presented, and in Section 3.5 we conclude the chapter.

3.2 The Overall algorithm

In Chapter 2 we described two algorithms for dictionary learning - the method of optimal directions (MOD) and the K-SVD. In addition, we introduced the idea of *parametric dictionaries*, described the motivation behind it, and gave several examples. In this section, we introduce the *Musically-Structured (MS) dictionary* for transcription of polyphonic music. This is a parametric dictionary which is suitable for the spectra of music.

The overall transcription algorithm which we offer is as follows:

1. Notes onsets detection - we conduct this stage manually, or extract the onsets from a MIDI file in the case of transcribing a synthesized MIDI musical piece.

2. After 32 ms - evaluation of the number of notes in a 64 ms time window - either manually or from a MIDI file. This number is defined as K and is used as the maximal cardinality of the sparse vector.

3. Constant Q transform is applied on the signal in the 64 ms time-window.

4. All the vectors of CQTs of the time-windows mentioned before are concatenated as columns to generate the matrix $\mathbf{Y}$.

5. A Musically-Structured dictionary learning algorithm is applied on the matrix $\mathbf{Y}$ to transcribe the music in each of the time-windows represented by its columns.

The reason for applying the transcription only on 64 ms time-windows 32 ms after the onsets of the notes, is that the acoustic spectrum of a tone changes significantly as a function of the stage in the ADSR envelope. We wish to sample all notes at the same stage, such that the atoms in the dictionary would represent them well. Costantini [25,26] et al. followed the same logic when applying their transcription method. We assume that after 32 ms the ADSR envelope is in its *sustained* stage, which is the most suitable stage for the spectral analysis because of its stability and relative long duration.

As mentioned, we apply the Constant Q transform (CQT) [12] as the spectral transformation, and not the Short time Fourier transform (STFT) as Abdallah and Plumbley [1] did, since it is more suitable to the auditory system. The STFT can be modeled as a bank of linearly spaced, consecutive non-overlapping filters, whose band-width is constant. Formally, for the k_{th} frequency band:

$$X[k] = \sum_{n=0}^{N-1} W[n]x[n]e^{\frac{-j2\pi kn}{N}}$$

where $W[n]$ is a certain window. On the contrary, the CQT can be modeled as a bank of *logarithmically* spaced, consecutive non-overlapping filters, whose band-width is growing logarithmically as a function of frequency. Formally:

$$X[k] = \frac{1}{N[k]} \sum_{n=0}^{N[k]-1} W[k,n]x[n]e^{\frac{-j2\pi Qn}{N[k]}}.$$

Q, the "quality factor", is a constant which equals the center frequency of the k_{th} bin f_k, divided by the k_{th} filter width δf_k:

$$Q = \frac{f_k}{\delta f_k}.$$

The window $W[n,k]$ is wider as the frequency grows and so is the window length $N[k]$, where $N[k]$ equals the sampling period divided by the k_{th} filter width. The relation between two consecutive filter widths is $\delta f_k = 2^{\frac{1}{n}} \cdot \delta f_{k-1}$, where n is the number of filters per octave. If we are interested, e.g., in a difference of a semitone, which is 100 cents, between two consecutive center frequencies, then we choose $n = 12$. This transform is suitable for the auditory system, since the latter can be modeled as a logarithmically-spaced filter bank itself [12]. The CQT also allows using less frequency bins by exploiting the human auditory system, thus reducing the computational complexity of the algorithm.

A block diagram of the overall algorithm is presented in Figure 3.1.

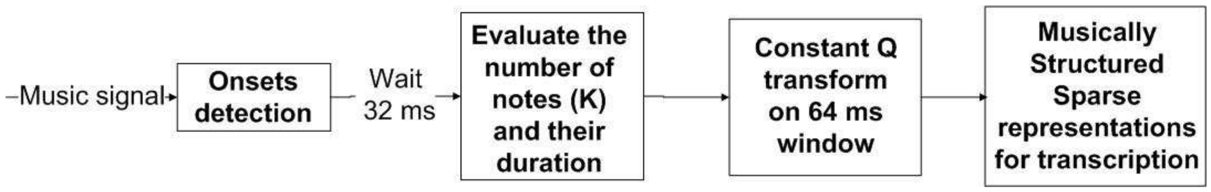

Music signal
Onsets detection
Wait 32 ms
Evaluate the number of notes (K) and their duration
Constant Q transform on 64 ms window
Musically Structured Sparse representations for transcription

Figure 3.1: The block diagram of the overall algorithm

We now turn to describe block of the sparse representations in the diagram.

3.3 Musically-Structured (MS) dictionary learning algorithm

3.3.1 Sparse coding

As described in Chapter 2, dictionary learning algorithms contain a stage of sparse coding. Sparse coding is a way to represent a signal such that most of the coefficients of the representation are zero. This method was proved to capture the meaningful characteristics of signals such as images and audio - an important property which allows using it for various tasks in signal processing, such as recognition, denoising and compression. We recall here the problem P_0

$$(P_0) : \min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{subject to} \quad \mathbf{A}\mathbf{x} = \mathbf{y}, \tag{3.1}$$

and its error-tolerant modified version for multiple signals:

$$\min_{\mathbf{X}} \|\mathbf{Y} - \mathbf{A}\mathbf{X}\|_F^2 \quad \text{subject to} \quad \|\mathbf{x}_i\|_0 < K, \quad 1 \leq i \leq M, \tag{3.2}$$

where $\mathbf{x}_i$ are the columns of $\mathbf{X}$. In our case, $\mathbf{Y}$ is a matrix whose columns are the CQTs of 64 ms time-windows, 32 ms after the onsets in a musical piece. $\mathbf{A}$ is the Musically-Structured dictionary matrix in which each atom represents a note. Each column in $\mathbf{X}$ encodes a linear combination of the notes from $\mathbf{A}$ which are played in the time-window which corresponds to the corresponding column in $\mathbf{Y}$.

In our algorithm we use a greedy method for sparse coding, since it allows to pre-define the cardinality (the number of non-zero elements) of the sparse vector, according to the evaluated number of notes at each time window. Specifically, we choose to use the OMP algorithm, which substitutes a good compromise between complexity and performance.

3.3.2 Musically-Structured (MS) dictionary

In order to develop a parametric dictionary which is suitable for the spectrum of music signals, we examine the spectrum of a musical note. If we examine, e.g, the short-time Fourier transform (STFT) of a A4 piano note (Figure 3.2), we can see that it is composed of peaks at the fundamental frequency and at its harmonics (which all together are called the *partials*). The magnitude of the Fourier transform of any note can be modeled by an impulse train which has been multiplied with a shaping filter, that is controlled by the factors mentioned in Chapter 1 - the musical instrument, the intensity, the duration of the note and the fundamental frequency. This is an idealized model, since actually the impulses have a certain non-zero width, because each note is finite in time.

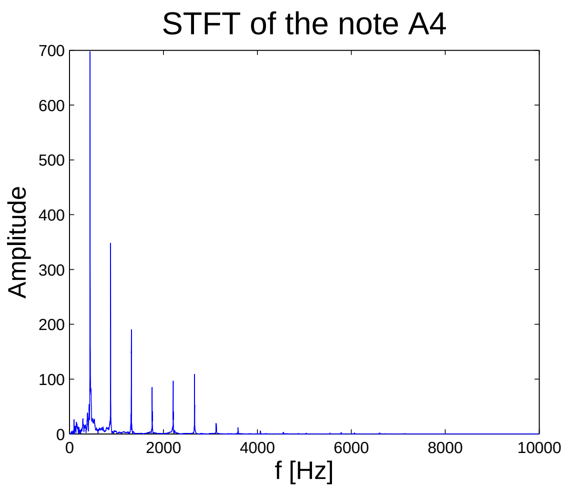


Figure 3.2: The Short Time Fourier Transform of A4 piano note

If we apply the Constant Q transform instead of the STFT on a A4 piano note, we get peaks of the fundamental frequency and its harmonics in gaps which become logarithmically smaller as the CQT bin grows (Figure 3.3).

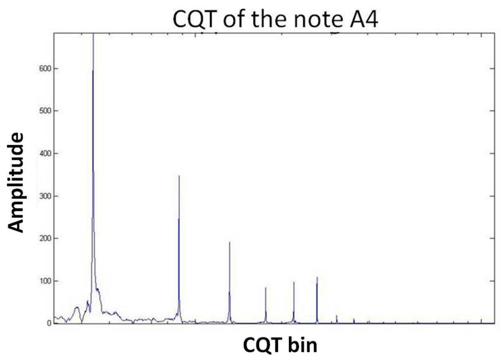


Figure 3.3: The Constant Q Transform of a A4 piano note

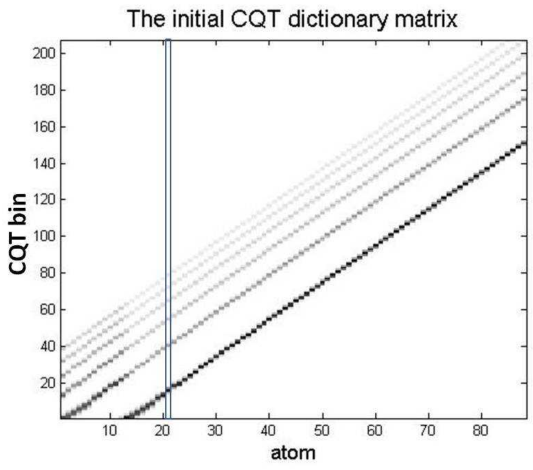
The dictionary which we offer is initialized by the evaluated CQTs of all the notes in the piano (total - 88 notes), where each note is represented by a different atom. More specifically, we initialize each atom by an impulse train of 6 elements, corresponding to the suitable fundamental frequency and its first 5 harmonics. This number of harmonics is a reasonable number for musical instruments [69], and it was chosen after an optimization procedure over several musical pieces. We multiply this finite impulse train by an initial shaping function $f(n) = \frac{1}{n}$, where n is the partial number. This initial shaping function models roughly the distribution of the intensities of the partials (although in some cases the amplitude of a higher harmonic is stronger than that of a lower one). Finally, we map the dictionary to the CQT scale, which is appropriate for the human auditory system and allows reducing the number of frequency bins compared to that of the STFT. We denote this dictionary the initial *Musically-Structured (MS) dictionary.* An image of this initial dictionary, and the CQT of a note represented by a certain atom in it, are presented in Figure 3.4.

In this parametric Musically-Structured dictionary, the *support* of the dictionary is constant, i.e., the location of the non zero elements, which are the CQT bins of the fundamental frequencies and their 5 first harmonics. The entries of the elements in the support are learned, i.e., the amplitudes of the fundamental frequencies and the amplitudes of their harmonics. These amplitudes represent the *timbre* of the signal, which was mentioned before.

The minimal resolution required for music transcription is 12 CQT frequency bins per octave (one for each 100 cent = semitone). However, using a higher resolution improves the transcription results for polyphonic music. In the experiments presented in Section 3.4 we show the results for a mapping to 24 frequency bins per octave (one for each 50 cents = quarter tone). In the case of a resolution of 12 frequency bins per octave, the dictionary is over-complete, and the system of equations $\mathbf{AX} = \mathbf{Y}$ is under-determined.

The advantages of the MS-dictionary are:

1. Avoids over-fitting - Our motivation is identification of the notes. For this task, each atom in the learned dictionary has to represent a single note, even if it doesn't appear individually in the data set (i.e., if it is played only in conjugation with other notes, such as in the case of a chord). Since the support of the dictionary is constant, and since each atom in the initial dictionary represents an individual note, we expect this requirement to be fulfilled.



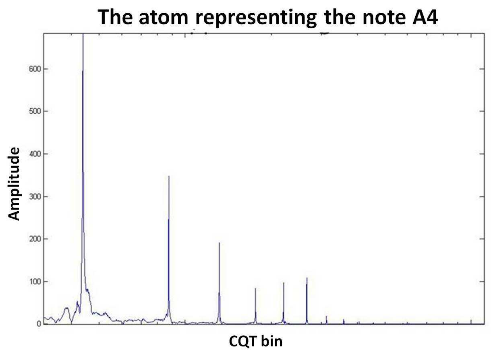


Figure 3.4: The initial MS dictionary (up) and the CQT of a note represented by a certain atom (down). The atom which represents the note in the bottom picture is marked by a rectangle in the dictionary.

2. Better representation of the signal than an analytic dictionary - We allow the entries in the support of the dictionary matrix to be learned according to the timbre of the signal, therefore achieving better representation than when using the initial dictionary as a pre-defined analytic dictionary.

3. Reduced complexity - The complexity of the dictionary is proportional to the number of the learned parameters. In an explicit dictionary, the number of learned parameters equals the size of the dictionary, i.e. $n \cdot m$. In the MS-dictionary, the number of learned parameters is $h \cdot m$, where h is the number of partials (fundamental frequency + harmonics), which we choose as 6. Since $h < n$, the complexity of the dictionary is reduced compared to the complexity of an explicit dictionary.

The formulation of the problem using an analytic dictionary was:

$$\min_{\mathbf{A},\mathbf{X}} \|\mathbf{Y} - \mathbf{AX}\|_F^2 \quad \text{subject to} \quad \|\mathbf{x}_i\|_0 < K, \quad 1 \le i \le M, \tag{3.3}$$

and the formulation of the new problem is:

$$(P_{MS}) : \min_{\mathbf{A},\mathbf{X}} ||\mathbf{Y} - \mathbf{AX}||_F^2 \quad \text{subject to } ||\mathbf{x}_i||_0 \le K \quad \forall i \in \{1, ...m\} \tag{3.4}$$
$$\mathbf{a}_j(P_j^c) = 0 \quad \forall j \in \{1, ...M\}$$

where P_j^c is the subset of indices which do *not* represent partials of the atom $\mathbf{a}_j$.

In order to approximate the solution to P_{MS}, we apply an OMP sparse coding stage as described in Chapter 2, and a modified MOD or K-SVD dictionary learning algorithm according to the MS parametric dictionary.

We shall describe the modified dictionary learning algorithms.

3.3.3 Musically-Structured (MS) MOD

The MS-dictionary is composed of 88 atoms, where each atom represents a different note. Musical pieces usually don't contain all of the 88 notes. Therefore, after the sparse coding stage, we detect the atoms which were used for the sparse coding - those that have corresponding rows in $\mathbf{X}$ whose ℓ_1 norm exceeds a certain threshold, and update only them in the dictionary learning stage. The rest of the atoms remain unchanged, and they are added to the dictionary after the update. We denote the dictionary matrix of the used atoms in the k_{th} iteration as $\widetilde{\mathbf{A}}_{(k)}$, and the corresponding coefficients matrix as $\widetilde{\mathbf{X}}_{(k)}$.

The dictionary matrix is updated as in the MOD algorithm:

$$\widetilde{\mathbf{A}}_{(k)} = \underset{\widetilde{\mathbf{A}}}{\operatorname{argmin}} \|\widetilde{\mathbf{Y}} - \widetilde{\mathbf{A}}\widetilde{\mathbf{X}}_{(k)}\|_F^2 = \mathbf{Y}\widetilde{\mathbf{X}}_{(k)}^{\dagger}.$$

After updating the atoms, we *zero* the elements out of the original support. The MS-MOD is summarized in Table 3.1. The main modifications compared to the MOD are emphasized and marked in red.

Table 3.1: The musically-structured MOD algorithm

Task: Train a dictionary $\mathbf{A}$ to sparsely represent the log spectrogram matrix $\mathbf{Y}$, by approximating the solution to Problem (3.4) .

Initialization: Initialize $k = 0$, and

- ***Initialize Dictionary***: Build $\mathbf{A}_{(0)} \in \mathbb{R}^{n\times m}$, with non-zero entries only in the locations corresponding to the fundamental frequencies and their harmonics.
- ***CQT mapping***: Map the dictionary to the CQT logarithmic scale.
- **Normalization:** Normalize the columns of $\mathbf{A}_{(0)}$.

Main iteration: Increment k by 1, and apply

- **Sparse Coding Stage:** Use a pursuit algorithm to approximate the solution of
  $\widehat{x_i} = \underset{x}{\operatorname{argmin}} ||y_i - \mathbf{A}_{(k-1)}x||_2^2$ subject to $||x||_0 \leq K$
  obtaining sparse representations $\widehat{x_i}$ for $1 \leq i \leq M$. These form the matrix $\mathbf{X}_{(k)}$.
- ***Define the used atoms***: The rows in $\mathbf{X}_{(k)}$ which have a ℓ_1-norm above a certain threshold correspond to the used atoms in the dictionary matrix.
  These rows are concatenated and defined as $\tilde{\mathbf{X}}_{(k)}$ and the matrix composed of the used atoms is defined as $\tilde{\mathbf{A}}$.
- **MOD Dictionary Update Stage:** Update the dictionary of used atoms $\tilde{\mathbf{A}}_{(k)}$ by the matrix $\tilde{\mathbf{X}}_{(k)}$:
  $\tilde{\mathbf{A}}_{(k)} = \underset{\tilde{\mathbf{A}}}{\operatorname{argmin}} ||\mathbf{Y} - \tilde{\mathbf{A}}\tilde{\mathbf{X}}_{(k)}||_F^2 = \mathbf{Y}\tilde{\mathbf{X}}_{(k)}^{\dagger}$.
- ***Zero the entries out of the support***: Zero the entries out of the support which was defined in the initial dictionary.
- ***Add the unused atoms***: Add the unused atoms to $\tilde{\mathbf{A}}_{(k)}$.
  This is the updated dictionary $\mathbf{A}_{(k)}$.
- **Stopping Rule:** If the change in $\|\mathbf{Y} - \mathbf{A}_{(\mathbf{k})}\mathbf{X}_{(\mathbf{k})}\|_F^2$ is small enough, stop. Otherwise, apply another iteration.

Output: The desired outputs are $\mathbf{A}_{(k)}$ and $\mathbf{X}_{(k)}$.

3.3.4 Musically-Structured (MS) K-SVD

As in the MS-MOD algorithm, we define the dictionary matrix of used atoms as $\widetilde{\mathbf{A}}_{(k)}$ and the corresponding matrix $\widetilde{\mathbf{X}}_{(k)}$, update only $\widetilde{\mathbf{A}}_{(k)}$ and then add the unused atoms to get the updated dictionary matrix $\mathbf{A}_{(k)}$.

In the MS-MOD algorithm we have to update the whole dictionary matrix, and then we zero the elements out of the support. In the MS-KSVD, we can update only the elements in the support, since each atom is updated individually. For each atom $\mathbf{a}_{j_0}$, the error matrix $\mathbf{E}_{j_0}$ is defined as in the K-SVD. Its columns are restricted as in the K-SVD, but now its *rows* are also restricted, according to the support of the atom $\mathbf{a}_{j_0}$, thus updating only the support of $\mathbf{a}_{j_0}$. We denote this restricted error matrix as $\widetilde{\mathbf{E}}_{j_0}^R$ and the elements in the support of $\mathbf{a}_{j_0}$ as $\widetilde{\mathbf{a}}_{j_0}$. The vectors $\widetilde{\mathbf{a}}_{j_0}$ and $(\mathbf{x}_{j_0}^R)^T$ are updated using rank-1 approximation of the error matrix $\widetilde{\mathbf{E}}_{j_0}^R$ by singular value decomposition.

The algorithm is summarized in Table 3.2. The main modifications compared to the K-SVD are emphasized and marked in red.

3.4 Experiments

3.4.1 The experimental setup

We perform transcription of synthesized piano music from MIDI files, and transcription of real recorded piano music. In the tests performed here we concentrate on the task of identifying the notes. We do not deal with finding the onset and offset instances of the notes, neither with finding the number of notes played at each time (which we use as the maximal number of notes - K, for the sparse coding). In a case that this information is not given, one has to evaluate it, which is also a difficult task in polyphonic music.

The stopping criterion for the algorithms is achieved when there is a change of less than 5% in the Frobinous norm of the residual.

We compare the performance of the MS-MOD and MS-K-SVD to that of an explicit dictionary, using the unmodified MOD and K-SVD algorithms, and to that of an analytic dictionary (un-learned) with OMP in the sparse coding stage. The analytic dictionary is the initial MS-dictionary.

Table 3.2: The musically-structured K-SVD algorithm

Task: Train a dictionary $\mathbf{A}$ to sparsely represent the log spectrogram matrix $\mathbf{Y}$, by approximating the solution to Problem (3.4) .

Initialization: Initialize $k = 0$, and

- ***Initialize Dictionary***: Build $\mathbf{A}_{(0)} \in \mathbb{R}^{n\times m}$, with non-zero entries only in the locations corresponding to the fundamental frequencies and their harmonics.
- ***CQT mapping***: Map the dictionary to a the CQT logarithmic scale.
- **Normalization:** Normalize the columns of $\mathbf{A}_{(0)}$.

Main iteration: Increment k by 1, and apply

- **Sparse Coding Stage:** Use a pursuit algorithm to approximate the solution of
  $\widehat{\mathbf{x}_i} = \underset{\mathbf{x}}{\operatorname{argmin}} ||\mathbf{y}_i - \mathbf{A}_{(k-1)}\mathbf{x}||_2^2$ subject to $||\mathbf{x}||_0 \le K$
  obtaining sparse representations $\widehat{\mathbf{x}_i}$ for $1 \le i \le M$. These form the matrix $\mathbf{X}_{(k)}$.
- ***KSVD Dictionary Update Stage***: Update the support of each atom $\widetilde{\mathbf{a}}_{j_0,\{j_0=1,2,\ldots,m\}}$ in the dictionary matrix by rank-1 approximation of its error matrix $\widetilde{\mathbf{E}}_{j_0}^R$, using SVD.
  $\widetilde{\mathbf{E}}_{j_0}^R$ is the restriction of the matrix $\mathbf{E}_{j_0} = \mathbf{Y} - \sum_{j\neq j_0} \mathbf{a}_j\mathbf{x}_j^T$ to the columns that correspond to the samples in $\mathbf{Y}$ which use the atom $\mathbf{a}_{j_0}$,
  and to the *rows* that correspond to the support of $\mathbf{a}_{j_0}$.
  After applying the SVD $\widetilde{\mathbf{E}}_{j_0}^R = \mathbf{U\Delta V^T}$, update the support of the dictionary atom by $\widetilde{\mathbf{a}}_{j_0} = \mathbf{u}_1$ and the non zero elements of the representation coefficients
  by $(\mathbf{x}_{j_0}^R)^T = (\mathbf{\Delta}[1,1]\mathbf{v}_1)^T$.
- ***Add the unused atoms***: Add the unused atoms to $\tilde{\mathbf{A}}_{(k)}$.
  This is the updated dictionary $\mathbf{A}_{(k)}$.
- **Stopping Rule**: If the change in $\|\mathbf{Y} - \mathbf{A}_{(\mathbf{k})}\mathbf{X}_{(\mathbf{k})}\|_F^2$ is small enough, stop. Otherwise, apply another iteration.

Output: The desired outputs are $\mathbf{A}_{(k)}$ and $\mathbf{X}_{(k)}$.

We use different measures for the evaluation of the results: The first is the *Accuracy* measure as defined by Dixon [32]

$$Accuracy = \frac{TP}{TP+FP+FN}. \tag{3.5}$$

The term TP is the number of true positives (correct detections), FP is the number of false positives and FN is the number of false negatives. When $Accuracy$=1, it means that all the notes are identified correctly and there are no false positives nor false negatives.

The second measure is the *transcription error score* (Poliner and Ellis [73]). If we denote by N_{sys} the number of reported pitches, by N_{ref} the number of ground-truth pitches and by N_{corr} their intersection, then the transcription error score across all time frames t is:

$$E_{\mathrm{tot}} = \frac{\sum_{t=1}^{T} \max(N_{\mathrm{ref}}(t), N_{\mathrm{sys}}(t)) - N_{\mathrm{corr}}(t)}{\sum_{t=1}^{T} N_{\mathrm{ref}}(t)}.$$

The MIDI files for the experiments include a monophonic musical piece, a simple polyphonic piece and a complicated polyphonic piece, as well as a piece of chords and a piece of octaves. These MIDI files are synthesized with a sampling frequency of 44.1 kHz, by FM-synthesis, using the "Matlab and MIDI" software (http://www.kenschutte.com/midi) by K. Schutte. The MS-K-SVD code is a modified version of the K-SVD written by R. Rubinstein (http://www.cs.technion.ac.il/~ronrubin/software.html).

The recorded piano files include a monophonic piece, a piece of chords and a piece of octaves. These pieces are recorded on a Yamaha U1 piano, and saved with a sampling frequency of 44.1 KHz.

We also compare our transcription method to former reported transcription results (Costantini et al. [25], Poliner and Ellis [73], Ryynänen and Klapuri [77] and Marolt [58]), which were examined on a set of polyphonic classical synthesized MIDI music pieces which were collected from the Classical Midi Page (http://www.piano-midi.de). The list of 130 piece set appears at [73]. The first minute from each song was taken. The 130 piece set was randomly split into 92 training, 24 testing and 13 validation pieces. In addition to the synthesized audio, piano recordings were made from a subset of the MIDI files using a Yamaha Disklavier playback grand piano. 20 training files and 10 testing files were randomly selected for recording. The recorded files are downloaded from (`http://labrosa.ee.columbia.edu/projects/piano/`).

3.4.2 Results

Synthesized MIDI music

First we present the performance of the MS-dictionary learning algorithm on a monophonic piece, a simple polyphonic piece and a complicated polyphonic piece. The transcription results are presented in Tables 3.3 and 3.4.

Table 3.3: Transcription *Accuracy* percentage for three different types of songs

	Monophonic music	Simple polyphonic music	Complicated polyphonic music
MS-MOD	**100**	**69.6**	**64.0**
MS-K-SVD	**100**	**67.7**	**64.5**
MOD	100	39.5	43.5
K-SVD	100	37.6	42.7
Analytic dictionary	100	45.8	41.0

Table 3.4: Transcription E_{tot} percentage for three different types of songs

	Monophonic music	Simple polyphonic music	Complicated polyphonic music
MS-MOD	**0**	**17.9**	**23.6**
MS-K-SVD	**0**	**19.3**	**23.2**
MOD	0	44.5	41.0
K-SVD	0	46.2	41.7
Analytic dictionary	0	37.2	43.3

Some observations from the tables are:

1. All methods identify perfectly the notes in the monophonic music.
2. The performance of the algorithms based on the parametric MS dictionary is better than that of the explicit dictionaries as well as the analytic dictionaries when identifying notes in polyphonic music. The advantage is more significant in the simple polyphonic music than in the complicated polyphonic music (where the data set of notes is larger and richer).

3. The performance of the MS-dictionary and the analytic dictionary are worse when transcribing the complicated polyphonic music compared to when transcribing the simple polyphonic music. On the contrary, the performances of the methods based on explicit dictionaries are improved. This implies that the explicit dictionaries need a large data set in order to achieve good performance.

The piano rolls and their identification by MS-MOD and MS-KSVD of the three songs are presented in Figures 3.5, 3.6 and 3.7. Some mistakes in the transcription can be identified as spurious notes (mistakes of semitones), which might be fixed with a higher frequency resolution, and as notes that share a similar spectral shape, such as notes with a difference of a multiple integer of an octave, or as notes that share common harmonics with the true note.

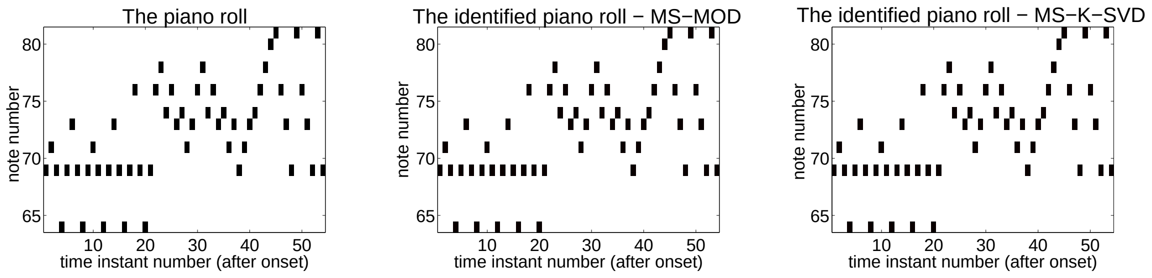


Figure 3.5: The real piano roll (left) and the identified piano roll of a monophonic piece using MS MOD (middle) and using MS K-SVD (right). Black = True positive , Red = False positive, Yellow = False negative.

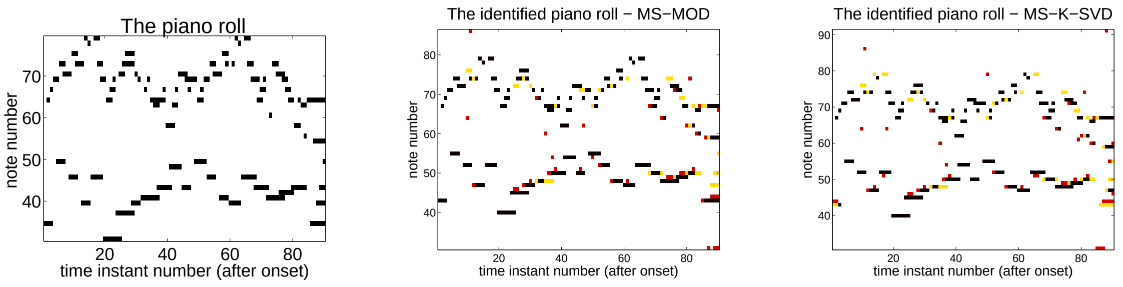


Figure 3.6: The real piano roll (left) and the identified piano roll of a simple polyphonic piece using MS MOD (middle) and using MS K-SVD (right). Black = True positive , Red = False positive, Yellow = False negative.

We now return to the problem which we mentioned in Chapter 1 - deduction of notes from chords. This problem is hard, since the notes in a chord share multiple harmonics.

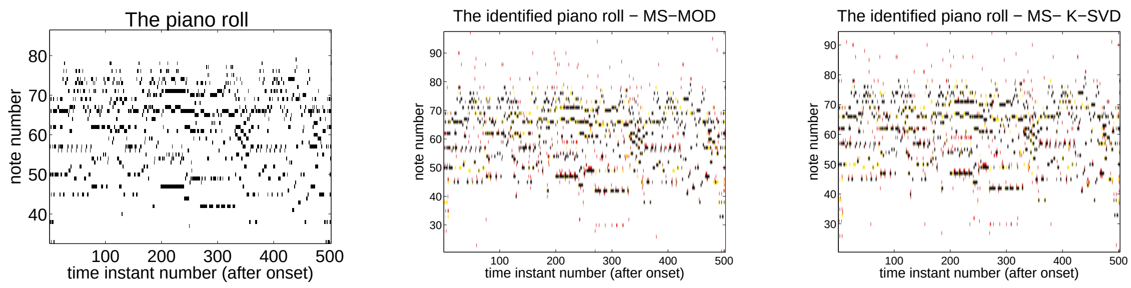


Figure 3.7: The real piano roll (left) and the identified piano roll of a complicated polyphonic piece using MS-MOD (middle) and using MS-K-SVD (right). Black = True positive , Red = False positive, Yellow = False negative.

An even harder problem, is that of deduction of notes from octaves. In an octave, all of the harmonics of the higher note are shared with those of the lower note. We compare the results of the MS-dictionary to that of an explicit dictionary, and of an analytic dictionary.

The piano rolls of the original and identified music are presented in Figures 3.8 and 3.9.

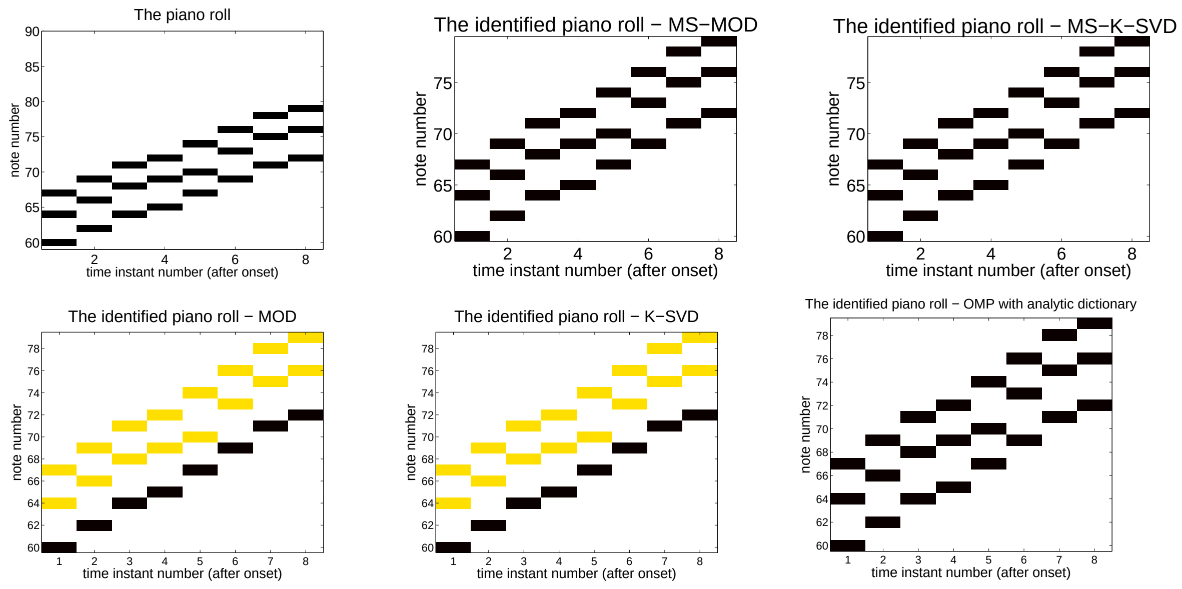


Figure 3.8: The real piano roll and the identified piano roll of synthesized chords using different methods. Black = True positive , Red = False positive, Yellow = False negative.

In the case of the chords, MS-MOD, MS-K-SVD and OMP with an analytic dictionary identify all the notes, despite of the difficulty of this task. The MOD and K-SVD identify only the lower notes. In the case of the octaves, the MS-MOD, MS-K-SVD and OMP

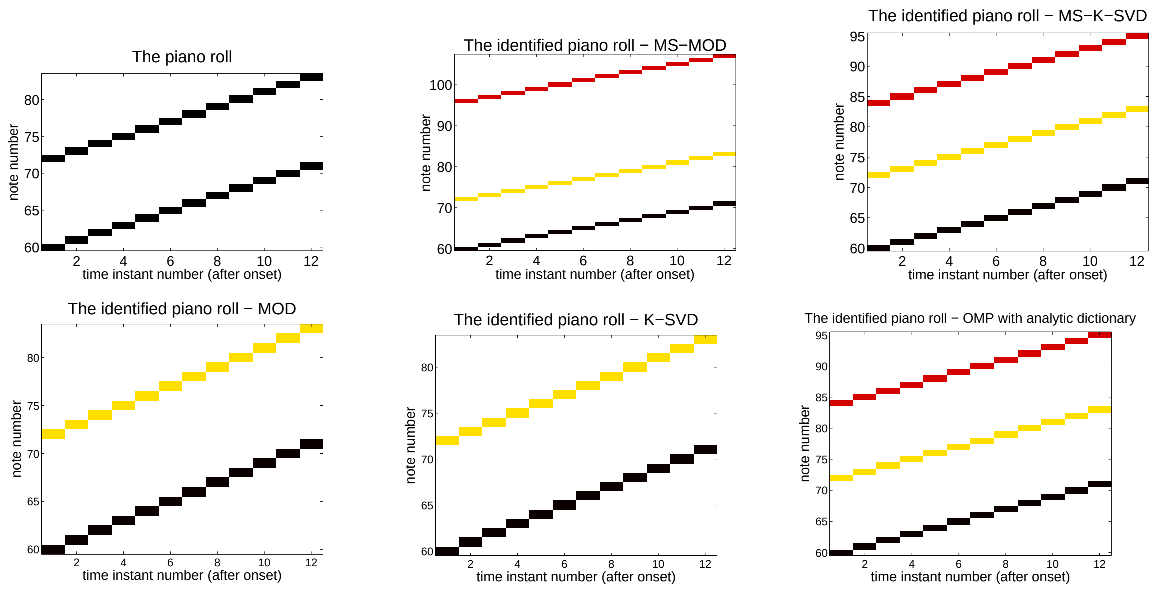


Figure 3.9: The real piano roll and the identified piano roll of MIDI synthesized octaves using different transcription methods. Black = True positive , Red = False positive, Yellow = False negative.

with an analytic dictionary identify the lower notes, and have a mistake of an octave or two in the higher notes, due to the similar spectral shape mentioned before. The MOD and K-SVD identify here also only the lower notes, which contain the harmonics of the upper notes.

We compare the former reported results (Costantini et al. [25], Poliner and Ellis [73], Ryynänen and Klapuri [77] and Marolt [58]), on the set of polyphonic classical music which was described in subsection 3.4.1, to the transcription of the testing set by MS-MOD, MS-K-SVD, MOD, K-SVD and OMP with an analytic dictionary.

However, the results should be compared carefully due to the following differences between the algorithms:

1. The former transcription algorithms are supervised methods, i.e. they are based on a the training set mentioned before. Their results presented here are after training on pieces which were written by the same composers as in the testing set. Our algorithm, and the other transcription methods based on sparse coding, are unsupervised methods, and they are tested on the same testing set *without training*.

2. The former methods applied automatic detection of onsets and offsets. Since we

Table 3.5: Transcription performance on synthesized polyphonic classical music

	Accuracy(%)	E_{tot} (%)
MS-MOD	**58.7**	**28.2**
MS-K-SVD	**59.8**	**27.2**
MOD	39.4	45.3
K-SVD	31.0	54.1
Analytic dictionary	39.1	45.1
Costantini et al.	72.3	20.1
Poliner and Ellis	64.7	41.7

focus only on the problem of notes identification, we detect the onsets and offsets manually or use the data from the MIDI file. In each case, the transcription results are reported with respect to the detection accuracy of onsets and offsets.

3. The number of notes in each frame was inserted as a parameter to the transcription methods based on sparse representations (MS-MOD, MS-K-SVD, MOD, K-SVD and OMP with an analytic dictionary), as the maximal cardinality K of each vector of coefficients $\mathbf{x}$.

A comparison of the results on synthesized polyphonic classical music are presented in Table 3.5. It can be seen that the measures *Accuracy* and E_{tot} are not always correlated in their performance, due to their different definitions. Therefore, they are both used for evaluation.

From the table one can see that the results of the transcription by MS-dictionary learning algorithms outperform those of the other unsupervised methods for transcription using sparse representations (MOD, K-SVD and OMP with an analytic dictionary). They are inferior (though not significantly) comparing to those of other transcription methods, but these methods are based on supervised learning, thus not directly comparable.

Recorded piano music

The task of transcription of real recorded piano music is much harder, since it entails several additional obstacles:

- *Beginning transients* - Beginning transients of tones contain noisy components. Further, the sinusoid partials are in slightly inharmonic relations during the beginning tens of milliseconds, due to the strike of a hammer at strings (e.g., in a piano).

- *Noises* - from the environment.

- *Formants* - The shape of the body of a piano (and of other instruments) produces several formants: frequency bands in which harmonic partials are louder and decay slower because of the resonance of the body. The different decay slopes also cause the color of sound to change gradually in the course of its playing.

- *Cross-resonance* - Playing just one note on the keyboard makes also the other free strings to start gently resonate along with the hammered string.

- *Strings inharmonicity* - In an ideal vibrating string, when the wavelength of a wave on a stretched string is much greater than the thickness of the string, the overtones are located at multiple integers of the fundamental frequency, i.e. at the harmonics. However, this property doesn't exist for the lowest and highest strings of the piano. The lowest strings, which would have to be the longest, are most limited by the size of the piano. The designer of a short piano is forced to use thick strings to increase mass density and are thus driven into inharmonicity. The highest strings have to be under the greatest tension, yet must also be thin to allow for a low mass density. The limited strength of steel forces the piano designer to use very short strings whose short wavelengths thus generate inharmonicity. When tuning a piano, adjustments to the inharmonicity are made by slightly sharpening the high notes and flattening the low notes so that the overtones of low notes have the same frequency as the fundamentals of high notes. More can be read about this subject in [92].

- *Reverberations* - When recording real music, there are always reverberations caused by the room. In order to reduce reverberations, the recording should be made in a small, sound-absorbing room.

First we perform some simple tests, as mentioned in subsection 3.4.1. We compare the performances of MS-MOD and MS-K-SVD to that of MOD, K-SVD, and OMP with an analytic dictionary , as presented in Figures 3.10, 3.11 and 3.12.

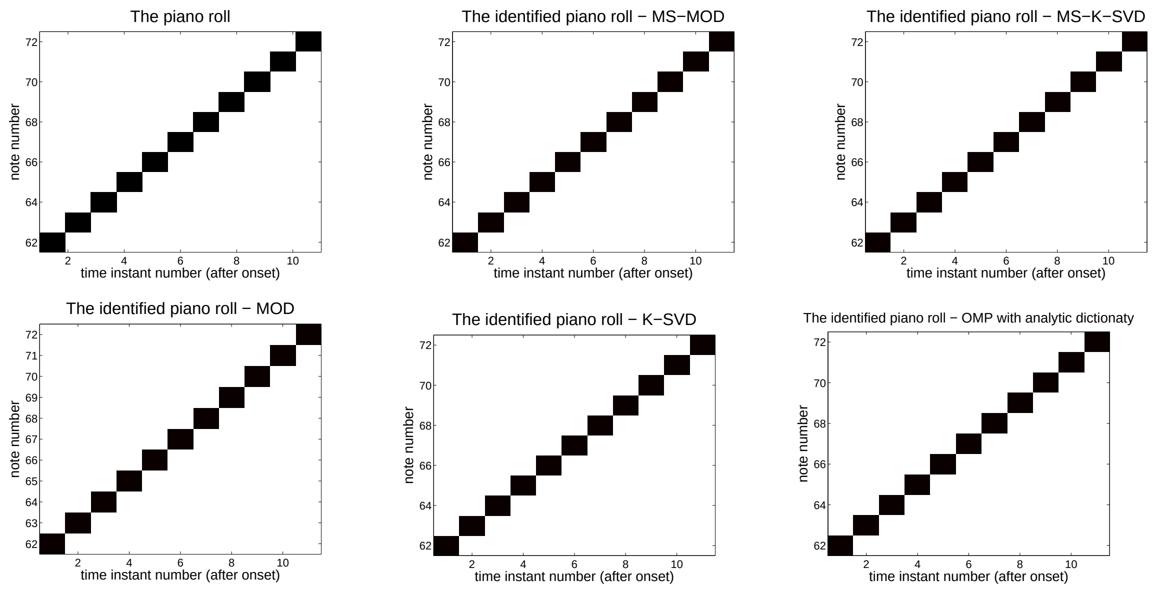


Figure 3.10: The real piano roll and the identified piano roll of recorded monophonic piano music using different transcription methods. Black = True positive, Red = False positive , Yellow = False negative.

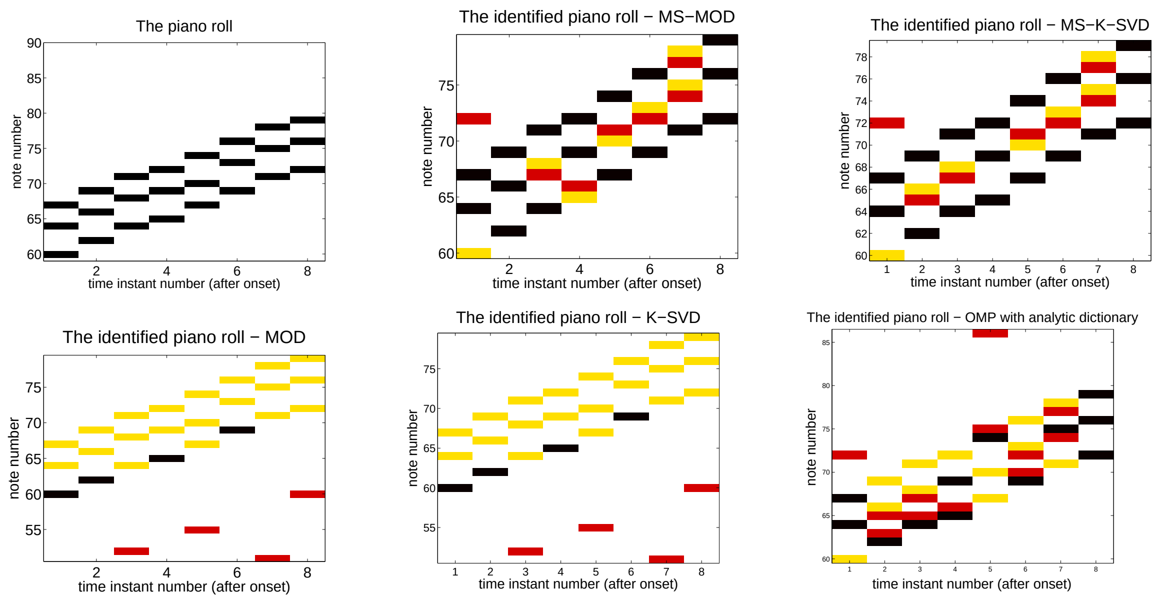


Figure 3.11: The real piano roll and the identified piano roll of recorded piano chords using different transcription methods. Black = True positive, Red = False positive , Yellow = False negative.

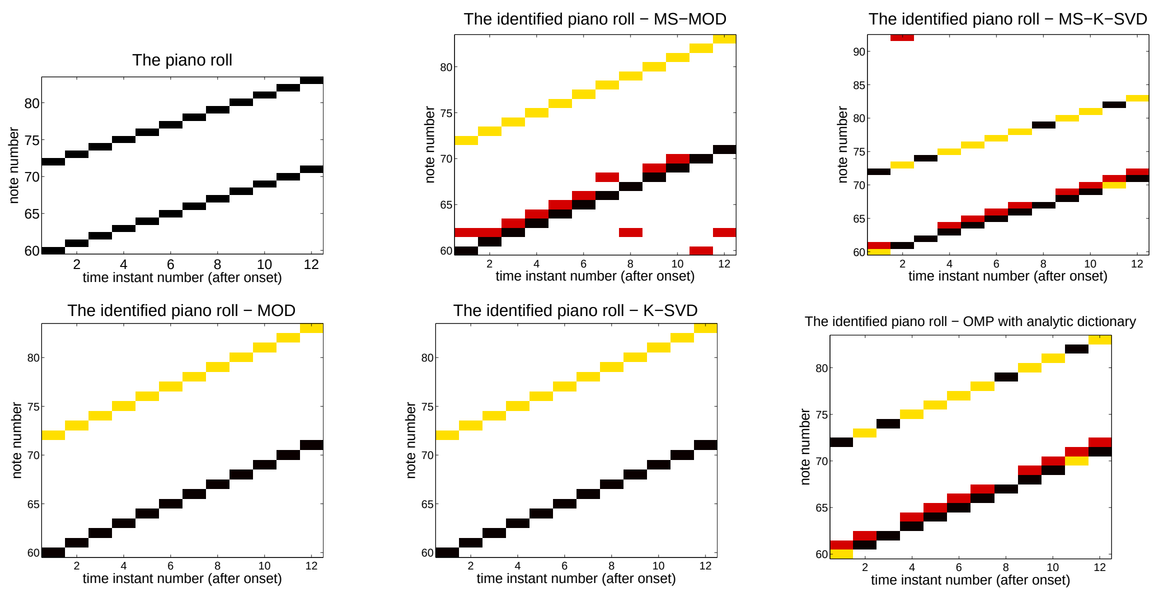


Figure 3.12: The real piano roll and the identified piano roll of recorded piano octaves using different transcription methods. Black = True positive, Red = False positive , Yellow = False negative.

All the methods identify perfectly the notes in the monophonic music. The detection of chords and octaves is worse than in synthesized music, and this may stem from the obstacles mentioned before. Still, the performances of the MS-MOD, MS-K-SVD and OMP with an analytic dictionary are better than those of the MOD and K-SVD, which identify one note at a time while there are actually two or three notes.

We compare the transcription results on the recorded classical polyphonic music pieces which were mentioned in subsection 3.4.1. The results of transcription on recorded polyphonic music are presented in Table 3.6, and the results of transcription on both synthesized and recorded polyphonic music are presented in Table 3.7.

In recorded classical piano music, similarly to synthesized classical piano music, the results of the transcription by MS-dictionary learning algorithms outperform those of the other unsupervised methods for transcription using sparse representations (MOD, K-SVD and OMP with an analytic dictionary). In this case they have similar results to other transcription methods, and even outperform some of them, although they are supervised methods.

Table 3.6: Transcription performance on recorded polyphonic classical music

	Accuracy(%)	E_{tot} (%)
MS-MOD	**54.3**	**30.9**
MS-K-SVD	**54.3**	**31.1**
MOD	22.9	63.3
K-SVD	19.8	67.4
Analytic dictionary	43.9	40.1
Costantini et al.	59.2	33.3
Poliner and Ellis	56.5	46.7

Table 3.7: Transcription performance on synthesized plus recorded polyphonic classical music

	Accuracy(%)	E_{tot} (%)
MS-MOD	**57.4**	**29.0**
MS-K-SVD	**58.2**	**28.3**
MOD	34.5	50.6
K-SVD	27.7	58.0
Analytic dictionary	40.5	43.6
Costantini et al.	68.0	24.6
Poliner and Ellis	62.3	43.2
Ryynänen and Klapuri	56.8	46.0
Marolt	30.4	87.5

3.5 Summary

We have presented a polyphonic music transcription system based on sparse representations. We showed the motivation for using sparse representations for transcription of polyphonic music, and developed a system for transcription of polyphonic music based on sparse representations with a structured parametric dictionary, which overcomes limitations of transcription by explicit or analytic dictionaries.

The main advantages of our algorithm are:

- Good adaptation to the timbre of the signal - outperforms analytic dictionaries.
- No over-fitting - deduces individual notes from their combination also when the data set is small, and when the notes are played only in conjugation with other notes, and with several overlapping harmonics (e.g., in chords), unlike explicit dictionaires.
- Reduced computational complexity - compared to explicit dictionaries.
- It is an unsupervised method, thus a training set is not required.

In Chapter 5 we offer an algorithm for classification of audio data, where the classification is based on a mapping by a manifold learning method named "diffusion maps". Transcription of audio data, which is performed in this chapter, can be used as a preliminary stage to the classification, in order to get meaningful features.

Chapter 4

Manifold learning

4.1 Introduction

In many problems of machine learning, we are given a high-dimensional data set, for which we wish to reveal the limited variables which produced it. For example, consider images taken by a camera in a 3-dimensional space. Each image is very high dimensional, since it contains hundreds of pixels, but it can be represented in a 3-dimensional space , according to its location in space. The low-dimensional representations of the data are often referred as "intrinsic variables", since these are the values that control the production of the data.

A need in reducing the dimensionality of the data stems from "the curse of dimensionality" - the number of all possible unique samples grows exponentially as their dimension grows. Therefore, as the dimension grows, it becomes more difficult to sample the space (the sample complexity grows), and an efficient way to reduce the dimensionality is needed.

Dimensionality reduction can be achieved by linear methods such as PCA, but such methods ignore the fact that data (especially natural data) lies on a non-linear manifold. The dimensionality reduction problem is formulated as:
Given $\mathbf{x}_1, ...\mathbf{x}_n \in \mathbb{R}^D$, find representations $\mathbf{y}_1, ...\mathbf{y}_n \in \mathbb{R}^d$, which preserve some relevant intrinsic information or structure, such that $d << D$.

The chapter is organized as follows. In Section 4.2, we describe linear dimensionality reduction. In Section 4.3, we describe several non-linear manifold learning techniques. In Section 4.4 we present several existing classification and clustering applications based on manifold learning, followed by a summary in Section 4.5.

4.2 Linear dimensionality reduction techniques

Linear methods for dimensionality reduction are based on the assumption that the data lies on a low-dimensional *linear* manifold. Naturally, this assumption is not always (and usually not) true, thus such methods might represent the data inaccurately. An example for such a method is principal component analysis (PCA).

Principal component analysis (PCA)

This is a linear orthogonal transformation that transforms the data to a low dimensional space, where its orthogonal coordinates are in the directions of the greatest variance of the data. Given a data set $\mathbf{x}_1, ...\mathbf{x}_n \in \mathbb{R}^D$ in a high dimensional space D, we aim to find low dimensional representations $\mathbf{y}_1, ...\mathbf{y}_n \in \mathbb{R}^d$ such that

$$\mathbf{y}_i = \mathbf{W}^{*T}\mathbf{x}_i,$$

and

$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmax}}\{Var(\mathbf{Y})\} = \underset{\mathbf{W}}{\operatorname{argmax}}\{Tr(\mathbf{W}^T\mathbf{X}\mathbf{X}^T\mathbf{W})\},$$

where $\mathbf{Y} = [\mathbf{y}_1, ...\mathbf{y}_n]$ and $\mathbf{X} = [\mathbf{x}_1, ...\mathbf{x}_n]$. I.e., $\mathbf{y}_i$ is a projection of $\mathbf{x}_i$, which maximizes the variance of the data. Since W and $\mathbf{Y}$ are orthogonal matrices, it can be shown that the solution W^* consists of the leading eigenvectors of $\mathbf{X}\mathbf{X}^T$, which are called the "principal components".

This projection is both *linear* and *global.* Another way to interpret PCA is as fitting the best *linear* manifold to the data, of a certain desired dimension. This linear manifold is defined by the projections of the data on the principal components. A more general model would fit some *general* manifold to the data.

4.3 Non-linear dimensionality reduction techniques

Manifold learning methods are a family of non-linear, non-parametric, geometrically motivated dimensionality reduction techniques, which aim to learn the geometry of the data. They share a common assumption, that the data lives on a low dimensional *non-linear*

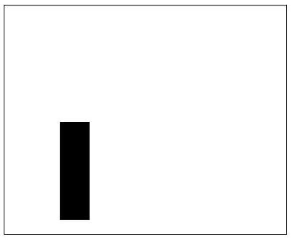


Figure 4.1: An image of a vertical bar v. The family of vertical bars represents a non-linear set

manifold (or close to it, if it is noisy). The manifold is embedded in a high dimensional space. Given a data set $\mathbf{x}_1, ...\mathbf{x}_n \in \mathbb{R}^D$ in a high dimensional space D, we aim to find low dimensional representations $\mathbf{y}_1, ...\mathbf{y}_n \in \mathbb{R}^d$, which capture the nature of the data, in the sense that they preserve intrinsic relevant information and structure.

For natural data, usually the assumption that it lies on a non-linear manifold is true. For example, we consider images, which are functions

$$f : \mathbb{R}^2 \to [0, 1],$$

where $f(x, y)$ is the intensity of the image at pixel (x, y), and we examine this family of images:

$$\mathcal{F} = \{f | \exists t, r, f(x, y) = v(x - t, y - r), \forall x, y\},$$

which are images of translations of the vertical bar v, as presented in Figure 4.1. The set $\mathcal{F}$ is not a linear space, since it is not closed to the addition of two elements in it. This set is controlled by two degrees of freedom, which are parameterized by t and r.

The aim of manifold learning is two-fold: The first is to learn a function whose domain is the manifold on which all the data lives, and whose range might be a finite set (if the goal is clustering or classification) or the real numbers (if the goal is regression or dimensionality reduction). The second is to learn the geometry of the manifold itself. The known data which we use for the learning are randomly (noisy) sampled points from the manifold.

Examples for such techniques include:

- Isomap (Tenenbaum et al., 2000) [86]

- Locally-linear embedding (LLE) (Rowies and Saul, 2000) [75]
- Laplacian eigenmaps (Belkin and Niyogi, 2001) [7]
- Hessian eigenmaps (Donoho and Grimes, 2003) [36]
- *Diffusion maps* (Coifman and Lafon, 2006) [22]
- Related: Kernel PCA (Schölkopf et al., 1998) [80].

Kernel PCA [80]

This is an extension of principal component analysis (PCA) using techniques of kernel methods. Suppose that we are given the set $\mathbf{x}_1, ...\mathbf{x}_n \in X \in \mathbb{R}^D$ and a positive semi definite kernel k. Then there exists a map $\phi : X \rightarrow \mathcal{H}$ into a dot product space $\mathcal{H}$ such that for all $\mathbf{x}_i, \mathbf{x}_j \in X$, we have $< \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) >= k(\mathbf{x}_i, \mathbf{x}_j)$. Kernel PCA computes the principal components of the points $\phi(\mathbf{x}_1), ...\phi(\mathbf{x}_n)$. The principal components and the embedding are computed similarly to PCA, but applied on the covariance matrix $\Phi(\mathbf{x})^T\Phi(\mathbf{x})$ (where $\Phi(\mathbf{x}) = [\phi(\mathbf{x}_1), ...\phi(\mathbf{x}_n)]$) instead of on $\mathbf{X}^T\mathbf{X}$, achieving a non-linear mapping.

Isomap [86]

This is a method in which geodesic distances on a weighted graph are incorporated with classic multi dimensional scaling (MDS). A neighborhood graph is built, where each data point in the neighborhood defines a vertex of the graph. The neighborhood is determined by the K closest neighbors, or by all the neighbors in a radius ε. The geodesic distances between all the pairs of points in the graph are estimated by computing their shortest graph distances in the graph, and the matrix of distances D is defined, where D_{ij} is the distance between vertex $\mathbf{x}_i$ to vertex $\mathbf{x}_j$. Finally, classical MDS is applied on the matrix of distances D, embedding the data into a lower-dimensional Euclidean space which best preserves the manifold's estimated intrinsic geometry. The embedding is achieved by computing the inner products between the data points from the distances between them

$$A = -\frac{1}{2}HDH,$$

where $A_{ij} =< \mathbf{x}_i, \mathbf{x}_j >$, and $H = I - \frac{1}{n}\mathbf{1}\mathbf{1}^T$. The embedding for $\mathbf{y}_i$ is a weighted set of the leading eigenvectors ϕ of the matrix A, at the i_{th} entry.

$$\mathbf{y}_i = \left(\sqrt{\lambda_1}\phi_1(i), ..., \sqrt{\lambda_k}\phi_k(i)\right) \in \mathbb{R}^d$$

It was shown by Zha and Zhang [93] , that isomap assumes that the data lie on a manifold which is globally isometric to a convex subset of a low-dimensional Euclidean space.

Locally linear embedding (LLE) [75]

Assuming that the data points are sampled from a smooth underlying manifold, and that the manifold is well sampled (there is sufficient data), we expect that each data point and its neighbors lie on or close to a locally linear patch of the manifold. The local geometry of these patches can be characterized by linear coefficients that reconstruct each data point from its neighbors.

Similarly to ISOMAP, a neighborhood graph is built. Weights W_{ij} are computed, which best linearly reconstruct each data point $\mathbf{x}_i$ from its neighbors, minimizing the cost in

$$\varepsilon(W) = \sum_i \left| \mathbf{x}_i - \sum_j W_{ij}\mathbf{x}_j \right|^2 ,$$

subject to two constraints: $W_{ij} = 0$ if $\mathbf{x}_j$ does not belong to the neighbors of $\mathbf{x}_i$, and $\sum_j W_{ij} = 1$. Finally, the low dimensional embedding vectors $\mathbf{y}_i^{\text{opt}}$ are computed according to the weights W_{ij}, minimizing the quadratic form

$$\Phi(\mathbf{y}) = \sum_i \left| \mathbf{y}_i - \sum_j W_{ij}\mathbf{y}_j \right|^2 .$$

Subject to constraints that make the problem well-posed, the optimal embedding $\mathbf{y}_i$ is obtained as i_{th} entries of the d bottom nonzero eigenvectors of the matrix $(I-W)^T(I-W)$.

It was shown by Belkin and Niyogi [8] that the LLE algorithm is equivalent to finding the eigenfunctions of the square of the normalized graph Laplacian of the data.

Laplacian eigenmaps [7, 8]

Similarly to the methods described before, first a neighborhood graph is built. The weights between the nodes are determined either according to the Heat kernel

$$W_{ij} = \exp^{\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{t}},$$

or simple-minded: $W_{ij} = 1$ if and only if vertices i and j are connected by an edge. Assuming that the graph is fully connected, the eigenvalues and eigenvectors for the generalized eigenvector problem are computed:

$$L\mathbf{y} = \lambda D\mathbf{y} \tag{4.1}$$

where D is a diagonal matrix and its diagonal entry $D_{ii} = \sum_j W_{ij}$ is the row sum of the i_{th} row in W. L is the Laplacian matrix

$$L = D - W,$$

which is symmetric and positive semi-definite. The bottom nonzero eigenvectors $(\mathbf{y}_1, ..., \mathbf{y}_d)$ that solve 4.1, are used for the embedding. I.e, the embedding of $\mathbf{x}_i$ is $(\mathbf{y}_1(i), ..., \mathbf{y}_d(i))$.

The Laplacian eigenmaps preserves local information. Choosing the d bottom nonzero eigenvectors for the eigen problem (4.1) is equivalent to solving the following approximation problem

$$\min_y \{\mathbf{y}^T L\mathbf{y}\} \quad \text{subject to} \quad \begin{aligned} \mathbf{y}^T D\mathbf{y} &= 1 \\ \mathbf{y}^T D\mathbf{1} &= 0. \end{aligned} \tag{4.2}$$

It can be shown that

$$\mathbf{y}^T L\mathbf{y} = \frac{1}{2}\sum_{i,j} \|\mathbf{y}_i - \mathbf{y}_j\|^2 W_{ij}.$$

Therefore, the minimization in (4.2) is equivalent to the minimization of the objective function

$$\frac{1}{2}\sum_{i,j} \|\mathbf{y}_i - \mathbf{y}_j\|^2 W_{ij},$$

where the constraints in (4.2) are intended to remove arbitrary scaling, and to eliminate the trivial solution of $\mathbf{y} = 1$, corresponding to the eigenvalue $\lambda_0 = 0$. In Laplacian eigenmaps, W_{ij} is large when $\mathbf{x}_i$ and $\mathbf{x}_j$ are close to each other, and this enforces $\mathbf{y}_i$ and $\mathbf{y}_j$ to be close to each other. Thus, this embedding preserves local information between points, and connected points stay as close together as possible.

For a uniform distribution of points on the manifold, the (normalized) graph Laplacian obtained from the data and its eigenvectors converge to the Laplace-Beltrami operator defined on the manifold and to its eigenfunctions respectively, as the number of points increase and if the kernel bandwith is selected appropriately. A proof is shown in [9]. The eigenfunctions of the Laplace-Beltrami operator have desirable properties for embedding - they are well adapted to the geometry and the topology of the manifold [54], justifying the use of the eigenvectors of the graph Laplacian for embedding.

Hessian eigenmaps [36]

This method is derived from a conceptual framework in which the manifold is locally isometric to an open, connected subset of a lower dimensional Euclidean space. In this method, unlike in isomap, this subset is not restricted to be convex, expanding the class of cases which can be solved by isometry principles.

Similarly to the methods described before, a neighborhood graph is built. The Hessian of f, where $f : \mathcal{M} \to \mathbb{R}$ ($\mathcal{M}$ denotes the manifold) is defined as H_f. We define the matrix $\mathcal{H}(f)$

$$\mathcal{H}(f) = \int_{\mathcal{M}} \|H_f(m)\|_F^2 dm,$$

which is a smooth quadratic form of the Hessian defined on all tangent spaces in points m of the manifold. The Hessian is defined by the orthogonal coordinates on the tangent spaces of the manifold. If the manifold is locally isometric to an open connected subset of a lower dimensional Euclidean space, then $\mathcal{H}(f)$ has a $d+1$ dimensional null space. The isometric coordinates, which are the desired embedding, can be recovered up to a rigid motion as a d dimensional basis of the null space of $\mathcal{H}(f)$. Practically, they are computed as the d bottom nonzero eigenvectors of $\mathcal{H}(f)$.

This method can be viewed as a modification of the Locally Linear Embedding, and the theoretical framework can be viewed as a modification of the Laplacian eigenmaps framework, where a quadratic form based on the Hessian is substituted in place of a one based on the Laplacian.

Diffusion maps [22]

This method leads to a similar mapping as Laplacian eigenmaps, but stems from a different point of view. A weighted graph is built, where the vertices are the data points and the connections between them are weighted by a symmetric point-wise positive kernel. A Markov process is defined on the graph, by defining the probability matrix

$$P = D^{-1}W,$$

where D and W are defined as in Laplacian eigenmaps. The diffusion process is applied by taking powers of P, as well as revealing the structure of the data points in different scales. The embedding of each data point is obtained by the multiplication of the leading eigenvalues by the leading eigenvectors of the matrix P, not including the trivial eigenvector which corresponds to the eigenvalue $\bar{\lambda}_0 = 1$.

The leading eigenvectors of P are equivalent to the bottom eigenvectors of the generalized eigenvalue problem of the Laplacian eigenmaps $L\mathbf{y} = \lambda D\mathbf{y}$, since:

$$\begin{aligned} L\mathbf{y} &= \lambda D\mathbf{y} \\ (D - W)\mathbf{y} &= \lambda D\mathbf{y} \\ (I - D^{-1}W)\mathbf{y} &= \lambda\mathbf{y} \\ D^{-1}W\mathbf{y} &= (1 - \lambda)\mathbf{y} \end{aligned}$$

and this is equivalent to the eigenvalue problem

$$D^{-1}W\mathbf{y} = \bar{\lambda}\mathbf{y}$$

with $\bar{\lambda} = 1 - \lambda$. Therefore, for a uniform distribution of points on the manifold, the leading eigenvectors used for the diffusion mapping can be considered, similarly as in Laplacian eigenmaps, as approximations of the eigenfunctions of the Laplace-Beltrami operator. If the distribution is not uniform, a normalization can be applied on the kernel, making it independant to the distribution of the points [22].

4.4 Classification based on manifold learning

In this section we present some examples of classification and clustering based on manifold learning.

Lafon et al. [53] performed lipreading based on frames from a movie of a subject's lips, where one of 10 spoken digits had to be recognized. The set of features for each frame was the gray values of all the pixels in the frame, leading to 15400 dimensional feature vectors. They built a nearest neighbor classifier, and used the geometric harmonics extension scheme for embedding a new frame. They haven't compared the classification results to other classification methods.

In a later work, Keller et al. [49] performed audio-visual recognition using diffusion maps. They combined recognition of spoken digits from visual movie frames with recognition from audio records. The frames were used similarly as in [53]. The audio records were divided to frames, fast fourier transform (FFT) was applied on each one, and the discrete cosine transform was applied on the result. Each vector was averaged into 256 uniform bins, leading to 256 dimensional feature vectors. Each feature vector was embedded in a 20 dimensional diffusion space, and classification was performed using this embedding. The classification results were compared to classification after dimensionality reduction by principal component analysis (PCA), and it was shown that the dimensionality reduction with diffusion maps was more effective for classification in this case.

Other works (Nadler et al. [65], Belkin and Niyogi [7]) applied *clustering* with embedding by manifold learning techniques, for different types of data - mixtures of Gaussians, words from the Brown corpus and phonemes in a sentence. However, they did not compare the results to other clustering algorithms. Nadler et al. also showed that spectral clustering may not work well on multi-scale data, leading to the need of a multi-scale approach for clustering. We show in Chapter 5 that our classification algorithm is based on a multi-scale embedding, by normalization of each feature by its standard deviation.

4.5 Summary

In this chapter we have reviewed manifold learning methods, and explained their advantages compared to linear methods for dimensionality reduction. Several algorithms of manifold learning were described. These algorithms share several properties - we showed that they follow a common framework, which includes building a graph from points randomly sampled from the manifold, defining a meaningful matrix from the points on the graph, and applying spectral decomposition of this matrix in order to embed the data points in a space which reveals the geometry of the manifold. Another common property is that several manifold learning algorithms (LLE, Laplacian eigenmaps and diffusion maps), are based on spectral decomposition of the graph Laplacian, whose eigenvectors represent well the geometry of the manifold, as approximations of the eigenfunctions of the Laplace-Beltrami operator on the manifold. Furthermore, in [46] Ham et al. showed that all kernel-based manifolds are special cases of kernel-PCA.

We have also presented several examples for classification and clustering using manifold learning techniques. In Chapter 5 we expand the talk on diffusion maps, and present our scheme for classification of audio and speech data based on diffusion maps.

Chapter 5

Classification of audio and speech data using Diffusion Maps

5.1 Introduction

Automatic classification of audio data is a key stage in many algorithms and applications in signal processing. In this work we focus on two main classification applications and examine the use of diffusion maps as part of the classification process. We examine its performance in two audio classification tasks - classification of songs by genre, and classification of unvoiced fricative phonemes. First we shall describe the applications and give the motivation for focusing on them.

The first application is classification of musical pieces according to their genre. Musical genres are labels created and used for categorizing and describing the vast universe of music. Different genres differ from each other in their instrumentation, rhythmic structure and pitch content of the music. They include, for example - classic music, jazz, rock etc. In recent years there has been a growing interest in automatically categorizing music into genres, as part of extracting musical information in general. Automatically extracting musical information is gaining importance as a way to structure and organize the increasingly large numbers of music files available digitally on the web. In addition, features evaluated by automatic genre classification can be used for tasks as similarity retrieval, segmentation and audio thumbnailing.

The second application is classification of the unvoiced fricative phonemes: /s/, /sh/,

/th/, /f/. Classification of phonemes is the process of finding the phonetic identity of a short section of a spoken signal [31]. It is a key stage in many speech processing algorithms and applications, such as spoken term detection, continuous speech recognition, and speech coding, but it can also be useful on its own, for example in selective processing of phonemes for the hearing impaired, or in the professional music industry. The unvoiced fricative phonemes are specifically important since they tend to be indistinguishable for the hearing impaired [6].

Traditional methods for classification of audio data consist of two main stages: feature extraction, in which relevant features (usually temporal and spectral) are extracted from the signal, and classification according to these features. There are two fundamental problems in these methods:

1. In order to capture the nature of the signals and to differ efficiently between them, the feature vectors usually need to be high dimensional. As the number of the signals increases, the computational complexity and the sample complexity increase as well, leading to the need for a dimensionality reduction technique.

2. Traditional classification techniques are not adaptive to the intrinsic geometry of the feature vectors. E.g, classification based on Euclidean distances does not capture the intrinsic distances between feature vectors, assuming that they lie on a non linear manifold.

Manifold learning techniques aim to discover the non-linear nature of the manifold on which the data lies, in order to characterize it better. We use a technique called "diffusion maps", which maps the connections on the manifold to Euclidean distances, leading to an efficient classification based on Euclidean distances. The diffusion maps technique leverages the relationship between the diffusion operator on a manifold and a Markov transition matrix operating on functions defined on the graph whose vertices were sampled from the manifold.

The chapter is organized as follows. In Section 5.2 we describe the proposed classification algorithm and give a theoretical motivation for using diffusion maps, in Section 5.3 we present experimental results of classification of musical pieces by genre and of identification of unvoiced fricative phonemes, and in Section 5.4 we conclude the chapter.

5.2 Classification Algorithm

As mentioned in Section 5.1, traditional methods for classification of audio data consist of two main stages: feature extraction, and classification according to these features. We offer to add an intermediate stage to this framework, such that the classification is applied in three steps:

1. Feature extraction - a characteristic vector is defined for each signal in the data set. It captures relevant spectral and temporal features.
2. Manifold learning by diffusion maps - the data is embedded into a lower dimensional non linear manifold using the diffusion maps algorithm, with geometric harmonics extension [22, 23, 53].
3. Classification - the data is classified according to its new parametrization using k-nearest neighbors algorithm.

Each of these steps is described in details:

5.2.1 Feature Extraction

There are some common spectral and temporal features which represent well the character of most audio signals. However, for each classification application there are ceratin features which suit it best. Since each one of the features has its own scale, the features are normalized by their standard deviation, before mapping and classification. We describe here the features which we extract for the two applications of classification described in Section 5.1.

Classifying music by genre

Features that characterize the genre of songs are divided to three main groups, according to Tzanetakis & Cook [88]: timbral texture features, rhythmic content features and pitch content features. We use only the timbral texture features, after empirically recognizing that the other features do not improve the classification results of our method. In the preprocessing stage, each phoneme segment is divided into consecutive non-overlapping short frames (15 ms) which are denoted as "analysis frames", and multiplied by a hamming window. We assume that the signal in such short frames is stationary, and therefore the frequency characteristics of the magnitude spectrum are relatively stable. Each frame is processed separately, and we later use the *mean value* and the *standard deviation* of the features in all the analysis frames for the classification, assuming that the texture features over all of the analysis frames in the song are relatively stable.

The features calculated over the analysis frames are:

1. *Spectral Centroid*: The spectral centroid is defined as the center of gravity of the magnitude spectrum of the STFT

   $$C_t = \frac{\sum_{n=1}^{N}(M_t[n] \cdot n)}{\sum_{n=1}^{N}(M_t[n])}. \tag{5.1}$$

   where $M_t[n]$ is the magnitude of the Fourier transform at the analysis frame t and frequency bin n.

2. *Spectral Rolloff*: The spectral rolloff is defined as the frequency R_t below which p percent of the magnitude distribution is concentrated

   $$\sum_{n=1}^{R_t} M_t[n] = p \cdot \sum_{n=1}^{N} M_t[n]. \tag{5.2}$$

   For this application $p = 85\%$ is used. The rolloff is another measure of the spectral shape.

3. *Spectral Flux*: The spectral flux is defined as the squared difference between the magnitudes of successive spectral distributions

   $$F_t = \sum_{n=1}^{N}(M_t[n] - M_{t-1}[n])^2. \tag{5.3}$$

   The spectral flux is a measure of the amount of local spectral change.

4. *Zero Crossing Rate (ZCR)*:

$$Z_t = \frac{1}{2} \sum_{n=2}^{N} |sign(x[n]) - sign(x[n-1])|. \tag{5.4}$$

where the *sign* function is 1 for positive arguments and -1 for negative arguments, and $x[n]$ is the time domain signal for the analysis frame t. The number of time domain zero crossings provides a measure of the noisiness of the signal.

5. *Mel-Frequency Cepstral Coefficients*: Mel-frequency cepstral coefficients (MFCC) are also based on the STFT. After taking the log-amplitude of the magnitude spectrum, the FFT bins are grouped and smoothed according to the Mel-frequency scaling. A discrete cosine transform is performed on the result, and the first five coefficients are taken.

Another feature is the *Low-Energy Feature.* It is defined as the percentage of analysis frames that have less RMS energy than the average RMS energy. The features are normalized by their standard deviation.

We use the GTZAN [88] data set in order to evaluate the performance of the algorithm. It consists of 30 sec segments, from different genres - blues, classic, country, disco, hiphop, jazz, metal, pop, reggae and rock.

Classifying unvoiced fricative phonemes

The dataset for this study includes more than 1100 isolated phonemes, excerpted from the TIMIT speech database, of both male and female speakers. The phonemes chosen for the analysis are the unvoiced fricatives /s/, /sh/, /f/ and /th/. These phonemes are specifically important since they tend to be indistinguishable for the hearing impaired. In the preprocessing stage, each phoneme segment is divided into consecutive non-overlapping short frames (8 ms) which are denoted as “analysis frames”, and multiplied by a hamming window. The reason for the short length of the frames is twofold: first, the classification of the whole phoneme can be improved by using a majority vote decision, and in addition, it can be used for a real-time application of phoneme spotting. Since the important information of the unvoiced fricatives is contained in the high frequency range, this choice is suitable. The features used to characterize the phonemes are mostly based on the spectral shape, but also on time domain parameters. The features are computed for each analysis frame, and include:

1. *Spectral Peak Locations* [5]: These include frequency locations of the peaks of the spectral envelope.

2. *Spectral Rolloff*: The same as defined for musical genre classification. For this application, p values of 25%, 50% and 75% are used.

3. *Spectral Centroid*: The same as defined for musical genre classification.

4. *Band Energy Ratio*: Band Energy ratio is defined as the ratio of the spectral energies of two bands

$$\widehat{E_t} = 10log_{10}\left(\frac{E_{B_1}}{E_{B_2}}\right) \tag{5.5}$$

   where E_{B_1} and E_{B_2} are the spectral energies of two frequency bands, (here $B_1 = 4 - 8kHz$ and $B_2 = 2 - 4kHz$).

5. *Zero Crossing Rate (ZCR)*: The same as defined for musical genre classification.

6. *Time Domain Zero Crossings Standard Deviation, Skewness and Kurtosis*: These moments of the ZCR are computed using the statistics of the time intervals between consecutive zero crossings.

7. *Mel Frequency Cepstral Coefficients*: Only the first three coefficients are found to be discriminative and they are used for the feature vector.

8. Lacunarity β parameter, as described in [45].

These features are chosen according to their effectiveness in identifying the unvoiced fricatives. The features are normalized by their standard deviation.

5.2.2 The Diffusion Framework

The next stage after feature extraction is mapping the feature vectors to the diffusion space by diffusion mapping. In this subsection we describe and explain this procedure, which was suggested and explained in detail by Coifman and Lafon in [22] and in [23]. In addition, we provide our theoretical justification for it, which is based on former works [21, 53, 65].

Embedding the Data into a Lower Dimensional Manifold

Let (X, A, μ) be a measure space, where A is the σ- algebra of X and μ is a measure function. The set $X = \{\mathbf{x}_i\}_{i=1}^{M}$ is the high-dimensional set of M feature vectors, and μ represents the distribution of the points (where each point is a feature vector) on X. Let $k : X \times X \to R$ be a kernel function representing a notion of similarity between two feature vectors. It is application oriented, chosen to yield meaningful connections, and constitutes our prior definition of the *local* geometry of X. For example, a Gaussian kernel between $\mathbf{x}_i$ and $\mathbf{x}_j$ is the following:

$$k(\mathbf{x}_i, \mathbf{x}_j) = exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\varepsilon}\right). \tag{5.6}$$

In *global* methods for dimensionality reduction, local connections between feature vectors are not taken into account. E.g., in principal component analysis [82] the global correlation matrix is computed. The kernel satisfies, for $(\mathbf{x}_i, \mathbf{x}_j) \in X$:

- It is symmetric: $k(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_j, \mathbf{x}_i)$
- It is point-wise nonnegative: $k(\mathbf{x}_i, \mathbf{x}_j) \geq 0$

The pair (X, k) defines a graph in an Euclidean space, whose vertices are the feature vectors, and the connections between the vertices are weighted by the kernel function. Following classical construction in spectral graph theory [21], a Markov random walk on the graph is defined:

$$p(\mathbf{x}_i, \mathbf{x}_j) = \frac{k(\mathbf{x}_i, \mathbf{x}_j)}{d(\mathbf{x}_i)}. \tag{5.7}$$

where $d(\mathbf{x}_i) = \sum_{j=1}^{M} k(\mathbf{x}_i, \mathbf{x}_j)$. The function p can be considered as the transition probability function of a Markov chain on X, since $\sum_{j=1}^{M} p(\mathbf{x}_i, \mathbf{x}_j) = 1$. The expression $p(\mathbf{x}_i, \mathbf{x}_j)$ represents the probability of transition in one time step from vertex $\mathbf{x}_i$ to vertex $\mathbf{x}_j$. Accordingly, the probability of transition from vertex $\mathbf{x}_i$ to vertex $\mathbf{x}_j$ in t time steps is given by $p_t(\mathbf{x}_i, \mathbf{x}_j)$. Let K denote the matrix corresponding to the kernel function $k(\cdot, \cdot)$, where its $(i, j)_{th}$ element is $k(\mathbf{x}_i, \mathbf{x}_j)$, and let $P = D^{-1}K$ be the matrix corresponding to the function $p(\cdot, \cdot)$ on the data set X, where D is a diagonal matrix with $D_{ii} = d(\mathbf{x}_i)$. Let $\mathbf{X}$ be the matrix consisting of the data samples

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_M]^T. \tag{5.8}$$

Advancing the random walk on the data set a single step forward can be written as $P\mathbf{X}$. Let χ_t denote the Markovian process defined by the transition matrix P. Then, the probabilistic interpretation of a single step is:

$$[P\mathbf{X}_i] = \sum_{j=1}^{M} P_{ij}\mathbf{x}_j = \sum_{j=1}^{M} Pr\{\chi_{t+1} = \mathbf{x}_j, \chi_t = \mathbf{x}_i\}\mathbf{x}_j = \mathbb{E}[\chi_{t+1}|\chi_t = \mathbf{x}_i], \tag{5.9}$$

where $[]_i$ extracts the i_{th} row. Advancing the random walk a single step forward gives the expected values of the random walker starting at feature vector $\mathbf{x}_i$ after a single step. Running the random walk t steps forward can be written as $P^t\mathbf{X}$, since P is a transition matrix. With a smart choice of parameters this process results in revealing the relevant geometric structure of X.

It was shown by Nadler et al. [65] that the random walk P, which is discrete in time and space, converges to the diffusion process, which is continues in time and space, when the number of points approaches infinity ($n \to \infty$) and the time scale approaches zero ($\varepsilon \to 0$).

The distances on the set X which represent the connectivity of the graph in scale t are called the *diffusion distances* and are denoted by $\{D_t\}_{t\in\mathbb{N}}$. If we define the probability distribution $p(t, \mathbf{y}|\mathbf{x})$ as the probability to move from vertex $\mathbf{x}$ to vertex $\mathbf{y}$ in t steps, then the diffusion distance D_t is defined by

$$D_t^2(\mathbf{x}_0, \mathbf{x}_1) = \|p(t, \mathbf{y}|\mathbf{x}_0) - p(t, \mathbf{y}|\mathbf{x}_1)\|_\omega^2 = \sum_{\mathbf{y}}(p(t, \mathbf{y}|\mathbf{x}_0) - p(t, \mathbf{y}|\mathbf{x}_1))^2\omega(\mathbf{y}) \tag{5.10}$$

with the specific choice of $\omega(\mathbf{y}) = 1/\phi_0(\mathbf{y})$ as the weight function, where $\mathbf{x}_0$, $\mathbf{x}_1$ and $\mathbf{y}$ are vertices of the graph, and $\phi_0(\mathbf{y})$ is the (empirical) local density of the vertex $\mathbf{y}$. In other words, $D_t(\mathbf{x}_0, \mathbf{x}_1)$ is a functional weighted L^2 distance between the two posterior distributions $p(t, \mathbf{y}|\mathbf{x}_0)$ and $p(t, \mathbf{y}|\mathbf{x}_1)$. The distance $D_t(\mathbf{x}_0, \mathbf{x}_1)$ will be small if the vertices $\mathbf{x}_0$ and $\mathbf{x}_1$ are highly connected. If we apply eigen-decomposition on the matrix P^t, we obtain the eigenvalues $\{\lambda_i\}_{i\in N}$ and eigenvectors $\{\psi_i\}_{i\in N}$, where N is the number of vertices of the graph.

Now we can define the family of *diffusion maps* $\{\Psi_t\}_{t\in\mathbb{N}}$:

$$\Psi_t(\mathbf{x}_i) \doteq \begin{bmatrix} \lambda_1^t\psi_1(\mathbf{x}_i) \\ \lambda_2^t\psi_2(\mathbf{x}_i) \\ \vdots \\ \lambda_{s(\delta,t)}^t\psi_{s(\delta,t)}(\mathbf{x}_i) \end{bmatrix} \tag{5.11}$$

Each component of $\Psi_t(\mathbf{x}_i)$ is termed a *diffusion coordinate*, and $s(\delta, t)$ is the number of components required to achieve a relative accuracy δ, given the scale t.

Before we head to the main result of this chapter, we first need to prove several lemmas:

Lemma 5.1. *The matrix P has a discrete sequence of non-negative eigenvalues $\{\lambda_l\}_l \geq 0$ and right eigenvectors $\{\psi_l\}_l \geq 0$ such that $1 = \lambda_0 > \lambda_1 \geq \lambda_2 \geq ... \geq 0$ and $P\psi_l = \lambda_l\psi_l$.*

Proof. The matrix P is a *right stochastic matrix*, meaning that its elements are non-negative real numbers, and each row sums to 1. According to the *Perron-Frobenius Theorem*, all the eigenvalues of the matrix P satisfy

$$\{|\lambda_l|\}_l \leq 1. \tag{5.12}$$

We will now show that all the eigenvalues are also non-negative.

The matrix P is defined by $P = D^{-1}K$, where K is the kernel matrix. We shall define the symmetric matrix

$$P_s = D^{-1/2}KD^{-1/2}. \tag{5.13}$$

The matrices P and P_s are similar matrices since

$$P = D^{-1}K = D^{-1/2}\left(D^{-1/2}KD^{-1/2}\right)D^{1/2} = D^{-1/2}P_sD^{1/2}, \tag{5.14}$$

therefore share the same eigenvalues. The kernel matrix K is positive semi definite by definition [81]. Let $v \in \mathbb{R}^N$, v a non-zero vector. We define $u = D^{-1/2}v$. Then

$$v^TP_sv = u^TD^{1/2}P_sD^{1/2}u = u^TD^{1/2}D^{-1/2}KD^{-1/2}D^{1/2}u = u^TKu \geq 0. \tag{5.15}$$

We have proved that P_s is positive semi definite. Therefore, all its eigenvalues are non-negative, and since the matrix P shares the same eigenvalues, its eigenvalues are non-negative as well. Therefore

$$1 = \lambda_0 > \lambda_1 \geq \lambda_2 \geq ... \geq 0. \tag{5.16}$$

□

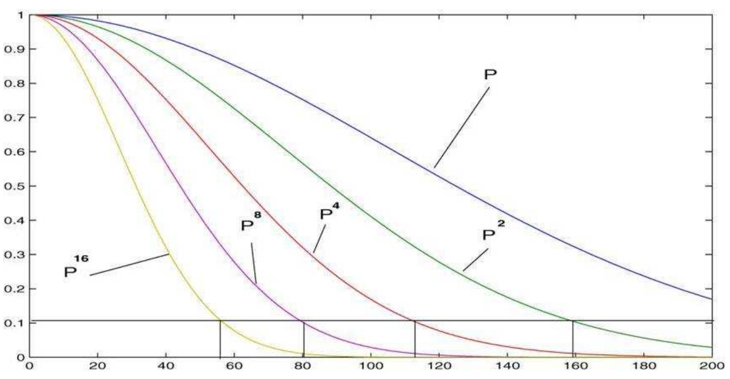


Figure 5.1: The eigenvalues of P raised by different powers t

The eigenvalues of P^t are given by $\lambda_0^t, \lambda_1^t, ..., \lambda_N^t$. Therefore they also satisfy $1 = \lambda_0^t > \lambda_1^t \geq \lambda_2^t \geq ... \geq 0$. The decay of the eigenvalues is faster as P is raised by a larger power t. A plot of the eigenvalues of P raised by different powers is presented in Figure 5.1, which is taken from [22].

Lemma 5.2. *The left and the right eigenvectors of the matrix P, denoted ϕ_i and ψ_j are bi-orthonormal:*

$$< \phi_i, \psi_j >= \delta_{i,j}. \tag{5.17}$$

Proof. The (right) eigenvectors of the symmetric matrix P_s, ν_j, form an orthonormal basis of $\mathbb{R}^N$. The left and right eigenvectors of P are related to those of P_s according to

$$\phi_j = \nu_j^T D^{1/2}, \quad \psi_j = D^{-1/2}\nu_j : \tag{5.18}$$

$$\begin{aligned} P_s\nu_j &= \lambda_j\nu_j \\ D^{1/2}PD^{-1/2}\nu_j &= \lambda_j\nu_j \\ P\,\underbrace{D^{-1/2}\nu_j}_{\psi_j} &= \lambda_j\,\underbrace{D^{-1/2}\nu_j}_{\psi_j} \end{aligned}$$

and since P_s is symmetric:

$$\begin{aligned} \nu_j^T P_s &= \lambda_j\nu_j^T \\ \nu_j^T D^{1/2}PD^{-1/2} &= \lambda_j\nu_j^T \\ \underbrace{\nu_j^T D^{1/2}}_{\phi_j} P &= \lambda_j\,\underbrace{\nu_j^T D^{1/2}}_{\phi_j}. \end{aligned}$$

A multiplication of the vector ϕ_i by ψ_j shows that they are bi-orthonormal. $\square$

The matrix P^t has the same eigenvectors as the matrix P. The matrix P can be decomposed using eigen decomposition: $P = Q\Lambda Q^{-1}$ (or similarly for P^t), such that the columns of the matrix Q are the right eigenvectors of P, and Λ is a diagonal matrix whose diagonal consists of the corresponding eigenvalues. According to lemma 5.2, ϕ_i and ψ_j are bi-orthonormal, and the matrix Q^{-1} can be replaced by a matrix whose columns are the left eigenvectors of P. If we write it element-wise:

$$p(t, \mathbf{y}|\mathbf{x}) = \sum_{j\geq 0} \psi_j(\mathbf{x})\lambda_j^t\phi_j(\mathbf{y}) = \phi_0(\mathbf{y}) + \sum_{j\geq 1} \psi_j(\mathbf{x})\lambda_j^t\phi_j(\mathbf{y}). \tag{5.19}$$

The element $\psi_0(\mathbf{x}) = 1$, since $\psi_0 = \mathbf{1}$ (a vector of ones) - this is the right eigenvector corresponding to $\lambda_0 = 1$, since the sum of the rows in P equals 1. Furthermore, ϕ_0, which is the left eigenvector corresponding to $\lambda_0 = 1$, is the stationary distribution of the Markov walk. For ε large enough all the points in the graph are connected, so that P is an irreducible and aperiodic Markov chain. This means that regardless of the initial starting point $\mathbf{x}$,

$$\lim_{t\to\infty} p(t, \mathbf{y}|\mathbf{x}) = \phi_0(\mathbf{y}). \tag{5.20}$$

Explicitly, it is given by

$$\phi_0(\mathbf{x}_i) = \frac{D_{i,i}}{\sum_j D_{j,j}}, \tag{5.21}$$

which means that $\phi_0(\mathbf{x}_i)$ also has an interpretation of the density estimate (empirical local density) at vertex $(\mathbf{x}_i)$.

We now introduce the main result of this chapter:

Theorem 5.1. *The diffusion map $\Psi_t : X \to \mathbb{R}^{s(\delta,t)}$ embeds the data set into an Euclidean space of $s(\delta, t)$ dimensions, so that in this space, the Euclidean distance is equal to the diffusion distance up to the relative accuracy δ, or equivalently*

$$D_t(\mathbf{x}_0, \mathbf{x}_1) = \left(\sum_{l=1}^{s(\delta,t)} \lambda_l^{2t}(\psi_l(\mathbf{x}_0) - \psi_l(\mathbf{x}_1))^2\right)^{\frac{1}{2}} = \|\Psi_t(\mathbf{x}_0) - \Psi_t(\mathbf{x}_1)\|_2 \tag{5.22}$$

Proof. Substituting the relation (5.19) in equation (5.10), we get

$$D_t^2(\mathbf{x}_0, \mathbf{x}_1) = \sum_{\mathbf{y}} \left(\sum_{j\geq 1} \lambda_j^t \left(\psi(\mathbf{x}_0) - \psi(\mathbf{x}_1)\right)\phi_j(\mathbf{y})\right)^2 \frac{1}{\phi_0(\mathbf{y})}. \tag{5.23}$$

Expanding the brackets and changing the order of summation gives

$$D_t^2(\mathbf{x}_0,\mathbf{x}_1) = \sum_{i,j\geq 1} \lambda_i^t(\psi_i(\mathbf{x}_0) - \psi_i(\mathbf{x}_1))\lambda_j^t(\psi_j(\mathbf{x}_0) - \psi_j(\mathbf{x}_1)) \sum_{\mathbf{y}} \frac{\phi_i(\mathbf{y})\phi_j\mathbf{y}}{\phi_0(\mathbf{y})}. \tag{5.24}$$

From relations (5.18), we get that

$$\frac{\phi_j(\mathbf{y})}{\phi_0(\mathbf{y})} = \frac{\nu_j^T(\mathbf{y})D^{1/2}(\mathbf{y},\mathbf{y})}{\nu_0^T(\mathbf{y})D^{1/2}(\mathbf{y},\mathbf{y})} = \frac{\nu_j(\mathbf{y})}{\nu_0(\mathbf{y})} = \frac{D^{-1/2}(\mathbf{y},\mathbf{y})\nu_j(\mathbf{y})}{D^{-1/2}(\mathbf{y},\mathbf{y})\nu_0(\mathbf{y})} = \frac{\psi_j(\mathbf{y})}{\psi_0(\mathbf{y})} = \psi_j(\mathbf{y}), \tag{5.25}$$

where we use the fact that the matrix D is diagonal, and the last equivalence stems from the fact that $\psi_0 = \mathbf{1}$. Now, the element $\sum_{\mathbf{y}} \frac{\phi_i(\mathbf{y})\phi_j\mathbf{y}}{\phi_0(\mathbf{y})}$ turns into

$$\sum_{\mathbf{y}} \frac{\phi_i(\mathbf{y})\phi_j\mathbf{y}}{\phi_0(\mathbf{y})} = \sum_{\mathbf{y}} \phi_i(\mathbf{y})\psi_j(\mathbf{y}) =< \phi_i, \psi_j >= \delta_{i,j}. \tag{5.26}$$

Therefore, equation (5.24) turns into:

$$D_t^2(\mathbf{x}_0,\mathbf{x}_1) = \sum_{i,j\geq 1} \lambda_i^t(\psi_i(\mathbf{x}_0) - \psi_i(\mathbf{x}_1))\lambda_j^t(\psi_j(\mathbf{x}_0) - \psi_j(\mathbf{x}_1)) \cdot \delta_{i,j} = \left(\sum_{j\geq 1} \lambda_j^{2t}(\psi_j(\mathbf{x}_0) - \psi_j(\mathbf{x}_1))^2\right). \tag{5.27}$$

Since the eigenvalues $\lambda_1, \lambda_2, ...$ tend to 0 and have a modulus strictly less than 1 (according to lemma 5.1), the above sum can be computed to a preset accuracy $\delta > 0$ with a finite number of terms: We define $s(\delta, t) = max\{l \in \mathbb{N} \text{ such that } |\lambda_l|^t > \delta|\lambda_1|^t\}$. Then, up to the precision δ we have

$$D_t(\mathbf{x}_0,\mathbf{x}_1) = \left(\sum_{l=1}^{s(\delta,t)} \lambda_l^{2t}(\psi_l(\mathbf{x}_0) - \psi_l(\mathbf{x}_1))^2\right)^{\frac{1}{2}} \tag{5.28}$$

as required. □

As t increases, the spectrum decay is faster, and $s(\delta, t)$ required for an accuracy δ can be smaller.

The importance of the theorem proved here, is that the diffusion mapping maps the diffusion distances to Euclidean distances. The Euclidean distances between the diffusion coordinates, which are the output of this mapping, are equivalent to the diffusion distances between the original feature vectors. Therefore, we expect a classification method which is based on Euclidean distances between the samples, to perform better after the diffusion mapping.

We propose a modified Gaussian kernel $k(\mathbf{x}, \mathbf{y})$:

$$k(\mathbf{x}, \mathbf{y}) = exp(-||(\mathbf{x} - \mathbf{y})./\sigma||)^2 \tag{5.29}$$

where σ is a *vector* that consists of elements proportional to the standard deviations of each of the features and "./" is an element-wise division. The element-wise division of $(\mathbf{x} - \mathbf{y})$ by σ leads to a *multi scale* embedding (which is the same as normalizing the features by their standard deviation).

A disadvantage of the diffusion maps method, is its sensitivity to the choice of the parameters. The parameters have been chosen after a grid search optimization procedure - the power of the matrix P is $t = 1$, and the dimension of the diffusion space is $s(\delta, t) = 10$. Therefore, the family of diffusion maps is reduced to:

$$\Psi(x) = \begin{bmatrix} \lambda_1 \psi_1(\mathbf{x}) \\ \lambda_2 \psi_2(\mathbf{x}) \\ \vdots \\ \lambda_{10} \psi_{10}(\mathbf{x}). \end{bmatrix} \tag{5.30}$$

Out-of-sample extension

The parametrization described in the previous subsection is conducted on a limited data set which is sampled from the training set, denoted as the "reference set", to maintain a limited computational complexity despite of the high complexity required from the eigen decomposition. The rest of the training data set, as well as new samples from the testing data, are mapped to the diffusion space by an extension method named "geometric harmonics" ([53], [23]). The geometry of the reference set has to represent well the geometry of the whole data set, therefore the reference set has to be sampled approximately evenly from the training data set.

Following "Nyström extension" [40], if we denote the reference set as X, then when a new feature vector $\bar{\mathbf{x}}$ from the rest of the training set or from the testing set arrives, its extended j_{th} eigenvector is calculated as follows:

$$\bar{\psi}_j(\bar{\mathbf{x}}) = \frac{1}{\lambda_j} \sum_{\mathbf{x} \in X} p(\mathbf{x}, \bar{\mathbf{x}}) \psi_j(\mathbf{x}). \tag{5.31}$$

The extended eigenvector is a weighted sum of the original eigenvectors, where the weights are proportional to the probabilities of transition between the original feature vectors to the new one.

The new family of diffusion maps for each new feature vector $\bar{\mathbf{x}}$ is:

$$\bar{\Psi}(\bar{\mathbf{x}}) = \begin{bmatrix} \lambda_1 \bar{\psi}_1(\bar{\mathbf{x}}) \\ \lambda_2 \bar{\psi}_2(\bar{\mathbf{x}}) \\ \vdots \\ \lambda_{10} \bar{\psi}_{10}(\bar{\mathbf{x}}) \end{bmatrix} \tag{5.32}$$

This extension allows performing the eigen-decomposition of the probability matrix only on the reference set.

Classification

A new signal from the testing set (song or phoneme) is classified using supervised learning, specifically by the k-nearest neighbors (k-NN) method, with k=5. If we denote the training set as $\bar{\mathbf{X}}$ and the testing set as $\tilde{\mathbf{X}}$, then each feature vector from the testing set $\tilde{\mathbf{x}} \in \tilde{\mathbf{X}}$ is classified according to the classification of its diffusion map $\{\tilde{\Psi}(\tilde{\mathbf{x}})\}$. Its diffusion map $\{\tilde{\Psi}(\tilde{\mathbf{x}})\}$ is classified according to the closest k nearest neighbors from the family of diffusion maps of the training set $\{\bar{\Psi}(\bar{\mathbf{x}})\}$. The measure distance for the k-NN is the Euclidean distance, which is a reasonable choice, based on the equivalence between the diffusion distances to the Euclidean distances in the diffusion space. We use k-nearest neighbors as the classifier because of its simplicity and because it is based on Euclidean distances.

For visualization, the embedding of the feature vectors of the classical and metal genres to a 2D mapping is shown in Figure 5.2, and that of the different phonemes in Figure 5.3. A good separation can be seen even in 2 dimensions, though we used a 10-dimensional embedding.

5.3 Experimental results

We now present the accuracy of classification for both applications, obtained after a 10-fold cross validation procedure. The training set and the testing set are randomly chosen. The parameters involved in the algorithm - the scale of embedding t, the number of dimensions

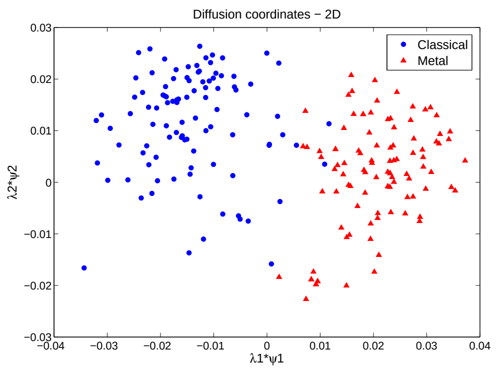


Figure 5.2: Diffusion coordinates (2D) of the classical and metal feature vectors

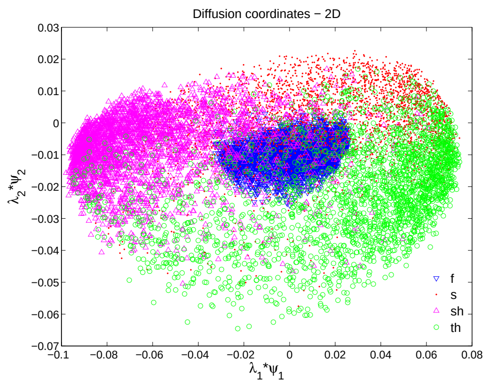


Figure 5.3: Diffusion map (2D) of the unvoiced fricatives feature vectors. The phonemes are marked by: /f/ - blue, /s/ - red, /sh/ - magenta, /th/ - green.

in the embedding $s(\delta, t)$, the radius of the Gaussian kernel σ (multi-dimensional and proportional to the standard deviation of each feature) and the number of neighbors for the k-nearest neighbors classification method, are determined by a grid search optimization procedure.

5.3.1 Classifying music by genre

The GTZAN dataset [88] is used to evaluate the performance of the algorithm. It consists of 1000 songs of 10 different genres, 100 from each genre: blues, classic, country, disco, hiphop, jazz, metal, pop, reggae and rock. The results presented here are the average and standard deviation of 100 iterations.

First, in order to evaluate the feasibility of the algorithm, we classify two distinct musical genres from GTZAN dataset - metal and classic (100 songs from each genre). The accuracy of the classification after the diffusion maps is $96.8 \pm 3.8\%$. Then, we classify the whole data set (10 different genres). The accuracy of classification after the diffusion maps is $56.3 \pm 4.6\%$.

The confusion matrix for classification of 10 genres with the diffusion maps is presented in Table 5.1. The names of genres without quotation marks represent the true genres, and those with the quotation marks represent the identified genres. From the table we see that the highest classification percentages are given to the correct genres in all cases. In addition, naturally, the largest confusion is between similar genres, e.g., rock and country (19% of the rock songs are classified as country).

Next, we examine the classification of 5 genres only - blues, classical, metal, pop, reggae. The accuracy of classification after the diffusion maps is $83.9 \pm 4.8\%$. The confusion matrix for the classification using the diffusion maps coordinates is presented in Table 5.2. Here the classification results are much better, and most of the songs from each genre are identified correctly.

In the next experiment we cluster the songs into pairs of genres - blues & country, classical & jazz, metal & rock, pop & hiphop and disco & reggae. The accuracy of classification after the diffusion maps is $65.0 \pm 4.3\%$. The confusion matrix is presented in Table 5.3. Most of the songs from each genre are identified correctly as well.

It is important to compare the classification with diffusion maps, to classification with-

Table 5.1: The Average Confusion Matrix Using Diffusion Coordinates - 10 Genres

	"Blues"	"Classic"	"Country"	"Disco"	"Hiphop"	"Jazz"	"Metal"	"Pop"	"Reggae"	"Rock"
Blues	**0.60**	0.00	0.09	0.04	0.05	0.02	0.07	0.00	0.04	0.10
Classic	0.01	**0.81**	0.05	0.01	0.00	0.08	0.01	0.00	0.00	0.03
Country	0.07	0.02	**0.48**	0.05	0.01	0.11	0.00	0.06	0.07	0.13
Disco	0.02	0.00	0.08	**0.37**	0.11	0.03	0.07	0.12	0.06	0.14
Hiphop	0.04	0.00	0.02	0.08	**0.52**	0.00	0.02	0.13	0.16	0.02
Jazz	0.06	0.12	0.10	0.04	0.00	**0.54**	0.01	0.04	0.02	0.07
Metal	0.06	0.00	0.01	0.08	0.02	0.00	**0.73**	0.00	0.01	0.10
Pop	0.00	0.00	0.07	0.08	0.06	0.02	0.00	**0.68**	0.08	0.03
Reggae	0.05	0.00	0.07	0.05	0.07	0.01	0.01	0.07	**0.62**	0.05
Rock	0.06	0.00	0.19	0.09	0.04	0.10	0.11	0.05	0.05	**0.30**

Table 5.2: The Average Confusion Matrix Using Diffusion Coordinates - 5 Genres

	"Blues"	"Classic"	"Metal"	"Pop"	"Reggae"
Blues	**0.85**	0.03	0.09	0.00	0.03
Classic	0.04	**0.91**	0.02	0.01	0.02
Metal	0.11	0.00	**0.88**	0.01	0.00
Pop	0.00	0.01	0.01	**0.88**	0.10
Reggae	0.12	0.01	0.01	0.13	**0.73**

Table 5.3: The Average Confusion Matrix Using the Diffusion Coordinates - After Clustering to 5 groups of Pairs

	"Blues & Country"	"Classical & Jazz"	"Metal & Rock"	"Pop & Hiphop"	"Disco & Reggae"
Blues & Country	**0.64**	0.08	0.14	0.04	0.10
Classical & Jazz	0.12	**0.75**	0.07	0.03	0.03
Metal & Rock	0.14	0.06	**0.63**	0.05	0.12
Pop & Hiphop	0.06	0.01	0.05	**0.69**	0.19
Disco & Reggae	0.12	0.02	0.13	0.20	**0.53**

Table 5.4: The classification results of musical pieces by genres, using k-NN method, with different dimensionality reduction types

	Dimensionality reduction type				
Number of classes	No dimensionality reduction	PCA	**Diffusion Maps**	Laplacian Eigenmaps	LLE
10	$61.9 \pm 4.4\%$	$60.3 \pm 4.5\%$	$\mathbf{56.3 \pm 4.6\%}$	$51.7 \pm 4.5\%$	$55.8 \pm 4.3\%$
5	$87.5 \pm 4.1\%$	$86.7 \pm 4.5\%$	$\mathbf{83.4 \pm 4.5\%}$	$83.2 \pm 5.0\%$	$85.5 \pm 4.7\%$
2	$98.4 \pm 2.6\%$	$97.8 \pm 2.7\%$	$\mathbf{97.5 \pm 3.2\%}$	$91.0 \pm 6.1\%$	$96.3 \pm 4.1\%$
5 groups of pairs	$69.1 \pm 4.0\%$	$67.4 \pm 4.3\%$	$\mathbf{65.7 \pm 4.3\%}$	$60.3 \pm 4.0\%$	$65.7 \pm 4.5\%$

out diffusion maps (using the feature vectors), and to classification with other methods of dimensionality reduction. Based on the theoretical analysis in Subsection 5.2.2, we expect that the mapping by diffusion maps would improve the classification results. In the same section we also mentioned the theoretical advantage over PCA, which is a linear and global method.

The comparison is performed in Table 5.4, where the classification is performed by k-NN. We can see that in practice, the classification results with diffusion maps are not better than classification without dimensionality reduction, nor than with PCA. A possible reason, is that the assumption that the feature vectors lie on a non-linear manifold doesn't hold true in this case. Perhaps it holds true if a wider set of features, which were not chosen wisely as in our case, are used for classification.

In Table 5.4 we also compare the classification results to other manifold learning algorithms - the locally linear embedding (LLE) algorithm and the Laplacian eigenmaps algorithm. They were both implemented using the Matlab Toolbox for Dimensionality Reduction, by L. van der Maaten (`http://homepage.tudelft.nl/19j49/Home.html`). We can see from the table that the LLE and Laplacian eigenmaps perform quite similarly to Diffusion maps.

The similarity in the classification results between the three manifold learning algorithms here, coincides with the comparison conducted by Wittman (`http://www.math.ucla.edu/~wittman/mani`). Wittman have created a GUI for comparison between different methods of dimensionality reduction, tested on different types of synthetic data, such as Swiss Roll, Swiss Hole, Twin Peaks and more. It turned out that non of the non linear manifold learning techniques had a significant advantage over

Table 5.5: The classification results of musical pieces to 10 genres with dimensionality reduction using "diffusion maps", with different classification types

	K-NN	QDA	LDA	SVM	QDA dimension=40	LDA dimension=50
With diffusion maps	$56.3 \pm 4.6\%$	$56.2 \pm 3.7\%$	$56.0 \pm 4.4\%$	$47.8 \pm 5.4\%$	$63.5 \pm 4.8\%$	$63.6 \pm 4.4\%$
Without diffusion maps	$61.9 \pm 4.4\%$	$61.9 \pm 4.2\%$	$61.2 \pm 4.1\%$	$56.5 \pm 5.2\%$	$61.9 \pm 4.2\%$	$61.2 \pm 4.1\%$

the others in inferring the geometry of the manifolds, and their success depended on the shape of the specific synthetic data. He found out that the diffusion maps method had an advantage of fast convergence, and a disadvantage of high sensitivity to parameters.

Now we examine adding the stage of diffusion maps to other classification techniques, instead of k-NN. We apply the diffusion maps as a preliminary stage for the linear discriminant analysis (LDA), quadratic discriminant analysis (QDA) and support vector machine (SVM) supervised classification techniques. We use a multi-class SVM open source toolbox, called osu-svm (http://sourceforge.net/projects/svm), with a radial basis function, and parameters optimized by a grid search optimization procedure. The results of classification of 10 genres, after 100 iterations, are presented in Table 5.5.

We can see from the table that for other classification methods, the embedding using diffusion maps doesn't improve the results as well. In SVM this can be explained by the non-linearity of the classification method itself, and therefore a non-linear embedding before classification is unnecessary and even deteriorating.

When raising the dimension of the diffusion space for LDA and QDA to 40 and 50 dimensions respectively (after an optimization procedure), the results using diffusion maps are improved, and are also somewhat better than those without using diffusion maps. Raising the dimension for k-NN or for SVM does not improve the classification results. SVM with a linear kernel leads to poor classification results, and the diffusion mapping does not improve them as well. The classifiers LDA and QDA perform better than k-NN for this data, maybe due to the fact that k-NN is a more naive classifier.

We compare the classification results with LDA and QDA when raising the dimension of the diffusion mapping, to that of classification with no mapping of the feature vectors, in Tables 5.6 and 5.7. The classification results using the diffusion maps in this case are

Table 5.6: The classification results of musical pieces by genres, using LDA, with high-dimensional diffusion mapping

	Mapping type	
Number of classes	No mapping	**Diffusion Maps - 50 dimensions**
10	$61.2 \pm 4.1\%$	$\mathbf{63.6 \pm 4.4\%}$
5	$85.2 \pm 5.0\%$	$\mathbf{86.8 \pm 5.0\%}$
2	$98.9 \pm 2.2\%$	$\mathbf{97.9 \pm 3.1\%}$
5 groups of pairs	$66.6 \pm 4.4\%$	$\mathbf{69.2 \pm 4.2\%}$

Table 5.7: The classification results of musical pieces by genres, using QDA, with high-dimensional diffusion mapping

	Mapping type	
Number of classes	No mapping	**Diffusion Maps - 40 dimensions**
10	$61.9 \pm 4.2\%$	$\mathbf{63.5 \pm 4.8\%}$
5	$84.7 \pm 4.8\%$	$\mathbf{84.2 \pm 5.4\%}$
2	$99.0 \pm 2.0\%$	$\mathbf{98.6 \pm 2.4\%}$
5 groups of pairs	$67.9 \pm 3.9\%$	$\mathbf{69.9 \pm 3.6\%}$

somewhat better than without the mapping.

In conclusion, for the task of musical genre classification, with this set of features, most of the songs were classified correctly, and the best classification results are achieved with LDA and QDA, after diffusion mapping to a *higher* dimension.

We also attempt to change the set of features and to follow the logic of former methods [7, 49, 53, 65], which used very long feature vectors . The very high dimensionality of the feature vectors does not allow classification without dimensionality reduction. We produce a long feature vector by concatenation of the energy of all the time-frequency bins of a song, where the frequency bins were spaced according to the critical frequency bands [79]. This produces a very long feature vector, but makes the classification results worse than before.

Another attempt is, similarly to Keller et al. [49], to apply discrete cosine transform

Table 5.8: Classification results of musical pieces by genres in former works

Reference	**Accuracy**
Panagakis et al. [70]	91%
Bergstra et al. [11]	82.5%
Li et al. [55]	78.5%
Lidy et al. [56]	76.8%
Benetos et al. [10]	75.0%
Holzapfel et al. [48]	74.0%
Tzanetakis et al. [88]	61.0%

(DCT) on the FFT of consecutive overlapping 32 ms frames, and to choose the features as the mean values and the standard deviations of these FFT bins over all the frames. This also doesn't improve the classification results and leads to worse results than with PCA.

In [70], there is an overview of notable results of music genre classification (of 10 genres) on the same data set (GTZAN data set). Some of the works were described in Chapter 1. Most of the results are better than ours, ranging from 61% to 91%. However, the diffusion maps method has several advantages over other methods, which include simplicity - both of the spectral analysis for the transition matrix, and of the classification. Other advantages are low computational complexity and high speed of convergence, which are achieved using the out of sample extension by geometric harmonics. The classification results of former works are presented in Table 5.8.

5.3.2 Classifying unvoiced fricative phonemes

The evaluation of the performance is conducted on the TIMIT database. From preliminary experiments it turns out that the results for male and female speakers are very similar. Therefore, for simplicity, the results presented here are only for male speakers. 254 phonemes are extracted for each of the four unvoiced fricative phonemes. Each of the phonemes is segmented to consecutive non-overlapping short frames of $8ms$, leading to approximately 10000 frames from each type of phoneme. 15-dimensional Feature vectors

Table 5.9: The confusion matrix for speech male data using K-NN - according to a majority based decision (left) and for segments (right)

	"[f]"	"[s]"	"[sh]"	"[th]"
[f]	**0.87**	0.01	0.03	0.09
[s]	0.02	**0.87**	0.11	0.00
[sh]	0.06	0.09	**0.85**	0.00
[th]	0.27	0.06	0.01	**0.66**

	"[f]"	"[s]"	"[sh]"	"[th]"
[f]	**0.70**	0.04	0.06	0.20
[s]	0.07	**0.68**	0.16	0.09
[sh]	0.10	0.09	**0.79**	0.02
[th]	0.34	0.15	0.04	**0.47**

are then produced, one for each frame.

The classification is conducted according to two options: In the first, each frame is classified independently. This method is useful for future real-time classification systems. In the second, each phoneme is classified in a majority-based decision, according to the classification of most of its frames. 100 iterations are conducted for each case. We classify using k-NN with k=5 , and compare the classification before and after the diffusion mapping. In classification of each analysis frame separately, the average identification results of the algorithm for the data with the diffusion maps is $66.6 \pm 2.1\%$, and in a majority-based classification, an accuracy of $78.6 \pm 5.4\%$ is obtained with the diffusion maps.

The confusion matrices of the results using k-NN with the diffusion maps for the majority-based decision and for separate segments are presented in Table 5.9.

We can see from the table that both in classification of phonemes and in classification of segments, the highest classification percentage is given to the correct phoneme in all cases. The largest confusion is between /f/ and /th/.

The classification results of phonemes based on majority decision, with a comparison to PCA, are presented in Table 5.10.

For this application as well, the classification results with diffusion maps are not better than classification without dimensionality reduction, nor than with PCA. This may be explained similarly to the results of musical genre classification - the assumption that the feature-vectors lie on a non-linear manifold is inaccurate, since the set of features is limited and well-chosen.

The classification results of phonemes based on majority decision, with a comparison

Table 5.10: The majority based classification results of phonemes with a comparison to other dimensionality reduction methods

	Dimensionality reduction type		
	No dimensionality reduction	**Diffusion maps**	PCA
Classification of segments	$68.8 \pm 2.1\%$	$\mathbf{66.6 \pm 2.1}\%$	$68.6 \pm 2.0\%$
Classification of phonemes - majority-based decision	$81.0 \pm 5.3\%$	$\mathbf{78.6 \pm 5.4}\%$	$81.0 \pm 5.3\%$

Table 5.11: The majority based classification results of phonemes with a comparison to other classification methods

	K-NN	QDA	LDA	SVM
With diffusion maps	$78.6 \pm 5.4\%$	$78.2 \pm 6.6\%$	$80.7 \pm 5.6\%$	$56.1 \pm 5.4\%$
Without diffusion maps	$81.0 \pm 5.3\%$	$79.4 \pm 5.8\%$	$81.1 \pm 5.7\%$	$66.5 \pm 4.2\%$

to other classification techniques are presented in Table 5.11.

From the table we can see that with LDA, SVM and QDA, most of the phonemes were classified correctly, though, here as well, the classification results are not improved comparing to classification based on the feature vectors. Raising the dimension of the diffusion space for k-NN, LDA, SVM or QDA does not improve the results in this case.

Other comparable results, though derived from a somewhat different data set from TIMIT, are by Frid and Lavner [41]. They reported an overall accuracy of 85% in classification of the same type of phonemes, and their major improvement comparing to our results is in the identification of /f/ from /th/.

5.4 Summary

We have shown theoretically the logic behind the use of the manifold learning method named "diffusion maps" as a preliminary stage for classification of audio signals. Our assumption is that temporal and spectral feature vectors of natural audio signals lie on non-linear manifolds, and therefore traditional classification methods based on Euclidean distances between the feature vectors do not capture their intrinsic geometry.

We proved theoretically the equivalence of the diffusion distances between the feature vectors on the manifold to the Euclidean distances between the diffusion coordinates in the space defined by the diffusion mapping. In addition, we suggested a framework for classification of audio data, in which the diffusion maps is an intermediate stage, which makes classification according to Euclidean distances more meaningful.

We examined empirically classification using diffusion maps, and showed that it is applicable, while gaining the advantage of dimensionality reduction. However, in most tests, adding the stage of diffusion maps didn't improve the classification results. The only case in which there was an improvement, was in classification of musical pieces by genre - when applying diffusion maps with LDA and *raising* the dimension of the mapping. This may implement that in this application the feature vectors lie on a non-linear *high-dimensional* manifold, since the number of chosen features was not enough.

We can explain the incompatibility between theory and practice, by the fact that in both applications described above, the chosen features were limited, distinguishing and effective for classification. This probably led to low redundancy between them, and weakened the assumption that the features lie on a non linear manifold. In addition, this assumption might be true for other types of data and should be tested for more applications.

It can be interesting to check the correctness of this assumption by evaluating to which degree the chosen features lie on a low dimensional manifold. Such evaluation is a suggestion for future research, and is not conducted under the scope of this work.

Manifold learning techniques, including diffusion maps, might be more effective in a case where the set of features is larger, not wisely chosen, and more redundant.

We also showed that diffusion maps performs similarly to other manifold learning methods for musical genre classification. Following the conclusions of Wittman (`http://www.math.ucla.edu/~wittman/mani`) on synthetic data sets, we can assume that the efficiency of diffusion maps comparing to other manifold learning techniques, when classifying audio data, depends on several factors, including the application and the data.

Chapter 6

Conclusion

6.1 Summary

We have addressed two problems in audio signal processing - Transcription of polyphonic music and classification of audio and speech data. We deal with both problems using algorithms that represent the data in a meaningful manner, in a sense that the problems can be solved more efficiently and easily using these representations.

For transcription of polyphonic music we developed an algorithm based on sparse representations. The power spectrum of the music signal at each time window is represented as a multiplication of a learned dictionary with a sparse vector. This framework is based on the fact that the number of notes played at a time is small comparing to the number of available notes, and on the fact that assuming random phase relationships, the power spectrum of different notes approximately add.

We offered a parametric dictionary, namely "Musically-Structured (MS) dictionary", based on the common features of the power spectra of music signals. This parametric dictionary is more suitable for transcription than an analytic dictionary or an explicit dictionary. Its advantage over an analytic dictionary is due to its adaptiveness to the data, which is expressed in learning the timbre of the signal. Its advantage over an explicit dictionary is twofold - it avoids over-fitting to the data in the sense that it is able to identify notes even if they are not played individually, or only played seldom, and it is more computationally efficient.

We developed modifications of two dictionary learning algorithms, MOD and K-SVD, which are denoted MS-MOD and MS-K-SVD, for learning the parametric dictionary. The identification of the notes is performed by sparse coding using OMP.

The performance of the MS-dictionary was examined empirically on MIDI synthesized music and on recorded music. In the polyphonic pieces we examined, most of the notes where recognized correctly. Relatively good performance was seen also in cases of small data sets with overlapping harmonics, such as chords or octaves. It is shown that the transcription using MS-dictionary is better than using an analytic or an explicit dictionary. The advantage over an explicit dictionary grows as the data set is smaller, and as there are more overlapping harmonics.

The performance was also compared to that of other algorithms for transcription of polyphonic music, and it was tested over a set of polyphonic classical piano music, both synthesized and recorded. The performance of our transcription algorithm is similar to that of other methods, though the comparison is somewhat biased, since our algorithm is unsupervised and the compared methods are supervised and use a training set for the transcription.

For classification of audio and speech data we used a manifold learning method named "diffusion maps". This mapping is added as an intermediate stage to traditional classification methods. Such methods apply feature extraction followed by classification, which is based on Euclidean distances between the feature vectors. We assume that the feature vectors lie on a non-linear manifold, therefore should be mapped to an Euclidean space before the classification.

In this method, a diffusion process is defined on a graph which was built from the feature vectors. A probability matrix representing this diffusion process is defined, and its weighted eigenvectors are used for the mapping. We showed theoretically that the Euclidean distances between the diffusion maps are equal (up to a relative accuracy) to the diffusion distances between the feature vectors, which represent their connectivity on the graph. Thus, classification based on Euclidean distances is more meaningful after the mapping.

We examined the performance of this method for two applications: classification of musical pieces by genre, and identification of unvoiced fricative phonemes. First, spectral and temporal features were extracted from short time windows of the signals, then mapping using diffusion maps was applied, and finally classification using the mapped feature vectors was performed.

The overall classification accuracy with k-nearest neighbors of 10 musical genres is $56.3 \pm 4.6\%$, meaning that most of the songs are classified correctly, and we also achieve dimensionality reduction. The main confusion of the algorithm is between similar genres such as rock and country. However, the results are not better than classification using the features without mapping, neither than classification after PCA. There is an improvement with diffusion maps to a higher dimensional space, with classification by LDA. The classification results using diffusion maps is similar to classification with Laplacian eigenmaps and LLE.

The overall classification accuracy with k-nearest neighbors of the phonemes /s/,/sh/,/f/ and /th/ is $78.6 \pm 5.4\%$, where also here most of the phonemes are identified correctly. Similarly to the previous application, the results do not outperform classification using the features nor using dimensionality reduction with PCA. The main confusion is between /th/ and /f/, which are both non-sibilants, and share many common features.

When applying the diffusion maps as an intermediate stage for other classification algorithms, which include SVM, LDA and QDA, it doesn't improve the classification results as well.

The possible reasons for the lack of improvement using diffusion maps for classification in the experiments conducted here, are as follows:

- The assumption that the features lie on a non-linear manifold is inaccurate when the set of features is limited, and when they are well-chosen in advance. Choosing a more redundant set of features might emphasize the advantage of diffusion maps.
- The assumption that the features lie on a non-linear manifold, or that they follow the diffusion framework, depends on the application. Perhaps for other applications of audio and speech classification diffusion maps does improve the classification accuracy (e.g., see [49]).
- The method of diffusion maps highly depends on the chosen parameters and on the kernel, which might have been chosen inappropriately for these applications.

6.2 Future research

We shall divide this section to future research in transcription of polyphonic music, and to future research in audio and speech classification.

Transcription of polyphonic music

Several suggestions for interesting modifications and expansions:

1. A dictionary which is more suitable for recorded music - in Chapter 3 we mentioned an important difference between synthesized music and recorded piano music - strings in a real piano do not act as ideal strings, leading to inharmonicity of the overtones. The MS-dictionary is defined such that the overtones are exact multiple integers of the fundamental frequency (i.e., perfect harmonics), and this might be inaccurate in some cases. A way to overcome this problem is to define the initial structure of the dictionary differently, or to train the dictionary on all the notes of the piano before transcription (although each piano needs its own training and then the transcription will turn to a supervised method).

2. Exploit high-level information - Former works [17], [73] used prior information on the structure, tempo, rhythm and key of the musical piece, as well as expected relations between consecutive notes. This might improve the transcription performance if integrated into our framework.

3. Expand the number of atoms in the dictionary - The timbre of the sound changes as a function of its intensity. Therefore, it is reasonable that each note would be represented by several atoms, where each atom represents a different intensity.

4. Robustness to noise - The algorithm of transcription was tested only on clean signals, and suppression of noise was not taken into account.

5. Transcription of several instruments - we performed only transcription of piano music. A future step would be to expand it to several instruments playing simultaneously. Since each instrument has its own timbre, a different atom in the dictionary should be allocated for each instrument.

6. Integration with classification of audio - notes identified by the transcription framework can be used as features for classification of musical pieces.

Classification of audio and speech data

For this application we also offer several expansions:

1. Choose feature vectors with higher redundancy - we used features which were empirically proven to be efficient for classification. In former works [49], there was an improvement using diffusion maps on much higher-dimensional features set, implying that it can be the same for the examined applications, if the set of features will be chosen appropriately.

2. Evaluate to which degree the feature vectors lie on a non linear manifold - this might assist in predicting or explaining success (or failure) of manifold learning techniques as an intermediate stage in classification.

3. Extracting more features - we tried to add pitch and rhythm content features, as well as higher dimensional feature vectors for musical genre classification, and additional features for the identification of the unvoiced fricatives. However, they didn't contribute to the classification results. Perhaps other extracted features would contain additional information and would improve the classification.

4. Robustness to noise - The algorithm of classification was tested only on clean signals, and suppression of noise was not taken into account.

5. Integration with other musical information retrieval (MIR) tasks - classification of audio can be used as a preliminary stage for other MIR applications, such as query by hamming.

6. Apply diffusion maps for other applications of audio and speech classification - Perhaps the framework of diffusion maps and the assumption on which it is based, are more compatible to other applications, and would lead to better classification results.

Bibliography

[1] S. A. Abdallah and M. D. Plumbley. Unsupervised analysis of polyphonic music by sparse coding. *IEEE Transactions on Neural Networks*, 17(1):179–196, January 2006.

[2] S.A. Abdallah and M.D. Plumbley. Sparse coding of music signals. Technical report, Department of Electronic Engineering, King's college London, 2001.

[3] M. Aharon and M. Elad. Sparse and redundant modeling of image content using an image-signature-dictionary. *SIAM Journal on Imaging Sciences*, 1(3):228–247, 2008.

[4] M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, 54(11):4311–4322, November 2006.

[5] A.M.A. Ali, J. Van der Spiegel, and P. Mueller. Acoustic-phonetic features for the automatic classification of fricatives. *The Journal of the Acoustical Society of America*, 109:2217–2235, May 2001.

[6] D. Bauer, A. Plinge, and M. Finke. Selective Phoneme Spotting for Realization of an /s, z, C, t/ Transposer. *Computers Helping People with Special Needs*, 2398:271–306, 2002.

[7] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in Neural Information Processing Systems*, 14:585–591, October 2002.

[8] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

[9] M. Belkin and P. Niyogi. Towards a theoretical foundation for Laplacian-based manifold methods. *The Journal of Computer and System Sciences*, 74(8):1289–1308, 2008.

[10] E. Benetos and C. Kotropoulos. A tensor-based approach for automatic music genre classification. In *Proc. 16th Europian Signal Processing Conferece (Eusipco 2008).*

[11] J. Bergstra, N. Casagrande, D. Erhan, D. Eck, and B. Kégl. Aggregate features and A DA B OOST for music classification. *Machine Learning*, 65(2):473–484, 2006.

[12] J.C. Brown. Calculation of a constant Q spectral transform. *The Journal of the acoustical society of America*, 89(1):425–434, 1991.

[13] J.C. Brown. Musical fundamental frequency tracking using a pattern recognition method. *The Journal of the Acoustical Society of America*, 92(3):1394–1402, 1992.

[14] J.C. Brown and M.S. Puckette. A high resolution fundamental frequency determination based on phase changes of the Fourier transform. *The Journal of the Acoustical Society of America*, 94:662–662, 1993.

[15] E. Candes, L. Demanet, D. Donoho, and L. Ying. Fast discrete curvelet transforms. *Multiscale modeling and simulation*, 5(3):861–899, 2007.

[16] E.J. Candes and D.L. Donoho. Curvelets: A surprisingly effective nonadaptive representation for objects with edges. In A. Cohen, C. Rabut, and L.L. Shumaker, editors, *Curve and Surface Fitting*. Vanderbilt University Press, Nashville TN, 1999.

[17] A.T. Cemgil, H.J. Kappen, and D. Barber. A generative model for music transcription. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(2):679–694, March 2006.

[18] P. Cheeseman and J. Stutz. Bayesian classification (AutoClass): Theory and results. *Advances in knowledge discovery and data mining*, 180:153–180, 1996.

[19] S.S. Chen. *Basis Pursuit*. PhD thesis, Stanford University, 1995.

[20] D.G. Childers, D.P. Skinner, and R.C. Kemerait. The cepstrum: a guide to processing. *Proceedings of the IEEE*, 65(10):1428–1443, 2005.

[21] F.R.K. Chung. *Spectral graph theory.* American Mathematical Society, 1997.

[22] R. Coifman and S. Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21:5–30, June 2006.

[23] R. Coifman and S. Lafon. Geometric harmonics: A novel tool for multiscale out-of-sample extension of empirical functions. *Applied and Computational Harmonic Analysis*, 21:31–52, June 2006.

[24] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[25] G. Costantini, R. Perfetti, and M. Todisco. Event based transcription system for polyphonic piano music. *Signal Processing*, 89(9):1798–1811, 2009.

[26] Giovanni Costantini, Massimiliano Todisco, Renzo Perfetti, Roberto Basili, and Daniele Casali. Svm based transcription system with short-term memory oriented to polyphonic piano music. In *Proc. MELECON 2010 15th IEEE Mediterranean Electrotechnical Conference*, pages 196–201.

[27] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 2002.

[28] R.B. Dannenberg and C. Raphael. Music score alignment and computer accompaniment. *Communications of the ACM*, 49(8):38–43, 2006.

[29] A. de Cheveigné and H. Kawahara. YIN, a fundamental frequency estimator for speech and music. *The Journal of the Acoustical Society of America*, 111:1917, 2002.

[30] P. De La Cuadra, A. Master, and C. Sapp. Efficient pitch detection techniques for interactive music. In *Proc. the 2001 International Computer Music Conference.* Citeseer, 2001.

[31] O. Dekel, J. Keshet, and Y. Singer. An online algorithm for hierarchical phoneme classification. *Machine Learning for Multimodal Interaction*, 3361:146–158, 2005.

[32] S. Dixon. On the computer recognition of solo piano music. In *Proc. Australasian Computer Music Conference.* Citeseer, 2000.

[33] M.N. Do and M. Vetterli. Contourlets: a new directional multiresolution image representation. In *Proc. Conference Record of the Thirty-Sixth Asilomar Conference on Signals, Systems and Computers, 2002*, volume 1, pages 497–501.

[34] M.N. Do and M. Vetterli. The contourlet transform: an efficient directional multiresolution image representation. *IEEE Transactions on Image Processing*, 14(12):2091 –2106, December 2005.

[35] D.L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.

[36] D.L. Donoho and C.E. Grimes. Hessian eigenmaps : locally linear embedding techniques for high-dimensional data. *Proc. the National Academy of Arts and Sciences of the United States of America*, 100:5591–5596, 2003.

[37] M. Elad. *Sparse and redundant representations - From theory to applications in signal and image processing*. Springer, 2010.

[38] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, 15(12):3736–3745, 2006.

[39] K. Engan, S.O. Aase, and J.H. Husøy. Multi-frame compression: Theory and design. *Signal Processing*, 80(10):2121–2140, 2000.

[40] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the Nyström method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):214–225, February 2004.

[41] A. Frid and Y. Lavner. Acoustic-Phonetic Analysis of Fricatives for classification using SVM Based Algorithm. In *Proc. 26th IEEE Convention of Electrical and Electronics Engineers in Israel (IEEEI'10) , Eilat, Israel, November 2010.*

[42] H. Fu, R. Rodman, D. McAllister, D. Bitzer, and B. Xu. Classification of voiceless fricatives through spectral moments. 1999.

[43] M. Gordon, P. Barthmaier, and K. Sands. A cross-linguistic acoustic study of voiceless fricatives. *The Journal of the International Phonetic Association*, 32(2):141–174, 2002.

[44] IF Gorodnitsky and BD Rao. Sparse signal reconstruction from limited data using FOCUSS: a re-weighted norm minimization algorithm. *IEEE Transactions on Signal Process*, 45(3):600–616, 1997.

[45] L.J. Hadjileontiadis. A texture-based classification of crackles and squawks using lacunarity. *IEEE transactions on biomedical engineering*, 56(3):718–732, 2009.

[46] J. Ham, D.D. Lee, S. Mika, and B. Schölkopf. A kernel view of the dimensionality reduction of manifolds. In *Proc. 21th international conference on Machine learning*. ACM, 2004.

[47] W.M. Hartmann. Pitch, periodicity, and auditory organization. *The Journal of the Acoustical Society of America*, 100:3491, 1996.

[48] A. Holzapfel and Y. Stylianou. Musical genre classification using nonnegative matrix factorization-based features. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2):424–434, 2008.

[49] Y. Keller, R.R. Coifman, S. Lafon, and S.W. Zucker. Audio-visual group recognition using diffusion maps. *IEEE Transactions on Signal Processing*, 58(1):403–413, 2009.

[50] A. Klapuri. Automatic transcription of music. Master's thesis, Tempere University of Technology, Tempere, Finland, 1998.

[51] A. Klapuri. Automatic music transcription as we know it today. *The Journal of New Music Research*, 33(3):269–282, 2004.

[52] A. Klapuri. Multipitch analysis of polyphonic music and speech signals using an auditory model. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2):255–266, 2008.

[53] S. Lafon, Y. Keller, and R. R. Coifman. Data fusion and multicue data matching by diffusion maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1784–1797, November 2006.

[54] B. Lévy. Laplace-Beltrami Eigenfunctions Towards an Algorithm That "Understands" Geometry. In *Proc. IEEE International Conference on Shape Modeling and Applications, 2006 (SMI 2006)*, pages 13–20.

[55] T. Li, M. Ogihara, and Q. Li. A comparative study on content-based music genre classification. In *Proc. 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 282–289. ACM, 2003.

[56] T. Lidy and A. Rauber. Evaluation of feature extractors and psycho-acoustic transformations for music genre classification. In *Proc. ISMIR*, pages 34–41. Citeseer, 2005.

[57] S.G. Mallat and Zhifeng Z. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397 –3415, December 1993.

[58] M. Marolt. A connectionist approach to automatic transcription of polyphonic piano music. *IEEE Transactions on Multimedia*, 6(3):439 – 449, June 2004.

[59] K.D. Martin. A blackboard system for automatic transcription of simple polyphonic music. Technical report, Massachusetts Institute of Technology Media Laboratory Perceptual Computing Section, 1996.

[60] C. McKay and I. Fujinaga. Automatic genre classification using large high-level musical feature sets. In *Proc. International Conference on Music Information Retrieval*, volume 525-530. Citeseer, 2004.

[61] G.J. McLachlan and J. Wiley. *Discriminant analysis and statistical pattern recognition*. Wiley Online Library, 1992.

[62] R. Meddis and M.J. Hewitt. Virtual pitch and phase sensitivity of a computer model of the auditory periphery. I: Pitch identification. *The Journal of the Acoustical Society of America*, 89(6):2866–2882, 1991.

[63] R. Meddis and L. O'Mard. A unitary model of pitch perception. *The Journal of the Acoustical Society of America*, 102:1811, 1997.

[64] J.A. Moorer. *On the segmentation and analysis of continuous musical sound by digital computer*. PhD thesis, 1975.

[65] B. Nadler, S. Lafon, R. Coifman, and I. Kevrekidis. Diffusion Maps - a Probabilistic Interpretation for Spectral Embedding and Clustering Algorithms. *Principal Manifolds for Data Visualization and Dimension Reduction*, 58:238–260, 2007.

[66] A.M. Noll. Cepstrum pitch detection. *The Journal of the Acoustical Society of America*, 41(2):293–309, 1967.

[67] A.M. Noll. Pitch determination of human speech by the harmonic product spectrum, the harmonic sum spectrum, and a maximum likelihood estimate. In *Proc. the Symposium on Computer Processing in Communication*, pages 779–798, 1969.

[68] B.A. Olshausen and D.J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, June 1996.

[69] H.F. Olson. *Music, physics and engineering*. Dover publications, 1967.

[70] Y. Panagakis, C. Kotropoulos, and G.R. Arce. Music genre classification via sparse representations of auditory temporal modulations. In *Proc. 17th European Signal Processing Conference (Eusipco 2009)*.

[71] M.D. Plumbley, SA Abdallah, JP Bello, ME Davies, G. Monti, and M.B. Sandler. Automatic music transcription and audio source separation. *Cybernetics and Systems*, 33(6):603–627, 2002.

[72] M.D. Plumbley, S.A. Abdallah, T. Blumensath, and M.E. Davies. Sparse representations of polyphonic music. *Signal Processing*, 86(3):417–431, March 2006.

[73] G.E. Poliner and D.P.W. Ellis. A discriminative model for polyphonic piano transcription. *EURASIP Journal on Applied Signal Processing*, 2007(1):154, January 2007.

[74] R. Rowe. *Machine musicianship.* The MIT Press, 2004.

[75] S.T Roweis and L.K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, December 2000.

[76] R. Rubinstein, M. Zibulevsky, and M. Elad. Double sparsity: Learning sparse dictionaries for sparse signal approximation. *IEEE Transactions on Signal Processing*, 58(3):1553–1564, 2010.

[77] M.P. Ryynanen and A. Klapuri. Polyphonic music transcription using note event modeling. In *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2005*, pages 319–322.

[78] S. Saito, H. Kameoka, K. Takahashi, T. Nishimoto, and S. Sagayama. Specmurt analysis of polyphonic music signals. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(3):639–650, 2008.

[79] B. Scharf. Critical bands. *Foundations of modern auditory theory*, 1:159–202, 1970.

[80] B. Schölkopf, A. Smola, and K.R. Müller. Kernel principal component analysis. *Artificial Neural Networks ICANN 1997*, 1327:583–588.

[81] J. Shawe-Taylor and N. Cristianini. *Kernel methods for pattern analysis.* Cambridge Univ Pr, 2004.

[82] J. Shlens. A tutorial on principal component analysis. Technical report, Center for Neural Science, New York University, New York City, April 2009.

[83] P. Smaragdis and J.C. Brown. Non-negative matrix factorization for polyphonic music transcription. In *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 177–180. Citeseer, 2003.

[84] A.D. Sterian. *Model-based segmentation of time-frequency images for musical transcription.* PhD thesis, The University of Michigan, 1999.

[85] S.S. Stevens. *Psychophysics.* Wiley New York, 1975.

[86] J.B. Tenenbaum, V. de Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, December 2000.

[87] J.A. Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Transactions on Information Theory*, 50(10):2231–2242, 2004.

[88] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, July 2002.

[89] E. Vincent and X. Rodet. Music transcription with ISA and HMM. *Independent Component Analysis and Blind Signal Separation*, pages 1197–1204, 2004.

[90] T. Virtanen. Sound source separation using sparse coding with temporal continuity objective. In *Proc. ICMC*, volume 3, pages 231–234. Citeseer, 2003.

[91] Changsheng Xu, N. C. Maddage, Xi Shao, Fang Cao, and Qi Tian. Musical genre classification using support vector machines. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '03)*, volume 5, pages 429–432, April 2003.

[92] R.W. Young. Inharmonicity of plain wire piano strings. *The Journal of the Acoustical Society of America*, 24(4):446–458, July 1952.

[93] H. Zha and Z. Zhang. Isometric embedding and continuum ISOMAP. In *Proc. 20th International Conference on Machine Learning (ICML-2003), Washington DC, 2003*, volume 20, pages 864–871.